INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

Université d'Ottawa · University of Ottawa

A Study of Speech Compression Algorithms for Voice Over IP

By

Christian Montminy, B. A. Sc.

A thesis submitted to the

Faculty of Graduate and Postdoctoral Studies

in partial fulfillment of the requirements of the degree of

Master of Applied Sciences

in Electrical and Computer Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering

School of Information Technology and Engineering

Faculty of Engineering

University of Ottawa,

May 2000

Abstract

Voice over IP (VoIP), the transmission of voice over an IP network, is gaining much momentum in the industry. Its benefits over conventional telephony are numerous, but the impairments present on an IP network make implementing such a system quite a challenge.

In this thesis, Voice over IP (VoIP) will be studied with a particular emphasis on the speech compression algorithms used in such a system. The desired characteristics of a speech compression algorithm for VoIP will be analyzed. These characteristics will then be used to judge how appropriate are some existing standards of speech compression algorithms for VoIP. A promising speech compression algorithm resulting from this analysis, ITU-T G.729A, will be used to analyze its ability to recover from packet loss since this is one of the main impairments of an IP network. Finally, a novel method to improve the performance of this algorithm in presence of packet loss will be described with results.

Table of Contents

Table of Figures

List Of Acronyms

AbS	Analysis-by-Synthesis
ACR	Absolute Category Rating
ADPCM	Adaptive Differential Pulse Code Modulation
CCR	Comparison Category Rating
CELP	Code-Excited Linear Prediction
CMOS	Comparison Mean Opinion Score
CNG	Comfort Noise Generator
CPU	Central Processing Unit
CS-ACELP	Conjugate-Structure Algebraic CELP
DPCM	Differential Pulse Code Modulation
DSP	Digital Signal Processing
DTMF	Dual Tone Multi-Frequency
FDDI	Fiber Distributed Data Interface
FEC	Forward Error Correction
IP	Internet Protocol
LAN	Local Area Network
ISP	Internet Service Provider
LMS	Least Mean Square
LP	Linear Prediction
LPC	Linear Prediction Coefficients

LSF	Line Spectrum Frequencies
LSP	Line Spectrum Pairs
MA	Moving Average
MAC	Medium Access Control
MBSD	Modified Bark Spectral Distortion
MIPS	Million of Instructions per Second
MOS	Mean Opinion Score
MS	Mean Square
MSE	Mean Square Error
NIC	Network Interface Card
PC	Personal Computer
PCM	Pulse Code Modulation
POTS	Plain Old Telephone Service
PSQM	Perceptual Speech Quality Measure
PSTN	Public Switched Telephone Network
QoS	Quality of Service
RF	Radio Frequencies
RSVP	Resource Reservation Protocol
RTP	Real Time Protocol
SNR	Signal to Noise Ratio
TCP	Transmission Control Protocol
UDP	User Datagram Protocol

VAD	Voice Activity Detector
VoIP	Voice over IP
WAN	Wide Area Network

Acknowledgements

I would like to express my deepest gratitude to my thesis supervisor, Dr. Tyseer Aboulnasr. Her constant encouragement, stimulating discussions and continuous support made doing this thesis an enriching experience.

I would also like to thank Dr. Henry Wong from Sedona Networks for his help on this project. He gave me much needed feedback on my work and permitted me to get some hands-on experience in the field of Voice over IP.

A special thanks goes out to Dr. Luis Orozco-Barbosa, Dr. Abbas Yongacoglu, Dr. Seyed Bahram Zahir Azami and Dr. Ahmed Ali. My collaboration with them on a CITO project was very beneficial to my work on this thesis.

I would like to give special thanks to my fiancée Véronique. Her loving support and her confidence in me gave me the needed motivation to do this thesis. I would also like to thank my parents, Sylvie and Germain. Without their nurturing and loving support I would never have made it this far.

Finally, I would like to thank the Natural Sciences and Engineering Research Council of Canada (NSERC) and the University of Ottawa for their financial support

1 Introduction

1.1 Motivation

This thesis deals quite extensively with Voice over IP (VoIP). Voice over IP is simply the transmission of real-time voice over an Internet Protocol (IP) network. The IP network used to transmit the voice can either be a private network or the public Internet. It is an alternate way to make a telephone call as compared to the Public Switched Telephone Network (PSTN). While Voice over IP has its origins as a cheap way to make a long distance call over the Internet, it is much more than that today. The first papers and experiments on the subject occurred in the early 1970's, but it took until the 1990's before the concept was taken more seriously [64]. Industry is now slowly recognizing the potential of Voice over IP and many VoIP applications have been developed in recent years. Even though VoIP is being recognized in industry, not much detailed research has been done on the subject. This is one of the motivations for doing this thesis.

One question that is worth asking is: why would we transmit voice over an IP network if we are already able to transmit voice on the conventional telephone network? The answer to this question is simple. Transmitting voice over an IP network has many benefits over transmitting voice on the PSTN. These benefits are what have motivated the industry to develop VoIP systems and can be summarized as:

- free or cheaper calls as compared to the PSTN
- it permits the integration of the voice and data networks while recognizing the reality that data is the dominant traffic
- it permits the use of state of the art speech compression algorithms and silence suppression to reduce the necessary bandwidth for a single call
- it permits the easier addition of additional features to telephony and a better user interface
- it permits corporations to make better use of their investments in data networks

These benefits are so great that they warrant using VoIP as an alternative or even possibly a replacement to the conventional telephone network.

Most industrial applications up to this day have concentrated on the use of VoIP on private networks since the impairments on such networks can be well controlled. The use of VoIP over the Internet, although attractive, has not been developed much by the industry due to the severe and unpredictable impairments present on the Internet. Most of the applications for VoIP over the Internet that are currently available are of limited quality [71]. The main impairments that exist on an IP network that are disturbing to VoIP applications are delay, jitter and packet loss. Since IP networks are inherently only *best effort* networks, the transmitted data is not guaranteed to arrive at its destination and the time it takes to get there is variable. This is not acceptable for VoIP and any reliable application has to compensate for these impairments. Examining the techniques to deal

with these impairments and trying to improve on them is also a main motivation of this thesis. Such research must be done if we want some day to have good quality VoIP over the Internet.

Speech compression is an essential part of any Voice over IP system. It determines how we encode the speech for transmission and has an enormous effect on the speech quality observed. We will see in this thesis that there are some specific desired characteristics for a speech compression algorithm to be used in a VoIP application. Packet loss occurs quite often in an IP network. As such, the speech compression algorithm used has to be able do provide good quality *during a period of packet loss* and to be able to recover well *following a period of packet loss* (i.e. provide good quality not just during the packet loss but in the period following it). One of the main motivations of this thesis was to examine the speech compression algorithms and determine which types of speech compression algorithms are better suited for VoIP. Such an analysis requires examining the performance of these speech compression algorithms during and after periods of packet loss.

1.2 Objectives

The objectives of this thesis are the following:

- Provide a complete literature review of Voice over IP and related subjects (speech compression and IP networks). This is necessary to properly understand the challenges that exist in designing a VoIP system.
- Provide a comparative study on speech compression algorithms for VoIP. It is desired to identify clearly what are the most important characteristics for a speech compression algorithm to be used in VoIP. Then we can use these characteristics to compare how appropriate the different standards for speech compression are for VoIP.
- Investigate the performance in periods of packet loss for a promising speech compression algorithm. For a speech compression algorithm for which our comparative study identifies as appropriate for VoIP, analyze how this algorithm performs during and after periods of packet loss. Since the performance of a speech compression algorithm in periods of packet loss is very important in VoIP and since it is not well understood, this represents an important contribution for this thesis.
- Propose means to improve the performance in presence of packet loss for this promising speech compression algorithm.

1.3 Thesis Layout

The five remaining chapters will be divided as follows:

Chapter 2: Voice over IP

In this chapter, we will define more specifically what we mean by a VoIP system. This chapter will start with a review of IP networks, the networking protocols that are used in VoIP and a discussion of the lack of Quality of Service (QoS) guarantee in IP networks. We then describe a typical VoIP system and the components involved in such a system. The benefits of performing Voice over IP will be discussed in details. Finally, the cause and effects of the different impairments of the IP network on the VoIP system will be discussed.

Chapter 3: Speech Compression

In this chapter, the different methods of performing speech compression will be presented. Since speech compression is a central part of a VoIP system and this thesis, it is important that it is properly understood. This chapter introduces the three different families of speech compression algorithms: waveform-based, perceptual-based and model-based speech compression. For each of these speech compression families, the basic principle behind the compression and some existing international standards belonging to these families will be discussed.

Chapter 4: Speech Compression for Voice over IP

This chapter builds on the foundation built in chapter 3 to discuss speech compression algorithms *in the context* of VoIP. The chapter begins by describing the desired characteristics of a speech compression algorithm for VoIP. It then uses these

characteristics to evaluate the appropriateness of the different speech compression algorithms for VoIP. This comparative study permits us to determine a speech compression algorithm that is deemed appropriate for VoIP. The remaining of this chapter discusses various topics of interest in speech compression for VoIP including the recovery from packet loss, the packetization of speech frames into VoIP packets and the evaluation of speech quality.

Chapter 5: Improving the Performance of G.729A in the Presence of Packet Loss

This chapter introduces a novel method to improve the performance in periods of packet loss of G.729A. ITU-T G.729A is a model-based speech compression algorithm that the comparative study in chapter 4 has identified as a good candidate for VoIP. In this chapter, the performance of this algorithm in periods of packet loss is first examined. This is followed by the introduction of a novel method, referred to as recovery-by-re-initialization, to improve the performance of the algorithm in periods of packet loss. Results in this section show that the proposed method does indeed improve the performance of the algorithm in periods of packet loss with minimal impact on complexity and loss-free performance.

Chapter 6: Discussion and Conclusion

This chapter concludes the thesis by discussing the main results and contributions of this thesis as well as possible future work that may enhance the results presented here.

2 Voice over IP

2.1 Overview

Voice over IP is essentially the transmission of voice over an IP network. Voice could also be transmitted over other data networks such as ATM or Frame Relay instead of IP. However, since IP is the most widely used network today, it is the preferred medium for transmission of voice over data networks [67]. Because of this fact, only Voice over IP will be considered in this thesis.

This chapter deals with Voice over IP at a high abstraction level. Further details on speech compression and the interaction of the speech compression algorithm in the Voice over IP system will be given in chapters 3 and 4 respectively. The chapter starts with a definition of what an IP network is and a description of the data networking protocols used for Voice over IP. The lack of Quality of Service on IP networks will also be addressed. This will be followed by a formal definition of Voice over IP and a thorough description of a Voice over IP system. Then, we will examine the many benefits of using a Voice over IP system as compared to the standard telephony network for both the telecommunication companies and their customers. Finally, a description of the impairments that are common to all Voice over IP systems, their origin and how these must be dealt with in order to ensure good quality will be given. The last section will

summarize what has been discussed in this chapter in order to highlight the facts that are the most important for this thesis.

2.2 IP Networks

2.2.1 Overview

Before discussing in detail what Voice over IP (VoIP) is, it is important to have a good understanding of what an IP network is since this is the medium used to transmit voice in a VoIP system.

When defining the IP network, an emphasis will be put on the aspects that are more important to Voice over IP. A typical IP network will be presented and explained. This will be followed by the introduction of data network protocols that are relevant to Voice over IP. Since protocols are the foundation of all data networks, a good understanding of those used for Voice over IP is important. Those discussed in this section will include IP, UDP, TCP, RTP and the ITU standard H.323. Finally, this section will conclude by discussing the basic principles of Quality of Service (QoS). Real-time services such as voice require a minimum quality of service to operate successfully; a quality of service that, as we will see, is not guaranteed by an IP network. This section will also introduce the RSVP protocol as a means to try to add Quality of Service to the IP networks.

2.2.2 Definition

We can simply define an IP network as being any data network that recognizes the IP protocol (details of the IP protocol will be given in section 2.2.3). Since the IP protocol is the most widely used data networking protocol, thanks mainly to the popularity of the Internet, insisting that the data network recognizes IP is not a severe limitation. In order to properly define an IP network, we must first define what a data network is. A data network is defined as a collection of terminals connected together for the purpose of exchanging information. The terminals involved are most commonly computers, but they can essentially be any device that is capable of communicating with the other terminals. The networks are traditionally classified as either being a local-area network (LAN) or a wide-area network (WAN). A LAN usually covers a small geographical region while a WAN covers a larger geographical region. Since the distinction between a LAN and a WAN has become somewhat blurred in recent years [69], no such distinction will be given in this thesis. Data networks are packet-switched networks meaning that the information exchanged between the different terminals is done through a small entity called the packet. Whenever a message is required to be exchanged between two or more terminals, the message must first be broken into packets before it can be transmitted.

A simple example of an IP network is given in Figure 2.1. As can be seen, the typical network is made up of two types of entities: terminals and routers. Terminals are simply the elements in a network that communicate with each other. Routers on the other hand are intermediate entities that ensure that the communication between all members of the

network is possible. Other possible entities are bridges and gateways, but these serve a similar purpose to the router. For simplicity, all of these entities will be referred as routers in this discussion. The lines that connect the entities in Figure 2.1 represent wires which can either be made of fiber optics, coaxial cable or twisted copper pair. Note as well that this applies equally to wireless networks connected through radio frequencies (RF).

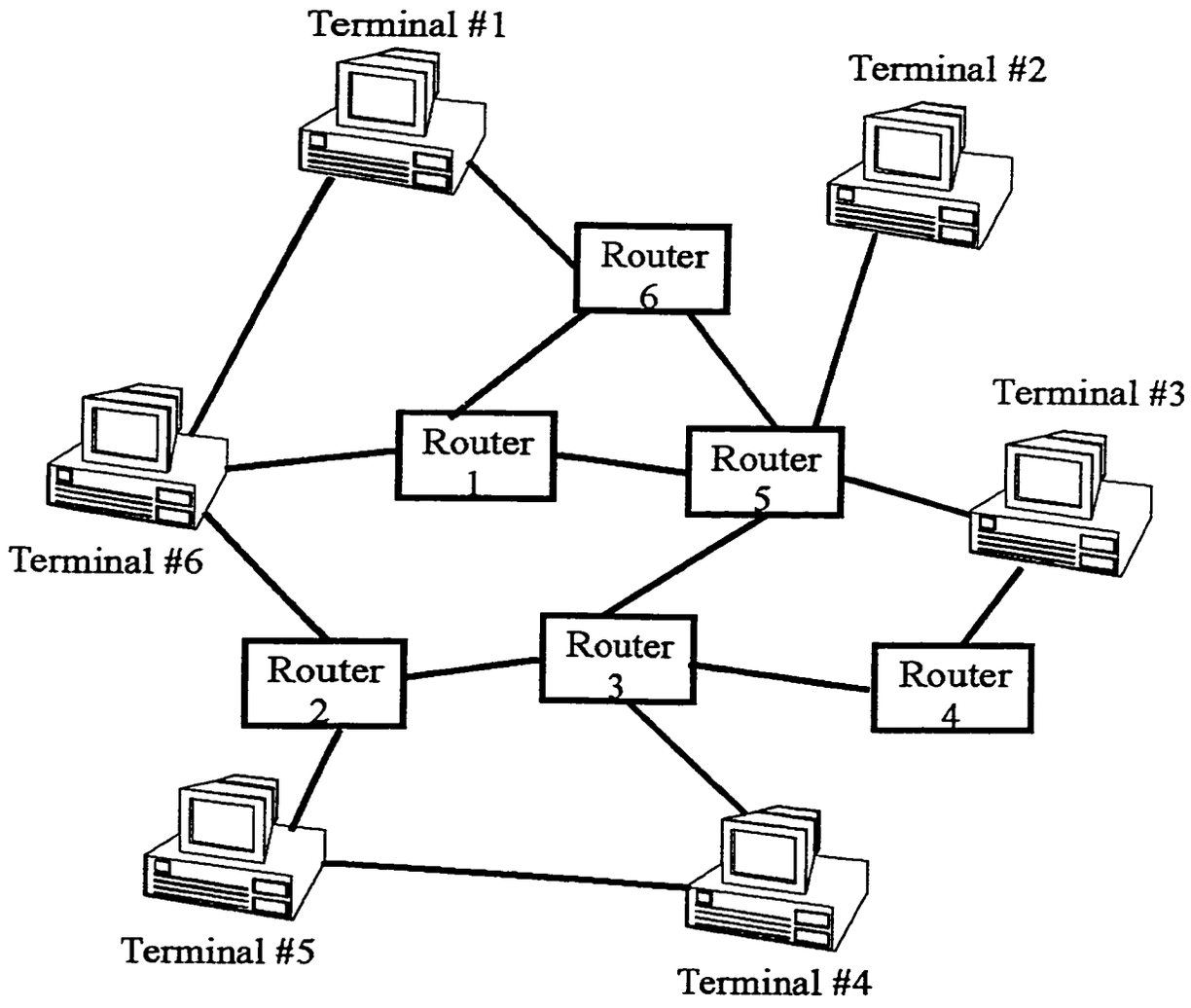

Figure 2.1: Example of an IP Network

The best way to explain the purpose of routers is through an example. Suppose terminal #5 has a message that it wants to exchange with terminal #3. Since terminal #3 is not directly connected to terminal #5, this cannot be done without the intervention of an external entity. In our example, terminal #5 probably doesn't know the location of terminal #3 in the network. In order to send the message, it is first decomposed into a series of packets indicating in each packet that the intended recipient is terminal #3. Then each packet is transmitted on the network over to router #2 since it is adjacent to terminal #5. The function of the router is to maintain knowledge of the network in order to determine the direction in which it should send the packets it receives. This is usually done by having the routers exchange information regularly and keep a table of the next link to use for each terminal in the network. Note that they are usually many possible routes that can be used and the routers must chose an optimum one to maximize the efficiency of the network and to ensure rapid delivery of the packets. In our example, assume that router #2 has already established that sending packets destined for terminal #3 to router #3 is the best choice. All packets that router #2 receives are then passed along to router #3. Similarly, router #3 passes the packets to router #4 that finally passes the packet to terminal #3. Since the network topology (connections + entities in the network) and conditions (traffic) are apt to change, requiring only the routers to keep track of the changes simplifies communication for the terminals. Another function of routers is to permit the interaction of several differing networks.

2.2.3 Protocols for Voice over IP

Protocols are defined as a set of conventions agreed upon for different entities to communicate with each other over a network [69]. Since they essentially define how communication takes place, good understanding of the protocols is essential to properly understand IP networks. In order to generate protocols that support many different types of networks and that are not too complex, a layered approach was used to define the protocols. Instead of creating one single complex protocol that all entities in a network must respect, many simple protocols were created. Many of these protocols are layered on top of each other. This means for example that if we have protocol #2 layered over protocol #1, protocol #2 assumes that protocol #1 is already being respected. Because of this, protocol #2 can deal with the problem at a higher level of abstraction since the lower level details are already covered by protocol #1. Another advantage of the layered approach is that many different networks can be treated with the same high level protocols. These different networks simply need different lower level protocols but they can share the same higher level protocol, permitting them to communicate efficiently with each other.

Given the variety of protocols for IP networks, we will limit the discussion to protocols commonly used in VoIP systems. The layering of these protocols for a VoIP system, commonly called a protocol stack for VoIP, is illustrated in Figure 2.2 [72][69][63].

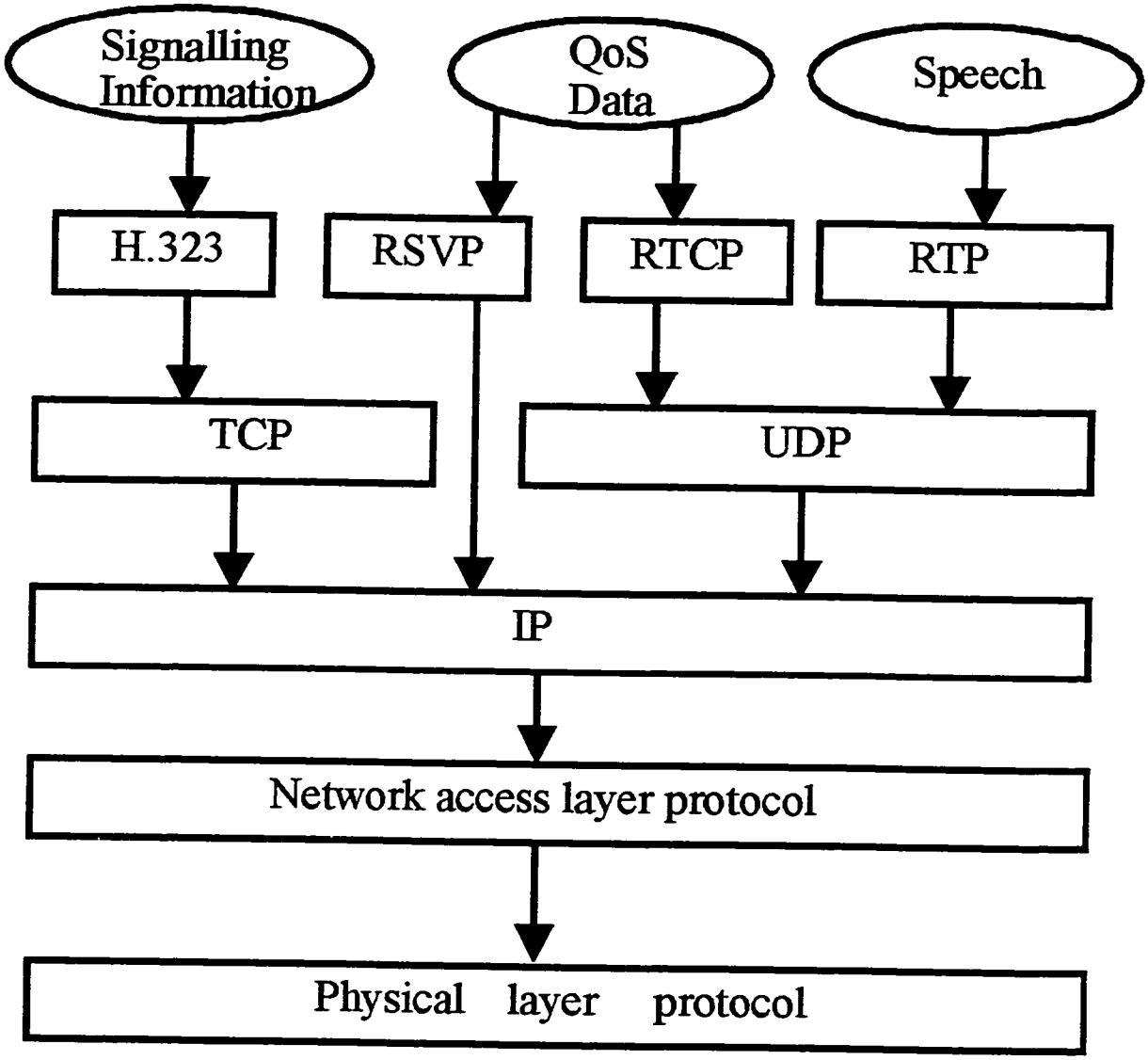

Figure 2.2: Typical Protocol Stack for Voice over IP

The first and most basic protocol on the protocol stack is the ***physical layer protocol.*** This protocol is responsible of interacting with the physical medium. The protocol defines what electrical signals must be used to represent each bit on the wire and at what physical rate transmission should be accomplished. Since such a protocol differs with

every different physical medium (fiber optics, twisted copper pairs, etc.), it can conveniently be changed without affecting any of the other higher level protocols.

The next protocol is the ***network access layer protocol***, which is sometimes called the Medium Access Control (MAC). It is necessary to ensure proper exchange of information within one homogeneous network. This protocol varies with each network technology. Examples of these are Ethernet, Token Ring, FDDI and X.25. This protocol requires adding additional information to every packet that is sent on the network in the form of a header. Information included in this header usually includes the destination of the message (the device on the network that this message is destined to) and requests to use network facilities such as priority [69].

The next higher level protocol is the ***Internet Protocol*** (IP). This level of protocol is required to provide internetworking i.e. the communication between different networks. Since there are many different types of networks, this cannot be done at the network access layer. Instead, dealing with internetworking at a higher level permits us to treat different networks in the same manner, since the details of the network are hidden in the preceding layer. The routers are essentially the entities that deal with internetworking in a network. In order to provide the routers enough information to direct the packets properly to their destination, information must be added to the user data in the form of an IP header. The content of the IP header has a minimum size of 20 bytes. A short description of the different fields in an IP header is given in Table 2.1 [69]. The

description given in this table is for version 4 of the protocol. Version 6 of this protocol already exists and has headers of fixed size of 40 bytes. The main differences between version 4 and 6 are IP addresses of 128 bits instead of 32 bits and extended capabilities for packet prioritization.

Table 2.1: Content of IP header (version 4)

Field Name	Size (bits)	Description
Version	4	version number of the protocol
Internet header length	4	length of header in 32 bit words
Type of service	8	used for priorities
Total length	16	length of IP packet in bytes (includes data)
Identifier	16	unique identifier of IP packet
Flags	3	options related to segmentation
Fragment offset	13	indicates offset of fragment in 64 bit word
Time to live	8	decreased at each router hop, packet is deleted when this count becomes zero
Protocol	8	indicates the next higher level protocol
Header checksum	16	permits error detection in the header
Source address	32	address of source of message
Destination address	32	address of destination of message
Options	variable	encodes options requested by end user
Padding	variable	pad header for size to be a multiple of 32 bit

The next higher level in the protocol stack is the ***transport layer***. This layer provides end-to-end data service making sure that the data is received by the proper application in the destination computer. Two different versions of such a protocol are used for Voice over IP: User Datagram Protocol (UDP) and Transmission Control Protocol (TCP).

UDP is essentially a very simple transport layer that does not guarantee reliable service. Because of this simplicity, the header for the protocol needs only 8 bytes. Its parameters are all of size 16 bit and include the following: source port, destination port, header length and checksum. The port number identifies the application at the source and destination that is involved in the communication. Since several communications can occur simultaneously on the same computer, it is important to specify which application is the intended recipient of each packet.

As opposed to UDP, TCP is considerably more complex and ensures that all the data is properly received (something that UDP does not do). This is achieved by adding a sequence number to every packet. Whenever the receiver detects that it has not received a certain packet (by looking at the sequence number), it requests retransmission of the packet. This mechanism is repeated until all packets are properly received. As well, the sequence number used in TCP permit the destination to properly order the received packets even when they arrive in the wrong order. The cost of such a scheme is the added delay required to request retransmission and a TCP header that has a minimum size of 20 bytes.

Four different higher level protocols are used in Voice over IP: ***H.323, RSVP, RTCP and RTP***. Discussion of RSVP will be postponed to section 2.2.4 after the principles of Quality of Service have been introduced. Among the remaining three, only H.323 is implemented over TCP while the two others work over UDP. To understand the role of these protocols, we must examine the user data that they are required to transmit. The Real Time Protocol (RTP) is used to transmit speech over UDP. As will be explained later in this chapter, speech does not tolerate high delay. This is the reason for selecting UDP over TCP. RTP is used as an added layer instead of using UDP directly because of the real-time nature of speech transmission. RTP adds a time stamp and a sequence number to the speech data in order to permit the proper ordering of the speech data before playback. As well, RTP permits multiple users to participate in the same Voice over IP call. In order to provide such services, RTP needs to add a header of a minimum size of 12 bytes to the speech data. The elements of the header are indicated in Table 2.2 [65].

A protocol that is associated with RTP is the Real Time Control Protocol (RTCP). It is essentially a control protocol for RTP that [65]:

- permits receiving feedback from all participants on the quality of the data received such as delay observed for the data and number of packets that were lost
- provides a canonical name, a transport level identifier for each source
- permits each participant to be aware of all of the participants in the conversation since control packets of all sources are received by all participants

- optionally can be used to transfer information about each user such as their names and telephone numbers.

Table 2.2: Content of RTP header

Field Name	Size (bits)	Description
Version	2	Version number of the protocol
Padding	1	Set if padding is used
Extension	1	Set if contains a header extension
CSRC count	4	Number of contributing sources (for multi-party calls)
Marker	1	User definable meaning
Payload Type	7	Identifies format of data
Sequence Number	16	Identifies with an incrementing number each RTP packet
Timestamp	32	Time the data was produced
SSRC	32	Unique identifier for the source of the data
CSRC list	variable	List of identifiers of all contributing sources (0 to 15, each requires 32 bits)

A final protocol implemented over the transport protocol is the H.323 protocol. This is an ITU standard that enables multimedia calls to be established between two or more parties on a packet network [72]. The main contribution of H.323 is to permit the proper signaling and resources to establish and terminate a call. It is also responsible for finding

the proper IP address of a given destination and for negotiating between all end users a proper format of transmission that is supported by all (essentially choosing a compression standard that is recognized by all). Given that this signaling information does not have stringent delay constraints and that we want this information to be received properly, H.323 is mostly implemented on top of TCP (small portion of it is done over UDP). The H.323 standard can accommodate any multimedia calls (video), but a voice-only subset of the standard is the one used for Voice over IP. The H.323 standard is actually a framework that is composed out of many smaller standards such as H.245 and H.235 that implement the individual functions of the standard [58]. H.323 is the most popular and most widely accepted standard to perform signaling for VoIP, but it is not the only existing standard. The Session Initiation Protocol (SIP) [63] performs similar functions as H.323 and is a recommendation from the Internet Engineering Task Force (IETF) but is not as widely used.

Before moving on to the next section, let's examine the structure of a typical Voice over IP speech packet. Since many intermediate protocols are used to send a speech packet, the speech data will be accompanied by many headers. These different headers and their lengths are illustrated in Figure 2.3 where the format of a typical speech packet is given. Note that the size of the speech data depends on the speech compression algorithm used and the packetization method, as will be discussed further in section 4.6, and that the size of the MAC (also called Network Access Layer) depends on the type of network used.

Finally, note that the sizes given for the IP and RTP headers represent their minimum sizes. The minimum size of the header for a VoIP speech packet is 40 bytes (320 bits).

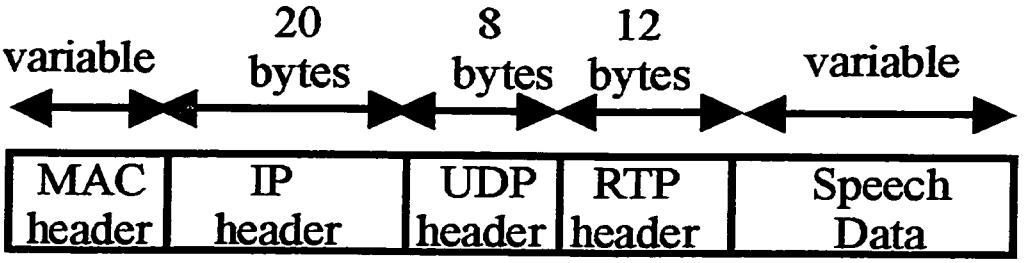

Figure 2.3: Typical Speech Packet for Voice over IP

2.2.4 Quality of Service

Quality of service refers to a guarantee on a certain minimal quality of the service done over the network[39]. Due to its origin as a network for exchanging computer data, the IP network does not guarantee any quality of service (QoS) to real-time applications such as VoIP. IP networks are essentially *a best effort* network where all components involved do their best to deliver the data to their correct recipient, but no guarantee is given that the data will ever get to its recipient or on how much time it will take to get there. When heavy traffic flows through the network, the routers can become congested and their buffers can become full. If this occurs, the routers will have no choice but to ignore the subsequent packets that it receives. These packets are as such lost and will never get to their intended destination. For conventional data, this is not a problem since the delay involved is not important and protocols such as TCP can be used in order to ensure that all data is properly received. But for real-time services such as Voice over IP, this is generally not sufficient. Voice over IP, as will be explained in more details in the

remaining of this chapter, necessitates a small amount of delay that is constant for each packet that is transmitted on the network. Also, it requires that all packets of speech reach their destination properly at the first transmission since TCP cannot be used to recover from lost packets due to delay constraints.

Up to this date, IP networks are not able to guarantee the QoS required by real-time services such as Voice over IP. IP networks cannot guarantee that the speech packets will not be lost due to congestion or that the data will not be corrupted due to noise on the transmission line. As well, the IP network cannot guarantee a fixed amount of delay since the route is chosen dynamically (this might differ from packet to packet) and the time it takes to traverse the network depends on the traffic that exists on the network at that time. The Resource Reservation Protocol (RSVP) was proposed as a remedy to this situation. As can be seen in Figure 2.2, RSVP is a protocol that is implemented directly over IP providing a standardized method for time sensitive applications to reserve bandwidth through requests to the routers on the IP networks [43][41]. It also permits application to specify what are their delay constraints and the routers will try to guarantee that these constraints are met [40]. However, due to the general complexity of the protocol [63] and the fact that it does not standardize a way for routers to implement it, it as yet to be widely deployed.

Since QoS is not guaranteed by the IP network and that efforts to add QoS to the IP network have so far not been sufficient, the only alternative for Voice over IP is to accept

these limitations of the IP network. As will be seen in the remaining of this chapter, a Voice over IP system must implement mechanisms to deal with these shortcomings of the IP network in order to have an acceptable level of service.

2.3 Definition and System Components

2.3.1 Definition

We can define Voice over IP (VoIP) as simply the transmission of voice over an IP network. The IP network that is used for transmission of Voice over IP can be either a private/corporate computer network or the Internet. Voice over IP is then an alternative to the Public Switched Telephone Network (PSTN) for performing a telephone call between two or more parties. The main difference between the two is the medium used to transmit the voice, which is a data network for VoIP and the telephone network for a conventional telephone call.

A typical system for a VoIP service is depicted in Figure 2.4 that follows [37]. All solid lines in this diagram represent a data network while all dotted lines represent a telephone network. It is interesting to note that even though we define VoIP as the transmission of voice over an IP network, part of the access path may use the telephone network. This is currently done for convenience but in the future the telephone network could be bypassed altogether.

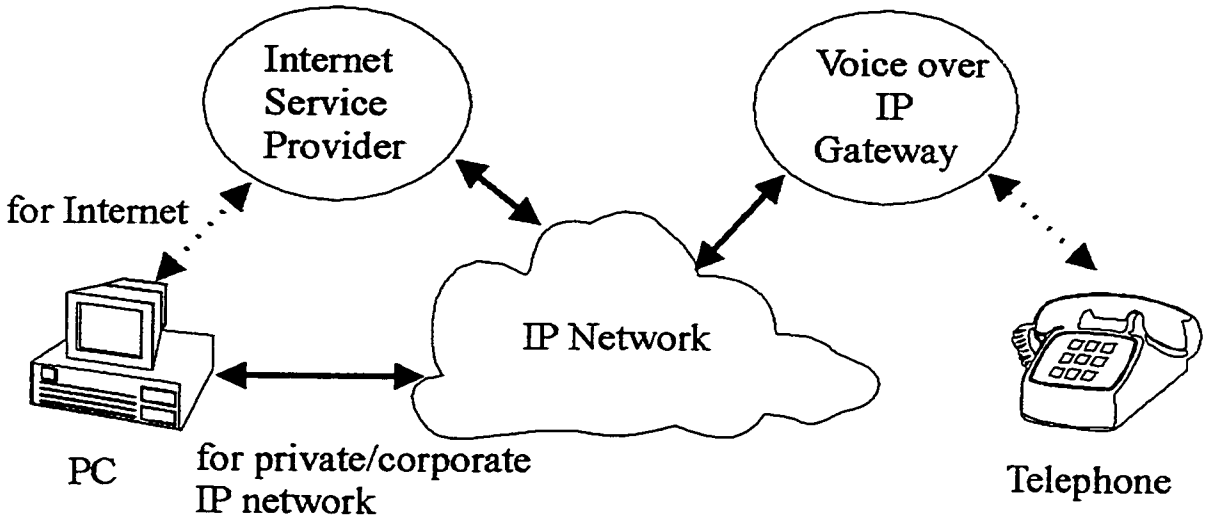

Figure 2.4: Block Diagram of VoIP System

2.3.2 Access Methods

Two different access methods for VoIP are acceptable: either a personal computer (PC) or a regular telephone. Both are acceptable methods of performing VoIP and the method preferred depends on the situation.

If we choose a PC as the access method to perform VoIP, we will require a multimedia PC for each user in the system. What is defined as a multimedia PC in this case is a PC that is equipped with a sound card, speakers and a microphone. Additional requirements for the PC used as an access method for VoIP depend on the type of IP network we are dealing with. If the IP network is a private/corporate network, the PC will require a network interface card (NIC) in order to gain access to this private/corporate network. The network interface card will permit the computer to communicate directly with the IP network for the VoIP system. If we are dealing with the Internet, a modem will be

required since an Internet Service Provider (ISP) is usually needed whereas the PC connects to the ISP through the regular phone line using a modem and the ISP is connected to the Internet. In this case, the access to the IP network is not direct but is done through the ISP. When the access method is a PC, all of the VoIP functionality (speech compression, echo cancellation, etc) is performed by the PC itself. For this reason, the PC used should be quite fast in order to be able to perform all of the VoIP tasks in real time.

If we choose the telephone as our access method to perform VoIP, a VoIP Gateway [42], also called an Internet Telephony Gateway in the literature, is required. The Voice over IP Gateway is needed for the two different types of IP network, the Internet and the private/corporate network. The Voice over IP Gateway is responsible for all of the VoIP functionality (speech compression, echo cancellation, etc) since the telephone, which is the same type of phone used for conventional telephony, is not capable of such functionality. An additional function of the VoIP Gateway is to convert the telephone number entered by the user into the proper IP address of the destination of the call. The requirements for the two different types of access method and the two different types of IP networks are resumed in
Table 2.3.

Any VoIP call can have as source and destination either a PC or a telephone. To route the call to the right destination, either the IP address of the intended destination or of the

appropriate VoIP Gateway when the destination is a telephone must be known in advance or a method to find it must be provided. Methods to convert a telephone number to an IP address or to locate the proper VoIP Gateway for a particular call from a PC to a telephone or from a telephone to a telephone have yet to be standardized [59].

Table 2.3: Requirements for VoIP for each Access Method

Access Method	Requirements	
	Internet	**Private/corporate Network**
telephone	VoIP gateway, telephone	VoIP gateway, telephone
PC	Multimedia PC, Internet access	Networked Multimedia PC

Finally, we note that there exists a third possible access method to perform VoIP. This would be a special-purpose hardware that is built only to perform VoIP. Such a piece of hardware is usually called a VoIP or Internet phone. An example of such an implementation is given in [8]. All of the functionality for the VoIP call is performed by this special-purpose hardware. This third access method is not illustrated in Figure 2.4 since at an abstract level, it is equivalent to using a PC.

2.3.3 VoIP Components

Figure 2.5 shows the different components required to perform VoIP at the encoder and the decoder for a typical VoIP system (some of these components are optional). The tasks illustrated in this figure only deal with the transmission of the speech across the IP

network. In a Voice over IP system, signaling must also be performed in order to establish the telephone calls, as was described for H.323 in section 2.2.3. The tasks required to perform the signaling and also to perform any billing will not be discussed any further in this thesis. Since both sides of a telephone conversation can speak and listen, an encoder and a decoder must be present at both sides. The functionality described in Figure 2.5 is what needs to be performed by either the PC or the VoIP Gateway to support a VoIP call. Note that this decomposition of the tasks for Voice over IP is simply for abstraction purposes; the specified functionality may be implemented using different components.

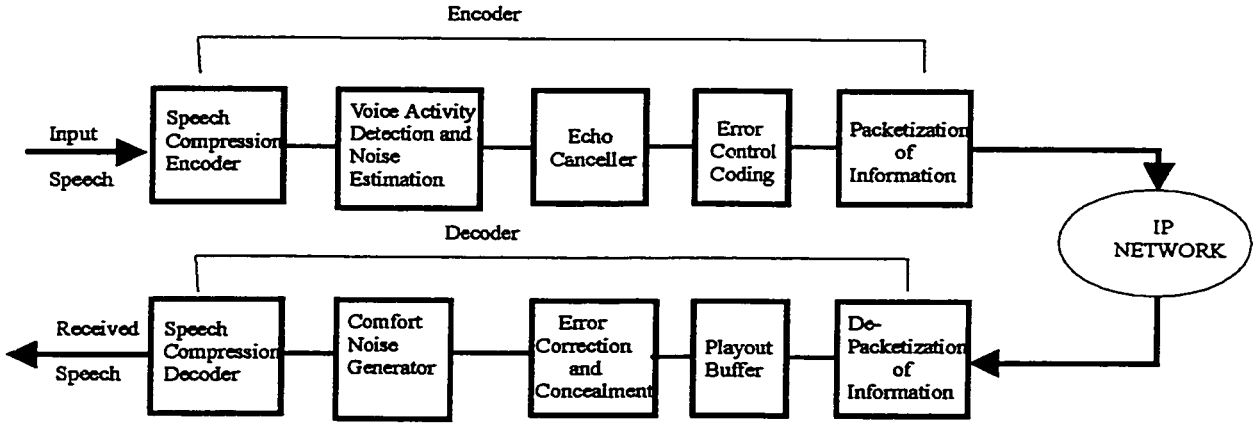

Figure 2.5: Typical VoIP Components

A central component of a VoIP system is the ***speech compression algorithm***. It is responsible for coding the speech for a VoIP call in an efficient manner. Both parties of the VoIP call must be equipped with a speech compression encoder and a speech compression decoder. Note that even though the choice of the specific speech compression algorithm to be used is flexible in a VoIP system, it is essential that both sides use the same algorithm in order to assure compatibility. Since the speech

compression is central to this thesis and to VoIP, it will be presented in more detail in chapter 3.

The ***Voice Activity Detection and Noise Estimation*** block is an optional block that is usually used in VoIP. It permits the system to save bandwidth by transmitting less data during silent periods of the speech. The Voice Activity Detector (VAD) is responsible for monitoring the input and deciding whether the person is speaking or whether we are simply hearing background noise. To prevent the user from assuming that the connection is lost when the other side is not speaking, estimates of the parameters characterizing the background noise are transmitted to the other side for reproduction of the noise. Given that the background noise parameters are more limited and do not need to be sent often, we still achieve considerable saving of bandwidth when the person is not speaking. At the receiver, the Comfort Noise Generator (CNG) takes these noise parameters and generates an approximate replica of the background noise when the other side is not speaking. This artificial background noise is more pleasing to the listener than hearing silence and does not give the user the sense that the connection is lost. More details on silence suppression and comfort noise generation can be found in [3].

The ***Echo canceller*** shown here serves the same purpose as the ones found on the PSTN. It removes the echo caused by the hybrid when the voice travels on the regular PSTN in Figure 2.4. Another type of echo canceller is the one used to remove the acoustic echo if we are in a hands-free environment. For example, this will occur in Figure 2.4 if we are

speaking through a microphone connected to the PC. The microphone would not only pick up our voice, but it would also pick up the voice of the other speaker that is being listened to on a pair of speakers. The result would be that the person speaking will hear an echo of his own voice, which can be very annoying if the delay involved in the transmission media is high. In such a situation, the echo canceller is used to remove this acoustic echo. Even though the echo canceller is a component that is already regularly used in telephony, we will see in the next section that the impairments that are found in a VoIP system make the work of the echo canceller much more complicated than it is for regular telephony. As such, much research is still being done in this area to improve the effectiveness of echo cancellation methods [18][16][22].

As is often used in telecommunications, ***error control coding*** can be used to protect the compressed speech against bit errors. At the decoder, error correction can then be performed if permitted or we will have to cover up the error through error concealment. Given the unique nature of errors in VoIP (whole packets can be lost), the error control coding and error concealment will differ from what is usually encountered in regular telephony. A thorough review of methods to deal with errors in a Voice over IP system will be given in section 4.5.

The ***Playout Buffer*** is a component that is present to remove the jitter from the system. As we will see in section 2.5, jitter is one of the important impairments in VoIP. More details on the playout buffer will be given in that section.

Before being sent over the IP network, the speech data must first be packetized into VoIP packets at the transmitter. De-packetization must then be done at the receiver to recover the speech data. The format of the speech packets was discussed in section 2.2.3. Further discussion on the packetization of the speech data will be deferred to section 4.6 after the principles of speech compression have been introduced.

2.4 Benefits of VoIP

2.4.1 Overview

One question that is worthy of asking is why transmit voice over an IP network when we are already able to transmit voice on the conventional telephone network? The answer to this question will be given in this section where we will discuss the benefits of performing Voice over IP as compared to conventional telephony. It will be shown that these benefits are important for both the telecommunication companies and their customers. These benefits are so great that they warrant using VoIP as an alternative or even in the future as a possible replacement to the conventional telephone network.

2.4.2 Benefits for the Telecommunication Companies

The first benefit for the telecommunication companies is improved utilization of the available resources. This comes in part from the fact that the IP network does not need to dedicate a line for each telephone conversation as is currently necessary for the Plain Old Telephone Service (POTS). Note that this is not exactly true if the access method

requires the use of a conventional telephone line (as discussed in section 2.3.1). The speech of the talker is divided into packets and these only need to be sent when the person is speaking. Due to the fact that a person is talking on average only 40% of the time during a phone conversation, this represents a net saving of 60% of the required bandwidth [67]. This saving of bandwidth is slightly reduced if comfort noise, discussed in section 2.3.3, is used instead of silence, but the speech quality is much better. The second way Voice over IP permits better utilization of the available resources as compared to POTS is that it permits the use of state of the art speech compression to reduce the bandwidth requirements even further. Such speech compression is not possible for the PSTN [43] because of the difficulty of changing an infrastructure that was designed for only pulse code modulation (PCM). For Voice over IP, this is not a problem. Since the network will see the speech as simple data, the choice of speech compression is completely unrestricted. It is then possible to choose a speech compression algorithm that will greatly reduce the bandwidth requirement as compared to PCM. More details on the different speech compression algorithms and their required bit rate will be given in Section 3 on Speech Compression.

The second advantage of using Voice over IP for telecommunication companies is the integration of the data and telephone networks. By using one network for transmitting voice and data instead of one network for voice and one for data, the maintenance costs are reduced since we are dealing with only one network. This reduced cost results in increased profits for the telecommunication companies or lower rates for the consumers.

Finally, the creation of additional features is greatly simplified with Voice over IP as compared to the PSTN. Supporting multi-party conversations and adding additional functions to the telephone is simplified [63] since it is easier done on a data network which is much more flexible with regards to the type of data being transmitted on the network as compared to the PSTN. The integration of other Internet services and the creation of new services can be done with relative ease in VoIP [52]. This is mainly due to the fact that these services can be done out of band as separate IP packets in VoIP while we are limited to in-band services for the PSTN. This can generate additional profits for the telecommunication companies.

2.4.3 Benefits for the Consumer

Currently, the top benefit of Voice over IP for the consumer is access to free long distance calls. Since VoIP permits the consumer to bypass the conventional telephone network to make a long distance call, the call will be free to the consumer. The call is not necessarily free if it requires an Internet connection; this Internet connection usually comes at a price but can be used to perform other functions than just VoIP. If the consumer is a company using its data network to perform VoIP, a major advantage is the reduced cost of telephony. Even though the company will have to purchase either computers (which it likely already has) or a Voice over IP gateway to perform VoIP, it will save the costs of calls made inside the company. Another benefit to the company is the better utilization of the company's investment in its data network [43]. Since many

companies already have large data networks that connect all of their corporate offices, these can now be use to make telephone calls across physically separated sites of the company at relatively no extra cost.

Even though VoIP calls presently cost less than conventional telephone calls, this is not likely to last forever. If the medium for the VoIP call is the Internet, considerable overhead is needed to purchase the appropriate VoIP gateways to permit VoIP. As well, these VoIP calls will use up much of the Internet's bandwidth and will require some upgrading of the current equipment for the Internet Service Providers. For these reasons, the owners of the VoIP gateways and the Internet Service Providers will probably start charging extra for VoIP calls. As well, faced with increased competition, the rates for conventional telephone calls will probably go down to compete with VoIP. As a result, the prices of VoIP calls over the Internet will probably eventually go up to a comparable level of that of regular telephony [57]. This means that the benefit of lower rates for VoIP calls over the Internet compared to regular telephony is probably only a temporary one for the consumer. If the medium is a private/corporate network though, the benefit of lower rates for VoIP calls will always exist. Since the customer is the owner of the network, no charges will ever be made for each telephone calls. The only costs will be to purchase the proper equipment.

Another important benefit for the customer is the added features that will be made possible with VoIP. As was discussed briefly in the section 2.4.2, the greater flexibility

of the data network will make additional features such as multi-party conversation easier to implement and consequently, more affordable. If the access method chosen for VoIP is a computer, we can have a better user interface than what we have right now with current telephones [63]. Another possibility would be to improve the quality of the speech [19]. Currently, the PSTN operates with speech sampled at 8 kHz. Even though such a sampling rate results in speech that is easily understandable and a speaker who is identifiable, the voice of the speaker on the telephone differs from the voice of the speaker in real life. This is due to the fact that we are not transmitting the higher frequencies of the voice, which would add improved faithfulness of the speaker's voice. Transmitting speech that is sampled at 16 or 32 kHz would increase the sensation of remote speaker presence to the listener [45]. With VoIP, it would be straightforward to increase the sampling rate to 16 kHz or beyond without requiring any changes to the network. The only changes that would need to be made are at the transmitter and the receiver where we would now need to support such a speech encoding. This would result in a quality of speech that is greatly superior and that resembles closely face to face conversation. In essence, VoIP makes possible a variable quality of service, where one can pay more to get better quality of service. If speech compression is used, this higher quality voice can even be transmitted at a lower bit rate than what is needed to currently transmit voice over the PSTN since current audio compression technology allows us to compress CD-quality audio at rates as low as 45 kb/s [68]. Such a variable quality of service is not possible for the PSTN since the equipment on the PSTN depends on the fact that the speech is transmitted at 8 kHz PCM. However it is easily achievable for

VoIP since the data network sees the voice as simply data and any format can be used as long as both sides of the conversation agree on the same format. In the age of the Compact Disk (CD) where better sound quality is expected, better sound quality might be used as an attractive feature of VoIP compared to regular telephony [57]. Another feature that could easily be added in VoIP telephony would be video in order to support video telephony, since voice and video essentially have the same characteristics. The number of additional features that can be added with VoIP telephony are endless and most of these probably do not currently exist. The adoption of Voice over IP will simply enable a new paradigm of services and features that are not possible to implement in today's PSTN [11]. Due to the effect such added features will have on how we communicate, the added features made possible by VoIP is probably the most important benefit of VoIP for the consumer [52].

2.5 Impairments of Voice Transmission over IP

2.5.1 Overview

If Voice over IP is so much better than regular telephony, why is everyone not using it already? The answer is simply that the IP environment suffers from different impairments that make implementing a good quality Voice over IP system quite a challenge. Since these impairments are more severe for the Internet than for a private/corporate IP network, most of the implementations so far have concentrated on private/corporate IP networks. In this section, the major impairments caused by the IP

network, delay, jitter and packet loss will be presented and a brief discussion of how these impairments are dealt with in a VoIP system will be given.

2.5.2 Delay

Because of all the components involved in a VoIP system and the fact that the data network was not designed for real-time applications, much delay is present in a VoIP implementation. This delay can be as high as 556 ms in some implementations [17]. Since user studies indicate that telephone users find round-trip delays of greater than 300 ms more like a half-duplex connection than a conversation [37], such a delay is excessive. Note that human tolerance for delay in a telephone conversation varies from user to user and that some more tolerant users are satisfied with delays of 300-800 ms [37]. Another impact of high delay is that the work of the echo canceller is greatly complicated. Since longer delays for the same echo are more disturbing, the echo canceller must remove more of the echo to keep the quality acceptable. VoIP applications then require better echo cancellers compared to the PSTN where the delay is low and the echo canceller does not need to perform as well. It is also important to determine the amount of delay in the system accurately. A method to do this is detailed in [78]. In implementing a VoIP system, one should be careful to limit the amount of delay introduced as much as possible for the two reasons described previously: delay is deterrent to proper two-way conversation and delay complicates the work of the echo canceller. This means that for all components over which we have control, the delay introduced by the components should be kept as low as possible. The voice delay is one

of the most important parameter affecting the perceived Quality of Service of a VoIP call [70].

Figure 2.6 specifies delays between components in a VoIP system while Table 2.4 provides the corresponding typical numerical values of these delays [17].

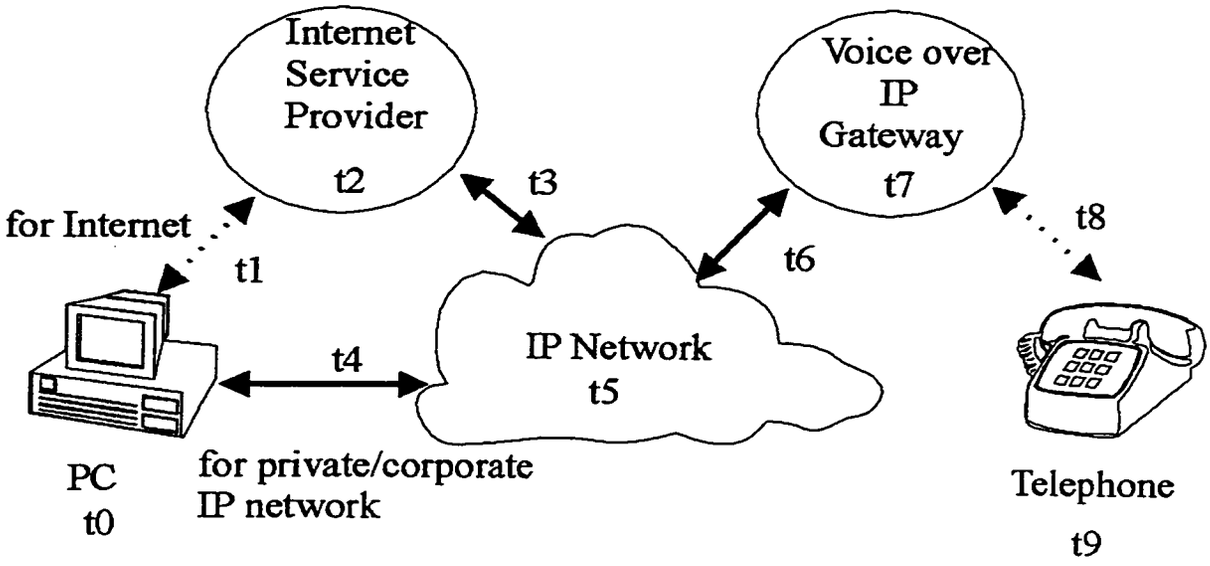

Figure 2.6: Delay in a VoIP System

As can be seen from Table 2.4, the main sources of delay in a VoIP system when the access method is a PC are the PC itself, the modems involved (if we are dealing with an ISP) and the IP network. If the access method is a telephone, the main sources of delay are the VoIP gateway and the IP network. As discussed in section 2.2.4, the delay associated with an IP network is quite variable and depends heavily on the amount of traffic in the network and the number of router hops required to get to the intended destination. The value given in Table 2.4 is only for a particular Internet connection.

Multi-hop transmission delay on the public Internet can be as high as 500 ms or more [67].

Table 2.4: Delay associated with Symbols and Explanation

Symbol	Delay (ms)	Explanation
t0	150	Delay associated with PC processing in sound card, speech compression algorithm and playout buffering
t1, t2, t3	150	Delay associated with modems at PC and Internet Service Provider
t4	negligible	Delay introduced by network interface card
t5	96	Delay associated with IP network (value given is for a particular Internet connection, value for corporate network would be lower)
t6	negligible	Delay for gaining access to IP network from VoIP gateway
t7	160	Delay introduced by VoIP gateway because of speech compression, playout buffering and other VoIP tasks as well as buffering delay (a VoIP gateway serves many sources)
t8	negligible	Delay for telephone line
t9	negligible	Delay introduced by telephone

2.5.3 Jitter

Jitter is defined as the variability of delay suffered by different packets. Since in an IP network the data does not always travel by the same route, the amount of delay experienced by consecutive packets can differ to the point where the order in which data is transmitted differs from the order in which it is received. For regular data, this does not represent much of a problem since we can easily re-assemble the data in the right order and wait until we have all the data we need before processing it. However, when dealing with speech, the data must be processed in a continuous manner such that the spacing between successive samples after decoding is identical to that of the transmitted signal. Jitter is then unacceptable for speech and must be properly dealt with to ensure good speech quality in a VoIP system. As discussed in section 2.3.3 and as displayed in Figure 2.5, the part of the VoIP system that deals with the jitter is the playout buffer. Examples of implementation of playout buffers are given in [12][55]. The playout buffer is essentially a buffer that inserts delay between the received packet in such a way that it has time to re-order them and to play them out in a constant manner.

The easiest way to understand the role of the playout buffer is to consider an example where the delay introduced by the playout buffer is fixed. Consider a playout buffer of delay 100 ms and packets that contain 10 ms of speech. To achieve good quality speech, a new speech packet must be played every 10 ms otherwise we will have a gap of silence. Suppose further that we received packet #1 at 10 ms, packet #3 at 80 ms and packet #2 at 100 ms as illustrated in Figure 2.7. Note that the number associated with the packet

indicates the order they were transmitted and the order in which they should be played at the destination to reproduce the correct speech. If no buffer is used, we would only start playing packet #1 when it arrives at 10 ms, causing an initial silence of 10 ms. We would then play packet #3 at 80 ms, causing a gap of silence of 60 ms, followed by packet #2 at 100 ms, causing an additional gap of silence of 10 ms. Taking into account the amount of silence introduced that was not initially there and the fact that the speech was not played in the right order, the quality of the speech for a system with no buffer is clearly unacceptable. Now reconsider the same case with a playout buffer with a delay of 100 ms. Packet #1 is received at 10 ms, but it will not be played before 100 ms due to the playout buffer. At 80 ms, we receive packet #3 and we simply buffer it since we do not need to play it at that time. At 100 ms, we receive packet #2 and the speech of packet #1 is played. At 110 ms, we play packet #2 and at 120 ms, we play packet #3 which was stored in the buffer. The result is speech that has been delayed by 100 ms but is perfectly intact compared to the original. This example clearly indicates the purpose of a playout buffer and why it is essential in a VoIP system.

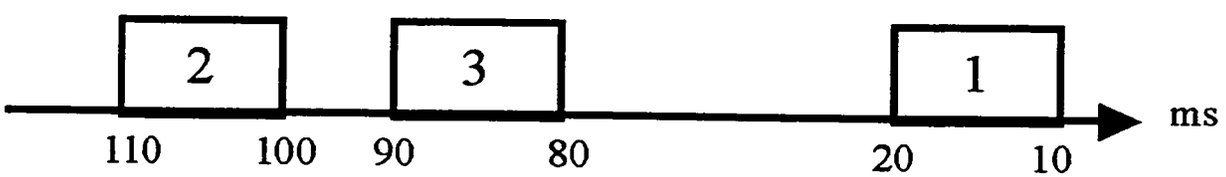

Figure 2.7: Packet Arrival Example

From this example, we can see that the delay introduced by the playout buffer is essentially the maximum amount of delay that is permitted for a packet in the system. If

a packet arrives past this maximum amount of delay, it will not be considered and the system will need to "play" something in its place in order to maintain the constant playout interval. The playout buffer essentially removes all the jitter of the system at a cost of additional delay and dropping packets that arrive too late. The amount of delay added by the playout buffer is chosen large enough to ensure that most packets have time to arrive but small enough to avoid adding too much delay to the system [12]. To optimize the performance of the playout buffer, the size of the buffer can be made adaptive. By tracking the delay suffered by each packet through a scheme such as RTCP (see section 2.2.3), the buffer can adjust its delay to minimize the overall delay introduced and maximize the number of packets received on time. In order that the change of playout buffer delay not be noticeable, it is usually done progressively and only during periods of silence in the speech. Adaptive playout buffers use algorithms such as LMS to optimize performance [12]. Since jitter is unacceptable for speech, the use of an adaptive playout buffer is essential in a VoIP system and translates the problem of jitter into a problem of delay and packet loss.

2.5.4 Packet Loss

The final impairment that has considerable impact on the performance of a VoIP system is the packet loss experienced in the system. The three different elements that contribute to packet loss in a VoIP system are: channel noise, router queues, and the jitter buffer. These are illustrated in Figure 2.8 that follows.

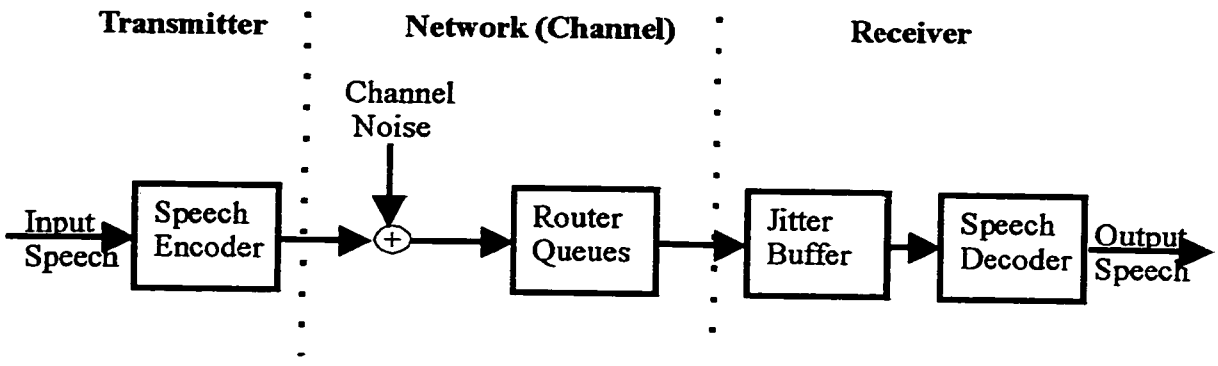

Figure 2.8: Sources of Packet Loss in a VoIP System

A packet can be lost due to any of the following reasons: arriving too late to the adaptive jitter buffer, severe congestion on the IP network causing overflow of the router queues or corruption of the IP header due to channel noise (a packet with a corrupted header is automatically dropped by a router). A packet loss clearly degrades the quality of the speech since the lost packet cannot be recovered due to delay constraints and must be covered up with error concealment methods. The frequency at which packet losses occur depends on the level of traffic on the IP network. In an experiment done on the Internet in [37], it was found that packet loss could reach 25% in some instances. It was also found to be highly correlated meaning that much of the packet loss occurred in bursts. Actual packet loss experienced on the Internet could be even higher than 25% since one experiment is not sufficient to characterize a complex system such as the Internet[47]. As such, properly handling of packet loss is an area of high importance in a VoIP system. Further discussion of error concealment methods will be made in 4.5 after the basics of speech compression have been properly introduced.

2.6 Summary

This chapter started with an introduction of IP networks. The role of routers in an IP network and the data networking protocols involved in Voice over IP (VoIP) were discussed. As well, it was shown that an IP network does not guarantee any Quality of Service (QoS) which is required for VoIP systems. An overview of Voice over IP (VoIP) and its major components was then presented. It was shown that it is possible to make a VoIP call either through a regular telephone or through a multimedia PC. Both of these two methods have their own advantages and disadvantages. The main components of a VoIP implementation were presented including speech compression, echo cancellation, playout buffer, voice activity detection, comfort noise generation, error control coding and packetization. The main benefits of VoIP over the regular telephone service were given for both the telecommunication companies and the customers. It was shown that VoIP permitted better utilization of the bandwidth, the integration of the voice and data networks, easier addition of additional features and a lower cost for making telephone calls. Finally, the main impairments that degrade the quality in a VoIP system were detailed. These are the high delay in the system, the jitter present in the IP network and the high probability of packet loss. It was shown that the jitter could be dealt with entirely by a playout buffer and translating the problem of jitter to a problem of additional delay and packet loss.

3 Speech Compression

3.1 Overview

Speech compression is essentially the encoding of a speech signal using fewer bits by exploiting the redundancies that exist in speech. Since speech compression is central to all Voice over IP implementations, it is important that the subject be well understood. As well, the background on speech compression given in this chapter will be of great use to understand the major contribution of this thesis in chapter 5.

This chapter will start by discussing the properties of a speech signal. This will be followed by the introduction of the simplest and most common family of speech compression, waveform-based speech compression. These algorithms code the speech waveform directly, converting the analog signal to a digital signal. A general definition of the family of waveform-based speech compression will be given as well as an overview of PCM. Next, the family of perceptual-based speech compression algorithms will be discussed. These algorithms exploit the fact that the human ear cannot hear some parts of a speech signal. A general description of the main idea behind this family of speech compression algorithms, a list of current international standards and a description of the main components of these algorithms will be given. Next, the family of model-based speech compression algorithms will be presented. These algorithms exploit the human speech production model in order to reduce the required bit rate. We start by

reviewing the human speech production model. This will be followed by the description of CELP, a popular member of the model-based speech compression family. Finally, we will examine different international standards for model-based speech compression and a full description of the ITU standard G.729A will be given.

3.2 Characteristics of the Speech Signal

A speech signal is a sound formed by humans in order to communicate. Like all other sound, it is a perturbation of the atmospheric pressure that travels in the form of a sound wave. This sound wave can propagate through the air and is perceptible when it enters our ears.

To better understand the properties of human speech, an examination of how it is formed is in order. The vocal tract is responsible for forming speech. It is basically an acoustic tube with a non-uniform cross-section consisting of two parts: the oral tract (from lips to the vocal cords) and the nasal tract (from the valum to the nostrils) [34]. Note that the valum is a small valve that opens and closes to allow the connection of the oral tract with the nasal tracts. In some analysis, the speech signal is broken down into its smallest portions, which we call phonemes. Here, we will instead describe the speech as either being voiced or unvoiced [36]. A voiced speech is formed with the use of the vocal cords and is characterized by its high energy content and its periodicity which is called pitch. Unvoiced speech on the other hand is not formed with the use of the vocal cords, but

instead by passing continuous air freely through the vocal tract. This results in a signal that is random in nature (it has no periodicity). It is important to note that some part of the speech is neither voiced nor unvoiced, but a mixture of the two. This is due to the transition period when the speaker changes from voiced to unvoiced speech or vice versa.

To use speech in communication over networks, we need a way to transform the speech signal from a sound wave to an electrical signal. This is usually done with a microphone and the result is an electrical signal where the voltage varies with the pressure of the speech signal. Like all other natural signal, the speech signal is originally an analog signal. This means that the electrical signal that we obtain using a microphone is also analog. It is usually preferred to use a digital version of the speech since digital transmission is more efficient and reliable than analog transmission [21]. Speech can be represented digitally in different ways using speech encoding algorithms. If the number of bits required to represent the digital signal is reduced further, this is called speech compression. The most popular families for speech encoding and compression will be discussed in the remaining of this chapter.

3.3 Waveform-based Compression

3.3.1 Overview

Waveform-based speech compression is the simplest family of speech compression algorithms. Members of this family include pulse code modulation (PCM), differential

pulse code modulation (DPCM) and adaptive differential pulse code modulation (ADPCM). These algorithms are classified as waveform-based compression algorithms because they code the speech waveform directly by representing the analog speech waveform in a digital form as faithfully as possible.

In this section, a general description of PCM will be given. A good understanding of PCM is important since it often serves as the input to other speech compression algorithms and it is the format used for conventional telephony. As well, the principles of sampling and quantization used in PCM are very important in speech compression. DPCM and ADPCM are essentially a slightly modified version of PCM that use prediction to reduce the redundancy in the speech and to adapt to the characteristics of the speech in order to obtain compression. They will not be discussed in this section since they are not relevant to this thesis.

3.3.2 PCM

Pulse code modulation is the standard way to encode an analog speech signal into a digital speech signal. Even though it requires a high bit rate, it is still widely used in telephony because of its simplicity.

An analog signal can have continuous values at any instant in time. For a digital signal, we cannot store values of the signal for all possible values of time and we cannot represent infinite precision since a computer must limit the number of bits it takes to

encode the sample values. For this reason, we must sample the analog signal in order to only keep the values of the analog signal at specific instances in time and only keep these values quantized at a finite number of bits. The basic principles of sampling and quantization are assumed to be known by the reader (the interested reader is referred to [53]). In order to clarify the notation, remember that regular quantization where each sample is quantized individually is called scalar quantization. There are many different types of scalar quantization including uniform, optimum, logarithmic and adaptive quantization [36]. Vector quantization is a different approach that does not consider each sample individually as in scalar quantization, it rather considers them in a group. The basic principle behind vector quantization is illustrated in Figure 3.1 [36]. The samples to be quantized are first buffered into a group of samples that we call a vector. A codebook that contains a list of possible vectors of the same length as the input vector is then searched for the vector that most closely resembles the input vector. The simplest method that can be used to search the codebook is to choose the codebook vector that is closest to the input vector in the least mean square (MS) sense. Once a vector has been chosen from the codebook, we only need to transmit the index of the vector since the receiver has the same codebook and can rebuild the vector with the index. The reduction in bit rate comes from the fact that the size of the index is usually quite smaller than the bits needed to encode the samples in the vector individually. For this to be true, the size of the codebook must be kept to a minimum. To ensure good performance, the vectors in the codebook must be chosen with care [36].

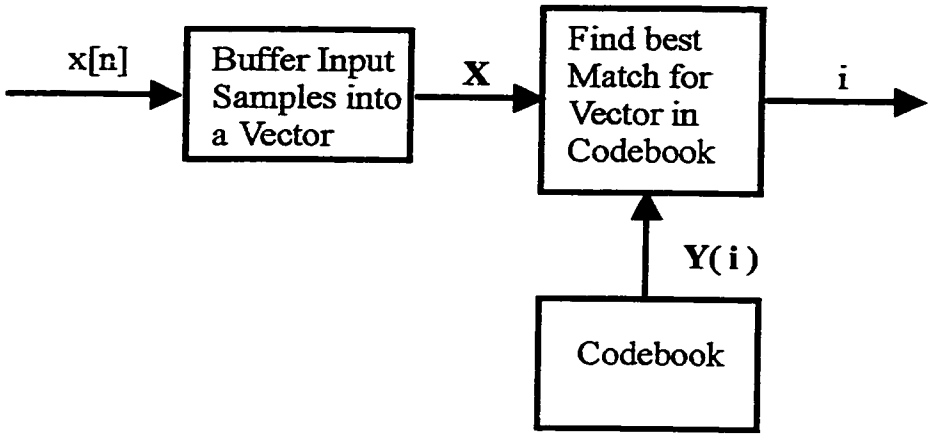

Figure 3.1: Block Diagram of Vector Quantization

The international standard for PCM is ITU-T G.711. This is the standard that is supported in conventional telephony. It uses logarithmic scalar quantization and represents each speech sample with 8 bits. The analog speech is first band-limited to the typical telephone bandwidth of 3.4 kHz and is sampled at 8 kHz. This results in a total bit rate of 64 kb/s. It is widely used due to its simplicity and very low delay. The only drawback of PCM is that it requires a high bit rate.

3.4 Perceptual-based Compression

3.4.1 Overview

The main idea behind perceptual-based speech compression is not to code the part of the speech that is inaudible. The human ear is not capable of hearing all sounds since some sounds do not meet the minimum threshold of audibility of the ear while others are masked (made not hearable) by another louder sound of similar frequency [7].

In perceptual speech compression, a perceptual model permits the encoder to distinguish the audible elements of the speech. By only encoding what is audible, a significant reduction in bit rate can be achieved with no loss of quality. In this section, some of the international standards for perceptual speech compression will be introduced. Note that even though these standards were initially developed for audio compression, nothing prevents us from using them for speech compression. This will be followed by a high level description of the components of a typical perceptual speech compression algorithm: the perceptual model, the filter bank and the perceptual quantizer.

3.4.2 International Standards

All standards for perceptual compression were developed for audio compression. This permits encoding of CD quality speech at only 64 kb/s [35]. The compression obtained by the perceptual modeling permits us to encode speech of higher quality than PCM at the same or lower bit rate.

Some international standards for perceptual compression are MPEG-1 Layer 3 [46] (commonly known as MP3), Dolby AC-2 and AC-3 [14], the Perceptual Audio Coder (PAC) [66], and MPEG-2 Advanced Audio Coding (AAC) [6]. Characteristics of these international standards are highlighted in Table 3.1.

Table 3.1: International Standards for Perceptual-based Speech Compression

Standard	Sample Rate [kHz]	Bit Rate [kbps]	Year finalized
Dolby AC-2	44.1	256/channel	1984
Dolby AC-3	44.1	32-384	1994
Perceptual Audio Coder (PAC)	44.1	128/stereo channel	1995
MPEG-1	32-48	32-448	1992
MPEG-2	16-48	16-	1994
MPEG-2 AAC	8-96	variable, max 48-	1997

Instead of describing one of these standard in particular, the remaining part of this section will discuss the main components of a typical perceptual compression algorithm.

3.4.3 Components of a Typical Perceptual Compression Algorithm

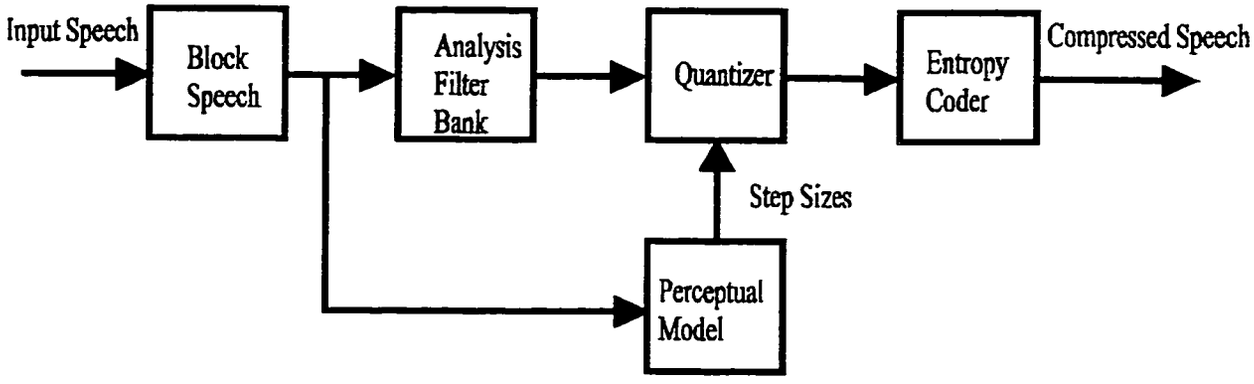

Figure 3.2: Block Diagram of a Perceptual Speech Compression Encoder

Figure 3.2 shows the block diagram of a typical perceptual speech compression encoder. The input to the system is digital speech, essentially a speech signal that has been quantized to 8 bits per sample, as is described for PCM in section 3.3. This digital speech signal is then decomposed into blocks of a fixed size. The size of the block can vary between 256 and 2048 samples and depends on the implementation. Note that the blocking of the speech is the main source of delay in a perceptual speech compression algorithm and that this delay can be quite significant when the block size is large. Each block of speech is then passed to the analysis filter bank and the perceptual model. The purpose of the filter bank is to decompose the speech into parts of the speech that have similar characteristics [73]. For sound, these are called critical bands and represent groups of frequencies where the masking is uniform. The purpose of the perceptual model is to determine which part of the speech is audible. The perceptual model is very important in perceptual compression and will be described in more details in section 3.4.4. Following this, the block of speech is passed to a perceptual quantizer. With the help of the perceptual model, the perceptual quantizer is responsible for allocating fewer bits for the less audible speech components [56]. The simplest technique is to use a series of uniform quantizers where the step size used is equal to the threshold of audibility for that frequency. Since the error introduced by a uniform quantizer is always less than the quantizer step size, this method ensures that the quantization error introduced is inaudible. Finally, the perceptually quantized block of speech is optionally passed to an entropy coder. The purpose of the entropy coder is to allocate fewer bits for symbols that are more probable. This can be accomplished with Huffman coding or arithmetic coding.

Entropy coding is lossless which means that it reduces the bit rate without affecting the quality of the speech [32]. Details of entropy coding will not be discussed here since it does not degrade the quality of the speech and is not relevant to this thesis.

3.4.4 Perceptual Model

In this section, further details will be given on the perceptual model since it is the central part of all perceptual speech compression algorithms. A proper understanding of the perceptual model is also important to the comprehension of the algorithm described in section 4.7.4.

The perceptual model plays a central part in perceptual speech compression since it is responsible for determining the inaudible parts of the speech signal. By taking into account the properties of the human ear and the current block of speech, the perceptual model is able to determine the masking threshold for all frequencies. The masking threshold is essentially the threshold of audibility for all tones in the current block of speech. In the sections that follow, the perceptual model will be further discussed.

3.4.4.1 Threshold in Quiet

The first thing the perceptual model takes into account is the threshold in quiet. This determines the sensitivity of the human ear to different frequencies. Through extensive psycho-acoustic testing, the level at which a particular tone must be for it to be audible by a human listener was determined [15]. The results of these tests were combined and

averaged out to give a curve of the threshold in quiet. Through curve fitting techniques, these results can be approximated by the following formula where f represents the frequency in Hertz [38]:

$$ATH(f) = 3.64(\frac{f}{1000})^{-0.8} - 6.5e^{-0.6(f/1000-3.3)^2} + 10^{-3}(\frac{f}{1000})^4 \quad (1)$$

To better understand the effect of the threshold in quiet, this formula is plotted in Figure 3.3. A low value of threshold in quiet indicates that a tone with a low magnitude at that frequency can be heard. From the plot of the threshold in quiet, we can see that the frequencies between 500-8000 Hz are the most easily audible while very low frequencies and high frequencies are harder to hear.

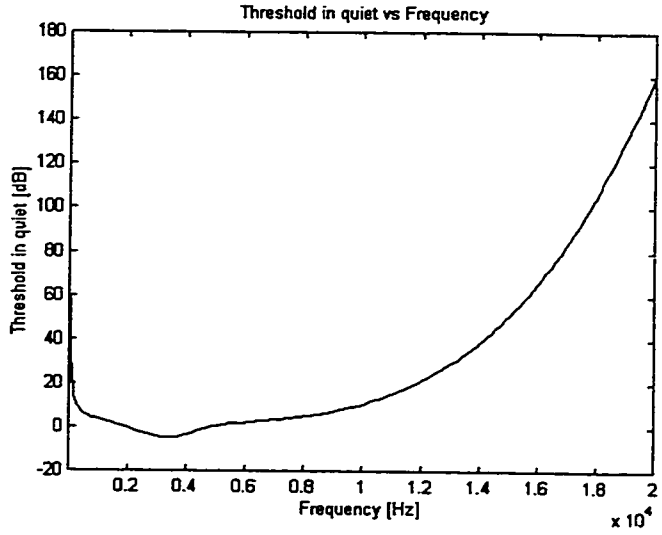

Figure 3.3: Threshold in Quiet

3.4.4.2 Simultaneous Masking

The next phenomenon that the perceptual model must take into account is simultaneous masking. This is the property of the ear that stipulates that a loud tone at a certain frequency can make another neighboring tone inaudible.

The first step in calculating the simultaneous masking is to calculate the discrete Fourier transform of the current block of speech using a Fast Fourier Transform (FFT). This permits us to examine the frequency content of the speech, which is a more appropriate representation for the simultaneous masking. We then consider each frequency component as a masker and calculate the amount of simultaneous masking each one causes. The total amount of simultaneous masking is the sum of the masking performed by each individual masker.

To calculate the amount of simultaneous masking caused by a certain masker to neighboring frequencies, we start by estimating the tonality of the masker [38]. Since noise is a better masker than a tone [46], tonality must be taken into account in calculating the amount of simultaneous masking. Methods for estimating the tonality of a certain frequency usually rely on the predictability of the Fourier Transform. Next, the fact that masking is more effective when the two tones are close by must be taken into account. This is done by a spreading function, an example of which is given in Figure 3.4.

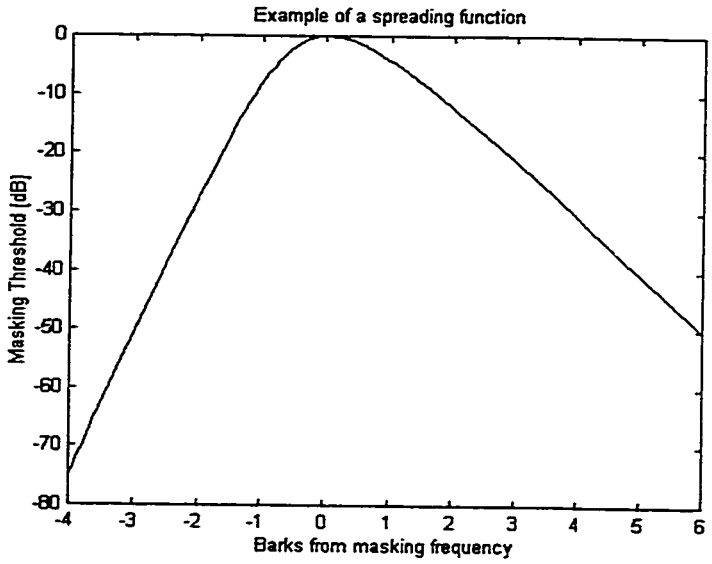

Figure 3.4: Spreading Function

The x-axis in Figure 3.4 is given in bark units, where one bark is equivalent to all the frequencies that belong to the same critical band. Examining the spreading function, we conclude that indeed masking is maximum within the critical band of the masker (the masker is at bark 0). As we distance ourselves from the masker, the amount of simultaneous masking is reduced. Note that the amount of masking reduces at a slower rate for frequencies that are higher than the masker as compared to frequencies that are lower than the masker.

Finally, the amount of simultaneous masking depends on the power of the masker. A very loud masker will mask more than a masker of lower power. It is then possible to calculate the amount of simultaneous masking that is caused by a certain masker to neighboring frequencies.

3.4.4.3 Temporal Masking

Temporal masking is similar to simultaneous masking except that the masker does not occur at the same time as the maskee (what is being masked). It has been shown through psycho-acoustic experiments that a loud tone that occurred up to 200 ms in the past can mask another tone in the present [15].

A simple way to take into account temporal masking is to interpolate the current masking threshold from values in the past. This way, a loud tone in a previous block is capable of masking a tone in the current block, though the amount of masking is reduced.

3.4.4.4 Masking Threshold

The masking threshold for a block of speech, which represents how audible the tones are at different frequencies, is formed by summing the effects of the threshold in quiet and the simultaneous masking. The temporal masking is taken into account by interpolating this sum with past values of the masking threshold. With the masking threshold calculated, the perceptual speech compression algorithm can determine the specific frequencies in this block of speech that are highly audible and can then quantize the block of speech accordingly.

3.5 Model-based Compression

3.5.1 Overview

Model-based speech compression algorithms use a model of human speech production to extract from the speech parameters for this model. Considerable compression can be obtained by transmitting these parameters in place of the actual speech.

In this section, we present a model of human speech followed by the class of code-excited linear prediction (CELP) algorithms. This class of model-based speech compression permits high quality speech at a low bit rate. International standards of model-based speech compression algorithms will then be presented. Finally, the description of the international standard ITU-T G.729A will be given. This is a member of the CELP category and is described here since it is central to this thesis and will be used extensively in chapter 5.

3.5.2 Model of Human Speech Production

A model of human speech production is given in Figure 3.5 [36]. To understand this model properly, we must remember what was discussed in section 3.2: speech can be characterized as being either voiced or unvoiced. In the model shown here, an impulse train generator represents voiced speech. The pulses generated by the impulse train generator are spaced out at equal intervals according to the pitch of the speech. A random noise generator represents unvoiced speech. To be able to properly represent the different speech levels, an adequate gain must be applied to the impulse train or the

random noise. The product of the gain and the impulse train for voiced speech or the product of the gain and the random noise for unvoiced speech form what is called the excitation. The excitation in itself does not represent speech since it does not posses the short-term predictability that is present in the speech. An all-pole adaptive filter implements this very important feature.

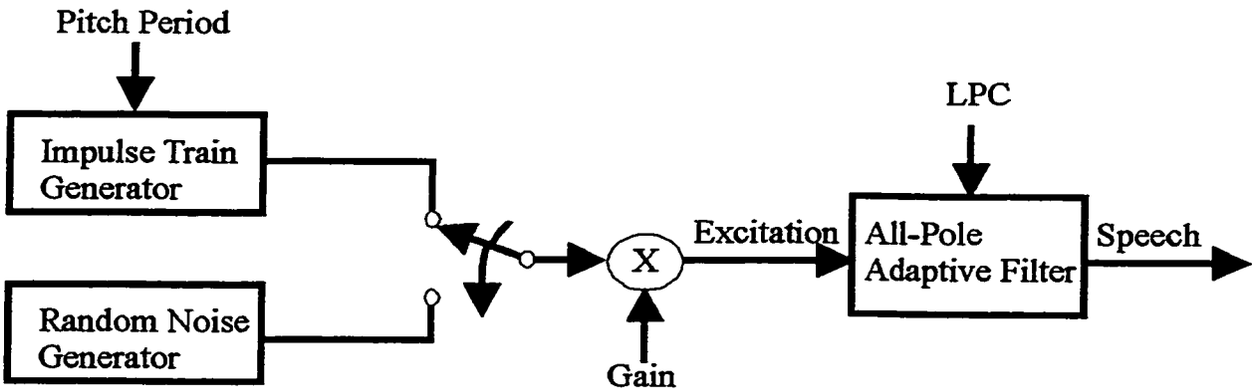

Figure 3.5: Model of human speech production

The all-pole adaptive filter can be represented in the Z-domain as being:

$$H(z) = \frac{1}{1 + \sum_{i=1}^{N} a_i z^{-i}} \qquad (2)$$

where a_i is the i^{th} linear predictor coefficient (LPC). The size of the filter, N, is typically chosen around 10 but can vary between 8 and 12 depending on the implementation. The LPC are extracted from the input speech using linear prediction. Since speech is highly non-stationary, these coefficients must be extracted from a fairly small segment [36]. The Levinson-Durbin algorithm is the most popular method to extract these coefficients [20].

To recapitulate, the model of human speech production first decides whether the current segment of speech is voiced or unvoiced. If the speech is judged to be voiced, an impulse train generator is used to generate pulses that are separated according to the value of the pitch that is determined from the segment of speech. This series of pulses multiplied by a proper gain forms the excitation. If the speech is judged to be unvoiced, a random noise generator generates a random noise that is multiplied by a proper gain to form the excitation. In both cases, the excitation is then filtered through a LP synthesis filter that uses as coefficients the LPC extracted from the speech. The output of the filter is speech that closely approximates the original.

3.5.3 Code Excited Linear Prediction

Code-Excited Linear Prediction (CELP) based speech compression algorithms compress voice using the model of the production of human speech described in the previous section. In contrast to vocoders which is a class of model-based speech compression algorithms that implement the above model directly, CELP modifies it slightly to produce higher quality speech. The modifications to the model are illustrated in Figure 3.6.

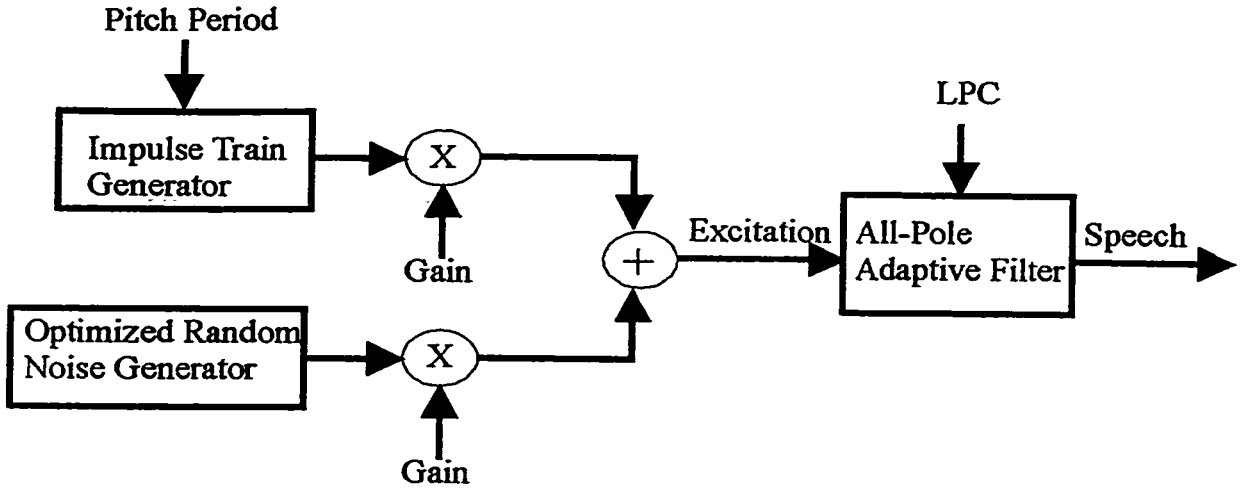

Figure 3.6: Modified Speech Production Model

One of the main differences is that CELP does not make a decision on each segment to determine if it is voiced or unvoiced. Instead, each speech segment is considered to be a mixture of voiced and unvoiced speech. For this reason, the excitation is always formed from the sum of the voiced speech excitation (impulse train) and the unvoiced speech excitation (random noise). Both have their own gain to control the level of the speech. This provides considerable improvement of the speech quality given that many speech segments are not totally voiced or unvoiced but a mixture of the two. This occurs quite often in the transition period between a voiced speech and an unvoiced speech. By always representing speech as a mixture of voiced and unvoiced speech, better quality can be obtained in such conditions.

Another main difference is the choice of the random noise to represent unvoiced speech. In a typical vocoder implementation, pure random noise is used. This means that the random noise to use for the production of the synthesis is not a parameter of the speech

model and can be chosen randomly at the decoder. This results in speech that is understandable but that has lost much of the speaker's unique characteristics [36]. In CELP, the random noise used to represent the unvoiced speech is a parameter of the speech production model. Using the analysis-by-synthesis (AbS) approach, the CELP coder chooses the random noise that will result in the best quality speech [54]. Analysis-by-synthesis is a technique that searches for the best random noise from a certain codebook by synthesizing the speech using a certain subset of this codebook. Each time the speech is synthesized, the quality of the speech is assessed by comparing it with the original speech segment (usually using a MSE criterion). The codebook vector that results in the least degradation is chosen. Once the encoder has determined the best codebook vector, the index of this vector is transmitted to the decoder. Since the decoder has the same codebook, the best random noise can be used to synthesize the speech at the decoder. Obviously, this results in better quality speech since the random noise is properly chosen to limit the degradation. Another advantage of this technique is to prevent errors from accumulating and degrading the quality of the speech [36]. Since the encoder synthesizes the speech in the same way as the decoder, the encoder can guarantee a certain quality at the decoder.

Now that the differences between CELP and vocoders have been highlighted, the general algorithm for CELP encoding can be presented in the form of pseudo-code for one segment of speech. Obviously, for encoding more than one segment of speech, this process would simply need to be repeated.

1. Take as input a small segment of speech (size of segment is between 5-35 ms).
2. Extract from the speech the LPC (usually use Levinson-Durbin algorithm to do this).
3. Extract from the speech the pitch through a pitch detection algorithm. Many algorithms exist for this and the one chosen depends on the implementation. In general, these algorithms use an analysis-by-synthesis technique to determine the value of pitch that causes the less degradation.
4. Gains to be used are extracted from the speech using the energy of the speech segment.
5. The best random noise is chosen from the codebook using analysis-by-synthesis. This involves synthesizing the speech following the speech model in Figure 3.6 using different random noise chosen from the codebook. The one that results in the least degradation is chosen.
6. Send all parameters (LPC, pitch, gain and codebook index for random noise) to the decoder.

The different members of CELP algorithms perform these tasks differently, but they all essentially need to find the same parameters. An example of a CELP algorithm, ITU-T G.729A, will be described more fully in section 3.5.5.

3.5.4 International Standards

Many different international standards exist for model-based speech compression. Most of these have been made popular by their use for cellular telephony or other specialized applications. A list of the most prominent standards is given in Table 3.2 with their associated bit rate, algorithmic delay and the year they were finalized [10].

Table 3.2: International Standards for Model-based Speech Compression

Standard	Bit rate	Algorithmic delay	Year finalized
ITU-T G.728 LD-CELP	16 kb/s	0.625 ms	1994
ITU-T G.729 CS-ACELP	8 kb/s	15 ms	1995
ITU-T G.723.1 MPC-MLQ	5.3 or 6.4 kb/s	37.5 ms	1995
ITU-T G.729A CS-ACELP	8 kb/s	15 ms	1995
GSM RPE-LTP	13 kb/s	20 ms	1987
FS-1015 LPC-10E	2.4 kb/s	111.5 ms	1984
FS-1016 CELP	4.8 kb/s	37.5 ms	1991

All recent entries in this table (after 1990) are CELP algorithms. This shows how popular these are for model-based speech compression. One standard vocoder was included for comparison. It is the FS-1015 LPC-10E. It is important to note that while the bit rate for this algorithm is significantly lower, the speech quality is not as good as other algorithms in this table. However, it is still used in some applications where low bit rate is more important than the quality of speech.

Among the algorithms in Table 3.2, the currently most popular speech compression algorithms for Voice over IP are ITU-T G.723.1 [25], ITU-T G.729 and ITU-T G.729A. Since these are recent algorithms, they all give good quality speech at a fairly low bit rate. In the next section, ITU-T G.729A will be examined more thoroughly.

3.5.5 ITU-T G.729A

The G.729A algorithm is an annex to the G.729 algorithm that offers the same toll quality speech in most cases as does G.729 at half of the complexity [62]. Because of this reduced complexity, G.729A is most often preferred over the more complicated G.729.

The G.729A algorithm bases itself on the previously defined speech production model to generate a selected set of parameters to best represent a frame (previously called segment) of speech. Each frame represents 10 ms of speech sampled at 8 kHz and is further sub-divided into two 5 ms sub-frame for finer resolution. The encoder requires the current frame, three previous sub-frames and one future sub-frame to extract the required parameters from the speech [26][27]. This means that the encoder has a total algorithmic delay of 15 ms (since it requires the current frame and one future sub-frame).

The algorithm follows the human speech production model quite well by generating an excitation that will then be filtered through a LP synthesis filter. The excitation is formed as the sum of an adaptive codebook vector which represents the long term periodicity in the speech (the voiced speech) and a fixed codebook vector which represents the randomness in the speech (the unvoiced speech). An appropriate gain is used for each vector to represent the speech level appropriately.

The encoder extracts most of the parameters from the speech through analysis-by-synthesis. This means that the speech encoder selects its parameters through a trial and

error type method by generating the synthesized speech with a given set of parameters and iterating until it finds the parameters that lead to the least perceptual difference between the original and the synthesized speech. The parameters that the encoder extracts for each frame (all of the parameters must be calculated for each sub-frame except the LPC that are kept the same for the full frame) are listed in Table 3.3. In all, the G.729A encoder generates 80 bits per frame for a total bit rate of 8 kb/s. Note that for many of these parameters, some fairly detailed algorithms are used to find the most appropriate values [61].

The easiest way to understand the G.729A speech compression algorithm is to examine the decoder. As for most compression algorithms, the decoder is simpler than the encoder, which makes it easier to understand. The decoder essentially uses the parameters that were determined by the encoder to synthesize the speech through a method quite similar to the speech-processing model of Figure 3.6. A block diagram of the decoder is illustrated in Figure 3.7.

Table 3.3: Parameters Transmitted by the G.729A Encoder

Parameter	Bits Needed	Description
LPC	18	Coefficients that represent the short term periodicity in the speech. As discussed in the speech production model, these are used to generate the speech from the excitation by using them as filter coefficients. These are actually transmitted in the form of line spectral frequencies (LSF), which is an alternative way to represent them that is better suited to quantization.
Pitch	14	Represents the long-term periodicity in the speech. This parameter is used to generate the adaptive codebook vector. Because of its importance, it is protected with one parity bit and is calculated to half sample accuracy.
Fixed codebook vector	34	Represents the randomness in the speech. It consists of four pulses of amplitude +/- 1 that are positioned at arbitrary positions between 0 and 39 in a sub-frame. The location and sign of the pulses are selected through analysis-by-synthesis.
Gains	14	These are the two gains that multiply the adaptive codebook vector and the fixed codebook vector. Both of these are vector quantized using one codebook. Prediction is used for the fixed codebook gain.

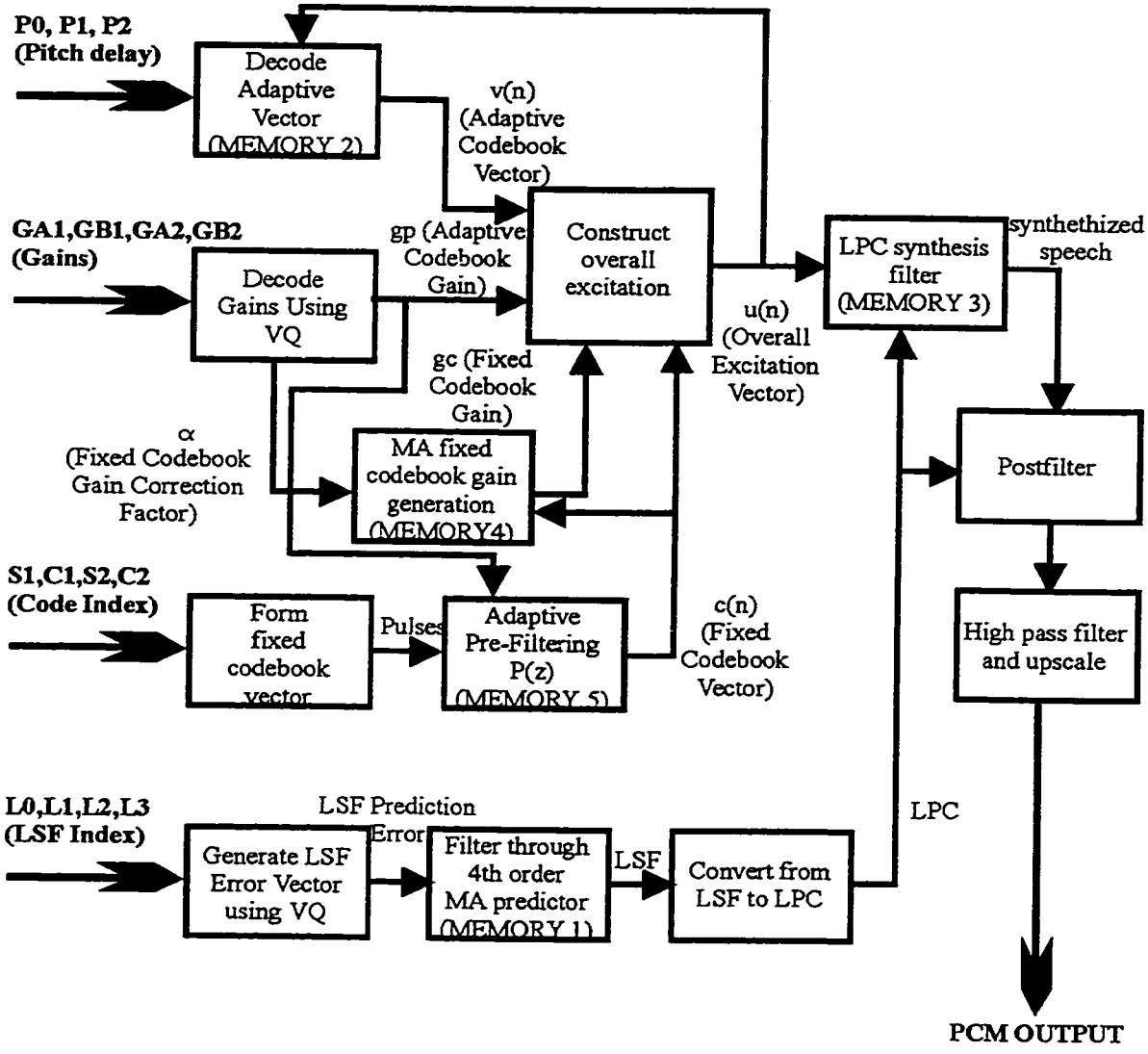

Figure 3.7: Block Diagram of G.729A Decoder

The decoder first starts by decoding the parameters that it has received from the encoder. It must decode the values of the pitch that it has received for each sub-frame and verify that no bit errors have occurred in the pitch value with the help of the parity bit. If the parity bit indicates a bit error in the pitch value, the decoder will disregard what was received and will treat the pitch the same way as if the whole packet was lost (to be explained at the end of this section). To build the adaptive codebook vector, the past

overall excitation values u(n) are interpolated according to the pitch value. For the fixed codebook vector, the excitation is built directly from the parameters received. This excitation then passes through an adaptive pre-filter to enhance the quality of the excitaiton. This pre-filter is a simple first order all-zero filter that uses as coefficient the past value of the adaptive codebook gain. Before the overall excitation can be formed, the gains must be decoded. Since both have been vector quantized, the first step is to retrieve the vector from the codebook. This results in the adaptive codebook gain and the prediction error of the fixed codebook gain. To get the value of the fixed codebook gain, we must pass this error through a 4^{th} order Moving Average (MA) predictor since the same predictor was used at the encoder to reduce the required number of bits for this parameter. Once the gains are decoded, we can form the overall excitation by multiplying the fixed codebook and adaptive codebook vectors by the appropriate gain and sum up the result. Before we can filter this excitation, we must decode the LPC. The LPC are transmitted as line spectrum pairs (LSP) in terms of line spectrum frequencies (LSF). This is a frequency domain representation of the LPC that is better suited for quantization and exhibits more correlation than the LPC [36]. Taking advantage of this correlation, a 4^{th} order Moving Average (MA) predictor is used to reduce the number of bits needed to send the LSP coefficients and only the prediction error is sent. Since the prediction error is vector quantized, the first step in obtaining the LPC at the decoder is to retrieve the appropriate vector from the codebook. The prediction error is then passed through a 4^{th} order MA prediction that is identical to the one used at the encoder in order to retrieve the LSP coefficients. The final step is to convert the LSP to LPC using the

relation that exists between the two representations [36]. These LPC are then used to filter the overall excitation through the LP synthesis filter. Note that a 10^{th} order LP synthesis filtering is used. Finally, an adaptive post-filter is used to enhance the quality of the speech. The output of the post filter is the speech that is in 16-bit PCM format. Further details on the decoding performed in G.729A are provided in Appendix A.

The G.729A algorithm has a standardized way to deal with the eventuality of packet losses. Whenever the decoder determines that a packet has been lost, the following steps are initiated to conceal the error:

- LPC are set to their respective values from the last properly received frame
- codebook gains are attenuated from the previously received values
- for the adaptive vector v(n), use the integer part of the pitch delay from previously received good frame (add 1 to it for each successive sub-frame that is lost)
- the fixed codebook vector c(n) is chosen randomly

3.6 Summary

In this chapter, the general properties of a speech signal were discussed. It was seen that speech can be characterized as being either voiced or unvoiced depending on the method used to produce the speech and on the properties of the speech signal. We then introduced the three different families of speech compression and encoding. The first is waveform-based compression where the analog waveform is encoded digitally with the

least degradation possible. It was shown that sampling and quantization need to be done in order to encode the analog signal into a digital form. Vector quantization, even though not usually used in waveform-based compression, was introduced in this section. The standard for conventional telephony and a member of the waveform-based compression family, Pulse Code Modulation (PCM), was introduced. PCM has the advantages of simplicity and good speech quality but requires a high bit rate as compared to other compression methods. The second family of speech compression algorithms, perceptual-based speech compression, was introduced. It was shown that these take advantage of the fact that the human listener cannot always hear all frequencies to encode only what is audible. The central part of these algorithms is the perceptual model since it is responsible of determining the audible component of the speech. A full description of a typical perceptual model was given. Finally, the third family of speech compression algorithm was introduced. It is the family of model-based speech compression. As was shown, these use the model of human speech production in order to reduce the number of bits needed to encode the speech. Instead of encoding the speech sample by sample, parameters of the speech production model are determined and transmitted. A description of Code-Excited Linear Prediction algorithms, a member of the model-based speech compression algorithms, was given. Finally, the ITU standard G.729A, a member of the CELP family, was described in detail since it will be used later in this thesis. Now that a good understanding of speech compression has been obtained, we move to determine which ones are more appropriate for Voice over IP.

4 Speech Compression for Voice over IP

4.1 Overview

In this chapter, we will examine topics of interest in the use of speech compression for Voice over IP. While it is important to have a good understanding of speech compression, it is also important to examine how the speech compression algorithm will perform in a Voice over IP system. This chapter is a combination of literature review and new work on the subject.

We start by examining the desired characteristics of a speech compression algorithm to be used in a Voice over IP system. It will be shown that the optimal speech compression algorithm for VoIP needs to have low delay, a good recovery mechanism from packet loss, good speech quality, low complexity and a low bandwidth requirement. This will be followed by a look at how the three different families of speech compression (waveform-based, perceptual-based and model-based) measure up to these desired characteristics. Most of the discussion will be based on the author's observations on the different speech compression standards. Next, we will consider different international standards and see how they measure up to these desired characteristics. We then discuss the different methods that can be used to recover from a packet loss. Since recovery from packet loss is not standardized in most international standards, it is important to examine how this can be achieved given that packet loss occurs quite often in a Voice over IP environment.

This will be followed by a discussion of how speech frames are packetized into VoIP packets. This again is not standardized and the way we packetize speech can affect the performance of the Voice over IP system. Next, we will look at how we evaluate the speech quality. Since good speech quality is a required characteristic of any speech compression algorithm for Voice over IP, we must specify how this quality is being assessed. In this section, we will review the different methods that already exist to evaluate speech quality and we will present a novel method to evaluate the perceptual quality of the speech that will be used to evaluate the results in this thesis. Finally, the chapter concludes with a summary.

4.2 Desired Characteristics of a VoIP Speech Compression Algorithm

4.2.1 Overview

Before we choose an algorithm, we must determine the algorithm's desired characteristics that we are looking for. In our case, we need to select a speech compression algorithm for Voice over IP. Before choosing the one to use, we will discuss the critical parameters that determine its performance.

As mentioned earlier, a good VoIP speech compression algorithm has to have: low delay, good recovery mechanism from packet loss, good speech quality, low complexity and

low bandwidth [33]. Each of these characteristics will be discussed in more details in the following paragraphs.

4.2.2 Low Delay

As discussed in section 2.5.2, people generally cannot tolerate high delay in a telephony system since it prevents them from communicating efficiently. Since we have seen that delay in a typical Voice over IP system is quite significant, it is important to keep the delay introduced by the speech compression algorithm as low as possible. While other sources of delay such as the IP network are out of our control, the delay introduced by the speech compression algorithm can be kept to a minimum by a good choice of the algorithm.

4.2.3 Good Recovery from Packet Loss

As discussed in section 2.5.4, packet loss is a major impairment that is unique to a Voice over IP system. Since most speech compression algorithms where not designed to deal with packet loss, it is important to evaluate how well such algorithms perform when we introduce a lost packet recovery mechanism. Different methods to recover from a packet loss will be discussed in section 4.5. What it is important to note is that not all speech compression algorithms can recover as easily from a packet loss because of the nature of the different algorithms. This ease of recovering from a packet loss is an important factor when we decide which speech compression algorithm to use for Voice over IP.

4.2.4 Good Speech Quality

The most obvious requirement for a speech compression algorithm to be used for Voice over IP is that it delivers good speech quality. Since most people are used to the good toll quality speech delivered by the PSTN, they will not be satisfied with any speech that does not measure up to this quality. In the age of CD-quality sound, people might also favor speech that is of higher quality than that delivered by the PSTN. A good question is how do we evaluate the quality of the speech delivered by a speech compression algorithm? The answer to this question will be given in section 4.7 where we examine different methods to evaluate speech quality. Until then, what must be remembered is that toll quality speech is the minimum speech quality that is acceptable for VoIP and that anything beyond that would be a real bonus.

4.2.5 Low Complexity

Even though we have CPUs and DSP chips with very high processing power, low complexity still remains a desired characteristic. The reason is that the most powerful CPUs or DSP chips cost more money. If we use a speech compression algorithm that is less complex, we can either use a less expensive CPU or DSP chip to implement it or we can implement more on the same chip. In other words, low complexity is a desired characteristic for a speech compression algorithm since it results in a lower cost of the Voice over IP system.

4.2.6 Low Bandwidth

Even though the bandwidth available on current telephone and data networks keeps on increasing, using less bandwidth is still a desired property of a speech compression algorithm for Voice over IP. The reason here again is money since an algorithm that consumes less bandwidth permits us to multiplex more lines on the same wire. This results in better utilization of the existing resources and more profits to the telecommunication companies. So even though more bandwidth is increasingly available, it comes at a high price.

4.3 Advantages and Disadvantages of Different Speech Compression Families

4.3.1 Overview

Having reviewed what the desired characteristics for a speech compression algorithm in a Voice over IP system are, we will now see how the different families of speech compression fare with these characteristics. The families of speech compression algorithms considered here were introduced in section 3: waveform-based speech compression, perceptual-based speech compression and model-based speech compression. The result of this comparison is highlighted in Table 4.1 and is discussed further in the paragraphs that follow.

Table 4.1: Table of Comparison for Different Speech Compression Families

Characteristic	Waveform-based	Perceptual-based	Model-based
Delay	very low	high	low
Packet loss recovery	fair	fair	very good
Speech Quality	toll quality	toll quality or better	toll quality (CELP), fair (vocoder)
Complexity	low	medium	high
Bandwidth Required	high (PCM), medium (ADPCM)	medium	low
Other	no memory problems	permits to use CD quality speech	does not reproduce telephone tones and music well, problem of tandem connections

4.3.2 Waveform-based Speech Compression

The main advantages of using waveform-based compression such as PCM is that it is very simple and produces good speech quality. The simplicity translates into an algorithm of low complexity that can be run easily on a CPU and does not require much memory. As well, the simplicity translates into negligible delay since each sample is ready to be transmitted as soon as it is sampled and quantized. The speech quality of

PCM defines what we expect out of toll-quality speech since it is the standard that we use currently for telephony. All other members of the waveform-based speech compression family also offer similar speech quality. A final advantage of PCM in particular is that it does not necessitate any memory. Since each sample is encoded independently, a bit error will only affect that particular speech sample. As well, the packet loss will cause degradation only to the speech samples that were lost. This differs from algorithms with memory where packet loss will degrade the speech quality for a longer duration due to corrupted memory.

The main disadvantage of waveform-based speech compression is its high bandwidth requirement. As seen previously, PCM requires a bit-rate of 64 kb/s, which is quite high as compared to other speech compression families. Using ADPCM permits us to reduce the required bit rate at the cost of higher complexity, but the reduction in bit rate does not compare to other methods of speech compression. Another disadvantage is that waveform-based speech compression algorithms do not by nature deal with packet loss well. The reason is that since they are inherently quite simple and have very limited memory, not many options exist to cover-up the packet loss without increasing significantly the complexity of the algorithm.

4.3.3 Perceptual-based Speech Compression

The main advantage of using a perceptual-based speech compression algorithm is the high quality speech that results. Since the algorithms use a perceptual model, the

difference between the original speech and the compressed speech is generally not audible. Another advantage is that these perceptual compression algorithms can easily encode music or sounds and can provide CD-quality speech. One final advantage is that the variable bit rate of the algorithms permits an easy way to implement silence suppression. Since each frame is always encoded with the number of bits that is required to faithfully reproduce the sound, periods of silence can be encoded with very few bits.

The main disadvantage of using perceptual-based speech compression algorithms for Voice over IP is the high delay that they require. Since good perceptual compression requires big frame sizes, it is not rare for these algorithms to require over 100 ms of delay. Another disadvantage is that they are relatively complex. Implementing a good perceptual model is not simple and uses considerable processor time. A final disadvantage is that the compression obtained for telephone bandwidth speech is not that considerable. Since, as was seen in section 3.4.4, most of the masking occurs in the higher frequency, not much compression is obtained when we consider only the frequencies up to 3.4 kHz.

4.3.4 Model-based Speech Compression

The main advantage of model-based speech compression is their low bandwidth requirement. This is by far the family of speech compression algorithms that obtain the highest compression ratio as compared to PCM. Another advantage is the low delay requirement. Most of these algorithms use a small frame size due to the non-stationary of

speech that results in low delay. Finally, a great advantage for model-based speech compression is that they can easily cover-up a packet loss due to the predictive nature of the algorithm. Since the algorithm extracts parameters of human speech production and since these parameters tend to be quite predictable from one frame to another, a packet loss can easily be covered up.

The main disadvantage of model-based speech compression is their high complexity. Because of the complex algorithms and of the lengthy search for the optimal parameters, implementing such algorithms in real time requires considerable processing power. The other disadvantage comes from the fact that model-based speech compression algorithms were designed only to compress speech. This means that these algorithms are not that good at encoding background noise, music, fax signals and telephone (DTMF) tones [33]. Methods do exist to detect fax and DTMF tones and encode them separately, but this results in higher complexity when implementing the speech compression algorithm. A final disadvantage is that these algorithms do not fare well with tandem connections. Tandem connections occur quite often in a VoIP system where the speech is compressed and decompressed many times before it arrives to its destination. Since model-based speech compression algorithms do not encode the speech waveform directly, much speech quality is lost in such tandem connections (it is the common problem of making a copy from a copy...).

4.4 Characteristics of International Standards

Having discussed of the characteristics of the speech compression families in general, we will now consider in more detail the characteristics of some existing international standards. These characteristics are summarized in Table 4.2 [6][10][67].

Table 4.2: Characteristics of International Standards

Standard	Compression Family	Bit rate [kb/s]	Algorithmic Delay [ms]
G.711 (PCM)	Waveform	64, 56 or 48	0
G.722 (SB-ADPCM)	Waveform	64, 56 or 48	0.125
G.726 (ADPCM)	Waveform	16, 24, 32 or 40	0.125
MPEG-2 AAC (for sampling at 8 kHz)	Perceptual	variable up to a maximum of 48	256
G.723.1 (MPC-MLQ)	Model	5.3 or 6.4	37.5
G.729 (CS-ACELP)	Model	8	15
G.729A (CS-ACELP)	Model	8	15

As can be seen from Table 4.2, model-based speech compression algorithms do indeed obtain the highest compression ratio and have a delay that is quite low. Because of this and what was discussed in the previous section, model-based speech compression algorithms are the most promising speech compression algorithms for Voice over IP [37].

Note that the table given so far did not include the complexity of the algorithms. This is because for many of these algorithms, no standard code exists and the complexity varies

from one implementation to the next. For the three most popular model-based speech compression algorithms, the source code for their implementation has been standardized. For this reason, it is possible to evaluate the complexity of these algorithms. This complexity is illustrated in Table 4.3 [10].

Table 4.3: Complexity of Three Model-based Speech Compression Standards

Attribute	**G.723.1**	**G.729**	**G.729A**
MIPS required	16	20	10.5
RAM requirement	2200 words	3000 words	2000 words

As can be seen from Table 4.3, the international standard for model-based speech compression standard that requires less DSP processor time (indicated by the lower amount in millions of instructions per second (MIPS)) and the lower memory is G.729A. Since this standard also has comparable speech quality to the other two and lower delay than G.723.1, it is quite an attractive speech algorithm for implementing a Voice over IP system. For this reason, this is the speech algorithm that is the subject of this thesis and will be examined further in chapter 5.

4.5 Recovery from Packet Loss

4.5.1 Overview

As discussed previously, one of the most important impairments in a Voice over IP system is packet loss. How the speech compression algorithm deals with packet loss is very important since this will be reflected in the quality of the speech during the periods of packet loss. Being able to hide the packet loss from the user is quite important since degradation due to packet loss will not be acceptable to a user who is used to the PSTN where no such impairments exist.

Methods to recover from packet loss can be divided into two broad categories: sender-based recovery and receiver-based recovery [48]. These will be discussed in the next two sub-sections through a literature review of the subject. Note that all of the methods reviewed here only deal with replacing the packet that was lost and not on minimizing the error following the packet loss. As will be discussed more fully in section 5.2, when replacing a lost frame in a model-based speech compression algorithm, we must also consider the effect it will have on system memory since the error will cause the transmitter and receiver to no longer be synchronized [36]. This is not considered in this section, but will be covered through the results of the thesis.

4.5.2 Sender-based Recovery

Sender-based recovery methods are all methods to cover-up a packet loss that are initiated by the transmitter of the data. Methods to do this range from retransmission, interleaving and forward error correction (FEC) and are illustrated in Figure 4.1 [48].

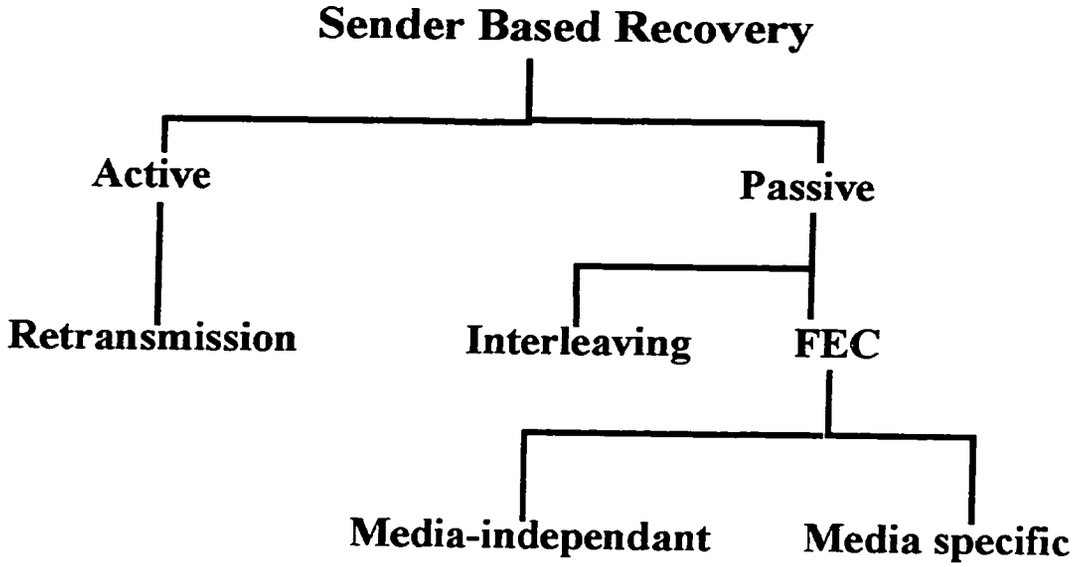

Figure 4.1: Methods for Sender-based packet loss recovery

The most common method of packet loss recovery on a data network is retransmission. The receiver detects that a packet was lost and requests from the transmitter another copy. This works well for data which do not have a stringent delay requirement, but it is unacceptable for Voice over IP since we cannot afford to wait for retransmission. Since the round-trip delay over an IP network such as the Internet can be over 200 ms, such a scheme would add too much delay to the system.

Another possible way to minimize the effect of packet loss is interleaving. This consists of changing the order of transmission of the packets from its original order to protect

against burst packet loss. An example of such a scheme for VoIP is given in [74] for waveform-based compression where even and odd samples are sent in alternate packets. If one of the packet is lost, recovery can be obtained through averaging. The problem with such a scheme is the added delay interleaving introduces. Since the receiver must wait longer to receive the packet that must be played next, this scheme is not appropriate for Voice over IP.

The only remaining alternative for Voice over IP is forward error correction (FEC). This is further divided into a media independent FEC and a media specific FEC. Media independent FEC uses standard methods from error control coding theory [76] to add parity bits to protect the data from packet loss. The only difference from conventional error control coding is that the codewords are formed from bits of different packets instead of bits from the same packet. For example, suppose we want to protect our system against any isolated packet loss within a group of 4 packets. We could then use a (7,4) BCH code since this code can correct up to 1 bit error [76]. To implement it, we would apply this code to bit #1 of packets 1, 2, 3 and 4, then to bit#2 of packets 1, 2, 3 and 4, and so on. The parity bits resulting from this code, of an equivalent size of 3 data packets, would be sent as a separate packet following the data. At the decoder, we can recover from the loss of any one of these packets by correcting the bit error in every one of the bit positions of the packet using the BCH code. One of the costs of such a system is the extra bit rate since for each 4 bits transmitted, we need to transmit 3 extra parity bits. The other cost is the extra delay that such a scheme adds. Assume packet #1 is lost.

The decoder will have to wait until it receives packet #4 before it is able to correct this packet loss since the parity bits cannot be generated until all four packets have been transmitted. Such added delay is not acceptable in a Voice over IP system. An alternative way that does not add as much delay is media specific FEC. In such a scheme, a redundant version of a packet is sent with the preceding or subsequent packet [4][5][51]. For example, we might send with packet #10 another copy of packet #9. This would permit us to recover from losing packet #9 at the cost of additional bits and delay (we have to wait until the next packet arrives). This redundant copy of the packet can either be encoded in the same way as the original packet or using another speech compression algorithm of lesser speech quality and bit rate. For example, if the packets are originally encoded as PCM, the redundant version could be encoded in G.729A format. The cost of such a scheme can be reduced further if it is only used in periods of high packet loss, as described in [4].

It is the author's opinion that the only sender-based recovery method that should be considered for Voice over IP is media-specific forward error correction (FEC). All other methods need to add too much delay to be as effective, which is unacceptable in a Voice over IP system. While media-specific FEC also adds some delay when a packet loss occurs, it is limited to one frame, which in most cases would be acceptable and preferable to losing the packet entirely. The proposition to use more than one speech compression algorithm though is, in our opinion, not acceptable for a Voice over IP system as it would add too much complexity to the system. Instead, we recommend only sending the

parameters that are the most important to the speech quality instead of sending a full redundant frame. In this way, the complexity is not increased and the recovery of the missing speech packet is still quite improved. This method will be discussed in more details in section 5.4.3 of this thesis.

4.5.3 Receiver-based Recovery

Another method to cover up a packet loss is to use receiver-based recovery (sometimes also called error-concealment). In such schemes, the receiver covers up the packet loss without the intervention of the transmitter. Obviously, such schemes are not as powerful as sender-based recovery since the receiver can only do so much too cover up the packet loss since it has no knowledge of the original data. For example, the receiver cannot conceal any packet loss that spans more than a syllable since it is not capable of predicting what the person will say. Because of this fact, receiver-based packet loss recovery is usually used in tandem with sender-based packet loss recovery in order to cover up the packets loss that were not corrected by the other scheme [48]. Different methods to perform receiver-based packet loss recovery are illustrated in Figure 4.2 [48].

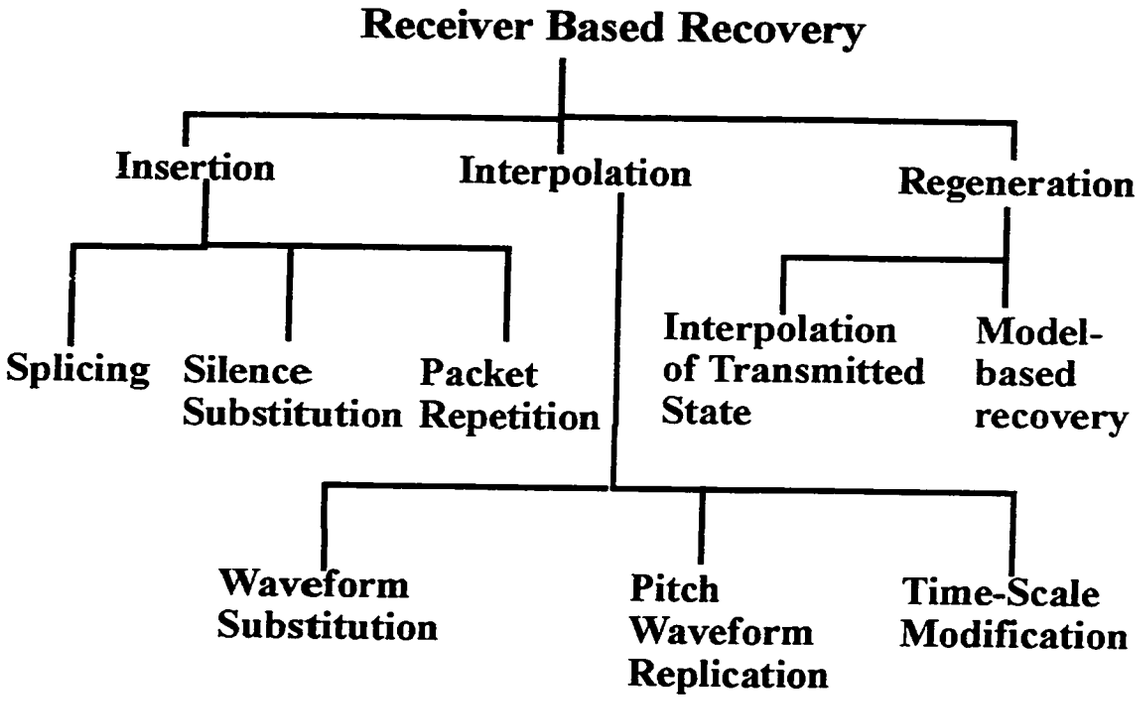

Figure 4.2: Methods for receiver-based packet loss recovery

Insertion-based methods are the simplest ways to perform receiver-based packet loss recovery. They include splicing, silence/noise substitution and packet repetition. Splicing consists of removing the gap left by the lost packet. For example, if we lose packet #4, play packet #5 in its place and so on. Obviously, this is not a very good method since it disturbs the original timing of the speech. Silence/noise substitution consists of substituting the packet that was lost with either silence or noise. Since the human ear is more sensitive to silence, noise substitution gives better performance than silence substitution [48]. Packet repetition consists of replacing the lost packet with the last packet that was correctly received. For example, if we lose packet #4, we replace it with packet #3 since this one was properly received. This scheme can be improved slightly by adding fading, whereas the values of the packet that is repeated are reduced in

intensity. This turns out to be an easy method that gives acceptable results. In terms of cost-effectiveness, packet repetition with fading is the receiver-based lost packet recovery method that is the best amongst those that can be applied to all speech compression algorithms [48]. Other methods that will be discussed next give a slightly better performance but are considerably more complex.

Another category of receiver-based packet loss recovery methods are those that use interpolation. Once again, these methods are quite general and can be used for any speech compression algorithms. One such method is waveform substitution [75], where the speech waveform that precedes and/or follows the lost frame is used to interpolate the values during the period of loss. These methods take advantage of the fact that speech is a continuous signal and that the interpolated waveform should not have any discontinuities. This method can be used more effectively if we use it in conjunction with a sender-based packet loss recovery mechanism as is done in [13]. In this case, the transmitter sends the energy of and the number of zero crossings in a packet of speech with the packet that precedes it. This results in better performance of the waveform substitution method. Another method in the category of interpolation is pitch waveform replication. In this method, a pitch detection algorithm must be implemented if one is not already implemented in the speech compression algorithm and a waveform with the predicted pitch value is used to cover up the packet during periods of voiced speech. A final method in the category of interpolation is time scale modification. In such a scheme, the speech waveform before and after the packet loss is stretched to cover up the

loss. In general, interpolation methods are quite complex and result in only slightly better performance than packet repetition.

The final category of receiver based packet recovery methods is regeneration schemes. These schemes use knowledge of the speech compression algorithm to derive the best values of the parameters for the lost packet. Regeneration methods give good results but are specific to a particular speech compression algorithm. For example, the method described in section 3.5.5 to cover up a packet loss in G.729A is a regeneration method that uses interpolation of transmitted state. Obviously, such a scheme would not work for any other speech compression algorithm since it depends on the parameters of the speech compression algorithm. These schemes work well for algorithms that have a sufficiently large number of states and are already quite complex such as model-based speech compression algorithms. For these algorithms, the regeneration method is the best choice since it results in little additional complexity and excellent performance. However, if we are dealing with a simple algorithm such as PCM, regeneration methods are possible but add much complexity to the algorithm. Such an example can be found in [9] where a model-based recovery is used to cover up packet loss in PCM. This method works quite well but requires an auto-regressive analysis and an estimation of the excitation for the loss period. Essentially, all this extra analysis required to cover up a packet loss in PCM renders it almost as complex as a model-based speech compression algorithm. Obviously, if in principle we accept adding such complexity, we might as well use a model-based speech compression algorithm!

In conclusion, receiver-based packet loss recovery methods are quite essential in speech compression for Voice over IP. For low rates of packet loss, such a scheme might be sufficient. As packet loss increases, a sender-based recovery method should be used to complement it. We have seen that for waveform-based speech compression algorithms, packet repetition with fading is the most cost-effective way to cover up the packet loss. Other methods work slightly better but come at a price of considerable more complexity. For model-based speech compression algorithms, regeneration based methods are preferred. These must be developed specifically for each compression algorithm but give great results at almost no extra complexity. They also take advantage of the complexity that already exists in the algorithm to cover up packet losses quite well.

4.6 Packetization of Speech Frames

Even though speech compression algorithms are based on speech frames, we must not forget that we will need to send the speech frame in a Voice over IP system as packets. How we packetize the speech frames into packets makes a difference in terms of delay in the system and in terms of bit rate efficiency.

Recall that a certain overhead exists when we are transmitting a packet. This overhead, as was discussed in section 2.2.3, is due to the header information that must accompany any data transmitted over a data network. As was seen in Figure 2.3, this header

information has a minimum size of 320 bits for a typical VoIP speech packet. Since the size of a typical speech frame is usually quite small, the additional bits required by the header information represents a considerable overhead. This is the case for the example of ITU G.729A that was examined in detail in section 3.5.5. Each speech frame for this algorithm has 80 bits. If we want to packetize one speech frame per VoIP packet, each packet would take 400 bits where 80% of which is occupied by the header. Obviously, such a scheme is quite wasteful on valuable bandwidth. The other alternative is to packetize many speech frames into one VoIP packet. The problem with this alternative is that this will require adding considerable delay (the delay increases with each extra speech frame we packetize) which as we have already discussed is unacceptable. Taking as an example G.729A again, packetizing 5 speech frames together would result in a VoIP packet of 720 bits where 44% of which is occupied by the header. The cost of reducing the header overhead is extra delay. In this case, since we need to wait for 5 speech frames before we send a packet and that each speech frame represents 10 ms of speech, we have now increased the delay from 15 ms to 55 ms. Since delay must be limited in all Voice over IP applications, packetizing many speech frames into one VoIP packet is not acceptable. Another reason packetizing many speech frames into one VoIP packet is not recommended is that the error concealment scheme would be less effective in such a case since one packet loss would result in the loss of many consecutive speech frames. As was discussed briefly in section 4.5, recovering from the loss of many consecutive speech frames is harder than recovering from isolated loss since it is much

more likely in such a case that a whole syllable of the speech has been lost. If this occurs, all error concealment methods perform badly since they cannot predict what was spoken.

So far we have seen that packetizing one speech frame per VoIP packet results in high header overhead while packetizing many speech frames into each VoIP packets results in high delay and poor error concealment . As discussed previously in section 4.2, keeping the delay low and recovering well from packet loss is essential in VoIP. For these reasons, packetizing one speech frame per VoIP packet resulting in high header overhead but no extra delay is favored by the author. Even though some articles do favor packetizing many speech frames in one VoIP packet to reduce the overhead [50], we believe the added delay and the poor error concealment resulting from such a scheme is unacceptable. As a result, we will assume that each VoIP packet contains only one speech frame in this thesis.

Even though we have limited ourselves to packetizing one speech frame per VoIP packet, the header overhead can still be reduced without introducing any extra delay. This can be achieved by sending other data in the packet other than the speech frame. This could be unrelated Internet data or data that the two people conversing are exchanging between them. This extra data can occupy the space available in the packet and result in a system where the header overhead is more acceptable. An alternative way to reduce the header overhead can be achieved using the system illustrated in Figure 4.3. Assume that four telephone calls occur simultaneously where the origin and destination of the calls share

the same Voice over IP gateways. Note that these telephone calls do not need to be from the same location since many users can share the same VoIP gateway. In this scheme, the header overhead can be reduced by packetizing all four telephone calls onto the same VoIP packet. This scheme adds minimal delay (some minimal delay less or equal to the algorithmic delay is required since not all sources produce their speech packets at the same time) and does not affect the efficiency of the error concealment method (even though the packet has multiple speech frames, we do not have any consecutive speech frames from the same source).

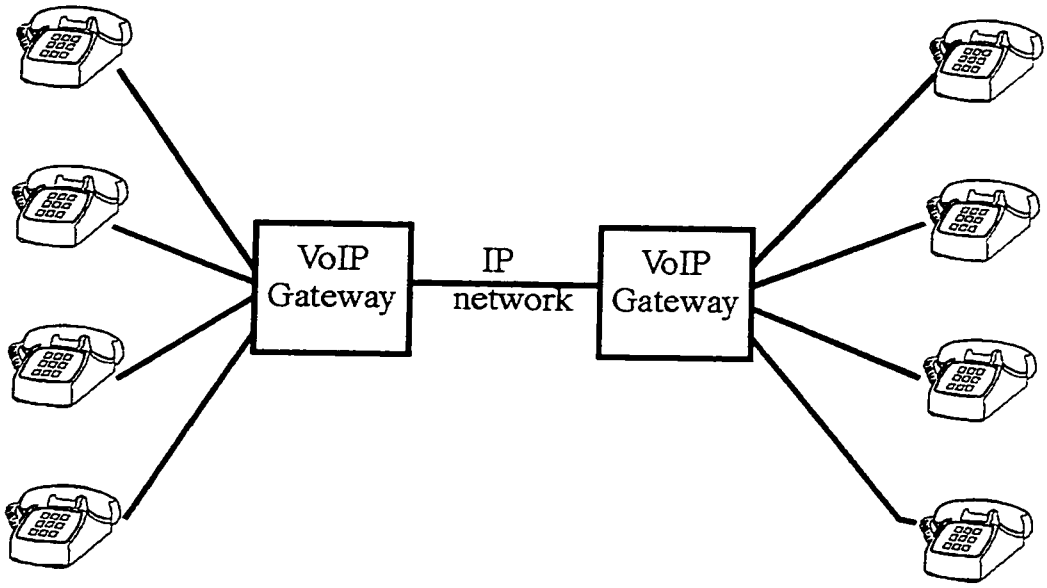

Figure 4.3: Example Voice over IP system

This scheme can also be extended to cases where the origin and the destination of the calls differ, but a portion of their path is the same. In the common portion of the IP path, the speech frames can be packetized together to reduce the header overhead. An implementation of such a scheme is proposed in [23]. In this implementation, some multiplexing delay is added at the VoIP gateway in order to increase the number of

speech frames that are multiplexed together. In such an implementation, there is a tradeoff between the multiplexing delay and the efficiency of the system.

4.7 Evaluation of Speech Quality

4.7.1 Overview

The evaluation of the subjective quality of speech is a topic of great importance in speech compression for Voice over IP as it is an important factor when users choose a telephone system. As well, evaluation of speech quality is an important part of non-intrusive monitoring of voice quality [1]. This is a mechanism by which the quality of the calls over a particular network is evaluated by the service provider in order to determine if the level of service provided is sufficient. To be able to do this efficiently, objective methods of evaluating speech quality must exist.

This section will start by reviewing the different methods that currently exist to evaluate speech quality. These can generally by subdivided into subjective tests and objective tests. Subjective tests generally require the interaction of humans to evaluate the quality of the speech. Standards for these tests will be discussed in the next sub-section. Objective tests on the other hand do not require the intervention of a human and are usually performed by a computer. Methods of objective tests on speech quality will be discussed following the discussion of subjective tests.

Since many of these methods are lacking and hard to implement, this section will finish with the introduction of a new system for objective analysis of "subjective" speech quality. As will be seen, this method is quite intuitive and effective. Since this system will be used to evaluate the speech quality in the remaining of this thesis, it is important to present it in detail.

4.7.2 Subjective Tests

Subjective tests rely on the fact that since a human will be the one using the telephony system, then a human should be evaluating its quality. Subjective testing is usually the preferred method to evaluate speech quality since not many effective objective methods are currently available and accepted. The main problem with such tests is that they are very time consuming and expensive to perform as they involve testing by human subjects. An additional problem is that due to their subjective nature, the results are impossible to reproduce and they can vary depending on the listeners.

The international standard for subjective speech quality assessment is ITU-T P.800 [29]. Another standard, ITU-T P.830 [30], specifies how these subjective tests should be performed. Several subjective tests are described in this standard, however the most accepted one is based on an Absolute Category Rating (ACR) [49]. In such a test, each subject listens to a list of short speech files and attributes to it a score between 1 and 5 where: Excellent = 5; Good = 4; Fair = 3; Poor = 2; Bad = 1 [29]. After the speech file has been evaluated by numerous listeners, the score attributed to the speech file is

averaged out to provide a Mean Opinion Score (MOS). The higher the value of the MOS, the better is the quality of the speech. The exact testing procedure and the recording environment for the speech files is stringently regulated in the standard. Since these details are not relevant to this thesis, the interested reader is referred to [29] and [30].

Another method of subjective testing is based on the Comparison Category Rating (CCR) [29][49]. This method is similar to ACR except that here the quality of the speech file is compared to a reference speech file. For each speech file, the subject listens to the reference speech and the compressed or modified one. The order in which they are presented is random in order to remove any bias from the experiment. The score assigned compares whether the second version of the speech file is better or worse than the first version. The score given can be: (3) much better, (2) better, (1) slightly better, (0) about the same, (-1) slightly worse, (-2) worse, (-3) much worse. Once each listener has attributed a score to each speech file, the scores are averaged out to give a Comparison Mean Opinion Score (CMOS). A positive value of CMOS indicates that the modified or compressed version is better while a negative value of CMOS indicates that the reference version is better. Even though CCR seems to be a better method than ACR to judge the quality of compressed speech, some tests done using the CCR did not give good results [49], which makes people more apt to use ACR.

4.7.3 Objective Tests

Even though subjective tests are the current practice to evaluate speech quality, methods of objective tests have ignited much interest because of their inherent advantages over a subjective test. The main advantage is that objective tests cost a lot less to perform and can easily be done on a large scale since they do not require human intervention. In subjective test, performing the test requires following stringent psychological test procedures and requires the involvement of many people. As well, due to the subjective nature of the tests, their validity sometimes comes in question since test results can vary with listeners and are hard to compare. Objective tests have the advantage that they can be done easily and quickly by a computer, they do not require much human intervention, and the results are reproducible.

The simplest objective test is to calculate the Mean Square Error (MSE) or the Signal to Noise Ratio (SNR) between the compressed speech and the original speech. This method simply takes the difference between speech samples, squares the difference and averages it out over many different test cases. The idea here is that a compressed speech file that is closer to the original speech file will sound better. Recalling the discussion in section 3.4, one can easily see that such a proposition is invalid. Since the human ear cannot hear all elements of a speech signal, simply looking at the difference of the speech file is not sufficient since this difference may not be audible to the listener. Because of this, the MSE is generally not a good indication of speech quality, but it can still be used in some isolated cases. An example is to compare two speech files where the error occurred at the

same location in the speech file. Since we are considering the same location, a smaller error obviously means that this error is less likely to be audible. Note though that we still did not determine that either one of the two errors is audible. In this thesis, the square error will sometimes be used to compare performance when errors occur in the same location since it is a simple method that is easy to visualize. Note however that in general, such a method is not acceptable to evaluate the quality of a complete speech file.

The most recent methods of objective speech quality assessment take into account some of the properties of the human ear. A good example is the ITU standard Perceptual Speech Quality Measure (PSQM) standardized in ITU-T P.861 [31]. This method tries to mimic the sound perception of subjects in real-life situations through the faithful representation of the human perception and judgement process. It is a computer program that takes as input the original speech file and the compressed speech file. It attributes a score to the speech file on a scale of 0 (ideal) to 6.5 (very poor) according to how degraded the compressed speech is as compared to the original. To calculate this score, it uses time-frequency mapping, frequency and intensity warping and a cognitive model [31][2]. Even though this method sounds impressing, its effectiveness for speech quality assessment has still not been fully evaluated. In the case of evaluating the quality of speech following packet loss, the standard indicates that insufficient information is available about the accuracy of PSQM in such a situation [31]. A free implementation of PSQM obtained from Lucent at [24] was evaluated but the tool was not acceptable in that the results did not agree with the speech quality of what was actually being heard. Even

though this is probably only a problem with this particular implementation (it had difficulty aligning the reference and degraded speech files), PSQM was not used for this thesis since this was the only implementation of the algorithm that could be obtained with the author's budget. Another existing objective measure to evaluate speech quality is the modified bark spectral distortion (MBSD) measure [77]. This measure is similar to PSQM except that it takes into account noise masking (same as simultaneous masking that was discussed in section 3.4.4.2) whereas PSQM does not. But since this is not an international standard, no implementations of this algorithm were available. Because of this, there was a clear need for a new objective test of the subjective or audible speech quality.

4.7.4 A Novel Objective Method to Evaluate Perceptual Speech Quality

In this section, we describe a novel system to objectively estimate speech quality. This system, shown in Figure 4.4, is based on a perceptual model and will be used to evaluate the speech quality of the results in this thesis.

The system starts by using a perceptual model that takes as input a block of 2048 speech samples and outputs the masking threshold for this block of speech. This is done for the original speech file (reference) and the degraded speech file whose quality is to be determined. The frequency content (FFT) of the block of speech is also calculated. We then subtract the masking threshold from the frequency content. Given that a tone is not audible if it is below the masking threshold at its frequency, we know that if this

subtraction results in a negative value, the tone in the block of speech is not audible. As well, the amount by which our tone is above the masking threshold tells us how audible that particular tone is. We then set to zero all negative values from the subtraction and what is left is a frequency representation of the audible tones in the block of speech. As well, the intensity of each tone is directly proportional to how audible it is in the block of speech. Having done this for both speech files, we then have a good representation of what is audible in each speech file. To determine the perceptual difference between the two, we need to take the absolute difference between the two frequency representations of the block of speech (original and degraded). For two speech files that sound exactly the same, this difference should be zero for all frequencies. For speech files that do not sound the same, the difference will be non-zero at the frequencies where they differ.

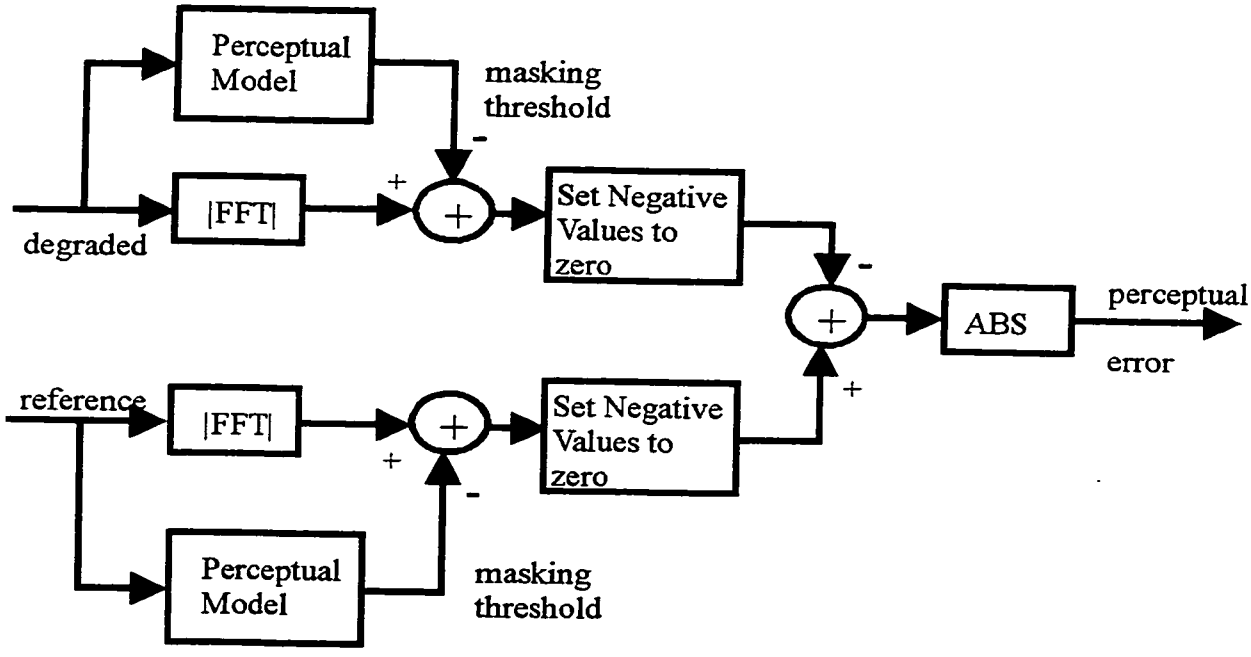

Figure 4.4: Block Diagram of my Perceptual Error Method

This is repeated for each block of the speech file. Two metrics are extracted from the output of the system to present a compact measure of the audible difference. The first output is the sum of all of the perceptual errors for all frequencies and blocks of speech divided by the number of speech samples in the speech file (in order to normalize for different sized speech files). This is referred to as Perceptual Error per Sample and is a good indication of the general quality of the speech file being considered. Obviously, a smaller Perceptual Error per Sample indicates a speech file that resembles the original more closely than a higher Perceptual Error per Sample. Note that summing over all frequencies is valid since all frequencies have equal significance once the masking threshold has been removed (the threshold in quiet is already considered in the calculation). The second output considered is the maximum perceptual error encountered amongst all frequencies and blocks of speech. This is referred to as Maximum Perceptual Error and is a good indication of how severe the difference between the two speech files is. A high value of the maximum perceptual error indicates a degradation that is quite audible while a low value indicates a degradation that is barely audible. It is necessary to use this second metric since for isolated degradations that cause audible degradation, the Perceptual Error per Sample will be low (the error is insignificant as compared to the size of the speech file) but the Maximum Perceptual Error will be high. Such a case would otherwise go unnoticed if we would only concentrate on the first metric.

This system was used extensively to objectively verify the results of this thesis in the sections that follow alongside with actual listening by the author. It was proven to be

quite effective since the values that were outputted from this method were always consistent with what was heard. The main purpose of this system is to objectively rate the different degraded versions of a given speech file. For similar conditions, the speech file that has lower values for these two outputs is of better quality than the speech file that has higher values. It is important to note that the values themselves do not have much significance. It is not correct to compare these values between different speech files or conditions. The conclusions based on this system objectively determining the subjective quality of speech were always confirmed by the author.

4.8 Summary

We have seen in this chapter many topics of particular interest to speech compression for Voice over IP. It was first determined that the ideal speech compression algorithm for a Voice over IP environment would need to have low delay, be capable of recovering well from packet losses, have good speech quality, and require low complexity and bandwidth. Through an examination of the different families of speech compression algorithms, it was shown that model-based speech compression algorithms present a family of speech compression algorithms that meet most of these characteristics. We then examined existing standards of speech compression algorithms and found that ITU-T G.729A was a good speech compression algorithm to use for Voice over IP. Next, we considered a topic of high importance for speech compression algorithms for VoIP, recovery from packet loss. It was seen that for waveform-based speech compression algorithms, the

most cost-effective method to cover up a packet loss is packet repetition with fading. It was seen that other methods give slightly better performance but the added complexity was too much to make them worthwhile. As for model-based speech compression algorithms, regeneration methods that take advantage of the details of the speech compression algorithm gave great performance at very little added complexity. It is for this reason that model-based speech compression algorithms recover better from packet loss than other families of speech compression. We then examined how we packetize speech frames into VoIP packets. It was shown that packetizing one speech frame per packet resulted in high header overhead while packetizing many speech frames per packet resulted in high delay and degraded performance of the error concealment. Since delay must be kept low and good error concealment is important in a Voice over IP system, packetizing one speech frame per packet was deemed the most appropriate and it is what is considered in this thesis. Methods to reduce the IP overhead in such cases by transmitting the speech frames with other data and multiplexing many sources together were also presented. Finally, we examined different methods to evaluate the speech quality. It was seen that the subjective tests such as Absolute Category Rating (ACR) are the current practice for evaluating speech quality. Because of the huge amount of time and money needed to perform such tests, alternative objective tests are desired. Existing objective tests either are not well established or do not give a good indication of the speech quality. A novel method to evaluate the speech quality objectively was presented. This method of calculating the perceptual error will be used to evaluate the speech quality of the results in the rest of this thesis.

5 Improving the Performance of ITU-T G.729A in the Presence of Packet Loss

5.1 Overview

After a thorough examination of the desired characteristics for a speech compression algorithm for VoIP in chapter 4, it was determined that ITU-T G.729A was a good speech compression algorithm for VoIP. Its low delay and bit rate, its reduced complexity compared to other model-based compression algorithms, its good speech quality and its good recovery mechanism from packet loss make it an ideal candidate for a speech compression algorithm for VoIP.

The ITU-T G.729A speech compression algorithm was described previously in section 3.5.5. In this description, we were able to note that indeed the algorithm had a low algorithmic delay of 15 ms and a low bit rate of 8 kb/s. In Table 4.3 given in section 4.4, it was shown that G.729A had a complexity of 10.5 MIPS, which is well below the complexity of other international standards for model-based speech compression. The good quality of the compressed speech in ITU-T G.729A is supported by many subjective listening tests such as the one highlighted in [61] and [28]. In these tests, it was determined that the algorithm had a MOS score of 3.92, which is an indication that the algorithm delivers speech of close to toll-quality. The only characteristic that is required of a VoIP speech compression algorithm that has not been sufficiently proven for ITU-T

G.729A is its performance in periods of packet loss. As was seen in section 2.5.4, packet loss is quite frequent in an IP network and can attain a packet loss rate of 25% or higher. It is then essential to use a speech compression algorithm that has good performance in these periods of packet loss. Some published results on the effect of packet loss on the speech quality of ITU-T G.729A are highlighted in [61] but they deal with a packet loss rate of up to 5% only. As is indicated in [49], such results are not sufficient and the performance of ITU-T G.729A for higher packet loss rate should be examined. As well, these results do not investigate the cause of this degradation in periods of packet loss, which we find to be very important in assessing the ability of a speech compression algorithm to recover from packet loss.

In this chapter, we will thoroughly examine the performance of ITU-T G.729A in periods of packet loss. Since this as yet to be done in the literature, this chapter will deal entirely with original work on the subject. This chapter will start by examining the effects of packet loss on G.729A. It will be shown that due to the memory present in the algorithm, the degradation of the speech lasts well beyond the period of packet loss. This degradation beyond the period of packet loss is due to the fact that some memory values at the encoder and the decoder are no longer synchronized, which we will call state error. A thorough examination of the state present in the algorithm will then be given. Rosenberg noted the existence of "states" in G.729A in [60], but this section will expand on his work to determine the exact time needed to recover from this state error and the relative importance of each state on the state error. Once the presence of the state in

G.729A is well understood, we will propose two novel methods for speedy recovery from this state error. The first one, recovery-by-retransmission, works in theory but requires much additional bit rate and is of no practical use. The second one, recovery-by-re-initialization, does not require much additional bit rate and is simple to implement. It will then be shown in results that this novel method to recover from state error does indeed improve the performance of ITU-T G.729A in the presence of packet loss. The results will be given for packet loss rates of 5 to 50 % to properly assess the performance of this method in different IP network conditions. This chapter will then conclude with a summary of what has been discussed in this chapter.

5.2 Effects of Packet Loss

5.2.1 Overview

This section will investigate the effects of packet loss on G.729A. In order to do this, a proper testing environment had to be created. Details of this testing environment, which will also be used throughout this chapter, will be given. Then, the effect of packet loss on G.729A will be explained with the help of a concrete example. To make the explanation as simple as possible, the square error will be used to illustrate the effects of packet loss. It is important to remember from section 4.7.3 that the square error is not a sufficient method for evaluating speech quality. This is why this will be followed by a subjective way to evaluate the effects of packet loss on G.729A.

5.2.2 Testing Environment

The most important part of the testing environment is a proper implementation of the existing algorithm. For this reason, a copy of the ITU-T simulation software for G.729A was obtained. This ANSI-C code is a fixed-point implementation of the algorithm and serves as the benchmark for any implementation of the algorithm. It comes as a supplement to the ITU-T standard for the algorithm in [26]. Since it does not perform the compression in real-time, it is not of much use for any actual implementations but it is still invaluable for doing simulations.

It was necessary to modify the code in order to simulate packet loss. As well, the code was encapsulated in a Windows graphical user interface (GUI) in order to facilitate user interaction for setting the necessary packet loss parameters. Other modifications to the code were later needed to enable testing different aspects of the algorithm. Note that throughout, no modifications were performed to the encoding and decoding sections in order to ensure compatibility with the standard.

5.2.3 Effects of Burst Erasure

With the test environment properly constructed, it was then fairly straightforward to examine the effects of packet loss on the algorithm. While packet loss was simulated many times, the general observations on the effect of packet loss on the algorithm can be illustrated in one example. For this example, a burst of three consecutive packet losses was performed on an important part of the speech. The distortion was clearly audible by

ear, but it was necessary to look at it objectively to better understand the degradation. To do this, the square error was used as an objective measure. The square error was calculated between the decoder output with no packet loss and the decoder output with the burst erasure (the scale of speech samples was limited to +/- 1). Note that we cannot do the comparison between the input and the output of the algorithm since, due *to the nature of the algorithm*, this error will always be large even if no packet loss occurs. Comparing the output with packet loss *to the output without packet loss* permits us to isolate the degradation caused by the packet loss (the degradation caused by the speech compression algorithm is ignored). A plot of this square error relative to the sample number is illustrated in Figure 5.1 (recall that 80 samples form 1 frame).

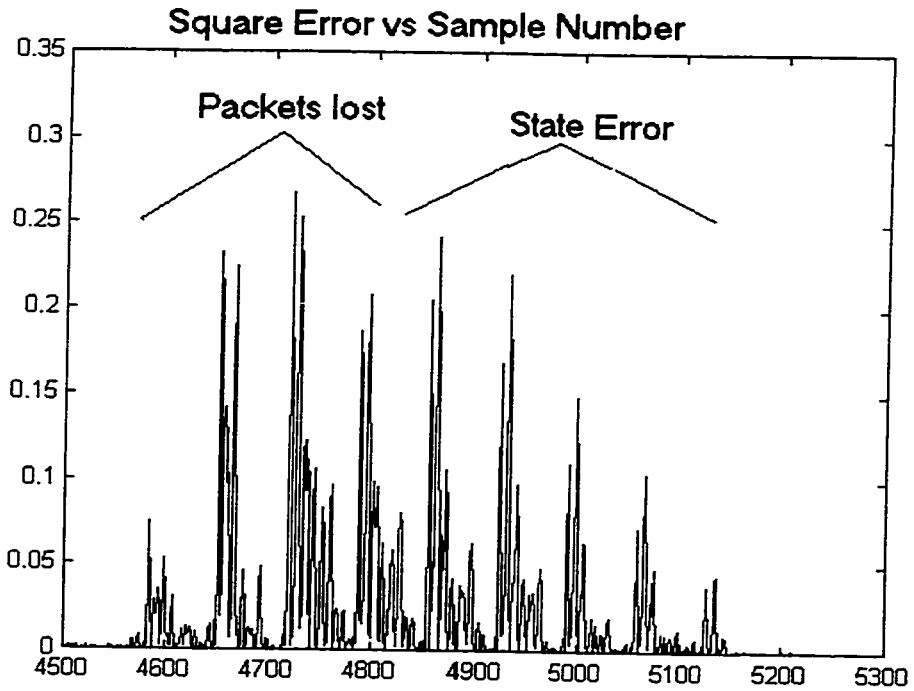

Figure 5.1: Graph of Square Error vs Sample Number for a Burst Frame Erasure

In this graph, we first note that the simulated packet loss occurred from sample #4560-4800. We can see that in this area, the square error is fairly high. This is to be expected since the G.729A algorithm cannot perfectly predict the part of the speech it has not received. What may be more surprising at first glance is that the square error lasts longer than the duration of the packet loss. This is illustrated by the section labeled 'state error' in the figure were we see that some square error is still present from sample #4800-5150. Even after the packets are no longer lost, the receiver still generates an output speech that is different from what it should be.

It was soon realized that the cause of this prolonged period of error is the fact that the states of the encoder and the decoder are no longer synchronized after a period of packet loss. The term "state" in this context refers to all memory that is common to both the encoder and the decoder. Further details on the location of the state in G.729A will be given in section 5.3. Despite the fact that these states are never exchanged between the encoder and the decoder, they have to be synchronized with each other for proper decoding. These states are updated after each frame in relation to the transmitted parameters. In periods where no packet loss occurs, there is no problem since both the encoder and the decoder have the same values for the parameters and thus will update the memories in the same way, assuring that both sides are always synchronized. But when a packet loss occurs, the decoder must guess the intended parameters using the frame erasure concealment scheme described in section 3.5.5. Since these guessed parameters are never exactly the same as what the encoder had transmitted, this means that the

decoder will update its memories using different values of parameters than the one the encoder is using. Because of this, the encoder and the decoder memories will no longer be synchronized even after no more packet loss is occurring. This synchronization between the encoder and decoder is crucial since when the encoder selects its parameters to transmit, it does so using the values in its memory. Thus, even if the decoder accurately receives the parameters, the output speech will still have errors if the decoder does not have the same memory values as the encoder.

Returning to Figure 5.1, we can see that in terms of square error, the deterioration during the period of packet loss and state error are similar. The peak square error at the beginning of the state error section is quite similar in magnitude to the peaks in the packet loss section. Even though the square error in the state error section decays gradually, we can see that it is still considerable and actually lasts longer than the actual packet loss section (lasts for 350 samples compared to 240).

5.2.4 Comparing Effects of Packet Loss to Effects of State Error

Since square error is not necessarily related to speech quality as was described in section 4.7.3, it is necessary to actually hear the speech with packet loss and with state error separately to judge which one is more disturbing. Since the duration of each degradation is fairly short (30 ms for packet loss 44 ms for state error) simply trying to hear each degradation separately was not feasible for the ear.

The best way to judge the impact of the two types of error on speech quality was to generate speech files that only contained one of the two errors. The procedure to generate these two speech files, which is similar to a method used in [60], will be detailed here with the help of the block diagram in Figure 5.2.

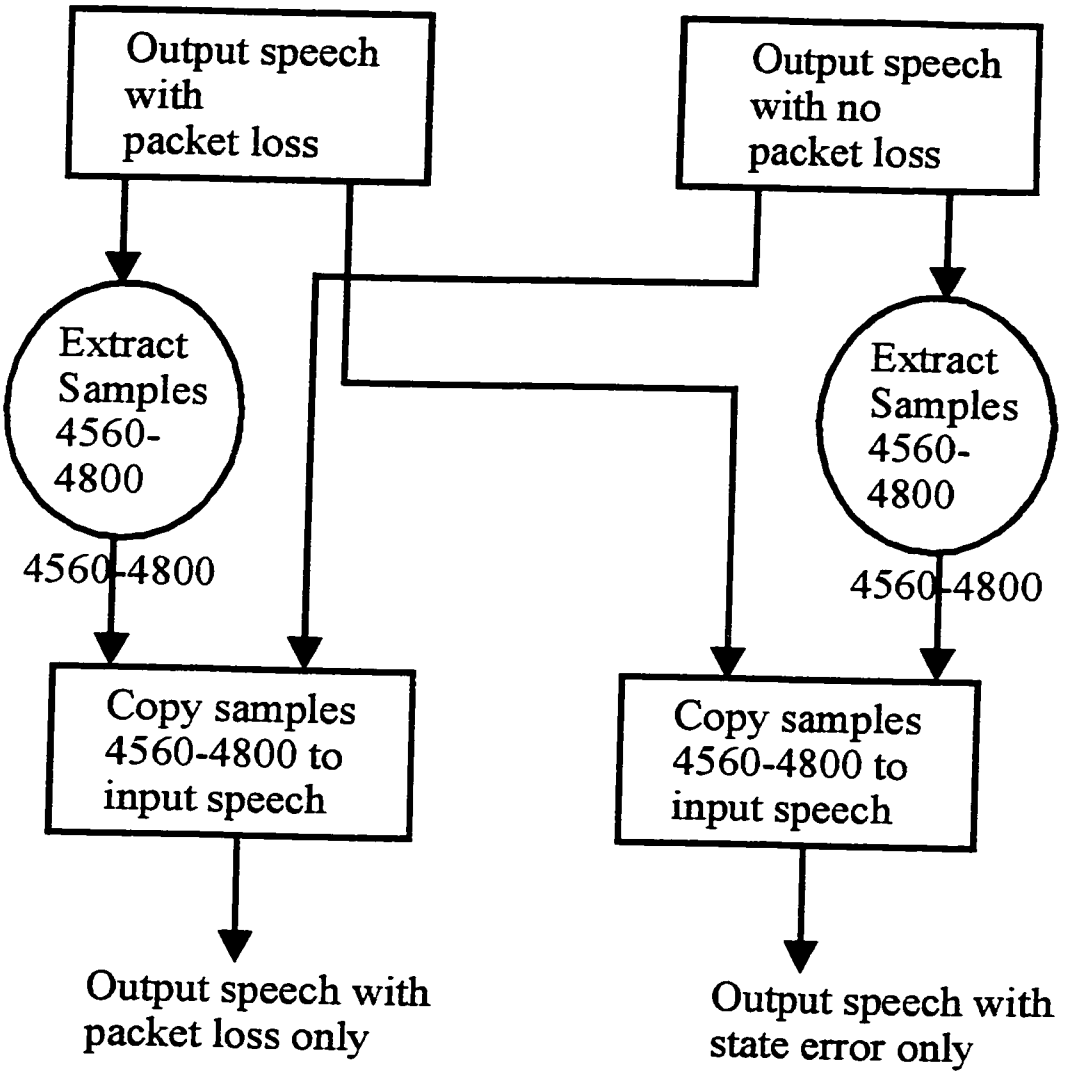

Figure 5.2: Isolating the Packet Error from the State Error

The method requires as input an output speech file with packet loss and an output speech file without packet loss. To generate a speech file that contains only the state error, we must remove the degradation during the period of packet loss from the speech file in which the packet loss has occurred. This can be done by replacing the speech samples during the period of packet loss in this speech file (samples #4560-4800) with the values of these samples in the speech file where no packet loss has occurred. Similarly, to generate an output speech file with only the packet loss, we can add the degradation during the period of packet loss to the speech file where no packet loss has occurred. This can be done by replacing speech samples #4560-4800 in this speech file with their values in the speech file where the packet loss has occurred.

It is then possible to listen to a full speech file that contains only one portion of the error at a time. Listening to both speech files, it was possible to observe a noticeable error in both, which means that both the packet loss and the state error are disturbing to the ear. In this example, the state error was more disturbing to the ear than the packet loss.

However, the purpose of this experiment was not to determine which error is the most disturbing since this will vary from case to case. Our objective was to verify that *both* the packet loss and the state error contribute to the degradation in the speech quality. This means that any attempts to improve the performance of the algorithm after a packet loss should consider reducing the error during the packet loss *as well as* the error due to state error. Since error recovery in the algorithm is simply concerned with recovering the

speech during a packet loss, it is necessary to explore ways by which the system recovers from an error in the state. This is the motivation of the rest of this chapter.

5.3 States in G.729A Algorithm

5.3.1 Overview

To be able to develop a method to recover from a state error, we must first fully understand the presence of the state in the G.729A algorithm. This is why in this section, the exact areas in the algorithm where state exists will be studied, as well as the time needed for each state to be resynchronized with the encoder. We will then examine which state is the biggest contributor to the state error since emphasis should be put on recovering this state.

5.3.2 Location of State

A state is defined as the contents of a memory that is common to both the encoder and the decoder. It is possible to examine either the encoder or the decoder to describe the states in the algorithm. For simplicity, only the decoder's state will be described.

Through a thorough examination of the algorithm, the areas where a state exists in the algorithm were identified. These are illustrated in the block diagram of the decoder in Figure 3.7 in section 3.5.5 as MEMORY x where x is an arbitrary number assigned to

each state. In all, five different areas of the algorithm were identified as having state memory. These are the same areas that were identified by Rosenberg in [60], but they will be described more thoroughly here. These five states are described in Table 5.1. Note that the state # is consistent with the memory # given in Figure 3.7.

Table 5.1: Description of the State Present in G.729A

State #	Size	Description
1	640 bits	• resides in memory of 4th order MA predictor used in the prediction of the line spectral frequencies (LSF) • represents the past four 16-bit inputs to the predictor for each of the 10 LSF
2	2464 bits	• resides in past values of overall excitation • these are used to form the adaptive codebook vector • represents the past 154 16-bit samples of the overall excitation
3	160 bits	• resides in memory of the 10th order all-pole LPC synthesis filter • represents the past 10 16-bit outputs of the filter
4	64 bits	• resides in memory of the 4th order MA predictor used to predict the value of the fixed codebook gain • represents the past 4 16-bit inputs to the predictor
5	16 bits	• resides in the gain of the fixed codebook vector pre-filter • this gain is chosen to be the adaptive codebook gain of the previous frame (16 bits)

5.3.3 Time Needed to Recover State

Now that the location of the state in the algorithm has been identified, it is important to examine how long it takes after a packet loss before the states of the encoder and decoder are fully resynchronized. For some states (states #1, 4 and 5), a simple analysis of the algorithm can determine this since these states are dependant on past inputs. Examining the size of the memory and how often new inputs are received, it was possible to determine that state #1 recovers after 4 frames, state #4 recovers after 2 frames and state #5 recovers after 1 frame. Analysis also tells us that state #3, the memory of the LP synthesis filter, will recover in a variable amount of time and this duration is dependant on the LPC impulse response (speech dependant). State #2 will also recover in a variable amount of time since it depends on past outputs. The results of this analysis are summarized in Table 5.2.

To get a better idea of the time it takes for the state #2 and #3 to recover, tests were done. The system setup was modified to generate the state values at the encoder and the decoder and to write them to a text file for further analysis. Thus, to identify how long the state needs to be resynchronized, the text file is examined to determine the # of frames following a series of packet losses needed for the state at the encoder and the decoder to resynchronize. For these tests, bursts of frame loss were inserted in different areas of the speech where the duration of the frame loss varied from one to five frames.

Table 5.2: Number of Frames Needed to Recover from Error in Different States

State #	# of frames	Details
1	4	• memory resynchronized when four consecutive correct new inputs enter the MA predictor • MA predictor updated once per frame
2	variable	• adaptive codebook vector is interpolated from the past overall excitation • overall excitation is formed with the adaptive codebook vector • state error should last indefinitely (error causes a new error...), but does eventually recover due to finite precision
3	variable	• memory lies in past output of LP synthesis filter • an error in this memory will lead to errors in outputs of filter • state error should last indefinitely, but does eventually recover due to finite precision
4	2	• memory resynchronized when four consecutive correct new inputs enter the MA predictor • MA predictor updated once per sub-frame (twice per frame)
5	1	• memory resynchronized when past value of the adaptive codebook gain is correct

After each test, the number of frames it took for state #2 and state #3 to recover was taken in note. The test was repeated for 100 different packet loss scenarios where the duration of the packet lost is kept fixed. These scenarios were repeated for different duration of frame loss from one to five frames. The average of these results is outlined in Table 5.3.

Table 5.3: Number of Frames Needed to Recover from State #2 and #3 Error

Length of Burst Frame Erasure	# of Frames Needed to Recover from State Error in memory #2 (past excitations)	# of Frames Needed to Recover from State Error in memory #3 (LPC memory)
1	32.75	30.97
2	31.72	29.45
3	31.95	29.61
4	31.78	29.52
5	32.50	29.97
Average	**32.14**	**29.90**

As can be seen by the results, the number of frames needed to recover from state error #2 and #3 does not depend on the length of the burst of frame erasures since the results are quite similar from one length of burst frame erasures to the next. Because of this, the average of the results which show that it takes 32.14 frames to recover from state error #2 and that it takes 29.90 frame to recover from state error #3 can be used as an estimate of how long it takes for these states to recover. The number of frames needed to recover from these state errors is in no way constant. This can be better understood by looking at

their variances, which are 129 for state memory #2 and 148 for state memory #3. What seems to be the most important factor in determining the recovery time is the area in the speech where the frame loss occurs. This is based on the observation that the number of frames needed to recover for two frame loss that occurred in the same region of the speech are fairly close.

Looking now at all the states, it takes on average 32.14 frames after the last frame erasure until all the state memory from the encoder and decoder are re-synchronized. However, this duration is variable and does not depend on the number of frames that was erased, rather on the area of the speech where the erasure was performed.

5.3.4 Relative Importance of the States

It should be noted that even though one state requires more time to recover from a state error than another one does, it does not necessarily mean that this state will contribute more to the error in the speech after the frame erasure. To clearly identify the significance of the degradation due to a given state error, state errors need to be isolated from one another. To do this, a burst of frame erasures was performed and the values of all states after the last frame erasure were saved. The speech compression algorithm is then repeated with no frame erasure while the state values are directly modified. The modifications are made only to the state that we are interested in isolating and the values that were previously saved (values when a packet loss had occurred) are used. This test was done to isolate the impact of an error in each of the five states. The results are shown

in the following figures in terms of resulting square error. Note that we are using the same definition of square error as the one that was used in section 5.2.3. In these figures, some small amount of square error is always present even if we use the same parameters each time because of the non-exhaustive searches done by the encoder which might result in different parameter values from one execution to the next. This explains the small square error visible in areas of the figures before the state is manually changed. The square error is used here instead of the perceptual error since it is easier to visualize. Since the error in all cases here is in the same area of the speech, a higher square error is more likely to be heard than a lower square error. The results in these figures are for one execution of the tests. The values of the square error though does not vary much if the tests are repeated. Finally, it is important to note that the scales used for the different figures differ greatly. The scales were chosen in order to fit the data values.

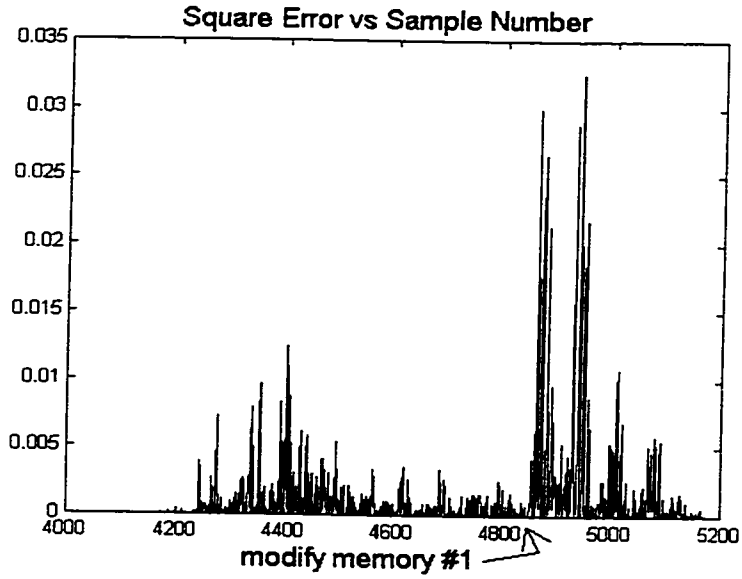

Figure 5.3: Effect on State#1

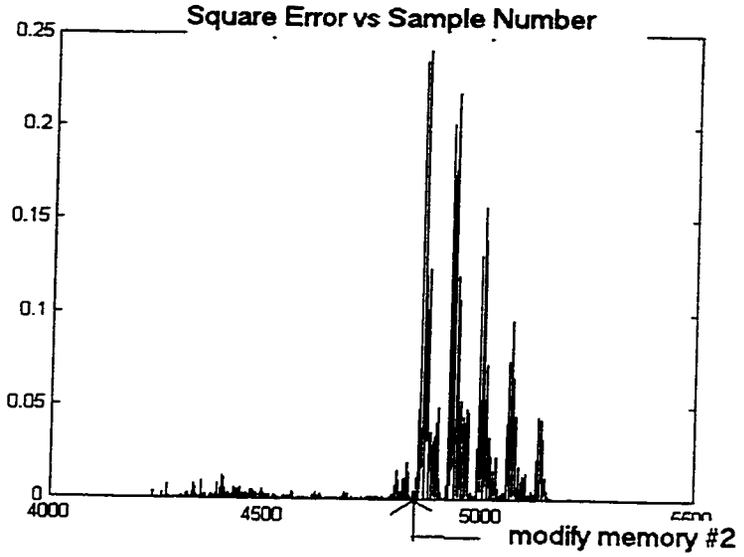

Figure 5.4: Effect on State#2

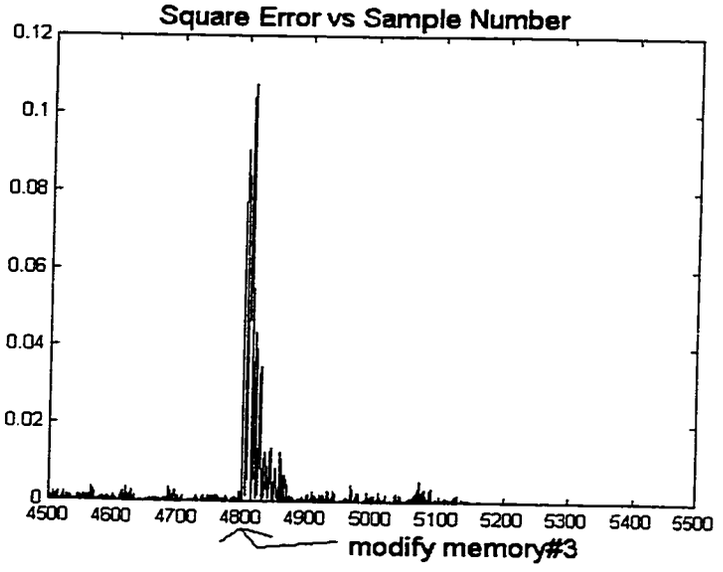

Figure 5.5: Effect on State#3

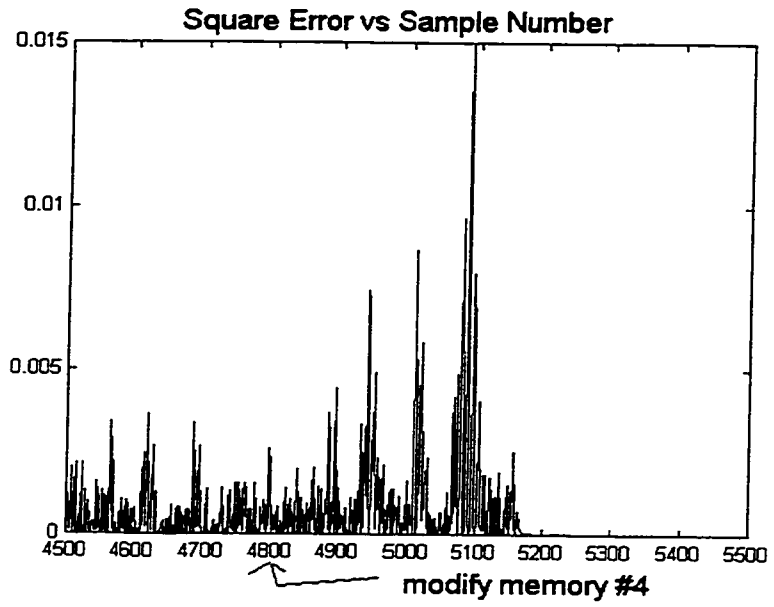

Figure 5.6: Effect on State#4

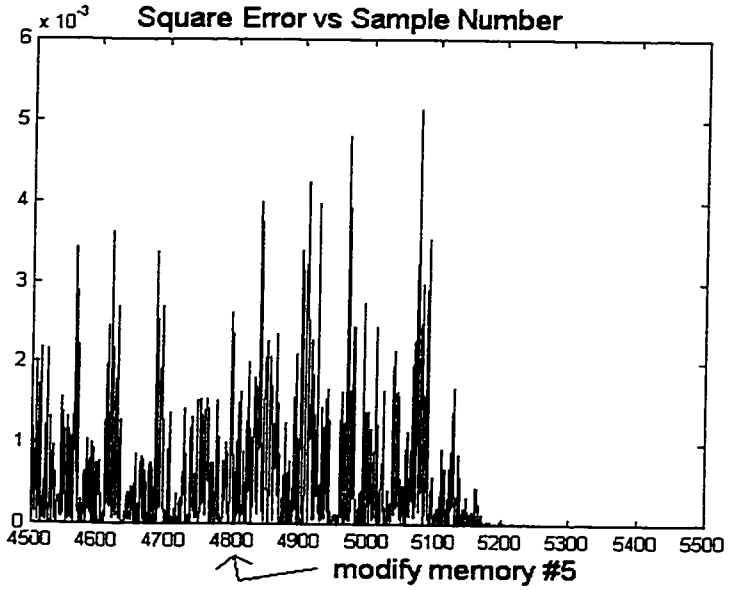

Figure 5.7: Effect on state #5

The results in these figures clearly show that memory #2, the memory associated with the adaptive codebook vector, is the one that causes the most error in the output speech when it is not synchronized. To put this in perspective, compare the squared error caused by modifying memory #2 with the squared error resulting from the packet loss in Figure 5.1. Noting that they are approximately the same, we can indeed conclude that the error in state memory #2 essentially causes most of the state error following a packet loss.

The second most important error is observed when we change memory #3. The maximum error here of 0.11 is approximately half the maximum error for memory #2. But this error is not nearly as important since it does not last long. After less than 50 samples, the error caused by modifying memory #3 disappears.

Compared to these two state errors, the other three state errors have little impact on the speech. None of the other states causes errors that exceed 0.05 in the output speech, which is negligible if we compare them to the 0.25 caused by memory#2.

Finally, the relationship between the different state errors is also important. All state errors will eventually cause some LPC state error (memory #3) since all of the states are either associated with the excitation, which is filtered by the LPC filter, or the LPC filter coefficients. Another relationship exists between state #4 and state #2. Any error in state #4 will also cause an error in state #2. This is due to the fact that an error in state #4

causes an error in the fixed codebook gain that is used to form the overall excitation (state #2).

Based on the results obtained in these experiments, we conclude that memory #2 is the most important contributor to the state error in the speech. Errors in memory#2 are audible in the output speech and last for a significant period of time. Compared to memory#2, all other state errors have negligible effect on the output speech error.

5.4 Methods for Recovering from a State Error

5.4.1 Overview

After having examined the state error following a frame loss and after having studied the origins of this state error in detail, we will now look at methods that can be used to recover from such a state error. Since it was shown that the state error degrades the quality of the speech after a packet loss as much as the packet loss itself, some method should be in place to reduce the impact of such an error. No such methods to reduce the amount of state error currently exist. In this section, two novel methods for recovering from state error will be presented. Preliminary results of applying these methods will be considered as well as the additional overhead required by these schemes. Results given in this section serve only to illustrate the basic principles of the schemes. More thorough results will be given in section 5.5.

5.4.2 Recovery by Retransmission

The most natural way to recover from a state error would be to have the encoder transmit its state values to the decoder. To reduce the overhead of such a scheme, it would be desirable to send the state values only after a packet loss. But since the encoder does not know before hand which frames will be lost, this is not feasible. Another option would be to have the decoder request a retransmission of the state whenever a frame loss occurs; but the delays involved cannot be allowed for interactive voice communication. The best compromise is to have the encoder periodically transmit the state to the decoder at regular intervals to reduce the state error due to packet loss as much as possible. Increasing the state retransmission interval increases the likelihood of some packet loss occurring during that interval and causing excessive state error before the state is recovered. Reducing the state retransmission interval adds extra overhead to the method since it will require more bandwidth to send the state more often. Based on the results in section 5.3.4 identifying the relative importance of each state error, the necessary bandwidth can be further reduced by sending state #2 only.

To test the effectiveness of this scheme, the code for the algorithm was modified to simulate this method. At some specific speech frame, the decoder was permitted to copy state # 2 from the encoder over its current state values. Results are illustrated in Figure 5.8 where after losing frames #58-60, the values of state #2 at the decoder were recovered at frame #61 (sample #4800) after the last frame loss.

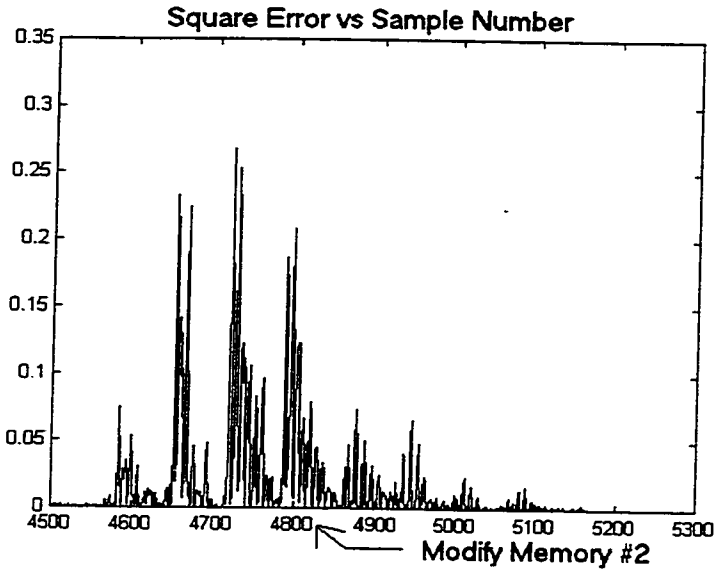

Figure 5.8: Effect of Recovering State #2 by Re-Transmission after Burst Erasure

Comparing the result in Figure 5.8 to the result in Figure 5.1 (both have suffered the same packet loss), the reduction of the state error is clearly visible. The square error after the packet loss, which originally had peaks near 0.25, now is reduced to 0.07. This is a reduction of the state error roughly by a factor of four.

As was discussed in section 5.3.4, state #4 has a close relationship with state #2 where any error in state #4 causes an error in state #2. Upon examination of the values of the state after the frame loss in the previous result, it was noted that state #2 did not stay synchronized for long. After the re-transmission, the error in state #4 causes additional errors in state #2. For this reason, the scheme can be further improved by re-transmitting the value of state #4 as well. A result of such a scheme is illustrated in Figure 5.9.

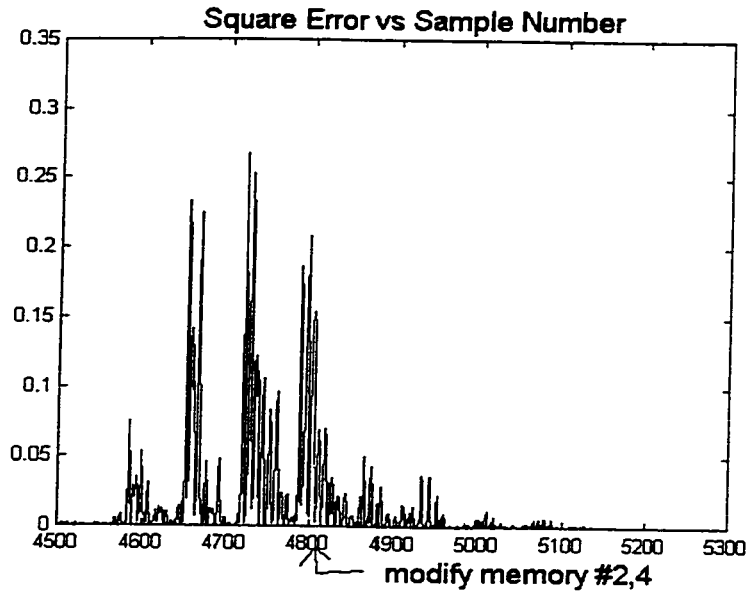

Figure 5.9: Effect of Recovering State #2,#4 by Re-Transmission after Burst Erasure

Comparing this result with the previous one, we see that re-transmitting memory #4 and #2 further reduced the state error in the case when re-transmission immediately follows the packet loss. However, for a more general case where the re-transmission does not occur directly after a frame loss, sending memory #4 would have no effect since memory #4 recovers by itself after two frames. In such a case, re-transmitting memory #2 only would be sufficient.

This mechanism was further tested on speech where many different isolated packet losses occurred. The square error without/with retransmission of memory #2 is illustrated in Figure 5.10 and Figure 5.11 respectively.

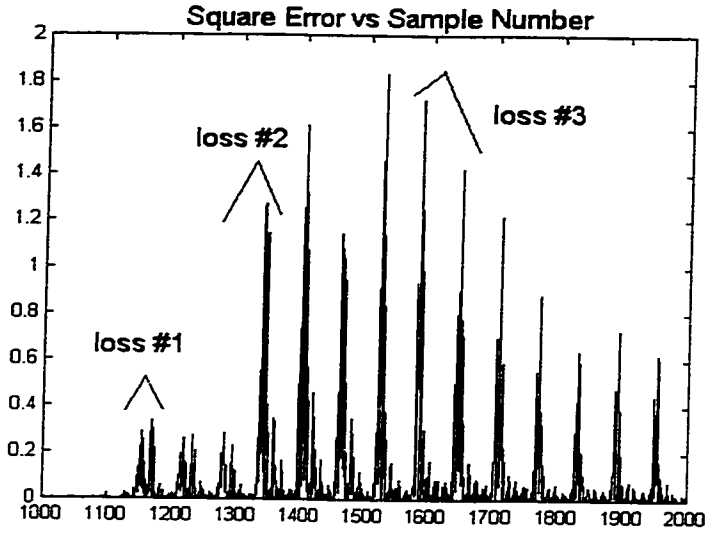

Figure 5.10: Random Packet Loss

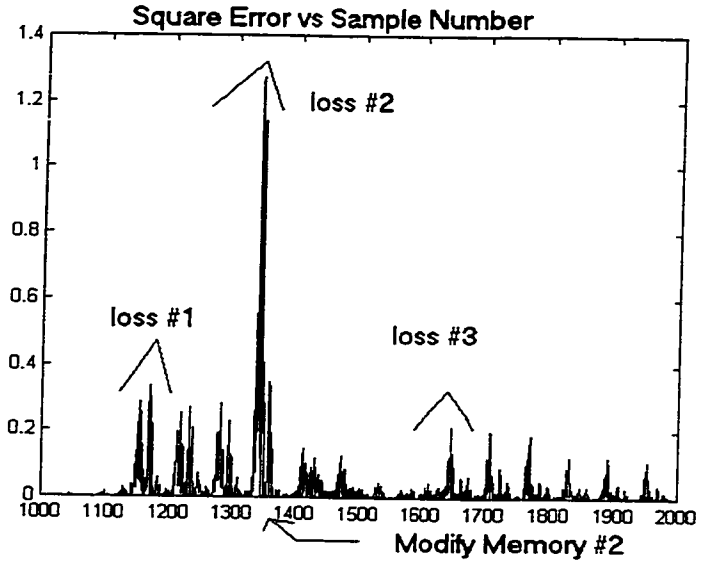

Figure 5.11: Loss with State Recovery

The results show a dramatic reduction in the square error when memory#2 is recovered after the second packet loss. The reduction of the error in the speech directly after a packet loss was to be expected since it is similar to what was observed in the isolated

packet loss case. What is different here is that we see that recovering state #2 after the second packet loss will also greatly reduce the square error of the third packet loss. This can be explained by the fact that the error we initially saw for the third packet loss was due to the lost packet and to the residual error in the state as a result of packet loss #2. Since the second and third packets are fairly close together (third loss is five frames after the second loss), the algorithm does not have the time to recover from the second state error before the next packet loss occurs. This has the effect of greatly increasing the error due to the next packet loss. What the results show here is that recovering the state value by re-transmission will not only reduce the state error following the last packet loss, but it will also reduce some error due to a subsequent packet loss when the interval between the lost packets is fairly short.

Results in this section have clearly shown that recovering state memory #2 by re-transmission of the state values from the encoder to the decoder greatly reduces the degradation due to state error. Also, we have seen that recovering state memory #4 as well will sometimes improve the performance of the scheme if we happen to recover directly following a packet loss. Before making a final judgement on such a scheme, it is necessary to evaluate its cost in terms of added complexity and bandwidth.

If we go back to Table 5.1, we see that the size of memory #2 is of 2464 bits and the size of memory #4 is of 64 bits. Since memory #4 is significantly smaller than memory #2, the additional overhead of transmitting it as well is negligible and will result in better

performance in some circumstances. Then the total size of the state to re-transmit is 2528 bits. The bandwidth required to transmit this state depends on the interval we choose for re-transmission. Even for a fairly long re-transmission interval such as one second, we see that such a scheme would cost an additional 2.528 kb/s. This represents an overhead of 31.6% compared with the 8 kb/s needed to transmit the speech frames. Such a high overhead is surely unacceptable particularly that at such a high re-transmission interval, the scheme would have very limited effects. The limited effectiveness of this scheme lies in the fact that many packets can be lost and fully recover from a state error before the re-transmission is done. A better scheme would be to re-transmit the state every 10 frames (100 ms) since most state errors do not have time to recover in 10 frames. Such a scheme would increase the required bandwidth by an additional 25.28 kb/s, which is three times the bandwidth required for the data and is clearly unacceptable. Another weakness of such a scheme is that the re-transmitted parameters could also be lost in periods of high congestion when it is greatly needed to improve the performance. In terms of the added complexity of this scheme, it is limited to the extra time required to send and receive the state values. Even though the scheme is fairly simple and gives good results, the additional overhead needed to get good performance is clearly too much. Since such a method will never be viable, an alternative way to reduce the state error will be examined next.

5.4.3 Recovery by Re-Initialization

Recall that for a given state, the encoder produces the best set of compression parameters to go with that state. While the quality of the compression clearly depends on the optimality of the state, the major cause of errors following a packet loss is the mismatch between the state and the compression parameters. Based on this argument, state error can be reduced by resetting state #2 at both the encoder and the decoder to a common value after a packet loss. As to what common values to choose, zero is the best choice since it will cause the least degradation in speech quality. Setting the values of state #2 to any other value than zero will add a past history to the system that is not valid. This invalid past history would clearly degrade the speech quality since the adaptive codebook vector would be interpolated from an invalid past excitation. Setting the values of state #2 to zero on the other hand should not cause as much degradation since this is equivalent to a system with no past history.

From this discussion, an alternative way to reduce the state error is to periodically re-initialize to zero the state values simultaneously at the encoder and the receiver. This method is referred to as recovery by re-initialization [44]. It is preferable to re-transmission since it does not require any additional bandwidth and does not add any complexity to the algorithm. In this section, the general method will be explained and a few illustrative examples will be given. The actual performance of this scheme will be detailed further in section 5.5 through many experimental results.

Since memory #2 is the most important contributor to the state error, this re-initialization will be performed only on memory #2. As to when to do the re-initialization, doing it periodically every 10 frames seems to be appropriate choice since the state error usually takes more frames than this to recover by itself. Further justification of such a re-initialization interval will be given by the results in section 5.5.

For such a scheme to be viable, it must not cause significant degradation to the speech when no frame loss occurs. To verify this, we must examine the effect of setting memory #2 to zero at the encoder and decoder when no frame loss occurs. The source code for the algorithm was modified to periodically reset to zero the memory associated with state #2 whenever the frame number was a multiple of 10. Implementing such a scheme does result in some small square error between the output of the system with no re-initialization and the system with re-initialization. But what may seem at first surprising is that to the ear, both outputs sound exactly the same. This is due to the fact that the amount of error is small and that most of the square error is masked and is not audible to the user.

In order to understand why such a scheme does not cause any more degradation, the exact effects of re-initialization on the algorithm must be investigated. To fully understand the effect of re-initialization, one must return to Figure 3.6 where the principle behind CELP was illustrated. Re-initializing memory #2 to zero sets all past overall excitations to zero. Since the adaptive codebook vector (in Figure 3.6, it is the impulse train generator) is

interpolated from the past overall excitations, when re-initialization is done the adaptive codebook vector will be zero. This means that in such a case the overall excitation will be formed only from the fixed codebook vector (in Figure 3.6, it is the random noise generator). For unvoiced speech, this is not a problem since unvoiced speech can be synthesized perfectly using only the fixed codebook vector. For voiced speech, some degradation will occur since we are not able to represent the long-term periodicity in the speech as well using only the fixed codebook vector. The reason this degradation is still not severe lies in the analysis-by-synthesis nature of the algorithm. The G.729A algorithm chooses its fixed codebook vector by a trial-and-error method. In such a method, different fixed codebook vectors are used to synthesize speech at the encoder and the one that results in the least amount of perceptual degradation between the synthesized speech and the original speech is chosen. This means that when memory #2 is re-initialized to zero at the encoder, this will be taken into account by the encoder which will chose the best fixed codebook vector for this situation. This is not the same fixed codebook vector that would have been chosen if no re-initialization would have occurred, but it is the best one that could be chosen to reduce the perceptual difference between the original and encoded speech. Another reason why the degradation is minimized is due to the presence of a fixed codebook vector adaptive pre-filter. Remember that this adaptive pre-filter filters all of the fixed codebook vectors. Since the gain of this filter is the past value of the adaptive codebook gain and since the filtering offset is chosen to be the integer part of the pitch value, this adaptive pre-filter adds some voiced speech components to the fixed codebook vector. This reduces the impact of

having no adaptive codebook vector in periods of voiced speech. These two reasons explain why recovery-by-re-initialization does not cause any audible difference in the speech when no packet loss occurs.

Having said that, it was noted that the simple technique described so far to perform recovery-by-re-initialization may occasionally cause audible degradation in the speech in periods of packet loss. Following extensive tests with various speech files and speakers, it was determined that the frame where we are doing the re-initialization and the next frame following the re-initialization are more sensitive to packet loss compared to when no re-initialization is done. The effect of losing any of these two packets when re-initialization is performed is often worse than it would be if no re-initialization had been performed. It can be explained by examining more closely the method the algorithm recovers from a packet loss as explained in section 3.5.5. Whenever a packet loss occurs, the value of the pitch is chosen to be the last pitch value that was properly received while the value of the fixed codebook vector is chosen randomly. If re-initialization occurs, the past overall excitation is set to zero resulting in no adaptive codebook vector. The excitation is formed only by the fixed codebook vector which in this case is chosen randomly. This often causes considerable degradation (mostly for voiced speech) since the optimized parameters chosen by the encoder are not used and are replaced by a random choice. The degradation is not as severe if no re-initialization is performed since the adaptive codebook vector is well chosen and forms an important part of the overall excitation.

The reason why the packet that follows re-initialization is more sensitive to packet loss is slightly different. At that frame, the past overall excitation is no longer zero since the excitation chosen for the previous frame is taken into account. Even though it is not completely zero, it still has many zeros in it since it only accounts for one past excitation. The presence of so many zeros in the past overall excitation increases the importance of the pitch if the packet is lost. Since the adaptive codebook vector is interpolated from the past overall excitation, a value of pitch that is only slightly different than what was transmitted can lead to an adaptive codebook vector that is quite different (this close value of pitch might cause interpolation by more/less zeros). The method of packet loss recovery implemented by the algorithm is very good but still results in a slightly different pitch value compared to what was transmitted. If no re-initialization occurred the frame before, this difference is not important since the resulting adaptive codebook vector will still be similar. But when re-initialization was done the frame before, this slightly different pitch value results in an adaptive codebook vector that is quite different. This causes more degradation than would have occurred if no re-initialization were done.

Tests were done in order to determine the best way to remedy this situation. It was determined that the best method to remedy the situation is media-dependant Forward Error Correction (FEC), similar to what was described in section 4.5.2. Assuming that the frame where re-initialization occurred is frame #n, the method can be explained by the following:

- Send with frame #'n+1' the value of the fixed codebook vector and the gains for frame #n (34 + 14 = 48 bits)
- Send with frame #'n+2' the value of the pitch and the gains for frame #'n+1' (14+14 = 28 bits)

This technique requires an additional 76 bits per re-initialization interval and greatly reduces the sensitivity of the frame where re-initialization occurs and the frame following the re-initialization. This improvement assumes that the following frame that contains the redundant information will not be lost. Since losing two frames in a row is less likely than losing one frame, the improvement is valid in most cases. With this extra protection, these frames are now slightly less sensitive to packet loss than the case where no re-initialization occurs. The presence of the FEC permits better recovery of the packet loss at these frames. If we consider a re-initialization interval of 10 frames, this schemes requires sending 8.76 kb/s of data instead of the conventional 8 kb/s. This represents a 9.5% increase in the bit rate but the benefit is better performance in periods of packet loss.

We will now illustrate the performance of this scheme in the presence of packet loss through a few examples. The first example consists of the output when frames #58-59 are lost. The square error for the output without/with re-initialization is shown in Figure 5.12 and Figure 5.13 respectively.

The results do show that using re-initialization did indeed reduce the amount of state error in the output speech. Note that some residual state error is still present when re-initialization is used since the other state memories are still in error.

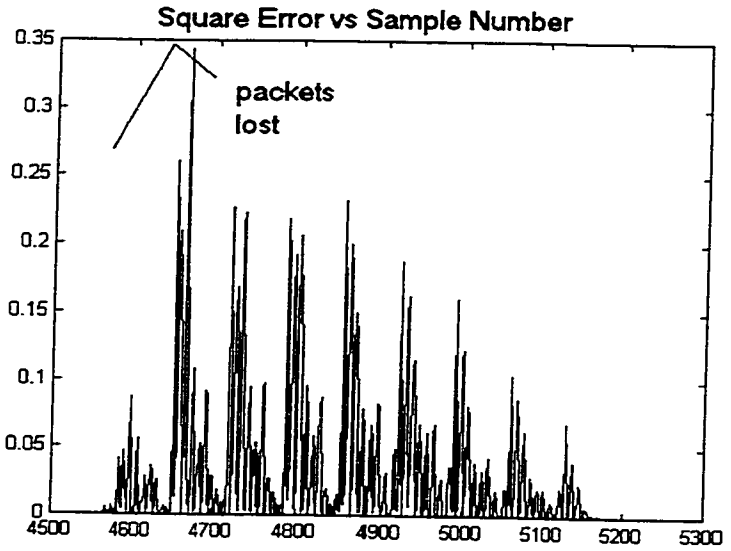

Figure 5.12: Without re-initialization

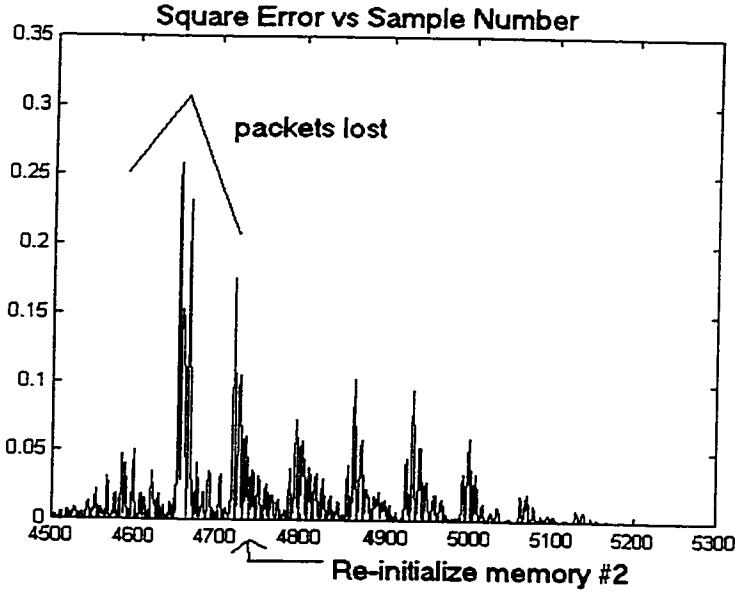

Figure 5.13: With re-initialization

Another example is shown next for isolated packet losses (frames #15,17 and 21 were dropped). Note also that frame #6 was also dropped but will not be shown in the results. Using the re-initialization every 10 frames, the re-initialization will occur at frame #20, which in this case is not directly after a frame loss. The square error for the output without/with re-initialization is show in Figure 5.14 and Figure 5.15 respectively.

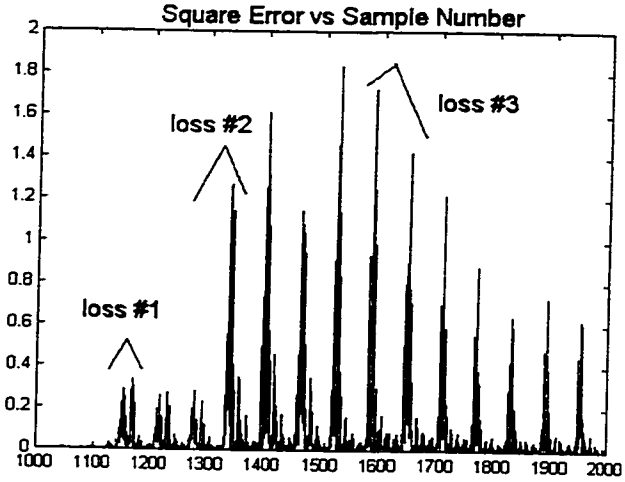

Figure 5.14: Without re-initialization

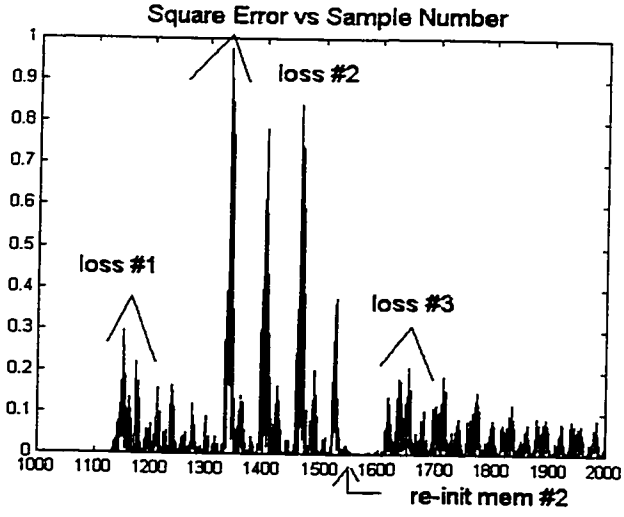

Figure 5.15: With re-intialization

The results clearly indicate that re-initialization reduced the state error, which in turn reduced the effect of frame loss #3. The effect of frame loss #2 was also reduced due to the re-initialization of the state at frame #10.

One final point to examine is to see whether it would be possible to re-initialize memory #4 in the same fashion. This should be considered since it was shown in section 5.4.2 that correcting memory #4 and memory #2 immediately after a frame loss reduced the state error even further in some instances. Re-initializing memory #4 as well as memory #2 every 10 frames was tested experimentally. The results though showed some audible degradation in the speech quality that made this unacceptable. The reason why re-initializing state #4 causes too much degradation is because it is associated with the fixed codebook gain. Since the gain is directly associated with the speech level, re-initializing the gain to any arbitrary value will cause a discontinuity in the speech level to which our ear is very sensitive. For this reason, the only way to re-synchronize memory #4 would be to re-transmit it as detailed in the previous section. Since such a scheme would need some additional bandwidth and that the improvement would only be noticeable if the re-initialization occurs directly following a frame loss, re-transmitting memory #4 does not seem to be warranted.

In conclusion, we demonstrated in this section through a few examples that recovery-by-re-initialization seems to be a promising method for reducing the state error following a packet loss. The basic principle behind this method, re-initializing periodically state #2

to zero, is very simple to implement. The only cost of this method is the additional 76 bits per re-initialization interval required to protect the frame where re-initialization takes place and the frame that follows it. The full potential of this scheme will be examined further in the following section where more results will be given.

5.5 Performance of Recovery-by-Re-initialization

5.5.1 Overview

In this section, the performance of the recovery-by-re-initialization scheme introduced in section 5.4.3 will be thoroughly tested. These tests are necessary to confirm that the scheme does indeed improve the performance of ITU-T G.729A in periods of packet loss. This section will start by examining the testing environment used to do these experiments. This will be followed by the evaluation of the performance of the scheme in periods when no packet loss occurs. This is done to justify that recovery-by-re-initialization does not degrade the speech quality significantly in such circumstances and to justify what re-initialization interval should be used. This section then ends in presenting results of using recovery-by-re-initialization in different periods of random packet loss. The results for a random packet loss rate from 5 to 50% will show quite well how this scheme can improve G.729A in periods of packet loss.

5.5.2 Testing Environment

The program used to perform the testing is the same as the one described in section 5.2.2. What will be specified further here though is how this program inserts random packet loss. Packet loss is inserted in the speech file according to a given input packet drop rate. For each encoded frame, a random number from 0 to 100 is generated. If this random number is below or equal to the packet drop rate, this packet is dropped; or else the packet is not dropped. If the packet is dropped, this is signaled to the receiver. The receiver then covers up this speech frame using the packet loss recovery mechanism. The results given in this section will be for packet drop rates of 5 to 50%. This simple model for generating packet loss was chosen since there is no accepted model for properly characterizing packet loss over the Internet. Since the Internet is such a huge and complex system, it is very hard to properly model it [47].

The speech files used for these tests were obtained from the ITU in [28]. These are speech files of 8 seconds each sampled at 8 kHz and used by the ITU to test the ITU-T G.729 algorithm. Each speech file contains two unrelated short sentences that have been padded with silence to ensure that each speech file lasts exactly 8 seconds. These speech files were recorded according to the requirements specified in the ITU standard ITU-T P.830 [30]. The naming convention of each speech file respects the following format:

l_kxxSyy

where *l* is a letter that identifies the laboratory where the speech file was recorded ('o' for Nortel and 'e' for AT&T), *k* indicates the gender of the speaker ('m' for male, 'f' for

female), *xx* is a number that identifies the speaker, S is always presented as is, and yy is a number that identifies the sentences that the speaker is reading. For example, the speech file o_m02s19 is a speech file that was recorded at Nortel. The speaker is male #2 and the words spoken are chosen from sentence #19. For the experiments done in this section, 20 different speech files were used. The speakers for these speech files contain 4 different male speakers and 4 different female speakers.

To evaluate the quality of the different output speech files, the perceptual error method described in section 4.7.4 was used. This method, developed by the author of this thesis, represents a good way to judge the perceptual difference between an original and a degraded speech file, i.e. an objective measure of the subjective speech quality.

5.5.3 Performance in periods with no packet loss

In this section, the degradation caused by recovery-by-re-initialization when no packet loss occurs will be evaluated. This is done for two reasons: to justify the choice of the re-initialization interval and to verify that this scheme does not cause any significant degradation in the quality of speech in such circumstances.

To help choose the proper re-initialization interval, the quality of four different speech files (2 male speakers and 2 female speakers) re-initialized at intervals from 1 to 20 was evaluated. The speech quality in terms of Perceptual Error per Sample and Maximum Perceptual Error is illustrated in Figure 5.16 and Figure 5.17. The results for Perceptual

Error per Sample demonstrate that the quality of speech is almost constant for a re-initialization interval from 10 to 20. As the re-initialization interval is reduced below 10, the value of the perceptual error per sample starts increasing until it reaches a maximum when the re-initialization interval is 1. The results for Maximum perceptual error do not seem to vary as greatly from one re-initialization interval to the next. It exhibits a fairly constant behavior until the re-initialization interval is reduced below 5 frames where in most cases it starts to increase.

As discussed in section 4.7.4, the Perceptual Error per Sample is the most important parameter to evaluate speech quality in situations where the degradation is distributed evenly throughout the whole speech file. Since this is the case here, we can see that any re-initialization interval that is higher than 9 will give the lowest amount of degradation in periods of no packet loss. This comes from the fact that the Perceptual Error per Sample is fairly constant above that interval.

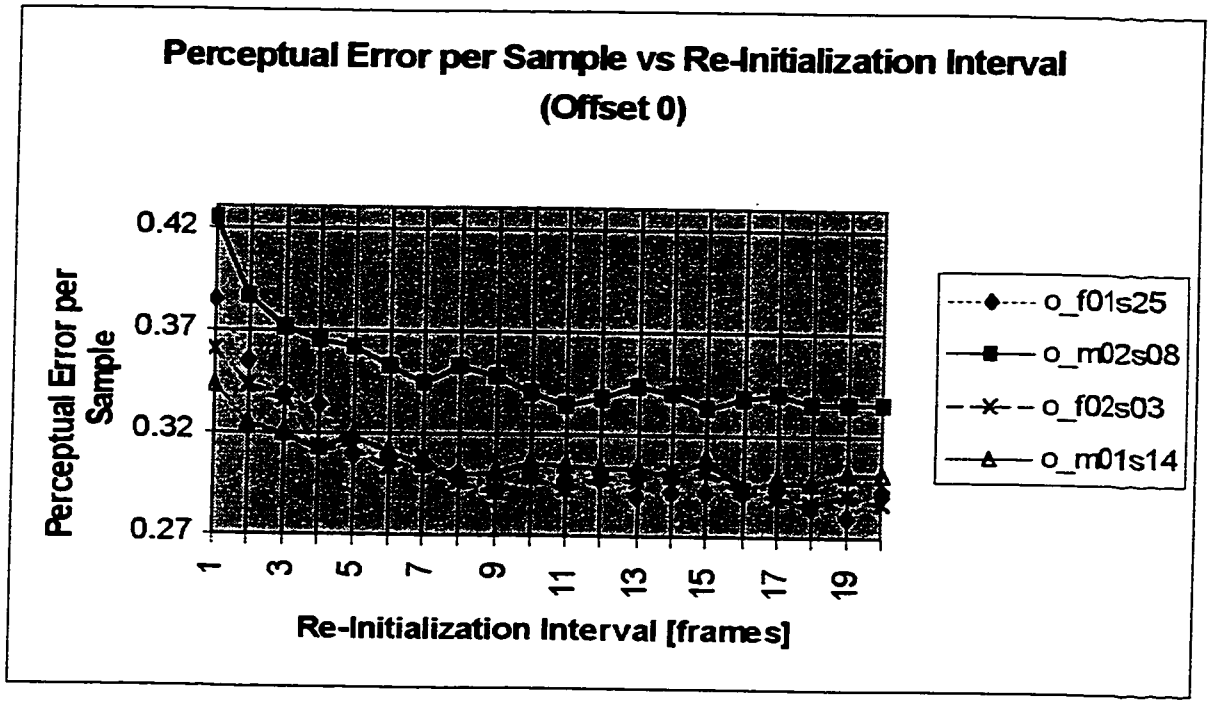

Figure 5.16: Perceptual Error per Sample for different Re-initialization Intervals

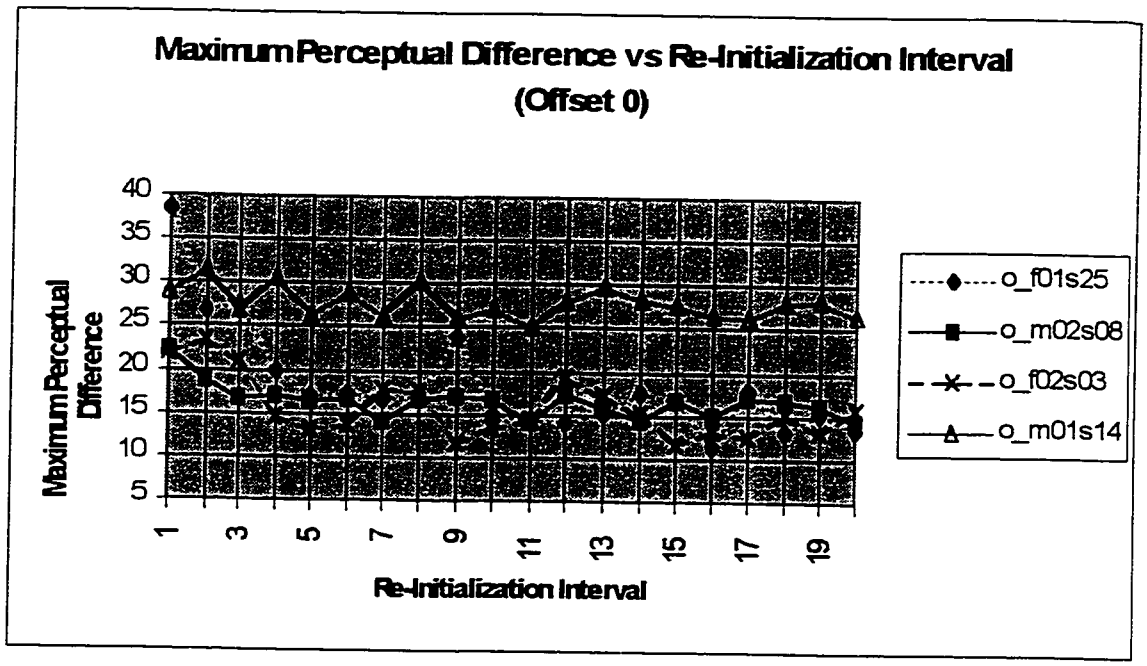

Figure 5.17: Max. Perceptual Difference for Different Re-initialization Intervals

A re-initialization interval of 10 frames was chosen since it respects this minimum and is still low enough to reduce the amount of state error in most periods of packet loss.

Another parameter that might affect performance in re-initialization is the re-initialization offset. This offset defines the exact interval at which frames will be re-initialized. For example, a re-initialization interval of 10 can be implemented by re-initializing at frame #10, 20, 30, 40, etc. or by re-initializing at frame #5, 15, 25, 35, etc. The offset tells us exactly which of these cases we are dealing with. Expressing it mathematically, a re-initialization scheme with re-initialization interval I and re-initializing offset O will perform the re-initialization at any frame #n where:

$(n+O) \bmod I = 0$

Using this definition, we see that re-initializing at frame #10, 20, 30, 40, etc. consists of re-initialization at an interval of 10 with offset 0 while re-initializing at frame #5, 15, 25, 35, etc. consists of re-initialization at an interval of 10 with offset 5. Since the offset depends only on the time when the algorithm starts compression, we want to be assured that different values of offset give similar results. Note that the value of offset used in Figure 5.16 and Figure 5.17 was 0. For the same four speech files used in the previous experiment, the quality of the speech for a re-initialization interval of 10 and an offset that varies between 0 and 9 was evaluated. These results are illustrated in Figure 5.18 and Figure 5.19.

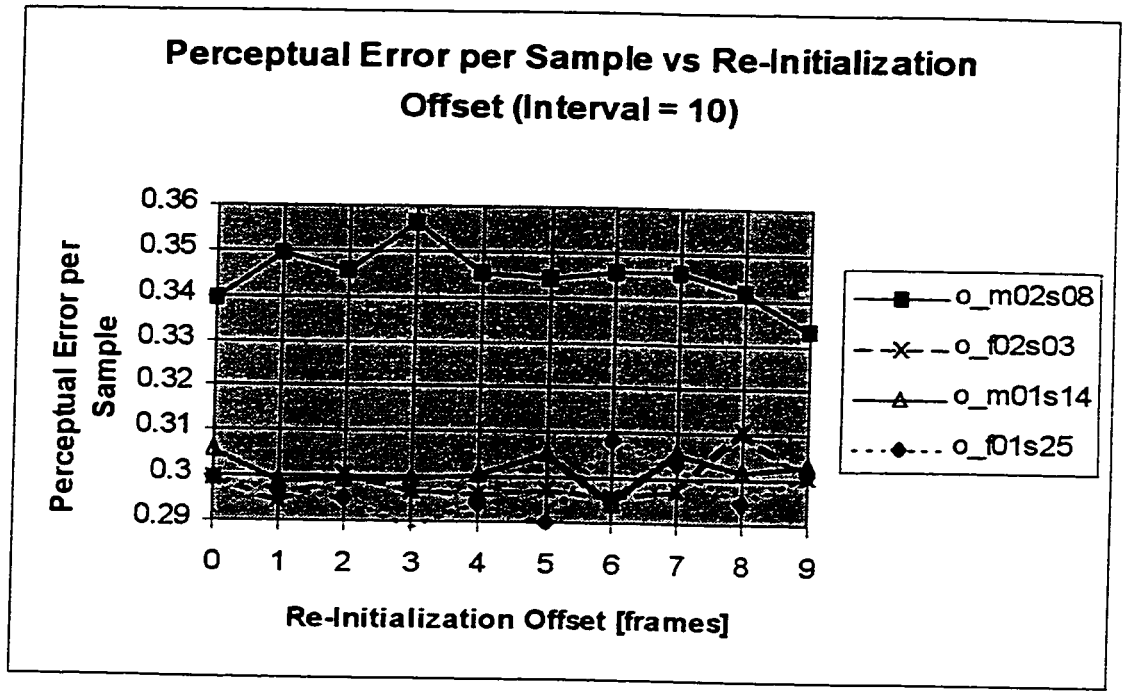

Figure 5.18: Perceptual Error per Sample for Different Re-Initialization Offsets

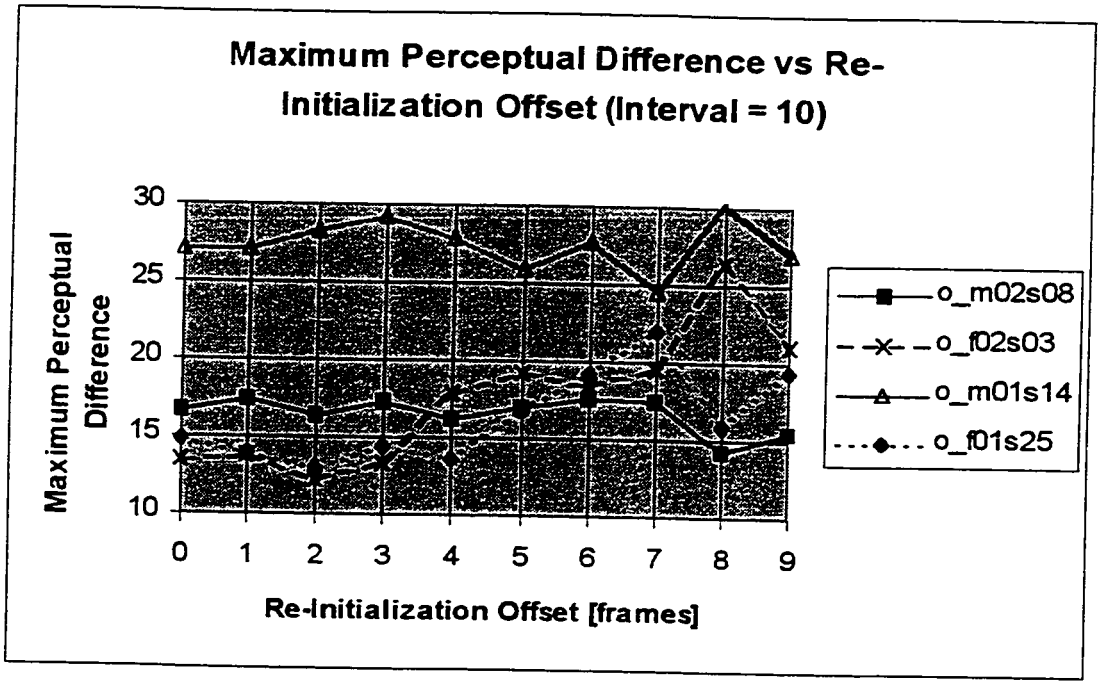

Figure 5.19: Maximum Perceptual Error for Different Re-Initialization Offset

These results do indicate some variance in the value of the Perceptual Error per Sample and the Maximum Perceptual Error when different offset values are used. For the most important parameter of Perceptual Error per Sample, that variance is quite small. More variance is experienced in the Maximum Perceptual Error, but this parameter is less related to the overall speech quality. These results do correlate quite well with the author's observation that the quality of the speech file for different re-initialization offset remains essentially the same. The slight change in quality from one offset to the other has a small difference in Perceptual Error per Sample, which makes it very hard to notice.

Now that an interval of 10 has been established as a good re-initialization interval, we can then proceed to compare the speech quality with and without re-initialization. For these tests, all 20 different speech files were used. For every speech file, the output of the speech compression algorithm when no re-initialization occurs and the output of the speech compression algorithm when re-initialization occurs (I=10, O=2) were recorded. The results in terms of Perceptual Error per Sample and Maximum Perceptual Error are illustrated in Figure 5.20 and Figure 5.21 respectively.

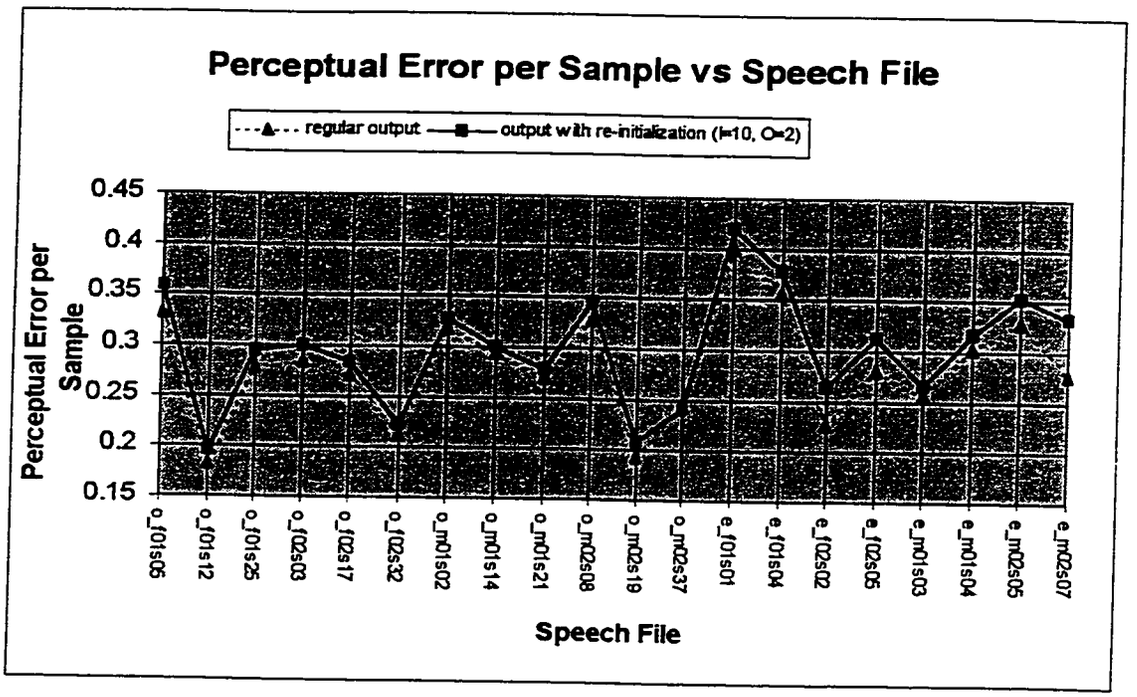

Figure 5.20: Perceptual Error per Sample with/without Re-Initialization

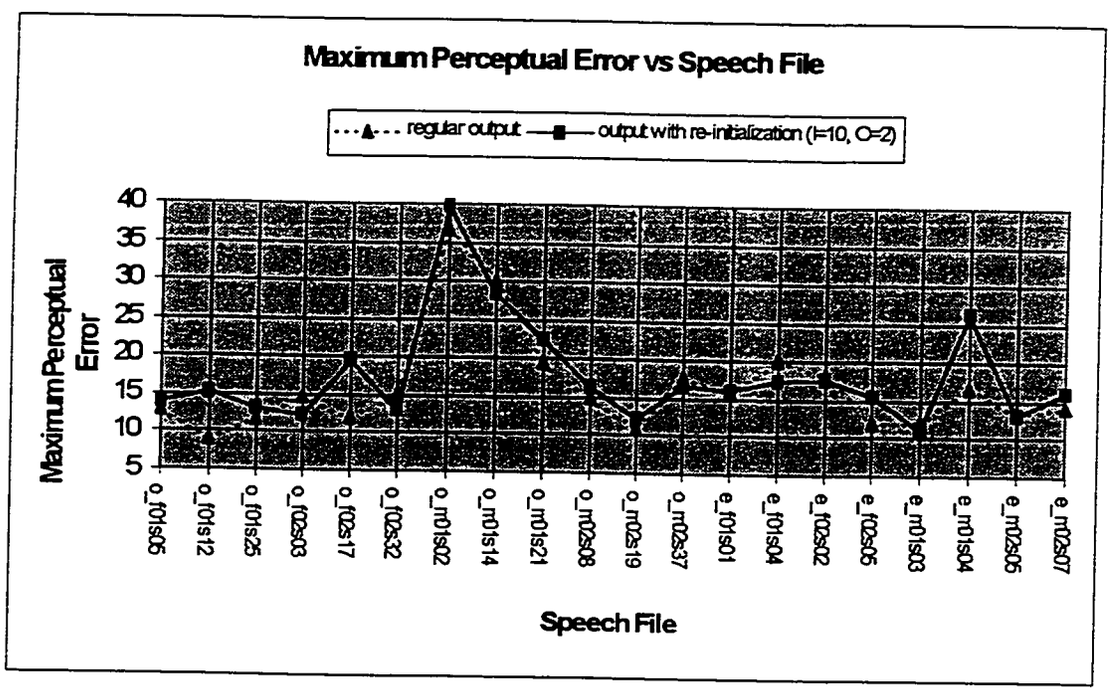

Figure 5.21: Maximum Perceptual Error with/without Re-Initialization

Examining the results, we can see that re-initialization results in an output speech file that has a slightly higher Perceptual Error per Sample. In terms of the maximum perceptual error, it is also in general slightly higher than it was when no re-initialization was performed. This degradation of the speech quality is quite small as compared to the degradation caused by the compression itself (degradation present when no re-initialization occurs). This explains why it is very hard to hear any difference between the speech with/without re-initialization. Another interesting observation is that the two curves for Perceptual Error per Sample have essentially the same shape. This is an indication to us that most of the degradation in both cases is common. This common degradation is the general speech compression algorithm that both use to encode the speech.

The results in this section permitted us to properly assess the performance of the recovery-by-re-initialization scheme in periods where no packet loss occurs. Even though this scheme was designed to improve the performance of the algorithm in periods of packet loss, it was important to make sure that it did not cause too much degradation when no packet loss is occurring. The main reason is that packet loss is hard to predict by the encoder and the scheme will need to run continuously even in periods where no packet loss occurs if we want it to be running when packet loss does occur. The results in this section indicated that the degradation caused by this scheme in periods with no packet loss is quite negligible if the re-initialization interval is chosen to be higher than 9. Also, it was found that this degradation did not vary significantly when the re-

initialization offset was modified. Finally, it was determined that a re-initialization interval of 10 was appropriate since such an interval would not cause any significant degradation in periods of no packet loss and would be quite apt to reduce the amount of state error in periods of packet loss.

5.5.4 Performance in periods of packet loss

Having verified that the scheme did not cause any severe degradation in periods when no packet loss occurs, we will now evaluate the scheme in periods of packet loss. Since recovery-by-re-initialization is meant to improve the performance of the speech compression algorithm in periods of packet loss, it is very important that we have some concrete evidence that shows that this is the case.

In this section, we will start by comparing the speech quality with/without re-initialization when packets are lost at a packet drop rate of 5%, 10%, 20%, 30%, 40% and 50%. For all of these experiments, all 20 speech files will be used. For each speech file, a random packet loss with the appropriate packet drop rate was first run using G.729A without re-initialization. This was then repeated dropping the same packets as before but now using re-initialization at an interval of 10 and offset 0. The speech quality of the two output speech files was then compared in terms of Perceptual Error per Sample and Maximum Perceptual Error. The results of these experiments are given in Figure 5.22 to Figure 5.33 that follow. To help with the comparison, the speech quality when no packet loss occurs and with no re-initialization will also be plotted as reference.

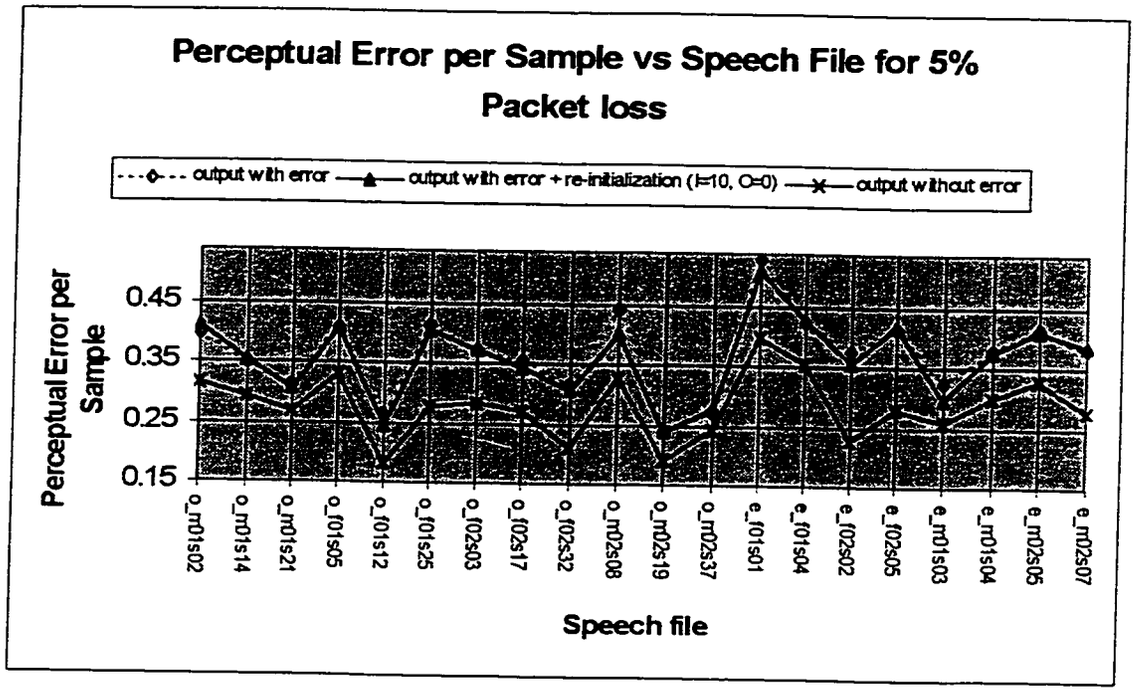

Figure 5.22: Perceptual Error per Sample for 5% Packet Loss

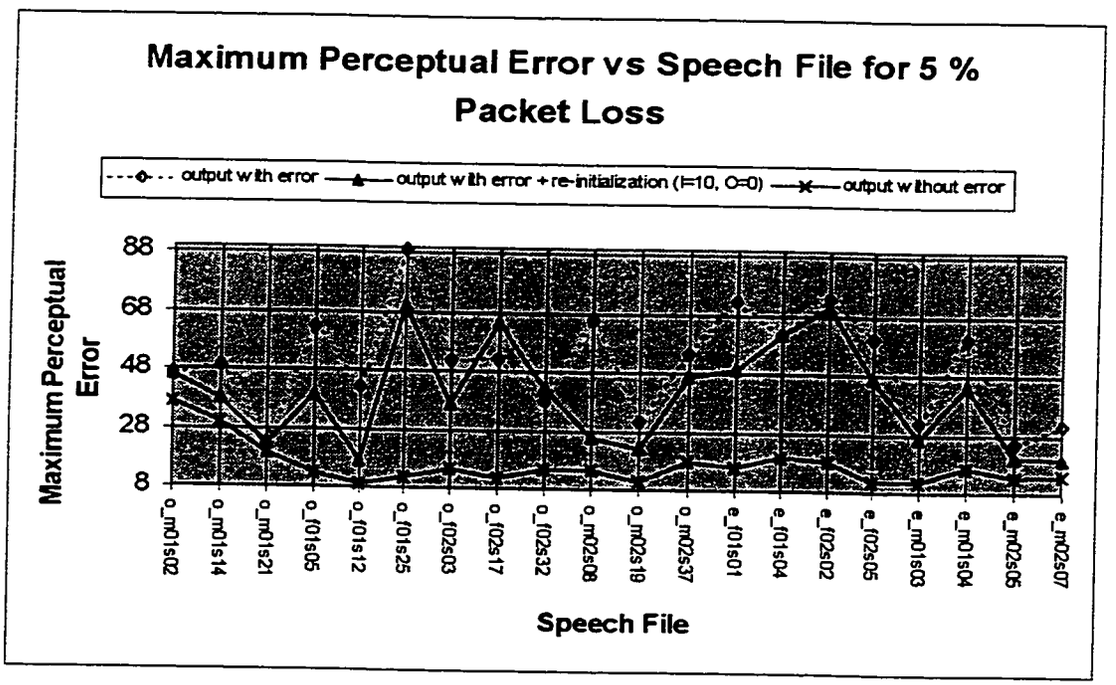

Figure 5.23: Maximum Perceptual Error for 5% Packet Loss

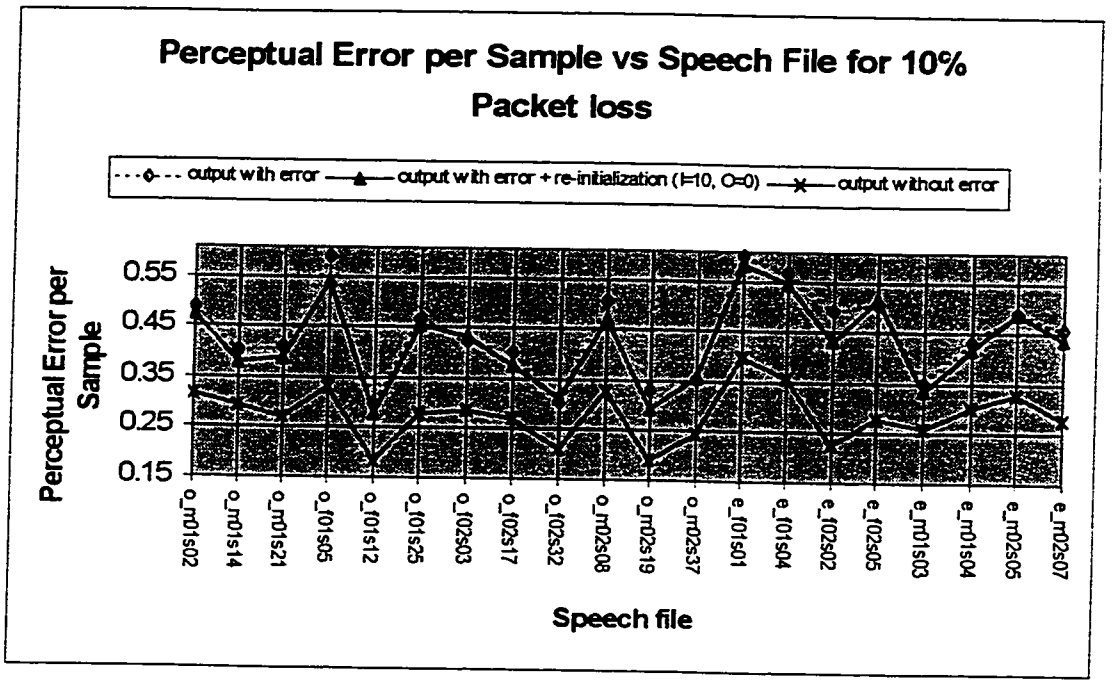

Figure 5.24: Perceptual Error per Sample for 10 % Packet Loss

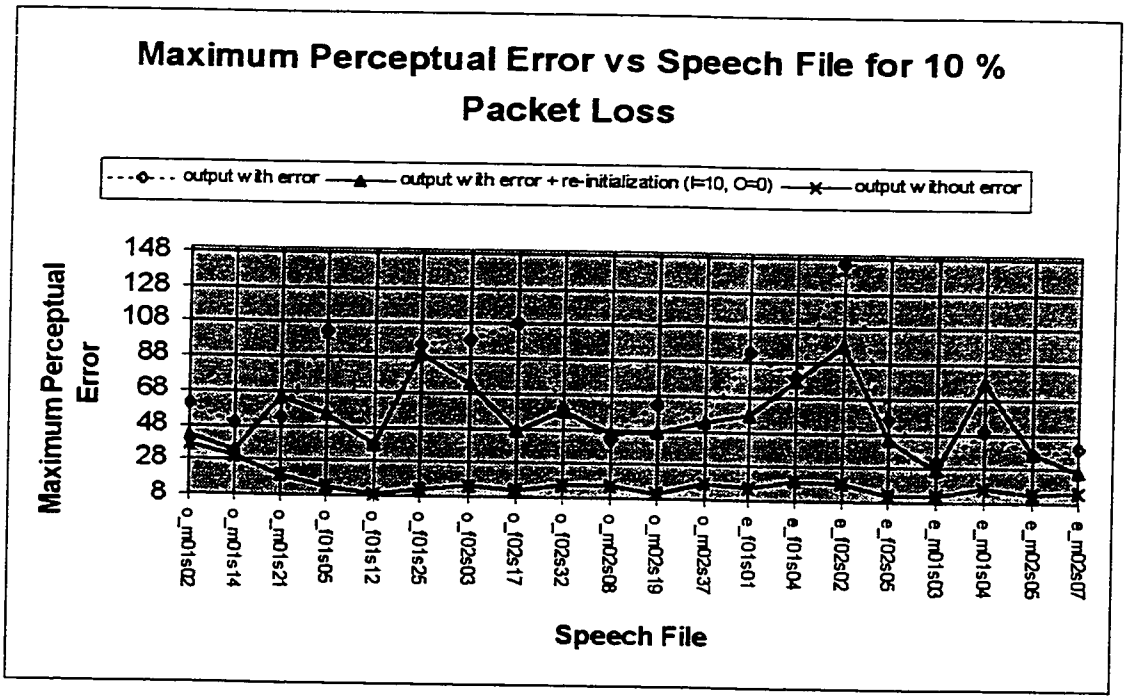

Figure 5.25: Maximum Perceptual Error for 10% Packet Loss

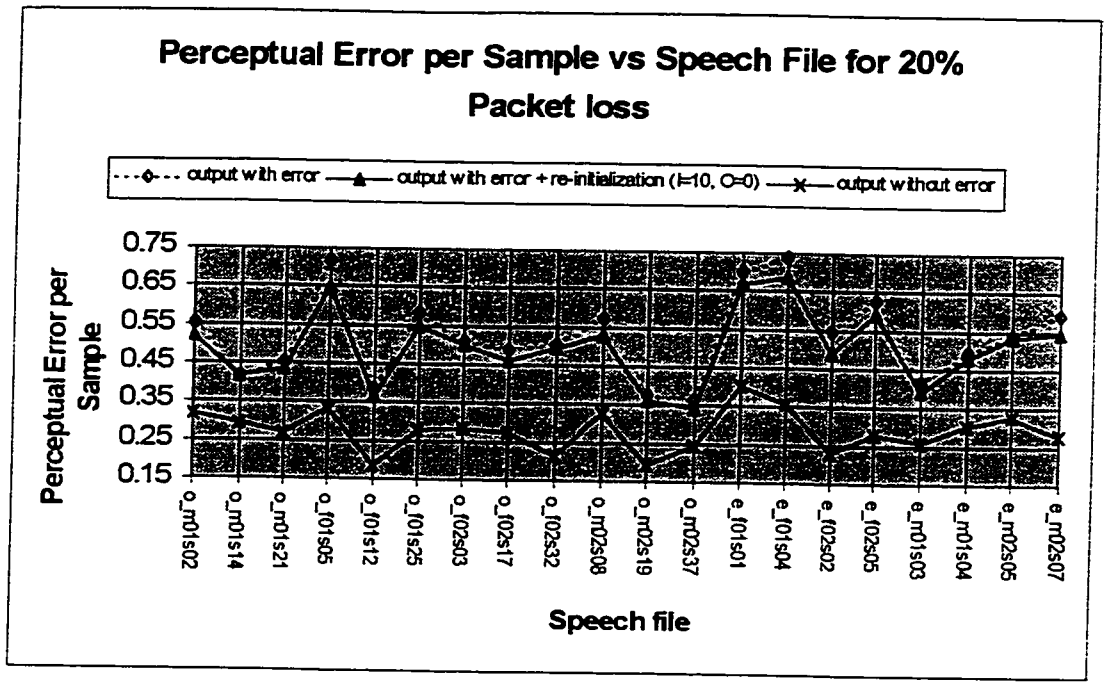

Figure 5.26: Perceptual Error per Sample for 20% Packet Loss

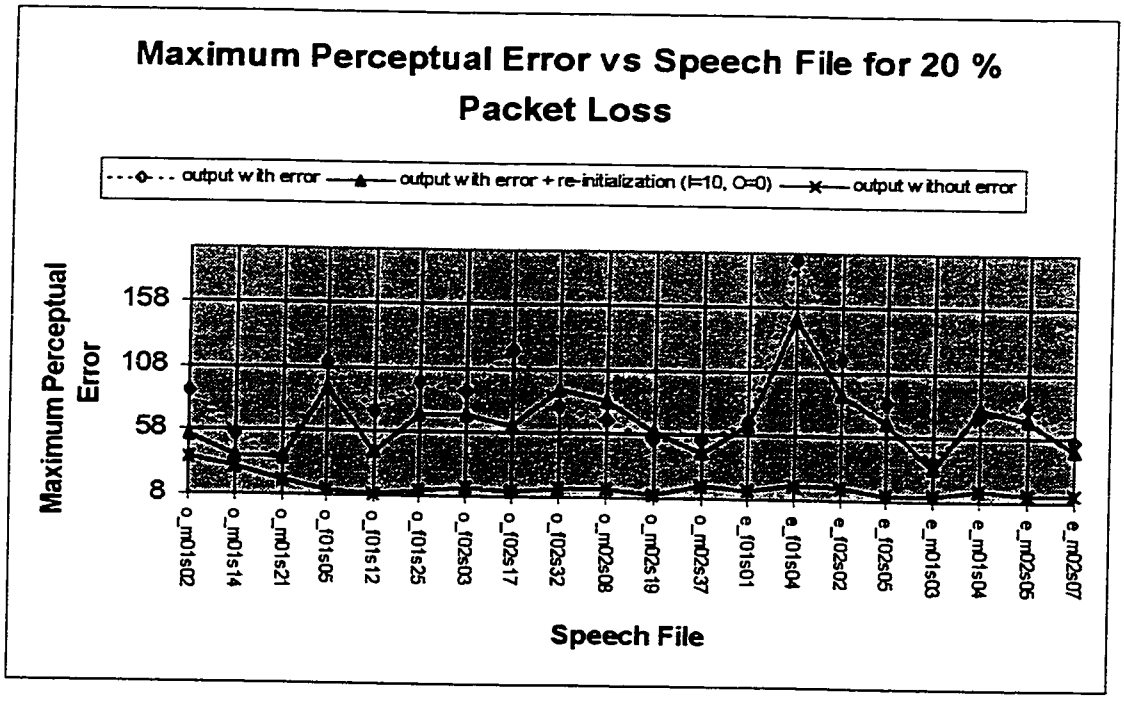

Figure 5.27: Maximum Perceptual Error for 20% Packet Loss

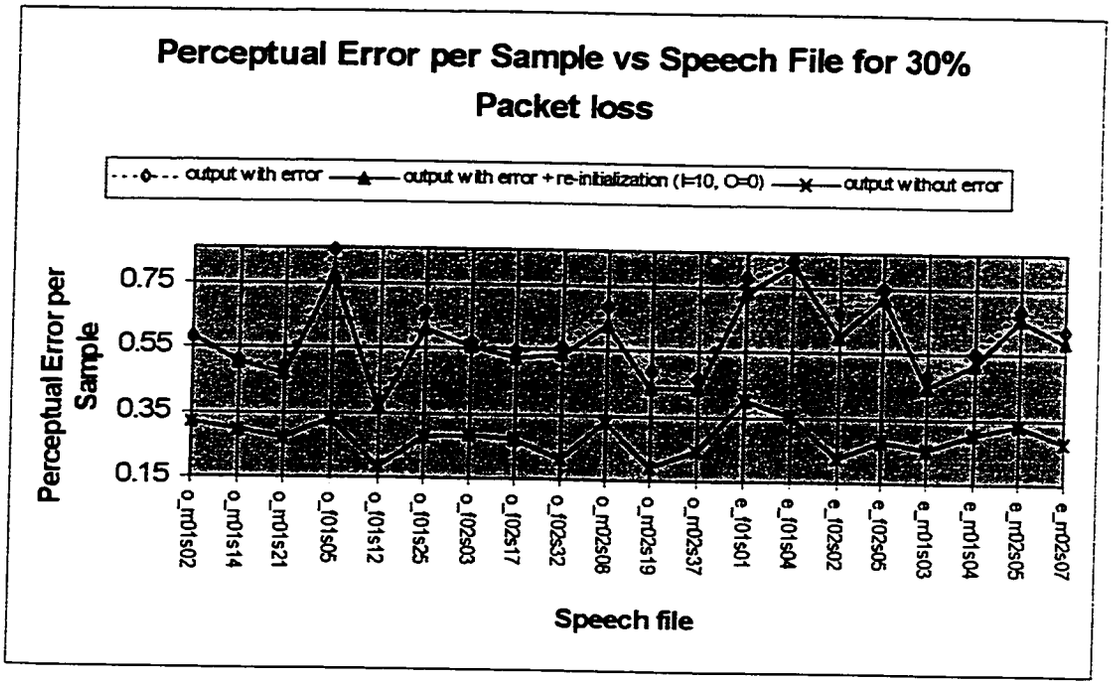

Figure 5.28: Perceptual Error per Sample for 30% Packet Loss

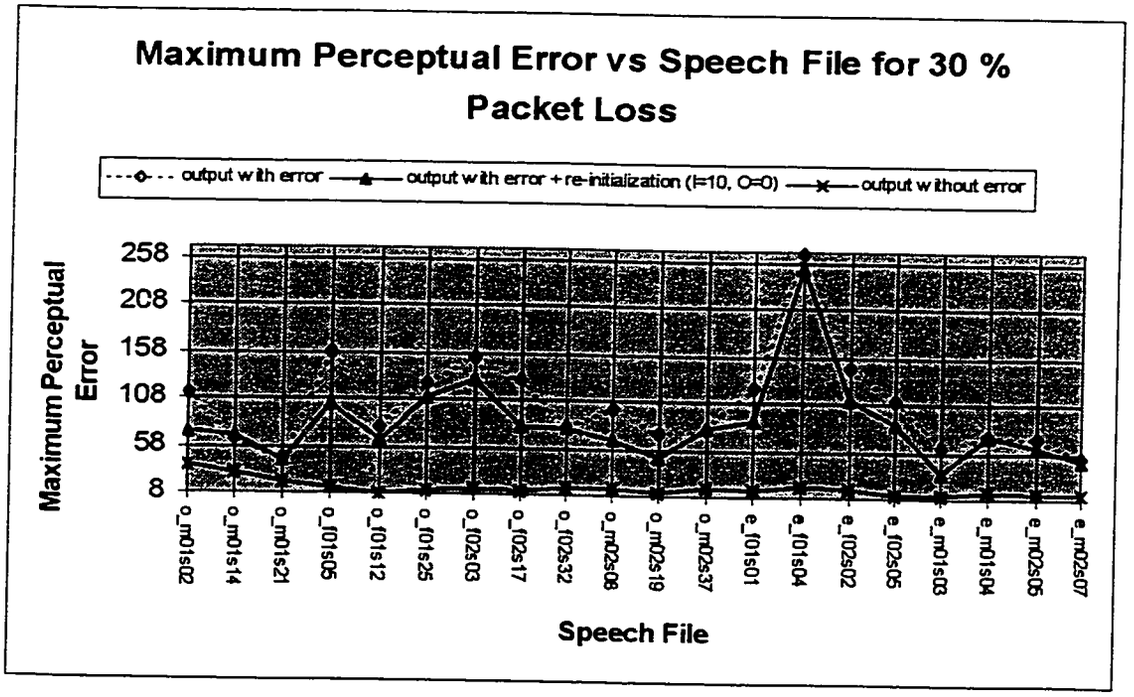

Figure 5.29: Maximum Perceptual Error for 30 % Packet Loss

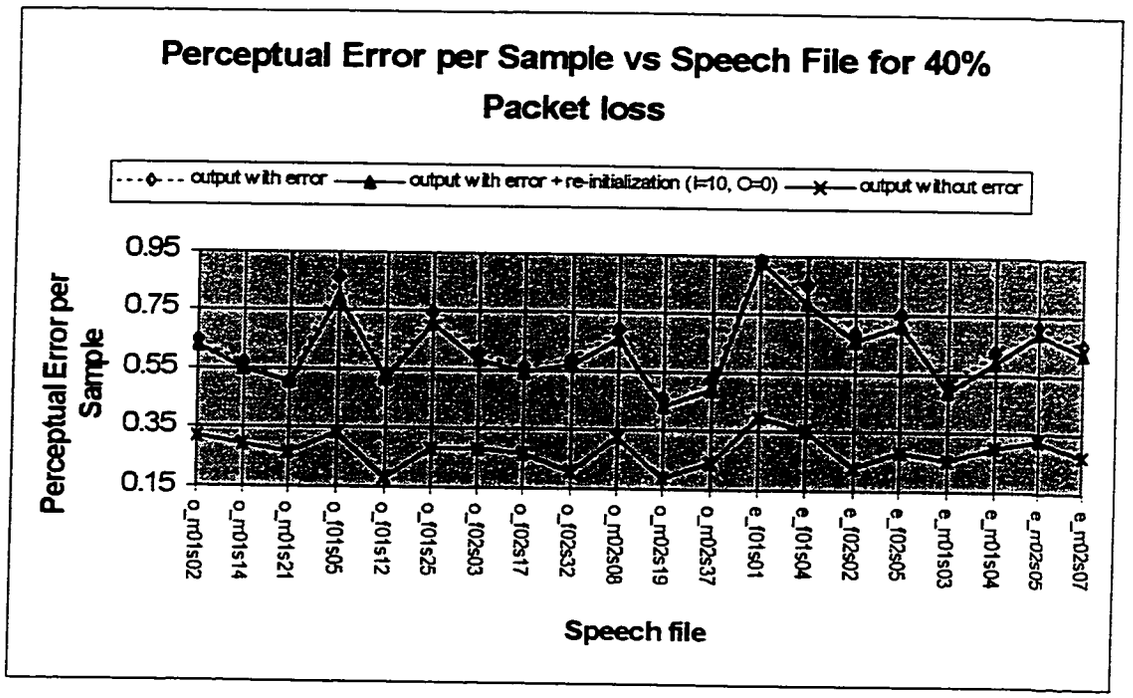

Figure 5.30: Perceptual Error per Sample for 40% Packet Loss

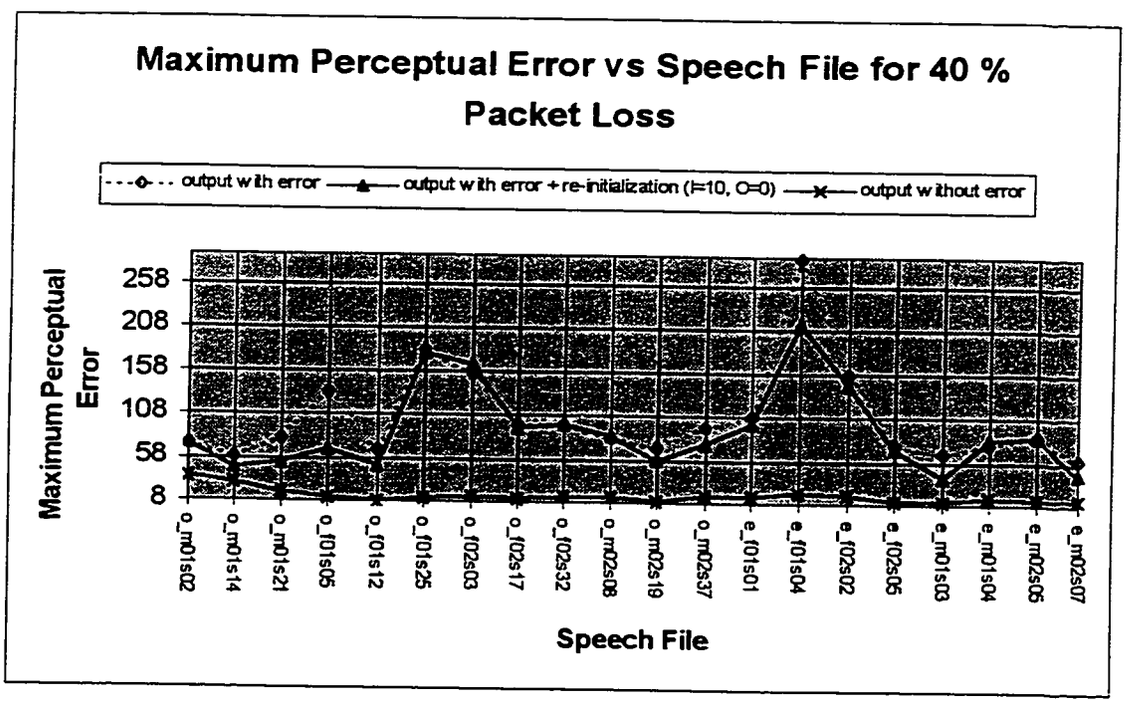

Figure 5.31: Maximum Perceptual Error for 40% Packet Loss

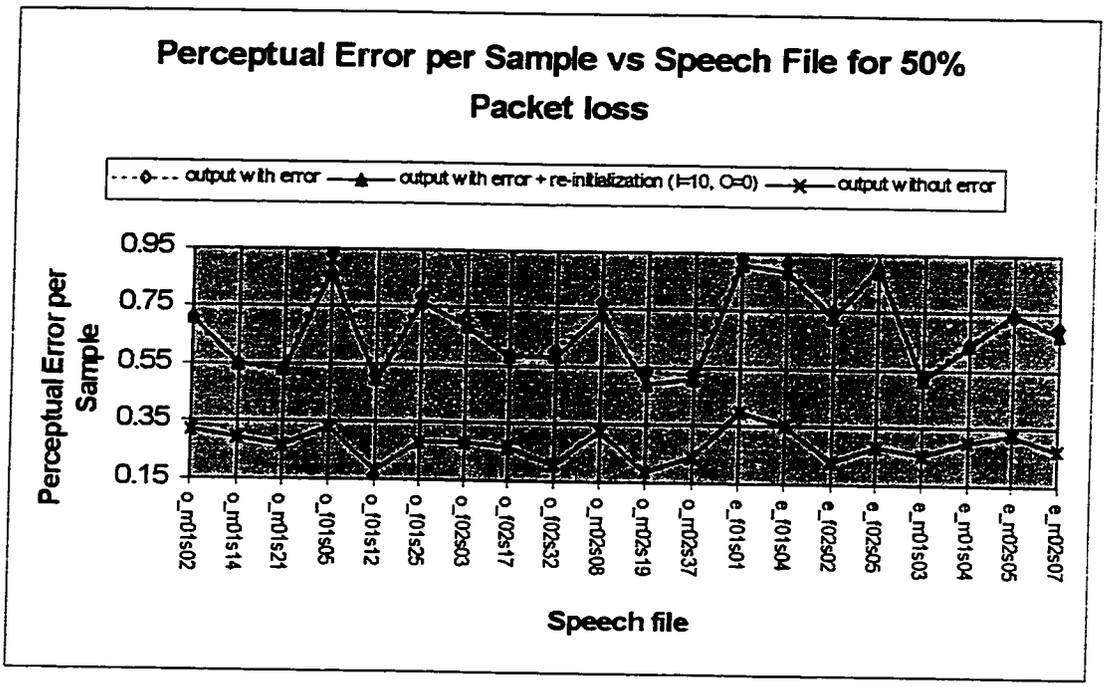

Figure 5.32: Perceptual Error per Sample for 50% Packet Loss

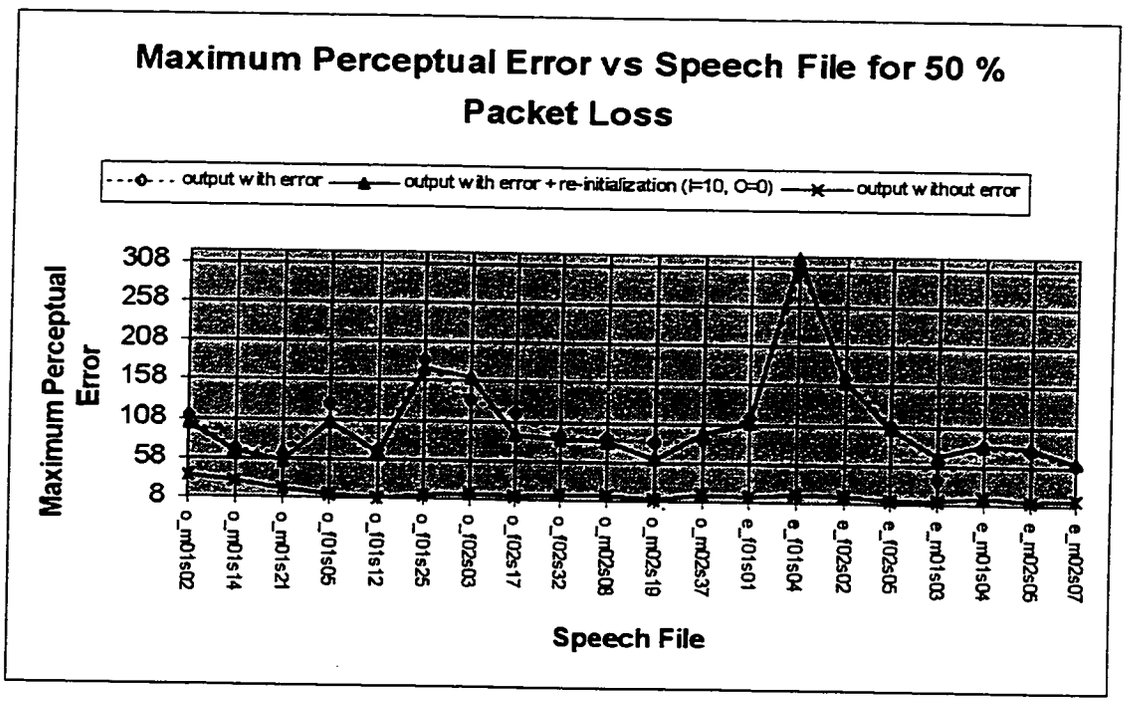

Figure 5.33: Maximum Perceptual Error for 50% Packet Loss

These results clearly show that recovery by re-initialization improves the speech quality in periods of packet loss. In terms of Perceptual Error per Sample, the results showed a clear, almost consistent improvement when using re-initialization as compared to not using re-initialization. Only a couple of isolated cases among all of the results did not show such an improvement and these can be attributed to the random nature of the frame loss recovery mechanism in G.729A. Since the number of such cases is very minimal compared to the number of tests performed, we can safely say that recovery-by-re-initialization improves the speech quality in all periods of packet loss. The results for the Maximum Perceptual Error are also quite similar. The cases where re-initialization has occurred have generally a lower amount for this maximum perceptual error as compared to the cases where no re-initialization was performed.

In all of these results, it is important to note that the actual improvements caused by re-initialization in periods of packet loss are even more significant than what can be viewed from the figures. Since re-initialization degrades slightly the speech quality when no packet loss occurs, the results presented for re-initialization are reduced slightly because of this. If we take this into account, we can conclude that the improvements caused by re-initialization are constantly higher than the degradations caused by re-initialization in periods of no packet loss.

In order to get a better understanding of the degree of improvement caused by re-initialization, the average improvement in terms of Perceptual Error per Sample and

Maximum Perceptual Error was calculated for each drop rate. The average improvement for these measures is simply the sum of the differences between the measure without re-initialization and the measure with re-initialization divided by the number of test cases (20). Since for both of these measures a smaller value indicates better quality, a positive average indicates an improvement caused by re-initialization. The values for these average improvements in terms of Perceptual Error per Sample and Maximum Perceptual Error are tabulated in Table 5.4 for each drop rate tested.

Table 5.4: Average Improvements caused by Re-Initialization

Drop Rate	Average Improvement in Perceptual Error per Sample	Average Improvement in Maximum Perceptual Error
5 %	0.006495	10.66
10 %	0.01846	13.24
20 %	0.02954	14.88
30 %	0.03410	20.06
40 %	0.02883	15.09
50 %	0.01698	2.42

The results in Table 5.4 do clearly show that in all test cases, recovery-by-re-initialization did generate an average improvement over recovery without re-initialization. Something else that we can deduce from this table is that this improvement increases with the drop rate up to a maximum at a drop rate of 30%. With a drop rate of 40% and 50%, the improvement reduces with increasing drop rate. The better performance of the scheme

for increasing drop rate from 5-30% can easily be understood. Since recovery-by-re-initialization reduces the amount of state error, it is quite normal that a speech file with a higher drop rate will be better improved since such a speech file is much more likely to have more state error. To explain why the improvements start to reduce as the drop rate exceeds 30%, we must listen to the quality of the speech file. From 5-30%, the quality of the output speech was still quite good. For the higher drop rates such as 20% and 30%, the degradation was audible but the speech was still intelligible. But when the drop rate attains 40% and 50%, the general quality of the speech degrades quite quickly. Some of the speech files with a 50% drop rate are no longer intelligible, hence the decision to stop at 50%. In such cases where the speech is highly degraded, re-initialization still removes some of the state error but this does not necessarily mean that the output speech will better resemble the original speech since it will still be highly degraded. The problem is essentially that the method of frame loss recovery in the algorithm cannot properly deal with such high packet loss. At such a high drop rate, many packets are more likely to be dropped consecutively which leads to whole syllables in the speech being dropped. Since the recovery method cannot predict what this syllable was, part of the speech is completely lost and cannot be recovered. We can then see that the reason that the improvement of recovery-by-re-initialization starts to drop as the drop rate exceeds 30% is because the frame loss recovery mechanism in G.729A cannot deal properly with such high packet loss.

Before continuing with further results, it is important to note that recovery-by-re-initialization is not a complete solution to the packet loss problem. Recovery-by-re-initialization only reduces the state error while the recovery mechanism standardized in G.729A is still responsible to keep the packet loss error as low as possible. Simply using recovery-by-re-initialization in a speech file with high packet drop rate will not lead to an output that is dramatically better than before since the error during the periods of packet loss will still exist. Instead, it will lead to subtle improvements in part of the speech and the reduction in the duration of some errors. The only way such dramatic improvement can be observed is if we limit ourselves to a case of isolated packet loss that has a severe state error. To illustrate the potential of the scheme in such cases, a few examples were chosen where re-initialization transformed a speech file with audible degradation to a speech file where no degradation is audible. The best way to understand this improvement is to listen to them. Since this is not possible here, the Maximum Perceptual Error with/without re-initialization will be given. Note that since these are isolated packet loss, the improvements cannot be noticed in the Perceptual Error per Sample since the period of improvement is quite small compared to the size of the speech file and will have minimal effect on this measure. The Maximum Perceptual Error is a better indication of the improvement for isolated packet loss. In the cases that follow, the re-initialization offsets were deliberately chosen to have the re-initialization occur three frames after the isolated packet loss. This is the optimal place to do re-initialization after a packet loss since the error in state#4 will have recovered by itself by that time. Since, as we have seen in section 5.3.4, an error in state #4 eventually leads to an error in state

#2, re-initialization after this error as recovered permits us to completely remove state error #2 after that point. These cases of dramatic improvements when isolated packet loss is performed are described in Table 5.5 that follows.

Table 5.5: Improvements for Isolated Packet Loss

Speech File	Frame # Dropped	Max. Perceptual Error no re-init.	Max. Perceptual Error re-init.
tom.wav	12	33.15	16.42
tom.wav	54	60.92	28.17
o_f01s25.wav	209	71.02	18.51
o_f01s25.wav	243	31.20	19.24

These results do show that re-initialization changes the value of the Maximum Perceptual Error from a value indicating degradation to a value that is quite close to that when no packet loss occurs. These results do show the potential that recovery-by-re-initialization has of dramatically improving the quality of the speech in periods of isolated packet loss.

5.5.5 Performance for Different Re-Initialization Intervals

So far, all experiments have been done using a re-initialization interval of 10. The reason we chose 10 was because it was the smallest re-initialization interval that caused minimal degradation in periods of no packet loss. It was assumed that choosing a higher re-initialization interval would lead to fewer improvements since less state error would be

removed. The only problem is that a smaller re-initialization interval results in an higher overhead since 76 extra bits must be transmitted for each re-initialization interval. In the following results, we will verify the improvements obtained if we choose a re-initialization interval of 5 or 20. A re-initialization interval of 5 was chosen even though such an interval does cause more degradation in the speech in periods of no packet loss. It is done in order to determine if the additional improvements of this scheme would warrant the extra degradation in periods of no packet loss and the additional overhead. The re-initialization interval of 20 was chosen to verify if it would be warranted to sacrifice some of the improvements for less overhead. A re-initialization interval of 5 requires a bit rate of 9.52 kb/s which represents an overhead of 19% compared to the 8 kb/s required when no re-initialization was performed. A re-initialization interval of 20 on the other hand requires a bit rate of 8.38 kb/s which represents an overhead of 4.75%. Since the overhead of re-initializing at every 10 frames is 9.5%, we want to examine the performance of the scheme if the overhead is increased or decreased by a factor of 2. In order to limit the number of tests to perform, the experiments were only done for a packet drop rate of 30% since this is where we experienced the best performance for re-initialization. The results in terms of Perceptual Error per Sample and Maximum Perceptual Error are illustrated in Figure 5.34 to Figure 5.37.

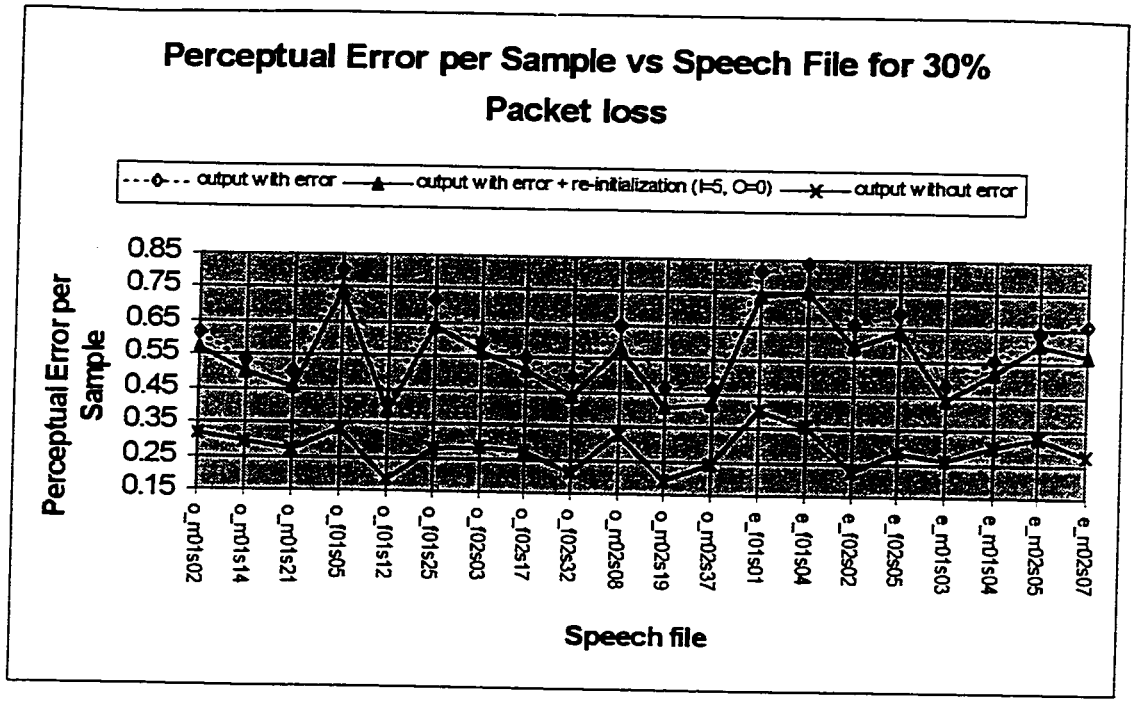

Figure 5.34: Perceptual Error per Sample for Re-Initialization Interval of 5

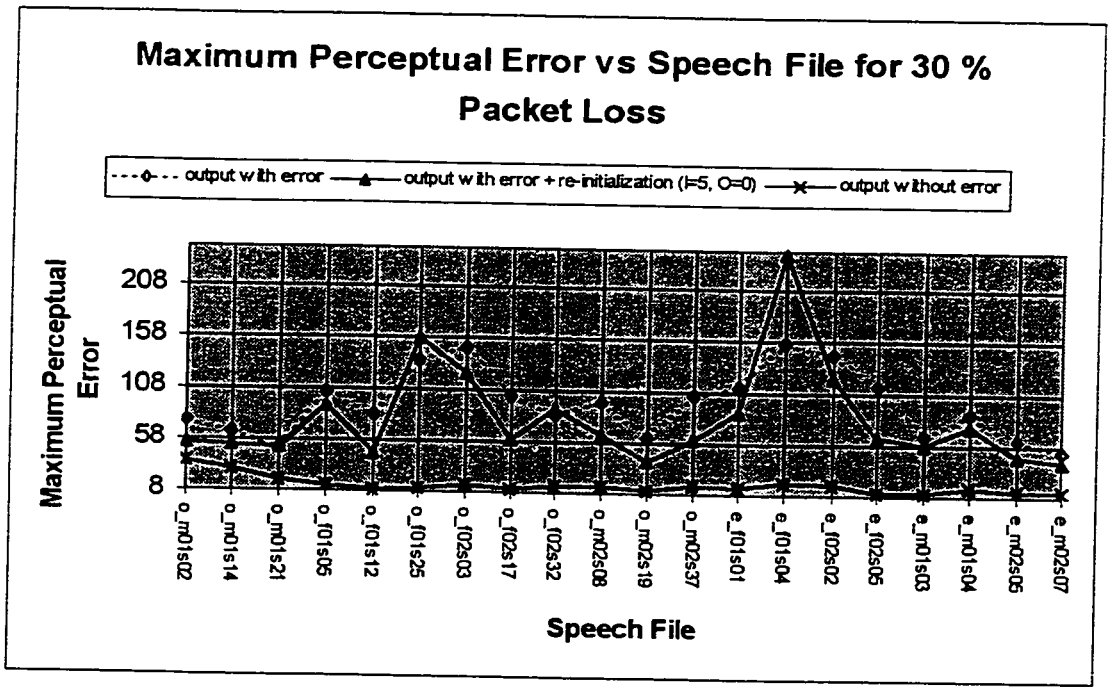

Figure 5.35: Maximum Perceptual Error for Re-Initialization Interval of 5

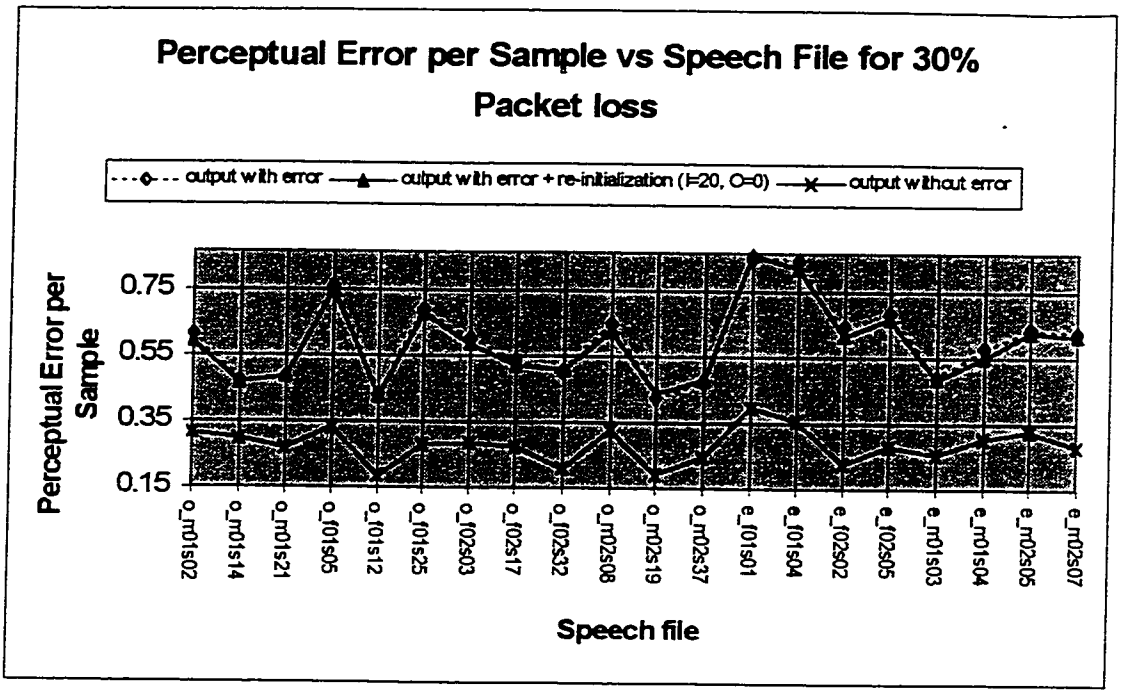

Figure 5.36: Perceptual Error per Sample for Re-Initialization Interval of 20

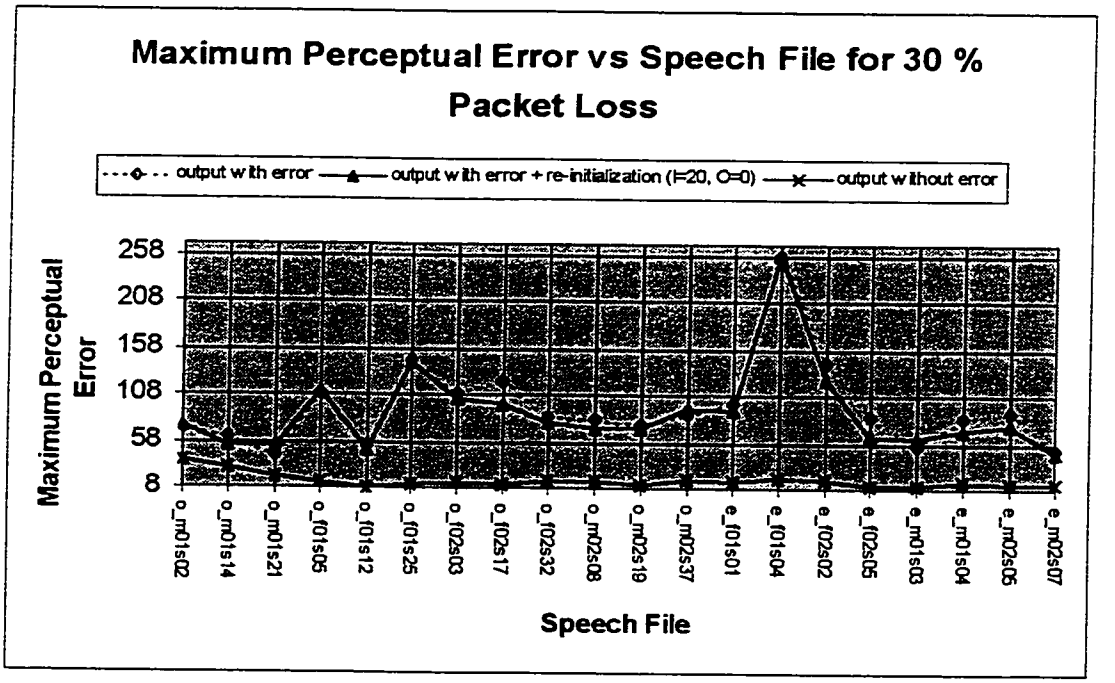

Figure 5.37: Maximum Perceptual Error for Re-Initialization Interval of 20

To properly compare these results with the those for a re-initialization interval of 10, a table of the average improvement for packet drop rate of 30% and a re-initialization interval of 5, 10, and 20 is given in Table 5.6.

Table 5.6: Average Improvements caused by Re-Initialization for Different Intervals and Packet Drop Rate of 30%

Re-Init. Interval	Average Improvement in Perceptual Error per Sample	Average Improvement in Maximum Perceptual Error
5	0.05412	14.11
10	0.03410	20.06
20	0.01139	6.16

This table and the figures of the results do show that reducing the re-initialization interval further does increase the average improvement of recovery-by-re-initialization in periods of packet loss. This was to be expected since re-initializing more often increases the probability that re-initialization will occur close to the packet loss. What we must now assess is if this improvement is worth the added overhead. The extra cost of re-initializing more often is twice the amount of overhead and a slightly more degraded speech quality in periods of no packet loss. The advantage is that we get an improvement by a factor of 1.42 in terms of Perceptual Error per Sample in periods of packet loss. As we are then paying twice the cost for less than twice the improvement, decreasing the re-initialization interval below 10 should only be done if the extra cost is justifiable.

Increasing the re-initialization interval to 20 results in a performance that is only 1/3 as effective as before in terms of Perceptual Error per Sample. Since the only benefit of this scheme is a reduction of the bit rate overhead by a factor of 2, such a reduction is not recommended.

5.6 Summary

In this chapter, we thoroughly examined the performance of the G.729A algorithm in periods of packet loss based on our results and analysis. It was shown that due to the memory present in the G.729A algorithm, the degradation to the speech due to the packet loss did not only last for the duration of the packet loss, but also extended for a certain period following the loss. This extra period of degradation following a packet loss was defined as the state error, which was due to the memory of the decoder no longer being synchronized with the memory of the encoder. It was shown through experimentation that the state error and the error during the packet loss caused approximately the same amount of degradation in speech quality. Since G.729A already includes measures to recover from a packet loss *during the loss period* and not for recovering from the state error that follows, it was deemed appropriate to investigate the addition of such a scheme.

To create a scheme to recover from state error, the states in the G.729A algorithm were thoroughly examined. It was shown that it took on average 32 frames after the last frame loss for the states between the encoder and decoder to be re-synchronized. It was also

shown that the number of frames needed to recover from a frame loss did not depend on how many frames were lost. Through extensive testing, it was shown that the most important state memory in terms of degradation it causes on the output speech if it is not properly synchronized was state #2 associated with the adaptive codebook vector.

The best way to recover from the state error was shown to be periodical re-initialization of state #2 to zero. This method required transmitting an extra 76 bits per re-initialization interval in order to properly protect the frame where re-initialization occurs and the one that follows it. Results presented in this section showed that this method did not cause any significant audible degradation in periods of no packet loss if a re-initialization interval of 10 or higher is chosen. Results in periods of packet loss showed the performance of the scheme for random packet loss rates from 5 to 50%. In all cases, recovery-by-re-initialization using an interval of 10 improved the speech quality as compared to the case where no re-initialization is performed. It was determined that the improvement caused by re-initialization augmented with the packet drop rate until a maximum improvement was obtained for a packet drop rate of 30%. For a packet drop rate between 40-50%, the improvements start to diminish since the quality of the speech becomes highly degraded. Other results were presented that showed how recovery-by-re-initialization can dramatically improve the speech quality for isolated packet loss. Finally, results were presented for re-initialization intervals of 5 and 20 for a packet drop rate of 30%. It was shown that augmenting or reducing the re-initialization interval from 10 was not warranted.

6 Discussion and Conclusion

6.1 Contributions

This section summarized the contributions of this thesis. These contributions are a good indication of how well the objectives set out for this thesis were met. The contributions will first be enumerated and then discussed in details. These contributions are:

- An in-depth study of VoIP and related subjects
- A comparative study of speech compression algorithms for VoIP
- Implementation of an objective measure to evaluate the subjective speech quality
- Analysis of the effects of packet loss on ITU-T G.729A
- Proposal of a new method to improve the speech quality of G.729A in periods of packet loss

6.1.1 Study of VoIP

This thesis presented a thorough review of Voice over IP and related subjects. By combining information from many sources and presenting it in a clear and efficient manner, the thesis established the current state of research on the subject. Chapter 2 provides a detailed description of VoIP while Chapter 3 describes speech compression. Sections 4.5, 4.6, and part of 4.7 in Chapter 4 also review topics of interest in speech compression for VoIP. Since understanding the problem is an important part of any solution, this is an important contribution of this thesis.

6.1.2 Comparative Study of Speech Compression Algorithms for VoIP

Sections 4.2, 4.3 and 4.4 in Chapter 4 establish a comparative study of speech compression algorithms for VoIP. This study uses known information on several different speech compression algorithms and the author's personal experience to determine which ones are better suited for VoIP. This represents a valuable contribution of this thesis, which, to our knowledge, has not been explicitly addressed in the literature. It's a consequence of this comparative study that ITU-T G.729A was determined to be a strong candidate for a speech compression algorithm to be used in VoIP.

6.1.3 New Method to Evaluate Speech Quality

In section 4.7 of chapter 4, it was determined that the existing objective methods to evaluate the speech quality were not sufficient for the needs of this thesis. This thesis required a method to evaluate the quality of speech that has suffered packet loss and existing methods were not appropriate. Subjective methods could have been used but these are quite expensive and not reproducible. We introduced a novel objective method to evaluate the subjective speech quality in section 4.7.4. This method referred to as Perceptual Error, uses principles from perceptual speech compression to calculate the audible difference between two different speech files. This method was proven to be quite effective in assessing the quality of different speech files in this thesis. Since such a

method could also be useful in evaluating the speech quality in other cases, this represents an important contribution of this thesis.

6.1.4 Analysis of the Effects of Packet Loss on ITU-T G.729A

In sections 5.2 and 5.3 of chapter 5, a thorough analysis of the effects of packet loss on ITU-T G.729A was presented. To our knowledge, this has not been done before. This thesis confirmed that a packet loss does not only cause an error during the period of packet loss, but also following it due to the differing states at the encoder and decoder (call this the state error). It also introduced the different states in G.729A and determined the time required for recovering from a state error for each of these states through experimentation. As well, the relative importance of these state errors to the error in the speech was determined through experimentation that led to the identification to the memory related to the past overall excitation as being the state that is the most important contributor to the error in the speech. This analysis also permitted us to identify that the synchronization between the receiver and the transmitter is the most important factor in reducing the state error. Since the principles developed here for G.729A can also be applied to determine the effects of packet loss on other model-based speech compression algorithm, this is a very important contribution of this thesis.

6.1.5 Improving Speech Quality of G.729A in Periods of Packet Loss

In sections 5.4 and 5.5 of chapter 5, a novel method to improve the speech quality of G.729A in periods of packet loss was introduced. It was shown that this method improves the speech quality of G.729A in periods of packet loss at a very low cost in terms of complexity and additional bandwidth. The method, recovery-by-re-initialization, is based on re-initializing to zero periodically the memory associated with the most important state. As well, it requires sending 76 extra bits of redundant information per re-initialization interval to better protect the speech against packet loss. If the preferred re-initialization interval of 10 frames is used, this represents only a 9.5% overhead in the bit rate requirement for the algorithm. Even though this thesis did not examine the extra bit error rate that would result from such an increased bit rate, we do not believe this to be significant since the bit error rate on data networks is usually quite low. Results given in this thesis demonstrated that this method consistently improved the speech quality for random packet loss of 5 to 50%. Some dramatic improvements can also be obtained in some cases of isolated packet loss. Since such a method can easily be modified to work with other model-based speech compression algorithms, this is a very important contribution.

6.2 Future Work

While doing this thesis, many topics requiring additional intention were identified. Since these topics were not related with the objectives of this thesis, they were not addressed and were left for future work. These topics are enumerated here:

- Investigate methods to turn on/off recovery-by-re-initialization as needed. Since this method requires some additional bandwidth and does not improve the speech in periods of no packet loss, it would be desirable to find a way to turn if off in such circumstances. But since the turning on/off would have to be synchronized at the encoder and receiver, such a method is not obvious and requires more attention.
- As described in the contributions of this thesis, the recovery-by-re-initialization method could be easily modified and applied to other model-based speech compression algorithms. Applying it to other popular model-based speech compression algorithms would be valuable work that could be done in the future.
- This thesis analyzed the effect of packet loss on G.729A. Doing such an analysis on other speech compression algorithms would be a worthy endeavor since not much work has been done in this field.
- This thesis did support the view that high-quality speech (CD bandwidth instead of telephone bandwidth) could be one of the features that make VoIP more attractive than conventional telephony. Currently, most perceptual-based compression algorithms have been developed for audio compression and are not very suitable for

VoIP. Developing new perceptual-based speech compression algorithms for VoIP would be important in permitting VoIP to attain such higher quality speech.

- In this thesis, we developed an objective speech quality measure to evaluate the subjective speech quality since existing methods were lacking. Even though the method presented in this thesis was very valuable for our needs, it would still need some improvements in order to be used as a universal measure for comparing the speech quality of different speech compression algorithms. Since a speech quality measure could also be used in VoIP to adapt the speech compression algorithm (do some changes when we find that our speech quality is getting low), developing new universal objective methods to evaluate the subjective speech quality would be a worthy endeavor.
- To properly adapt to the network conditions, a speech compression algorithm would need to be able to react to these different conditions. This is important since in periods of congestion, the situation might be remedied by having all speech compression algorithms lower their bit rate. This would permit a graceful degradation of the speech quality in periods of congestion. Current standards for speech compression are quite fixed in nature and do not permit much adaptation to the network conditions. It would be important to develop algorithms in the future that can easily adapt to such different network conditions.

Appendix A: Details of G.729A Algorithm

The following two block diagrams give further details on the decoding of compressed speech in G.729A. These figures are essentially more detailed version of Figure 3.7 presented in section 3.5.5.

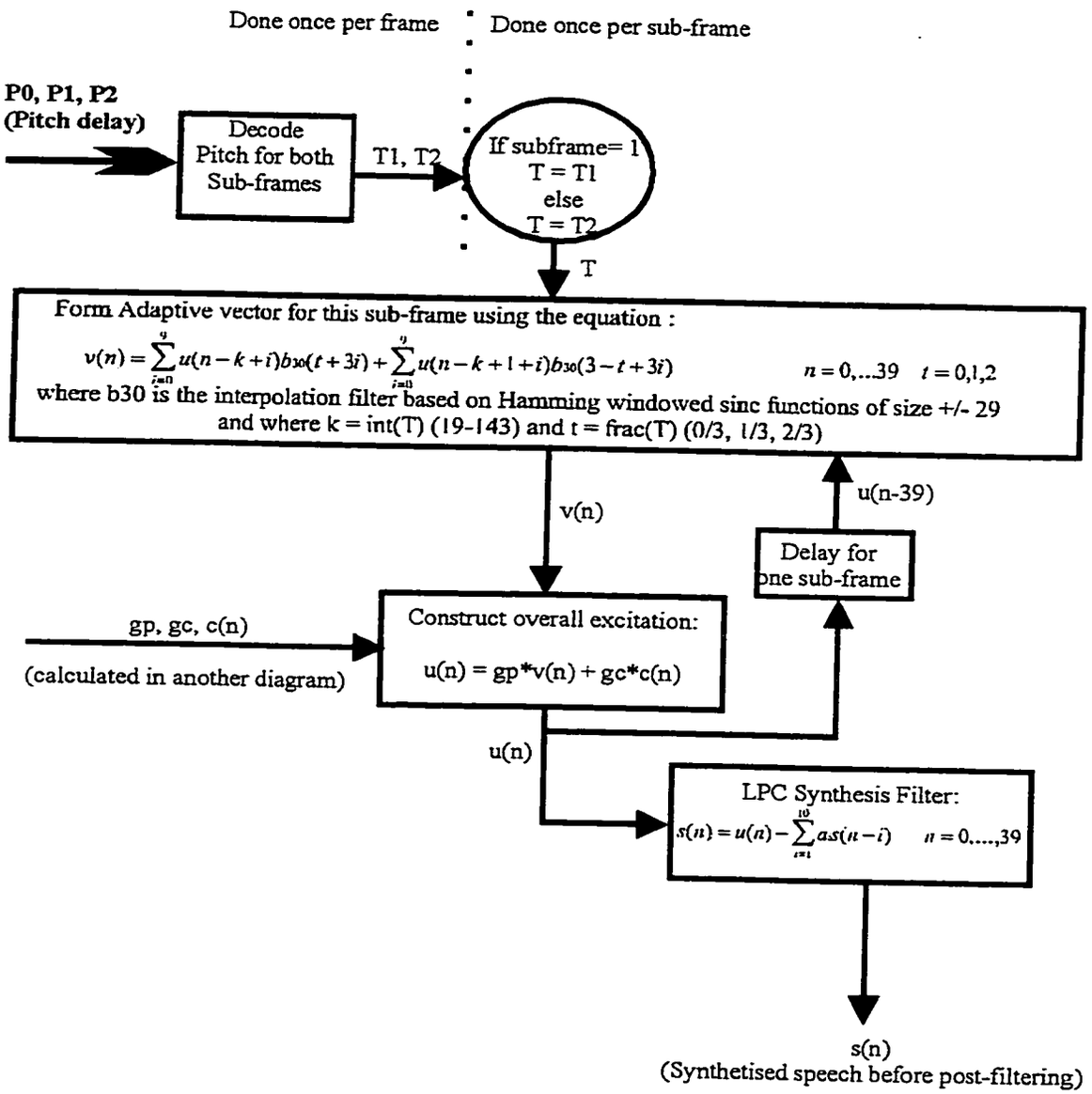

Figure A.1: Block Diagram of Adaptive Codebook Vector Decoding

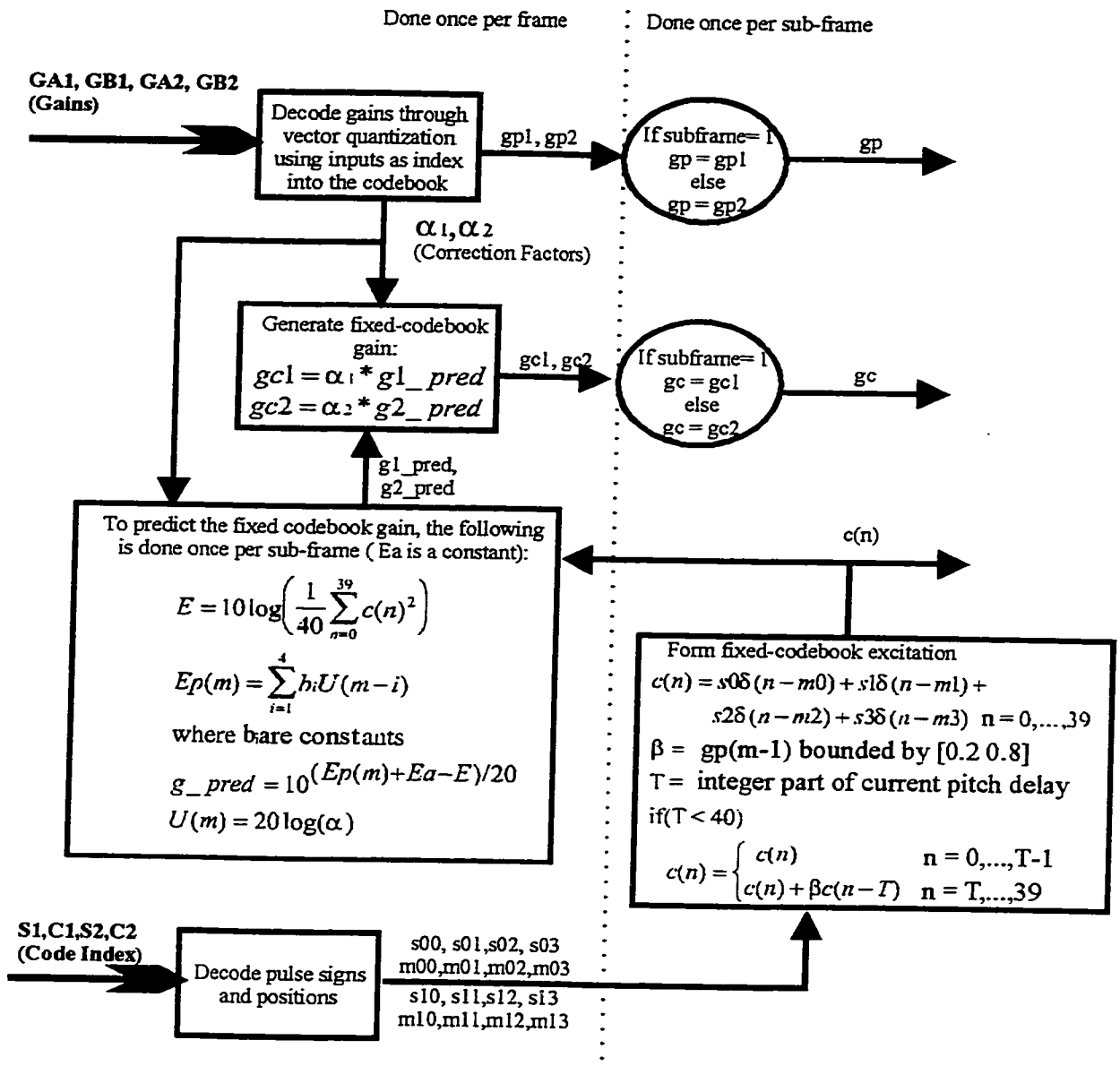

Figure A.2: Block Diagram of Gain and Fixed Codebook Vector Decoding

Bibliography

[1] Alwyn L. and Broom S. "Feasibility of Non-Intrusive Monitoring of Voice Call Quality in Internet Telephony", *British Telecommunication Engineering*, August 1999, pp.15-20.

[2] Beerends J. and Stemerdink J. "A Perceptual Speech Quality Measure based on a Psychoacoustic Sound Representation", *Journal of the Audio Engineering Society*, March 1994, pp. 115-123.

[3] Benyassine A. et al. "ITU-T Recommendation G.729 Annex B: A Silence Compression Scheme for use with G.729 Optimized for V.70 Digital Simultaneous Voice and Data Applications", *IEEE Communications Magazine*, September 1997, pp. 64-73.

[4] Bolot J.-C. and Vega-Garcia A. "Control Mechanisms for Packet Audio in the Internet", IEEE INFOCOM 1996, vol. 1, pp. 232-239.

[5] Bolot J.-C. et al. "Adaptive FEC-Based Error Control for Internet Telephony", *IEEE INFOCOM 1999*, vol. 3, pp. 1453-1460.

[6] Bosi M. et al. "ISO/IEC MPEG-2 Advanced Audio Coding", *Journal of the Audio Engineering Society*, vol.45, no. 10, October 1997, pp. 789-814.

[7] Bradenburg K. "Perceptual Coding of High Quality Digital Audio", *Applications of Digital Signal Processing to Audio and Acoustics*, Kluwer Academic Press, 1998, pp. 39-84.

[8] Chen R.-X., Chen L.-G. and Tsai T.-H., "An I-Phone System Design and Implementation with a Portable Speech Coding Coprocessor", *IEEE Transactions on Consumer Electronics*, vol. 43, no. 3, November 1997, pp. 1262-1269.

[9] Chen Y. and Chen B. "Model-Based Multirate Representation of Speech Signals and Its Application to Recovery of Missing Speech Packets", *IEEE Transactions on Speech and Audio Processing*, vol. 5, no. 3, May 1997, pp. 220-231.

[10] Cox R. V. "Three New Speech Coders from the ITU Cover a Range of Applications", *IEEE Communications Magazine*, September 1997, pp. 40-47.

[11] Dalgic I. et al. "True Number Portability and Advanced Call Screening in a SIP-based IP Telephony System", *IEEE Communication Magazine*, vol. 37, no. 7, July 1999, pp. 96-101.

[12] DeLeon P. and Sreenan C. "An Adaptive Predictor for Media Playout Buffering", *Proceeding ICASP 1999*, vol. 6, pp. 3097-3100.

[13] Erdol N. "Recovery of Missing Speech Packets using the Short-Time Energy and Zero Crossing Measurements", *IEEE Transactions on Speech and Audio Processing*, vol. 1, no. 3, July 1993, pp. 295-303.

[14] Fielder L. D. et al. "AC-2 and AC-3: Low-Complexity Transform-Based Audio Coding", *Papers on Digital Audio Bit Rate Reduction*, 1997, pp. 54-72.

[15] Gelfand S. A. *Hearing: An Introduction to Psychological and Physiological Acoustics*, Third Edition, Marcel Dekker Inc., New York, 1998

[16] Gilloire A. "Performance Evaluation of Acoustic Echo Control: Required Values and Measurement Procedures", *Annales de Télécommunications*, vol. 49, 1994, pp.368-372.

[17] Goodman B. "Internet Telephony and Modem Delay". *IEEE Network*, May/June 1999, pp. 8-16.

[18] Hansler E. "The Hands-free Telephone Problem: an Annotated Bibliography Update", *Annales de Télécommunications*, vol. 49, 1994, pp. 360-367.

[19] Hassan M. and Nayandoro A. "Internet Telephony: Services, Technical Challenges, and Products", *IEEE Communications Magazine*, April 2000, pp. 96-103.

[20] Haykin S. *Adaptive Filter Theory: Third Edition*, Prentice Hall, Englewood Cliffs, New Jersey, 1996.

[21] Haykin S. *Communication Systems : Third Edition*, John Wiley & Sons, Toronto, 1994.

[22] Homer J. "A Review in the Developments in Adaptive Echo Cancellation for Telecommunications", *Journal of Electrical & Electronics Engineering Australia*, September 1998, pp. 149-164.

[23] Hoshi T. et al. "Voice Stream Multiplexing between IP Telephony Gateways", *IEICE Transactions on Information and Systems*, April 1999, pp. 838-845.

[24] http://audioquality.denver.co.us

[25] *ITU-T G.723.1: Dual Rate Speech Coder for Multimedia Communications Transmitting at 5.3 and 6.3 kb/s*, International Telecommunications Union, March 1996.

[26] *ITU-T G.729 Annex A: Reduced Complexity 8 kb/s CS-ACELP Speech Codec*, International Telecommunications Union, November 1996.

[27] *ITU-T G.729: Coding of speech at 8 kb/s using conjugate-structure algebraic-code excited linear-prediction (CS-ACELP)*, International Telecommunications Union, March 1996.

[28] *ITU-T P Supplement 23: ITU-T Coded-Speech Database*, International Telecommunications Union, February 1998.

[29] *ITU-T P.800: Methods for objective and subjective assessment of quality*, International Telecommunications Union, August 1996.

[30] *ITU-T P.830: Subjective performance assessment of telephone-band and wide-band digital codecs*, International Telecommunications Union, February 1996.

[31] *ITU-T P.861: Objective Quality Measurement of Telephone-Band (300-3400 Hz) Speech Codecs,* International Telecommunications Union, February 1998.

[32] Jain A. K., *Fundementals of Digital Image Processing*, Prentice Hall, Englewood Cliffs, New Jersey, 1989.

[33] Juang B.H. and Tsuhan C. "The Past, Present and Future of Speech Processing", *IEEE Signal Processing Magazine*, vol. 15, no. 3, May 1998, pp.24-48.

[34] Junqua J.-C. and Haton J.-P., *Robustness in Automatic Speech Recognition: Fundamentals and Applications*, Kluwer Academic Publishers, Boston, 1996.

[35] Kahrs M. "The Past, Present, and Future of Audio Signal Processing", *IEEE Signal Processing Magazine,* vol. 14, no. 5, September 1997, pp. 30-57.

[36] Kondoz A. M., *Digital Speech: Coding for Low Bit Rate Communications System*, John Wiley & Sons, Toronto, 1994.

[37] Kostas T. et al. "Real Time Voice over Packet-Switched Networks", *IEEE Network*, January/February 1998, pp. 18-27.

[38] Lincoln B. "An Experimental High Fidelity Perceptual Audio Coder", 1998, http://www-ccrma.stanford.edu/~bosse/proj/node1.html

[39] Luoma M. et al. "Quality of Service for IP Voice Services – Is it Necessary?", *Internet Routing and Quality of Service*, Proceedings of SPIE 1998, pp. 242-252.

[40] Martens J. et al. "Voice over IP: The Impact of RSVP", *Internet Routing and Quality of Service*, Proceedings of SPIE 1998, pp. 391-397.

[41] Metz C. "RSVP: General-Purpose Signaling for IP", *IEEE Internet Computing*, May/June 1999. pp. 95-99.

[42] Mier E. et al. "Voice-over-IP Gateways: Sounding Good", *Business Communications*, February 1998, pp. 23-29.

[43] Minoli D. and Minoli E. *Delivering Voice over IP Networks*, Wiley Computer Publishing, Toronto, 1998.

[44] Montminy C. and Aboulnasr T. "Improving the Performance of ITU-T G.729A for VoIP", paper #173, to appear in *International Conference on Multimedia & Exposition 2000 (ICME 2000)*, July 30 - August 2 2000.

[45] Murgia C. et al. "Very Low Delay and High Quality Coding of 20Hz-15kHz Speech Signals at 64 kbit/s", *Proceedings of ICLSP 1996*, vol.1, pp. 302-305.

[46] Noll P. "MPEG Digital Audio Coding", *IEEE Signal Processing Magazine*, September 1997, pp. 59-81.

[47] Paxson V. and Floyd S. "Why we don't Know how to Simulate the Internet", *Proceedings of the 1997 Winter Simulation Conference*, 1997, pp. 1037-1044.

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

[49] Perkins M. et al. "Characterizing the Subjective Performance of the ITU-T 8 kb/s Speech Coding Algorithm – ITU-T G.729", *IEEE Communications Magazine*, September 1997, pp. 74-81.

[50] Perkins M. et al. "Speech Transmission Performance Planning in Hybrid IP/SCN Networks", *IEEE Communications Magazine*, July 1999, pp. 126-131.

[51] Podolsky M, Roner C. and McCanne S. "Simulation of FEC-based Error Control for Packet Audio on the Internet", *IEEE INFOCOM 1998*, pp. 505-515.

[52] Polyzois C. et al. "From POTS to PANS: A Commentary on the Evolution of Internet Telephony", *IEEE Internet Computing*, May/June 1999, pp. 83-91.

[53] Proakis J. G. and Manolakis D. G. *Digital Signal Processing: Principles, Algorithms, and Applications*, Third Edition, Prentice-Hall, Upper Saddle River, New Jersey, 1996.

[54] Ramachandran R. and Mammone R., *Modern Methods of Speech Processing*, Kluwer Academic Press, 1995.

[55] Ramjee R. et al. "Adaptive Playout Mechanisms for Packetized Audio Applications in Wide-Area Networks", *IEEE INFOCOM 1994*, vol. 2, pp. 680-688.

[56] Rao K. R.and Huang J.J., *Techniques & Standards for Image, Video & Audio Coding*, Prentice Hall, Toronto, 1996.

[57] Rinde J. "Telephony in the Year 2005", *Computer Networks*, February 1999, vol. 31, no.3, pp. 157-168.

[58] Rizzetto D. and Catania C. "A Voice over IP Service Architecture for Integrated Communications", *IEEE Internet Computing*, May/June 1999, pp. 53-62.

[59] Rosenberg J. and Schlzrinne H. "Internet Telephone Gateway Location", *IEEE INFOCOM '98*, pp. 488-496.

[60] Rosenberg J., "G.729 Error Recovery for Internet Telephony", Columbia University report, Spring 1997, http://www.cs.columbia.edu/~jdrosen/e6880/index.html

[61] Salami R. et al. "Design and Description of CS-ACELP: A Toll Quality 8 kb/s Speech Coder", *IEEE Transactions on Speech and Audio Processing*, vol. 6, no. 2, March 1998, pp. 116-130.

[62] Salami R., "ITU-T G.729 Annex A: Reduced Complexity 8 kb/s CS-ACELP Codec for Digital Simultaneous Voice and Data", *IEEE Communications Magazine*, September 1997, vol. 35, no.9, pp.56-63.

[63] Schlzrinne H. and Rosenberg J. "Internet Telephony: Architecture and Protocols – an IETF Perspective", *Computer Networks*, February 1999, pp.237-255.

[64] Schulzrinne H. "Converging on Internet Telephony: Service for Telecom Version 2", *IEEE Internet Computing*, May/June 1999, pp. 40-43.

[65] Schulzrinne H. et al., RFC 1889, *RTP: A Transport Protocol for Real-Time Applications*, January 1996.

[66] Sinla D. et al. "The Perceptual Audio Coder (PAC)", *The Digital Signal Processing Handbook*, CRC/IEEE Press, 1998, pp. 42-1:42-17.

[67] Soulhi S. "Telephony over Packet Networks", *IEEE Canadian Review*, Winter 1999, pp.4-8.

[68] Srinivasan P. and Jamieson L. H., "High Quality Audio Compression using an Adaptive Wavelet Packet Decomposition and Psychoacoustic Modeling", *IEEE Transactions on Signal Processing*, vol.46, no. 4, April 1998, pp. 1086-1093.

[69] Stallings W. *Data and Computer Communications*, Fifth Edition, Prentice Hall, Upper Saddle River, New Jersey, 1997.

[70] Su D. et al. "Investigating Factors Influencing QoS of Internet Phone", *IEEE International Conference on Multimedia Computing and Systems*, vol. 2, pp. 541-546.

[71] Sundstrom K. et al. "Internet Telephony Compression Algorithms", *IEEE WESCANEX 1997*, pp. 13-18.

[72] Toga J. and Ott J. "ITU-T standardization activities for interactive multimedia communications on packet-based networks: H.323 and related recommendations", *Computer Networks*, February 1999, vol.31, no. 3, pp.205-223.

[73] Vetterli M. and Kovacevic J. *Wavelets and Subband Coding*, Prentice Hall, Englewood Cliffs, New Jersey, 1995.

[74] Wah B. and Lin D. "Transformation-Based Reconstruction for Real-Time Voice Transmissions Over the Internet", *IEEE Transactions on Multimedia*, vol. 1, no. 4, December 1999, pp. 342-351.

[75] Wasem O. et al. "The Effect of Waveform Substitution on the Quality of PCM Packet Communications", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol.36, no.3, March 1988, pp. 342-347.

[76] Wicker S. *Error Control Systems for Digital Communication and Storage*, Prentice Hall, Upper Saddle River, New Jersey, 1995.

[77] Yang W. and Vantorno R. "Comparison of two Objective Speech Quality Measures: MBSD and ITU-T Recommendation P.861", *1998 IEEE Second Workshop on Multimedia Signal Processing*, pp. 426-431.

[78] Yensen T. et al. "Determining Acoustic Round Trip Delay for VoIP Conferences", *1998 IEEE Second Workshop on Multimedia Signal Processing*, pp.161-166.