Lost VOIP Packet Recovery in Active Networks

Yousef Darmani

M. Eng. Sc. Sharif University of Technology, Tehran, Iran

Thesis submitted for the degree of

Doctor of Philosophy

in

Department of Electrical and Electronic Engineering

The University of Adelaide

Supervisors:

Professor Langford White

Mr. Michael Liebelt

June 2004

Contents

List of Figures

List of Tables

Abstract

Current best-effort packet-switched Internet is not a perfect environment for real-time applications such as transmitting voice-over it (Voice Over Internet Protocol or *VOIP*). Due to the unlimited concurrent access to the Internet by users, the packet loss problem cannot be avoided. Therefore, the VOIP based applications encompass a problem namely: ”voice quality degradation caused by lost packets”.

The effects of lost packets are fundamental issues in real-time voice transmission over the current unreliable Internet. The dropped packets have a negative impact on voice quality and concealing their influences at the receiver does not deal with all of the drop consequences. It has been observed that in a very lossy network, the receiver cannot cope with all the effects of lost packets and thereby the voice will have poor quality. At this point the *Active Networks*, a relatively new concept in networking, which allows users to execute a program on the packets in active nodes, can help VOIP regenerate the lost packets, and improve the quality of the received voice. Therefore, VOIP needs special voice-packing methods. Based on the measured packet loss rates, many new methods are introduced that can pack voice packets in such a way that the lost packets can be regenerated both within the network and at the receiver. The proposed voice-packing methods could help regenerate lost packets in the active nodes within the network to improve the perceptual quality of the received sound. The packing methods include schemes for packing samples from low and medium compressed sample-based codecs (PCM, ADPCM) and also include schemes for packing samples from high compressed frame-based codecs (G.729).

Using these packing schemes, the received voice has good quality even under very high loss rates. Simulating a very lossy network by NS-2 and testing the regenerated voice quality by an audience showed that significant voice quality improvement is achievable by employing these packing schemes.

Statement of Originality

I hereby declare that this work contains no material which has been accepted for the award of any other degree or diploma in any university or other tertiary institution and to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text.

I give consent to this copy of my thesis, when deposited in the University Library, being available for loan and photocopying.

Mohammad Yousef Darmani
1 June 2004

Acknowledgments

First of all I would like to give thanks to the Almighty for His mercy upon me in guiding me through this world of unknowns.

This thesis exists because of my parents whom I admire and respect above all else. Their encouragement and support, which made this all possible, go beyond words.

I owe a great dept of gratitude to my supervisors who have given me the greatest support during my postgraduate research. Professor Langford White has been always there to answer my questions and guide me with his valuable advice. Mr. Michael Liebelt gave me confidence and smoothed the path of my research.

My special thanks go to my wife, Maryam, who had to leave her higher education to provide a comfortable environment for me. Also, I would like to thank my son Amirhossein who did his best to let me do my work. They have shown remarkable support, understanding and patience in being taken from their home country to Australia, and in learning what it means to live like a graduate student.

Finally, I wish to thank all others who, directly or indirectly, assisted me in my Ph.D. including my brothers and their families, my wife's family and all of my friends.

Publications

1. **Y. Darmani**, L. White and M. Liebelt, “Two Novel Voice Packing Schemes for VOIP to Recover Lost Packets Using Active Networks”, *Submitted to IEEE/ACM Transactions on Networking*, May 2004.
2. **Y. Darmani**, L. White and M. Liebelt, “New Methods for Lost Voice Packet Recovery in Active Networks”, *Proceedings of the 11th IEEE International Conference on Networks, ICON2003*, Sydney, Australia, pp. 57-62, September 2003.
3. **Y. Darmani**, L. White and M. Liebelt, “A New Voice-packing Scheme to Recover Lost Packets using Active Networks”, *Proceedings of the International Symposium on Telecommunication, IST2003*, Isfahan, Iran, pp. 95-99, August 2003.

Chapter 1

Introduction

1.1 Internet telephony

The Internet represents a powerful communication network spanning the whole world. Improving its overall abilities by increasing its bandwidth and decreasing its transmission delay changed it from a scientific research environment to a mass medium for everyday use. The Internet tempted the researchers to use it for something it had not been designed for: real-time multimedia communication. Research efforts have been conducted towards sending voice over Internet Protocol (IP) based networks. The IP telephony is attracting attention as a platform for the new generation of multimedia communication networks. The new IP telephony uses the current Public Switched Telephone Network (PSTN) to make connections among conversation participants. Here are some important motivating points for sending voice over the Internet.

- Cost: Making a long-distance call by the Internet is much cheaper compared to PSTN. This is due to the usual flat-rate charge that the Internet Service Providers (ISP) currently charge for their services [65]. In this case, only the marginal networks (at both sides of a call) use PSTN and the users have to pay for them.

- Signalling: A PSTN phone can only carry Dual Tone Multi Frequency (DTMF) signals in-band (in the same voice channel). In contrast, IP based calls can send signals out-of-band as a separate set of IP packets. Consequently, the end point device can act as an intelligent device providing sophisticated services [78].

- Other services: Using the Internet, it is feasible to have other services besides the voice in a conversation such as sending text messages or generating an email for unanswered calls.

Of all these points the first point, cost, strongly pushes the use of the Internet as a voice transmission medium. In contrast to traditional telephony, which is based on a fixed-

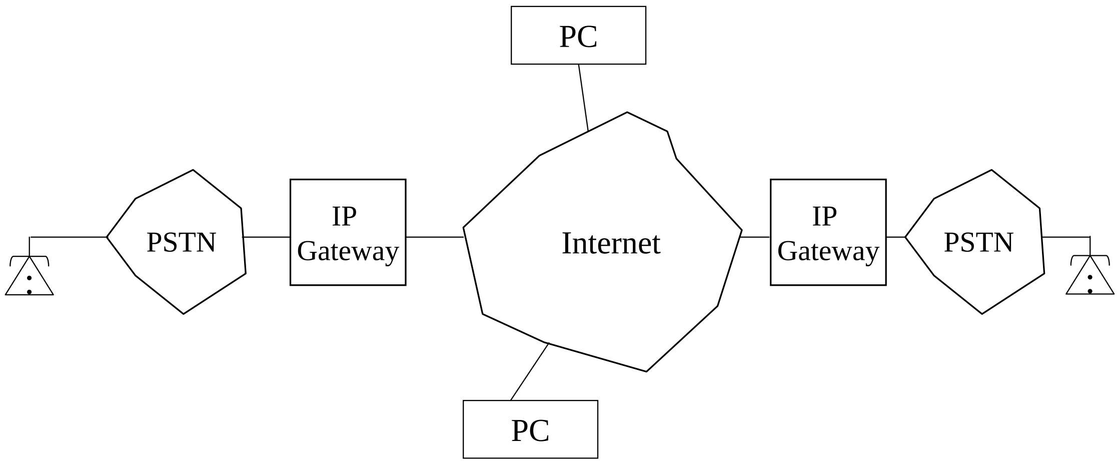

Figure 1.1: IP telephony configuration.

bandwidth, circuit-switched, end-to-end connection, IP telephony is based on IP networking, which offers the potential to have more services than voice. Besides that, VOIP offers different connection types for its users:

- voice communication between PCs using the Internet telephony software,
- voice communication between an existing telephone and a PC through the Internet, and
- voice communication between existing telephones via the Internet.

Figure 1.1 shows the configuration of the IP telephony in the current Internet.

New technologies help the Internet become faster. It is a best-effort, packet-switched network. Although transmitting voice over the Internet has many obvious benefits, its speech quality still is not comparable to the quality achieved by PSTN for many important reasons, such as delay and lost packets. Real-time applications place heavy demand on the network, therefore delay and losses occur. VOIP suffers from packet loss problem and its voice quality demotes drastically with lost packets. Most lost packets are due to congestion, bit error or delay. To mitigate the problem, many methods were introduced to recover the lost packets. In the current Internet, lost packet recovery is done on an end-to-end basis by high-layer protocols such as TCP, which retransmits the lost packets. Obviously, this lost packet recovery strategy cannot be used for VOIP due to its excessive delays. Voice quality degradation caused by packet loss, motivates researchers to find special voice-packing schemes to pack, transmit and regenerate voice for VOIP based applications.

To transmit voice samples over the Internet several techniques were introduced to encode and pack the samples. Most of them try to compress the voice, exploit the redundancy of the speech and reduce the data rate. Conventional coding schemes simply try to send voice bit streams to the receiver. Therefore, there is no way to cope with all the effects of lost packets and the voice will have poor quality. Since they do not take packetization into account, they do not have a specific compensating scheme when a packet is lost, so they cannot be used under very lossy networks. In short, these packing schemes aim to reduce congestion in the network by compressing the data but they do not offer an effective compensating routine when a packet is lost. Although VOIP uses different protocols, it has no specific mechanism to regenerate lost packets nor to cope with the drop problem [109]. In contrast with the conventional voice-coding schemes, many methods are introduced that are specially designed to send voice packets through the Internet. They are structured to pack voice in such a way that it can tolerate some lost packets.

1.2 Problem statement

The current Internet design is based on the best-effort principle, which does not guarantee the non-delay and loss-less environment needed for real-time applications such as VOIP. Voice is delay-sensitive by nature, thus its end-to-end delay is restricted. Consequently, besides the real lost packets, all packets delayed over a certain time limit are considered lost packets. In contrast with the file transmission over the network, voice samples do not need to be 100% precise at the receiver to generate acceptable sound. In view of this, the receiver can estimate some lost samples by other available samples.

The purpose of this thesis is to introduce new methods to pack voice samples in such a way that the lost packets can be regenerated by the receiver or within the network. New coding schemes aim to improve the received voice quality in a lossy transmission environment. The schemes are based on theoretical aspects of voice properties.

This thesis tries to introduce new algorithms to encode and pack the voice samples at the sender as well as to decode and regenerate the voice at the receiver. The algorithms can be implemented in active nodes as well as at the end nodes and they allow the active nodes to do ’on-the-fly’ regeneration of some lost packets. They are simple to implement and quick in terms of processing time. Moreover, they do not add intolerable redundant data or delay in the real time voice transmission environment. The schemes improve the Quality of Service (QoS) emphasis on packet drops and compensate for those effects. The schemes work on packing speech based on its specifications and on the drop behaviour of the Internet. Using these packing methods, active networks can help VOIP regenerate lost packets in each active node and help improve the quality of sound at the receiver. The regenerated voice is evaluated by subjective tests. In general, it is tried to use both source and channel coding methods to pack voice packets so as to cope with the packet loss problem.

1.3 Thesis outline and methodology

This thesis proposes many new voice-packing schemes using the following steps.

- Using results of network measurements (own tests and results available in the literature), a brief review of packet loss, its nature and reasons are studied. Methods and metrics to model the lost packets are considered and evaluated. A loss model is chosen for all the simulations (Chapter 2).
- The nature of speech and its properties are investigated and periodic and non-periodic voice and their specifications are studied (Chapter 3).
- Methods to cope with packet loss problem are stated and a method is selected for the simulations (Chapter 3).
- Different protocol used for VOIP are investigated and using a patch on RTCP, the RTP/UDP/IP is selected (Chapter 3).
- Active network, its structure and its node specifications are studied and it is mathematically proved that an active node can improve the voice quality in VOIP (Chapter 4).
- A very lossy network based on measured Internet behaviour is established, an active node is implemented and the whole active path is used for all the simulations. The path contains an end-to-end loss model with adjustable parameters. The implemented active node and the selected simulation environment allow control of the loss pattern (Chapter 4).
- Detached Samples Packing Method (DSPM) as a voice-packing scheme based on speech properties for uncompressed voice samples is introduced (Chapter 5).
- Detached Adaptive Differential Pulse Code Modulation (DADPCM), a voice-packing method based on DSPM and ADPCM algorithm, is explained (Chapter 6).
- Different methods to choose ”important” samples that help improve the voice quality of the regenerated lost samples are introduced (Chapter 6).
- Detached Adaptive Differential Pulse Code Modulation 5 (DADPCM5), a voice-packing method suitable for very lossy networks that can tolerate burst loss packets is presented (Chapter 7).
- Discrete Fourier Transform (DFT) of voice segments, its specifications and many accomplished experiments based on transmitting DFT coefficients of voice samples, accompanying their results are explained (Chapter 8).

- Detached Fourier Coefficients Packing Method (DFCPM) as a voice-packing method based on sending the DFT coefficients of a voice segment and regenerating the voice by applying the inverse Fourier transform at the receiver is introduced (Chapter 8).

- Enhanced DFCPM (EDFCPM), an enhanced version of DFCPM which carries two series of data that improve the voice quality of regenerated lost samples is presented (Chapter 8).

- Enhanced Code Excited Linear Prediction (ECELP) as three voice-packing schemes that send the extracted parameters of the Linear Prediction (LP) model of voice are introduced and the way they choose the important samples are explained (Chapter 9).

- Complimentary Adaptive Differential Pulse Code Modulation (CADPCM) as an improved version of Hardman's algorithm that inserts redundant data based on the voice specifications is introduced (Chapter 10).

The last Chapter summarizes the thesis.

Chapter 2

Packet loss models and metrics

Internet technology has increased interest in developing high quality, Internet-based multimedia applications such as VOIP, teleconferencing and telemedicine. Quality of received data is the key to the success of such applications. Packet loss is one of the most important factors determining the quality experienced by delay-sensitive multimedia applications. Increasing loss causes drastic degradation in the quality of received data in such applications. This Chapter aims to provide understanding of the voice packet loss rate experienced at the receiver. It discusses why packet losses occur on the Internet and it explains some loss models and describes the chosen model for simulations.

2.1 Packet loss

Generally, when a packet does not arrive at the receiver or in the case of real time application, if it arrives late, it is classified as a lost packet. The latency is defined later. Boyce and Gaglionello [11] measured the loss rate and unsurprisingly, they observed higher loss rates during weekday business hours and lower rates in the middle of the night. They measured packet loss rates ranged from 0.018% to 100% for a 5-minute sample during the day. They concluded that the detail of loss is hidden if only the average loss rate of 7.123% is considered. To avoid having a simple but inaccurate loss model, different causes of lost packets are examined and it is tried to find a good model to describe and simulate lost packets on the Internet.

2.1.1 Causes of packet loss

Three distinctive causes for packet loss are identified.

- Congestion: When the input load is higher than the forwarding speed of the routers, the packets are temporarily stored in the queue of the routers, waiting for their turn to be processed and forwarded to the other nodes. As the queue has finite size for holding packets, when it is full, any new, incoming or even some of the queued packets are dropped (based on the policy of the router). Therefore, packets are lost due to the lack of queue space in network nodes which is a consequence of limited queue size and congestion in the network. Due to the expiry of Time-To-Live (TTL) in the header part of the IP packets, Nagle [71] showed that even with an infinite queue size, packet loss could not be eliminated. As the congestion is responsible for lost packets, many network researchers tried to measure its role in the network. Jain [51] explained the general behaviour of a network responding to load. He stated that when the load is small, the throughput of the network increases as the load increases. When the incoming rate is higher than the network capacity, a queue starts to form within some nodes in the network, which is the starting point of congestion. During congestion, packets may either be dropped from the tail of the queue or be randomly dropped from within the queue, when the average queue size exceeds a threshold limit [39]. Moreover, he suggested measuring the round trip delay of packets at the congestion starting point (T_c). When a packet is lost, the measured round trip delay is compared with T_c. If T_c is lower, the loss is probably due to the congestion and if T_c is higher, the loss is probably due to other reasons.

- Bit error: Packet loss due to noise often happens in a transmission system with lossy links. This issue is very significant in wireless radio links, particularly those with a

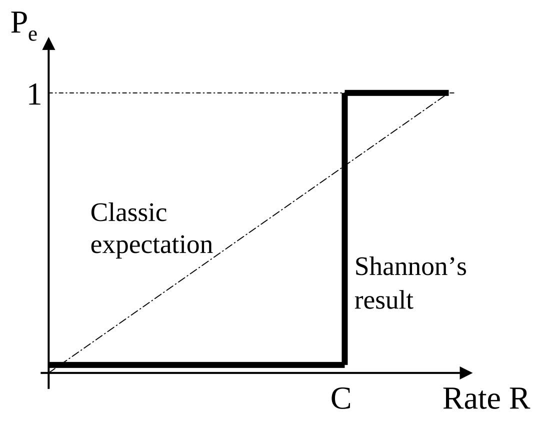

Figure 2.1: Shannon's error characteristics.

long propagation delay, such as satellite links [88]. Based on Shannon's theory, when transmission rate (R) is less than the channel capacity (C), the probability of having error (P_e) is low (which happens in the wired or optical links). When the transmission rate is higher than the channel capacity, the error rate is quite high (which is the case for some wireless and satellite links) [15]. Figure 2.1 shows the relation between P_e and R based on that theory.

This type of error is classified as bit error. Typical bit error rates in the satellite networks are in the order of 1 error per 10^7 bits transmitted or less [2].

- Out of order and Delay: Although Internet routers employ FIFO queueing, any time a route changes, if the new route offers a lower delay than the old one, then reordering can occur [69]. Like the lost packet rate, the out of order packet rate varies with time of day. [11] shows that a majority of the out of order delays are only by a single packet. Reordering the packets requires buffering the data prior to decoding, but using a play-out buffer adds to the delay in communication, which is not acceptable in most real-time applications such as VOIP. Therefore, the out of order packets are classified as lost packets. In any case when a packet arrives later than its play-out time, it is useless to send it to the output stage and so it should be considered a lost one. Network delay consists of propagation delay, processing delay, transmission delay and queueing delay caused by congestion [54]. Congested networks cause most of the delayed packets. If the network delay is not constant, packets are subject to Jitter and the receiver may experience gaps that interrupt the continuous play back of the voice. Dempsey and his colleagues in their novel S-ARQ scheme proposed applying an adaptive *Control Time* before playing back the first packet to compensate the Jitter effect [25].

A compensating scheme in which the receiver can overcome the loss problem should be dependent on the loss nature. Therefore, efforts were made to judge the cause of losses.

Some algorithms try to differentiate between loss caused by congestion and loss caused by bit error.

- Liu and his colleagues showed that congestion and non-congestion related losses are difficult to differentiate by round trip time under high load because the bottleneck queue occupancy shows less variation under very high network utilization [61].
- Biaz and Vaidya [4] proposed an algorithm based on comparing arrival time of in-sequence and out-of-order packets at the receiver.
- The Spike scheme [104] is based on measuring the Relative One-way Trip Time (ROTT) that a packet takes to travel from the sender to the receiver.
- The ZigZag scheme [16] classifies losses based on the number of losses and on the difference between the ROTT of one packet and the average ROTT of all packets.

In spite of all these algorithms, there is no clear and decisive way to classify lost packets in all networks. However, it is possible to say that if the lost packets are separated and the arrived packets are not delayed, the loss is due to the bit error problem [89], and when some consecutive packets are lost (burst loss) and when the arrived packets are delayed, the loss is due to the congestion problem.

2.1.2 Loss rate and packet length

Currently in the Internet, a transmitted IP packet is either received entirely or lost entirely. If a packet is longer than a specific threshold named Maximum Transfer Unit (MTU), it is divided into fragments [21]. If any of those fragments is lost (due to any reason), the whole IP packet is lost. Therefore, one can expect that packets shorter than MTU will have lower loss rates than longer packets (MTU is 1500 bytes for Ethernet segments). From [21], it can be concluded that packet loss rate is proportional to the closest integer to the packet size divided by MTU.

2.1.3 Statistical information about packet loss

In order to find the packet loss rate in the current network, some measurements were undertaken. In a practical test, 1000 UDP packets of length 400 bytes and 50 ms interval from The University of Adelaide in Australia were sent to some sites in China, Malaysia and Iran. Each packet was treated as a 50 ms message that had 400 byte voice samples. These packets were sent to the echo ports of the above sites and during each 50-second test, the received packets were logged. The tests were done many times during April 2002. A program written

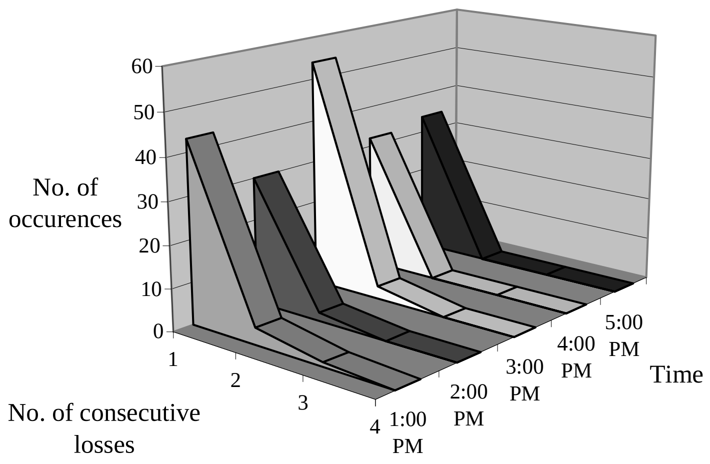

Figure 2.2: Loss rate in 61.129.72.34 (China).

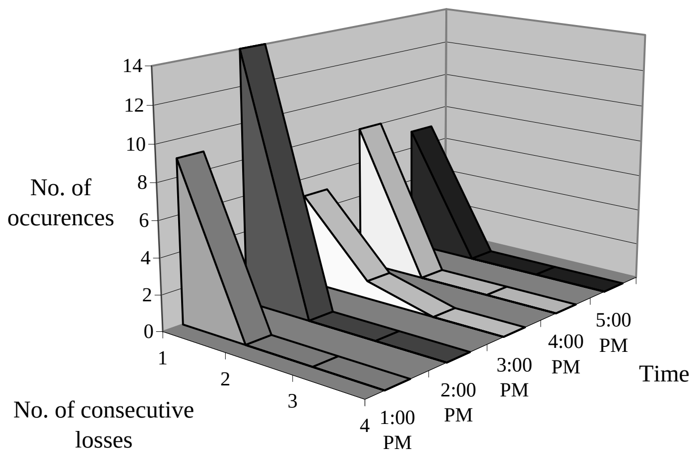

Figure 2.3: Loss rate in 195.146.45.162 (Iran).

in C analyzed the log files and produced the number of lost packets and their locations. In addition, it calculated the number of individual and consecutive lost packets. Some of the results are shown in Figures 2.2 to 2.4.

These results show that although most lost packets are separated and consecutive losses have a lower probability of occurrence, UDP packet losses are not well modelled by totally independent loss patterns. Other network researchers obtained similar results [6, 107, 11]. Bolot and his colleagues [7] had the same conclusion with their 320-byte UDP packets from INRIA in France to UCL in the United Kingdom. Although many researchers envisaged burst losses, they stated that most of the lost bursts consisted of isolated single losses [112, 105]. As sending TCP packets has a totally different strategy to sending UDP packets, measurements using TCP packets will generate completely different results in which greater correlation between lost packets will be observed [75].

Practical experience showed that the loss rate at some sites is quite high, reaching up

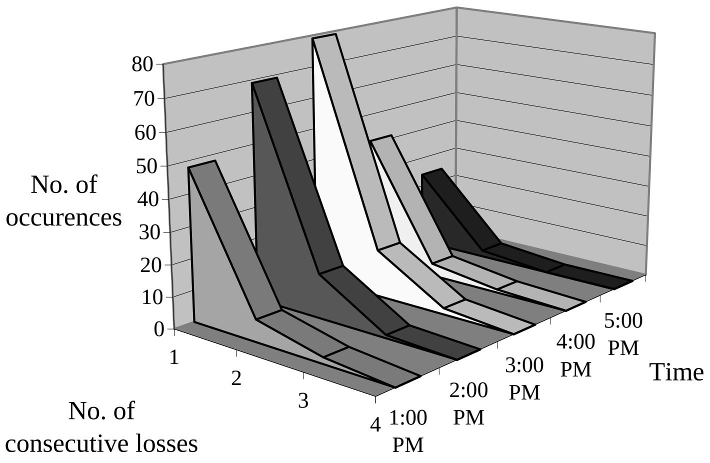

Figure 2.4: Loss rate in 202.186.124.32 (Malaysia).

to 10%. Losing 10% of voice packets has a severe negative impact on voice quality at the receiver specially when high compressed voice packing methods are used. The tests were repeated with 16 byte length packets and the same loss rate was measured. As is expected, the measurements described in Subsection 2.1.2 show that when the packets' length is less than MTU, the loss rate is only minimally related to the packets' size, because most routers are packet-limited rather than byte-limited [107].

2.2 Loss models

This Section describes some loss models and their specifications. Each loss model is based on several statistical assumptions, therefore, to select an accurate loss model, which matches well with the measured loss patterns, its assumptions and features must be considered. At first, an n^{th} order Markovian chain as a general model is chosen and explained. Then some of its special cases are considered and analyzed. Finally, taking into account the measured loss rates, a suitable model for simulation is chosen.

2.2.1 n-state Markov chain

This model has a memory of all n past events for both loss and non-loss packets [113]. In this model, the probability of receiving or losing the next packet depends on the past n packets, regardless of whether these packets were lost or received and it is described by

$$P(X_i = x_i | X_{i-1} = x_{i-1}, ..., X_{i-n} = x_{i-n}) \;=\; P(X_i | X_{i-1}, ..., X_{i-n}). \tag{2.1}$$

As this model needs to have all the last n states, it needs 2^n states to track the history of n packets and $2 * n$ conditional probability matrix to store all possible combinations. By

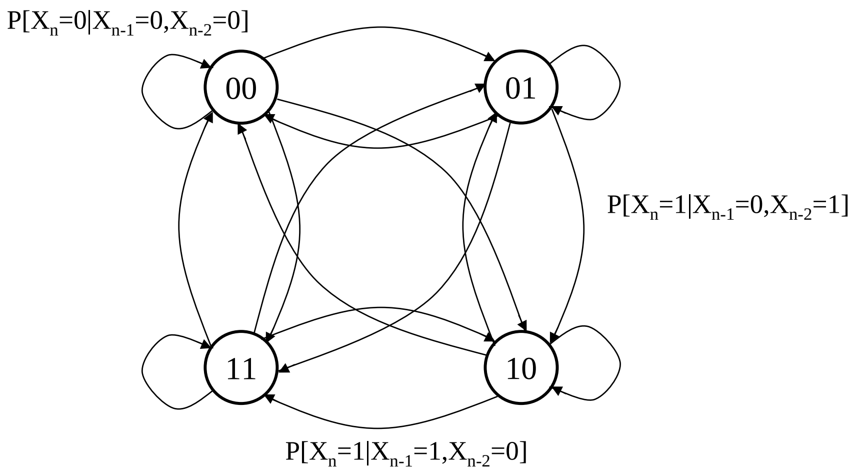

Figure 2.5: State diagram of a Markov chain with n=2.

definition, X_i is a Markov chain of order n if the conditional probability $P(X_i = x_i|X_{i-h} = x_{i-h})$ is independent of X_{i-h} for all $h > n$ [5]. Figure 2.5 shows the state diagram for $n = 2$.

This model is a general model and even for a small n, it is difficult to evaluate all of its parameters. The Markov chain can be simplified to have Bernoulli [46], Gilbert [8, 113] and Extended Gilbert models [92].

2.2.2 Bernoulli model

In this model the probability of losing or receiving a packet is totally independent of the situation of the previous packets. This simple model, which is known as Independent and Identical Distribution (IID) has a single parameter, P, which expresses the probability of losing a packet [46]. P can be calculated by

$$P = \frac{No.\ of\ lost\ packets}{Total\ No.\ of\ packets}. \tag{2.2}$$

This model that is a Markov chain model with the order of 0, has no memory about the lost packets. The observed loss patterns have some consecutive losses; therefore, this simple model cannot be classified as an accurate model for this purpose. Moreover, an average loss rate cannot explain the detailed loss patterns in the Internet. Different loss patterns that have different impacts on VOIP may have the same average loss rate.

2.2.3 Gilbert model

Observed patterns of lost packets in the Internet revealed that the probability of having k consecutive lost packets decreases approximately geometrically with k. This probability is

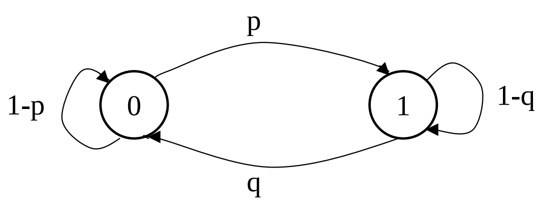

Figure 2.6: State diagram of Gilbert model.

stated by the Gilbert model [8, 113]. This model, as a two state Markovian chain is shown in Figure 2.6.

State $X = 0$ represents the state of a passed (no-loss) packet and state $X = 1$ represents the state of a lost packet. Receiving or losing a packet in this model is dependent on its previous packet, so it has one level memory. The Gilbert model as a special case of the Markovian chain with the order of one has two probabilities. p and q denote the corresponding state transition probabilities, and are defined by

$$p = P(X = 1 \mid X = 0) \ = \ P(packet\ n\ is\ dropped \mid packet\ n - 1\ is\ passed)$$

and

$$q = P(X = 0 \mid X = 1) \ = \ P(packet\ n\ is\ passed \mid packet\ n - 1\ is\ dropped).$$

p expresses the probability of losing a packet when the previous packet has been received and $1 - q$ expresses the probability of losing a packet when the previous packet has been lost. For steady state conditions, the transition and state probabilities can be expressed by

$$\begin{bmatrix} 1-p & q \\ p & 1-q \end{bmatrix} \begin{bmatrix} P0 \\ P1 \end{bmatrix} \ = \ \begin{bmatrix} P0 \\ P1 \end{bmatrix} \tag{2.3}$$

where

$$P_0 \ + \ P_1 \ = \ 1. \tag{2.4}$$

P_0 and P_1 as the probability of being in state 0 and 1 respectively, are gained from (2.3) and (2.4), as

$$P0 \ = \ \frac{q}{p+q} \tag{2.5}$$

and

$$P1 \ = \ \frac{p}{p+q}. \tag{2.6}$$

Based on this model, the probability of having k consecutive lost packets is given by

$$L_k = q(1-q)^{k-1} \tag{2.7}$$

and the probability of receiving k consecutive packets is given by

$$R_k = p(1-p)^{k-1}. \tag{2.8}$$

Upon defining

$$O_k \ = \ No.\ of\ k\ length\ losses \tag{2.9}$$

$$a \ = \ total\ No.\ of\ packets \tag{2.10}$$

and

$$d \ = \ \sum_{k=1}^{\infty} O_k k \ = \ total\ No.\ of\ lost\ packets \tag{2.11}$$

the p and q can be calculated by

$$p \ = \ \sum_{k=1}^{\infty} \frac{O_k}{a} \tag{2.12}$$

and

$$q \ = \ 1 \ - \ \frac{\sum_{k=1}^{\infty} O_k(k-1)}{d-1}. \tag{2.13}$$

Although Jiang [55] found that this model underestimates the probability of consecutive losses for his TCP based packets, from the simulation study described here it was found that the Gilbert model had enough accuracy to fit all its UDP based lost packets.

2.2.4 Extended Gilbert model

The extended Gilbert model [92] has one state $X = 0$ for all non-loss packets and m states (or infinite states) for lost packets. It is believed that when n is a sufficiently large number (depending on the voice-coding scheme), losing n or $n+1$ consecutive packets produces the same voice quality degradation in VOIP. Therefore, it is not necessary to study the infinite state model. In the extended Gilbert model, for any k, when $k < m$, $P(X = k)$ is the

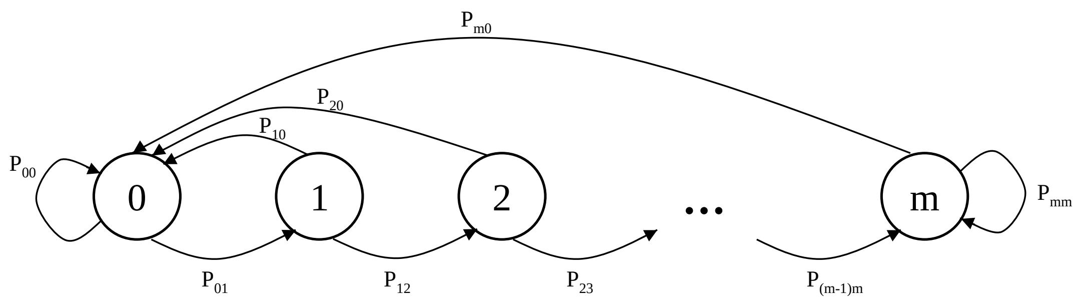

Figure 2.7: State diagram of extended Gilbert model.

probability of having exactly k losses and $P(X \geq k)$ is the probability of having at least k losses. Figure 2.7 shows this model.

In this model the probability of the next loss depends on the number of previously experienced consecutive losses. As in the case with the Gilbert model, the probability of going from a non-loss state to either a loss or a non-loss state is independent of the number of previous successfully received packets.

2.2.5 Poisson Model

The occurrence of lost packets can be modelled as a Poisson process with exponential inter arrival times [113]. This model appears to fit well when the packet loss inter arrival times can be shown to be exponential and lost packets can be shown as independent events. According to the collected information about lost packets, this model cannot be considered as an accurate model.

2.3 Model evaluation

Based on the measured loss pattern from the practical tests, a loss model is described, which can accurately explain the lost packets.

- Bernoulli model: Jiang and Schulzrinne in [54] compared the Bernoulli and the Gilbert model and they concluded that the Bernoulli model over-predicts single losses and under-predicts burst losses. As this model has only a simple average loss rate, they concluded that it cannot explain consecutive losses.

- Gilbert model: From (2.12) and (2.13), and measured loss patterns (Subsection 2.1.3), the p and q are calculated as

$$p = 3.3\% \qquad (2.14)$$

and

$$1 - q \ = \ 0.2\%. \tag{2.15}$$

It has been found that the Gilbert model underestimates the probability of having more than 6 consecutive losses [62]. In view of Subsection 2.2.4, the Gilbert model appears to be sufficiently accurate within the region of interest. Considering the fact that it is desired to find some voice-packing schemes to pack and transfer voice packets through the Internet, the Gilbert model expresses quite good loss estimation for simulation purposes. Moreover, the Gilbert model is used to analyze the role of active nodes in VOIP. For a very heavy lossy environment, the simulation uses the Gilbert model wherein $p = 50\%$ and $1 - q = 3.3\%$ which has the same p to $1 - q$ ratio with higher absolute value of p and $1 - q$.

- Extended Gilbert model: In spite of the complexity of this model, it was used for the simulation purposes. However, it was found that this model generates almost the same voice quality as the Gilbert model in a practical situation. Therefore, for the sake of simplicity, the Gilbert model is adopted in a subsequent analysis.

In order to identify the relation between the packet sending rate and packet loss rate, some work have been done by researchers [60]. Feng et al. observed significant packet loss even when the offered load was less than half of the available network bandwidth [27]. They stated that this behaviour is due to simultaneous bursts of traffic coming from client application and overflowing the buffer size in the bottleneck nodes. They showed that with only 100 μsec of spacing between packets, the throughput remains the same, but the packet loss drops to 35%. Therefore, in all of the voice-coding schemes, packet spacing with more than 10 ms interval between two packets is used.

2.4 Summary

Packet loss and its reasons are investigated in this Chapter. Some performed tests are described which revealed that the loss rate is mostly related to the number of packets rather than the packets' length. Therefore, the network loss rate is measured dynamically and consequently, redundant data based on that loss rate is inserted. As a result, the packets' length is dependent on the condition of the network and the voice-coding scheme. In so far as the redundant data is small and the packets' length is less than MTU, the loss rate would be minimally related to the packet size. In this Chapter, different models for analyzing packet loss are considered and the Gilbert model is chosen as a practical model for simulation purposes.

Chapter 3

Voice Over Internet Protocol (VOIP)

This Chapter begins with a brief review of speech properties and classification of speech segments. Different voice quality measurement techniques are discussed in the next section and the voice quality assessing method is explained briefly. Speech-coding schemes related to VOIP and some methods that attempt to mitigate packet losses are described within the next two sections. Methods for modelling QoS in the Internet and VOIP protocols are also described.

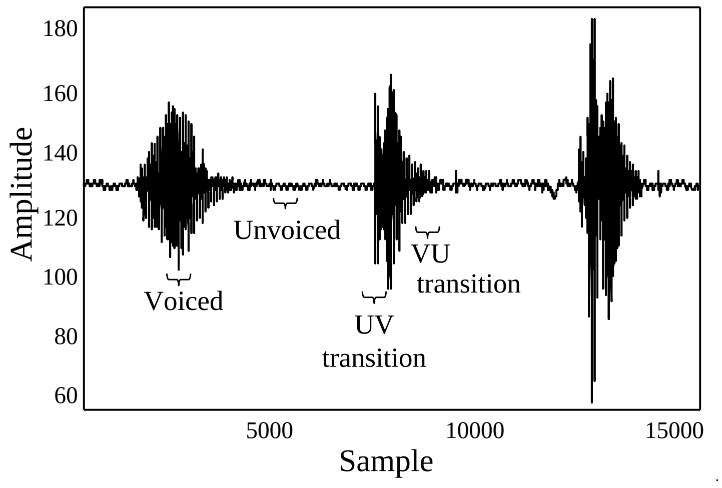

Figure 3.1: Speech waveform and its different parts.

3.1 Properties of speech

Vibrating vocal cords and the shape of the lips and the nose produce speech sounds. Speech sounds can be divided into four parts as follow.

- Voiced: Which is produced by closing and opening the vocal cords.
- Unvoiced: Which is produced by forcing air through a constriction of the vocal cords.
- Unvoiced to voiced transition.
- Voiced to unvoiced transition.

Figure 3.1 is an example of the author's voice saying: ”one two three”.

The voiced sounds have higher energy compared with the unvoiced ones and they are more important to the perceptual sound quality [13]. Voiced sounds have quasi-periodic characteristics in the time domain that is a useful property used in the DSPM packing method. In contrast with the voiced sound, the unvoiced sounds have a broad spectrum in the frequency domain and usually a random signal is used to model them. Packet losses have different effects on perceptual voice quality if they occur in different parts [100]. The losses occurring within unvoiced segments have little impact and loss occurring within the transition from unvoiced to voiced segments have the severest impact in total voice quality. Sun et al. in [100] showed the relationship between loss location and voice quality for different codecs. In addition, they showed the dependency between loss location and its convergence time. According to their results, loss of voiced segments has a different effect on the voice quality based on its location within the voice segment. The nearer the loss to the beginning of the voiced segment, the severer its effect on the total voice quality. The quality of regenerated voice is drastically influenced by loss within 150 ms after an unvoiced to voiced transition [48]. Some statistics calculated over a sample size of 64 kbyte are presented in Table 5.1.

3.2 Voice quality measurement

Speech quality deals with two concepts: Intelligibility (what the speaker has said) and Neutrality (how the speaker has said something). A very low-bit-rate vocoder can be considered as an Intelligible coder but it does not provide natural sound. It is tried to pack voice samples in such a way that in spite of some lost packets; the regenerated voice will be 100% intelligible. Therefore, it is desired to concentrate on improving the quality of sound by increasing its Neutrality as much as possible.

The quality of the regenerated voice is a subjective response by the listener. Mean Opinion Score (MOS) is a common benchmark to measure the quality of the reproduced sound at the receiver. MOS needs a wide range of listeners to hear and judge the quality of a voice sample on a scale of 1 (bad) to 5 (excellent). In order to judge the voice quality, the same strategy with scores between 0 (for very poor quality) and 10 (for the original quality) is used. To test the voice, an audience is asked to listen to the original voice and mark it 10. Then the regenerated voice samples were played and the audience had to mark them relative to the original voice. Therefore, all voice qualities are marked by *relative quality*, which means that they are judged in relation to the original voice. Test environment and all of the voice samples are detailed in Appendix C. The term ”acceptable voice quality” is used in this dissertation and its meaning depends on the current VOIP test situation. On the one hand, VOIP needs a real-time communication environment, therefore, the overall delay between sending and receiving a voice packet should be low. On the other hand, the Intelligibility and Neutrality of the received voice should be high enough to let the listener feel comfortable using VOIP instead of PSTN.

In contrast with the subjective tests, there is another method to measure the voice quality named the ”objective test”. In the objective tests, speech quality is evaluated by measuring the distortion of the regenerated voice compared to the reference voice. It is believed that an objective test is not an accurate method of judging the voice quality because when packets are lost, the distortion of the regenerated voice compared to the original voice does not offer a good measure of voice quality. Signal to Noise Ratio (SNR) [77], Diagnostic Rhyme Test (DRT), Diagnostic Acceptability Measure (DAM), VQmon [19] and the E Model [37] are other means of measuring the voice quality related to VOIP.

3.3 Voice Over Internet Protocol (VOIP)

To transfer speech over a digital system, it must be filtered and digitized to make voice samples. Waveform analysis shows that there is significant redundancy between contiguous samples. It is found that the correlation coefficient between adjacent samples is generally 0.85 or higher [3]. Therefore, it is usually possible to estimate a lost sample using its adjacent samples. In order to reduce the needed bandwidth to transfer voice samples three speech-coding methods are employed [91].

- Waveform coders: These types of coders try to directly encode speech samples. The Pulse Code Modulation (PCM) is the simplest waveform codec in which the amplitude of the signal is quantized to one of a discrete set of values.
- Vocoders: They try to model speech signals by a set of parameters and they decode the parameters at the receiver.
- Hybrid coders: They use both waveform and vocoder principles to encode voice samples. They try to minimize the distortion at the cost of a higher bit rate compared to the vocoders.

Vocoders and hybrid coders, which model the vocal track by a linear filter, usually operate at a lower bit rate compared with the waveform coders and they normally operate on fixed size of speech frames. Hybrid coders have better speech quality than vocoders using their higher bit rates. DSPM (Chapter 5), DADPCM (Chapter 6) and DADPCM5 (Chapter 6) use waveform coders and hybrid coders are used in ECELP (Chapter 9).

Apart from speech-coding schemes, VOIP needs methods to cope with packet loss problems when it uses a lossy network as a medium to transfer its packets. The following Section introduces these methods.

3.4 Methods to cope with packet loss problem

There are four well-known schemes to overcome the loss problem in VOIP applications:

3.4.1 Sender and receiver based scheme

In this method, a feedback type protocol called Automatic Repeat Request (ARQ) is applied between sender and receiver. When the receiver detects a lost packet, it requests the sender to retransmit that packet. ARQ is robust and reliable, but apparently, it adds significant delay in the packet transmission system. Interactive audio applications are very sensitive to

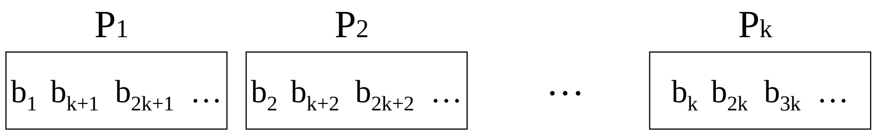
P1
b_1 b_{k+1} b_{2k+1} …
P2
b_2 b_{k+2} b_{2k+2} …
…
Pk
b_k b_{2k} b_{3k} …

Figure 3.2: Packet structure with interleaving factor k.

end-to-end delays and the delay needs to be less than 250 ms [12], however, International Telecommunication Union-Telecommunication Standardization Sector (ITU-T) G.114 recommends that one way delay in telephony should be less than 400 ms. ARQ is suitable for applications that are more loss-sensitive than delay-sensitive, such as file transmission over the Internet. This scheme is not typically used for compensating the lost packets by real-time applications.

3.4.2 Sender based scheme

In this scheme, the sender employs techniques to help the receiver regenerate the lost packets. The following describes some of the techniques.

- Interleaving: In this technique a speech segment is divided into blocks and the sender resequences the blocks before transmission, so adjacent blocks in the original voice segment are separated by a specific distance in the transmitted stream. The receiver buffers the incoming packets and reorders them into their original sequence. Suppose that m is the size of a voice segment in byte, n is the block size in byte and k in the interleaving factor.

  There are m/n blocks and they are sent toward the receiver in the sequence that is shown in Figure 3.2.

  In case of losing $P2$ in the network, the receiver will have

  $$b_1 \ \otimes \ b_3 \ldots b_k \, b_{k+1} \ \otimes \ b_{k+3} \ldots b_{2k} \, b_{2k+1} \ \otimes \ b_{2k+3} \ldots$$

  sample sequence (a lost sample is shown by $\otimes$). Thus, the receiver envisages n consecutive lost bytes every k blocks. Obviously, the lower n and the higher k, the better voice quality at the receiver. In contrast to the quality, the delay gets worse when n and k increase. Generally, the receiver has to wait to get all packets before decoding the voice. The total delay is proportional to

  $$Delay \ \simeq \ No.\, of \ blocks \ per \ packet \ * \ n \ * k.$$

In conclusion, the lower the n is, the better the quality and less the delay at the receiver are. Therefore, it is tried to use a small n. On the one hand the lower the k is, the less the delay that the receiver envisages but the worse voice quality the receiver gets. k is set to 2 for most voice-packing schemes but DADPCM5 uses $k = 5$ (DADPCM5 is a suitable voice-coding scheme for very lossy networks). Guo and his colleagues used inter-frame data interleaving and intra-frame data encryption in their scheme [45]. They proposed using interleaving to encrypt data and to protect data against packet loss problems. They used packet repetition as their lost packet recovery scheme.

- Forward Error Correction (FEC): This scheme is based on adding repair data to a stream, from which the contents of lost packets may be recovered [76]. The FEC schemes can be categorized in two groups: schemes that are independent of the content of the stream (data independent FEC) and schemes that add redundant data based on the nature of the data in the stream (data dependent FEC).
  - Data Independent Forward Error Correction: Parity coding [84] and Reed-Solomon coding [64, 82] are examples of schemes in which each codeword of m length main data generates a p length codeword of redundant data, generating an $m + p$ length packet (the redundant data might be sent in another packet).
  - Data Dependent Forward Error Correction: In this category, the redundant data is chosen based on the nature of the main stream. For example, in VOIP applications, the redundant data is chosen in such a way that the lost voice samples can be regenerated by the available redundant data in the next arrived packet. Inserting compressed voice codes of each packet into the next packet is an example of this error correction scheme [47, 77, 10]. These FEC schemes have the advantage of having low-latency compared with other sender-based techniques, because they need a single-packet delay for adding the redundant data to the main stream.
- Adaptive Packetization / Concealment (AP/C): Assuming that most lost packets are isolated and that the packet loss probability is equally distributed in relation to the packet size, Sanneck [90] changed the packet length based on the segment type and its pitch period. He tried to pack voiced segments in short packets and unvoiced segments in long packets. In his algorithm, when the receiver detects a lost packet, it copies and resamples the last arrived packet to fit the gap.

Rubenstein and Towsley in their work tried to combine both ARQ and FEC methods to cope with the loss problem in the network [86, 72]. They added parity to packets and tried to retransmit some packets that could not be corrected due to high loss rate in the network.

They added FEC packets with no concern for the specifications of the media, especially the voice. [93] tried differentiation between non-important and important voiced packets by marking the important ones and allowing them to be retransmitted if they were lost. Some works have been undertaken based on using more than one recovery scheme at a time. In another approach, Hoene et al. [48] proposed to add a booster in the data link layer of the sender and the receiver to improve the voice quality. They depend on ARQ and FEC to retransmit and recover the lost packets at the receiver end.

3.4.3 Receiver based recovery scheme

In these schemes, when some lost packets are detected, the receiver tries to recover them without interaction with the sender [58]. They do not need additional processing time or additional data overhead at the transmitter. These schemes, which rely on producing a replacement for a lost packet work for relatively isolated losses and small loss rates (less than 10%) and for short packets (less than 20 ms of voice).

The following describes some techniques, which have been introduced to regenerate the lost packets or conceal their effects on voice quality at the receiver.

- Splicing: Splicing together the audio on either side of the lost packet can cover the gap and conceal the loss effect. Gruber and Strawczynski [44] showed that this scheme generates poor voice quality when the loss rate is more than 3%.

- Silence substitution: This method simply inserts silence in place of the lost packet. It is very easy to implement but the sound has poor quality if the drop rate is higher than 2% [91, 59]. Performance of silence substitution degrades rapidly as packet length increases (more than 5 ms). Since this scheme does not need much processing time, it is suitable for an environment where computers do not have much processing power.

- Noise insertion: The sender inserts comfort noise information when the speaker is silent, and the receiver uses this information to generate noise in place of the lost packet. This method gives better performance than silence substitution [67, 108], but it is still poor if the drop rate is not very low.

- Packet repetition: This method simply repeats the last arrived packet in place of the lost packet. This scheme assumes that the speech characteristics do not change much from packet to packet. As the starting point of the repeated packet usually does not match the end point of the last packet, this method produces unacceptable sound quality when the drop rate is higher than 4% [59]. Also this scheme does not work well for long packets (more than 40 ms).

- Pattern matching: This method replaces the lost packet with the most identical n arrived packets previously received. This may cause an echo effect when the loss rate is higher than 6%. The two-sided pattern matching method works on matching both sides of a lost packet but it needs a shorter n to search for the matched packet [59].
- Pitch waveform replication: This scheme generates the main frequency of the last arrived packet and inserts it in place of the lost packet. A double phase matching technique helps to match both edges of the substitute packet in order to reduce the clicking distortion. The packet loss rate tolerable in this method is less than 10% and this scheme has increased CPU overhead [109].
- Time Scale Modification: Stenger et al. [99, 94] present a scheme that finds overlapping vectors of pitch cycles in both sides of a lost packet and stretches them to cover the gap. Although this technique regenerates better voice quality compared with pitch waveform replication, it increases CPU overhead.

Receiver or sender-receiver based schemes which are being used in the current network have the following problems:

- in the case of having a high packet loss rate, these schemes take rather a long time to improve the quality of sound, thus compromising the real-time environment,
- with a very lossy network, these schemes cannot cope with the lost packet effect and sound quality is poor. This is particularly so when consecutive packets are lost in the network.

3.4.4 Network and receiver based recovery scheme

Using traditional networks, the only way to cope with the lost packet problem is to recover them at the receiver end, or retransmit them. Some coding schemes allow regenerating and recovering the lost packets within the network as well as at the end point. Executing recovery programs in the network reduces the total amount of work that needs to be done by end users and it reduces the overall delay in communication [110]. This strategy is adopted herein to regenerate the lost voice packets within the network to improve the voice quality at the receiver.

3.4.5 Packing methods used in the simulations

Many methods are proposed to pack voice packets for VOIP. To compare their regenerated voice quality with the other available methods' voice quality, three schemes are chosen.

- Two methods from the receiver based recovery schemes.
  - Silence substitution: It is very easy to implement and it does not need much processing time [91, 59].
  - Packet repetition: As it simply repeats the last arrived packet in place of the lost packet, it does not need much processing power in the receiver side [59].
- A method from the sender based recovery schemes.
  - Data dependant FEC: It inserts compressed voice codes of each packet in the next packet [47, 77, 10]. As an advanced method, a scheme which inserts redundant low bit rate (6.4 kbps) LPC codes piggy backed on the high quality 64 kbps main PCM (or 32 kbps ADPCM) codes is used. Although this scheme produces good voice quality (compared with the other two methods), it increases CPU overhead (refer to Subsection 10.6.3).

The voice quality of all proposed schemes are compared with these methods' voice quality and their specifications are discussed in the conclusion subsection of each chapter.

3.5 Quality of Service (QoS)

By definition, QoS is the capability of a network to provide better traffic service over various underlying conditions. Service availability, throughput, packet loss rate, delay and jitter are all important measurements of QoS in a network. The network tries to maximize the first two items and minimize the rest to improve the quality of its service.

3.5.1 QoS model

Three major models for implementing QoS in packet networks are available.

- Best-Effort service: In this model, an application is allowed to send any amount of data and there is no QoS guarantee for it except that a best effort to deliver the packets will be made. In the IP network, which is an example of this model, the network cannot limit the maximum packet loss rate or delay of an application. item Integrated Service (IntServ): This model is defined by Internet Engineering Task Force (IETF) group [49]. The IntServ model adopts a per-flow [1] approach, which means that each flow is handled separately at each node. It assumes that network resources (mainly bandwidth) must be managed in order to meet application requirements (resource reservation). Each flow

[1]Flow: Packets with the same source and destination address and the same port number.

can request a specific quality of service and IntServ reserves the requested QoS for the flow [41]. In this model, the application must be aware of the nature of traffic that it would put on the network and the network nodes must reserve the required resources for that application. Two important issues related to IntServ are:

  – IntServ as a packet handling mechanism needs to be implemented at every network node, and all network nodes between sender and receiver must support this protocol in order to benefit the user,
  – when a route changes, there must be some mechanism to set up a reservation along the new route.

- Differentiated Service (DiffServ): This model, which is defined by IETF, provides QoS on a per-aggregate basis. In this scheme, some predefined classes of QoS exist and a flow is assigned to one of those classes. It is based on a resource allocation function, which is logically located at the edge of the network. It is easier to implement than the IntServ scheme and it offers quantified end-to-end QoS guarantees. It uses the Resource Reservation Protocol (RSVP) when requesting QoS [18].

The last two schemes are concerned with improving QoS on the network by changing its edge nodes or the whole network. They focus on delay, trying to reduce it in order to improve the quality of real-time applications. In the case of congestion, they introduce methods to drop the unimportant packets. For example, they use the enhanced Random Early Detection (RED) technique, providing a better handling mechanism for drop problems in high traffic situations [18]. To sum up, these schemes emphasize the reduction of overall delay, but they do not cope with the drop problem and they cannot compensate for lost packet effects. Alternatively, it is desired to reduce the packet loss problem by regenerating lost packets in the network or at the receiver. This does not require a total change in the whole network and it can be implemented in a few nodes in the current network. As the current network functions based on the best-effort principle, the first model is adopted herein.

3.6 Protocol

In order to simulate a packet transmission environment, a number of standard protocols are adopted. Each voice packet is generated by applying a voice-packing scheme on a voice segment and packing the gained data over the Real-time Transport Protocol / User Datagram Protocol / Internet Protocol (RTP/UDP/IP) header. To get detailed feedback from the transmission situation (mainly, loss rate), some alterations on protocols are necessary. RTP, UDP, IP and their alterations are explained in the following subsections.

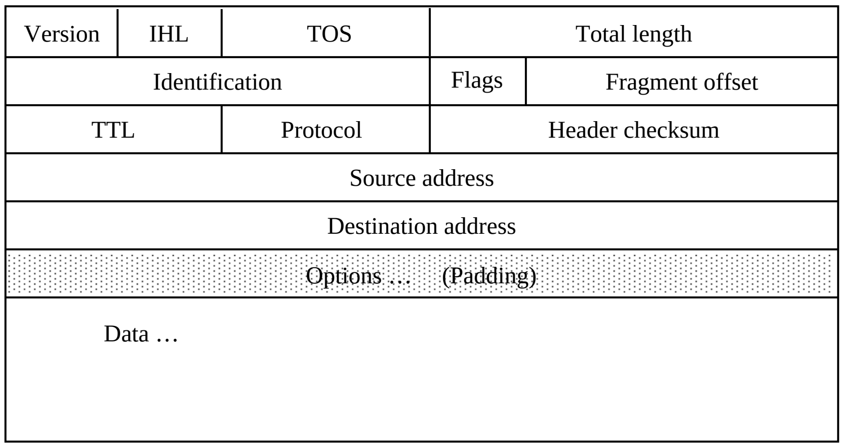

Figure 3.3: IP header fields.

3.6.1 Internet Protocol (IP)

The Internet Protocol provides a connectionless datagram transport service across the network, which is considered unreliable. That is because the network cannot guarantee the packet delivery due to communication errors or network congestion. In other words, IP does not provide a mechanism for flow control. The size of an IP header varies from 20 to 60 bytes according to the Internet Header Length (IHL) field. IP datagrams are able to carry messages shorter than 65535 bytes length. Since the IP option part of the IP header (Subsection 4.8) is changed, the header structure is introduced briefly. The IP header fields are shown in Figure 3.3.

The fields have the following meaning.

- Version: Specifies the IP version of the packet.
- Internet Header Length (IHL): Length of datagram header in 32 bit words.
- Type Of Service (TOS): Allows different class of service and priority for packets.
- Total length: Indicates data plus header length.
- Identification: Used for packet fragmentation.
- Flags: Used for packet fragmentation and reassembly.
- Fragment offset: Used for packet fragmentation.
- Time To Live (TTL): Maximum number of hops that one packet is allowed to take.
- Protocol: Indicates the higher-level protocol for this packet.
- Header checksum: Checksum of the header.

- Source address: IP address of the packet sender.
- Destination address: IP address of the packet receiver.
- Options: A set of options that may be applied to the packet. The IP option part of the IP header is used to mark a packet as an active packet. It is explained in Chapter 4.

3.6.2 User Datagram Protocol (UDP)

This connectionless protocol that runs on top of the IP networks provides an unreliable packet delivery environment. UDP does not guarantee reliable communication and it does not guarantee preserving the packet sequence at the receiver side. So the UDP packets can arrive out of order or not at all. The receiver does not acknowledge the packets; therefore, the sender does not know whether the transmission was successful. The upper-level application layer is responsible for detecting transmission errors and having a strategy to provide a reliable delivery environment.

3.6.3 Real-time Transport Protocol (RTP)

In addition to UDP/IP that provides a connectionless and unreliable environment, a transport protocol is needed to deliver voice packets from the sender to the receiver. The transport protocol should have the following facilities.

- Sequencing: In the case of out-of-order packets, the protocol should be able to identify the packets and help the upper layer to reorder them.
- Lost packet detection: The protocol should help application to detect the lost packets.
- Intra-media synchronization: Considering the fact that no data is usually sent during the silent period in speech, the duration of the silence must be detected and reconstructed properly at the receiver.
- Feedback on network quality: As inserting redundant data based on the packet loss pattern in the network is considered, the protocol should report the number of lost packets and their location (loss pattern) to the sender.

To have the above facilities, Real-time Transport Protocol (RTP) on top of UDP/IP is used, which provides end-to-end delivery service for data with real time characteristics [31]. RTP has a 12-byte length header and it has a payload type identifier, which allow the user to specify the voice-packing scheme (such as PCM, ADPCM, LPC or user defined methods). RTP header has a sequence number that allows the receiver to reconstruct the sender's packet

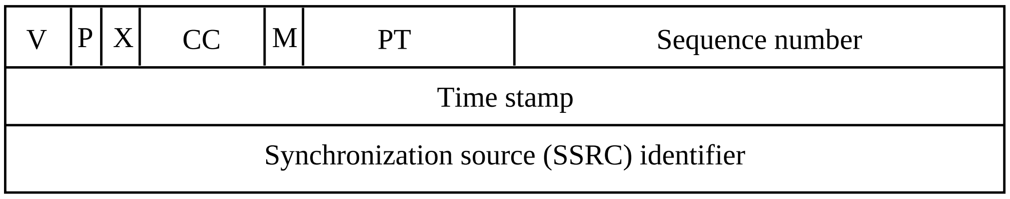

Figure 3.4: RTP header fields.

sequence and it helps detect the lost packets. To satisfy the intra-media synchronization condition, it has a time stamp with witch the receiver can reconstruct the silent part of the received sequence. Although RTP provides a multicast communication environment, a unicast, single-sender, single-receiver session which is the case for the IP telephony is used. Figure 3.4 shows the data structure of a RTP header. Note that the optional part of the header is not shown in the Figure.

The fields have the following meaning.

- Version (V): Identifies the version of RTP.
- Padding (P): Enables the user to insert data at the end of RTP.
- Extension (X): Enables the user to have one header extension.
- CSRC Count (CC): Contains the number of contributing source identifiers that follow the fixed header.
- Marker (M): A profile defined marker.
- Payload Type (PT): Identifies the format of the RTP payload.
- Sequence number: Increments by one for each RTP data packet sent.
- Time stamp: Reflects the sampling instant of the first octet in the RTP data packet.
- SSRC: A randomly chosen number which identifies the synchronization source.

Table 3.5 shows the header fields and their assigned values.

Although RTP provides some packet delivery facilities, it does not guarantee timely delivery or other quality of service mechanisms. In order to have minimum overhead, a low amount of redundant data is needed to improve the voice quality when packets are lost. Therefore, it is necessary to have a feedback report on network quality at the sender.

Parameter	Value	Result
Version	2	
Padding	0	No padding at the end of RTP
Extension	0	No extension
CC	0	No contributing source is needed
Marker	0	No packet marking is needed
Payload type	81 … 89	Each voice packing scheme uses a specific number (unassigned numbers in RFC 1890)

Figure 3.5: RTP assigned value.

Real-time Transport Control Protocol

Besides the RTP, which carries the data, RTCP is used for monitoring the quality of service. The RTCP as a feedback on the packet loss rate helps the voice-coding schemes to adjust their parameters and retain an adaptive amount of redundant data. The Bolot algorithm [9] tries to maintain the loss rate after reconstruction at the receiver. Bolot suggests having a high and low threshold and adjusting the amount of redundant data based on loss rate reported by RTCP. As he calculates the percentage of lost packets after reconstruction, the calculated rate may not be representative under all network conditions and it may lead to an incorrect amount of redundant data. The USF algorithm [74] uses two additional values that the receiver sends to the sender. The number of lost packets before and after reconstruction helps the USF algorithm adjust its parameters. A small alteration in the option part of RTCP provides the sender with detailed information about the packet loss.

RTCP consists of a fixed header part similar to that of RTP, followed by many structured elements that vary depending upon the RTCP packet type. RTCP is periodically transmitted to all session participants. Recommended semi-random RTCP intervals depend on some parameters such as the number of users in a session and transmission bandwidth. It is recommended that 5% of the session bandwidth should be allocated to the RTCP packets. Alternatively, RTCP can be used by reduced interval values, which is 360 divided by the session bandwidth in kbps [31]. RTCP with an interval varying between 2 and 4 seconds is used to adjust voice-coding parameters without introducing a heavy load to the Internet. In this way, RTCP uses very little bandwidth.

As the amount of redundant data is adjusted by feedback on the network quality, RTCP has a profound effect on voice quality herein. Figure 3.6 shows the data structure of a RTCP

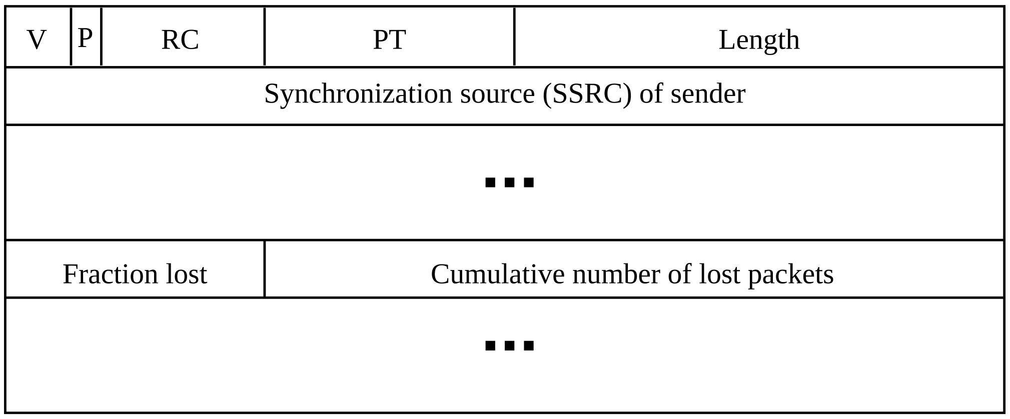
V
P
RC
PT
Length
Synchronization source (SSRC) of sender
...
Fraction lost
Cumulative number of lost packets
...

Figure 3.6: RTCP header fields.

packet. The optional parts are not shown in the Figure.

The fields have the following meaning.

- Version (V): Identifies the version of RTCP (same as RTP version).
- Padding (P): If it is set, the RTCP packet contains some additional padding data at the end of RTCP.
- Reception report Counter (RC): Number of reception report blocks in this packet.
- Packet Type (PT): Identifies the type of RTCP packet.
- Length: Length of RTCP packet in 32 bit words.
- SSRC: SSRC from the sender.
- Fraction lost: The number of lost RTP data packets since the previous RTCP packet was sent, divided by the number of expected packets. If the loss is negative due to duplicates, the fraction lost is set to zero.
- Cumulative number of packets lost: The total number of RTP data packets that have been lost since the beginning of reception. The late arrival packets are not considered lost and the cumulative lost may be negative if there are duplicates.

Table 3.7 shows the RTCP header fields with their assigned values.

RTCP does not provide any detailed information about the packet loss pattern, therefore, it has been decided to add some data at the end of the RTCP packet.

Parameter	Value	Result
Version	2	Same as RTP
Padding	1	Four bytes are added at the end of RTCP
RC	0	No reception blocks
PT	200	Constant number

Figure 3.7: RTCP assigned value.

Patch on RTCP

In order to have detailed statistical information about the loss pattern, the padding (P) bit in the RTCP header is set and four bytes at the end of the RTCP packets are added. One byte reports the number of individual lost packets, and two other bytes report the number of two and three consecutive lost packets. The fourth byte must be set to 1 in order to indicate the added data size in 32 bits. Using these three bytes and having the total number of lost packets (fraction lost), it is possible to adjust the transmitted data rate (change the amount of redundant data and voice-coding strategy) to cope with different loss patterns. Figure 3.8 shows the used RTCP packet. Voice-coding schemes dynamically adjust their coding parameters based on loss pattern and rate reported by RTCP. [17] mentions a method to classify the observed loss pattern to two computed loss rates; the current loss rate, which indicates a short-term condition and smoothed loss rate, which reflects a long-term trend. Using almost the same strategy, the written program for the simulation computes the loss rate at each moment, and the sender routine adjusts the amount of redundant data for each packet.

3.6.4 RTP/UDP/IP

To send voice packets, IP for routing packets and device-to-device communication, UDP to provide an end-to-end datagram service and RTP to provide end-to-end delivery services are used. To pack a voice packet, voice payload is time-stamped using RTP that introduces a 12-byte header. The resulting segment is carried by a UDP datagram which further adds an 8-byte header. Finally, when encapsulated into an IP datagram, a 20-byte IP header is added. Figure 3.9 shows the structure of a VOIP packet.

Although Marzegalli et al. [63] proposed a method to compress the header part of VOIP packets , the whole header for the simulations is used.

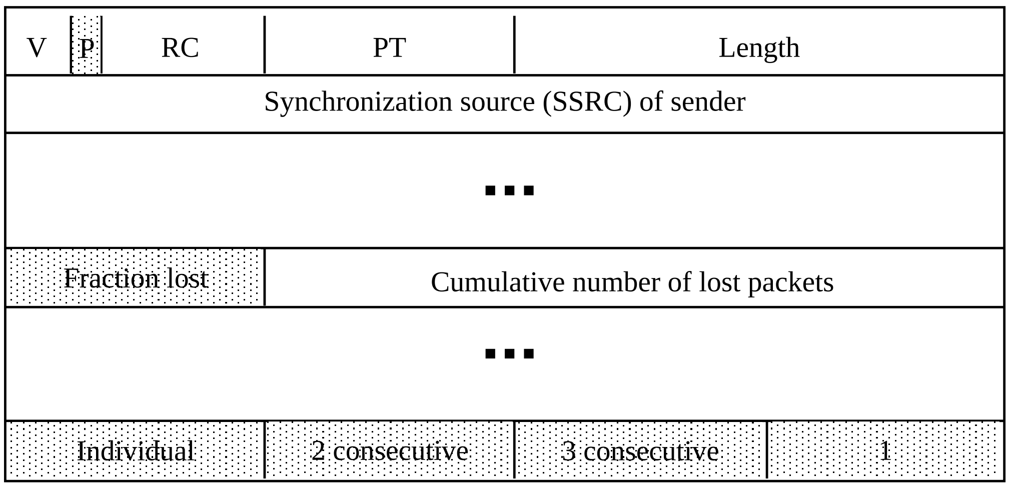
V
P
RC
PT
Length
Synchronization source (SSRC) of sender
...
Fraction lost
Cumulative number of lost packets
...
Individual
2 consecutive
3 consecutive
1

Figure 3.8: Proposed RTCP patch.

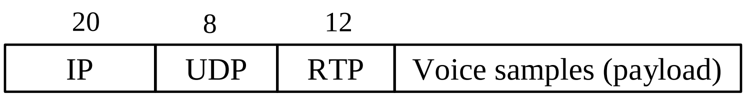

Figure 3.9: VOIP packet structure.

3.6.5 Other protocols for VOIP

ITU-T recommends using the H.323 standard to provide technical service for voice communication over networks, assuming that no quality of service is being provided by the networks [68]. Session Initiation Protocol (SIP) is an application level protocol for voice or multimedia session establishment on the Internet.

Both H.323 and SIP that are suitable to be used over TCP/IP networks are used to establish, talk and disconnect a call.

3.7 Summary

Loss of different parts of speech has a different impact on perceptual voice quality and this fact is taken into consideration in many of the voice-packing schemes. Among applicable methods to cope with packet loss problems, the network and receiver based recovery scheme is used. The proposed RTP/UDP/IP based packing methods work, based on the lost packet recovery within the network and at the receiver. The schemes adjust their coding parameters based on the loss pattern reported by the RTCP packets. In order to judge the voice quality, an audience is asked to compare the quality of different voice samples.

Chapter 4

Active node

This Chapter presents the passive and active network specifications, followed by active node structures and their characteristics. The application of the active nodes, their hardware and software, are explained in detail. Three mathematical analyses concerning the role of the active node in improving the quality of received sound in VOIP, are presented. At the end of this Chapter, a simulation environment and the procedure for implementing an active node, is stated.

4.1 Reasons to use active networks

Today's Internet, as a packet-switched network, offers a very simple QoS scheme. It functions on the *best-effort* principle but does not guarantee a reliable transmission environment, nor does it guarantee a limit to the maximum packet drop rate and delay, which is needed for real time applications. The traditional networks are only responsible for routing packets and they cannot help users cope with some network related problems such as packet loss. In the case of VOIP, some voice packets may be discarded in the network and consequently, the received voice will have poor quality. Some high level protocols including TCP, which work based on retransmission, help users overcome the loss problem. In the case of a real time application, such protocols cannot be used because they increase the overall delay. Instead of wasting time retransmitting lost packets in the traditional network, some of them can be regenerated in active nodes within the network. In active networks, when an active node receives an active packet, the node has to execute the packet program code, or a pre-stored routine in the node's structure. Therefore, in some applications, the effect of lost packets can be compensated within the network in some nodes [13].

4.2 Active and passive networks

Network technology provides a way to transmit data packets from a data source to a destination using network nodes. Network nodes obey special rules, called protocols, to treat and route packets. In traditional network nodes, packet processing is limited to routing, scheduling, queueing and packet forwarding which is called header processing. In other words, packet processing is limited up to the network layer related to the OSI model. In these networks, each node receives and stores a packet, analyzes the packet's destination address and transmits the packet to another node. This process happens repeatedly until the packet reaches its destination. Such nodes are only responsible for communication. These sorts of networks can be modelled as *store-forward* networks, because nodes store packets and forward them on to their destination [22]. Figure 4.1 shows a typical structure of a traditional network node.

In the traditional networks, packets are passive containers carrying data, and their payload is just a sequence of bits. The data of a packet is only meaningful to the application layer at the end nodes. These traditional networks suffer from some limitations and boundaries. Such restrictions are due to the networks' structures and protocols. Although some limitation mechanisms have beneficial influences on networks, such as immunization against viruses, other restrictions highlight the weak points in the total network's performance. For example, changing the nodes protocol or developing a new routing algorithm is difficult and requires

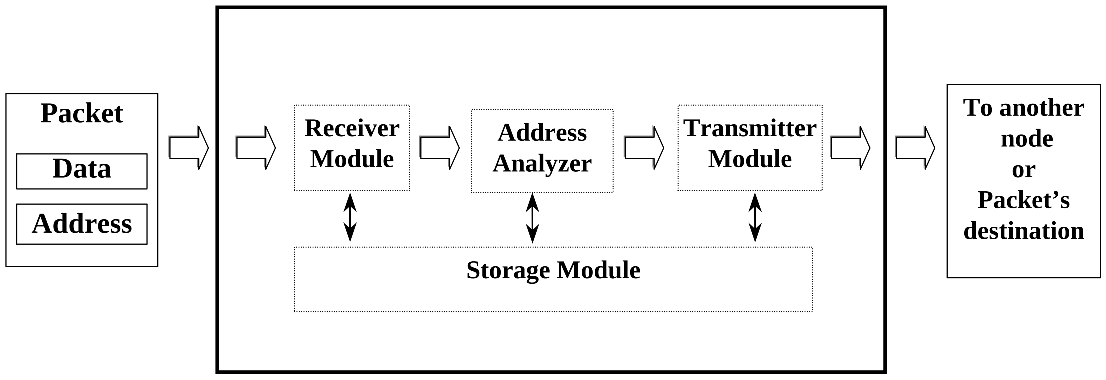

Figure 4.1: Node structure in traditional network.

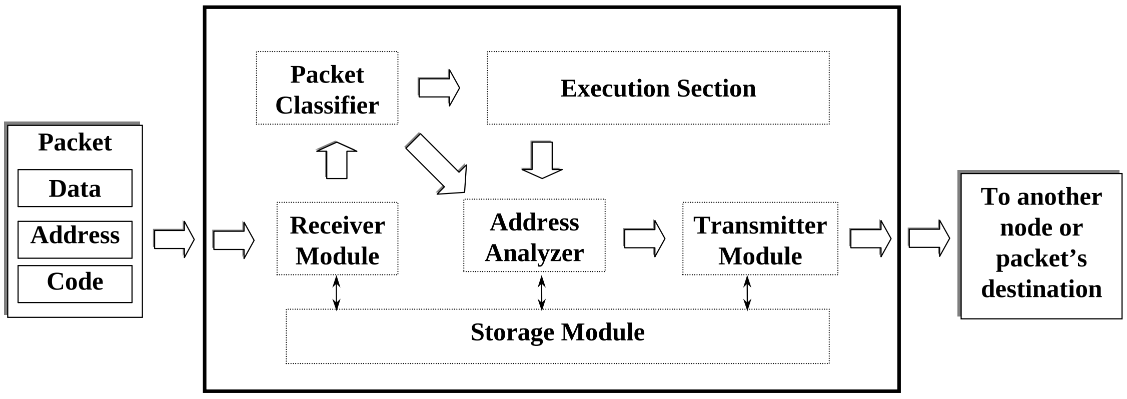

Figure 4.2: Node structure in active network.

plenty of time. Furthermore, nodes are not able to execute a program on a packet, so the packet's data is not changeable when the packet passes through the network. Moreover, traditional network nodes cannot help users cope with some network related problems.

The active networks concept was introduced to overcome these limitations. The main idea of active networks is enabling network nodes to perform computation on the data of specific packets. In these types of networks, each packet can be treated as a program-data packet, called an *active packet* [30]. Each active packet may carry program codes that can be executed along the network. In addition, a packet may have data, which can be processed within a node when the packet is passing through the network nodes. Each active node has an execution environment besides its communication environment. These networks can be modelled as *store-compute-forward* networks, because nodes receive and store packets, execute their codes and forward them on to their destination [22]. Although the main reason for active networks is communication not computation, they are designed to execute some programs on active packets [14].

Figure 4.2 shows a basic structure of an active node. The packet classifier routes active packets to the execution section and it leaves the other packets to pass on the traditional path.

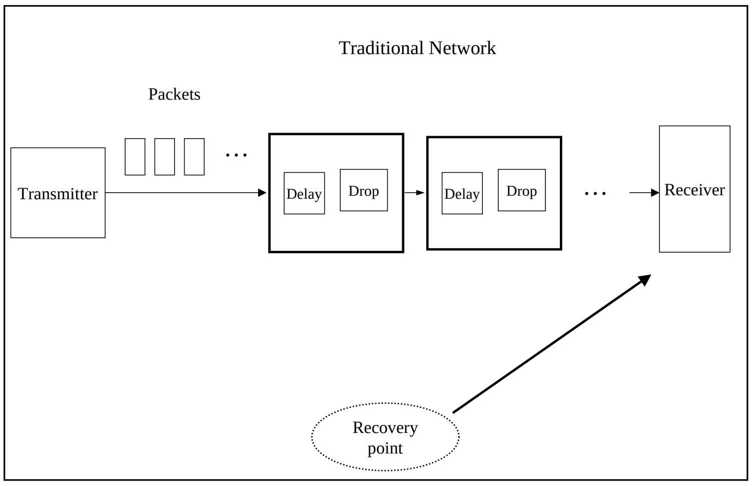

Figure 4.3: Traditional network packet recovery point.

From the OSI model viewpoint, an active node can be considered as an intermediate user which has access up to the application layer of the model. Data processing within the network reduces the total amount of work that needs to be done by an application layer at the end node [110].

Active packet program codes can be divided into three categories. In the first category, the program codes can change a node's behaviour such as the node's operating system or its routing algorithm. In this case, nodes can be programmable by downloading new software using active packets [32]. In the second category, the program codes can contain some application programs that must be executed on the data of the packet. In this case, a packet's codes have influence on the current packet. In the third category, the packet program codes can download a program into node structure that can be executed on special packets, which may arrive in the future.

4.3 Active node and lost packets

Active nodes have the ability of executing a program on a packet in the network. In such cases, nodes can help specific applications to cope with network related problems such as packet loss. Each network node can be modelled as a node that applies delay between receiving and transmitting a packet or a node that can drop some packets. Although not all lost packets are dropped in the nodes and some of them are dropped in the transmission lines, it is possible to summarize all drops in network nodes. Figure 4.3 shows a simple node model in the traditional network and it shows that the receiver is the only section that is responsible for packet recovery in a session.

Plainly, all effects of the dropped packets must be compensated at the end point. Taking

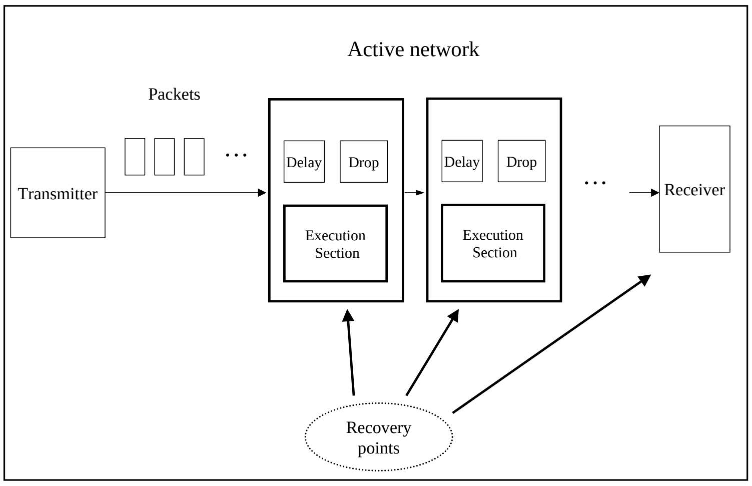

Figure 4.4: Active network packet recovery points.

advantage of active networks, users are able to execute a lost packet recovery routine in the active nodes to have better voice quality at the receiver. Figure 4.4 shows a simplified model of the active nodes in an active network.

It can be seen that the lost packet recovery points are distributed in the active networks. The *execution section* of each active node can execute a lost packet recovery routine; therefore, the receiver will have better voice quality. The recovery routine uses packets adjacent to a dropped packet to regenerate the lost packet and then it sends the recovered packet toward the receiver. The main idea of using active nodes to regenerate the lost packets is shown in Figure 4.5 (which is applicable to the DADPCM5 method).

The Figure represents the voice source which generates five packets and sends them toward the receiver. Packet numbers two and four are dropped in the first lossy network environment and the active node recovers them using their adjacent packets. Having the recovered packets, the receiver can recover the third packet, which is dropped in the second lossy network environment. As a result, the regenerated voice will have good quality at the destination. Detecting a lost packet, the active node delays the received packet until the lost packet has been reconstructed to avoid reordering.

4.4 Active path

The Internet as a packet-switch based environment does not setup a whole path from the voice source to its destination to transfer voice packets. Packets may pass through different paths to get to the destination, so it is difficult to keep track of all packets during a conversation. In order to benefit from active nodes to regenerate the lost voice packets, it is necessary to have a unique path between the sender and the receiver. That *active path* should have

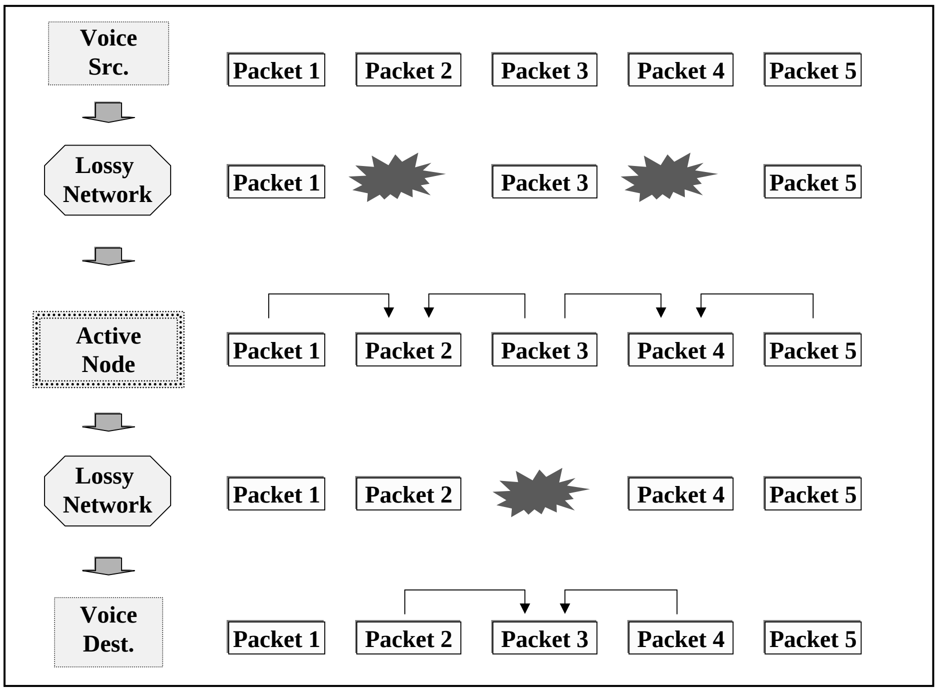

Figure 4.5: Lost packet recovery by an active node.

active nodes and each node should have a specific packet recovery program. If a fixed path from the voice source to its destination is not possible, it is necessary to ensure a path that enables packets to pass through some specific active nodes in the network. Using Virtual Private Network (VPN) concept, one can reserve a whole path from the voice sender to its receiver, in order to have a virtual circuit switching system [20].

4.4.1 Virtual Private Network (VPN)

A VPN which has an active nodes in its pathway is used for the simulations. VPN establishes a private data network over a non-private network to transfer its packets. It creates a tunnel between two endpoints to send packets via the Internet. The tunnelling is a method of using a network infrastructure to send a packet from one network over another network. The tunnelling method encapsulates a frame in an additional header, which provides routing information in an intermediate network. The new packet is sent to the tunnel endpoint over the inter network. When the packet arrives at the tunnel endpoint, it is un-encapsulated and sent to its destination.

An active node is chosen as a tunnel endpoint, which de-encapsulates the packets on arrival, processes their data and sends them to their destination. Despite the fact that the data encryption is one of the most important concepts behind VPN, the voice packets are not encrypted for the simulations. This type of tunnelling that is called IP over IP technique,

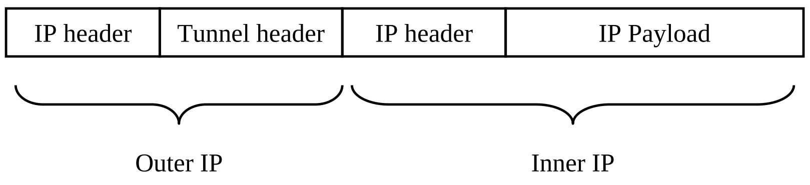

Figure 4.6: IP over IP packet structure.

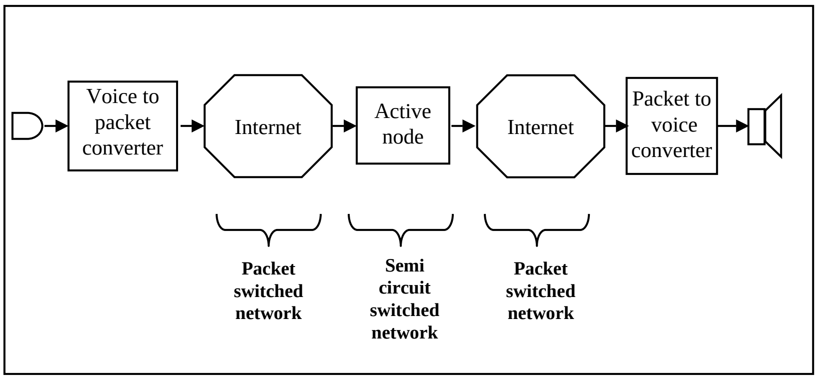
Voice to packet converter
Internet
Active node
Internet
Packet to voice converter
Packet switched network
Semi circuit switched network
Packet switched network

Figure 4.7: Virtual Active Path structure.

encapsulates IP packets in an additional IP header before sending them across an IP inter network [97]. In IP over IP technique, an outer header is added before the main IP header. The outer header identifies the source and destination of the tunnel and the main header identifies the original sender and recipient of the datagram [98].

Figure 4.6 shows the packet structure for IP over IP technique.

4.4.2 Virtual Active Path (VAP)

Using the tunnelling and the VPN concept, a Virtual Active Path (VAP) between voice source and its destination is used. The topology of VAP is shown in Figure 4.7. The first packet-switched network transfers voice packets to the semi circuit-switched network, which has an active node in its structure.

The active node has the lost packet regeneration algorithm in its structure. It recovers the lost packets and sends them to the voice destination via another packet-switched network.

4.5 Active Packet

Packets in the traditional networks, called passive packets, need only to carry their data, source and destination addresses. Thus, there is no routing information included in the passive packets. In contrast, in the active networks, the active packets have a more complicated

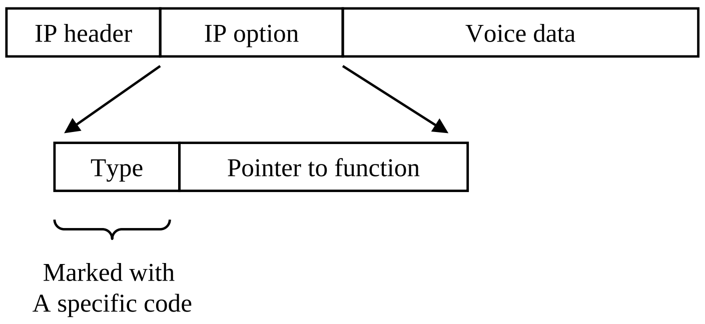

Figure 4.8: IP option part.

structure. Like the passive ones, they carry their data, source and destination addresses. Moreover, they have to carry path, node and protocol dependant information, which adds overhead to them. Active packets can be classified in three categories.

- Packets with program code: These packets carry codes to change the kernel software of a node. Using these packets, active nodes can change their software, such as a routing algorithm or Operating System, by downloading new software, packet by packet.
- Packets with program code and data: These packets have a program code that must be executed on the data of that packet. They carry the code that influences the data received by the current packet. Therefore, the code changes the behaviour of only the node seen by this packet. Active nodes dedicate a temporary memory to hold and execute the program. The nodes retake that memory after the program has finished its task. This method is useful when the program is short.
- Packets with data and a program pointer: These packets have a pointer to the function of a pre-stored program on the node structure, which must be executed on the data of the packet.

The third category is used for the simulations and its packet structure is explained in Subsection 4.6.6. To help the active node distinguish between active and passive packets, a modification in the IP option part of the IP header is necessary [111]. Figure 4.8 demonstrates the IP header structure for this purpose.

Two parts in the IP option section of the IP header are assigned. The Type part is marked with a specific code, which helps nodes (active or passive) distinguish between active and passive packets. The passive nodes ignore the option part and forward the packet to the next node. Active nodes can detect that code and treat the received packet as an active packet. The second part is a program identifier that calls up the recovery program in the active node structure.

4.6 Active node problems

Recently, researchers have been showing significant interest in the active node structure [38, 35, 36]. Most works try to cope with the fundamental problems related to general-purpose active nodes that must execute unknown packet programs. They address the resource allocation, security and other problems related to executing unpredictable programs carried by active packets. A specific application of active nodes is adopted herein which is related to VOIP and it uses a well known, fully tested and pre-stored program in the node structure. Therefore, this Chapter provides detailed information about the above-mentioned application.

4.6.1 Active node in VOIP

From an active packet point of view, nodes can be categorized into two groups.

- Nodes without an application program: In this kind of node, it is the packet's responsibility to carry its own program code as well as its own data.

- Nodes with an application program: Many pre-installed programs exist on the node structure and each active packet carries an identifier of one of those programs. The identifier is a short length *pointer to function*, which calls up the voice lost-packet recovery routine that must be executed on the data of the packet. The program runs in the shell environment of the node structure and the node deletes the temporary environment of that program when the task is finished.

As the voice-packet-regeneration code is quite a long program, the last method is used for the simulations.

4.6.2 Node security

Having programmability in the packet level raises security problems in the node structure. Active nodes must be made safe and secure from a packet's malicious program, and active packets must be immunized against other packets' unauthorized access. In addition, an active packet may ask to create another active packet. This process may happen repeatedly, causing an increasing number of packets across the whole network [95]. To call up a tested routine which is a pre-stored routine in the node's structure is recommended. Therefore, security is not a serious issue for the simulations.

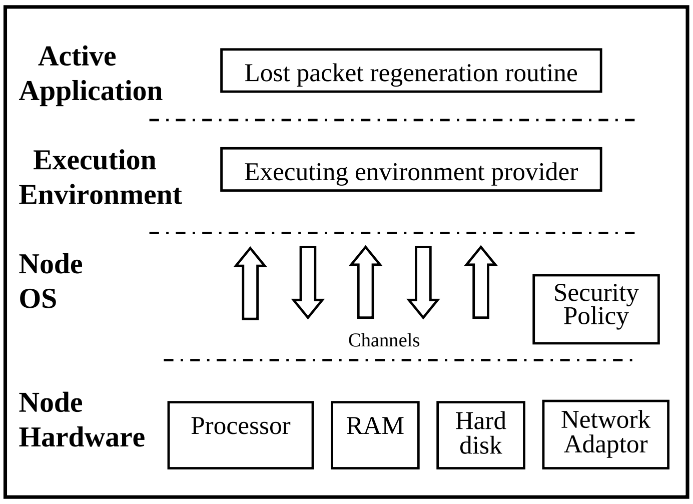

Figure 4.9: Active node software structure.

4.6.3 Node structure

Active nodes are responsible for providing an executing environment for active packets and for helping them to perform their related tasks. A typical active node structure is shown in Figure 4.9.

Each active packet may have some Active Applications (AA) and data. It must be noted that an active packet may have more than one active application. Each active voice packet has an active application, which is called up by a *pointer to function*, carried by the active packet. Each active node runs a Node Operating System (NodeOS) and one or more Execution Environments (EE). The execution environment, which is responsible for providing an interface between AA and NodeOS, operates as a 'shell' in ordinary computer systems [14] and it provides a framework for the activity that arises from incoming events/packets. Each EE implements a virtual machine that interprets active packets, arriving at a node. Furthermore, an EE isolates a packet's executing environment from other types of program running circumferences that may be concurrently executing in the node. Every AA must be assigned to an EE in order to be executed.

As soon as an active packet arrives at an active node, the node creates an execution environment for the packet. The created EE provides certain available resources, which are necessary for that application. The NodeOS is the most important part of an executive section and it is responsible for allocating, scheduling and managing the system resources (CPU cycles, memory, link bandwidth etc.). Besides that, it prepares a set of interfaces allowing EE's to access and use those resources. The NodeOS should allocate and resorb the node's facilities to each EE and it should prevent a malfunctioning EE to waste all of the node's resources [14]. In the lower row of the Figure, the node hardware is outlined. This includes a processor, memory, storage module, network adaptor as well as other equipment.

4.6.4 Node resource reservation

The driving motivation behind active networking, is to provide a framework that facilitates the execution of a program at the active nodes. In order to execute the program codes, an active node must provide a good environment, namely, one which provides processing power for each active packet. Processing power involves processor time, memory, I/O units, storage and other node facilities [36]. The two main factors are CPU and memory reservations as shown in the following subsections.

- CPU reservation: The lost packet regeneration routine of each method needs a specific time to recover the lost packets. This time varies from node to node because of the nodes' structure. Goltier et al. classified the sources of time variability related to the node hardware as the following [35]:

  - Frequency of the processor,
  - Architecture of the processor,
  - Amount of system memory,
  - Speed of buses,
  - Technology of the persistent storage, and
  - Type of network card.

  Apart from these parameters, the amount and type of processor cache is another factor which influences the processing time. In regard to software, Galtier and his colleagues paid attention to the node's software structure and the implementation of its operating system. Based on some practical experiments, they concluded that even on a specific machine, identical NodeOS system calls could require different CPU time [35, 36]. To sum up, it is not easy to designate a constant time for recovering the lost packets within different active nodes over the whole network. For the sake of simplicity, reservation in this instance is explained for a specific node. The lost packet regeneration routine needs an execution time that is contained between two constant times.

  - Lower limit: The minimum time is required when there are no lost packets. The node needs this time to check the packets' sequence and to detect whether the packets have all arrived.
  - Higher limit: In the worst case, this time is required for the node to regenerate all the lost packets using the previously arrived packets.

  For example, in the simulation study conducted with Pentium IV and 512 Mbyte RAM, the worst case for CADPCM lasts 200 μsec and the interval between two successive

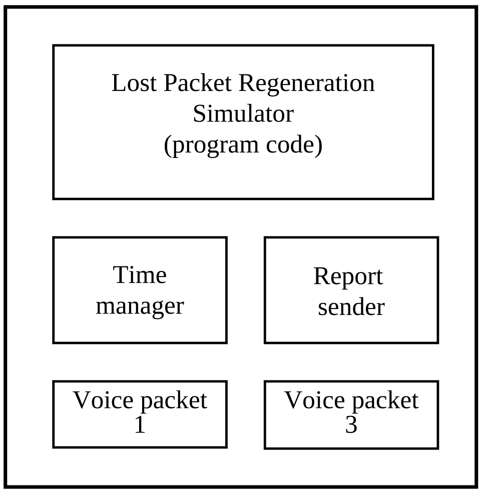

Figure 4.10: Calibrator packet structure.

packets is 20 ms. Therefore, it is necessary to reserve 200 μsec out of each 20 ms period to recover those packets. As discussed earlier, this processing time varies from node to node, so it is suggested to send a test packet containing a specific routine to each node so that it can measure its execution time. The Calibrator Packet (CP) contains a program code that simulates the worst case packet recovery for a specific algorithm. For example, the calibrator packet for DADPCM5 contains the code for regenerating three packets from two non-consecutive packets, which would be the worst case for DADPCM5 in a very lossy network environment. Figure 4.10 shows the proposed calibrator packet structure.

CP contains a lost packet regenerator simulator program which acts as the real recovery program. It simulates the sequence of regenerating three lost packets from the two accessible packets included in the CP structure. When the calibrator packet is executed on the node, the time-manager block measures the execution time of the routine. It can measure both low and high limits of the packet recovery time and it reports those limits by generating and sending a report packet. Therefore, the network manager can reserve an appropriate time for that algorithm.

- Memory reservation: The required memory for each recovery algorithm can be divided into four groups.

  - Permanent code memory: Part of an active node's main storage is used to store the program code of the lost packet detection and regeneration routine. It needs few kilobytes of memory.

  - Temporary code memory: Each arrived voice packet calls up the recovery routine. The routine needs a few kilobytes of memory to be executed and that memory can be freed when the recovery routine is finished.

  - Permanent status memory: Each active node needs to keep track of all voice packets related to a session, so an active node needs a few bytes of RAM to maintain the status of the voice packets and their sequences. In addition, a node needs a few bytes of RAM to save the status of each voice packet series.
  - Permanent data memory: In the case where a packet is lost, active nodes are expected to regenerate the lost packet using relevant data received by the adjacent packets. Therefore, the nodes need to have some data memory to save the packets. This memory is limited to one packet length for DADPCM, DFCPM and DPCM and is limited to a maximum of four packets for DADPCM5.

4.6.5 Node software specifications

Active nodes limit all the resources offered to an active application. The nodes strategically define a framework within which the active application must work. The limitations are applied for the following reasons.

- Protection: For security reasons, the active node may apply restrictions on some of its vital resources and programs.
- Avoiding over-consumption of resources: An active application that over-uses the node's capability causes starvation of other AAs. Nodes may apply restrictions on the following issues:
  - CPU time,
  - Memory, and
  - Bandwidth (An active application might generate many packets and wish to send them to other nodes which could increase the network load).

Apart from meeting the node reservation limitation, it is suggested that software which is written for active nodes should obey the following rules.

- Portability: The program code must be written in such a way that it can be executed in any active node. This means that it has to be independent of the node hardware and software structure, and that it must not require a special environment to be executed.
- Security: It must be secure from other programs and viruses that may want to change it.
- Robustness: The program code must be completely robust and reliable so that no active packet can force it to undertake unpredictable tasks.

- Release: The program must not use resources that cannot be freed after finishing the task. There should not be any program or part of a program still running after the packet has left the execution section. For example, it seems that there must not be any *timer* in a node's software structure. As timers measure time between two events, they take resources of nodes (mainly, CPU time) and use them for a considerable time. Apart from the above mentioned reasons, it is avoided to use timers because the arrival time of a subsequent packet is not predictable. This is because the previous network might be a packet-switched one, and so might not be able to guarantee the timely delivery of the packets.

Although active nodes are responsible for providing an execution environment for active packets by allocating resources to them, they are also responsible for resorption of the program execution facilities in order to dedicate them to other packets. Otherwise, networks suffer from over-consumption of resources [35].

4.6.6 Node software structure for voice packet recovery

A lost packet regeneration program is installed in the node's structure. Voice packets that have a *pointer-to-function* for that purpose, call this *pre-stored* program. The program is written on an event-driven basis and it works on the *state* of a voice flow and its input *events*. As each packet is an *event* for each node, it is possible to call up that program as a packet-driven software. This software is based on ITU-T recommendation Z100, Specification and Description Language (SDL). From the active node point of view, every voice series stays in a *state*. In each state the software receives an event and calls up a function, which performs the tasks related to that state and that particular event. Then there is a change of state and software awaits the arrival of the next event (packet). The software expects to receive all possible events related to that specific state. Figure 4.11 shows possible states, events, and action symbols in each branch of SDL.

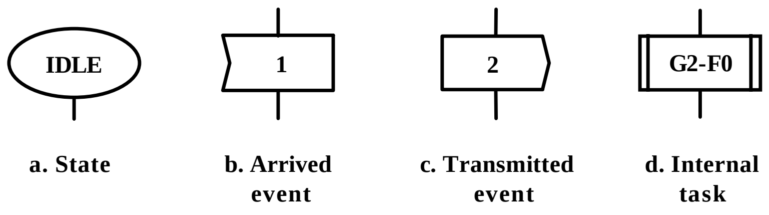

Figure 4.11: SDL symbols.

Figure 4.11.a shows an IDLE state and Figure 4.11.b shows an incoming event, an arrived packet. The symbol shows that packet number one has arrived. The symbol in Figure 4.11.c shows the sending of packet number two to the next node. The last symbol is used when the

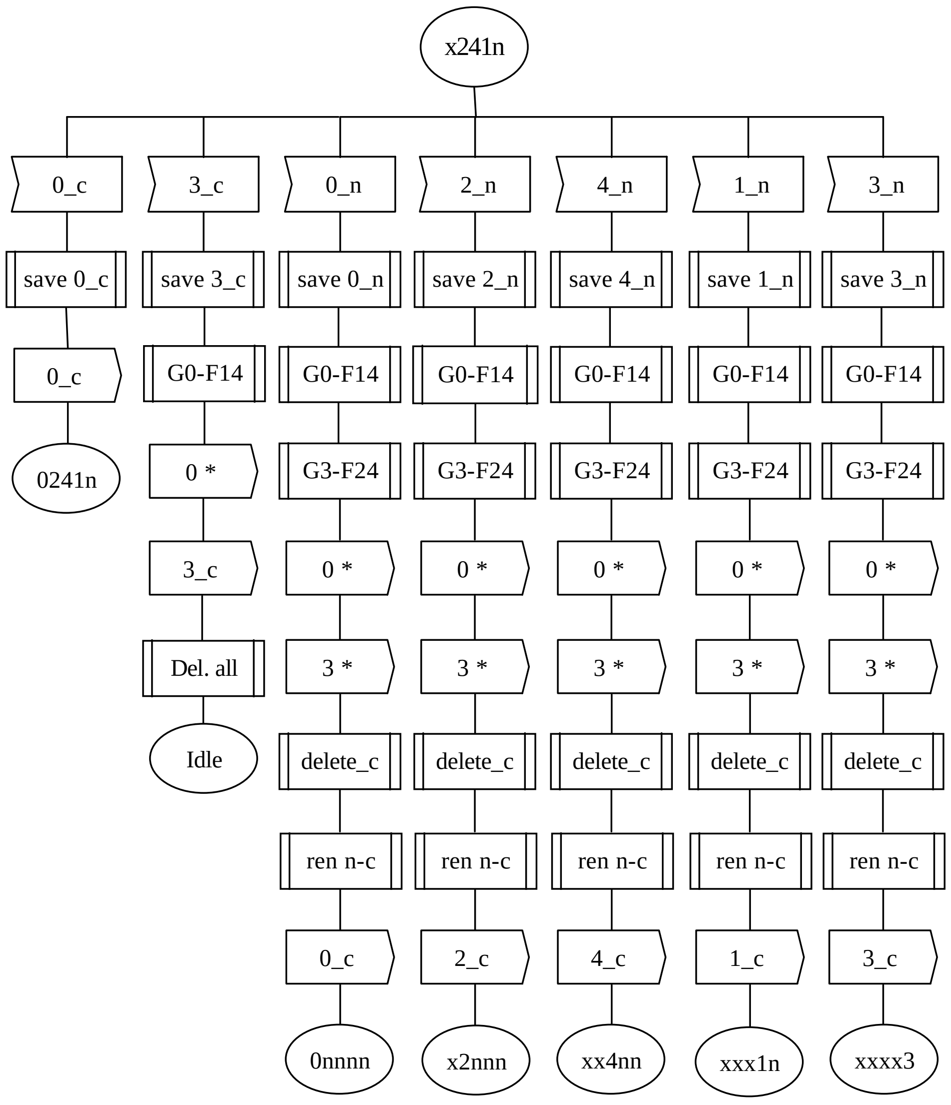

Figure 4.12: SDL x241n of DADPCM5.

program needs to perform an internal task such as generating packet number two from packet number zero. For this structure, the software related to all states and events was written and the whole program was installed in an active node. A *state-event* table, which holds all states as rows and all possible events as columns, was generated. Each cell of the table that presents a particular event in a specific state has a pointer-to-function point to a branch of SDL. All that the node requires is the state of the voice flow, and the type of event (packet) arrival to find the pointer and call the function. The function completes the tasks of the application and then there is a change of state and software awaits the next packet. The procedure of calling a SDL branch related to the DADPCM5 coding scheme is explained briefly. Suppose that the voice flow is in $x241n$ state and a packet has arrived at the active node. The SDL for this state is shown in Figure 4.12.

A voice flow in state $x241n$ may receive packets numbers 0 and 3 from the current voice

Event / State	0_c	3_c	0_n	2_n	4_n	1_n	3_n
x241n	EPL_x241n_0c	EPL_x241n_3c	EPL_x241n_0n	EPL_x241n_2n	EPL_x241n_4n	EPL_x241n_1n	EPL_x241n_3n
0241n	EPL_0241n_0c	EPL_0241n_3c	EPL_0241n_0n	EPL_0241n_2n	EPL_0241n_4n	EPL_0241n_1n	EPL_0241n_3n
Idle	EPL_idle_0c	EPL_idle_3c	EPL_idle_0n	EPL_idle_2n	EPL_idle_4n	EPL_idle_1n	EPL_idle_3n
0nnnn	EPL_0nnnn_0c	EPL_0nnnn_3c	EPL_0nnnn_0n	EPL_0nnnn_2n	EPL_0nnnn_4n	EPL_0nnnn_1n	EPL_0nnnn_3n
x2nnn	EPL_x2nnn_0c	EPL_x2nnn_3c	EPL_x2nnn_0n	EPL_x2nnn_2n	EPL_x2nnn_4n	EPL_x2nnn_1n	EPL_x2nnn_3n
xx4nn	EPL_xx4nn_0c	EPL_xx4nn_3c	EPL_xx4nn_0n	EPL_xx4nn_2n	EPL_xx4nn_4n	EPL_xx4nn_1n	EPL_xx4nn_3n
xxx1n	EPL_xxx1n_0c	EPL_xxx1n_3c	EPL_xxx1n_0n	EPL_xxx1n_2n	EPL_xxx1n_4n	EPL_xxx1n_1n	EPL_xxx1n_3n
xxxx3	EPL_xxxx3_0c	EPL_xxxx3_3c	EPL_xxxx3_0n	EPL_xxxx3_2n	EPL_xxxx3_4n	EPL_xxxx3_1n	EPL_xxxx3_3n

Figure 4.13: State-event table of DADPCM5.

series and all packets from the next voice series. Therefore, it must have an Event Processor Logic (EPL) for each event. The state-event table for this software is shown in Table 4.13.

For example, when packet 3 arrives at this stage, the active node indexes the $x241n$ state and 3_C event in the table and finds a pointer to the EPL_x241n_3c function. The node calls the function and executes it. The EPL_x241n_3c function is:

```
void epl_x241n_3c (void)
{
    save_received_packet();
    mark_saved_packet(3);
    generate_lost_packet_from(1, 4);
    name_regenerated_packet(0);
    send_packet(0);
    send_packet(3);
    delete_all_memory();
    change_state(IDLE);
}
```

As can be seen, the EPL_x241n_3c function processes the packet and then changes the state to *IDLE*. The next arrived packet (event) will find the flow in the IDLE state and the node will find another EPL for that event.

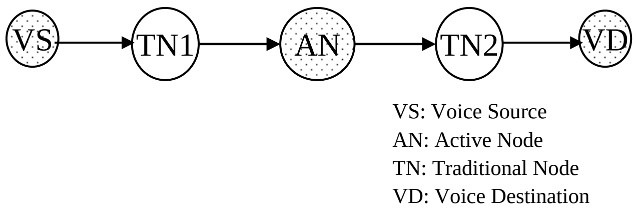

Figure 4.14: Network topology for mathematical analysis.

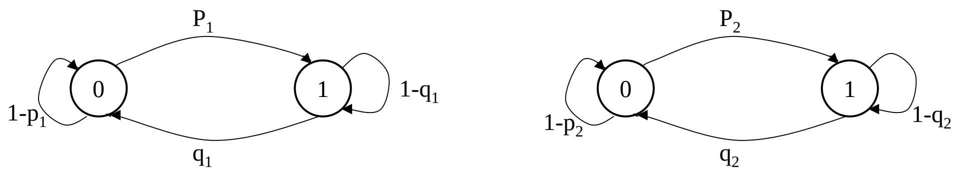

Figure 4.15: Gilbert loss model for nodes TN1 and TN2.

4.7 Role of active node

Active nodes play a simple but significant role in improving the voice quality herein when the communication link suffers from heavy packet loss problems. To analyze their role, the following simple topology of the network is presented.

The voice sender (VS) packs and sends packets to the voice destination (VD) node via a path that has an active node (AN) in its structure. There are two traditional nodes named TN1 and TN2 that drop packets in the Gilbert model.

In this drop model, each traditional node has two states, state $X = 0$ represents the state of transferring a packet successfully and state $X = 1$ represents the state of dropping a packet. The Gilbert probability formulae in Subsection 2.2.3 are used in all analysis.

To analyze the role of an active node in regenerating lost packets, one should pay attention to the voice coding schemes. The proposed schemes can easily regenerate individual lost packets using other arrived packets. Thus, in Figure 4.14, if the receiver envisages some nonconsecutive lost packets, no matter whether the active node is there or not, it can easily recover the lost packets and produce good voice quality. Therefore, it is desired to discover the probability of having two or more consecutive lost packets at the receiver. To investigate the improvement of voice quality by active nodes, many analyses were done. The first analysis is valid when the regenerated packets are 100% identical to the lost packets (when the redundant data can regenerate perfect packets and insert them in place of the lost packets). Other analyses investigated more general situations.

4.8 First analysis

For the sake of simplicity, the first part of this analysis is written for two consecutive losses and a general n consecutive lost packet situation is stated in the next part.

4.8.1 Two consecutive losses

In the following subsections the probability of having two consecutive lost packets when the active node is operative (on) and when it is not operative (off) is calculated and it is shown that when the active node generates lost packets, the probability is much lower than when the node is not operative.

Active node is off

In this situation, the active node acts as an ordinary node, transferring all arrived packets to the receiver (all drops before TN2 are assumed to happen in TN1 and the active node is considered loss-free, passing all packets to the receiver).

The probability of having two consecutive lost packets at the receiver can be calculated by adding:

- the probability of losing two consecutive packets in TN1 when TN2 passes all packets,
- the probability of losing one packet in TN1 and one packet in TN2, and
- the probability of losing two consecutive packets in TN2 when TN1 passes all packets

as

$$\begin{aligned} P_{off} \; = & \frac{q_1}{p_1+q_1}\, p_1\,(1-q_1)\,q_1\,\frac{q_2}{p_2+q_2}\,(1-p_2) \; + \\ & \frac{q_1}{p_1+q_1}\, p_1\, q_1\,\frac{q_2}{p_2+q_2}\, p_2\, q_2 \; + \\ & \frac{q_1}{p_1+q_1}\,(1-p_1)\,\frac{q_2}{p_2+q_2}\, p_2\,(1-q_2)\,q_2. \end{aligned} \tag{4.1}$$

Active node is on

In this case, and according to the voice-packing scheme, the active node generates one lost packet out of each series of consecutive losses. Therefore, the probability of having two consecutive lost packets at the receiver can be calculated by adding:

- the probability of losing three consecutive packets in TN1 when TN2 passes all packets,

- the probability of losing two consecutive packets in TN1 and losing one packet in TN2 (In this particular case, TN2 may lose a packet before, after or containing the regenerated packet by TN1. Assuming that the regenerated packet is 100% identical to the lost packet, all of these cases provide the same result), and
- the probability of losing two consecutive packets in TN2 when TN1 passes all packets.

In the last item, TN1 can lose a packet and active node can regenerate it. For the sake of simplicity, this situation is not considered in

$$
\begin{aligned}
P_{on} = & \frac{q_1}{p_1+q_1}\, p_1\, (1-q_1)^2\, q_1\, \frac{q_2}{p_2+q_2}\, (1-p_2)^2 \; + \\
& \frac{q_1}{p_1+q_1}\, p_1\, (1-q_1)\, q_1\, \frac{q_2}{p_2+q_2}\, p_2\, q_2 \; + \\
& \frac{q_1}{p_1+q_1}\, (1-p_1)^2\, \frac{q_2}{p_2+q_2}\, p_2\, (1-q_2)\, q_2
\end{aligned}
\tag{4.2}
$$

which obviously does not make much difference. Comparing both formulae it can be seen that:

- the first term of (4.2) is equal to the first term of (4.1) multiplied by $(1-q_1)\,(1-p_2)$,
- the second term of (4.2) is equal to the second term of (4.1) multiplied by $(1-q_1)$, and
- the last term of (4.2) is equal to the last term of (4.1) multiplied by $(1-p_1)$.

As $(1-q_1)$, $(1-p_1)$ and $(1-p_2)$ have small values (obviously, less than one), the probability of having two consecutive lost packets when the active node is operative is much less than when the active node is inoperative. Therefore, the regenerated voice will have much better quality when the active node is operative.

4.8.2 n consecutive losses

To investigate the degree of voice quality improvement by active node, the general form of having n consecutive lost packets at the receiver is considered.

Active node is off

The probability of having n consecutive lost packets can be calculated by adding:

- the probability of losing n consecutive packets in TN1 when TN2 passes all packets,

- the probability of losing $n-1$ consecutive packets in TN1 when TN2 loses one packet,
- the probability of losing $n-2$ consecutive packets in TN1 when TN2 loses two consecutive packets,
- ...,
- the probability of losing one packet in TN1 when TN2 loses $n-1$ consecutive packets, and
- the probability of having TN1 as a loss-free node when TN2 loses n consecutive packets

as

$$\begin{aligned}P_{off} &= \frac{q_1}{p_1+q_1}\, p_1\,(1-q_1)^{n-1}\, q_1\, \frac{q_2}{p_2+q_2}\,(1-p_2)^{n-1}\; + \\ &\frac{q_1}{p_1+q_1}\, p_1\, q_1\, \frac{q_2}{p_2+q_2}\, p_2\, q_2 \sum_{i=0}^{n-2}(1-q_1)^i\,(1-q_2)^{n-2-i}\,(1-p_1)^{n-i-2}\,(1-p_2)^i \\ &\frac{q_1}{p_1+q_1}\,(1-p_1)^{n-1}\,\frac{q_2}{p_2+q_2}\, p_2\,(1-q_2)^{n-1}\, q_2. \end{aligned} \tag{4.3}$$

Active node is on

Since the active node generates one lost packet out of each series of consecutive losses, the probability of having n consecutive lost packets at the receiver can be calculated by adding:

- the probability of losing $n+1$ consecutive packets in TN1 when TN2 passes all packets,
- the probability of losing n consecutive packets in TN1 when TN2 loses one packet,
- ..., and
- the probability of having TN1 as a loss-free node when TN2 loses n consecutive packets

as

$$\begin{aligned}P_{on} &= \frac{q_1}{p_1+q_1}\, p_1\,(1-q_1)^{n}\, q_1\, \frac{q_2}{p_2+q_2}\,(1-p_2)^{n}\; + \\ &\frac{q_1}{p_1+q_1}\, p_1\, q_1\, \frac{q_2}{p_2+q_2}\, p_2\, q_2 \sum_{i=0}^{n-2}(1-q_1)^{i+1}\,(1-q_2)^{n-2-i}\,(1-p_1)^{n-i-1}\,(1-p_2)^{i+1} \\ &\frac{q_1}{p_1+q_1}\,(1-p_1)^{n}\,\frac{q_2}{p_2+q_2}\, p_2\,(1-q_2)^{n-1}\, q_2. \end{aligned} \tag{4.4}$$

Again by comparing both formulae it can be seen that:

- the first two terms of (4.4) are equal to the first two terms of (4.3) multiplied by $(1 - q_1)\ (1 - p_2)$, and
- the last term of (4.4) is equal to the last term of (4.3) multiplied by $(1 - p_1)$.

To sum up, the formulae prove that when the active node is operative, the probability of having n consecutive lost packets is much less than when the active node is not operative. The voice quality degrades drastically with consecutive lost packets and active nodes can reduce the probability of having such lost packets in the network, thus improving the voice quality. Therefore, It is considered that an operative active node improves the quality of regenerated sound.

4.9 Second analysis

As mentioned in the first analysis, the proposed coding schemes help the receiver overcome all individual lost packets (whether the active node is operative or not). To investigate the role of an active node, this analysis calculates the probability of having consecutive lost packets which cannot be covered by most coding schemes. For the sake of simplicity, the first part is dedicated to finding the probability of having two consecutive lost packets and a general n consecutive lost packets situation is considered in the next part. In the following subsections, symbol V_n stands for the n^{th} voice packet, which has arrived properly, symbol R indicates a regenerated packet and L^n represents the n consecutive lost packets.

4.9.1 Two consecutive losses

This Subsection tries to find the possibility of having a $VRLV$ pattern at the output stage of the receiver in Figure 4.14. The pattern symbolizes two consecutive lost packets in which one of them has been regenerated (either in active node or at the receiver). The possibility is considered in two situations depending on the status of the active node.

Active node is off

All possible situations that lead to such a packet pattern at the end stage are listed as follow.

- Losing two consecutive packets in TN1 when TN2 passes all packets (V_x and V_{x+1} are lost in TN1).
- Losing one packet in TN1 and another packet in TN2 which has two possibilities.

  - V_x is lost in TN1 and V_{x+1} is lost in TN2.
  - V_x is lost in TN1 and V_{x-1} is lost in TN2.

- Losing two consecutive packets in TN2 when TN1 passes all packets (V_x and V_{x+1} are lost in TN2).

Active node is on

In this case, the active node generates one lost packet and inserts it in the network. All possible situations that generate the $VRLV$ pattern at the output stage are:

- losing two consecutive packets in TN1 when TN2 passes all packets (V_x and V_{x+1} are lost in TN1),
- losing one packet in TN1 and another packet in TN2 (V_x is lost in TN1 and V_{x-1} is lost in TN2), and
- losing two consecutive packets in TN2 when TN1 passes all packets (V_x and V_{x+1} are lost in TN2).

As the worst case, the second term has only one possible situation because the other case (V_x is lost in TN1 and V_{x+1} is lost in TN2) leads to another packet pattern ($VRRV$) which is much better than the desired $VRLV$ pattern. Adding those four possibilities when the active node is off, and adding these three possibilities when the active node is on, leads to the conclusion that: The possibility of having $VRLV$ pattern at the output stage is less when the active node is on, compared to when the node is off.

4.9.2 n consecutive losses

In this Subsection, the probability of having a VRL^nV pattern, that is, the probability of having n consecutive lost packets and one regenerated packet, is calculated.

Active node is off

All possible situations that produce such a packet pattern are listed as follow.

- Losing $n+1$ consecutive packets in TN2 when TN1 passes all packets ($V_x \ldots V_{x+n+1}$ are lost in TN2).
- Losing n consecutive packets in TN2 when TN1 drops one packet, which has two possibilities.

  - V_x is lost in TN1 and $V_{x-1} \ \ldots \ V_{x-n}$ are lost in TN2.
  - V_x is lost in TN1 and $V_{x+1} \ \ldots \ V_{x+n}$ are lost in TN2.

- Losing $n-1$ consecutive packets in TN2 when TN1 drops two consecutive packets (two possibilities).
  - V_x, V_{x+1} are lost in TN1 and $V_{x-1} \ \ldots \ V_{x-n-1}$ are lost in TN2.
  - V_x, V_{x+1} are lost in TN1 and $V_{x+2} \ \ldots \ V_{x+n}$ are lost in TN2.

- ...

- Losing $n+1$ consecutive packets in TN1 when TN2 passes all packets ($V_x \ \ldots \ V_{x+n+1}$ are lost in TN1).

if X^n represents the probability of losing n consecutive packets in TN1 and Y^n represents the probability of losing n consecutive packets in TN2, the total probability is given by

$$P_{off} \ = \ X^0\,Y^{n+1} \ + \ X^1\,Y^n \ + \ X^2\,Y^{n-1} \ + \ \ldots \ + \ X^{n+1}\,Y^0. \tag{4.5}$$

Suppose that $P_{a,b}$ represents the probability of losing packets number a to b in TN1 and $Q_{a,b}$ represents the probability of losing packets number a to b in TN2. Based on the fact that TN2 loses packets which are located either before or after the lost packets (caused by TN1), the $X^1\,Y^n$ consists of

$$X^1\,Y^n \ = \ P_{x,x}\,Q_{x-n,x-1} \ + \ P_{x,x}\,Q_{x+1,x+n}. \tag{4.6}$$

What is mentioned for $X^1\,Y^n$ is applicable for all other $X^a\,Y^b$ except when a or b are zero. Therefore, the total probability is given by

$$P_{off} \ = \ X^0\,Y^{n+1} \ + \ \sum_{i=1}^{n} P_{x,x+i-1}\,Q_{x-n+i-1,x-1} \ + \ \sum_{i=1}^{n} P_{x,x+i-1}\,Q_{x+i,x+n} \ + \ X^{n+1}\,Y^0. \tag{4.7}$$

Active node is on

In this situation, the following possible cases produce the desired packet pattern.

- Losing $n+1$ consecutive packets in TN2 when TN1 passes all packets ($V_x \ \ldots \ V_{x+n+1}$ are lost in TN2).

- Losing n consecutive packets in TN2 when TN1 drops one packet (V_x is lost in TN1 and $V_{x-1} \ \ldots \ V_{x-n}$ are lost in TN2.

- Losing $n-1$ consecutive packets in TN2 when TN1 drops two consecutive packets (V_x, V_{x+1} are lost in TN1 and $V_{x-1}\ \dots\ V_{x-n-1}$ are lost in TN2).
- ...
- Losing $n+1$ consecutive packets in TN1 when TN2 passes all packets ($V_x\ \dots\ V_{x+n+1}$ are lost in TN1).

The probability of having such a pattern at the output is given by

$$P_{on} \ = \ X^0\, Y^{n+1} \ + \ \sum_{i=1}^{n} P_{x,x+i-1}\, Q_{x-n+i-1,x-1} \ + \ X^{n+1}\, Y^0. \tag{4.8}$$

When the active node is operative the probability of having this pattern is less than the probability when the node is off because (4.8) does not have the second term of summation in (4.7). Therefore, the active node reduces the probability of that pattern and it can be concluded that the active node can improve the voice quality at the receiver.

4.10 Third analysis

In contrast with the previous analysis, this one attempts to assess voice quality when packets are dropped in the network. Using the packet notation from the previous analysis (V for voice, R for regenerated and L for lost packets) and considering Figure 4.14 as the topology of the network, this analysis compares the number of consecutive lost packets at the output stage when the active node is on and when it is off. Suppose that the sender transmits an infinite number of voice packets and TN1 and TN2 lose n and m consecutive packets respectively.

Active node is off

Losing n consecutive packets in TN1, the packet pattern at the input point of the active node will be VL^nV and as the active node is off, the output of the active node will have the same pattern. Losing m consecutive packets in TN2 causes the receiver to have VL^mL^nV packet pattern at its input point and as the receiver generates one packet, the output voice will be generated from $VRL^{n+m-1}V$ pattern (m and n are bigger than zero).

Active node is on

The input point of the active node has the VL^nV pattern and as the active node is operative, it regenerates one packet and the pattern at the output of the active node will be $VRL^{n-1}V$. At

Table 4.1: Possible packet patterns in the test network

Location	AN is off	AN is on	
Sender	VV	VV	VV
AN input	VL^nV	VL^nV	VL^nV
AN output	VL^nV	$VRL^{n-1}V$	$VRL^{n-1}V$
Rec. input	VL^mL^nV	$VL^mRL^{n-1}V$	$VL^mL^{n-1}V$
Rec. output	$VRL^{m+n-1}V$	$VRL^{m-1}RL^{n-1}V$	$VRL^{m+n-2}V$

this stage the TN2 loses m consecutive packets which might happen in two different ways:

- TN2 loses m packets after the regenerated packet, or
- TN2 loses m packets containing the regenerated packet.

In the first instance, the receiver will envisage $VL^mRL^{n-1}V$ and regenerate another packet from the second lost series, so the output voice will be generated by $VRL^{m-1}RL^{n-1}V$. The second instance forces the receiver to have the $VL^mL^{n-1}V$ pattern and after regenerating a packet, the voice will be reproduced by $VRL^{m+n-2}V$ pattern. Table 4.1 summarizes all the packet patterns in the network.

Comparing the lost length of the pattern when the active node is on and off, it can be seen that the output stage envisages $(m + n - 1)$ consecutive lost packets when the active node is off and it envisages $[m + n - 2]$ or $[(m - 1)$ and $(n - 1)]$ consecutive lost packets when the active node is on.

One can conclude that, in the worst case (the second way of losing packets in TN2), the active node reduces the number of consecutive lost packets by one. Therefore, it is obvious that the active node improves the quality of sound at the receiver. Apart from regenerating lost packets, using a small buffer, the active node can reorder the arrived out-of-order packets and send them to the destination. Therefore, the overall delay in VOIP will be reduced.

4.11 Active node location

In order to find the best place to have an active node to do ”on-the-fly” regeneration of the lost packets, Le et al. chose a simple packet loss model (Bernoulli model) [57]. This Subsection extends their work by using the Gilbert model. Figure 4.16 shows the network topology.

Suppose the first lossy network cloud drops packets with the probability of p_1 and q_1 and the second cloud loses packets with p_2 and q_2. With Regard to (2.8), the probability of having

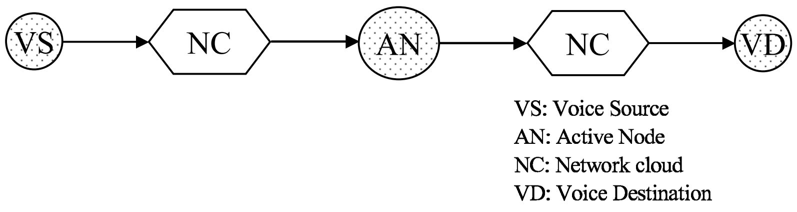

Figure 4.16: Simple network topology for tests.

k consecutive packets at the receiver is given by

$$P = [p_1(1-p_1)^{k-1}][p_2(1-p_2)^{k-1}]. \tag{4.9}$$

To find the best placement of the active node, suppose that the whole probability of losing a packet in the network is a constant value (N). Therefore, $p_1 + p_2 = N$ and the probability is given by

$$P = [p_1(1-p_1)^{k-1}][(N-p_1)(1-N+p_1)^{k-1}]. \tag{4.10}$$

For the sake of simplicity, suppose that $k = 2$. It then follows that

$$\begin{aligned} P &= (p_1 - p_1^2)(N - N^2 + (2N-1)p_1 - p_1^2) \\ &= p_1^4 - 2Np_1^3 + (N^2+N-1)p_1^2 + (N-N^2)p_1. \end{aligned} \tag{4.11}$$

To have minimum P, $\frac{\partial P}{\partial p_1} = 0$, as a result

$$4p_1^3 - 6Np_1^2 + 2(N^2+N-1)p_1 + (N-N^2) = 0. \tag{4.12}$$

Following this calculation, $p_1 = \frac{N}{2}$ and from (4.9), $p_2 = \frac{N}{2}$.

This shows that the optimal place to install an active node is where the packet loss rate from the sender to that location is equal to the loss rate from that location to the receiver. According to Chapter 2, active nodes have a benefit at the interface between a wireless network and the optical fibre-based Internet.

4.12 Simulation and test environment

In spite of the fact that the simulation may alter the overall performance of the algorithms and may hide some details, it was decided to use a network simulator on a computer for a practical test environment. Using packet-level simulation, it is possible to study the overall packet behaviour and the data can be collected at any node in the network. The test environment should provide some facilities for detailed information about voice-packing schemes. Therefore, it must be able to do the following tasks.

- Implement active nodes.
- Implement different packet loss patterns.
- Mark different packets with different headers.
- Specify the source and destination of each packet.
- Generate and change background traffic.
- Change the packet size.
- Change links' speed and their packet dropping strategy.
- Alter the buffer size of each node in the whole network.
- Change the transmission delay of each link.
- Support different transmission protocols.

It was decided to use a well-designed network simulator, which has a model of reality with many real-world implementations. Consequently, the *Network Simulator 2* (NS-2) is chosen. NS-2 was developed as a part of Virtual Inter Network Testbed (VINT) project by the University of Southern California, Berkeley and some other network researchers [33].

NS-2, which works under the Unix operating system, needs simulation codes written in C++ and Object Tool Command Language (Otcl). Its software architecture provides high-performance packet level simulation (using C++) besides a flexible configuration environment (using an interpreted language Otcl [34]. There is one-to-one correspondence between a class in C++ and a class in Otcl. NS-2 is a discrete event simulator for networking research, which works at the packet level. It provides substantial support to simulate different protocols such as TCP, UDP, FTP and HTTP for unicast and multicast sessions with simple or complex topologies. It supports different queue management mechanisms such as Drop tail, Random Early Detection (RED) and Class-Based Queueing (CBQ). Moreover, NS-2 supports the event scheduler in which an event is a unique packet with scheduled time and an object that handles the event by a pointer. NS-2 uses the drawn strategy in Figure 4.17 to make new objects and work with its current objects.

A file with Otcl script gives the configuration and control information for simulation to NS-2. The simulation objects such as nodes, links and all traffic sources are described in the script and immediately mirrored in the compiled hierarchy. The Otcl file determines the number and configuration of the network nodes, the links and their specifications (physical setup), the protocols used in the connections and the start and stop time of each flow (simulation topology).

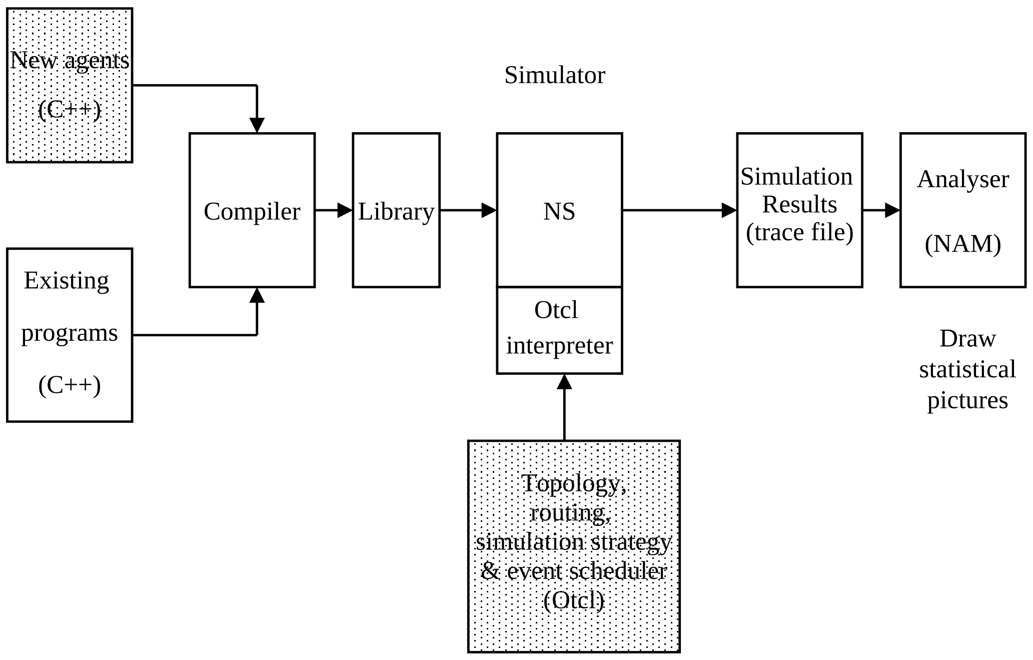
New agents (C++)
Simulator
Compiler
Library
NS
Simulation Results (trace file)
Analyser (NAM)
Existing programs (C++)
Otcl interpreter
Draw statistical pictures
Topology, routing, simulation strategy & event scheduler (Otcl)

Figure 4.17: NS-2 software structure.

One of the output files of NS-2 is a file with *.nam* extension. It has a complete description of the simulation, which gives detailed information about each packet. For simplification, a Network Animator software is available that allows the study of the simulation results graphically.

4.12.1 Active packet structure in the simulations

In order to have active packets, a data structure with different fields was chosen.

- Seq-no: Two bytes that hold the sequence number of each voice flow.
- Send-time: Four bytes that contain the packet creation time.
- Data: An array of bytes, which carries the main data of the flow. The array size varies with the voice-coding scheme.
- Data-type: A byte to identify the type of the data field. It indicates the first, last and other packets of a session.
- Data-length: Two bytes, which hold the data length in the data field.
- Packing-type: An identifier byte, which defines the voice-packing scheme. Different packing methods have different numbers.

- Nof-important: A byte to hold the number of important bytes in the main data array.
- Imp-address: A byte-array, which holds the address of important bytes.
- Imp-data: A byte array to keep the important data.
- Imp-parm: Two bytes to indicate the threshold level in some packing schemes.
- Decode-sample: Two bytes to hold the decoding sample in some voice-packing schemes. They are used for ADPCM related coding methods.
- Decode-index: A byte to hold a pointer to a decoder table for ADPCM related coding schemes.
- Indicator-index: A byte to define the way of indicating the important bytes in some coding schemes.

It must be considered that the mentioned data structure is a general structure that can be used for all voice-packing algorithms introduced herein. Some fields are not used for some specific methods.

4.12.2 Active node implementation

The main code to implement an active node in NS-2 is written in the form of C++ files, which must be compiled and linked to the other parts of NS-2. In the following lines code snippets are only shown when needed. The complete codes for implementation are detailed in Appendix B.

Active agent header

The first step for a new service in NS-2 is to write a program in C++ to define the framework. The object-oriented environment in NS-2 helps the user with an active-agent to implement active nodes (the agent is an autonomous software entity that can interact with its environment). The following program defines an active-agent in NS-2.

```
class ActiveAgent : public Agent {
 public:
  ActiveAgent();
  int command(int argc, const char * const * argv);
  void recv(Packet*, Handler*);
  int m_param;
  int node_status;
  int prob_p1, prob_q1, prob_p2, prob_q2;
  protected :
    int off_active_;
};
```

An integer parameter named *m-param* is passed to the agent to set the coding parameter in some voice-packing schemes.

Active class

In order to use the agent in NS-2, it is necessary to define a class. The following program generates an active-class from the active-agent.

```
 static class ActiveHeaderClass : public PacketHeaderClass {
  public :
    ActiveHeaderClass() : PacketHeaderClass("PacketHeader/Active", sizeof(hdr_active)) {}
  }class_activehdr;

static class ActiveClass : public TclClass {
  public :
    ActiveClass() : TclClass("Agent/Active") {}
    TclObject * create(int, const char * const*) {
    return (new ActiveAgent());
    }
} class_active;
```

NS-2 uses compiled C++ codes plus interpreted Otcl scripts to handle a simulation. To transfer parameters between C++ and Otcl, a simple binding code must be used:

```
 ActiveAgent :: ActiveAgent() : Agent(PT_ACTIVE) {
  bind("packetSize_", &size_);
  bind("off_active_", &off_active_);
  bind("m_param_tcl", &m_param);
  bind("node_status_tcl", &node_status);
  bind("prob_p1_tcl", &prob_p1);
  bind("prob_q1_tcl", &prob_q1);
  bind("prob_p2_tcl", &prob_p2);
  bind("prob_q2_tcl", &prob_q2);
}
```

Active agent

A simple instruction in Otcl is enough to generate and send a packet.

```
#ns at 2 "$p0 send pcm"
```

After two seconds from the beginning of simulation, this instruction generates a packet and passes "send" as the first and "pcm" as the second argument to it. The above instruction needs the following program to check the arguments, allocate a packet and assign an active header to it.

```
int ActiveAgent::command(int argc, const char *const * argv)
{
  if (strcmp(argv[1], "send") == 0)
  {
    if (strcmp(argv[2], "pcm") == 0)
    {
      Packet * pkt = allocpkt();
      hdr_active * hdr = (hdr_active*)pkt- > access(off_active_);
      hdr- > type = 0;
      send_function_pcm(hdr, PCM);
      send(pkt, 0);
      + +total_trans_pack_no;
    }
    return (TCL_OK);
  }
  return (Agent::command(argc, argv));
}
```

The program calls up a related voice-packing scheme (send-function), which prepares voice data, and sends the packet to the next node.

Preparing a voice packet

The send-function prepares data from an input voice stream and packs them in the data field of the packet structure.

```
void send_function(hdr_active * shdr, char treat_type)
{
  FILE *in, *out;

  shdr- > seq_no = send_seq_no;
  + +send_seq_no;
  shdr- > send_time = Scheduler::instance().clock();
  shdr- > treat_type = treat_type;

  if ((in = fopen("in.wav", "rb")) == NULL)
  {
    Tcl& tcl = Tcl::instance();
    tcl.eval("puts "Cannot open input file "");
    exit(0);
  }

  if (file_seq_no == 0)
  {
    if ((out = fopen("out.wav", "wb")) == NULL)
    {
      Tcl& tcl = Tcl::instance();
      tcl.eval("puts "Cannot open out.wav file"");
      exit(0);
    }
```

```
        fread(in_charb, FILE_OFFSET, 1, in);
        fwrite(&in_charb, FILE_OFFSET, 1, out);
        fclose(out);
    }

    fseek(in, FILE_OFFSET + (file_seq_no * MAX_SEG_SIZE), SEEK_SET);
    shdr− > data_length = fread(shdr− > data, 1, MAX_SEG_SIZE, in);
    shdr− > data_type = DATA_PACKET;

    if(!shdr− > data_length)
        shdr− > data_type = END_OF_DATA_PACKET;

    fclose(in);
    + +file_seq_no;
}
```

It fills all relevant fields of the packet structure such as packet-sequence-number, send-time, treat-type, data-length and data-type. At this stage, the ActiveAgent sends the packet to its destination.

Active node

ActiveAgent is a procedure, which can receive a packet and check its fields. At first, it checks the treat-type field of the packet and calls an appropriate function to handle the packet (such as *an-pcm-function*). This is the main function that checks the sequence number of the packets. In case of detecting a lost packet, this function generates the lost packet using its adjacent packets. This done, the ActiveAgent sends the new packet to its destination. The code for receiving packets, detecting the lost packets and regenerating them is implemented in NS-2 using the following programs:

```
void ActiveAgent :: recv(Packet ∗ pkt, Handler∗)
{
    hdr_ip ∗ hdrip = hdr_ip :: access(pkt);
    hdr_active ∗ hdr = (hdr_active∗)pkt > access(off_active_);
    if (hdr > type == 0)
    {
        if (hdr > treat_type == PCM)
        {
            Packet ∗ pktret = allocpkt();
            hdr_active ∗ hdrret = (hdr_active∗)pktret− > access(off_active_);
            hdrret− > type = 1;

            an_pcm_function(hdr, hdrret);
            Packet :: free(pkt);
            send(pktret, 0);
        }
    }
    else
    {
        recv_function(hdr);
        sprintf(out, "%s dest_recv %d %3.1f", name(), hdr->seq_no,
          (Scheduler :: instance().clock() - hdr->send_time) * 1000);
        Packet :: free(pkt);
    }
}
```

Receiving a voice packet

The recv-function receives and saves a packet, analyzes it and detects the lost packets. In addition, the function calls up other functions to regenerate the lost packets. The reconstructed voice is sent to the output stage.

```
void recv_function(hdr_active * rhdr)
{
  if (rhdr->treat_type == PCM)
    pcm_packet(rhdr);
}
```

Statistical information

The following code generates statistical information about the lost packets:

```
unsigned char diff = abs(ihdr->seq_no - an_nodeseq_no);

if (diff > 1)
{
  ++an_nof_dropped_packets;
  for (i = (an_nodeseq_no + 1) ; i < ihdr->seq_no ; i++)
    tcl.evalf("puts"AN : Packet No. %d is lost. It took %d ms. to be regenerated."", i, time2 - time1);

  switch (diff)
  {
    case 2 : ++an_one_consequitive_lost;  break;
    case 3 : ++an_two_consequitive_lost;  break;
    case 4 : ++an_three_consequitive_lost;  break;
    case 5 : ++an_four_consequitive_lost;  break;
    default : ++an_more_consequitive_lost;  break;
  }
}
an_nodeseq_no = ihdr->seq_no;
```

To sum up, when a packet reaches the active node, the node checks the packet header and if the packet is an active one, it is sent to a specific program. If the packet is an ordinary one, its address field is checked and it is sent to its destination. Active packets are classified based on their treat-type field. Each voice-coding scheme has its own node program. The node checks the packet to find the number and location of the lost packets (it can be done by testing the packet's sequence number). If no lost packet is detected, the node saves a copy of the packet data field in its structure and sends the packet to the next node. If a lost packet is detected, the node tries to regenerate it using the saved data from the previously arrived packets. The regenerated packet is marked (with a tag in the data-type field) and is sent to the next node.

4.12.3 Simulation topology

The topology of the network is pre-determined and defined in an Otcl file, which is summarized here.

```
 set ns [new Simulator]

#Create three active nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]

#Create three ordinary nodes
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]
set n7 [$ns node]
set n8 [$ns node]
set n9 [$ns node]

#Connect the nodes with two links
$ns duplex − link $n0 $n3 1Mb 10ms DropTail
$ns duplex − link $n3 $n9 $linkspeed 10ms SFQ
$ns duplex − link $n9 $n1 1Mb 10ms DropTail
$ns duplex − link $n1 $n4 1Mb 10ms DropTail
$ns duplex − link $n4 $n2 1Mb 10ms DropTail
$ns duplex − link $n5 $n3 1Mb 10ms DropTail
$ns duplex − link $n6 $n3 1Mb 10ms DropTail
$ns duplex − link $n4 $n7 1Mb 10ms DropTail
$ns duplex − link $n4 $n8 1Mb 10ms DropTail

set p0 [new Agent/Active]
$ns attach − agent n0 p0
set p1 [new Agent/Active]
$ns attach − agent n1 p1
set p2 [new Agent/Active]
$ns attach − agent n2 p2
$ns connect p0 p1
$ns connect p1 p2

for {set i .5} {$i <= 170} {set i [expr $i + 0.05]} {
$ns at $i ”$p0 send pcm” }
$ns at 10 ”finish”
$ns run
```

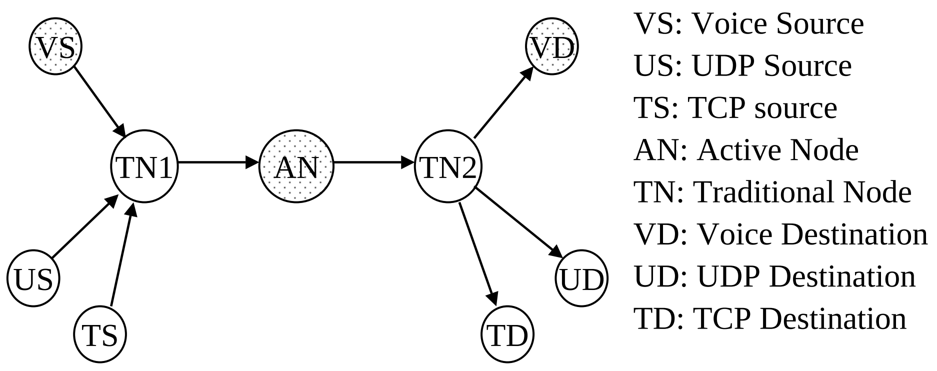

Figure 4.18: Simulation topology for tests.

Figure 4.18 shows the topology for the simulation.

Using a voice-packing method, node Voice Source (VS) packs a voice packet and sends it to VD, which acts as a Voice Destination node. The first Traditional Node (TN1) collects packets and sends them to the Active Node (AN). AN has the lost packet recovery program. Detecting a lost packet, it regenerates the packet using its internal execution environment. AN sends the received and regenerated packets to TN2. Since a practical test needs background traffic, nodes TS and US are assigned to generate the traffic. As the most dominant protocols in the Internet are TCP and UDP [103], node US generates UDP packets and sends them to node UD and node TS generates TCP packets and sends them to node TD. The UDP packets have 400 bytes of payload. They are transmitted at 1 ms intervals. The TCP packets have 400 bytes length and a 32-packet size window.

All links have 1 Mbit/sec speed except the link between node TN1 and AN, which has variable speed. All propagation delays are set to 10 ms. As a bottleneck in the topology, a lossy link is established between node TN1 and AN. The variable link speed between TN1 and AN, and the limited number of packets in queue for TN1 mean the active node and the receiver experience various drop rates. Away from the bottleneck, nodes TN1 and TN2, because of the way they are designed, drop packets based on the Gilbert model. Each of them receives two parameters as p and q from the input configuration file which sets its drop strategy based on those parameters (refer to Subsection 2.2.3). The packet drop distribution is adjusted to what is measured from the practical tests described in Chapter 2.

Simulation parameters

In addition to the other simulation parameters, which are mentioned in the previous subsections, the input *.tcl* file uses the following instructions to set the simulation parameters.

```
set  m            4500   ;; Set the M parameter for different schmes
set  active_node  ON     ;; Turn the active node On and OFF
set  prob_p1      0.5    ;; p1 for Gilbert model for TN1
set  prob_q1      0.1    ;; q1 for Gilbert model for TN1
set  prob_p2      0.5    ;; p2 for Gilbert model for TN2
set  prob_q2      0.1    ;; q2 for Gilbert model for TN2
set  link_speed   1Mb    ;; Set the bottleneck link speed
```

The above parameters are altered to set the active node operative and non-operative and change the other simulation parameters to get different loss rates and patterns.

4.13 Summary and conclusion

In this Chapter, active and passive networks are described and the role of the active networks in lost packet detection and regeneration is discussed. Mathematical analyses prove that by reducing the number of consecutive lost packets, an active node improves the voice quality at the receiver. Therefore, active networks are used to regenerate the lost packets. Simple calculation shows that the best place to allocate an active node is where the packet loss rate from the sender to that location is equal to the loss rate from that location to the receiver. In addition in this Chapter, a practical way to implement active nodes in the current network is proposed and an active node structure is explained. Finally, codes for implementing an active node in a network simulator environment, followed by the simulation topology, is introduced.

Chapter 5

Detached Samples Packing Method (DSPM)

This Chapter presents a voice-packing scheme based on speech properties for VOIP. This technique, Detached Samples Packing Method (DSPM) introduces an algorithm to encode and pack voice samples at the sender as well as decode and regenerate voice at the receiver. Its main undertaking is to compress the speech samples by finding the similar voiced segments in speech. It can be implemented in active networks as pre-stored programs that can help active nodes do ”on-the-fly” regeneration of lost packets. It works on packing speech based on its specifications and on the drop behaviour of the Internet. The following sections describe DSPM based on its sender, receiver and active node routines followed by some of its test results.

5.1 DSPM sender routine

DSPM sender routine consists of a packing and a main routine.

5.1.1 Packing Routine

Suppose PCM voice samples (8000 samples/s and 8 bits/sample) are transmitted from source to destination through a network, which has active nodes in its structure. The routine gets voice samples and divides them to make blocks. Each block is formed by 400 samples as

$$X_0, X_1, \cdots, X_{399}.$$

The Packing routine separates even-numbered samples from odd-numbered ones and frames them into two groups [23]. Consequently, each frame will have half of the block samples as

$$X_0, X_2, \cdots, X_{398}$$

and

$$X_1, X_3, \cdots, X_{399}.$$

If each frame is packed in a packet, it is possible that some packets will be lost due to the lossy environment in the network. It is possible to reconstruct the lost packets from the other available packets in the receiver or in the active node in the network. If the odd samples are lost, each lost byte can be estimated by a simple linear interpolation, for example

$$X_1 = (X_0 + X_2)/2$$

and

$$X_3 = (X_2 + X_4)/2.$$

Using this method, it is possible to regenerate the lost packet in an active node and insert it into the network. The quality of sound will not be very good because it is not possible to have good samples with the simple averaging method. For example, in Figure 5.1, it is obvious that X_{i+1} cannot be accurately regenerated from X_i and X_{i+2}. The averaging method error will be very high and the quality of sound will be poor. To cope with the problem, Wah and his colleagues tried to design a linear transformation to change voice samples before interleaving [107, 105]. They used a matrix where regenerated voice quality depends on dimension. On the one hand, a big matrix generates good voice quality when a packet is lost

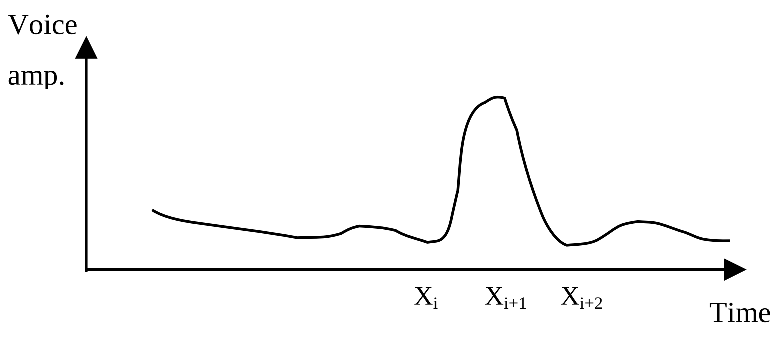

Figure 5.1: Example of a voice segment.

but on the other hand, when a big matrix is used in a loss-less environment, it generates poor voice quality compared to the original voice.

In contrast to the approach of [107, 105], DSPM tries to pack packets in such a way that the regenerated voice quality does not degrade in loss-less networks. To do this, the DSPM algorithm finds and marks *important samples* of the original voice and transmits them in both odd and even packets.

In Figure 5.1, the protocol has to transmit X_{i-2}, X_i, X_{i+1}, X_{i+2}, X_{i+4}, ... samples in the even packet and X_{i-3}, X_{i-1}, X_{i+1}, X_{i+3}, ... samples in the odd packet, so the important sample (X_{i+1}) will be transmitted in both packets. The protocol would mark the important sample in the even packets and let the receiver know that it is not an ordinary byte. This means, in case of losing the odd packet, the reconstruction routine would pay attention to that byte.

If the following condition is true for X_i,

$$|\frac{X_{i-1} + X_{i+1}}{2} - X_i| \ > \ M \tag{5.1}$$

then X_i will be marked as an important byte, where M is a predefined threshold. Using a large M, packets will have short length but the result of regeneration will not be very accurate. On the other hand, using small M, the packets will have long length but accurate reconstructed samples will be generated. Moreover, the M parameter will determine the amount of redundant data that is not required if both odd and even frames are received at the destination. In short, M defines the amount of overhead that DSPM adds to the voice samples. Figure 5.2 shows the overhead percentage versus M parameter in 4500 bytes of female voice samples where X_i varies between 0 and 255. It can be seen that the overhead percentage where M equals 50 is less than 4%. The previous studies have shown that the drop rate is not closely related to the packet length, so inserting 4% of overhead to the voice stream in the Internet will not have a severe negative effect in the whole transmission environment. Besides, this amount of overhead helps the active nodes (or the receiver) regenerate the lost

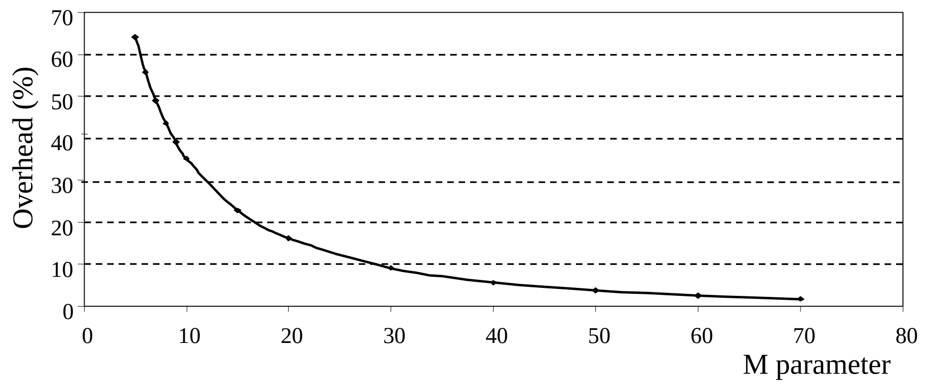

Figure 5.2: DSPM overhead versus M.

Table 5.1: Voice segment occurrence

Segment	No. of occurrence
Unvoiced	248
Un2voiced	34
Voiced	484
V2unvoiced	34

packets more efficiently.

5.1.2 Main Routine

This routine gets sound samples and processes them in several steps. In the first step, the Main routine divides sound samples into segments of 80 bytes in length (10 ms). The energy of each segment is calculated and is stored in an array by

$$E^2 = \sum_{n=n_1}^{n_2} |x(n)|^2 = \sum_{n=1}^{80} |x(n)|^2. \tag{5.2}$$

A classifier routine classifies segments based on their energy. Segments with lower energy than a predefined *voice-unvoiced-threshold* are classified as *Unvoiced* and the others are marked as *Voiced*. The transitions between these two classes are marked as *Voiced2unvoiced* and *Unvoiced2voiced* segments. Table 5.1 shows the number of each segment in 8 seconds of female voice samples.

In the next step, the consecutive segments of the same classification are joined together to make blocks. As both transitions only happen in one segment, they have 80 byte, fixed-sized blocks. DSPM is allowed to make blocks no longer than five segments (Maximum 40 ms. blocks). Then the Main routine selects a block and encodes it based on its class.

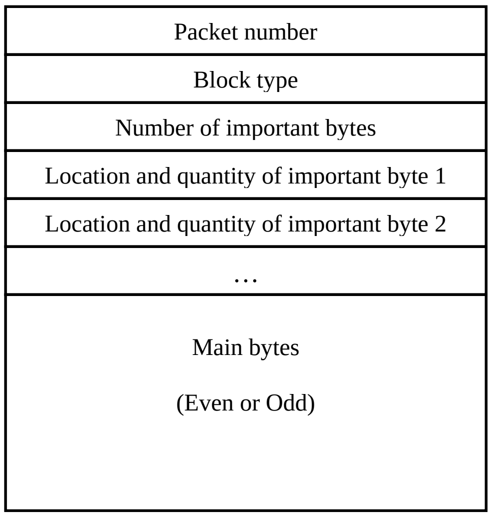
Packet number
Block type
Number of important bytes
Location and quantity of important byte 1
Location and quantity of important byte 2
…
Main bytes
(Even or Odd)

Figure 5.3: DSPM Unvoiced2voiced-block packet structure.

- Unvoiced block: For an unvoiced block, a very short packet is generated containing the size of the block and its energy. At the receiver, a random noise generator can simulate this block if its noise power is adapted to the power of the original unvoiced block. That packet is inserted in the network twice, because of its similarity with the packets of other blocks.

- Unvoiced2voiced block: As this block has remarkable impact on the voice quality, it must be packed in such a way that it can be regenerated properly if a packet loss occurs during the transmission. So DSPM calls the packing routine up with low M in order to have a higher redundant data. The packet structure for this segment is shown in Figure 5.3.

- Voiced2unvoiced block: This block is not very important and losing bytes in this part of sound does not have a serious negative impact on the sound quality at the receiver. So DSPM calls the packing routine up with high M to have low redundant data. This segment has the same packet structure as shown in Figure 5.3.

- Voiced block: This block can be packed the same as transition blocks but a detailed survey of the speech shows a periodic waveform in some parts of the voice segment [40]. Figure 5.4 shows the voiced segment when ”one” is pronounced by the author and it has a 9 ms period. The period depends on the speaker’s pitch, and men and women have different periods [87]. There are few differences between adjacent parts (a, b and c in Figure 5.4) and each part can be easily estimated from its neighbors. Therefore, DSPM searches to find the period of that segment and marks it in the packet in order to compress these parts and not send unnecessary data.

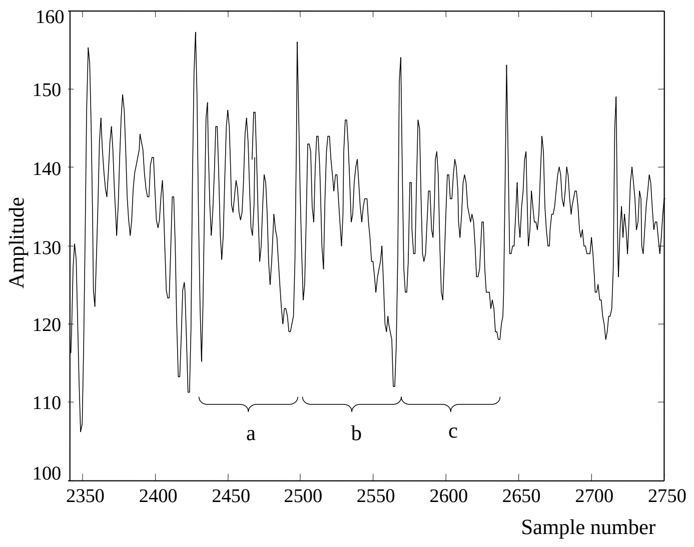

Figure 5.4: Similar parts in a voiced segment.

There are techniques to find the periodic part of a voice segment. Low pass filtering [26], man-interacting pitch detection procedure [66, 81] and pitch detection using a zero-crossing interval sequence [40] are some examples of those algorithms. To find the periodic part of a voice segment, DSPM calculates the autocorrelation of the voice block [80]. The autocorrelation is calculated by

$$Y(n) = \sum_{n=-\infty}^{\infty} x(n)x(n-k). \tag{5.3}$$

The interval between the two first adjacent peaks is considered the period of the voice. Figure 5.5 shows a voiced sample of female speech and Figure 5.6 shows its autocorrelation.

It can be seen that the interval between the two first adjacent peaks in Figure 5.6 is 44 which is equal to the period of voice in Figure 5.5. If there is no clear periodic part in the autocorrelation, the first segment (80 bytes) of the block is truncated. The truncated segment is considered to be packed as an independent voiced block. The rest of the block is sent to be reprocessed by autocorrelation. Figure 5.7 shows samples of a non-periodic voice and Figure 5.8 shows its autocorrelation.

Having obvious periodic peaks, the cross correlation between that period and the rest of the block is checked to find similarity within the block. The cross correlation is obtained by

$$Y(n) = \sum_{n=-\infty}^{\infty} x(n)p(n-k). \tag{5.4}$$

To do this, the amplitude of the first cross correlation peak is divided by the amplitude of each peak. Similarity is defined as a ratio higher than a predefined *Similarity-threshold*. The higher the Similarity-threshold, the lower the achieved compression rate, therefore the higher

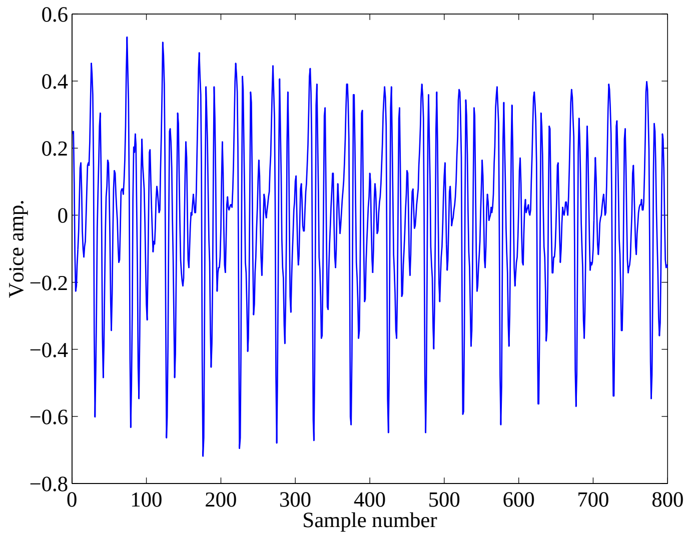

Figure 5.5: A periodic voice segment.

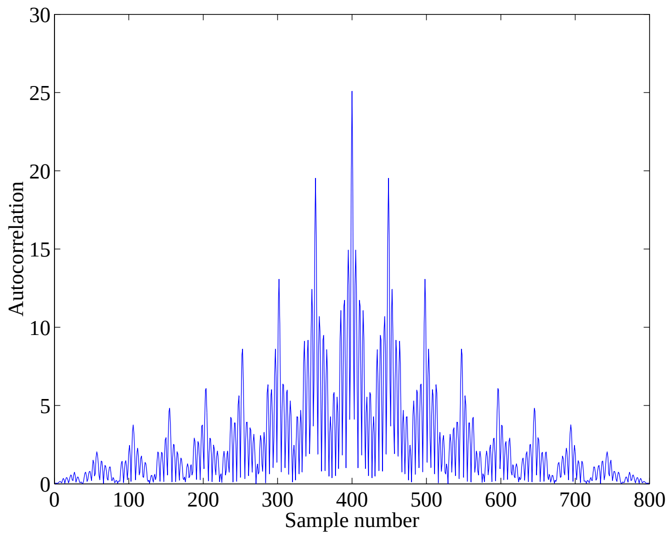

Figure 5.6: Autocorrelation of the periodic voice segment.

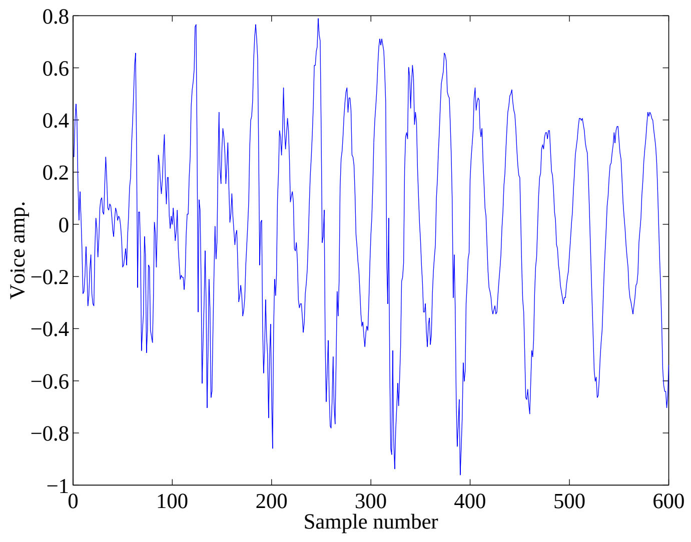

Figure 5.7: A non-periodic voice segment.

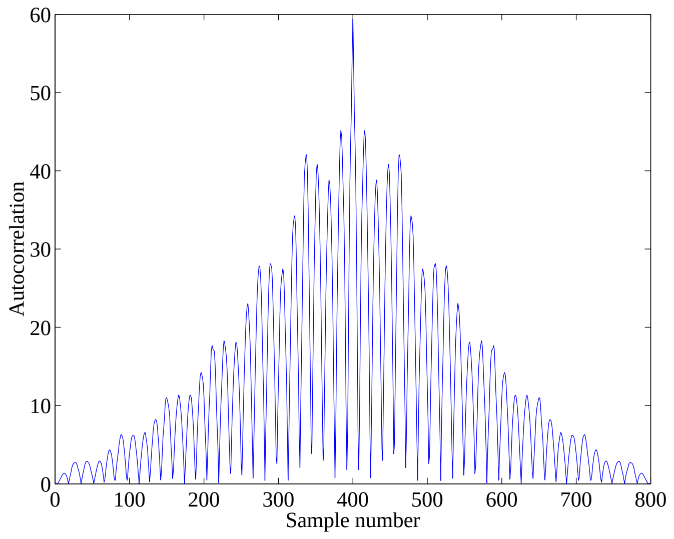

Figure 5.8: Autocorrelation of the non-periodic voice segment.

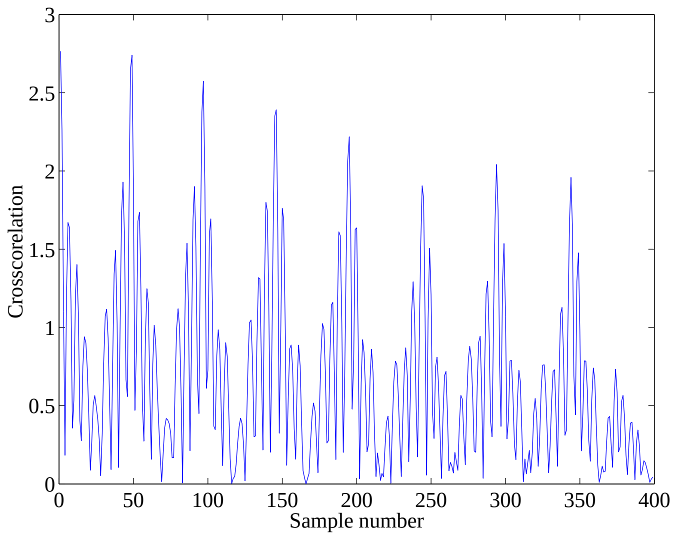

Figure 5.9: Cross correlation of the periodic voice segment.

the number of bytes required to transmit and the higher the voice quality at the receiver. Figure 5.9 demonstrates the cross correlation of the voice in Figure 5.5.

Having identical parts, samples of the first period and the number of its repetitions are sent to the Packing routine. These samples accompanied by medium level redundant data (M) form packets. If no similarity is detected in the cross correlation, the first period of the block is truncated and packed as an independent voiced block, and the rest of the block is sent to the autocorrelation portion of this routine. In a practical test, a 300-byte voiced block was compressed into a 76-byte packet. Figures 5.10 and 5.11 show the flow chart of the sender routine and the packet structure of DSPM respectively.

5.2 Receiver or active node routine

As each block is separated into two consecutive packets with odd and even samples, if a packet is lost in the network, the active node (or the receiver) will have another part of that block for recovery. If the packet is an unvoiced one and it is alone (no other adjacent unvoiced packets are available), the active node can generate a copy of it and insert the copy in the network. Receiving a single packet in other parts of the sound, the active node can easily regenerate the lost packet using interpolation between adjacent samples of the received packet. At the end of interpolation, the important bytes from the packet will be inserted into their proper place.

If the active node runs the regenerating and packing routine for all received single packets and inserts the reconstructed packets into the network, the receiver will have more received

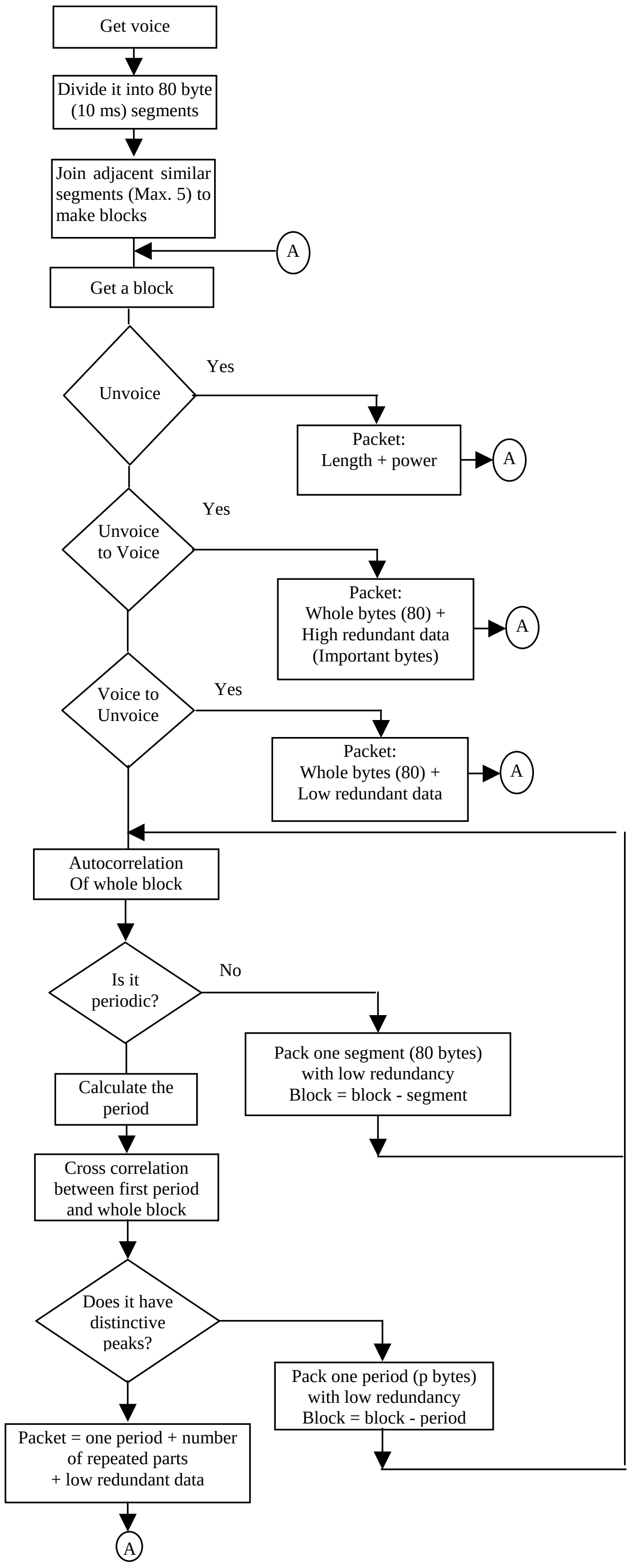
Get voice
Divide it into 80 byte (10 ms) segments
Join adjacent similar segments (Max. 5) to make blocks
A
Get a block
Unvoice
Yes
Packet: Length + power
A
Unvoice to Voice
Yes
Packet: Whole bytes (80) + High redundant data (Important bytes)
A
Voice to Unvoice
Yes
Packet: Whole bytes (80) + Low redundant data
A
Autocorrelation Of whole block
Is it periodic?
No
Pack one segment (80 bytes) with low redundancy Block = block - segment
Calculate the period
Cross correlation between first period and whole block
Does it have distinctive peaks?
Pack one period (p bytes) with low redundancy Block = block - period
Packet = one period + number of repeated parts + low redundant data
A

Figure 5.10: Flowchart of DSPM packing routine.

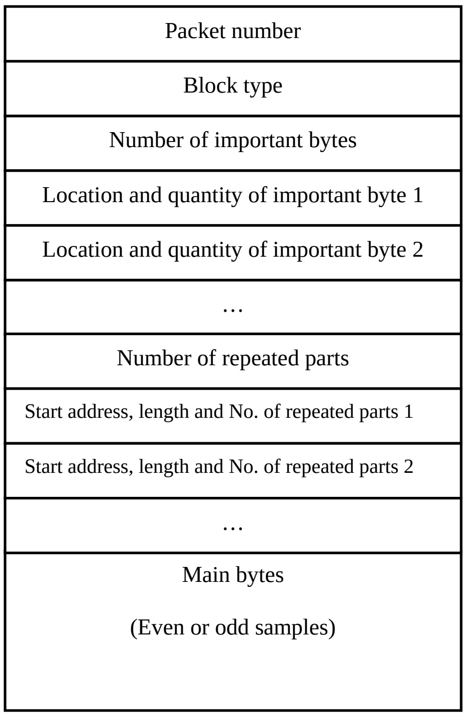
Packet number
Block type
Number of important bytes
Location and quantity of important byte 1
Location and quantity of important byte 2
…
Number of repeated parts
Start address, length and No. of repeated parts 1
Start address, length and No. of repeated parts 2
…
Main bytes
(Even or odd samples)

Figure 5.11: DSPM packet structure.

packets to regenerate the original sound. It has been observed that in a periodic waveform, the amplitude changes gradually between adjacent periods. At the receiver, a program was written to compensate its effect using the change rate. By testing it was found that this routine had no noticeable impact on voice quality and it has been deleted from all programs.

5.3 Protocol specification and verification

In order to specify and verify the DSPM active node routine and test its correctness, two models were chosen and the routine tested in both of them.

5.3.1 Finite State Machine (FSM) model

In this model, the active node rests in a *state* waiting for an *event* to happen [102] [101]. Each state consists of all the values of its variables. The active node has two states. It awaits either an even-packet or an odd-packet. The node receives three packets from the sender; even, odd and an unknown packet. Figure 5.12 represents a state of FSM.

The FSM of the DSPM active node routine is indicated in Figure 5.13.

There are seven transitions listed in Table 5.2 for this FSM.

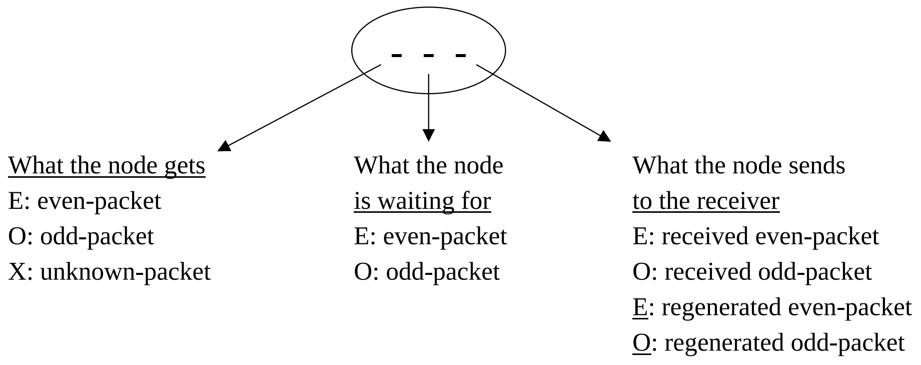

Figure 5.12: FSM states.

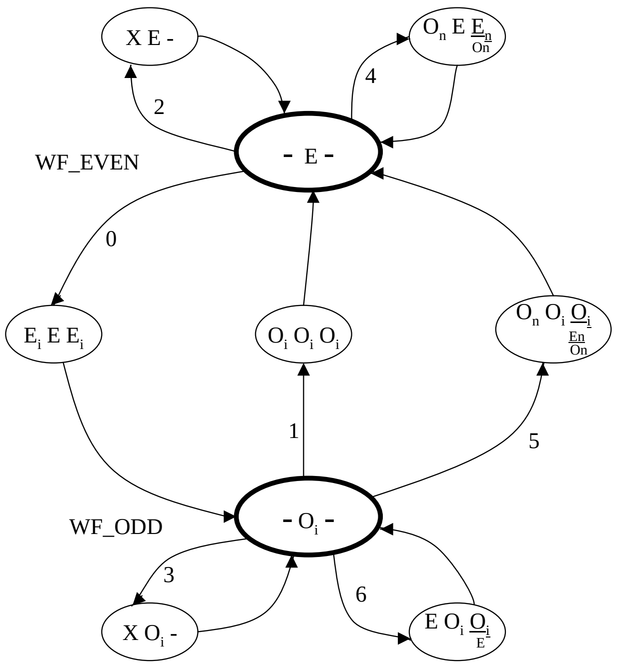

Figure 5.13: FSM of DSPM active node routine.

Table 5.2: FSM transition list for DSPM.

Transition	Packet accepted	Packet regenerated	Packet sent
0	E	-	E
1	O	-	O
2	-	-	-
3	-	-	-
4	On	$\underline{\text{En}}$	$\underline{\text{En}}$, On
5	On	$\underline{\text{Oi}}$, $\underline{\text{En}}$	$\underline{\text{Oi}}$, $\underline{\text{En}}$, On
6	E	$\underline{\text{Oi}}$	$\underline{\text{Oi}}$, E

As the protocol has only two states, it is not necessary to do the reachability analysis to determine if the protocol is correct [101].

The DSPM active node routine has a fairly simple protocol with two states. In case of more complicated protocols (such as DADPCM5), the FSM method will present many complicated states and transitions that are very difficult to analyze. Therefore, this thesis has examined SDL, which is explained in Subsection 4.6.6.

5.3.2 Specification and Description Language (SDL) model

SDL presents the protocol based on what is drawn in Figure 4.11. Again the DSPM node routine has two states.

- WF_E_i in which it awaits for an even-packet numbered i.
- WF_O_i in which an odd-packet numbered i is anticipated.

E_n and O_n represent the numbered packets other than i arriving at the active node. Figure 5.14 shows the SDL for that routine.

As can be seen, this method presents a much easier way to verify that the protocol does not have a deadlock or end less loop in its structure.

5.4 Results of DSPM

The voice quality of DSPM is evaluated in terms of its data rate and its robustness against the packet loss problem.

Quality versus data rate

In a test, the Similarity-threshold was selected as 0.8 and 4500 bytes of female voice samples were packed into 2916 bytes. The lossy network simulator discarded 50% of all packets and the reconstruction routine had to regenerate sound with its 1473 received bytes. The upper waveform in Figure 5.15 shows the original voice and the lower waveform represents the regenerated voice using 1473 received bytes at the receiver.

As is mentioned in Subsection 5.2, the amplitude changed gradually during adjacent periods. The *check* point in the original voice indicates the change in amplitude during consecutive periods. The regenerated voice has equal amplitude for the periods (*check* point in the lower Figure). The regenerated voice from 32% of the original voice has acceptable quality judged by the audience (Appendix C).

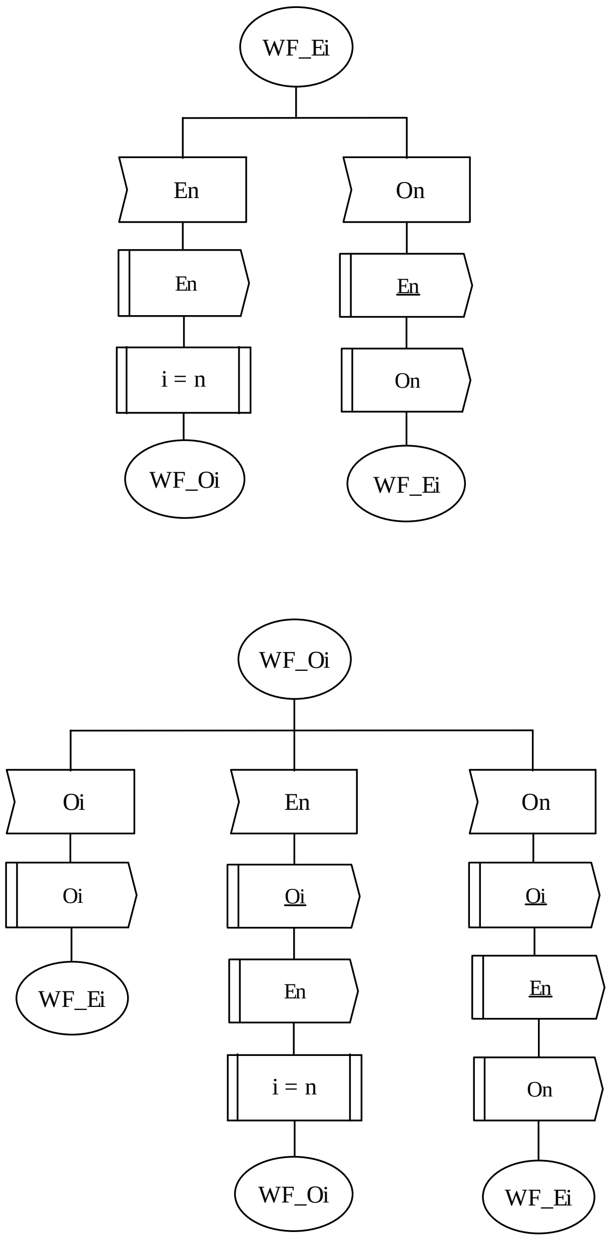

Figure 5.14: SDL of DSPM active node routine.

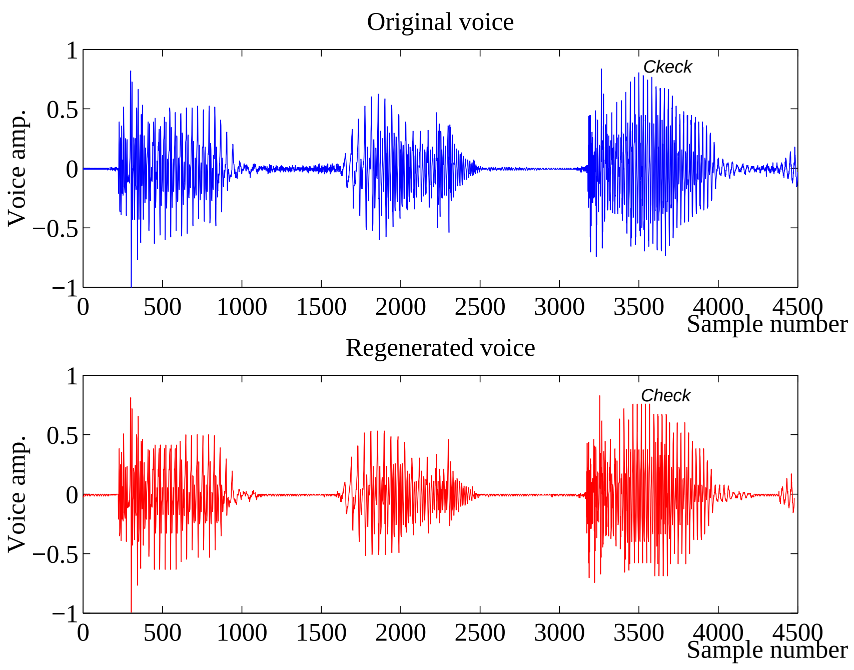

Figure 5.15: Original and regenerated voice samples.

In another test, a very lossy network which loses 50% of packets was considered. The audience was asked to mark the regenerated voice based on its quality, from zero for unacceptable to ten for very good. Figure 5.16 shows the relative quality of the voice versus its overhead percentage.

As would be expected, the lower the Similarity-threshold, the lower the transmitted data rate. Figure 5.17 shows the relation between the transmitted data versus the Similarity-threshold for 64000 byte of a female voice.

It can bee seen that the DSPM sends 49784 bytes out of 64000 bytes even when the threshold is one. That is because the DSPM compresses the unvoiced parts of the speech.

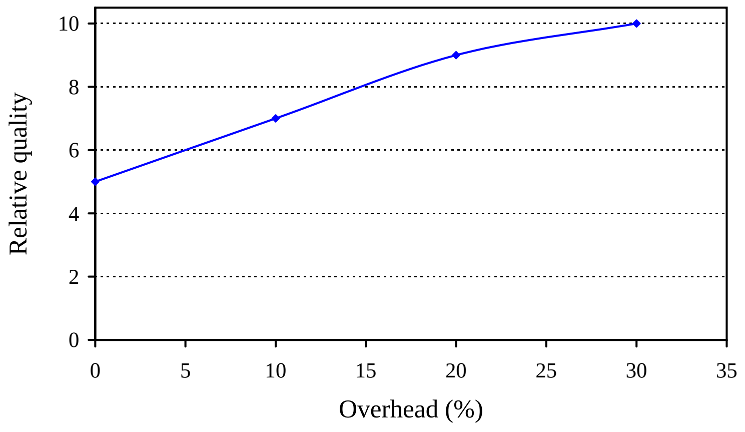

Figure 5.16: Quality versus overhead with 50% lost packets.

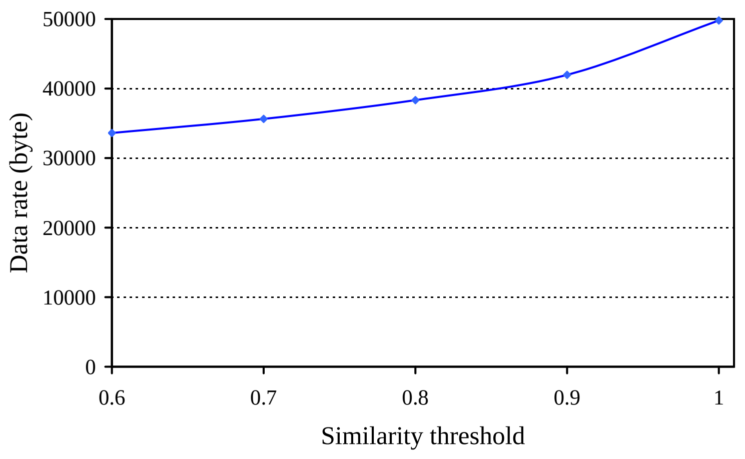

Figure 5.17: DSPM Data rate versus the similarity-threshold.

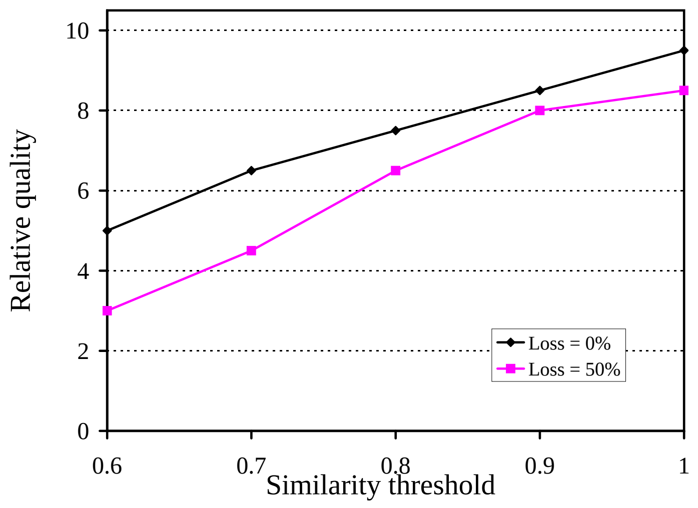

Figure 5.18: DSPM relative quality versus the Similarity-threshold.

Figure 5.18 represents the relation between the regenerated voice quality and the Similarity-threshold when the network passes all packets as opposed to when it loses 50% of the packets.

Quality versus loss rate

In this test, five schemes were tested to compare their voice quality after passing through a lossy network.

- Silence substitution: Inserts silence instead of lost packets.
- Packet repetition: Repeats the last arrived packet in place of the lost packet.
- FEC: This scheme uses redundant low bit rate (6.4 kbps) LPC codes ”piggy-backed”

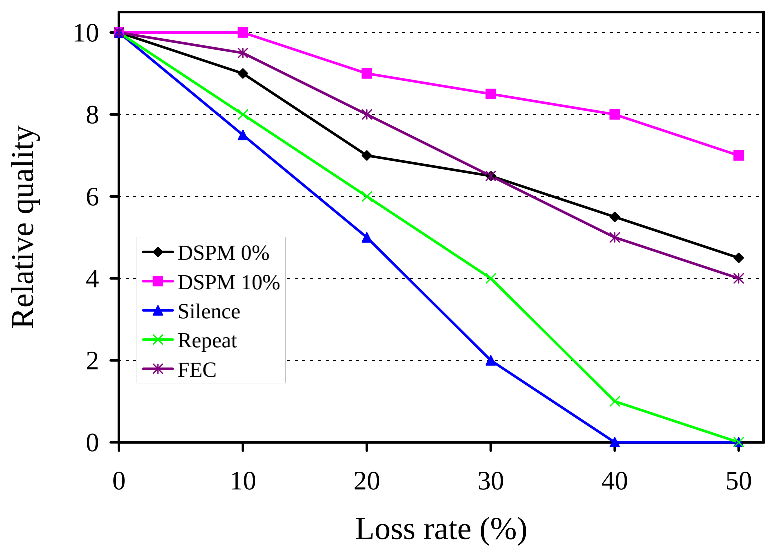

Figure 5.19: Comparison of voice quality of different schemes.

on the high quality 64 kbps main PCM codes.

- Separated odd-even samples: Separates a voiced-block into two packets with odd and even numbered samples.
- Detached Sample Packing Method: The explained method with M equal to 50 for the voiced part and Similarity-threshold equal to 0.8.

In all tests, a native English-speaking woman was selected and her 64 kbyte voice was packed at the sender. The audience was asked to judge the regenerated voice based on its quality (Appendix C). The tests were done with different loss rates. Figure 5.19 shows the result.

The result implies that the silence substitution scheme cannot cope with more than 10% loss rate. The other four schemes have almost similar quality in low loss rate but DSPM has better quality at higher loss rates. Although the M parameter was selected to be 50 and the overhead was less than 4%, DSPM has significantly higher quality in a very lossy network. Even though FEC offers better quality compared with DSPM's 0% (without redundant data), it must be recalled that FEC has 6.4 kbps of redundant data and also it increases CPU overhead.

5.5 Summary and conclusion

DSPM as a voice-packing scheme can compress the voice samples and pack voice packets based on the voice specifications. It divides voice samples into four segments and inserts redundant data based on the specification of each segment. Therefore, its packet length varies from packet to packet. It adds a low amount of redundant data to packets, which allows the scheme to tolerate lost packets. It compresses the unvoiced and voiced parts of speech to enable a lower bit rate. Results show that the voice quality of DSPM depends on its Similarity-threshold and likewise, its data rate depends on that threshold. Comparing different schemes, it is revealed that DSPM offers good voice quality even when the network suffers from heavy losses. Considering the quick packet regeneration algorithm of DSPM compared with the FEC algorithm, DSPM can be classified as a suitable voice packing method for very lossy networks.

Chapter 6

Detached Adaptive Differential Pulse Code Modulation (DADPCM)

This Chapter introduces a voice-packing scheme based on the main idea of DSPM. The idea of inserting important bytes in different packets is implemented by the ADPCM algorithm in this scheme. This method is suitable when compressed voice samples must be transmitted from source to destination through a network which has active nodes in its structure. Although the packing scheme can be implemented in a traditional network, active nodes can support the scheme which offers improved voice quality during higher packet loss rates. The DADPCM algorithm compresses, encodes and packs voice samples at the sender and regenerates those voice samples at the receiver. To overcome the packet loss problem, the algorithm inserts important bytes to the coded data in order to regenerate accurate voice samples at the receiver. Besides that, this Chapter introduces new methods for choosing the important samples. The following subsections describe DADPCM based on its sender, receiver and active node routines followed by an efficient method to mark the important parameters and the tests results.

6.1 DADPCM sender routine

This routine separates voice samples into blocks and executes the following steps on each block.

6.1.1 First step: Separate samples

For the sake of simplicity, this step is explained without compression (which is the main concept of ADPCM). Suppose a block is formed by

$$X_0, X_1, \cdots, X_{399}$$

samples. As was mentioned in DSPM routine, the Sender routine separates even-numbered samples from odd-numbered ones and puts each in separate streams [53, 52]. Consequently, each stream will have half of the block samples as

$$X_0, X_2, \cdots, X_{398}$$

and

$$X_1, X_3, \cdots, X_{399}.$$

If each stream is packed and sent to the receiver via the Internet, some streams might be lost because the Internet does not guarantee a reliable transmission environment. By simple linear interpolation, DADPCM would be able to reconstruct the lost packets from the other available packets in the receiver or in the active node in the network. Although this method regenerates the lost samples, the quality of the regenerated sound will not be very good because of the averaging method error.

As was mentioned in the DPCM method, the algorithm would find and mark important samples of the original voice and transmit them in both odd and even packets. The protocol would mark the important samples in both packets and let the receiver know that they are not ordinary bytes. This routine uses the same strategy as is mentioned in Subsection 5.1.1 to choose important samples. The M parameter plays the same role in packet length and voice quality as is mentioned for DSPM. M defines the amount of overhead that the DADPCM adds to the voice samples. Figure 6.1 shows the packet overhead percentage versus M parameter in 64000 bytes of female voice samples when X_i varies between 0 and 65535. It can be seen that the overhead percentage for M which is equal to 4200, is less than 10%. The performed tests have shown that the packet drop rate is not closely related to the packet length, so inserting 10% of overhead to a compressed voice stream on the Internet will not have a severe negative effect on the whole transmission environment. Furthermore, this amount of overhead helps the active nodes regenerate the lost packets efficiently.

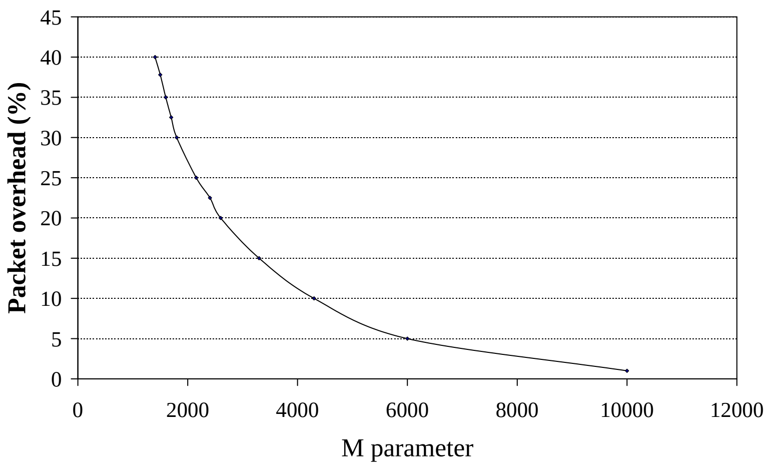

Figure 6.1: DADPCM packet overhead versus M.

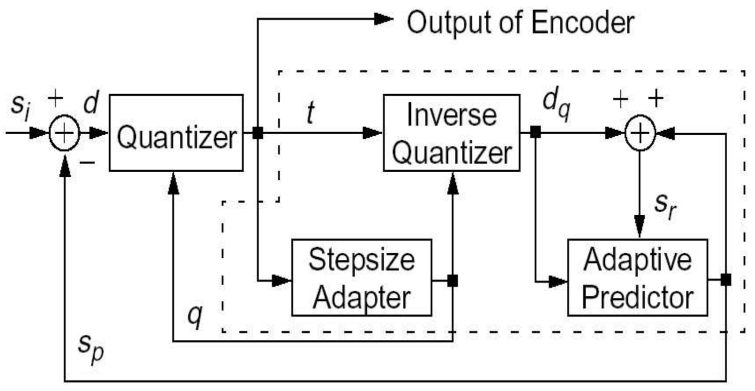

Figure 6.2: DADPCM encoder and decoder block diagram.

6.1.2 Second step: Encoding and decoding samples

This step receives a stream from the first step and applies the ADPCM compression algorithm to it. The ADPCM algorithm takes advantage of the high correlation between consecutive speech samples. The selected ADPCM 32 kbps algorithm is based on the Interactive Multimedia Association (IMA), ITU-T recommendation G.721. The algorithm encodes the difference between a predicted sample and the speech sample, so it provides efficient compression with a reduction in the number of bits per sample [83]. It converts each eight-bit voice sample to a four-bit ADPCM code. Figure 6.2 shows a block diagram for the ADPCM compression routine.

The 16-bit two's complement, input sample, Si, is subtracted by the predicted sample Sp, and the difference is applied to an adaptive quantizer, which generates a four-bit encoded, output code. In order to predict the next sample, a full decoder is implemented in the encoder structure. The decoder uses four bit codes to update the inverse quantizer. The decoder needs its previous predicted-sample and quantizer step-size-index in each iteration. To summarize, the encoder generates ADPCM codes that are not only based on the current input sample but also based on the previous samples. This is an important point, which must be considered in the packing scheme otherwise each received packet following a lost packet might generate wrong samples. To prevent this effect, DADPCM inserts three bytes as decoding coefficients in all packets. Consequently, each packet can be decoded individually.

6.1.3 Third step: packing the encoded samples

This step receives coded samples from the second step, inserts decoder coefficients and also inserts the location of the important bytes, which is necessary to regenerate the voice at the receiver. The important bytes are inserted in both even and odd packets so where there are both packets, the receiver will have those samples twice. The receiver needs to know the important bytes' location in order to prevent them occurring twice at the output. As each important byte is coded in four bits, it forms half of a byte of a packet. Therefore, the sender must mark each important nibble. Three different schemes for indicating the location of important codes are considered. They are compared in terms of their total amount of overhead. In the following text, letter I stands for the number of important codes and letter A stands for the number of main codes.

- Define a byte containing the address of each byte in the packet with an important code and assign one bit to indicate the low or high nibble of the location. The overhead is $I/2$ byte for codes, I byte for pointers and $I/8$ for nibble indicators given by

$$Overhead = I + I/2 + I/8 = 13I/8.$$

- Assign a bit to each nibble in the whole stream (main or important codes) to point the important codes. The overhead is $I/2$ byte for codes plus $(A + I)/8$ byte for pointers given by

$$Overhead = I/2 + A/8 + I/8 = (A + 5I)/8.$$

- Assign a bit to each byte in the whole stream to indicate whether it has an important code or not and assign a byte to each important code. The overhead is I byte for codes plus $A/16$ byte for pointers to main codes plus $I/8$ byte for pointers to the important codes given by

$$Overhead = I + A/16 + I/8 = (A + 18I)/16.$$

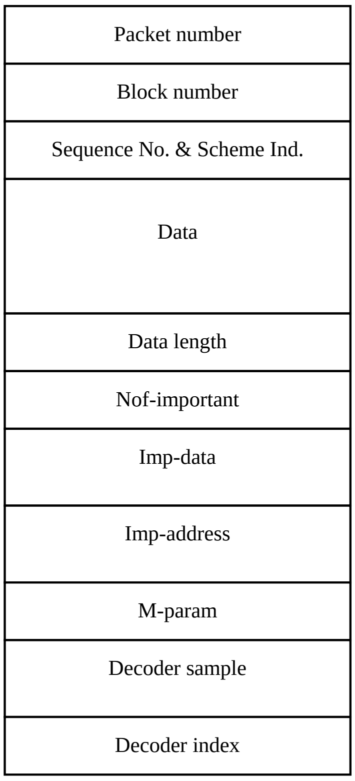

Figure 6.3: DADPCM packet structure.

Comparing these schemes, it is identified that if $I > A/8$, the second scheme has the lowest overhead and if $I < A/8$, the first scheme will introduce the lowest overhead. In order to have minimum overhead, DADPCM dynamically changes its indicating scheme based on the number of important and main codes. The sender measures the ratio of main and important code numbers and chooses an indicating scheme and sends the scheme's number to the receiver via each packet.

The structure of a packet in DADPCM is shown in Figure 6.3.

The *Packet Number* is a counter that holds the number of transmitted packets. The *Block Number* counts the 400-byte size block numbers. The *Sequence number* is an even or odd packet indicator and one bit of it indicates the scheme that points the important numbers. The *Data* field has the transmitted data in ADPCM format and its length is held by the *Data-length*. The important data, their length and their addresses are respectively carried by *imp-data*, *nof-imp* and *imp-address*. The M parameter with witch the packet was generated is in the *m-param*. The three-decoder bytes that provide the feasibility of decoding a packet independently of the other packets are in the *decoder-sample* (two bytes) and *decoder-index* (one byte).

Flow chart 6.4 describes the packing routine of DADPCM.

The whole sender routine of DADPCM is coded in Appendix B.

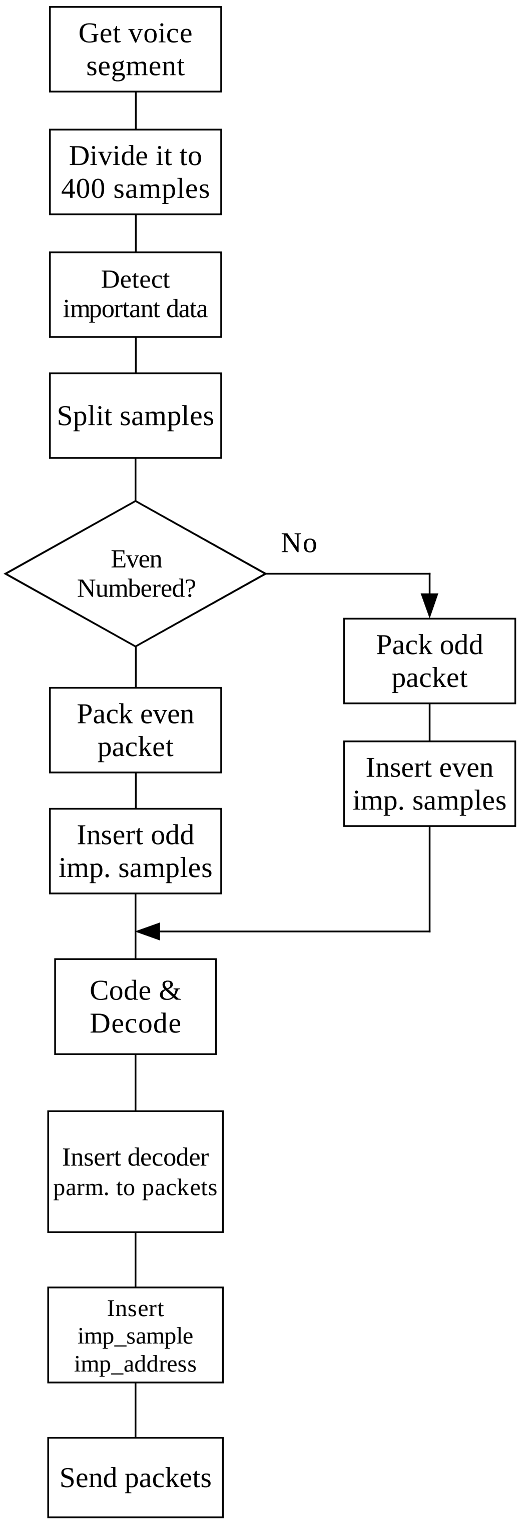

Figure 6.4: Flow chart of DADPCM packing routine.

6.2 Receiver routine

This routine receives arrived packets and tries to regenerate the original voice with one of its two strategies. Using the packet sequence number, the first strategy works based on receiving both even and odd packets and the second strategy works based on detecting a lost packet.

- First strategy: Having both even and odd packets, the receiver decodes them separately. Each packet can be decoded individually because it has its own decoder coefficients. Moreover, each packet contains its scheme number for indicating the important codes. Decoding even and odd packets, the receiver merges the samples into one voice stream. The receiver also deletes the important codes using their addresses. It generates the main voice samples and then the routine sends them to the output audio stage.

- Second strategy: Losing one packet in the network, the receiver decodes the arrived packet to gain voice samples. The samples are divided to main and important codes. Suppose that the odd packet is lost. The receiver gains the even samples as well as some important odd samples in the even packets. The receiver interpolates all the even samples to get estimated, odd samples and inserts the received odd, important samples among them. At the end of this process the voice stream, which contains arrived, even samples, estimated odd samples and arrived, important odd samples is sent to the output.

Chart 6.5 shows the simplified SDL's for the receiver routine.

In order to have analog voice, the output samples must convert to an analog signal and pass through a filter to eliminate the aliasing problem. As the DADPCM algorithm does not change the total sampling rate, it does not need another filter for the aliasing effect.

6.3 Active node routine

Each active node receives a packet, saves it and transmits it toward the other nodes or to the receiver. Detecting a lost packet, the node uses the same routine as the receiver to reconstruct voice samples. Then the node executes the same routine as the sender to regenerate the lost packet. Eventually the regenerated packet is sent to the other nodes or to the receiver.

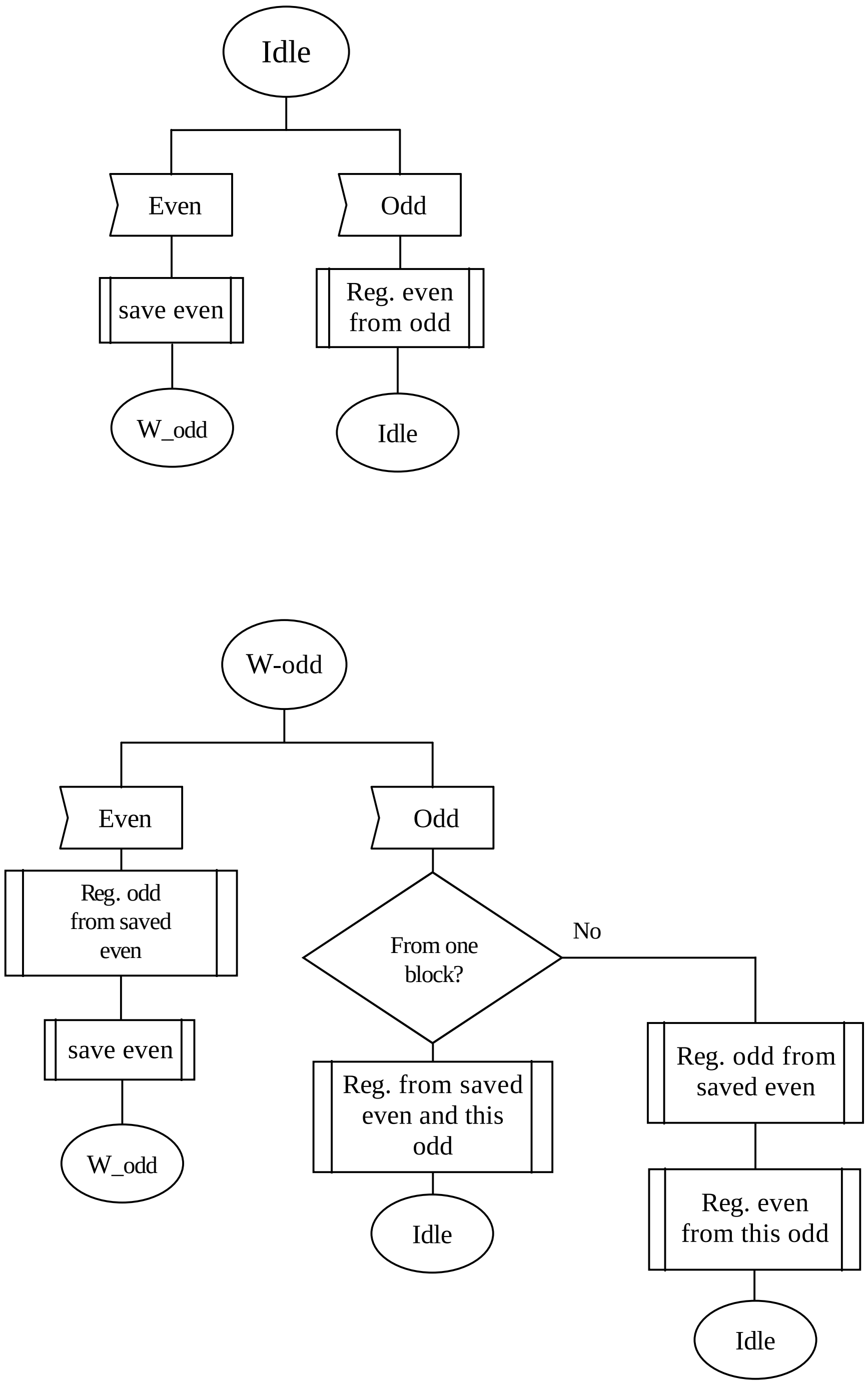

Figure 6.5: SDL's of DADPCM receiver routine.

6.4 Results of DADPCM

In a series of tests, a native English-speaking woman was selected and her 64000-byte voice samples were packed at the sender. The lossy link discarded some packets and the active node had to regenerate the lost packets. Using NS-2, five schemes were tested to compare their voice quality after passing through the lossy link.

- Silence substitutions: This inserts silence instead of lost packets.
- Packet repetitions: This repeats the last arrived packet in place of the lost packet.
- FEC: This adds LPC codes of each ADPCM compressed packet to the next packet.
- Adaptive Differential Pulse Code Modulation (ADPCM): This scheme compresses each eight-bit voice sample to a four-bit code. It uses several previous samples to generate a new sample so when a sample (or in this case, a packet) is lost, ADPCM accumulates the effect of the lost sample in numerous later samples. As a result, ADPCM itself cannot deal with the packet loss problem at all. The tests showed that the voice quality is not good even with 5% lost packets. In another test the decoder coefficients were added to all packets to let the receiver decode each packet individually. As a result, the regenerated voice had the same quality as the silence-substitution scheme, so it is not shown in Figure 6.6.
- Detached ADPCM: The explained method with different M.

The performed tests aimed to evaluate DADPCM in terms of its voice quality, its data rate and its required processing time. The results are explained in the following subsections.

6.4.1 Voice quality

The audience was asked to judge the regenerated voice based on its quality (Appendix C). The tests were done with different loss rates. Figure 6.6 shows the result.

The result implies that the silence substitution and packet repetition schemes cannot cope with more than 10% loss rate and their related voice quality decreases drastically with upper loss rates. The DADPCM scheme without any important bytes (0% overhead) has better quality even with a very high drop rate. FEC has almost the same voice quality as DADPCM without important bytes, but it increases CPU overhead for decoding its LPC part. The more overhead it has, the better quality it achieves. The chart shows that with 24% overhead, DADPCM has quite good quality even when it loses half of its packets. The quality of

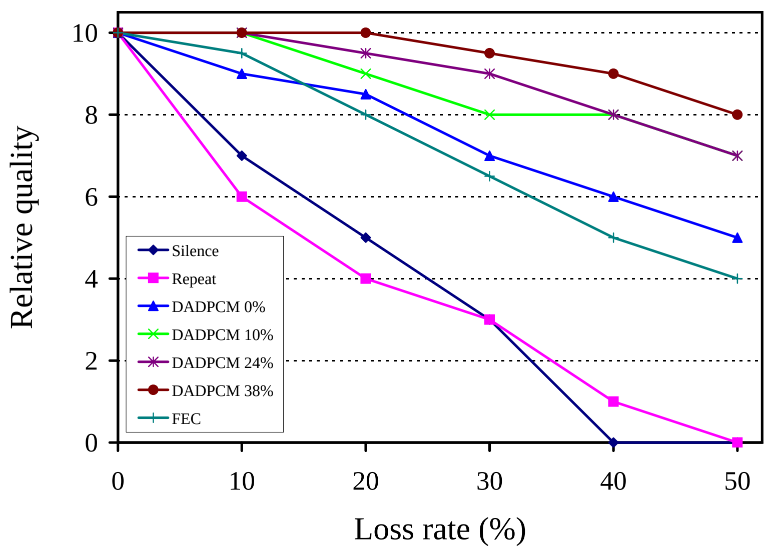

Figure 6.6: Comparison of voice quality of different schemes.

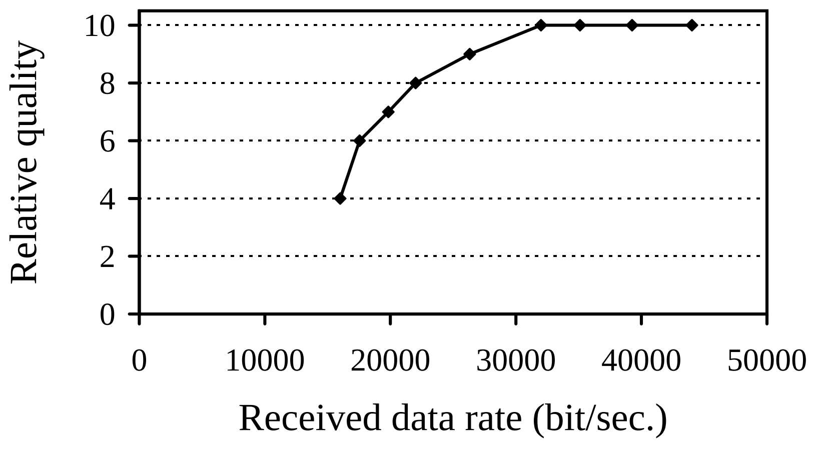

Figure 6.7: DADPCM voice quality versus data rate.

regenerated voice with DADPCM is double compared to other schemes, even with 10% overhead under 20% loss rate.

6.4.2 Data rate

DADPCM has 32000-bit/sec data rate when it has no important bytes. Bit rate increases when the number of important bytes goes up. In lossy networks, the more lost packets in the network, the less received packets at the active node or at the receiver. So when the number of lost packets increases, the number of received bytes decreases and consequently, the total bit rate decreases. Figure 6.7 shows the average quality versus received data rate at the receiver.

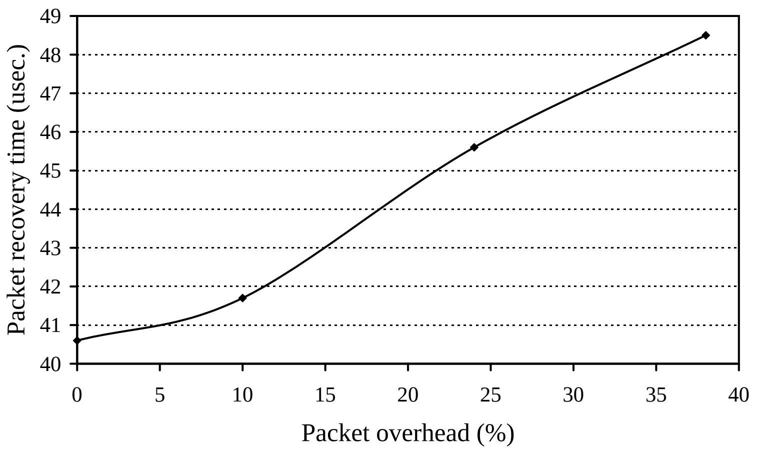

Figure 6.8: DADPCM processing time versus overhead.

It shows that even with 20-kbit/sec data rate, the received voice has good quality (eight out of ten score).

6.4.3 Processing time

In DADPCM, the total amount of processing that must be done by end users, as well as the processing time that must be reserved in active nodes should be considered. In order to evaluate the DADPCM routine in a practical test, the active node was implemented in a computer with Pentium IV-2.4 GHz as processor and 512-MByte RAM. In the worst case, the lossy network appeared to drop all odd packets (50% drop rate) and the active node had to regenerate them from its received even packets. The average processing time that is needed to recover each lost packet was measured. The packet recovery time varies with M parameter and chart 6.8 shows the recovery time versus packet overhead on the above-mentioned computer.

It can be seen that increasing packet overhead from 0 to 38% (with decreasing M) increases recovery time from 40.6 to 48.5 μsec.

Chart 6.9 shows the increased percentage of voice quality and recovery time versus the packet overhead (M).

It implies that with 20% packet overhead, the recovery time increases 10% and the voice quality increases up to 25%.

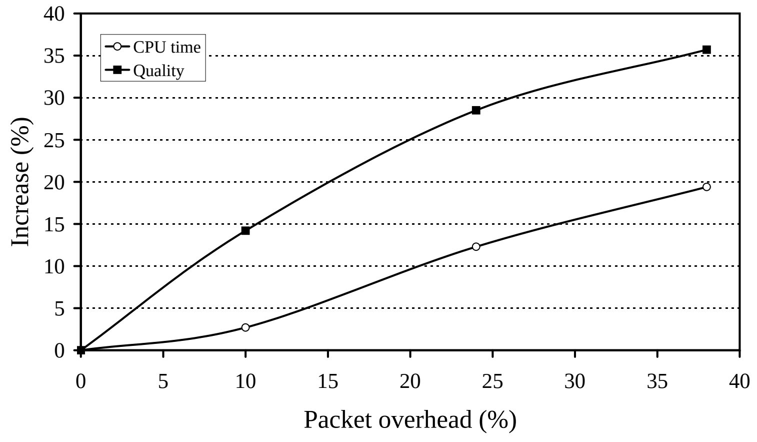

Figure 6.9: DADPCM quality and processing time versus overhead.

6.5 Important samples

Inserting important samples from one packet to another packet helps voice coding schemes regenerate lost packets. In lossy networks, the more important samples a packet has, the better voice quality it generates. Unfortunately, inserting these redundant data makes long packets. Therefore, optimum amounts of important samples are desired to use in all packing schemes. DSPM and DADPCM use (5.1) to choose important samples. These schemes use RTCP based messages to adjust the M parameter to keep the voice quality high and the redundant data low. This Section introduces four other methods for choosing important samples.

6.5.1 Fixed-sized packets

It has been observed that using (5.1), the packet length varies with the M parameters. This method proposes adjusting M to meet a predefined amount of redundant data. To do this, a simple program that changes M and measures the overhead, is written. Comparing measured overhead by the predefined overhead in a loop, the program increases or decreases M to achieve the desired amount of redundant data. In view of the fact that, this method increases CPU overhead to adjust the M parameter, it is not studied in detail.

6.5.2 Fixed distortion

Suppose that voice samples are divided into segments of length N, when N is an even number. $x(n)$ as a segment is separated into two streams with even-numbered and odd-numbered samples. Using (5.1), some important samples are calculated and inserted into each stream.

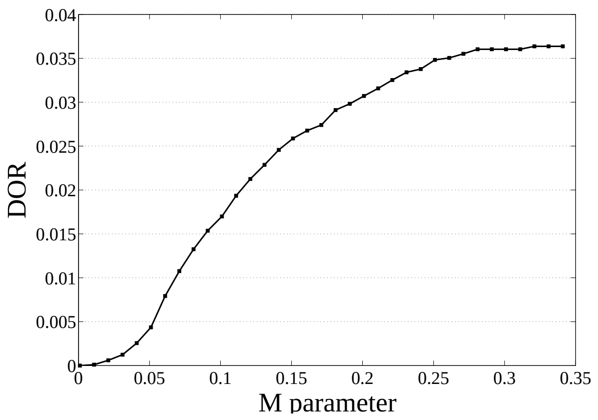

Figure 6.10: DOR versus M.

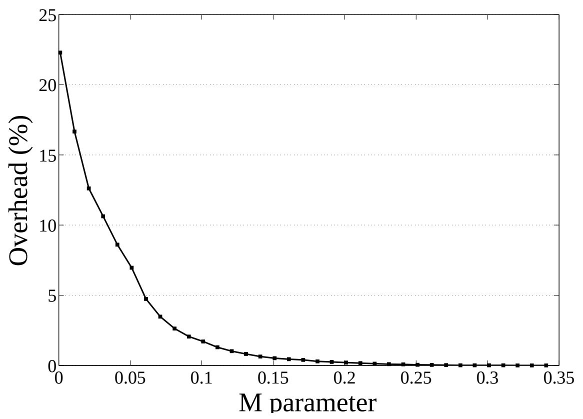

Figure 6.11: Overhead versus M.

As a worst case scenario, consider that all odd samples are lost. The lost samples can be estimated by linear interpolation and they can be modified by important odd samples available in the even stream. Using these data, $\hat{x}(n)$ which contain the original even samples, estimated odd samples and original important odd samples are gained. A Distortion-to-Origin Ratio (DOR) is defined by

$$DOR = \frac{\sum_{i=0}^{N-1}(x(i) - \hat{x}(i))^2}{\sum_{i=0}^{N-1}(x(i))^2}. \tag{6.1}$$

Obviously, the lower the M parameter, the more important the samples from the odd series to the even series, and the lower the DOR.

Figure 6.10 shows the relation between DOR and the M parameter for 16000-byte of a voice sample. It is presented that with low M, DOR is very low, and when M increases, DOR increases until it reaches saturation point (0.036 in the Figure). The relation between the overhead and M is shown in Figure 6.11.

As expected, when M increases, the overhead decreases. Combining these two figures, the relation between DOR and the overhead is shown in Figure 6.12.

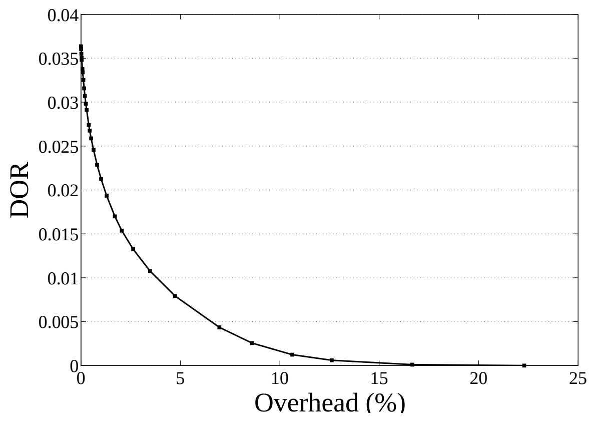

Figure 6.12: DOR versus overhead.

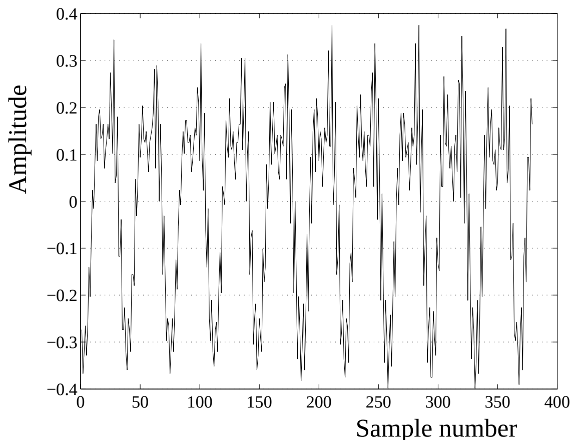

Figure 6.13: A voiced segment contains high frequency.

Obviously, the higher the overhead, the lower the DOR. In Figure 6.10, this method emphases the determining of a predefined DOR, and finding its relevant M. Different voiced segments are tested and it is realized that the curve of DOR versus M changes by the nature of the voiced segment. Voiced segments shown in Figure 6.13 and its DOR versus M in Figure 6.14, and also voiced segments shown in Figure 6.15 and its DOR versus M in Figure 6.16 are two examples of the voiced segments of a speech and their related DOR curve.

As expected, for a constant M (namely, 0.2), DOR in the first voiced segment is higher than DOR in the second segment (0.12 compared to 0.04). Due to the higher frequency components in Figure 6.13 compared to Figure 6.15, it is clear that the first voiced segment should have higher DOR than the second one. Therefore, using a high-pass filter, the power of high frequency components of a voiced segment is measured. The calculated power is divided by the power of the voiced segment to gain a Relative High Frequency Power (RHFP).

It is observed that the higher the $RHFP$ in a voiced segment, the higher the DOR in

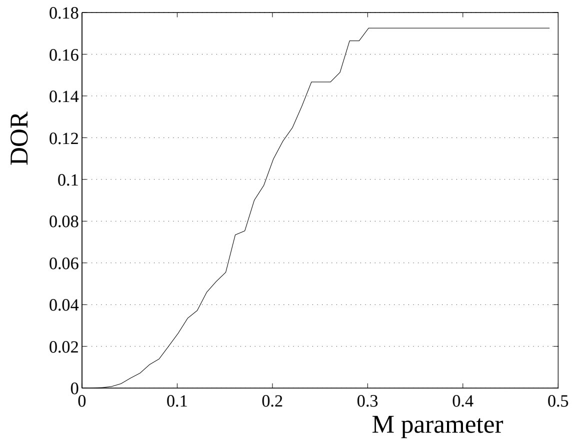

Figure 6.14: DOR of a voice containing high frequency.

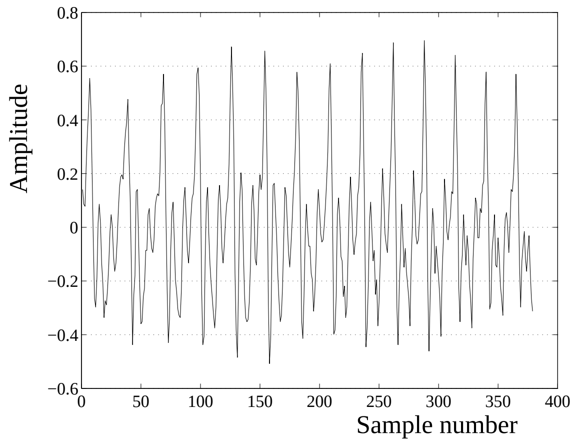

Figure 6.15: A voiced segment contains low frequency.

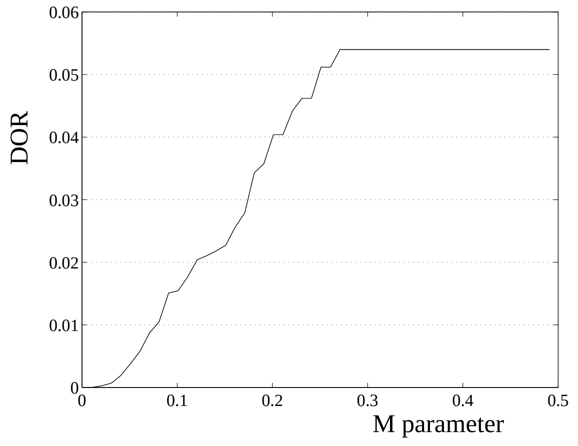

Figure 6.16: DOR of a voice containing low frequency.

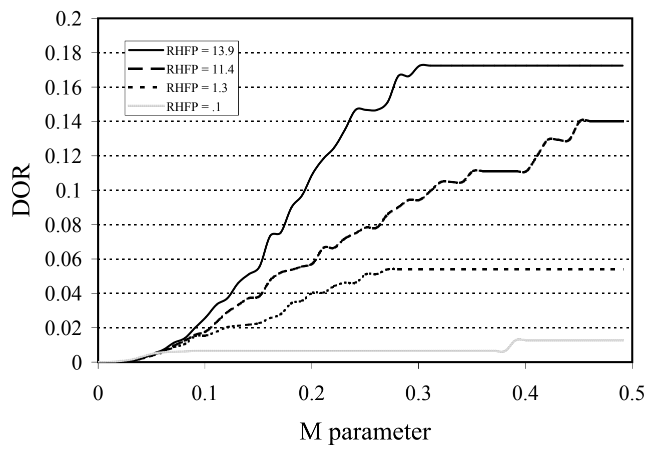

Figure 6.17: DOR versus M for four classes.

a specific M. Consequently, it is possible to classify the voiced segments based on their $RHFP$ and determine M from a DOR_M table assigned to each class. As a result, the following steps provide a procedure to find an M for a predefined DOR.

- Determine the signal power.
- Apply a high pass filter to the voiced segment and calculate the power of the passed signal.
- calculate $RHFP$ by

$$RHFP = \frac{power\ of\ high\ frequency\ components}{signal\ power}. \tag{6.2}$$

- Find a class and related DOR_M table for the class.
- Determine the M parameter by looking up the table.

Based on the measurements of 16000 bytes of female speech samples, voiced segments are divided into four classes with the following $RHFP$ specifications

$$\begin{aligned} class\,1 &: & RHFP &< 1, \\ class\,2 &: & 1 \le RHFP &< 5, \\ class\,3 &: & 5 \le RHFP &< 10, \\ class\,4 &: & 10 \le RHFP&. \end{aligned}$$

Figure 6.17 shows DOR versus M for those classes.

It is proposed to use four tables based on the four curves of Figure 6.17 to determine an M parameter for a given DOR. In contrast with the first method, this procedure does not need a great deal of processing time due to the nature of the look up table method.

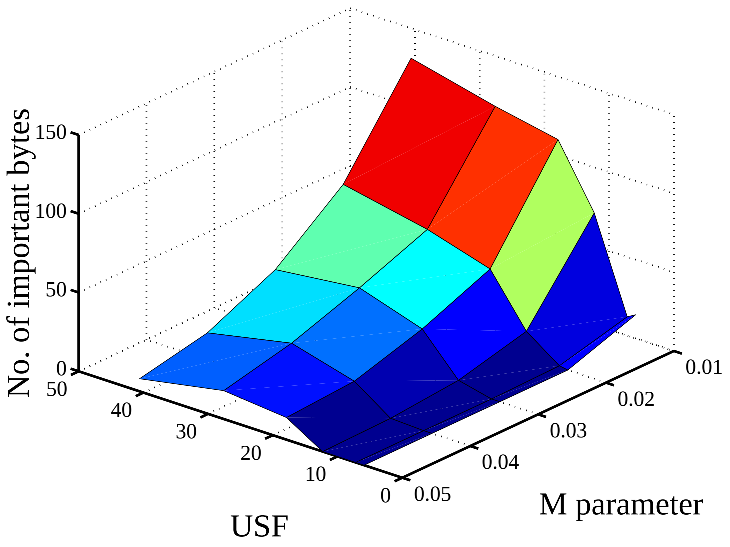

Figure 6.18: Number of important bytes versus USF and M.

6.5.3 Next correlation

This method is based on calculating the covariance between a voice stream started from the k^{th} sample and another stream started from the $k+1^{th}$ sample of a voice segment. Similarly to the previous method, voice samples are divided into segments of length N, when N is an even number. Samples starting from k and ending at $k+N-1$ are extracted and stored in $x(n)$, and samples starting from $k+1$ and ending at $k+N$ are extracted and stored in $y(n)$. Therefore, $x(n)$ and $y(n)$ are two equal sized streams of a voice segment which have only one sample difference in their starting points.

To be independent of the amplitude, both $x(n)$ and $y(n)$ are normalized by dividing them by their power. Using the covariance of $x(n)$ and $y(n)$ and the variance of $x(n)$, an *UnSeemly-Factor* (USF) is defined by

$$USF = 100(1 - \frac{cov(x,y)}{var(x)}). \tag{6.3}$$

Obviously, when the difference between two consecutive samples are low (the next sample can be estimated by its previous one), the covariance is high and USF is low. Conversely, when the next sample is totally different to its previous sample, the covariance is low and USF is high. Therefore, USF as a criterion of similarity between even and odd samples of a voice segment can help schemes determine the M parameters for the (5.1) formula. Figure 6.18 shows the relation between USF and the number of important bytes in different equal-sized voice segments with different M (M is normalized by the power of $x(n)$).

It can be seen that high USF has the same impact on the number of important bytes as the low M. Therefore, it is possible to estimate M based on the USF value of a voice segment.

6.5.4 Modified fixed distortion

This method sets out to choose an M parameter to meet a predefined distortion when a packet is lost. As in Subsection 6.5.2, suppose that voice samples are divided into segments of length N, when N is an even number. Each segment is classified as voiced or unvoiced based on its energy. $x(n)$ as a voiced segment is split into two streams with even-numbered and odd-numbered samples. As a worst case scenario, suppose that all odd samples are lost, then $y(n)$ is defined by

$$y(n) = \begin{cases} x(n) & n = 2k \\ \frac{x(n-1)\ +\ x(n+1)}{2} & n = 2k+1, \qquad where:\ 0 \leq k \leq \frac{N}{2} - 1. \end{cases} \tag{6.4}$$

To be independent of the amplitude, both $x(n)$ and $y(n)$ are normalized by dividing them by the power of $x(n)$. Let $w(n)$ denote an array of length N and defined by

$$w(n) = (x(n) - y(n))^2, \qquad 0 \leq n \leq N-1. \tag{6.5}$$

The $w(n)$ has $\frac{N}{2}$ of squares of the distance between $x(n)$ and $y(n)$, and $\frac{N}{2}$ zero elements. Then $w(n)$ is sorted in descending order. A simple program starts from the biggest element of $w(n)$ and attempts to eliminate the elements in such a way that the sum of the rest of the elements of $w(n)$ become less than the predefined distortion multiplied by the power of $x(n)$. The index of the last eliminated element helps the program find a pointer (ptr) to the related element in $x(n)$. The M parameter would be

$$M \ = \ |\frac{x(ptr-1) + x(ptr+1)}{2} - x(ptr)|. \tag{6.6}$$

Having the M parameter and (5.1), the program chooses the important samples from the current segment and inserts them in related packets to compensate the lost effects. Thus, this method tries to find the M parameter for each voiced segment of a speech, based on the distortion of the voice when half of the samples are lost. M varies from segment to segment based on the segment specification. Using the determined M for each segment, $z(n)$ is defined by

$$z(n) = \begin{cases} x(n) & n = 2k \\ y(n) & n = 2k+1 \quad if \quad |x(n) - y(n)| \leq M \\ x(n) & n = 2k+1 \quad if \quad |x(n) - y(n)| > M. \end{cases} \tag{6.7}$$

Normalized distortion before compensation is defined by dividing the power of $x(n) - y(n)$ by the power of $x(n)$ as

$$Dist\_before \ = \ \frac{\sum_{i=0}^{N-1}(x(i) \ - \ y(i))^2}{\sum_{i=0}^{N-1}(x(i))^2} \tag{6.8}$$

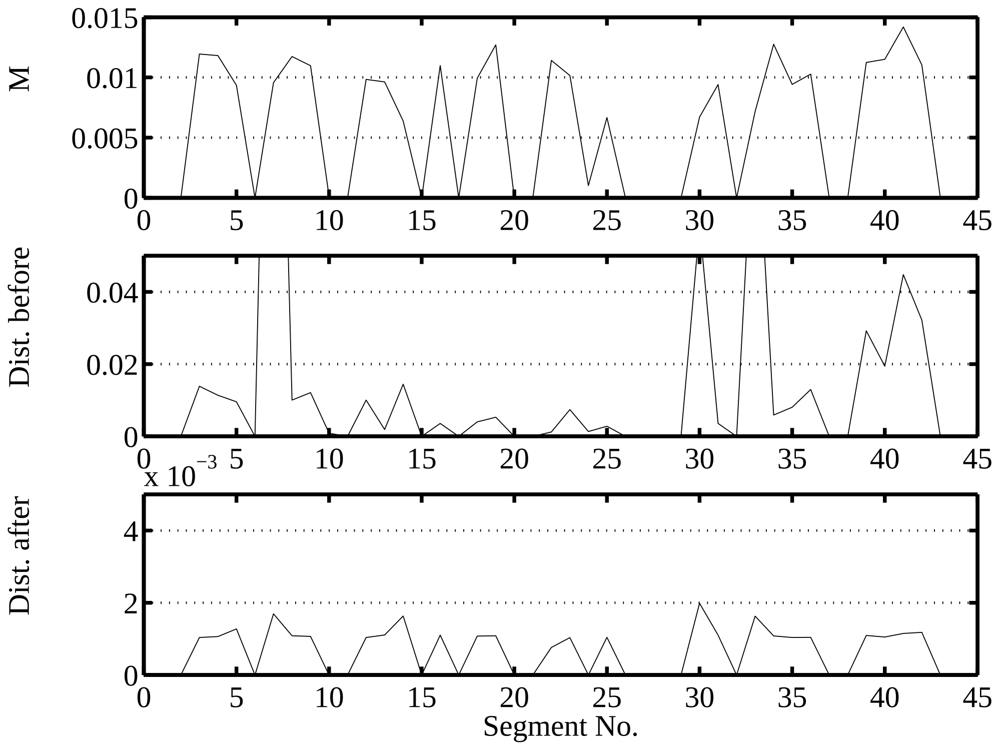

Figure 6.19: Relation among M, $Dist\_before$ and $Dist\_after$ in a 18 kbyte voice samples.

and normalized distortion after compensation is defined by

$$Dist\_after = \frac{\sum_{i=0}^{N-1}(x(i) - z(i))^2}{\sum_{i=0}^{N-1}(x(i))^2}. \tag{6.9}$$

Figure 6.19 shows M, before and after distortion compensation of 45 segments, 400 bytes in length, from 18000 bytes of female speech.

It can be seen that by varying M from segment to segment, the high amount of distortion before compensating changes to a low amount of distortion after compensating (considering that the distortion after compensating must be less than the predefined distortion which is 0.002 in the Figure). The program is implemented in Matlab on a Pentium IV - 2.4 GHz computer and it is revealed that each voiced segment needs 500 μsec of processing time to be completed.

6.6 Summary and conclusion

DADPCM is based on splitting voice samples into two streams and packing them in two packets. It extracts important samples from one stream and inserts them into another stream.

Consequently, its packet length varies from packet to packet. The tests demonstrated that DADPCM as a voice-packing scheme based on ADPCM principle shows markedly better voice quality compared with the other recovery schemes in VOIP even in a severely lossy network. Its data overhead and processing time has been tested and in spite of increasing these two parameters by inserting important samples, the scheme produces good voice quality despite high packet loss rate. The tests showed that the more important bytes it has, the better the voice quality achieved at the receiver. To find important samples four different methods are examined and in comparing their specifications, it is revealed that the Modified fixed distortion method offers a quick and accurate way to choose the important samples.

Comparing DADPCM processing time (Figure 6.8) with the processing time required by the FEC scheme (Subsection 10.6.3), it can be concluded that DADPCM is a suitable method for transferring voice samples in a very lossy network. The DADPCM algorithm is modified to DADPCM5 in order to tolerate burst loss which is explained in the next chapter.

Chapter 7

Detached Adaptive Differential Pulse Code Modulation 5 (DADPCM5)

This Chapter explains DADPCM5, a modified version of the DADPCM scheme, which copes with the burst packet loss problem in the network. The DADPCM5 algorithm converts the voice samples to codes, transmits those codes toward the receiver, and regenerates voice at the receiver. The main idea of DADPCM5 is the same as the basic idea of DADPCM but this scheme divides voice samples into five streams and packs them in five packets. Moreover, DADPCM5 improves the procedure of selecting important samples and applies more accurate important samples in the packets. The following subsections describe DADPCM5, based on its sender, receiver and active node routines, and gives some of its test results.

7.1 Sender routine

The DADPCM5 sender routine gets voice samples and divides them into 400-byte blocks taking the following steps on each block:

7.1.1 First step: Separate samples

For the sake of simplicity, this step is explained without compression, as it is the main idea behind ADPCM. Each block is formed by 400 samples as

$$X_0, X_1, \cdots, X_{399}.$$

The Sender routine separates every fifth samples, and puts each in a separate stream [23]. Consequently, there will be five streams, each with $\frac{1}{5}$ of block samples (80 byte length),

$$\begin{gathered} X_0, X_5, \cdots, X_{395} \\ X_1, X_6, \cdots, X_{396} \\ X_2, X_7, \cdots, X_{397} \\ X_3, X_8, \cdots, X_{398} \\ X_4, X_9, \cdots, X_{399}. \end{gathered}$$

If each stream is packed and sent to the receiver via the Internet, some streams might be lost. DADPCM5 would be able to reconstruct the lost packets from the other available packets in the receiver or in the active node in the network in the same way as in DADPCM routines. Each lost sample can be estimated by a simple linear interpolation given by

$$X_{i+1} = \frac{X_i + X_{i+2}}{2}. \tag{7.1}$$

Using this method, it is possible to regenerate the lost packet in an active node and insert it into the network. To have better voice quality, the algorithm would find and mark important samples of the original voice and transmit them in two consecutive packets. For instance, if X_{i+1} is detected as an important sample, the sender routine packs a packet with $X_{i-4}, X_{i+1}, X_{i+6}, \cdots$ and it packs another packet with $X_{i-3}, X_{i+1}, X_{i+2}, X_{i+7}, \cdots$. Therefore, the important sample (X_{i+1}) will be transmitted twice. The sender routine marks the important samples in the packets and lets the receiver know that it is not an ordinary byte.

DADPCM introduces CPU demanding methods to choose M. This coding scheme improves the simple routine introduced in DSPM to pick more accurate samples. In DSPM if $|\frac{X_{i-1}+X_{i+1}}{2} - X_i| > M$, then X_i is marked as an important byte (M is a predefined threshold).

Suppose that M is set to 150 when X_i varies between 0 and 65535. In Figure 7.1.a, as the value of X_{i+1} is 200 higher than X_i and X_{i+2} it will be marked as an important byte. But in

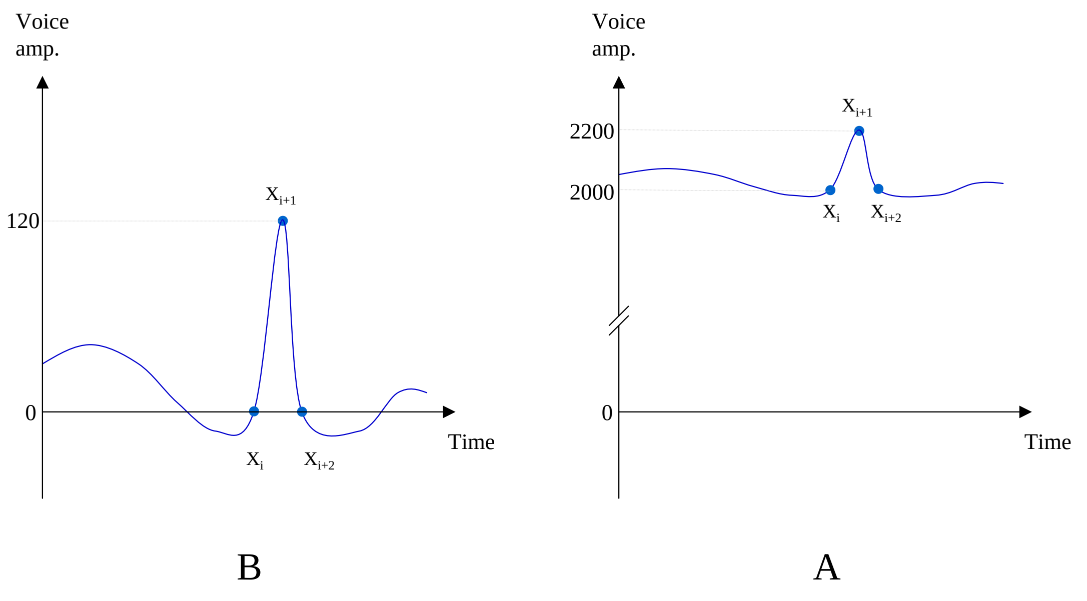

Figure 7.1: Voice samples with different amplitudes.

Figure 7.1.b the X_{i+1} will not be marked because it is only 120 higher than the X_i and X_{i+2}. To improve this procedure DADPCM5 uses a relative way to choose the important samples.

The new procedure divides the difference between the calculated average and the main byte by the main byte.Therefore, if

$$\frac{|\frac{X_{i-1}+X_{i+1}}{2} - X_i|}{|Xi| + 1} > M, \tag{7.2}$$

then X_i will be marked as an important sample. This formula can be written as

$$|\frac{X_{i-1} + X_{i+1} - 2X_i}{|2X_i| + 1}| > M. \tag{7.3}$$

Using the above formula, the DADPCM5 sender routine chooses some important samples and inserts them into the next packet. Same as DADPCM and in one hand, using a large M, packets have short length, but the resulting regeneration is not very accurate. On the other hand, using small M, the packets will have long length which will generate accurate reconstructed samples. Again, the M parameter determines the amount of redundant data, which is not required if all packets are received at the destination. In short, M defines the amount of overhead that DADPCM5 adds to the voice samples. Important samples from the first stream are extracted and inserted into the second stream and important samples from the second stream are inserted into the third one etc. In the end, important samples of the last stream are inserted into the first one as is shown in Figure 7.2.

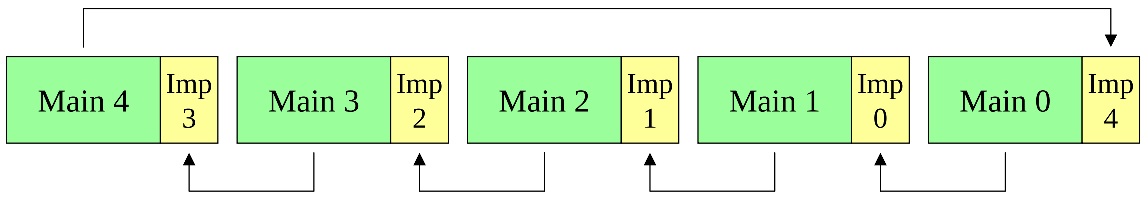

Figure 7.2: Allocation of important samples in DADPCM5.

7.1.2 Second step: Encoding and decoding samples

This step receives a stream from the first step and applies the ADPCM compression algorithm to it. The algorithm encodes the difference between a predicted sample and the speech sample, so it provides efficient compression with a reduction in the number of bits per sample [83]. It converts each eight-bit voice sample to a four-bit ADPCM code. As with DADPCM, DADPCM5 inserts three bytes as decoding coefficients in all packets. Consequently, each packet can be decoded individually.

7.1.3 Third step: Packing the encoded samples

This step receives coded samples from the second step, inserts decoder coefficients and the location of important bytes to prevent them appearing twice at the output. As each important byte is coded in four bits, it forms half of a byte of a packet. Therefore, the sender must mark each important nibble in the packet.

In the end, there will be five packets named P_0, P_1, P_2, P_3 and P_4. The sender tries to send them in such a way that no two consecutive voice samples will be transmitted in two consecutive packets. Therefore, the sender rearranges the packets and sends them in the order of: P_0, P_2, P_4, P_1, P_3 as shown in Figure 7.3.

This transmitting scheme allows DADPCM5 to cope with burst losses as well as individual losses. For instance, suppose that the second and the third transmitted packets are lost within the network. The receiver (or the active node) can regenerate the $x_2, x_7, \cdots$ samples by $x_1, x_6, \cdots$ and $x_3, x_8, \cdots$ from the arrived packets. Using the important samples of P_2 that is available in P_3, the receiver can achieve a better estimation of the lost samples. P_4 can be estimated by P_0 and P_3, and using its important samples in P_0, the receiver will produce good voice quality even when the network loses two consecutive packets. The structure of a packet for DADPCM5 is the same as the packet structure for DADPCM in Figure 6.3.

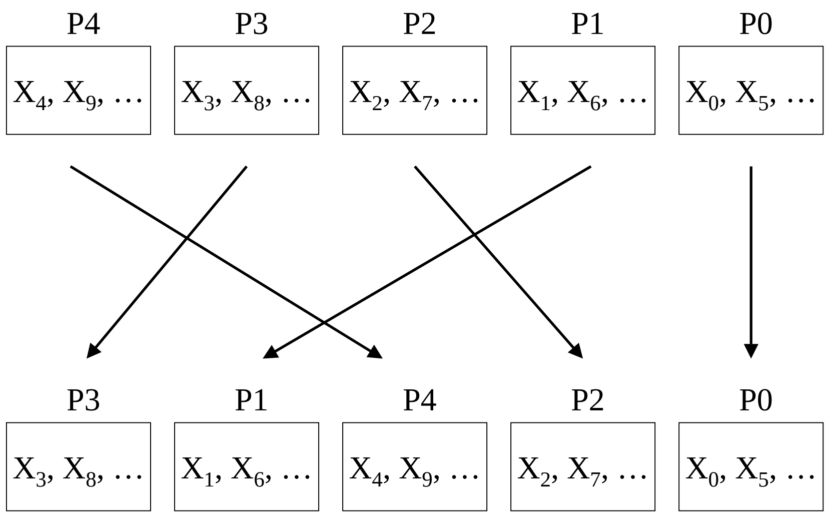

Figure 7.3: DADPCM5 packet arrangement.

7.2 Receiver routine

In order to regenerate the voice samples, the receiver program is written on an event-driven basis and based on the state of a voice flow and its input events. The software that is written based on the SDL format has 32 states and 9 events per state. The whole SDL's of the receiver routine is available in Appendix B. This routine checks the voice flows and regenerates sound based on the number and position of the arrived packets.

7.2.1 No loss

On receiving all 5 packets of a segment (no loss detected), the receiver decodes the packets separately and merges the samples into one voice stream. The receiver also deletes the important codes using their addresses.

7.2.2 Individual loss

Losing a packet in the network, the receiver decodes other arrived packets to obtain voice samples. The samples are divided into main and important codes. As was mentioned in the third step of the sender routine, the receiver interpolates all the samples of the previous and the following packets adjacent to the lost packet and collects the voice samples. Using important samples of the lost packet, located in the following packet, the receiver gains better sample estimation and sends the result to the output stage.

If two lost packets of a block are detected, the DADPCM5 receiver routine decides to regenerate them based on their location. The receiver checks the sequence of the lost packets before rearranging. For instance, although P_0 and P_2 are consecutive packets in the network, if they are not available at the receiver, they are considered as two individual lost packets because they were not consecutive before rearranging. So DADPCM5 generates them based on what it does in a single packet loss situation.

When the two lost packets are consecutive there is no way to regenerate accurate samples because a series of voice samples and their important bytes are lost. For example, suppose that P_0 and P_1 are lost. Having the important samples of P_1 in P_2, the routine copies the P_2 as P_1 and then it generates P_0 from the arrived P_4 and regenerated P_1. So in this case, the estimation error is higher than in the previous case.

7.2.3 Heavy burst loss

Where there are three or four lost packets of a block, the routine does the same as mentioned in the two consecutive packet loss situations and regenerates the lost packets.

7.3 Active node routine

Each active node receives a packet, saves it and transmits it toward the other nodes or to the receiver. Detecting a lost packet, the node uses the same routine as the receiver to reconstruct voice samples. Eventually the regenerated packet is sent to the other nodes or to the receiver.

7.4 Results of DADPCM5

In a series of tests, 64000 bytes of a native English-speaking woman's voice samples were selected and packed at the sender. The lossy link discarded some packets and the active node had to regenerate the lost packets. Using NS-2, four schemes were tested to compare their perceived speech quality after passing through the lossy link.

- Silence substitution: It uses 64 kbps PCM codes and it makes a packet with 20 ms length. It inserts silence instead of the lost packets.

- Packet repetition: Same packing method as the silence substitution but it repeats the last arrived packet in place of the lost packet.

- FEC: It uses 64 kbps PCM codes of a segment and 6.4 kbps LPC codes of its previous segment to make a packet.

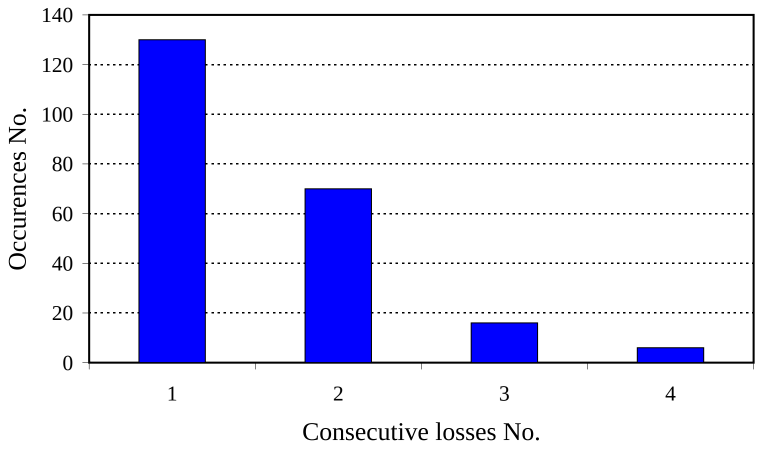

Figure 7.4: Number of occurrence in a burst loss pattern.

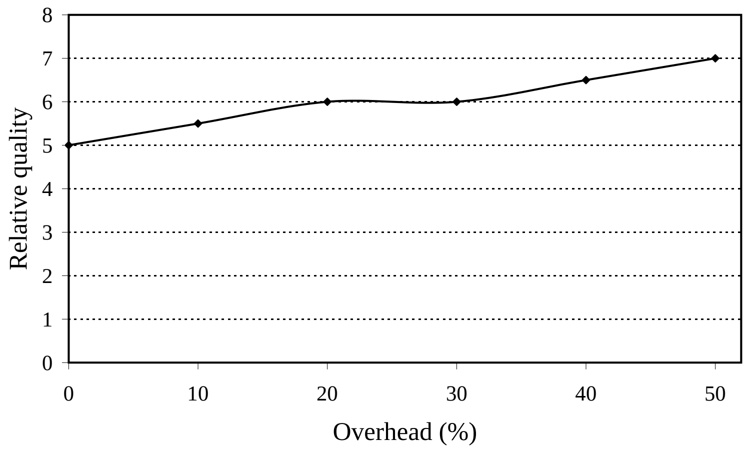

Figure 7.5: DADPCM5 quality versus data rate.

- DADPCM5: uses various values of M.

The tests aimed to evaluate DADPCM5 in terms of its voice quality and its required processing time. The results are explained in the following subsections.

7.4.1 Quality versus data rate

In the test, a very lossy network which loses 40% of packets is considered. The number of occurrences versus the number of consecutive losses for this burst loss pattern is shown in Figure 7.4.

The audience was asked to assess the regenerated voice based on its quality (Appendix C). Figure 7.5 shows the relative quality of the voice versus its overhead percentage.

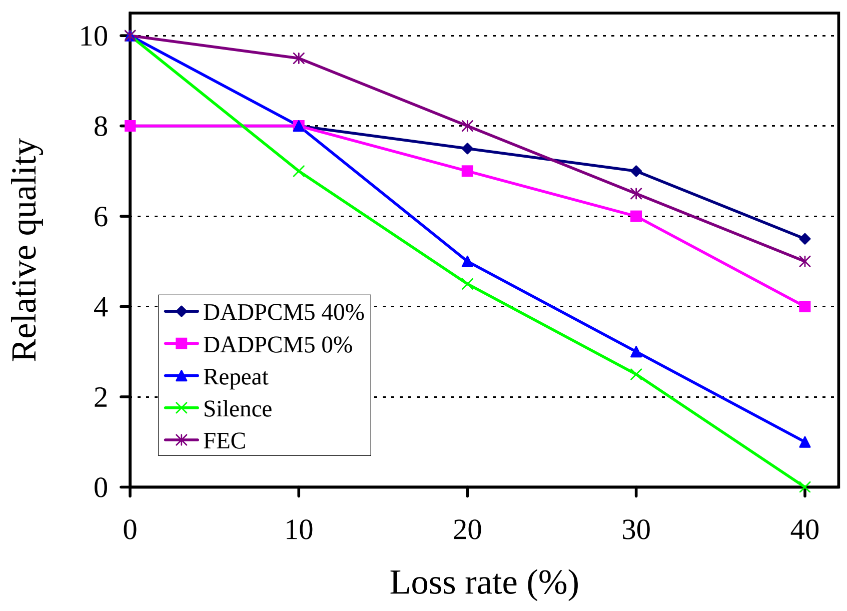

Figure 7.6: Comparison of voice quality of different schemes (individual losses).

7.4.2 Quality versus loss rate

These tests were done with different loss patterns and rates.

Individual loss

Figure 7.6 shows the result of the tests that were done with different loss rates when the lost packets were individual. This loss pattern is likely to happen when the network suffers from bit error problems.

The result implies that the voice quality of silence substitution and packet repetition schemes decreases drastically with upper loss rates. DADPCM5 scheme without any important bytes (0% overhead) has good quality even with a very high drop rate. The more overhead it has, the better quality it achieves. The chart shows that with 40% overhead, DADPCM5 has good quality even at 40% packet loss. The quality of regenerated voice with DADPCM5 is significantly higher than with the other schemes even with lower overhead. Besides, the chart shows that FEC offers better quality than DADPCM5 in low loss rates. Considering the high amount of processing time needed by FEC to decode its LPC part, it can be concluded that DADPCM5 offers almost the same quality as FEC with lower computing time.

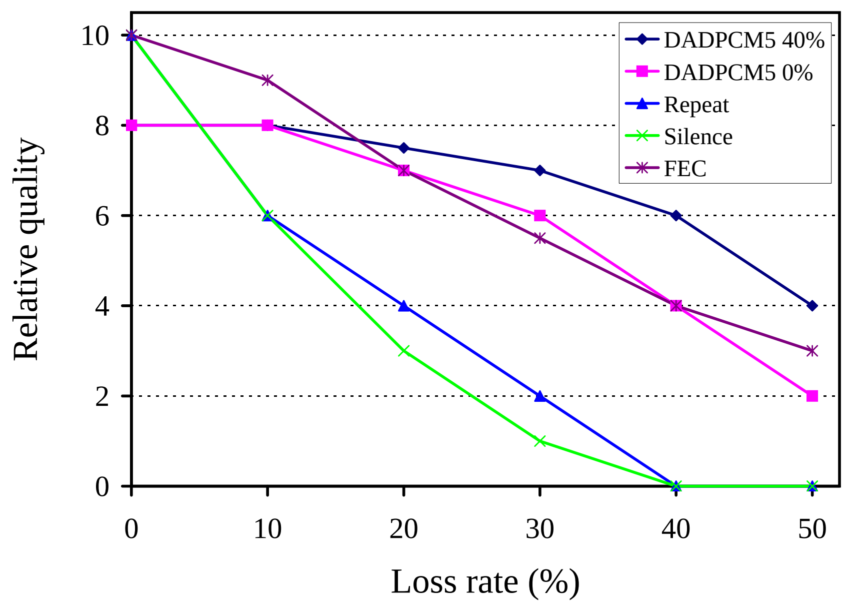

Figure 7.7: Comparison of voice quality of different schemes (burst losses).

Burst loss

In another test, the background traffic, bandwidth of bottleneck link, and other simulation parameters were adjusted to generate burst losses. This loss pattern, which is matched well with the Gilbert model, is likely to happen when the network suffers from congestion problems.

Figure 7.7 presents the results of the tests. It shows that silence substitution and packet repetition cannot cope with high burst loss rate. It can be seen that FEC offers better quality than DADPCM5 in low loss and DADPCM5 offers better quality than FEC in high loss rates.

7.4.3 Processing time

As in Subsection 6.4.3, the active node was implemented in a computer with Pentium IV 2.4 GHz and 512-MByte RAM running Windows 2000. In the worst case scenario, the lossy network dropped half of the packets and the active node had to regenerate them from its received packets. The average processing time that was needed to recover each lost packet was measured. The packet recovery time varied with M and Figure 7.8 shows the recovery time versus the packet overhead.

It can be seen that increasing packet overhead from 0 to 39% (with decreasing M) increases recovery time from 82.3 to 115.4 μsec. In DADPCM5, on one hand, decreasing M increases the packet overhead and also increases recovery time while on the other hand, it increases the quality of the regenerated voice in the receiver.

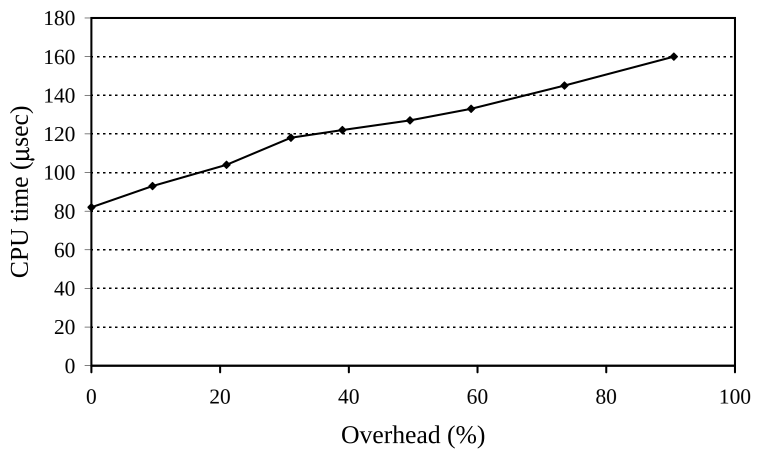

Figure 7.8: DADPCM5 processing time versus overhead.

7.5 Summary and conclusion

DADPCM5 as a modified version of DADPCM can cope with burst packet losses in the network. It splits voice samples to five streams and it extracts important samples from one stream and inserts them in another stream. Therefore, it produces different length packets. Rearranging the packets before transmitting them, makes DADPCM5 an applicable voice packing scheme for a burst loss environment. The performed tests evaluated its efficiency in regenerating lost voice packets in the network. The tests showed that the scheme produces much better quality compared with the other schemes even in a burst loss situation. The tests showed that the more important bytes it has, the better the voice quality it achieves at the receiver. Comparing the test results in an individual loss situation reveals that DADPCM5 has worse voice quality than DADPCM but it can be used to cope with burst loss problem.

Chapter 8

Detached Fourier Coefficients Packing Method (DFCPM)

This voice-packing scheme works based on sending the discrete Fourier transform coefficients of a voice segment and regenerating the voice by applying the inverse Fourier transform routine to the arrived coefficients at the receiver. In this Chapter the whole idea of transmitting the coefficients followed by some of the successful and unsuccessful experiments is explained.

8.1 Discrete Fourier Transform

Suppose that voice samples are divided into some segments of length N. Considering a segment as a finite-length sequence of $x(n)$, the amount of $x(n)$ outside the range $0 \leq n \leq N-1$ will be zero. A finite-duration sequence $x(n)$ of length N has a Fourier transform of

$$X(w) = \sum_{n=0}^{N-1} x(n)\, e^{-jwn}, \qquad 0 \leq w \leq 2\pi. \tag{8.1}$$

When $X(w)$ is sampled at equally spaced frequencies $w_k = 2\pi k/N$ and $k = 0, 1, ..., N-1$, the Discrete Fourier Transform (DFT) of $x(n)$ can be calculated by

$$X(k) = \sum_{n=0}^{N-1} x(n)\, e^{-j2\pi kn/N}, \qquad 0 \leq k \leq N-1. \tag{8.2}$$

Its Inverse Discrete Fourier Transform (IDFT) will be [42]

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k)\, e^{j2\pi kn/N}, \qquad 0 \leq n \leq N-1. \tag{8.3}$$

Defining $w_N = e^{-j2\pi/N}$, the formulae can be written as

$$X(k) = \sum_{n=0}^{N-1} x(n)\, w_N^{kn}, \tag{8.4}$$

and

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k)\, w_N^{-kn}. \tag{8.5}$$

$X(k)$ as DFT of $x(n)$ has some important properties that must be considered. It can be written as a series of complex numbers such as $A + Bi$ with different orders except the 0^{th} order, which is a pure real number representing the DC component of the voice segment. Quantities A and B present the amplitude of cosine and sine waves with the frequency related to their order. Suppose that the m^{th} order coefficients of $X(k)$ is $X(m) = A+Bi$. It presents a cosine wave with the frequency of $\frac{m-1}{N}$ and amplitude of $\frac{A}{N}$ and a sine wave with the same frequency and amplitude of $\frac{B}{N}$. If $x(n)$ is a discrete signal with N samples, the $X(k)$ will have N series of complex coefficients. Therefore, the voice sender routine can extract all N complex coefficients of the voice segment and transmit them to the receiver. The receiver can easily regenerate the voice using (8.5). The real part of IDFT of the received coefficients will be the voice, which must be sent to the output stage. Deeper investigation of the coefficients reveals that the second half of the complex coefficients of $X(k)$ is conjugated with its first half, so it is not necessary to send all the coefficients to the receiver. The sender routine can send half of the coefficients and the receiver can easily regenerate the main signal [23]. To calculate DFT of a signal with the length of N, the direct evaluation requires N^2 complex multiplication and additions, and for large N, it is rather costly in computing time. To find a faster way to calculate DFT, it is necessary to pay attention to some of the DFT's properties. The linearity property of DFT is favoured, which is explained as follows. If two finite-duration sequences of $x_1(n)$ and $x_2(n)$ with the length of N are linearly combined as in

$$x_3(n) = ax_1(n) + bx_2(n), \tag{8.6}$$

then the DFT of $x_3(n)$ is

$$X_3(k) = aX_1(k) + bX_2(k) \tag{8.7}$$

with the duration of N. This property is used in the decimation in time Subsection.

8.2 Decimation in time

If the length of a voice segment is an even number, it is possible to separate the segment into two streams, a stream with all even-numbered samples and another stream with all the odd-numbered samples as it is shown in Figure 8.1. It then follows that

$$x_{even}(l) = x(2n), \qquad 0 \le n \le \frac{N}{2} - 1 \tag{8.8}$$

and

$$x_{odd}(l) = x(2n+1), \qquad 0 \le n \le \frac{N}{2} - 1. \tag{8.9}$$

Their DFT are given by

$$X_{even}(k) = \sum_{l=0}^{\frac{N}{2}-1} x_{even}(l)\, w^{2lk} \tag{8.10}$$

and

$$X_{odd}(k) = \sum_{l=0}^{\frac{N}{2}-1} x_{odd}(l)\, w^{2lk}. \tag{8.11}$$

As each stream has $\frac{N}{2}$ samples, DFTs of these sequences will have $\frac{N}{2}$ points. In other words, they may be treated as two periodic sequences with the period of $\frac{N}{2}$ points. Substituting the new variable r as

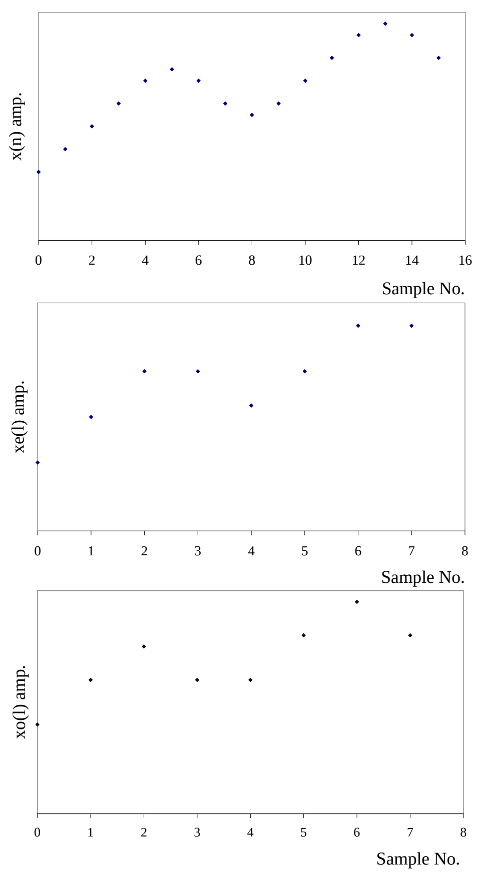

Figure 8.1: Decimation in time for a voice sample $x(n)$.

$$n = 2r, \qquad \text{for } n \text{ even}$$

and

$$n = 2r+1, \qquad \text{for } n \text{ odd}$$

the formulae can be written as

$$\begin{aligned} X(k) &= \sum_{r=0}^{\frac{N}{2}-1} x(2r)\, w_N^{2rk} \;+\; \sum_{r=0}^{\frac{N}{2}-1} x(2r+1)\, w_N^{(2r+1)k} \\ &= \sum_{r=0}^{\frac{N}{2}-1} x(2r)\, (w_N^2)^{rk} \;+\; w_N^k \sum_{r=0}^{\frac{N}{2}-1} x(2r+1)\, (w_N^2)^{rk}. \end{aligned} \tag{8.12}$$

Calculating those DFTs, two series of complex coefficients will be generated. The sender transmits these two series to the receiver and the receiver which requires the DFT of the entire sequence obtains the whole DFT by

$$X(k) = \sum_{l=0}^{\frac{N}{2}-1} (x_{even}(l)\, w^{2lk} + x_{odd}(l)\, w^{(2l+1)k}). \tag{8.13}$$

Considering the properties of DFT, it can be calculated by

$$X(k) = \sum_{l=0}^{\frac{N}{2}-1} x_{even}(l)\, w^{2lk} + w^k \sum_{l=0}^{\frac{N}{2}-1} x_{odd}(l)\, w^{2lk} \tag{8.14}$$

or

$$X(k) = X_{even}(k) + w^k\, X_{odd}(k). \tag{8.15}$$

That important result implies that instead of having N^2 operations, DFT can be calculated with only $N + \frac{N^2}{2}$ operations, which is much less that N^2 when N is a big number [79]. At the receiver, the main DFT can be achieved by

$$X(k) = \begin{cases} X_{even}(k) \;+\; w^k\, X_{odd}(k), & 0 \le k \le \frac{N}{2} - 1 \\ X_{even}(k - \frac{N}{2}) \;+\; w^k\, X_{odd}(k - \frac{N}{2}), & \frac{N}{2} \le k \le N - 1. \end{cases} \tag{8.16}$$

8.3 Motivation

A voice segment can be regenerated using its DFT coefficients. The sender must calculate the DFT of a voice segment, pack the DFT's coefficients and transmit them toward the receiver. Using the coefficients, the receiver must make the complex coefficients and after applying the IDFT formula, the real part of the result will be a voice that can be sent to the sound card. Two subjects are considered.

8.3.1 Compression:

Sending all the DFT coefficients needs a long packet, which is not suitable for the Internet. Therefore, routines to make shorter packets that can regenerate the voice segment need to be addressed. To have shorter packets, tests have been done that are explained in the following subsections.

8.3.2 Lost packet effect compensation:

As some packets will be dropped during the transmission, it is desired to find a method to pack the packets in such a way that the regenerated voice can tolerate loss during the transmission. To do that, packets need a packing strategy that helps the receiver regenerate the lost packets and reconstruct the voice at the receiver.

8.4 Experiments

To satisfy the compression and lost packet compensation aims, a number of experiments have been conducted. The experiments tried to split a voice segment or its DFT coefficients in such a way that they can generate two or more packets. Those packets were dropped with different loss patterns. Using the arrived packets, the regenerated voice qualities were studied. The following subsections explain some successful and unsuccessful experiments and their related results.

8.4.1 First experiment

In this experiment a voice segment is chosen and using (8.4), its DFT coefficients are extracted. If the voice segment has N samples, its DFT has N complex coefficients. As the second half of the coefficient is conjugated with its first part, it is not necessary to transmit them toward the sender. A threshold was chosen and all the coefficients whose real and imaginary parts are lower than that threshold, were deleted. A packet containing the real and imaginary parts of the remaining coefficients is packed and sent to the receiver. At the receiver, the real and imaginary coefficients are merged to make the complex coefficients. The real part of IDFT coefficients generates the sound, which is sent to the sound card. The quality of the sound depends on the threshold and it is obvious that the lower the threshold, the better the achieved quality at the receiver. In a practical test with 1000 bytes of female voice samples with different thresholds, the quality of the regenerated voice is tested. Table

Table 8.1: Threshold and quality of a voice segment

Threshold	No. of bytes	Quality
0.001	994	10
0.01	984	10
0.1	862	10
0.5	446	10
0.6	382	9.5
0.7	336	9
0.9	282	8
1	258	7

Table 8.2: No. of bytes to send of a voice segment

Segment length	No. of bytes to send
40	12
42	20
44	14
...	...
118	38
120	60
122	34

8.1 shows the threshold, the number of bytes to send and the sound quality from 0 to 10 for that voice segment.

In this experiment two points must be considered.

Data rate

It has been observed that with a constant threshold, the number of bytes to send is strongly related to the length of the input voice segment. Table 8.2 shows the number of sending bytes versus the segment length for a female voice.

It can be seen that changing the segment length from 118 to 120 and then to 122 causes a drastic change in the sending bytes from 38 to 60 and finally to 34. Having an integer number of complete periods of a waveform generates lower bytes compared with the non-integer or uncompleted periods of that waveform.

Table 8.3: DFT coefficients samples

Order	Coefficients
0	12 + 0i
1	13 + 14i
2	0
3	16 + 17i
4	0
5	18 + 19i
6	0
7	11 + 10i

Coefficient order

As mentioned in Subsection 8.1, in order to regenerate the voice, not only are the coefficients important, but also their order must be considered. This experiment omits some coefficients and sends the rest to the receiver. To indicate the order of the sent coefficients, some *order identifiers* must be inserted into the packet. Order identifiers are bytes in which some bits related to the coefficient order are set. For example suppose that the coefficients, as shown in Table 8.3 must be sent to the receiver.

With the non-zero coefficients, a byte with 0,1,3,5 and 7 bits set must be sent to help the receiver put the coefficients into their proper order. Therefore, in the above example 11 bytes must be transmitted, ten bytes for five complex coefficients and one byte as an order identifier. To sum up, the number of transmitted bytes is more than is mentioned in the Table 8.1.

8.4.2 Second experiment

In the second experiment, which includes some tests, a voice segment is chosen and its DFT coefficients are extracted. As in the first experiment, the number of coefficients is reduced by comparing them with a threshold. Tests are then performed on the real and imaginary parts of the remaining coefficients.

Test one:

The real parts are packed in a packet and the imaginary parts are packed in another packet and those packets are sent to the receiver (Figure 8.2).

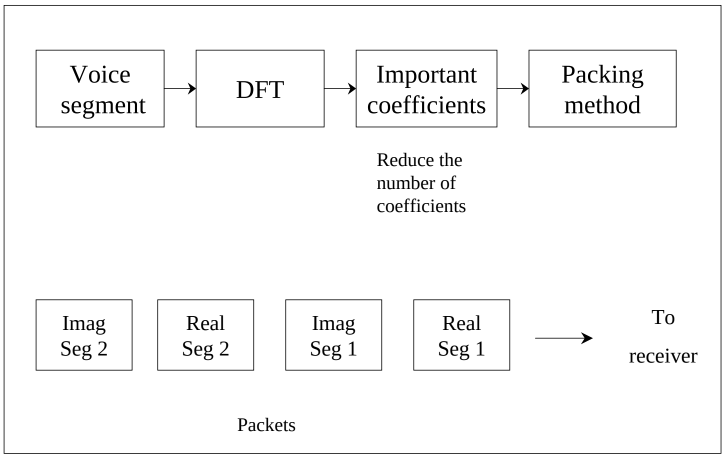

Figure 8.2: Packing scheme for test one.

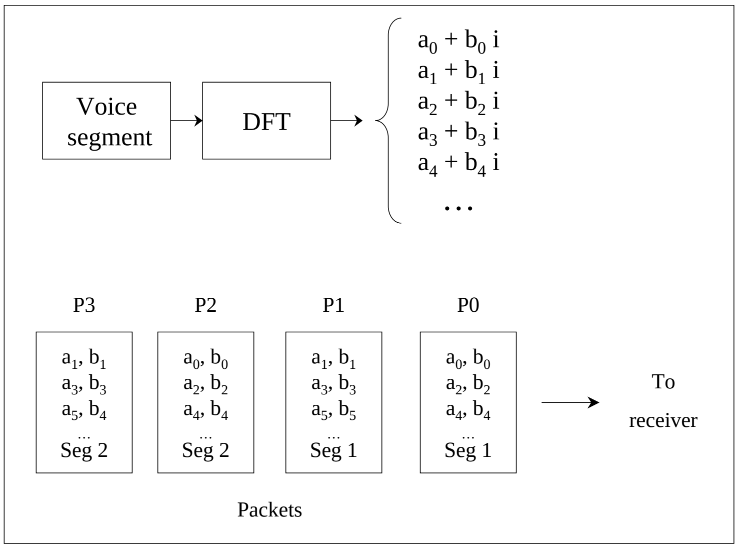
Voice segment
DFT
a0 + b0 i
a1 + b1 i
a2 + b2 i
a3 + b3 i
a4 + b4 i
…
P3
P2
P1
P0
a1, b1 a3, b3 a5, b4 … Seg 2
a0, b0 a2, b2 a4, b4 … Seg 2
a1, b1 a3, b3 a5, b5 … Seg 1
a0, b0 a2, b2 a4, b4 … Seg 1
To receiver
Packets

Figure 8.3: Packing scheme for test two.

Each packet contains its length, segment number, coefficients and their type and order identifiers. It has been observed that when a packet is lost, the other arrived packet cannot regenerate the voice segment and the sound has very poor quality. This result has occurred even when the threshold is set to zero and all the coefficients are sent to the receiver.

Test two:

Setting the threshold to zero, this test splits the coefficients based on their order into two packets. Even-numbered, ordered coefficients are packed in a packet and odd-numbered, ordered coefficients are packed in another packet (Figure 8.3).

In this test each packet contains its length, real and imaginary parts of the specified coeffi-

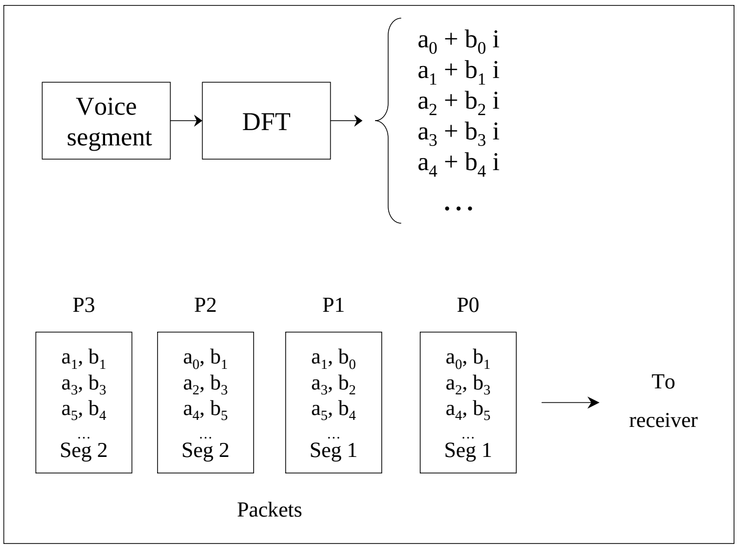
Voice segment
DFT
a0 + b0 i
a1 + b1 i
a2 + b2 i
a3 + b3 i
a4 + b4 i
...
P3
P2
P1
P0
a1, b1 a3, b3 a5, b4 ... Seg 2
a0, b1 a2, b3 a4, b5 ... Seg 2
a1, b0 a3, b2 a5, b4 ... Seg 1
a0, b1 a2, b3 a4, b5 ... Seg 1
To receiver
Packets

Figure 8.4: Packing scheme for test three.

cients and voice segment number. It is revealed that on losing a packet, the regenerated sound has very poor quality and therefore this scheme cannot cope with the packet loss problem when transmitting voice over the Internet.

Test three:

In this test, the real parts of the even-numbered complex coefficients with the imaginary parts of the odd-numbered coefficients are packed in a packet and the rest of the coefficients are packed in another packet (Figure 8.4).

The practical test revealed that this packing scheme cannot cope with the packet-loss problem and if a packet is lost, the regenerated voice has very bad quality.

8.4.3 Third experiment

This experiment tries to pack two consecutive voice segments in a number of packets in order to estimate a lost packet of one segment from the other available packets. To do that, two small equal-sized voice segments are chosen (x_1, x_2) and the DFT coefficients of each segment are calculated. Each DFT has two parts; so four different parts are generated. This experiment examines packing those four components in two packets.

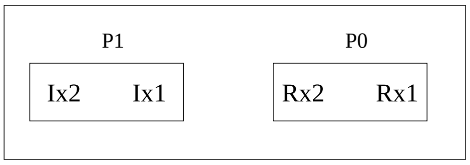

Figure 8.5: Packets of test one.

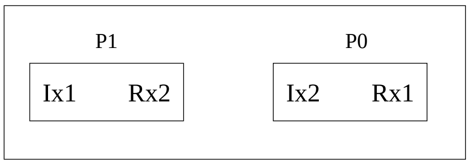

Figure 8.6: Packets of test two.

Test one:

The real parts of both segments (R_{X1}, R_{X2}) are packed in a packet and the imaginary parts of them are packed in another packet (Figure 8.5).

This test is the same as the first test in the second experiment and this packing scheme cannot be used in lossy environments.

Test two:

Packing a packet with R_{X1} and I_{X2} and packing another packet with R_{X2} and I_{X1} generates two packets, which are shown in Figure 8.6.

Again this test failed to generate acceptable voice quality. With these experiments, it must be mentioned that the quality of regenerated voice strongly depends on the length and position of the two input voice segments. If they are similar or if they are two periods of a periodic part of a voiced segment, the regenerated sound has good quality, but if they are not, the regenerated sound cannot be classified as good sound.

8.4.4 Fourth experiment

Similar to the third experiment, this experiment tries to pack two consecutive voice segments in a number of packets. Two small equal-sized voice segments (x_1, x_2) are chosen and each of them is separated into two streams (one with even-numbered samples and another with odd-numbered samples). Therefore, the streams will be

$$x_{1e}(l) = x_1(2n), \qquad 0 \leq n \leq \frac{N}{2} - 1,$$

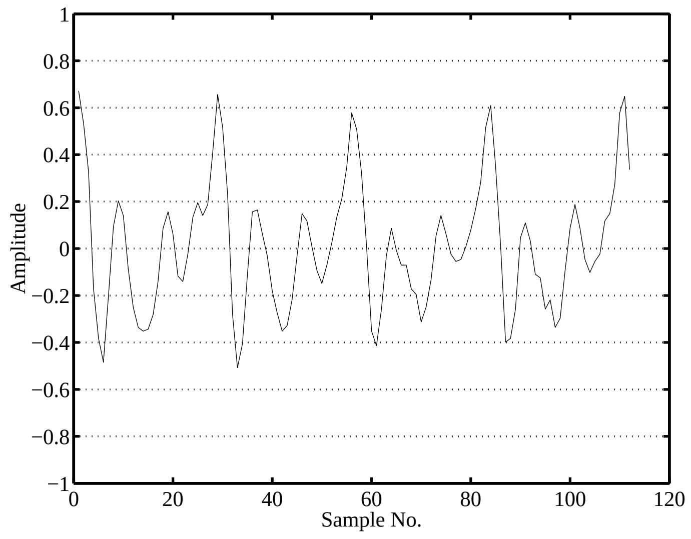

Figure 8.7: Periodic voice segment chosen for the fourth experiment.

$$x_{1o}(l) = x_1(2n+1), \qquad 0 \leq n \leq \frac{N}{2} - 1,$$

$$x_{2e}(l) = x_2(2n), \qquad 0 \leq n \leq \frac{N}{2} - 1$$

and

$$x_{2o}(l) = x_2(2n+1), \qquad 0 \leq n \leq \frac{N}{2} - 1.$$

The DFT coefficients of those four streams are calculated and as a result, eight series of coefficients are generated. This experiment aims to find similar coefficients among those parameters, so a lost one can be regenerated from the other ones. For example, if the real part of x_1 and the real part of x_2 were similar, it would be possible to pack each of them in packets and regenerate one of them from the other one if a packet is lost. To check the similarity, two voice segments have been chosen which are shown in Figure 8.7. The first segment is chosen from the 1^{st} to 56^{th} sample and the second one is chosen from the 57^{th} to 112^{th} sample of a voice segment.

All eight parameters (real and imaginary parts of each four DFT coefficients) are shown in Figure 8.8.

It can be seen that, although real parts of x_{1o} and x_{2o} have some similar sections, the real parts of x_{1e} and x_{2e} are not similar, and also Ix_{1o} and Ix_{2o} cannot be classified as similar waveforms. To sum up, this packing scheme, that is based on packing the DFT coefficients of even and odd numbered streams of two consecutive voice segments cannot tolerate lost packets on the Internet.

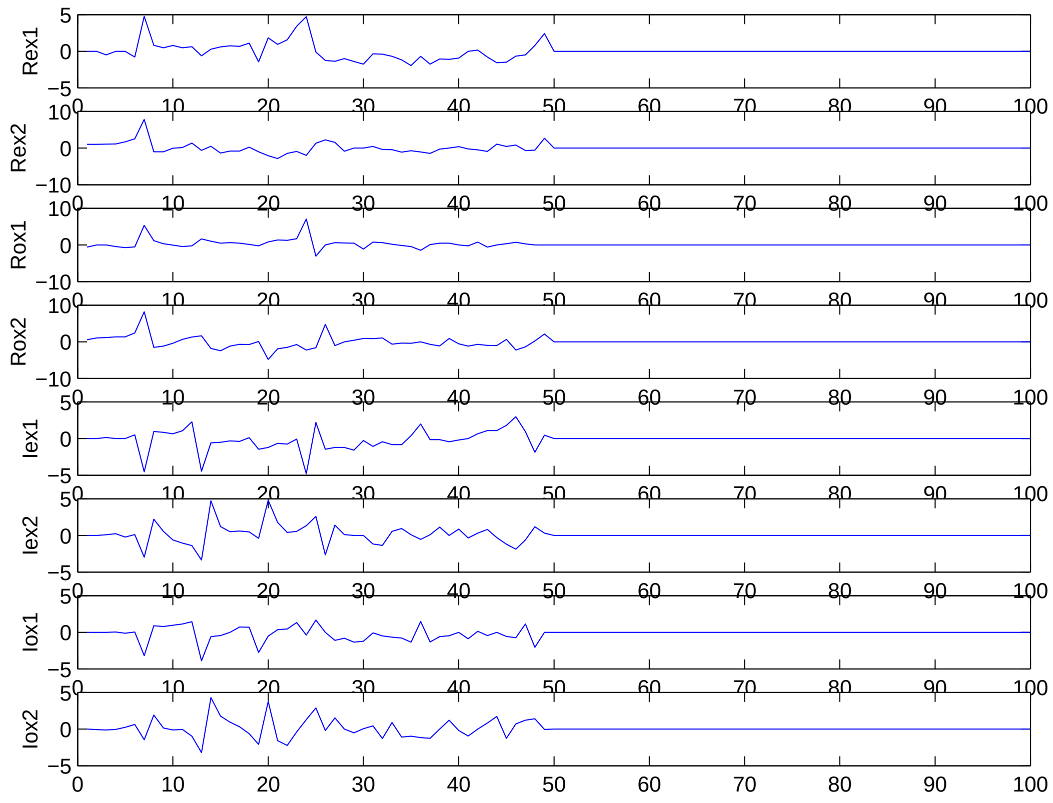

Figure 8.8: DFT coefficients of the chosen voice segment.

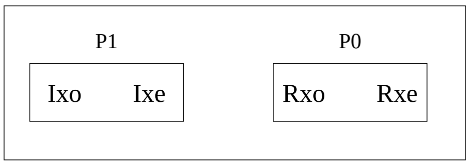

Figure 8.9: Packets of test one.

8.4.5 Fifth experiment

In contrast with the previous packing scheme, this experiment packs only one voice segment in two packets. A voice segment is chosen and is split into two streams (one with even-numbered samples and another with odd-numbered samples of the voice segment $(x_e,\ x_o)$). The DFT coefficients of these series are calculated resulting in four series of coefficients (two real and two imaginary series).

Test one:

The real parts of each series are packed in one packet and the imaginary parts of them are packed in another packet (Figure 8.9).

This test failed to generate good voice quality when a packet was lost.

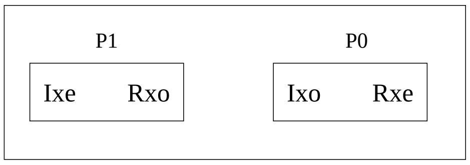

Figure 8.10: Packets of test two.

Test two:

The real part of the even series with the imaginary part of the odd series are packed in a packet and the real part of the odd series with the imaginary part of the even series are packed in another packet (Figure 8.10).

This test also failed to produce acceptable sound when a packet was lost.

Test three:

When a packet is packed with all coefficients of the even series and another packet is packed with all coefficients of the odd series, the result is good enough to be studied in detail. Therefore, this coding scheme referred to as Detached Fourier Coefficients Packing Method (DFCPM) which is explained in the next Section. A modified version of DFCPM named Enhanced DFCPM (EDFCPM) is explained later.

8.5 Detached Fourier Coefficient Packing Method (DFCPM)

The following subsections describe DFCPM based on its sender, receiver and active node routines, followed by some of its test results.

8.5.1 Sender routine

The DFCPM sender routine gets sound samples and divides them into some 80 byte length (10ms) segments. As in DSPM, the classifier routine classifies segments based on their energy. Then the consecutive segments of like classification are joined together to make blocks. There will be four block types, Voiced, Unvoiced and two transitions between them. As was explained in the DSPM packing scheme, the unvoiced blocks are still packed with their length and power so that a random noise generator could regenerate them at the receiver. In order to pack the voiced blocks, they are sent to a specific packing routine in which the block is separated to odd-numbered and even-numbered $(x_o,\ x_e)$ samples described by

$$x(n) \rightarrow \begin{cases} x_e(l) = x(2n), & 0 \leq n \leq \frac{N}{2} - 1 \\ x_o(l) = x(2n+1), & 0 \leq n \leq \frac{N}{2} - 1 \end{cases} \tag{8.17}$$

where N is the length of the input block. Using (8.10) and (8.11), DFCPM calculates the DFT coefficients of these two series by

$$X_e(k) = \sum_{l=0}^{\frac{N}{2}-1} x_e(l)\, w^{2lk} \tag{8.18}$$

and

$$X_o(k) = \sum_{l=0}^{\frac{N}{2}-1} x_o(l)\, w^{2lk}. \tag{8.19}$$

As the input voiced block has N samples, the x_e will have $\frac{N}{2}$ samples, so X_e will have $\frac{N}{2}$ real and $\frac{N}{2}$ imaginary coefficients [73]. Instead of having N^2 mathematical operation to gain the DFT coefficients of the voice segment, DFCPM does $2 * (\frac{N}{2})^2$ operation because of its packet length. As mentioned, the second half of the X_e is conjugated with its first half, so DFCPM deletes the second half. As a result, X_e will have $\frac{N}{4}$ real and $\frac{N}{4}$ imaginary coefficients.

In order to find the important coefficients, a threshold is applied. DFCPM finds the biggest coefficients among all coefficients, multiplies it by the threshold and gets a limit (a better way to calculate the threshold is explained in Subsection 8.5.4). The routine omits all coefficients of X_e less than the limit, and packs and sends the other coefficients to the receiver. Therefore, the number of transmitted bytes depends on N, the threshold and the coefficients of X_e. The routine follows the same process for X_o, so at the sender two packets will be available: one packet for odd-numbered and another packet for the even-numbered samples of the block.

To regenerate the original sound, the coefficients and their locations must be transferred to the receiver. The order identifiers in this scheme work as was explained in the first experiment (Subsection 8.4.1). It is assumed that the coefficients are converted from floating-point to a signed byte format, so it is necessary to transfer the maximum quantity of real and imaginary parts to regenerate the floating point coefficients at the receiver. The real and imaginary parts of the even series are packed in a packet and the real and imaginary parts of the odd series are packed in another packet (Figure 8.11).

The structure of a packet in DFCPM is shown in Figure 8.12.

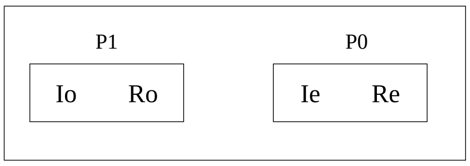

Figure 8.11: DFCPM packets.

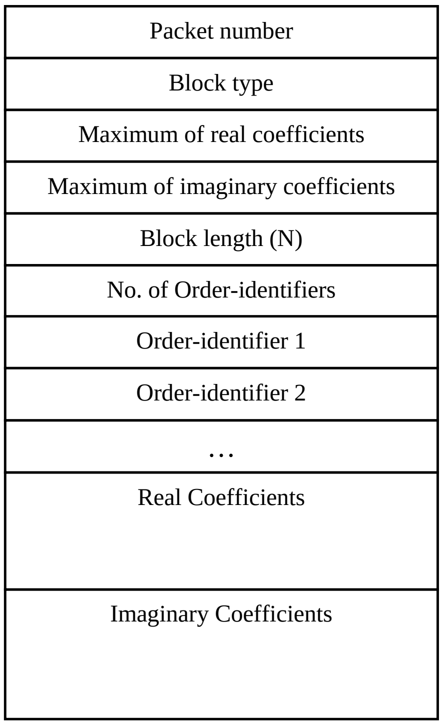

Figure 8.12: DFCPM packet structure.

Flow chart 8.13 describes the packing routine of DFCPM.

The whole sender routine of DFCPM is coded in Appendix B.

As the voiced-to-unvoiced and the unvoiced-to-voiced segments have different levels of importance in perceptual voice quality, they are packed the same as voiced blocks but with different threshold levels.

8.5.2 Receiver routine

The receiver routine is explained in two parts, one for the loss-less environment and another one for a lossy communication.

Receiving both packets

Where both packets $(X_e,\ X_o)$ of a block are received, the receiver requiring the DFT of the entire voice segment, generates the DFT using (8.15) as

$$X(k) = X_e(k) + w^k\, X_o(k), \qquad w = e^{\frac{-j2\pi}{N}}. \tag{8.20}$$

The easier way to calculate the result is using two half-length calculations with (8.16), described by

$$X(k) = \begin{cases} X_e(k)\ +\ w^k\, X_o(k), & 0 \le k \le \frac{N}{2} - 1 \\ X_e(k-\frac{N}{2})\ +\ w^k\, X_o(k-\frac{N}{2}), & \frac{N}{2} \le k \le N-1. \end{cases} \tag{8.21}$$

Gaining the $X(K)$, the receiver generates the Inverse Discrete Fourier Transform (IDFT) of the $X(K)$ to obtain the voice segments. The real part of IDFT will have the sound samples,

$$voice = real\ (IDFT(X(K))$$

and it can be sent to the sound card.

Receiving one packet

In case of losing one packet of the voice segment, the IDFT of the arrived packet will be calculated. Applying inverse decimation in time to the real part of IDFT will produce a sound, which must pass through a low pass filter to obtain better quality. DFCPM applies a 20^{th} order low pass FIR digital filter with 3400 Hz cut-off frequency to the result. Flow chart 8.14 describes the receiving routine of DFCPM.

The receiver routine of DFCPM program is available in Appendix B.

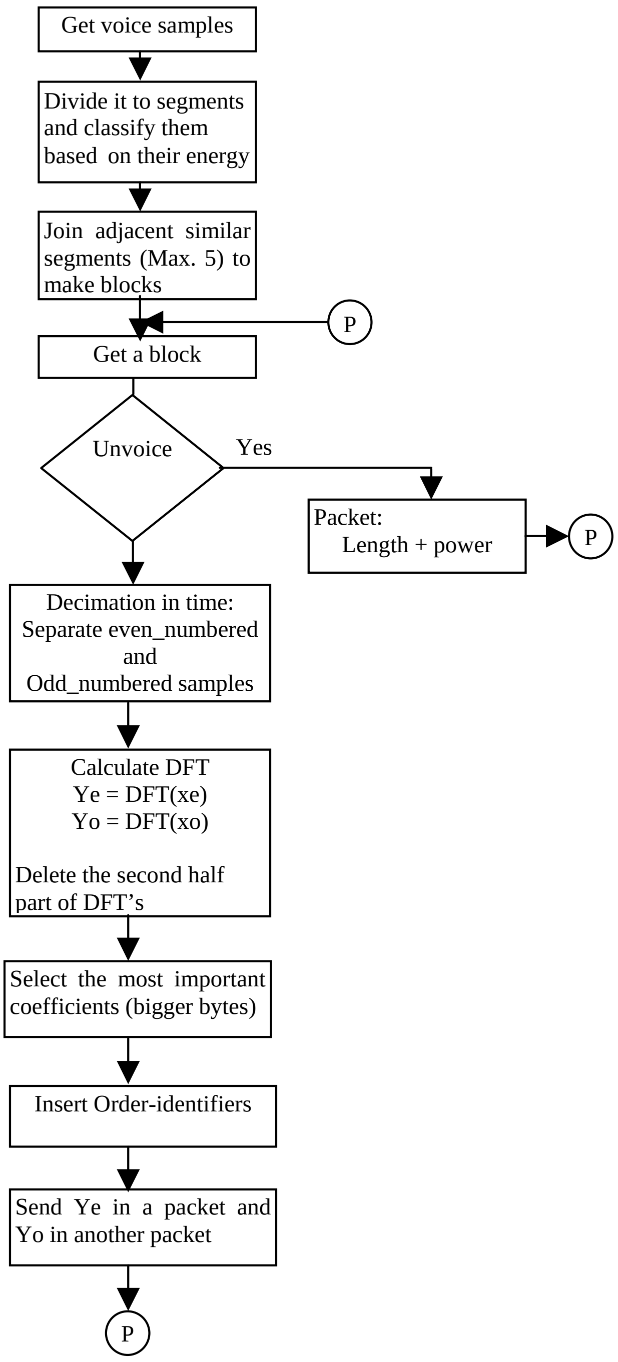
Get voice samples
Divide it to segments and classify them based on their energy
Join adjacent similar segments (Max. 5) to make blocks
P
Get a block
Unvoice
Yes
Packet: Length + power
P
Decimation in time: Separate even_numbered and Odd_numbered samples
Calculate DFT Ye = DFT(xe) Yo = DFT(xo)
Delete the second half part of DFT's
Select the most important coefficients (bigger bytes)
Insert Order-identifiers
Send Ye in a packet and Yo in another packet
P

Figure 8.13: Flowchart of DFCPM packing routine.

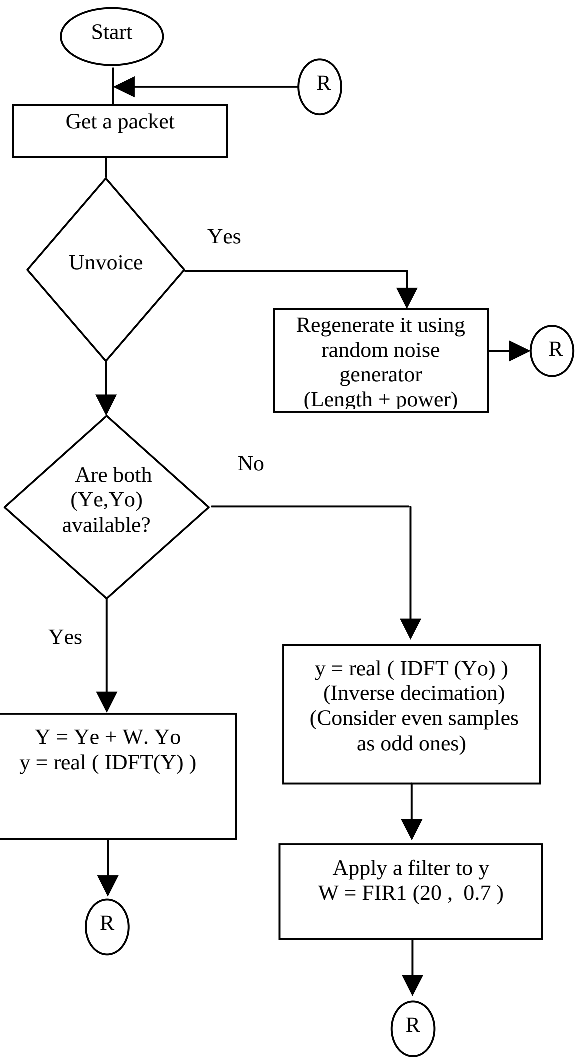
Start
R
Get a packet
Unvoice
Yes
Regenerate it using random noise generator (Length + power)
R
Are both (Ye,Yo) available?
No
Yes
y = real (IDFT (Yo)) (Inverse decimation) (Consider even samples as odd ones)
Y = Ye + W. Yo
y = real (IDFT(Y))
Apply a filter to y
W = FIR1 (20 , 0.7)
R
R

Figure 8.14: Flowchart of DFCPM receiver routine.

8.5.3 Active node routine

In active node, DFCPM benefits from both receiver and sender routines. The program checks the voice flow and if a lost packet is detected, it regenerates the packet using both above-mentioned routines.

8.5.4 Threshold

As discussed, the regenerated voice quality depends directly on the threshold. Also, the number of transmitted bytes is related to the threshold. In short, the voice compression rate and its quality depend on the threshold. The modified DFCPM program is written in such a way that it needs a data *transmission rate* as an input parameter. A voice block is split into two streams and each stream is sent to a section that calculates the DFT of that stream. An array includes the bigger value between the real and the imaginary parts of each coefficient of DFT which has been generated and sorted.

The program finds the length of the array and multiplies its length by the input data transmission rate. The result is a number whose ceiling points to a specific cell in the array that has the desired *limit*. To find the important DFT coefficients, the real and imaginary parts of each coefficient are compared with that limit, and if they are bigger than the limit, they will be considered important coefficients. At the end of this routine, a series of important coefficients will be generated in such a way that the series' length is equal to the length of the input block, multiplied by the input transmission rate divided by four.

For instance, suppose that 400 voice samples are fed to the DFCPM routine and it is required that they be compressed by 60% (data transmitted rate must be 40%). The routine generates two 200-byte streams. Using the DFT algorithm, each stream will generate 200 complex coefficients and half of them will be deleted because of their properties. The routine will find a limit that eliminates 60% of the coefficients, so each stream will have 40 real and 40 imaginary coefficients making an 80-byte length voice packet. There will be two 80-byte packets with a total length of 160 bytes that is 40% of the input 400-byte voice segment.

8.5.5 Result of DFCPM

To evaluate the DFCPM packing scheme, tests to examine quality versus data rate and quality versus loss rate, were performed.

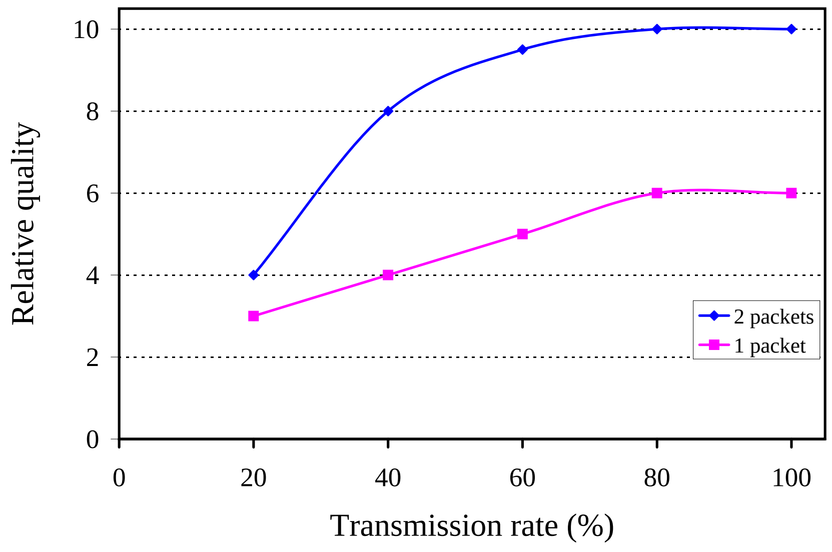

Figure 8.15: DFCPM quality versus threshold.

Quality versus data rate

A native English-speaking woman was selected and her 64000-byte voice samples were packed with the DFCPM coding scheme with different transmission rates. The audience was asked to judge the voice quality and assess the reconstructed voice (Appendix C). They had heard regenerated voice samples with one or two packets (even or both even-odd DFT coefficients at the receiver). Figure 8.15 shows the relative quality of those voice samples versus the transmission rate.

DFCPM transmits 46500 bytes even with 100% transmission rate because it compresses the unvoiced parts.

Quality versus loss rate

The quality of different coding schemes versus drop rate was investigated in another test. Figure 8.16 shows the relative quality versus loss rate for DFCPM with 80% and 40% transmission rates, silence substitution, packet repetition and FEC schemes.

It can be seen that in low loss rates, FEC offers better quality than DFCPM at the cost of a higher data rate.

Processing time

In order to measure the voice-packing and regenerating time of DFCPM, some measurements have been done. For the sake of simplicity, the DFCPM routine is written in Matlab code. Therefore, an interpreter running on a computer with Pentium-IV 2.4 GHz and 256 MByte

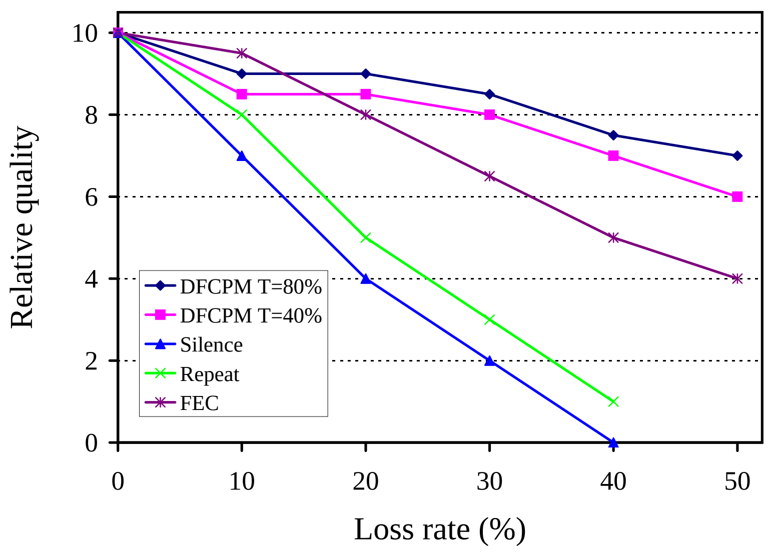

Figure 8.16: Comparison of voice quality of different schemes.

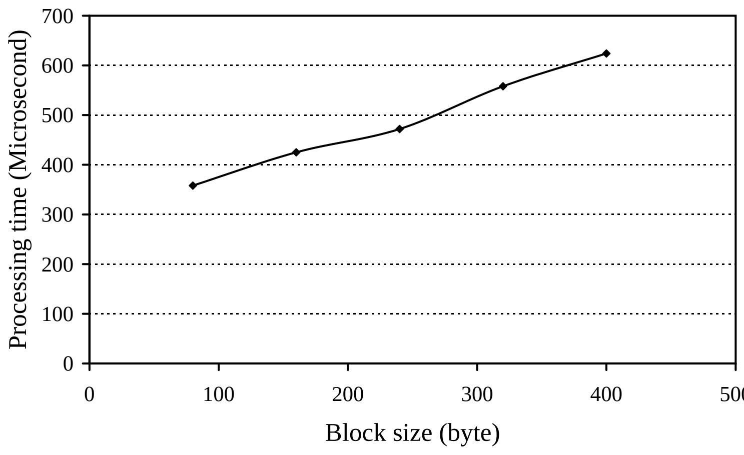

Figure 8.17: DFCPM processing time versus block size.

RAM running windows-2000 was used for the test. It has been observed that the time for voice-packing and regenerating is independent of the threshold and is directly related to the block length. The longer the block, the slower execution of the routine.

Figure 8.17 shows the processing time (including the packing and voice regenerating) versus the block size for an active node.

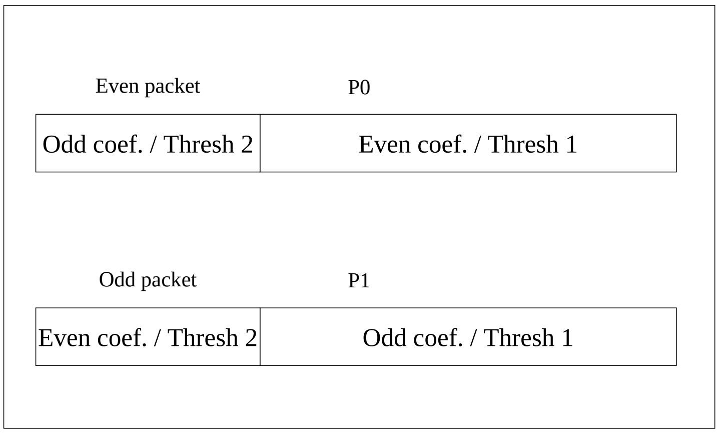

Figure 8.18: Packet structure for EDFCPM.

8.6 Enhanced Detached Fourier Coefficient Packing Method (EDFCPM)

The main concept of this packing scheme is the same as DFCPM but it is enhanced by carrying more important-data in each packet, which allows the scheme to tolerate higher loss rates.

8.6.1 Sender routine

The EDFCPM sender routine gets sound samples and divides them into some 80 byte length (10ms) segments. A classifier routine classifies segments based on their energy, and consecutive similar segments are joined together to make blocks. As in DFCPM, the unvoiced blocks are packed with their length and power, thus a random noise generator can regenerate them at the receiver. The other blocks are sent by a specific routine in which the block samples are separated to odd-numbered and even-numbered (x_o, x_e) series.

The DFT coefficients of both series are calculated and using a predefined transmission rate, a limit is found which can eliminate some coefficients. As a result, the important coefficients are packed in two packets. In contrast with DFCPM, the EFDCPM routine uses another transmission rate (which is lower than the first one) to generate more compressed coefficients, which are called redundant coefficients. A packet is packed with the main coefficients of the even series and the redundant coefficients of the odd series. Similarly, another packet is packed with the main coefficients of the odd series and the redundant coefficients of the even series. Figure 8.18 shows the structure of a packet in EDFCPM.

The amount of the redundant data depends on the second limit, gained from the second transmission rate as an input parameter. As with DFCPM, order identifiers, maximum real and imaginary values, block length and other packet related components as mentioned in Subsection 8.5.1 must also be added to this scheme.

8.6.2 Receiver routine

A routine for a loss-less and another routine for a lossy environment are explained in this Section.

Receiving both packets

In obtaining both packets $(X_e,\ X_o)$ of a block, the receiver ignores the redundant parts in each packet and regenerates voice based on the method mentioned in Subsection 8.5.2.

Receiving one packet

In this situation, the redundant part of the arrived packet is considered as the coefficients of the lost packet, and the routine, regenerates voice, based on the two series of coefficients. To sum up, EDFCPM uses the same strategy for decoding voice, either when a packet, or when both packets of a block, are received at the receiver. The EFDCPM's active node routine and its threshold calculation are the same as previously mentioned for DFCPM.

8.6.3 Result of EDFCPM

The voice quality of EDFCPM is evaluated in terms of its data rate and its robustness against the packet loss problem.

Quality versus data rate

Voice samples of a native English-speaking woman were packed with different transmission rates for main and redundant parts. In the test, a very lossy network which loses 50% of all packets is examined. Therefore the regenerated voice is based on receiving only the even packets. Figure 8.19 shows the relative quality of the voice versus the two transmission rates.

Quality versus loss rate

Considering three other coding schemes, Figure 8.20 shows the relative quality versus loss rate for EFDCPM with first and second transmission rates set to 80% and 16% respectively.

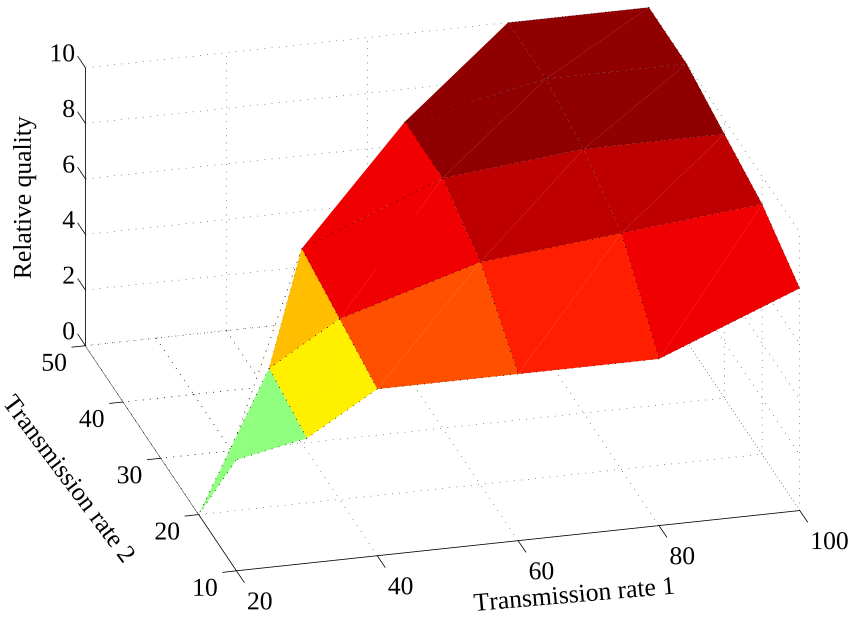

Figure 8.19: EDFCPM quality versus its two transmission rates.

In the tests, the transmission rate of the redundant part was selected as 20% of the transmission rate of the main part. It can be seen that it generates good voice quality even when it loses 50% of its packets.

Processing time

As EDFCPM uses the same packing strategy as DFCPM, its processing time is almost equal to that is measured for DFCPM.

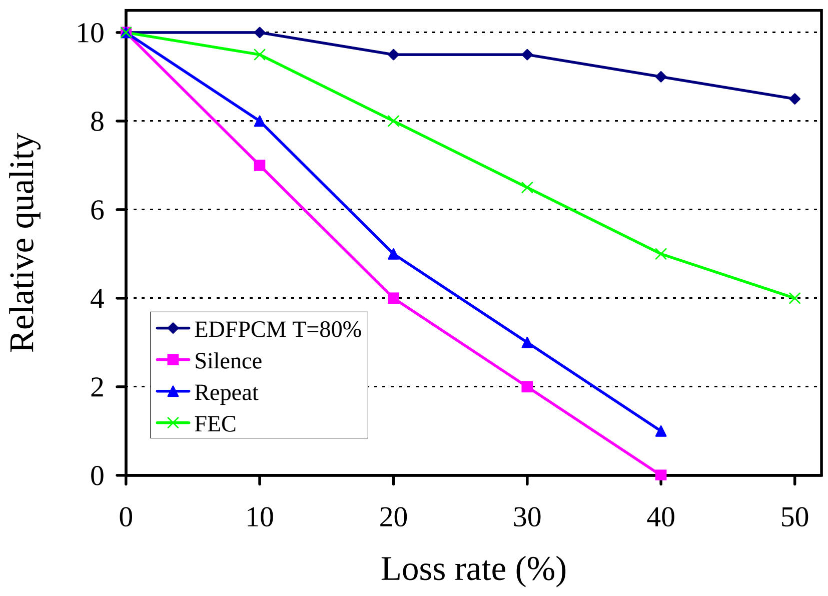

Figure 8.20: Comparison of voice quality of different schemes.

8.7 Summary and conclusion

DFCPM and EDFCPM as two voice-packing schemes can satisfy the two motivation points, which are compression, and lost packet effect compensation. They both work based on packing important DFT coefficients of a voice segment in different packets. Therefore, their packet length varies from packet to packet. Although DFCPM offers bass sound, compared with the original voice, its quality is good enough to be considered as an acceptable voice-coding scheme for VOIP. In contrast with DFCPM that cannot tolerate very high loss rate, EDFCPM can offer good voice quality even with very lossy networks. Comparing the results gained by different coding schemes, it is revealed that EDFCPM is able to generate good voice quality despite losing packets, but its processing time is its major weak point. When an application uses short sized blocks (Figure 8.17) and when it needs an adjustable compression rate (Figure 8.19), EDFCPM is a useful choice.

Chapter 9

Enhanced Code Excited Linear Prediction (ECELP)

This voice-packing scheme sends the extracted parameters of the Code Excited Linear Prediction (CELP) model of a voice segment and regenerates the voice by decoding the parameters at the receiver. This scheme, Enhanced Code Excited Linear Prediction (ECELP) is implemented in three programs with 13.2, 6.4 and 4.75 kbps data rates. This Chapter presents the theory of transmitting and regenerating the parameters. The schemes and the results of simulations are presented subsequently. The first Subsection describes the theory of coding voice with 6.4 kbps data rate.

9.1 Conjugate Structure Algebraic Code Excited Linear Prediction (CS-ACELP)

Taking advantage of high correlation between adjacent samples and considering the fact that speech signals are mostly stationary over relatively short time intervals, hybrid coders try to extract a set of parameters from a speech and regenerate it by decoding the parameters at the receiver [3]. They model speech by a linear filter and they estimate a speech sample $x(n)$ by a linear combination of its previous samples, which is called *linear prediction coding* (LPC) [91] defined by

$$\hat{x}(n) = \sum_{i=1}^{l} a_i.x(n-l). \tag{9.1}$$

A linear prediction filter that assesses the feasibility of determining a sample by the linear combination of its previous samples, is the most important part of LPC. ITU-T recommendation G.729 defined an algorithm for coding speech signals at 8 kbps using CS-ACELP which works based on linear prediction method. In this algorithm, after passing through a 300-3400 Hz band pass filter, the analog voice is sampled at 8 kHz to obtain 16-bit linear PCM codes.

Speech samples are divided into some 80 sample frames corresponding to 10 ms of speech. Each frame is analyzed to extract the parameters of the CELP model [43]. The parameters are encoded and transmitted to the receiver in a bit stream format. The receiver uses these parameters to retrieve the excitation and synthesis filter parameters. Filtering the excitation parameters through a short-term synthesis filter, the speech is regenerated and is sent to pass through a long-term or pitch synthesis filter, which reproduces the speech [96, 56].

9.1.1 Encoder

In G.729 encoder, the locally decoded signal is compared with the original signal. The filter parameters are selected in such a way that they can minimize the mean-square weighted error between the original and regenerated signals [28]. To do this, the input signal is passed through a 140 Hz high-pass filter and scaled in the pre-processing block. A tenth-order linear prediction (LP) analysis is performed once per 10 ms of frame to compute the LP filter coefficients that are quantized in the Line Spectrum Pairs (LSP) domain with 18 bits. Figure 9.1 shows the encoding principle of CELP.

The input frame is divided into two 5 ms subframes. Interpolated LP coefficients are applied to the first subframe and the quantized and unquantized LP filter coefficients are used for the second subframe [50]. The excitation in each subframe is represented by a fixed

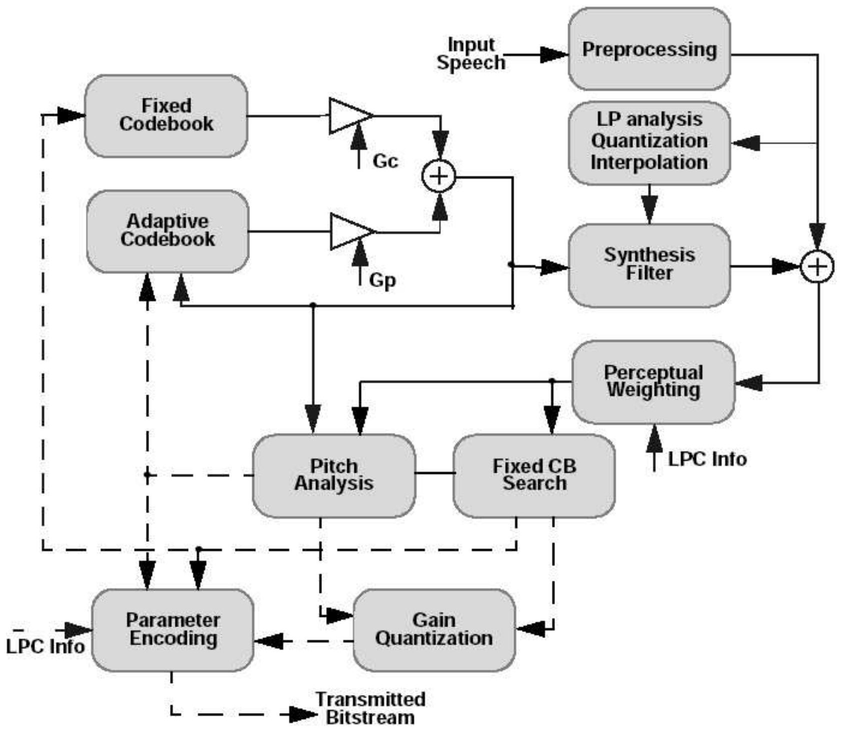

Figure 9.1: CELP encoder block diagram.

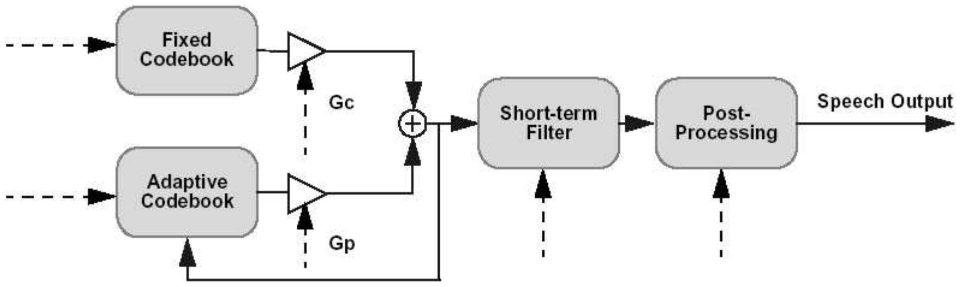

Figure 9.2: CELP decoder block diagram.

and an adaptive *codebook*. The fixed codebook is for unvoiced sounds and the voiced sound is simulated by an adaptive codebook, which represents the periodicity of the sound. The pitch period is encoded with 8 bits in the first subframe and relatively encoded with 4 bits in the second subframe. The fixed codebook is encoded with 17 bits for each subframe.

9.1.2 Decoder

The G.729 decoder principle is shown in Figure 9.2.

Parameter indices are extracted from the received bitstream and decoded to obtain the parameters corresponding to 10 ms of speech frame [28]. These parameters include:

- LSP coefficients,
- two fractional pitch delays,
- two fixed codebook vectors, and
- two sets of adaptive and fixed codebook gains.

The LSP coefficients are interpolated and converted to LP filter coefficients for each subframe. Then for each 5 ms subframe the following steps are done.

- The excitation is constructed by adding the adaptive and fixed codebook vectors scaled by their respective gains.
- The speech is reconstructed by filtering the excitation through the LP synthesis filter.
- The reconstructed speech signal is passed through a post-processing stage, which includes an adaptive post filter based on the long-term and short-term synthesis filter, followed by a high-pass filter and scaling operation.

9.2 ECELP64

This algorithm works based on the ITU-T G.729 annex H recommendation, which is used in some speech transmission systems because of its good quality and low bit rate. The recommendation uses Conjugate-Structure Algebraic-Code-Excited Linear-Prediction (CS-ACELP) algorithm to encode voice samples and it transmits the codes in 6.4 kbps data rate. Analog voice is sampled at 8 kHz and is converted to 16-bit linear PCM codes. The codes are separated into 80 sample size frames. A frame in turn is divided into two 5 ms subframes. Each frame and its two subframes are analyzed to extract the parameters of the CELP model.

The extracted parameters and their lengths are listed as follow.

- From the whole frame:
    - LP filter coefficients first codebook (8 bits), and
    - LP filter coefficients second codebook (10 bits).
- From the first subframe:
    - adaptive codebook (pitch period-8 bits),
    - fixed codebook (9 bits),

  - fixed codebook sign (2 bits), and
  - pitch and fixed codebook gains (6 bits).

- From the second subframe:
  - relative adaptive codebook (pitch period-4 bits),
  - fixed codebook (9 bits),
  - fixed codebook sign (2 bits), and
  - pitch and fixed codebook gains (6 bits).

CELP extracts 10 parameters with the total length of 64 bits and transmits them toward the receiver [50]. Therefore, each 80-byte speech segment is coded with 64 bits obtaining 6.4 kbps total data rate.

If each 64 bit segment is packed in a packet and sent to the receiver, the receiver can easily regenerate the sound using its decoder. In a lossy network, some packets might be dropped and consequently, the voice will have very poor quality. To cope with that problem, Wah and Lin tried to separate a long voice segment to many small size parts and to encode the small parts by the CELP algorithm [106]. They changed the frame size and they packed packets by parameters of different parts of a segment. Although they did not insert redundant data in their packets, they produced lower voice quality compared to CELP even under a loss-less environment because they changed the frame size. In another approach, Yong proposed to pack the LP and pitch parameters into high priority packets and pack the excitation vectors into low priority packets [114]. The weak point of this method is that when a lot of low priority packets are lost, there will be no received excitation vectors; therefore, the receiver cannot decode the voice samples correctly. In contrast with their approaches, ECELP believes that where a frame is lost, it can be reconstructed by important parameters from the original frame and unimportant parameters from the previous (or next) frame.

ECELP64 marks the following parameters as important ones.

- From the whole frame:
  - LP filter coefficients first codebook (8 bits), and
  - MSB bits of the LP filter coefficients second codebook (4 bits).
- From the first subframe:
  - adaptive codebook (pitch period-8 bits), and
  - MSB bits of the fixed codebook (4 bits).

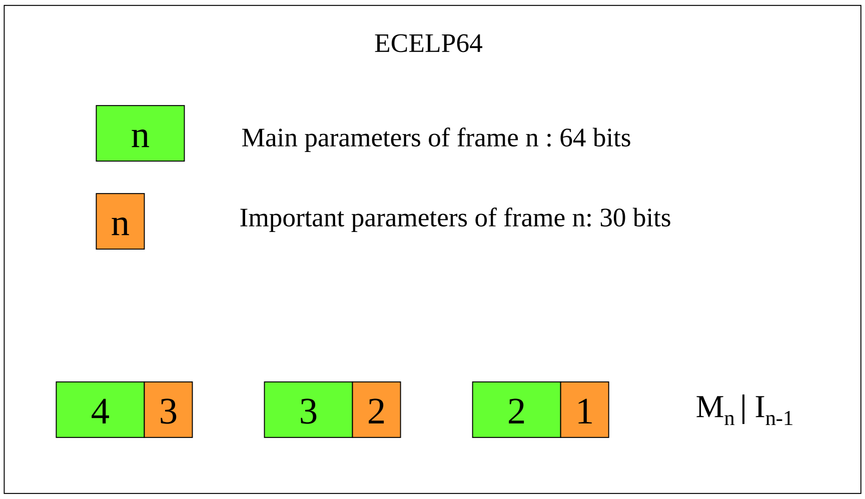
ECELP64
n
Main parameters of frame n : 64 bits
n
Important parameters of frame n: 30 bits
4 3
3 2
2 1
$M_n \mid I_{n-1}$

Figure 9.3: ECELP packet structure in individual loss.

- From the second subframe:
    - pitch and fixed codebook gains (6 bits).

ECELP64 uses the above 30 important bits from each frame and 34 unimportant bits from the next frame to regenerate the lost frame. To do that, the ECELP64 sender routine extracts whole parameters of a frame and inserts them in a packet. It adds the important parameters from the previous frame to this packet and sends the packet toward the receiver. Therefore, each packet has its own whole parameters and the previous packet's important parameters. Figure 9.3 shows packet structure for ECELP64 when the lost packets are individual.

The total data rate will be 9.4 kbps. In a lossy network, a lost packet can easily be regenerated using the important parameters from the next arrived packet and the arrived packet's unimportant parameters.

9.3 ECELP13

This coding scheme works based on the Regular Pulse Excited Long Term Prediction (RPE-LTP) which has been adopted as the speech-coding method in the GSM 06.10 standard. In RPE-LTP, the speech signal is divided into 160 sample frames and is pre-processed to emphases the high-frequency component, partly compensating for its low energy [70]. An eighth-order optimum linear predictor is applied. The reflection coefficients (related to the filter coefficients) are nonlinearly mapped to another set of values called Log-Area Ratio(LAR). The 8 LAR parameters are quantized using 6, 6, 5, 5, 4, 4, 3 and 3 bits. Then the frame is filtered producing the signal $u(n)$. $u(n)$ is then divided into 4 sub-frames (5ms each, 40 samples). Long-term prediction is performed for each sub-frame. The lag is quantized to 7 bits and 2 bits represent the gain. Long-term prediction produces a signal named $e(n)$ which is down-sampled by a factor 3. For each sub-frame, there are 4 down-sample patterns, needing 2 bits to specify the pattern used. The down-sampled $e(n)$ has 13 samples. The majority of them is quantized to 6 bits, others are normalized then represented by 3 bits. Therefore, in each sub-frame, $e(n)$ is represented by 6+13*3 = 6+39 bits. A frame has 4 sub-frames, so in total 4*(6+39) = 180 bits are generated.

The extracted parameters and their length are as follow.

- From the whole frame:
  - 8 LAR coefficients (36 bits).
- From each subframe:
  - pattern code (2 bits),
  - lag (7 bits),
  - gain (2 bits),
  - coded maximum amplitude (6 bits), and
  - normalize RPE samples (3*13 bits).

Each coded frame has 36 bits plus 56 bits for each subframe. Totally, four subframe in a frame makes 260 bits for each 40 ms speech frame, that is 13 kbps data rate.

ECELP13 marks the following parameters as important ones.

- From the whole frame:
  - 8 LAR coefficients (36 bits).
- From each subframe:
  - MSB bit of pattern code (1 bit),
  - MSB bits of lag (5 bits),
  - MSB bit of gain (1 bit),
  - MSB bits of coded maximum amplitude (4 bits), and
  - MSB bits of normalize RPE samples (1*13 bits).

ECELP13 uses the above 132 important bits from each frame and 128 unimportant bits from the next frame to regenerate the lost frame. Therefore, the total data rate will be 19.6 kbps.

9.4 ECELP475

This has the same principle as ECELP64 but it uses the Global System for Mobile communication - Adaptive Multi Rate for Narrow Band (GSM-AMR-NB) speech coder with 4.75 kbps data rate. In this algorithm, each 160 samples make a frame, which generates 17 parameters with the total length of 95 bits. These parameters can be grouped in three categories with 23, 38 and 34 bits length respectively. ECELP475 names 23, 13 and 9 bits from these categories as important parameters and adds these 45 bits from each frame to the next one, so the total transmission rate will be 7 kbps. Apparently, in a lossy network, the more important parameters the ECELP adds, the better voice quality it achieves.

9.5 Redundancy and loss rate

Bolot and his colleagues investigated the relation between the place and the amount of redundant data and its efficiency in recovering the lost packets [8]. Hardman et al. proposed that the placement of redundancies should depend on the network conditions [47]. In a low loss environment, they put redundant data in the next packet and in a high loss situation; they inserted them into the second next packet. ECELP uses the same strategy with more dependency on the loss pattern. Based on the detailed drop rate reported by RTCP, ECELP inserts whole parameters from one frame M_n, plus important parameters from the previous frame I_{n-1} in one packet when losses are individual. If the network suffers frequently from two consecutive packet losses, ECELP makes a packet with M_n and I_{n-3} (upper scheme in Figure 9.4). Having burst loss, ECELP inserts important parameters from $n-5$ and $n-3$ frames to the whole parameters of frame n (lower scheme in Figure 9.4). In this situation, ECELP switches to more than one series of important parameters, so the total transmission rate will be 26.2, 12.4 and 9.25 kbps for ECELP13, ECELP64 and ECELP475 respectively.

Figure 9.4 shows the packet structure for ECELP64 in this situation.

Unlike the other ways of adding low bit rate redundancy to the main voice stream [47], ECELP does not add any delay and it does not need any additional processing power to calculate the important parameters. That is because it has all the previous parameters when it is packing a new packet. Although in the last mentioned packing method, when many consecutive packets are lost, the receiver has to wait for a few packets to reconstruct the lost ones, this method can overcome 1, 2, 3, 4 and 5 consecutive losses. Even in a frequent burst loss environment, it can generate good voice quality.

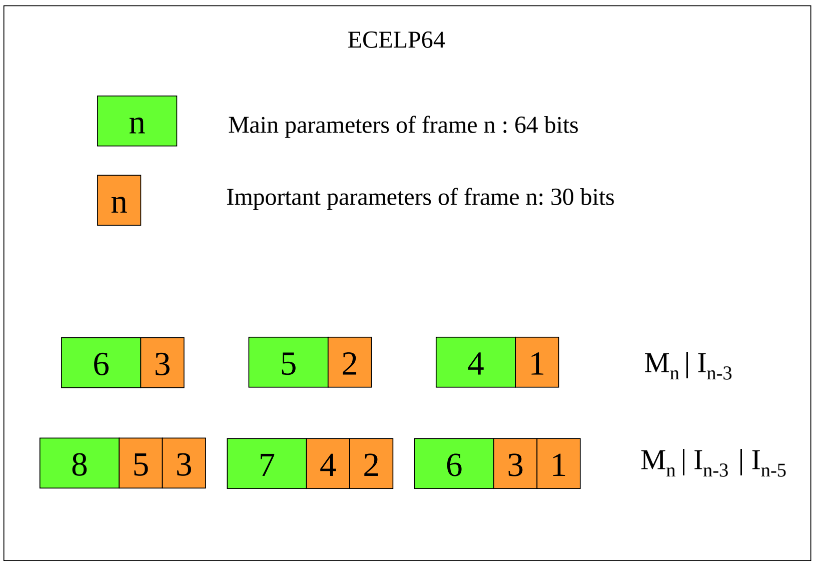
ECELP64
n
Main parameters of frame n : 64 bits
n
Important parameters of frame n: 30 bits
6 3
5 2
4 1
$M_n \mid I_{n-3}$
8 5 3
7 4 2
6 3 1
$M_n \mid I_{n-3} \mid I_{n-5}$

Figure 9.4: ECELP packet structure in burst loss.

9.6 ECELP in active node

In a lossy environment, active nodes play an important role in improving the voice quality with this scheme. Active nodes keep track of packets and detect the lost ones using the sequence number provided by the RTP headers. The nodes can quickly regenerate the lost packets using the important and unimportant parameters from the next arrived packet. The active node processing time is negligible due to the fact that it only copies a few bits to regenerate a lost packet. Although ECELP allows the receiver to regenerate the lost voice samples in the traditional networks, active nodes can help ECELP produce a better voice quality in a very lossy environment. As the active node does not encode and decode the ECELP codes, voice quality degradation due to the tandem-encoding problem is not the case for the simulations.

9.7 Results of ECELP

Voice samples of a native English-speaking woman were packed at the sender (Appendix C). Using NS-2, six schemes were tested to compare the voice quality after passing through the lossy link. They were

- silence substitution (64 kbps),
- packet repetition (64 kbps),

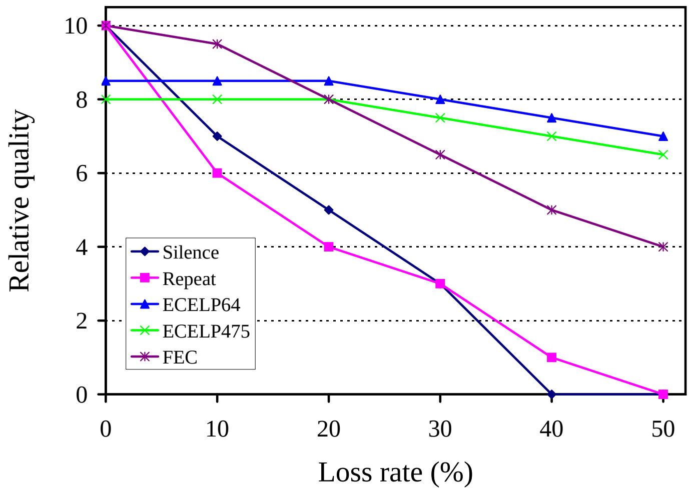

Figure 9.5: Comparison of voice quality of different schemes (individual losses).

- FEC (ADPCM 32 kbps + LPC 6.4 kbps),
- ECELP64 (9.4 kbps),
- ECELP475 (7 kbps), and
- ECELP132 (19.6 kbps).

The performed tests aimed to evaluate ECELP algorithms in terms of their voice quality using different loss patterns. The audience assessed the regenerated voices based on their quality (Appendix C). As the ECELP13 produces almost the same voice quality as ECELP64, its results are not shown in the following figures.

9.7.1 Individual loss

Figure 9.5 shows the result of the tests that were done with different loss rates when the lost packets were individual.

It can be seen that in low loss conditions, the schemes that use 64 kbps data rate have better quality compared with ECELP methods. That is because the CELP 6.4, 4.75 and 13.2 kbps have less voice information (and quality) than PCM 64 kbps. The result implies that the silence substitution and packet repetition schemes cannot cope with more than 10% loss rate. The FEC scheme has better quality than ECELP in low loss rates because it has 32 kbps data in its ADPCM part. FEC generates glitches when its ADPCM and LPC decoded parts are joined, as a result, its quality is lower than the ECELP scheme in higher loss rates. The Figure shows that ECELP schemes can provide quite good quality even when half the packets are lost.

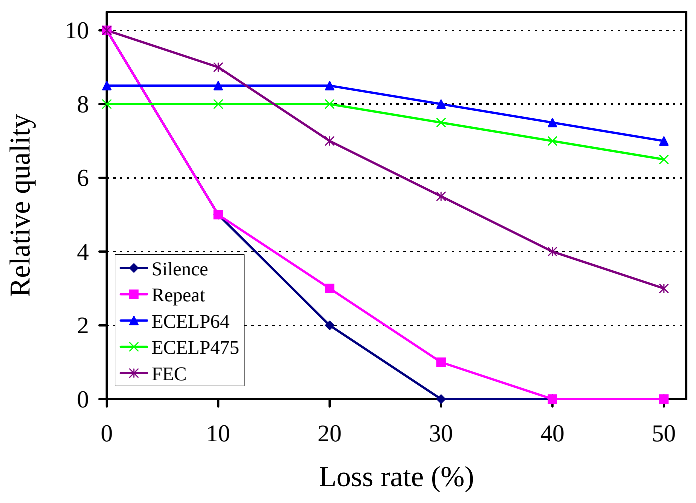

Figure 9.6: Comparison of voice quality of different schemes (burst losses).

9.7.2 Burst loss

In another test, the simulation parameters were adjusted to generate burst losses. In these tests, ECELP methods use more than one series of important parameters.

Figure 9.6 presents the result of the tests. It shows that silence substitution and packet repetition cannot cope with more than 5% burst loss rate. The FEC scheme fails to generate good voice quality for the same reason, as mentioned in the individual loss Subsection.

9.8 Summary and conclusion

ECELP schemes, as three methods for packing voice samples work based on sending extracted parameters of the CELP model of a voice segment. A series of tests evaluated them and the audience assessed the quality of their regenerated voice samples (Appendix C). The tests showed that they produce much better quality compared with other ordinary schemes in a very lossy network. Considering the fact that ECELP has a very low bit rate, it generates reasonable sound quality in very lossy environments. Although ECELP introduces delay in a burst loss environment, it generates good voice quality even when 66% of the packets are lost. These routines can be implemented in active networks, so the active nodes can regenerate some lost packets and insert them into the network.

Comparing the results gained by the tests, it can be concluded that ECELP can be used to transfer voice over the Internet with low bit rate. It generates fixed-sized packets but, if it detects a high burst loss situation (reported by RTCP packets), it switches to make use of more than one series of important samples. Therefore, its packet length switches between two constant values.

Chapter 10

Complimentary Adaptive Differential Pulse Code Modulation (CADPCM)

This voice-packing scheme improves Hardman's algorithm [47] by inserting redundant data based on voice specifications. It uses two voice compression routines to generate voice packets in order to make them robust against lost packets. This Chapter explains the routine and some of its test results.

10.1 CADPCM

This scheme uses Adaptive Differential Pulse Code Modulation (ADPCM) and Global System for Mobile communication - Adaptive Multi Rate for Narrow Band (GSM-AMR-NB) methods to generate voice packets. The ADPCM algorithm converts each 16-bit voice sample to a 4-bit ADPCM code which is not only based on the current input sample, but also based on the previous samples [83]. As is mentioned in the DADPCM scheme, CADPCM inserts three bytes as decoding coefficients in all packets and consequently, each packet can be decoded individually. In AMR-NB, voice samples are separated into 160-sample size frames. A frame in turn is divided into two 10 ms subframes. Each frame and its two subframes are analyzed to extract the parameters of the CELP model [1]. The 17 extracted parameters are coded in 95 bits and are transmitted to the receiver, so the total data rate is 4.75 kbps. CADPCM uses the same packing strategy as Hardman et al. [47] (later used by Podolsky et al. [77]) in individual loss conditions and it uses another strategy to pack a packet in burst loss conditions. It uses ADPCM as the main voice-coding scheme and it uses AMR-NB as a redundant coding scheme.

10.2 Packing scheme

The CADPCM algorithm gets sound samples and processes them in the following steps [24]. In the first step, the algorithm divides sound samples into some 160-byte length (20 ms) segments. Using (5.2), a classifier routine segments are classified based on their energy. Segments with lower energy than a predefined *voice-unvoice-threshold* are classified as *unvoiced* and the rest are classed as *voiced*. The transitions between these two groups are classed as *voiced2unvoiced* and *unvoiced2voiced* segments. Next, it selects a segment and encodes it based on its class.

10.2.1 Unvoiced block

For an unvoiced segment, a very short packet containing its power is generated. At the receiver, a random noise generator can simulate this block if its noise power is adapted to the power of the original unvoiced block. This parameter is mentioned as Noise-par in Table 10.1.

Table 10.1: Packet payload length in CADPCM

Current segment	Previous segment	Main code	Redundant code	Payload length
Voiced	Voiced	ADPCM	AMR-NB	80 + 12
Voiced	Unvoiced	ADPCM	Noise-par	80 + 12
Unvoiced	Unvoiced	Noise-par	-	2
Unvoiced	Voiced2un	Noise-par	-	2
Voiced2un	Voiced	ADPCM	AMR-NB	80 + 12

10.2.2 Voiced and Unvoiced2voiced blocks

As these segments have remarkable impact on the voice quality (Subsection 3.1), they must be packed in such a way that they can be regenerated well if packet losses occur during the transmission of them. So the CADPCM compresses these segments with an ADPCM algorithm to get the main codes, and also compresses them with the AMR-NB scheme to get redundant codes.

10.2.3 Voiced2unvoiced block

This block is not very important and losing this part of sound, does not have serious negative impact on the sound quality. So CADPCM calls the ADPCM routine to compress this part. The CADPCM algorithm packs a packet with the main code of the current segment and the redundant code of its previous segment. Table 10.1 shows the possible packet configurations and their length.

It can be seen that the packets' payload lengths are varied between 2 and 92 bytes.

10.3 Redundancy and loss rate

CADPCM uses the same strategy as ECELP to pack a packet. Based on the detailed drop rate reported by RTCP, CADPCM inserts the main code from one frame M_n, plus the redundant code from the previous frame R_{n-1} in one packet, when losses are individual. If the network frequently suffers from two consecutive packet losses, CADPCM makes a packet with M_n and R_{n-3}. In the case of having three or more consecutive packet losses, CADPCM packs a packet with M_n, R_{n-3} and R_{n-5}. In this situation, CADPCM switches to make use of more than one series of redundant codes. As mentioned in ECELP routines, when many consecutive packets are lost, the receiver has to wait for a few packets to reconstruct the lost ones. This method can overcome 1, 2, 3, 4 and 5 consecutive losses. Even in frequent burst packet loss conditions, it can generate good voice quality.

10.4 CADPCM in active node

Active nodes keep track of packets and detect the lost ones using the sequence number provided by the RTP headers. The nodes can regenerate a lost packet using the main codes from the previously-received packet and redundant data from the currently arrived packet. If the currently arrived packet has no redundant data, CADPCM interprets its redundant part as a noise segment and generates a packet with the last-arrived noise parameters.

10.5 CADPCM at the receiver

CADPCM uses the same strategy as is use of in the active nodes to regenerate the lost voice segments (lost packets) at the receiver, and it sends the regenerated voice to the output stage.

10.6 Result of CADPCM

A native English-speaking woman was selected and her 64000-byte voice samples were packed at the sender (Appendix C). The lossy link discarded some packets and the active node had to regenerate the lost packets. Using NS-2, three schemes were tested to compare their voice quality after passing through the lossy link. They were:

- silence substitutions,
- packet repetitions, and
- CADPCM.

The tests aimed to evaluate CADPCM in terms of its voice quality, data rate and its required processing time. The results are explained in the following subsections.

10.6.1 Voice quality

The audience marked the regenerated voice based on its quality, from zero for unacceptable to ten for very good (Appendix C). The tests were designed to evaluate CADPCM with different loss patterns and loss rates.

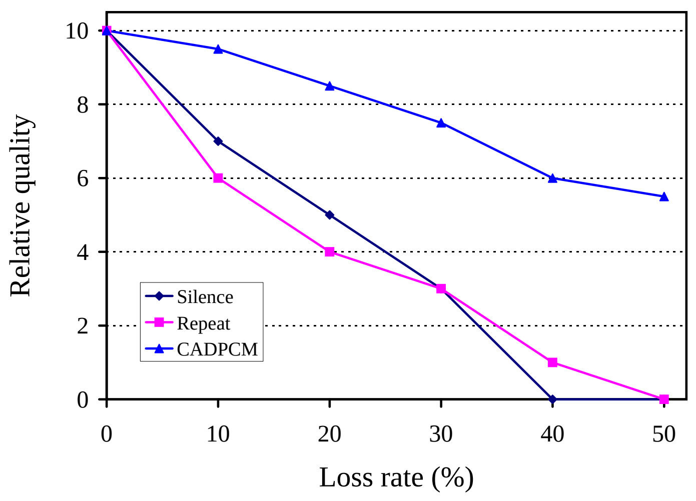

Figure 10.1: Comparison of voice quality of different schemes (individual loss).

Individual loss

Figure 10.1 shows the result of the tests that were done with different loss rates when the lost packets were individual.

The result implies that the silence substitution and packet repetition schemes cannot cope with high loss rate and their related voice quality decreases drastically with higher loss rates. The Figure shows that CADPCM can provide quite good quality even when half the packets are lost.

Burst loss

In another test, the simulation parameters were adjusted to generate burst losses. Namely, when 40% drop rate was selected, the simulator dropped 117 single packets, 13 two consecutive, 6 three consecutive and 1 four consecutive packets out of 400 packets. In these tests, CADPCM uses two series of redundant data, so the payload length varies between 2 and 104.

Figure 10.2 presents the results of the tests. Although CADPCM is delayed in a burst loss environment, it generates good voice quality even when 66% of the packets are lost. To sum up, the performed tests showed that CADPCM produces much better quality compared with the other schemes in a very lossy network.

10.6.2 Data rate

ADPCM has a throughput of 32 kbps and CADPCM adds 4.75 kbps redundant data to each packet, but the total data rate is less than 36.75 kbps. That is because CADPCM does not

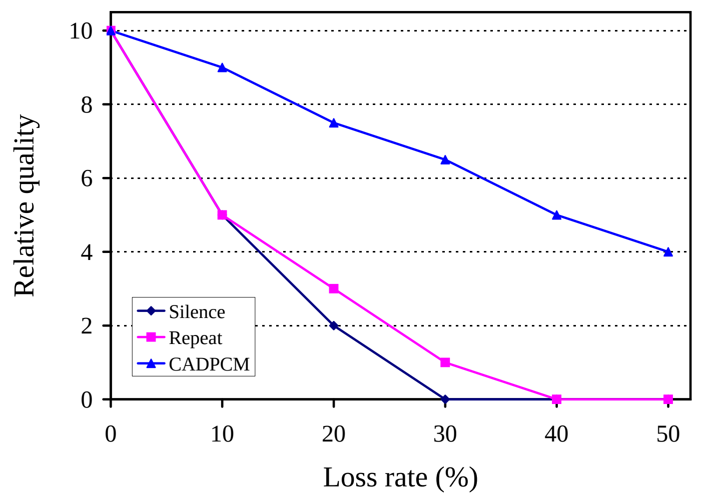

Figure 10.2: Comparison of voice quality of different schemes (burst loss).

send a long packet when the speaker is silent and also, it does not send redundant data when it is not necessary.

In the test with the selected voice samples, it was observed that in 254 cases voiced packets were followed by another voiced packet. The number of voiced packets followed by voice2unvoiced packets was 30, and a total of 56 consecutive unvoiced packets were in the network. The number of unvoiced packets followed by voiced packets was 30, and 27 consecutive voice2unvoiced and unvoiced packets were transmitted. According to the Table 10.1, the total number of transmitted byte was 28754 out of 64000 bytes. Therefore, the transmitted data rate was 28.75 kbps, which is 28% less than 36.75 kbps and 11% less than the standard ADPCM data rate.

10.6.3 Processing time

To evaluate the CADPCM routine, the active node was implemented in a computer with Pentium IV-2.4 GHz as processor and 512-MByte RAM running Windows 2000. The lossy network dropped packets, which had to be regenerated by their related redundant data from the following packets. It was found that each lost packet recovery needs 159 μsec to decode its AMR-NB part and 40.8 μsec to regenerate its ADPCM part. In conclusion, there is the cost of a 200 μsec recovery time for each lost packet.

10.7 Summary and conclusion

A series of evaluation tests made manifest that CADPCM can be used as a voice-packing scheme for VOIP. In tests, it was observed that in spite of inserting redundant data, CADPCM has less data rate than the standard ADPCM and it can easily overcome the packet loss problem in the network. CADPCM as a simple voice-coding scheme works based on packing a packet by ADPCM codes of the current voice segment, plus CELP codes of its previous segment. CADPCM inserts redundant data based on the nature of the speech, therefore, it has a low data rate. Although CADPCM generates glitches when its ADPCM and ECELP decoded parts are joined, considering its data rate, it generates quite good voice quality in high loss rates.

Chapter 11

Conclusion

This dissertation concludes by summarizing the contributions, the methodology and the major results, followed by a future research plan. Voice quality degradation while transmitting voice over the Internet as a lossy packet-switched network is addressed, and many new voice-packing schemes that help VOIP regenerate lost packets and improve the voice quality, are described.

11.1 Contributions

The following items are the principle contributions of this thesis.

- The VOIP voice quality profoundly depends on the number and location of the lost packets. Therefore, packet loss, its pattern and its reasons are studied and many metrics and methods to model the lost packets in the Internet are explained. The measured packet loss shows that the number of consecutive lost packets is much smaller than the number of individual lost packets. Using the results of network measurements the models are evaluated and the Gilbert model is chosen for all simulations (Chapter 2).

- Investigation on the speech properties reveals that speech sounds can be divided into four parts and a loss at the beginning of a voiced segment leads to significant degradation in speech quality. In order to cope with the packet loss problem, many methods are explained and the network and receiver based recovery scheme is selected for simulations. Current available protocols for transmitting voice over the Internet are investigated and RTP/UDP/IP is chosen. As the voice packing schemes need to adjust their voice coding parameters by a feedback on the network quality, RTCP is patched to provide detailed information about the loss pattern (Chapter 3).

- Active network, its concept, its structure and its node specifications are studied and using the tunnelling and the VPN concept, a VAP is introduced and is used for simulations. In order to benefit from an active node, an active packet is introduced and its specifications are explained. In a theoretical aspect, it is mathematically proven that an active node can improve the voice quality in a VAP.

Based on the measured Internet behaviour, a very lossy network is established and an active node is implemented in its structure. The whole VAP is used for all simulations. It contains a voice source, an active node, a voice destination and two ordinary nodes. To have a practical simulation environment, four other nodes are installed that are responsible for generating and handling the background traffic. The VAP has an end-to-end loss model with adjustable parameters (Chapter 4).

- DSPM as a voice-packing scheme based on transmitting uncompressed PCM samples is introduced. It classifies speech samples into four segments and it splits a segment into two streams and finds and inserts important samples of each stream to the other one. To compress the transmitted data, it finds the similar parts in a voiced segment and codes them in such a way that the receiver can easily regenerate the whole voiced segment by the codes. DSPM's voice quality versus packet loss rate and its overhead, data rate and quality versus its coding parameters are investigated (Chapter 5).

- DADPCM, a voice-packing method based on the DSPM principle and ADPCM algorithm is explained. Its ADPCM algorithm compresses each 8 bit voice sample to a 4 bit voice code. As it sends compressed data for each important sample, three different schemes for indicating the location of important codes are considered. DADPCM's voice quality versus packet loss rate and data rate, and also its lost packet recovery time versus its coding parameters, are investigated.

  Different methods to choose important samples that help the schemes improve the regenerated voice quality in lossy environment are explained. They try to introduce minimum data and CPU overhead. It is revealed that the modified fixed distortion method has optimum overheads among the other methods (Chapter 6).

- DADPCM5, a modified version of DADPCM, as a voice-packing scheme suitable for very lossy networks that can overcome burst loss is presented. It splits a voice segment into five streams and finds and inserts the important samples of each stream to its next stream. In order to cope with burst packet loss problem, it reorders the streams and sends them toward the receiver. Its voice quality versus packet loss rate in both individual and burst loss situations and its lost packet recovery time are investigated (Chapter 7).

- DFT of voice segments, its specifications and many performed tests based on transmitting the DFT coefficients of a voice segment are explained. The tests aim to regenerate the voice, based on having some part of its DFT coefficients. The experiments led to the DFCPM voice-packing method. DFCPM is based on sending the DFT coefficients of a voice segment and regenerating the voice by applying the inverse Fourier transform at the receiver. Its quality versus its coding parameters and packet loss rate and its packet processing time are investigated.

  EDFCPM, an enhanced version of DFCPM which carries two series of important data that can improve the voice quality of regenerated lost samples in lossy networks is presented. Its voice quality versus its transmission parameters and loss rate are investigated (Chapter 8).

- ECELP as three voice-packing methods for hybrid codecs that send the extracted parameters of the Linear Prediction model of voice are introduced. ECELP64 uses ITU-T G.729 annex H recommendation. It extracts 30 bits out of each 64 bits and names them as important parameters. It packs a packet by 64 bits from each frame plus 30 bits from its previous frame. Therefore, its total bit rate is 9.4 kbps. ECELP13 and ECELP475 use the same strategy and their total rates are 19.6 and 7 kbps respectively. In order to cope with the burst packet loss problem, they switch to use more than one series of important samples and as a result their data rate increases. Their voice quality versus packet loss rate in both individual and burst loss situations are investigated (Chapter 9).

- CADPCM as an improved version of Hardman's algorithm is introduced. It classifies speech segments and it inserts redundant data based on the voice specifications. It uses more than one series of redundant data in a burst packet loss situation. Its voice quality versus packet loss rate in both individual and burst loss environment are investigated. CADPCM combines waveform and hybrid codecs (Chapter 10).

11.2 Methodology

Many efficient methods for transmitting voice traffic over a lossy packet-switched network are introduced in this monograph. To accomplish that

- packet loss, its models and metrics is studied and the Gilbert model is chosen,
- using NS-2, a very lossy network based on measured Internet behaviour is simulated,

- an active node is implemented in the lossy network,
- voice-packing schemes based on theoretical aspects of voice properties are implemented,
- changing the simulation network conditions, different loss patterns are applied to each scheme, and
- the results are evaluated by subjective tests.

11.3 Major results

Results of regenerated voice show their efficiency in improving the voice quality in a lossy network. The comparative performance of each coding scheme is discussed at the conclusion of each chapter. Table 11.1 shows a summary of all voice-packing schemes and their specifications under 40% individual lost packet.

Table 11.1: Summary of results

Scheme name	Interleaving factor	Individual loss	Burst loss	Data rate (kbps)	Processing time (μsec)	Relative Quality
DSPM	2	Y	N	64 + *Ib* - *C*	Variable	8
DADPCM	2	Y	N	32 + *In*	41 - 49	8
DADPCM5	5	Y	Y	32 + *In*	82 - 115	5.5
DFCPM	2	Y	N	Variable	350 - 625	7
EDFCPM	2	Y	N	Variable	360 - 625	9
ECELP64	-	Y	Y	9.4	160	7.5
ECELP13	-	Y	Y	19.6	160	7.5
ECELP475	-	Y	Y	7	160	7
CEDPCM	-	Y	Y	0.8-36.8	200	6

Ib represents the total amount of the important bytes, *C* represents the aggregate of the compressed bytes, and *In* represents the total amount of the important nibbles and their identifiers. It must be considered that DFCPM and EDFCPM offer shrilled sound compared to other coding schemes. Therefore, they are marked quite highly by the audience.

Many methods for packing voice samples in order to find solutions to lost packet recovery problems for many VOIP applications are introduced.

- DSPM is an effective routine to pack PCM samples in such a way that it can easily recover the lost packets. It introduces a novel way to compress voiced parts of a speech segment. It generates good voice quality even when it loses half of its packets.

- DADPCM and DADPCM5 are effective methods to pack compressed codes in order to overcome the packet loss problem in the Internet. They use ADPCM 32 kbps codes to make packets. DADPCM generates quite good voice quality under 50% lost packets. Using a simple strategy, the DADPCM5 routine can overcome the burst loss problem. Its regenerated voice quality under 40% burst loss shows a remarkable improvement compared to the other available methods.

- DFCPM and EDFCPM are novel ways to pack and transmit voice samples by their DFT coefficients. Their variable qualities and compression rates make them suitable for many applications. For instance, they can be used for broadcasting a message for many users with different available bandwidth in a multi cast session.

- ECELP is a novel way to choose and insert redundant data in very low bit rate voice coders. All of its three implementations show its worthiness in regenerating voice when it loses half of its voice packets. ECELP is suitable for transmitting voice using a low bandwidth over a very lossy network.

- CADPCM improves the methods of choosing and inserting redundant data in ADPCM codes of Hardman's algorithm. It refuses to send a long packet when it is not necessary.

In practical aspect, an application which needs high quality voice can choose DSPM or DADPCM and an application that needs low bit rate can use ECELP as its voice-coding scheme. It has been desired to minimize the amount of redundant data based on quality feedback from the receiver.

To improve the voice quality of VOIP, the introduced methods can be used under many predefined formats. As a data independent FEC scheme, RFC2733 which is engineered for FEC algorithms can be considered [85]. It sends main payload and FEC data in separate streams, so it can easily use the DFCPM, ECELP and CADPCM schemes to recover the lost packets. Although RFC2733 uses FEC independent of the nature of the media, it is designed to transmit some FEC codes that do not require the original media to be sent. Therefore, the other schemes, namely, DSPM, DADPCM and DADPCM5 can be used by this format.

RFC2198 which describes a payload format for use with RTP, is another predefined format that can benefit the introduced schemes. In RFC2198, packets have to carry a primary encoding and one or more redundant encoding data (along with encoding type identifier and packet length identifier) [29]. In short, all of the introduced coding schemes can be used by this RFC.

11.4 Future directions

With regard to the current unreliable packet-switched Internet, several methods are introduced to pack voice samples in such a way that a lost packet can be regenerated. The schemes help VOIP produce good voice quality when a lossy network is used. Assuming that the network does not need to change the voice-packing routine, each of these methods can introduce a specific way to overcome the packet loss problem.

Coping with the tandem-encoding problem in each coding method can be a new research aim in the near future. ”How to choose important parameters so that a tandem-encoder can regenerate good voice quality” is an unanswered question that can be addressed by future work.

Finding a packing scheme for transmitting video over the Internet is an important, as yet unresolved issue for future work. Extracting important parameters for voice as described, means the important parts of image can probably be extracted and packed so that video can tolerate some lost packets on the Internet.

Appendix A

SDLs of DADPCM5

All SDLs related to DADPCM5 are available in the attached CD in post-script format files. They are accessible form D:\SDL\∗.ps address. The following table shows the state and file name of all SDLs.

State	File name	State	File name
Idle	sdl1_1	xxxx3	sdl1_16
0nnnn	sdl1_2	0x4nn	sdl1_17
02nnn	sdl1_3	0xx1n	sdl1_18
024nn	sdl1_4	0xxx3	sdl1_19
0241n	sdl1_5	02x1n	sdl1_20
x2nnn	sdl1_6	02xx3	sdl1_21
x24nn	sdl1_7	x2x13	sdl1_22
x241n	sdl1_8	xx41n	sdl1_23
x24x3	sdl1_9	xx4x3	sdl1_24
024x3	sdl1_10	xxx13	sdl1_25
x2413	sdl1_11	0x41n	sdl1_26
x2x1n	sdl1_12	0x413	sdl1_27
x2xx3	sdl1_13	0xx13	sdl1_28
xx4nn	sdl1_14	0x4x3	sdl1_29
xxx1n	sdl1_15	02x13	sdl1_30

Appendix B

Source codes

All written programs are available in the attached CD in C++ format files. They are accessible form D:\Program\ and they are listed as follow.

- ns-allinone-2.1b9a-gcc32 : The NS-2 installation file that the whole simulator can be installed with.
- active.h : A header file that contains all parameters and constant values for simulation.
- active.cc : A program that contains the C++ codes to define an active node and packet simulation environment.
- sender.cc : A program to encode voice segments and send them towards the receiver.
- activenode.cc : A program to implement the active node in simulation. It contains necessary codes to detect and regenerate lost packets.
- receiver.cc : A program to decode and regenerate voice as a receiver.
- active.tcl : A TCL file to define simulation parameters and define its environment.

Appendix C

Voice samples and test environment

C.1 Voice samples

All regenerated voice samples by different voice packing schemes are available in the attached CD in .WAV format files. They are accessible form D:\voice samples\scheme name\∗.wav address. All files have the following name format (unless otherwise):

$$name\ \gamma\_\alpha\_\beta.wav.$$

γ is either b for burst loss or i for individual loss situation. α is the amount of redundant data that the scheme has applied to the voice sample and β is the loss percentage that the sample has envisaged in the network. The following are the file names and their locations and specifications.

- D:\voice samples\DSPM\
  - DSPM0_0 $\cdots$ DSPM0_50 : DSPM files with 0% overhead and different loss rates.
  - DSPM10_0 $\cdots$ DSPM10_50 : DSPM with 10% overhead and different loss rates.
  - FEC_0 $\cdots$ FEC_50 : FEC files with different loss rates.
  - R_0 $\cdots$ R_50 : Packet repetition scheme with different loss rates.
  - S_0 $\cdots$ S_50 : Silence substitution with different loss rates.
  - ST60_0 $\cdots$ ST60_50 : DSPM with 60% similarity threshold and different loss rates.
- D:\voice samples\DADPCM\
  - D0_0 $\cdots$ D0_50 : DADPCM files with 0% overhead and different loss rates.

- D10_0 $\cdots$ D10_50 : DADPCM with 10% overhead and different loss rates.
- $\cdots$
- FEC_0 $\cdots$ FEC_50 : FEC files with different loss rates.
- R_0 $\cdots$ R_50 : Packet repetition scheme with different loss rates.
- S_0 $\cdots$ S_50 : Silence substitution with different loss rates.

• D:\voice samples\DADPCM5\

- Ci0_0 $\cdots$ Ci0_50 : DADPCM5 files with 0% overhead and different loss rates (individual).
- Cb40_0 $\cdots$ Cb40_50 : DADPCM5 with 40% overhead and different loss rates (burst).
- $\cdots$
- FECi_0 $\cdots$ FECi_50 : FEC files with different loss rates (individual).
- FECb_0 $\cdots$ FECb_50 : FEC files with different loss rates (burst).
- Ri_0 $\cdots$ Ri_50 : Packet repetition scheme with different loss rates (individual).
- Rb_0 $\cdots$ Rb_50 : Packet repetition scheme with different loss rates (burst).
- Si_0 $\cdots$ Si_50 : Silence substitution with different loss rates (individual).
- Sb_0 $\cdots$ Sb_50 : Silence substitution with different loss rates (burst).
- Qb_0 $\cdots$ Qb_50 : DADPCM5 with different overhead and 40% loss rate.

• D:\voice samples\DFCPM\

- P1_20 $\cdots$ P1_100 : DFCPM with different transmission rates and 50% loss rate.
- P2_20 $\cdots$ P2_100 : DFCPM with different transmission rates and no loss.
- T40_0 $\cdots$ T40_50 : DFCPM with 40% transmission rate and different loss rates.
- $\cdots$
- FEC_0 $\cdots$ FEC_50 : FEC files with different loss rates.
- R_0 $\cdots$ R_50 : Packet repetition scheme with different loss rates.
- S_0 $\cdots$ S_50 : Silence substitution with different loss rates.

• D:\voice samples\EDFCPM\

- P100_10 $\cdots$ P100_50 : EDFCPM with 100% different transmission rates and different loss rates.

- T80_16_0 $\cdots$ T80_16_50 : EDFCPM with 80% transmission rate one and 16% transmission rate two and different loss rates.
- $\cdots$
- FEC_0 $\cdots$ FEC_50 : FEC files with different loss rates.
- R_0 $\cdots$ R_50 : Packet repetition scheme with different loss rates.
- S_0 $\cdots$ S_50 : Silence substitution with different loss rates.

- D:\voice samples\ECELP\
  - Di_0 $\cdots$ Di_50 : ECELP files with different loss rates (individual).
  - Db_0 $\cdots$ Db_50 : ECELP files with different loss rates (burst).
  - $\cdots$
  - FECi_0 $\cdots$ FECi_50 : FEC files with different loss rates (individual).
  - FECb_0 $\cdots$ FECb_50 : FEC files with different loss rates (burst).
  - Ri_0 $\cdots$ Ri_50 : Packet repetition scheme with different loss rates (individual).
  - Rb_0 $\cdots$ Rb_50 : Packet repetition scheme with different loss rates (burst).
  - Si_0 $\cdots$ Si_50 : Silence substitution with different loss rates (individual).
  - Sb_0 $\cdots$ Sb_50 : Silence substitution with different loss rates (burst).
- D:\voice samples\CADPCM\
  - CADi_0 $\cdots$ CADi_50 : CADPCM files with different loss rates (individual).
  - CADb_0 $\cdots$ CADb_50 : CADPCM files with different loss rates (burst).
  - $\cdots$
  - Ri_0 $\cdots$ Ri_50 : Packet repetition scheme with different loss rates (individual).
  - Rb_0 $\cdots$ Rb_50 : Packet repetition scheme with different loss rates (burst).
  - Si_0 $\cdots$ Si_50 : Silence substitution with different loss rates (individual).
  - Sb_0 $\cdots$ Sb_50 : Silence substitution with different loss rates (burst).

The main (original) file is named $main.wav$ and is located at D:\voice samples\main.wav. A Matlab code to play and study the voice files is available at D:\voice samples\play.m.

C.2 Test environment

In all voice-packing schemes, a subjective test was carried out to evaluate the quality of voice segments. A 128 kbyte female voice sample with 8 sec duration was chosen as the main voice (PCM 16 bit linear, sampled at 8 kHz). The main voice was packed with different voice-packing schemes and the packets were passed through a lossy network with adjustable loss rate. The regenerated voice samples were played twice using a loudspeaker in a quiet room. In average, twelve non-expert listeners evaluated the overall quality of each regenerated voice samples on a 10 category scale. Their marked results were used for drawing the charts.

Bibliography

[1] 3GPP, *Performance characterization of the amr speech codec*, Tech. report, 3GPP TR 26.975, January 2001.

[2] M. Allman, D. Glover, and L. Sanchez, *Enhancing tcp over satellite channels using standard mechanisms*, Tech. report, Internet Engineering Task Force, RFC2488, 1999.

[3] J. Bellamy, *Digital telephony*, John Wiley Publishers, New York, USA, pp. 123-130, 1982.

[4] S. Biaz and N. Vaidya, *Discriminating congestion losses from wireless losses using inter-arrival times at the receiver*, Proceedings of IEEE Symposium on Application-Specific Systems and Software Engineering and Technology (1999), 10–17.

[5] P. Billingsley, *Statistical inference for markov processes*, University of Chicago Press, 1961.

[6] J. Bolot, *End-to-end packet delay and loss behaviour in the internet*, SIGCOMM 93 (1993), 289–298.

[7] J. Bolot, H. Crepin, and A. Vega-Garcia, *Analysis of audio packet loss in the internet*, NOSSDAV, Network and Operating System Support for Digital Audio and Video, 5th International Workshop (1995), 163–174.

[8] J. Bolot, S. Fosse-Parisis, and D. Towsley, *Adaptive fec based error control for internet telephony*, Proceedings of IEEE INFOCOM, New York, USA (1999), 1453–1460.

[9] J. Bolot and A. Vega-Garcia, *Control mechanisms for packet audio in the internet*, Proceedings of IEEE INFOCOM (1996), 232–239.

[10] ______, *The case for fec-based error control for packet audio in the internet*, ACM Multimedia Systems (1998).

[11] J. Boyce and R. Gaglianello, *Packet loss effects on mpeg video sent over the public internet*, ACM Multimedia (1998), 181190.

[12] P. Brady, *Effects of transmission delay on conversational behaviour on echo-free telephone circuits*, Bell System Technical Journal **50** (1971), 115–134.

[13] S. Bush and K. Kulkarni, *Active network and active network management*, Kluwer Academic/Plenum Publishers, New York, 2001.

[14] K. Calvert, *Architectural framework for active networks*, Active Network Working Group, Draft (1999, Available: http://www.dcs.uky.edu/calvert/archlatest.ps).

[15] B. Carlson, P. Crilly, and J. Rutledge, *Communication systems, an introduction to signals and noise in electrical communication*, fourth ed., McGraw-Hill companies, Inc. 1221 Avenue of the Americas, New York, NY 10020, USA, 2002.

[16] S. Cen, P. Cosman, and G. Voelker, *End-to-end differentiation of congestion and wireless losses*, Proceedings of Multimedia Computing and Networking (MMCN) conference (2002), 1–15.

[17] K. Chin, S. Hui, and S. Foo, *Packet voice recovery techniques for real-time internet voice communication*, Proceedings of 4th Asia-Pacific Conference on Communications/6th Singapore International Conference on Communication systems APCC/ICCS 98 (1998), 122–126.

[18] Y. Chu and H. Chun, *Differentiated service and qos*, TL technical journal **31** (2001), no. 6.

[19] A. Clark, *Modeling the effects of burst packet loss and recency on subjective voice quality*, Proceedings of IP Telephony Workshop (2001).

[20] R. Cohen and G. Kaempfer, *On the cost of virtual private networks*, IEEE/ACM Transactions on Networking **8** (2000), no. 6, 775–784.

[21] D. Comer, *Internetworking with tcp/ip, principles, protocols and architecture*, third ed., vol. 1, Prentice Hall, 1995.

[22] S. Covaci, *The first international working conference on active networks*, Active Networks (1999).

[23] Y. Darmani, L. White, and M. Liebelt, *New methods for lost voice packet recovery in active networks*, IEEE International Conference on Networks, ICON2003 (2003), 57–62.

[24] ______, *A new voice packing scheme to recover the lost packets using active networks*, International Symposium on Telecommunication, IST2003 (2003), 95–99.

[25] B. Dempsey, J. Liebeherr, and A. Weaver, *A new error control scheme for packetized voice over high-speed local area networks*, Proceedings of 18th Conference on Local Computer Networks, Minneapolis (1993).

[26] H. Dudley, *The vocoder*, Tech. Report 17, Bell Labs Record, 122-126, 1939.

[27] A. Feng et al., *Packet spacing: An enabling mechanism for delivering multimedia content in computational grids*, The Journal of Supercomputing **23** (2002), 51–62.

[28] B. Costinescu et al., *Itu-t g.729 implementation on starcore sc140*, Application note AN2094/D, Motorola, Feb. 2001.

[29] C. Perkins et al., *Rfc 2198 - rtp payload for redundant audio data*, Tech. Report RFC2198, Network Working Group, September 1997, Available: http://www.faqs.org/rfcs/rfc2198.html.

[30] G. Fatta et al., *Adaptive routing in active networks*, Proceedings of OpenArch 2000. IEEE Third conference on oopen architecture and network programming, Tel Aviv (2000).

[31] H. Schulzrinne et al., *Rtp: A transport protocol for real-time applications*, Tech. report, Internet Engineering Task Force, RFC 1889, 1999.

[32] K. Calvert et al., *Directions in active networks*, IEEE communication magazine (1998).

[33] S. Bajaj et al., *Virtual internetwork testbed: Status and research agenda*, Tech. report, University of Southern California, 98-678, July 1998.

[34] ______, *Improving simulation for network research*, Tech. report, University ofSouthern California, 99-702, March 1999.

[35] V. Galtier et al., *Expressing meaningful processing requirements among heterogeneous nodes in active network*, Proceedings of the 2nd International Workshop on Software Performance WOSP2000 (2000).

[36] ______, *Predicting and controlling resource usage in a heterogeneous active network*, The 3rd International Workshop on Active Middleware Services (2001).

[37] ETR250, *Etsi speech communication quality for mouth to ear for 3.1 khz handset telephony across networks*, Tech. report, ETSI, 1996.

[38] A. Fernando, *A dynamically updatable active networking architecture*, Ph.D. thesis, Basser department of Computer Science, University of Sydney, Sydney, Australia, October 2001.

[39] S. Floyd and V. Jacobson, *Random early detection gateways for congestion avoidance*, IEEE/ACM Transactions on Networking **1** (1993), no. 4, 397–413.

[40] N. Geckinli and D. Yavaz, *Algorithm for pitch extraction using zero crossing internal sequence*, IEEE Transactions on Acoustics, Speech and Signal Processing **ASSP-25** (1977), no. 6, 559–564.

[41] P. Giacomazzi, S. Giordano, M. Listanti, C. Raffaelli, and F. Ricciato, *Overview of the research results on traffic handling in intserv and diffserv ip networks*, Annual report, MURST Project Techniques for quality of service guarantee in multiservice telecommunication networks, December 1999.

[42] B. Gold and C. Rader, *Digital processing of signals*, Robert Krieger publishing company, Malabar, Florida, ch. 6., 1983.

[43] R. Goldberg and L. Riek, *A practical handbook od speech coders*, CRC Press, Ch. 2-9, 2000.

[44] J. Gruber and L. Strawczynski, *Subjective effects of variable delay and speech clipping in dynamically managed voice systems*, IEEE Transactions on Communications **COM-33** (1985), no. 8, 801–808.

[45] J. Guo, J. Yen, and H. Pai, *A new voice over internet protocol technique with hierarchical data security protection*, IEE Proceedings- Visions, Images, and Signal Processing **149** (2002), no. 4, 237–243.

[46] T. Hacker, B. Noble, and B. Noble, *The effects of systemic packet loss on aggregate tcp flows*, Proceedings of the 2002 ACM/IEEE conference on Supercomputing, Baltimore, Maryland (2002), 1–15.

[47] V. Hardman, M. Sasse, M. Handly, and A. Watson, *Reliable audio for use over the internet*, Proceeding of INET 95 (1995), 27–30.

[48] C. Hoene, I. Carreras, and A. Wolisz, *Voice over ip: Improving the quality over wireless lan by adopting a booster mechanism - an experimental approach*, Proceedings

of SPIE 2001 - Voice Over IP (VoIP) Technology, Denver, Colorado, USA (2001), 157–168.

[49] IETF, *Rfc3006 integrated services in the presence of compressible flows*, Tech. report, Internet Enginnering Task Force, December 2000, Available: http://www.ietf.org/html.charters/OLD/intserv-charter.html.

[50] ITU-T G.729 Recommendation International Telecommunication Union, *Coding of speech at 8kbit/s using conjugate-structure algebraic-code-excited linear prediction (cs-acelp)*, Tech. report, 1996, Available: http://www.ece.cmu.edu/ ece796/documents/g729.pdf (access: 1/12/03).

[51] R. Jain, *A delay based approach for congestion avoidance in interconnected heterogeneos computer networks*, ACM Computer Communication Review **19** (1989), no. 5, 56–71.

[52] N. Jayant, *High quality networking of audio-visual information*, IEEE Communication Magazine **31** (1993), no. 9, 84–95.

[53] N. Jayant and S. Christensen, *Effects of packet losses in waveform coded speech and improvements due to an odd-even sample-interpolation procedure*, IEEE Transactions on Communications **COM-29** (1981), no. 2, 101–109.

[54] W. Jiang and H. Schulzrinne, *Qos measurement of internet real-time multimedia services*, Tech. report, CUCS-015-99, Columbia University, December 1999.

[55] ———, *Modeling of packet loss and delay and their effect on real-time multimedia service quality*, 10th International Workshop on Network and Operating System Support for Digital Audio and Video, Chapel Hill, NC (2000).

[56] H. Kumar and K. Sundaresan, *Implementation of the code excited linear predictive (celp) codec for voip*, Tech. report, State University of New York, USA.

[57] N. Le, H. Sanneck, G. Carle, and T. Hoshi, *Active concealment for internet speech transmission*, Proceedings of the Second International Working Conference on Active Networks, IWAN, Tokyo, Japan (2000).

[58] N. Long Le, *Development of a loss-resilient internet speech transmission method*, Ph.D. thesis, Department of Electrical Engineering, Technical University of Berlin, Berlin, Germany, Ch. 2, 1999, Available: ftp://ftp.fokus.gmd.de/pub/glone/papers/Le9906_Loss-resilient.pdf.gz (access: 1/12/03).

[59] W. Liao, J. Chen, and M. Chen, *Adaptive recovery techniques for real-time audio streams*, IEEE INFOCOM 2001 **2** (2001), 815–823.

[60] D. Lin, *Real-time voice transmission over the internet*, Master's thesis, Department of Electrical Engineering, University of Illinois, Urbana, 1999 Available: http://nms.lcs.mit.edu/ kandula/data/mult_measurements.ps.

[61] J. Liu, I. Matta, and M. Crovella, *End-to-end inference of loss nature in a hybrid wired/wireless environment*, Proceedings of WiOpt'03: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (2003).

[62] V. Markovski, *Simulation and analysis of loss in ip networks*, Master's thesis, School of Engineering Science, Simon Fraser University, October 2000, Available: http://www.ensc.sfu.ca/ ljilja/cnl/pdf/markovski.pdf.

[63] L. Marzegalli, M. Masa, and M. Vitiello, *Adaptive rtp/udp/ip header compression for voip over bluetooth*, European Wireless Conference, 25-28 February 2002.

[64] H. Mattson and G. Solomon, *A new treatment of bose-chaudhuri codes*, Journal of the Society of Industrial and Applied Mathematics (SIAM) **9** (1961), no. 4, 654–669.

[65] N. Maxemchuk and S. Lo, *Measurement and interpretation of voice traffic on the internet*, IEEE International Conference on Communications **1** (1997), 500–507.

[66] C. McGonegal, L. Rabiner, and A. Rosenberg, *A semiautomatic pitch detector (sapd)*, IEEE Transactions on Acoustics, Speech, and Signal Processing **ASSP-23** (1975), no. 6, 570–574.

[67] G. Miller and J. Licklider, *The intelligibility of interrupted speech*, Journal of the Acoustic Society of America **22** (1950), no. 2, 167–173.

[68] D. Mitra, *Network convergence and voip*, Tech. report, TATA Consultancy Services, March 2001-2.

[69] J. Mogul, *Observing tcp dynamics in real networks*, Proceedings of SIGCOMM 92 (1992), 305–317.

[70] M. Mouly and M. Pautet, *The gsm system for mobile communications*, Cell & Sys., 1992.

[71] J. Nagle, *On packet switches with infinite storages*, IEEE Transactions on Communications **35** (1987), no. 4, 435–438.

[72] J. Nonnenmacher, E. Biersack, and D. Towsley, *Parity-based loss recovery for reliable multicast transmission*, IEEE/ACM Transactions on Networking **6** (1998), no. 4.

[73] A. Oppenheim and R. Schafor, *Discrete-time signal processing*, second ed., Prentice Hall, New Jersey, 1998.

[74] C. Padhye, K. Christensen, and W. Moreno, *A new adaptive fec loss control algorithm for voice over ip applications*, Proceedings of IEEE International Performance, Computing and Communication Conference (2000).

[75] V. Paxson, *End-to-end internet packet dynamics*, IEEE/ACM Transaction on Networking **7** (1999), no. 3, 277–292.

[76] C. Perkins, O. Hodson, and V. Hardman, *A survey of packet loss recovery techniques for streaming audio*, IEEE Network **12** (1998), no. 5.

[77] M. Podolsky, C. Romer, and S. McCanne, *Simulation of fec-based error control for packet audio on the internet*, Proceedings of IEEE INFOCOM, San Francisco, California, USA **2** (1998), 505–515.

[78] C. Polyzois, K. Purdy, P. Yang, D. Shrader, H. Sinnreich, F. Mnard, and H. Schulzrinne, *From pots to pans: A commentary on the evolution to internet telephony*, IEEE Network (1999).

[79] J. Proakis and D. Manolakis, *Digital signal processing*, third ed., Prentice Hall, New Jersey, 1996.

[80] L. Rabiner, *On the use of autocorrelation analysis for pitch detection*, IEEE Transactions on Acoustics, Speech, and Signal Processing **ASSP-25** (1977), no. 1, 24–33.

[81] L. Rabiner, M. Cheng, A. Rosenberg, and C. A McGonegal, *A comparative performance study of several pitch detection algorithms*, IEEE Transactions on Acoustics, Speech, and Signal Processing **ASSP-24** (1976), no. 5, 399–418.

[82] I. Reed and G. Solomon, *Polynomial codes over certain finite fields*, Journal of the Society of Industrial and Applied Mathematics (SIAM) **8** (1960), no. 2, 300–304.

[83] Rw. Richey, *Adaptive differential pulse code modulation using pic16/17 microcontrollers*, Tech. Report DS00643A, Microchip Technology Inc., 1996, Available: http://www.microchip.com/download/appnote/pic16/00643b.pdf (access: 1/12/03).

[84] J. Rosenberg, *Reliability enhancements to nevot*, Project report, Columbia University, 1997.

[85] J. Rosenberg and H. Schulzrinne, *An rtp payload format for generic forward error correction*, Tech. Report RFC2733, Network Working Group, December 1999, Available: http://www.faqs.org/rfcs/rfc2733.html.

[86] D. Rubenstein, J. Kurose, and D. Towsley, *Real-time reliable multicast using proactive forward error correction*, Proceedings of NOSSDAV'98, Berlin (1998).

[87] S. Saito and K. Nakata, *Fundamentals of speech signal processing*, Tokyo; New York: Academic Press, 1985.

[88] N. Samaraweera, *Non-congestion packet loss detection fot tcp error recovery using wireless links*, IEE proceedings of Communications **146** (1999), no. 4, 222–230.

[89] N. Samaraweera and G. Fairhurst, *Reinforcement of tcp error recovery for wireless communication*, ACM SIGCOMM Computer Communication Review **28** (1998), no. 2, 30–38.

[90] H. Sanneck, *Concealment of lost speech packets using adaptive packetization*, Proceedings of IEEE International Conference on Multimedia Computing and Systems, Austin, Texas (1998), 140–149.

[91] ______, *Packet loss recovery and control for voice transmission over the internet*, Ph.D. thesis, Department of Electrical Engineering, Technical University of Berlin, Berlin, Germany, 2000, Available: http://sanneck.net/research/publications/thesis/Sann0010_Loss.pdf (access: 1/12/03).

[92] H. Sanneck and G. Carle, *A framework model for packet loss metrics based on loss runlengths*, Proceedings of the SPIE/ACM SIGMM Multimedia Computing and Networking Conference 2000 (MMCN 2000), San Jose, CA, USA (2000), 177–187.

[93] H. Sanneck, N. Le, M. Haardt, and W. Mohr, *Selective packet prioritization for wireless voip*, Fourth International Symposium on Wireless Personal Multimedia Communication, Aalborg, Denmark (2001).

[94] H. Sanneck, A. Stenger, K. Ben Younes, and B. Girod, *A new technique for audio packet loss concealment*, IEEE Global Telecommunications Conference (1996), 48–52.

[95] V. Sawant, *A survey of active networks programming interfaces*, Tech. report, Computer Science Department, University of North Carolina, Chapel Hill, (COMP 391), December 2000.

[96] M. Schroeder and B. Atal, *Code-excited linear prediction (celp): High quality at very low bit rates*, Proceedinds of International Conference on Acoustic, Speech and Signal Processing, ICASSP-85 (1985), 937–940.

[97] Microsoft Windows NT server white paper, *Virtual private networking: An overview*, Tech. report, Microsoft Corporation, June 25 1998.

[98] W. Simpson, *Rfc 1853, ip in ip tunnelling*, Tech. Report 1853, Network Working Group, http://rfc.sunsite.dk/rfc/rfc1853.html, October 1995.

[99] A. Stenger, K. Ben Younes, R. Reng, and B. Girod, *A new error concealment technique for audio transmission with packet loss*, Proceedings of European Signal Processing Conference, EUSIPCO 96, Trieste, Italy (1996).

[100] L. Sun, G. Wade, B. Lines, and E. Ifeachor, *Impact of packet loss location on perceived speech quality*, Proceedings of 2nd IP-Telephony Workshop (IPTEL '01), Columbia University, New York (2001), 114–122.

[101] A. Tanenbaum, *Computer networks*, second ed., Prentice-Hall, New Jersey, 1988.

[102] K. Tarnay, *Protocol specification and testing*, Plenum Press, ch. 4, 1991.

[103] K. Thompson, G. Miller, and R. Wilder, *Wide-area internet traffic patterns and characteristics*, IEEE Network Magazine **11** (1997), no. 6, 10–23.

[104] Y. Tobe, Y. Tamura, A. Molano, S. Ghosh, and H. Tokuda, *Achieving moderate fairness for udp flows by path-status classification*, Proceedings of 25th Annual IEEE Conference on Local Computer Networks (LCN 2000) (Tampa, FL, 2000), 252–261.

[105] B. Wah and D. Lin, *Transformation-based reconstruction for real-time voice transmissions over the internet*, IEEE Transactions on Multimedia **1** (1999), no. 4, 342–351.

[106] ———, *Lsp-based multiple-description coding for real-time low bit-rate voice over ip*, IEEE Transactions on Multimedia (accepted to appear).

[107] B. Wah, D. Lin, and X. Su, *Streaming real-time audio and video data with transformation-based error concealment and reconstruction*, Proceedings of the First International Conference on Web Information Systems Engineering, WISE 2000, IEEE Computer Society **1** (2000), 2–13.

[108] R. Warren, *Auditory perception*, Pergamon Press Inc., 1982.

[109] O. Wasem, D. Goodman, and C. Dvorak, *The effect of waveform substitution on the quality of pcm packet communication*, IEEE Transactions on Acoustics, Speech and Signal processing **36** (1988), no. 3, 342–348.

[110] D. Wetherall, U. Legedza, and J. Guttag, *Introducing new internet service: Why and how*, IEEE Network (1998), no. 3, 12–19.

[111] D. Wetherall and D. Tennenhouse, *The active ip option*, Proceedings of the 7th ACM SIGOPS European Workshop, Connemara, Ireland (1996).

[112] M. Yajnik, J. Kurose, and D. Towsley, *Packet loss correlation in the mbone multicast network*, IEEE Global Internet Conference (1996).

[113] M. Yajnik, S. Moon, J. Kurose, and D. Towsley, *Measurement and modelling of the temporal dependence in packet loss*, Proceedings of IEEE Infocom '99, New York, USA (1999), 345–352.

[114] M. Yong, *Study of voice packet reconstruction methods applied to celp speech coding*, Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP **2** (1992), 125–128.