The Voice Activity Detection (VAD) Recorder and VAD Network Recorder

A thesis presented in partial fulfilment of the requirements

for the degree of

Master of Science in Computer Science at Massey University

Feng Liu

2001

Acknowledgment

First I would like to thank Professor Chris Jesshope, my supervisor, for introducing me to the field of VoIP and computer telephony, for his guidance and comments during the evolution of this project, and especially for his patience on my English writing.

Second I would like to thank my father, HangZhang Liu, for his cultivation and encouragement.

I also would like to thank my colleagues, YongQiu Liu, and Phoebe Wang, for their helps.

Last I would like to thank my wife, Liping Cai. Without her continuous support, this research can never be completed.

Feng Liu, Bs (Bs)

Master of Science (Computer Science) candidate,
Massey University,
Computer Science,
Institute of Information Science and Technology,
Massey Unviersity,
Palmerston North,
New Zealand.

Abstract

The project is to provide a feasibility study for the AudioGraph tool, focusing on two application areas: the VAD (voice activity detector) recorder and the VAD network recorder. The first one achieves a low bit-rate speech recording on the fly, using a GSM compression coder with a simple VAD algorithm; and the second one provides two-way speech over IP, fulfilling echo cancellation with a simplex channel. The latter is required for implementing a synchronous AudioGraph. In the first chapter we introduce the background of this project, specifically, the VoIP technology, the AudioGraph tool, and the VAD algorithms. We also discuss the problems set for this project. The second chapter presents all the relevant techniques in detail, including sound representation, speech-coding schemes, sound file formats, PowerPlant and Macintosh programming issues, and the simple VAD algorithm we have developed. The third chapter discusses the implementation issues, including the systems' objective, architecture, the problems encountered and solutions used. The fourth chapter illustrates the results of the two applications. The user documentations for the applications are given, and after that, we analyse the parameters based on the results. We also present the default settings of the parameters, which could be used in the AudioGraph system. The last chapter provides conclusions and future work.

Table of Contents

List of Acronyms

ABS	:	Analysis-by-Synthesis
ADPCM	:	Adaptive differential PCM
ATM	:	Asynchronous Transfer Mode network
CCS7	:	Common Channel Signalling System number 7
CS-ACELP	:	Conjugate-Structure Algebraic Code Excited Linear Prediction
DSP	:	Digital Signal Processor
ESTI	:	European Telecommunications Standards Institute
ET	:	Energy Threshold
FEC	:	Forward Error Correction
FIFO	:	First In First Out
GAET	:	Geometrically Adaptive Energy Threshold
GSM	:	Global System for Mobile Communication
IP	:	Internet Protocol
ISO	:	Internetworking Operating System
ITU	:	International Telecommunication Union
LCD	:	Liquid Crystal Display
LCR	:	Least-Cost-Routing
LDAP	:	Lightweight Directory Access Protocol
LPC	:	Linear Prediction Coding
LSPE	:	Least-square Periodicity Estimator
MGCP	:	Media Gateway Control Protocol
MPLPC	:	Multi-pulse LPC
QoS	:	Quality of Service
PBX	:	Private Branch Exchange
PCM	:	Pulse Code Modulation
PSTN	:	Public Switched Telephone Network
RELP	:	Residual excited linear predictive coding
RPE-LTP	:	Regular pulse excitation long-term predictor
RSVP	:	Resource Reservation Protocol
RTP	:	Real-time Transport Protocol
SGCP	:	Simple Gateway Control Protocol
SIP	:	Session Initiation Protocol
SNMP	:	Simple Network Management Protocol
SNR	:	Signal-to-Noise Ratio
SS7	:	Signalling System number 7
TCP	:	Transport Control Protocol
UDP	:	User Datagram Protocol
VAD	:	Voice Activity Detection
VoIP	:	Voice over IP
WRED	:	Weighted Random Early Detection
WFQ	:	Weighted Fair Queuing
ZCR	:	Zero Crossing Rate

List of Figures

List of Tables

Chapter 1 Introduction

This project applies a number of speech capture and compression techniques to two application areas. The first is automatic voice capture for and application to record and subsequently playback, on demand and on the web, multimedia teaching material. The second is the ability to send voice over Internet again for educational purposes.

1.1 Background

1.1.1 VoIP

Voice over IP (VoIP) has become one of the fastest-growing technologies in telecommunications since 1995. It challenges the traditional technology of telephony, PSTN, which is the Public Switched Telephone Network and shows the trend towards substituting the PSTN in the future. As a result of the huge market demands, most of all computer vendors have dived into this "golden river" to develop leading edge products, and to provide services that we might have never heard of before, such as offering free toll call to your country from New Zealand. The sparkle behind this fact is that the Internet has been changed to support voice traffic, even though there are some issues of this technology that need to be resolved. But, eventually, the Internet and telephone network will be merged as one and the same [4].

After five years development, there are many of VoIP products around the world. But there is a common problem, which is how the VoIP system can provide the same quality of service (QoS) as PSTNs when the voice and data networks are combined as one. We have not got a perfect solution thus far, although people have been spending a lot of time and money into this research. In the following section, we will explore some of the

techniques used in this technology. We will outline the basic architecture of the VoIP, what advantages it has, and what kinds of issue have arisen when VoIP tries to satisfy the integration of voice and data networks.

Major components of VoIP system

Different vendors of VoIP technology can have their own variations of the overall VoIP network architecture and algorithms, but the backbone of the functionality should be the same for all of them.

VoIP Gateways – VoIP Gateways are a bridge between the local PSTN and the IP endpoint, performing such tasks as preparing data from analog to digital for the network, decompressing digital data back to analog signal when data is received from the network, etc [2]. Some of the VoIP Gateways might do a little bit more than those described above, such as connecting to the destination gateways, having the capability of demodulation and remodulation [4], etc.

VoIP Gatekeepers – VoIP Gatekeepers are used to manage all active clients within a network, used to provide real-time communication. The VoIP Gatekeepers' functionality can be implemented in two ways. One is that it is distributed among all VoIP Gateways and the other is that it is centralised at one or more locations. "When gatekeeper functions are embedded in each gateway, all gateways of the overall VoIP network act autonomously to coordinate their actions. With a centralised gatekeeper, all gateways of the network coordinate their actions with respect to the centralised gatekeeper rather than acting independently" [2].

A gatekeeper is required to perform the following functions [4][5]:

- Address Translation: Alias-address to transport-address translation must be provided.
- Admissions Control: Access to LAN is based on call authorisation, bandwidth or some other criteria.
- Bandwidth Control: Terminals send requests for network bandwidth to the gatekeeper.
- Zone Management: A gatekeeper is required to provide address translation, admissions control, and bandwidth control to all endpoint that have registered with it.

In VoIP it is needless to say, that, the analog voice signals are digitised, compressed and transmitted as a stream of packets over a digital data network, and it is the backbone of the technology. Some vendors may provide support for dynamic bandwidth allocation, packet loss recovery, adaptive echo cancellation, and speech processing to deliver voice quality as high as possible, etc.

In reality, VoIP can be implemented in many forms. The following five forms have covered most of products involved VoIP technology thus far:

- PC to PC service model
- PC to phone service model
- Phone to phone service model
- Network service model
- Service to service provider model

Benefits of VoIP technology

The reason why VoIP has become one of the hottest fields of research and development is that VoIP technology provides more significant benefits than those relevant technologies being widely applied today, such as PSTN. This can be summarised as four aspects described below:

- Cost reduction: The Public Switched Telephone Networks' toll services can be bypassed using the Internet backbone [4], which means slashing prices of the cost of data communication, such as the long distance calls and long distance fax.

- Integration of Voice and Data: The integration of voice and data traffic will be demanded by multi application software such as international telephone service provider or multimedia real-time education system, etc.

- Simplification: An integrated infrastructure, which covers several mature infrastructures and which supports all forms of communication, allows more standardization and simplifies equipment management [4]. The result is a fault tolerant design, and finally, the use of the same telephone line for voice and data will become a realization although it will take time.

- Bandwidth Saving and Network Efficiency: Various good compression solutions reduce the bandwidth demand, and data packetized over IP also releases the resource while voice is not being produced during the real-time communication, e.g. one is listening and not talking. In other words, reducing bandwidth increases network efficiency. Typically, some 50% of a conversation is silence. This provides a huge opportunity to remove the redundancy like silence in certain speech patterns so as to reduce the bandwidth. Therefore, the network efficiency can be increased significantly.

If the benefits described above are so attractive, why do we not directly move forward to fully adopt this technology? The answer is very straightforward -that is to say- there are many issues that need to be resolved, which will have a great impact on the performance of systems based on VoIP technology. This must be solved before we can move any further.

Issues

VoIP technologies are based on the infrastructure of the Internet network, but the original design of Internet network did not support real-time operations. Obviously, there is already a big puddle, which needs to be buried, so that we can walk through the Internet safely.

The main criterion for a successful real-time voice application is that the application can provide at least the same quality of voice when compared to results produced by PSTNs. So seven issues that may affect the result of quality of voice require considerable attention, and will be discussed respectively below:

The first issue is delay. Delay will be divided into two categories according to its predicability. Category one, including processing delay, buffering delay, delay of analog data converted to digital, delay of queuing for being sent or received, delay of digital data converted back to analog for playback, delay jitter and playback buffering, can be minimised under some good algorithms. Category two includes the delay of data across the IP network and private

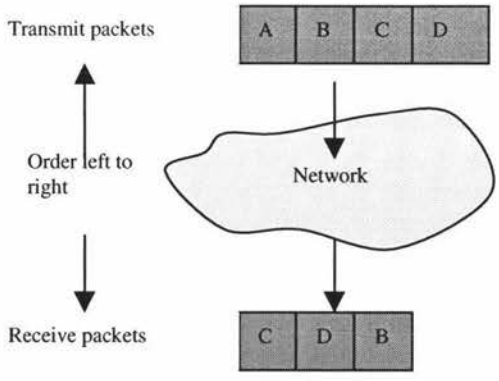


Figure 1.1.1-1 Delay jitter

networks and may not be predicable.

Within category one, most delays are caused by very straightforward reasons, so here we will not consider these easily understood delays. But delay jitter is more complicated and it is worth saying something about it.

Jitter delay is where the data across the network might not arrive in the same order as it was sent. A network does not act like a FIFO stack, where the element that comes in first will be the one that goes out first, so packets across the network will arrive in any order; In the worst case, some of the packets might be lost. Figure 1.1.1-1 shows the situation, where the packet A has been lost.

This kind of delay is really hard to handle for real-time applications. A common solution from most vendors is to use a playback buffer (adaptive jitter buffer) to tolerant this kind of delay. This is achieved by providing extra information in each packet such as time-stamping information, sequencing information, etc. At some stage if a packet such as the packet A in Figure 1.1.1-1 does not arrive we discard it and continue. This is based on the assumption that all packets are very small, so that discarding a packet would not cause serious problem during the speech reconstruction. But the solution also increases the sum of delay because of the buffering technique.

When the sum of the overall delay is greater than 250ms, real-time applications do not work quite so well and are unable to produce the same quality of sound as PSTNs. In order to remove stepping from the voice, which comes from the other end, people must avoid speaking at the same time at both ends. That is the reason why most IP phone vendors choose a protocol, which allows people to either speak or to listen at any one time.

As for category two, the delay comes from the network, such as that caused by congestion. There are some ways to reduce this kind of delay in order to achieve toll-

quality voice effects. Firstly, using an IP packet segmentation technique we can remove the delay caused by very large data packets. Secondly, prioritizing IP packets allows the network to maintain the highest voice quality over a congested network. Thirdly, using a Digital Signal Processor (DSP) architecture we can achieve high performance, especially when we need to apply some sophisticate algorithms, such as Weighted Random Early Detection (WRED), to reduce the delay caused by a congested network.

The second issue is the compression technique. As is well known, codec stands for encoder and decoder. The codec chosen for VoIP applications must provide good quality of voice and require as small a bit rate for transmission as possible.

To date, Pulse Code Modulation (PCM), which is a waveform coding technique based on a three-step process: sampling, quantisation, and coding, (at desired sample rate) is the most commonly used codec on a worldwide basic. Under PCM, an analog signal is sampled at say 8000 times per second. For 8 bit samples, this requires a transmission rate of 8000 samples per second $\times$ 8 bits/sample, or 64 Kbps. Obviously the PCM format does provide good quality of voice. However, it requires a large transmission rate. This means that the delay associated with transmission will be quite significant especially in the time sensitive case. Therefor we must choose another codec, which requires a smaller bit rate with transmission, to reduce the delay while keeping the quality of the voice as good as the quality provided by PCM.

The International Telecommunication Union's (ITU) officially recommended codec for all wide area networking applications is G.729, Conjugate-Structure Algebraic Code Excited Linear Prediction (CS-ACELP) (described in section 2.2) [3][7][8]. This codec uses a sample rate of 8000 samples per second. It also uses a 10-ms frame size plus a 5-ms look-ahead. To analyze the data properly it is sometimes necessary to analyze data beyond the frame boundary, and this is referred to as look-ahead. This results in a total of a 15-ms algorithmic delay (a delay results from buffering a frame's worth of data to analyze the speech), and provides near-toll quality. It only requires approximately 8 Kbps transmission rate. Recently some vendors have added a proprietary silence suppression

capability to the G.729 coding mechanism that reduces the demand of bandwidth required in a conversation down to 4 Kbps.

The third issue is echo cancellation. "In a traditional telephony network, echo is normally caused by a mismatch in impedance from the four-wire network switch conversion to the two-wire local loop and controlled by echo cancellers" [6]. But in voice-packet based applications such as multimedia applications, the playback of the voice received through the network might have a chance to be recorded as input again. If so, the background noise would be rapidly increased, and echo cancellation becomes a critical issue in that kind of application. Common solutions for echo cancellation are to use echo cancellers or to use protocols to ensure echo suppression. The Echo cancellers are built into the low bit-rate codecs and are operated on each DSP. The protocols for echo cancellation are widely used on commercial products nowadays such as IP phones.

The fourth issue is VoIP forward error correction. Most of the VoIP applications choose UDP as its IP protocol in order to satisfy the real-time demands. The disadvantage of using UDP/IP is that there is no guarantee that the destination end will receive all data sent by the source end. So data corruption and loss while transmitting through the network will need to be compensated for in order to obtain good quality voice. VoIP forward error correction (FEC) does this for us. Usually there are two kinds of FEC, one is Intra Packet FEC and the other is Extra Packet FEC.

- With Intra Packet FEC, extra bits are added into the packet so that the receiving end can determine whether the data received is correct at playback.

- With Extra Packet FEC, extra information is added to each packet that allows the receiving gateway to extrapolate from the previously received good packet and to reconstruct the missing or severely corrupted packet.

The fifth issue is bandwidth consumption. As is well known, receiving or sending data through a modem is the bottleneck of the Internet infrastructure. To reduce bandwidth demand so as to maximise the use of modem is one of our goals. Although choosing good codecs can significantly reduce the bandwidth requirement, silence suppression also can achieve that goal. Removing silence from a speech or even from words can reduce the bandwidth demand in order to get the maximum performance of the modem. The most commonly used technique to suppress silence within a conversation is called voice activity detection (VAD), which will be discussed later in this report. After having used VAD, when the sound is reconstructed, there might be a need to smooth the sound in order to make it sound as natural as it was. So comfort noise generation (CNG) is also required.

The sixth issue is IP protocol. The UDP/IP protocol is widely used in time-sensitive products, because the TCP/IP protocol will always retransmit the corrupted packets and this causes additional delays. Most of the VoIP products are real-time applications, which could not stand the delays caused by retransmitting. But in some applications, which are not time-sensitive, the TCP/IP protocol would be the better alternative, such as a Fax application [4]. However, UDP is only a best-effort protocol so that it does not provide reliable service. This could results in a situation that when a person is speaking, you cannot hear his/her speech in its original order. So another application layer level protocol such as RTP (discuss in section 2.3.2) could be employed. The RTP protocol provides mechanisms that can detect the loss of packets, and provide sufficient information for reconstructing the speech in its original order.

The final issue is security. To reduce cost and provide higher productivity, VoIP is widely integrated with the Internet. Therefore, security issue becomes critical. The issue involves access control, authentication, and encryption with respect to transmission over a public packet network [8].

Summary

VoIP is still in the relatively early stage of deployment, but the economics are compelling because of its significant bandwidth efficiencies over traditional PSTN. Unfortunately the toll quality provided by this technology depends on the implementation. Clearly, there is a need to standardise this technology. No matter when the standard shows up, VoIP technology is becoming one the fastest-developed technologies towards the rapid transition of all networks based on the digital/packet-based architecture.

1.1.2 Commercial VoIP Products Reviews

Only five years have passed since the first commercial VoIP product was introduced; yet VoIP technology is being widely used to build up commercial products. Many kinds of products have flooded the market. In this section, let us review some typical commercial products used in different areas of VoIP and trace the development of different kinds of solutions that are being applied to those issue mentioned on the last section by commercial vendors. According to the functionality of the VoIP products the type of products can be categorised as Gateways, Gatekeepers, IP phones, VoIP related products and PC based software phones.

Gateways

- **Cisco AS5300/Voice Gateway**

This product earns product of the Year 2000 Award from *Network Magazine* and it was the only product awarded in the VoIP category [32]. The product represents Cisco's mature technology in VoIP area.

With Cisco AS5300/Vocie Gateway, users can easily interface with PSTN or Private Branch Exchange (PBX) and billing servers. Moreover, this gateway also supports the full range of available high-compression/low-delay ITU recommended Codecs, such as G.711 [8], G.729, G.729a [7], and G.723.1 (in section 2.2). In addition to being a H.323 compliant, which is a standard that defines LAN-based (local area network) video conferencing, and enables multi-vendor interoperability [26], this makes it easily inter-operate with other vendors' H.323 gatekeepers.

The Cisco AS5300/Vocie gateway also has advanced features that make it become the industry-leading gateway. In order to provide toll-quality services, Cisco's

solutions that result in a leading-edge gateway are to provide extra support including voice feature cards and voice-enabled Cisco Internetworking Operating System (IOS) software. The extra supports enable the gateway to support carrier-class VoIP and fax over IP services.

All of the voice feature cards are coprocessor cards, each of which has a powerful Reduced Instructions Set Computer (RISC) engine and dedicated, high-performance (100 MIPS) digital signal processors (DSPs). This coprocessor design minimises delay and packet loss during the voice encoding and packetisation process. More specifically, sophisticated techniques are implemented on the gateways and backbone routing infrastructure so as to provide a low-latency, high-reliability path for sensitive voice traffic through today's networks.

Let us have a close look at these sophisticated techniques now. Firstly, the IP Precedence technique that sets up a priority rate for different types of data (which will cross the IP network, and makes sure voice packet will have the highest priority) is adopted. This assures the highest voice quality data across congested networks. Secondly, in order to minimise the jitter delay, the Resource Reservation Protocol (RSVP) [21] has been implemented as the role of allowing the gateway to request/ reserve the required bandwidth for a call. Thirdly, in order to ensure transparent delivery of toll-quality voice and fax, Weighted Fair Queuing (WFQ) [20] has been implemented as well, providing the capabilities that reserve appropriate bandwidth and prioritise voice and fax traffic. Fourthly, the Weighted Random Early Detection (WRED) [18][19] technique has been implemented. It drops packets selectively based on IP precedence. More clearly, packets with a higher IP precedence are less likely to be dropped than packets with a lower precedence. Therefore, it guarantees that the voice packets, which are higher priority traffic, are delivered with a higher probability than data packets, which are lower priority traffic. Finally, for reducing large data packets into small packets and preventing voice packets from being stuck behind large packets, Multi-class Multi-link PPP (MP) Fragmentation and Interleaving techniques have been implemented.

Furthermore, the gateway adopts both echo cancellation and VAD techniques. Within the gateway, echo cancellation is done within the circuit-switched network with a trial of up to 32 ms; and the VAD technique is used to further reduce the bandwidth requirements. It guarantees that only the useful speech, not including silence, will be packetised and transmitted across the network.

With respect to jitter delay, the gateway provides not only the RSVP solution to minimise it, but also an adaptive jitter buffer to balance delay and packet loss through the gateway itself for maximum call clarity and quality.

The Cisco's IOS software offers a powerful array of quality-of-service (QoS) mechanisms, variable frame sizing, and standards-based H.323 controls, which provide industry-leading voice quality and call control routing to deliver enhanced services.

As for security, a critical issue for a commercial product, the gateway provides voice prompts and digit collection to authenticate the user and identify the call destination via an integrated Interactive Voice Response (IVR) application.

So far, we have seen how Cisco's solutions achieve QoS. We can now surmise that VAD technique and low bit-rate audio codecs are crucial for gateways on today's network infrastructures.

Gatekeepers

Commercial products called gatekeepers will now be discussed. The direction that gatekeepers are taking is worth discussing because the development of most of gatekeepers show that they converging--that is—their functionality is being embedded into gateways.

- **VocalTec Gatekeeper**

VocalTec Gatekeeper [30] provides centralised addressing, security, accounting, and database management for IP telephony networks.

Within the gatekeeper, Dialling Plan Management, which is a flexible and rule-based application, is provided to ensure full control over call routing to all VocalTec Telephony Gateways. Routing can be configured through permission, restrictions and hours of service. Least Cost Routing is supported, by assigning priorities to termination gateways. Load Balancing ensures even distribution of call load between available gateways. Moreover, for network Security, the gatekeeper authenticates user ID/passwords and authenticates users who want to access the IP telephony system. Cryptographic access tokens allow secured control to network elements in compliance with the International Telecommunication Union (ITU) H.235 standard, which is a network security protocol. Furthermore, the gatekeeper provides a centralised open interface for all Call Detail Records (CDR), enabling credit and debit mode billing.

In addition to inter-domain capability, the network manager can establish a gatekeeper hierarchy for networks managed by separate organizations (domains). Each gatekeeper defines its own view of the network and communicates with other gatekeepers when necessary to contact a destination outside its span.

As for standards compliance, the gatekeeper is fully compliant with the ITU H.323 standard. It implements the Registration, Admission, and Status (RAS) version 2, which is a protocol within the H.323, and which performs registration, admissions, status, and disengage procedures between the H.323 VoIP gateway and the H.323 VoIP gatekeeper. Also the gatekeeper is compliant with the Simple Network Management Protocol (SNMP) [13] for remote management of workstations on the network.

After all, there is one more unique feature - that is - database management and backup is supported through an Oracle database.

- **Future of gatekeepers**

The gatekeeper is one of the key components of VoIP, and the trend is that its functionality is being embedded in the whole VoIP system as shown by the gatekeepers' commercial vendors. The following is taken from David Essex's paper [31].

"Nearly all vendors in the IP telephony market, both server-based gateway vendors and stand-alone appliance vendors, are building gatekeeper functionality into their products." The reason why they are doing so is that without a gatekeeper, incoming calls that lack a destination IP address can not be routed to H.323 endpoint. However, this feature is still in the early stages of deployment.

Elemedia's GK2000S [33] is a gatekeeper platform toolkit, used to build gatekeeper functionality into a wide range of hardware and software products. "It is designed to be modular and flexible, supplementing typical gatekeeper functions with a Lightweight Directory Access Protocol [12] (LDAP)-based authenticator and router. The GK2000S (also known at the H.323 Gatekeeper Platform) runs on Sun Solaris and Windows NT."

"Ericsson's H.323 Gatekeeper System is optimised for carrier-class networks. It adds a Least-Cost-Routing (LCR) [34] server to standard gatekeeper functions, such as security, authentication, and maintenance of end-user profiles."

"RADVision's GK-323 [35] gatekeeper includes such enhancements as support for multiple gatekeeper zones, control of resource use, network topology-based call

routing, and PBX-like call processing, such as transferring, forwarding, and line hunting. It works with Windows, Unix, and the VxWorks and pSOS real-time operating systems for embedded systems."

A proposed standard, Simple Gateway Control Protocol (SGCP) [11], is applied to eventually make dedicated gatekeepers embedded into gateways by giving gateways the functionality they need to route calls and data among themselves. SGCP scatters gatekeeper functions into each gateway in a network via call agents.

IP phones

- **Cisco IP phones**

The Cisco IP Phone 7960 [22] voice instrument is a second-generation full-featured IP phone primarily designed for executives and managers. It provides six programmable line/feature buttons and four interactive soft keys that guide a user through call features and functions. The 7960 also features a large pixel-based LCD (liquid crystal display) display. The display provides features such as date and time, calling party name, calling party number, and digit dialled. The display also provides feature and line status, speaker (hands free) and headset features, and a mute button, which controls speaker or handset or headset microphones.

VoIP related products

- **Texas Instruments (TI) IP phone Chipset**

TI has earned two prestigious "Product of the Year" awards from *Internet Telephony* and *Communications Solutions (formerly CTI)* magazines for the TI IP Phone chipset [14].

The TI IP Phone chipset consists of a DSP-based processor chip and a codec chip. The processor uses TI's low-power TMS320C54x DSP core integrated with a RISC microprocessor, and includes an integrated dual port 10/100 Mbps Ethernet switch and all other necessary peripherals for an IP Phone. The codec integrates wide-band analog-to-digital and digital-to-analog converters, with microphone, speaker and handset amplifier interfaces and comes in both single- and dual-channel versions. This high level of integration significantly reduces the component cost and component count in an IP Phone.

TI is the first vendor to provide IP phone manufacturers with all the system software and support needed to enable quick and easy development of standards-based IP Phones with industry leading voice quality. Not only will the bundled software consist of all required DSP and microprocessor software and all embedded microprocessor software including TCP/IP, MGCP (Media Gateway Control Protocol) [10], SIP (Session Initiation Protocol) [9], and H.323 protocols, but does it include low-bit rate and wide-band vocoders, echo cancellation and signalling software [23].

- **Motorola's Vanguard 6560 Multimedia Access device**

Motorola's Vanguard 6560 Multimedia Access device [17] is an expandable network access and concentration platform that integrates legacy, LAN, analog/digital voice and future multimedia traffic. It features High Performing Dual Core Routing and Bridging, which results in low response times, bandwidth efficiency, quality voice transmission, and Multimedia transport capability. More specifically, it offers voice support by an integration of voice with data traffic. 8/16 Kbps compression minimises the network band requirement. Whilst, it provides support for analog and digital voice

port connections. Furthermore, it offers the widest range of protocol support, integral payload data compression, as it consolidates WAN networks, minimises data communications equipment costs, and capitalises on the lowest available WAN tariffs.

- **Lucent Softswitch**

The Lucent Softswitch [16] is a programmable, multi-protocol software system that allows intercommunication between different signalling systems, such as ITU-T Signalling System (SS7), SIP, and Common Channel Signalling System (CCS7) [28].

The basic approach of Lucent Softswitch is premised on the unbundling of the core functionality of a conventional circuit switch and the distribution of this functionality across the backbone of a packet network in the form of software components that run on commercial standard computers. Thus, the system begins by providing the same functionality as a circuit switch, but it does so in packet-based networks designed to interconnect with PSTNs.

It is a scalable, distributed software system that is independent of a specific underlying hardware/operating system and is capable of handling a variety of synchronous communication protocols. Clearly, it can be run on commercial computers and operating systems, supporting multi-protocol, such as circuit-based PSTN, packet-based ATM and IP protocols, and providing open application programming interfaces for third-party developers to create next-generation services. Whilst, it can scale from very small networks to very large networks, and has the capability for the evolving synchronous communication control network to support diverse back-office systems including billing, network management, and other operation support systems.

PC based software phones

- **Microsoft NetMeeting**

Microsoft's NetMeeting [24] provides the most complete conferencing solution for communications with both audio and video. More clearly, Communication with the NetMeeting can occur on a variety of levels, including full or half-duplex audio, video-based conferencing (compliant with the ITU H.323 standard), whiteboard conferencing, application sharing, and text-based chatting. Furthermore, NetMeeting allows for files and sound clips to be transferred between users [27].

In addition to application sharing, NetMeeting is first and foremost a conferencing application, allowing groups of users (regardless of location) to work together efficiently and productively [27].

- **CU seeMe Pro**

CUseeMe Pro [25] is video collaboration software, enabling real-time interaction over the Internet or Intranet using IP-based networking technology. It is fully standards-compliant via H.323, easily connecting to other video desktop including Microsoft NetMeeting, Intel Proshare Video System, Intel TeamStation, and PictureTel LiveLan, and features full-color video, audio, application sharing and whiteboard. Now CUseeMe users can connect to other video desktops such as Microsoft netmeeting, and etc. It is the only group conferencing client on the market today that allows you to see up to 12 video windows simultaneously.

The two products above are both H.323 compliant. Within H.323, the speech codec, which is compulsory, is G.711, and others are optional such as G.722, G.723, G.728, and G.719. Now most of the H.323 terminals support a G.723.1 codec, which is much more

efficient and produces good quality at bit rate around 5.3 to 6.3 kbps. In this project, we use GSM 6.10 codec instead of the G.723.1 codec. Details of different ITU's recommendations and the reason why we do not use G.723.1 will be discussed in chapter 2 of this report. But this kind of VoIP products shows a convergence that they are all H.323 compliant. So the AudioGraph should move towards this direction if it is going to become a synchronous education tool.

Summary

Most of vendors are putting their efforts to solve those issues affecting QoS, unfortunately there has not been a standard yet. As a consequence of having faster machines, DSP architectures are being widely applied to the VoIP area. Therefore, more sophisticated methods can be used to improve the QoS and to manage the network loading. By complying to H.323, most vendors' VoIP product can interoperate, allowing users to communicate without concern for compatibility. The evolution of gatekeepers shows us a trend that its functionality is being moved down to gateways. Meanwhile, some vendors are trying to provide intercommunication between existed network infrastructures.

1.1.3 The Audio Graph tool

The AudioGraph was developed collaboratively by Massey University (NZ) and Surrey University (UK). Professor C. Jesshope has given all details about the AudioGraph in his papers [37][39][40][41]. In this section, let us briefly look at what the AudioGraph system is (The information about AudioGraph has been summarised from C. Jessphope's papers).

The AudioGraph is a toolset for generating and playing multimedia presentations on the World Wide Web. The design goal of the system is to provide simple and easy to use tools for academics, rather than for multimedia and technology professionals, and to produce and delivery course-ware via the World Wide Web. The system has been developed for teaching via the web. More specifically, It has the following features:

- Provide simple-to-use tools capable of producing multimedia presentations with little or no experience of multimedia editing including sound, images and handwriting
- Small enough presentation size makes a large corpus of multimedia tuition on a modestly sized server possible
- Support modem-speed download without any significant delay
- Support cross platform delivery of multimedia content by browser plug-in
- Plays as it downloads because streaming is supported

The system consists of two main components, which are an authoring sub-system called AudioGraph recorder and plug-ins for web browsers to playback the recorded presentations. Clearly, the AudioGraph is actually an authoring multimedia system plus some small piece of software, which can be embedded in browsers to handle the recorded presentations it produces. Before we are going any further to see how AudioGraph works, let us look at what an authoring system is. Fortunately, J. Silgar [38] gave us a clear point of view about it, i.e., "An Authoring System is a program which has pre-programmed elements for the development of interactive multimedia software titles. Authoring systems vary widely in orientation, capabilities, and learning curve". "Whether you realise it or not, authoring is actually just a speeded-up form of programming; you do not need to know the intricacies of a programming language, or worse, an API, but you do need to understand how programs work" [38]. Also there is not a criterion to tell us what is the best authoring system, since people might give different answers to an authoring system depended upon their needs. Now that we have got an idea of what kind of system the AudioGraph recorder is, we can move on to look at how the design goals are achieved.

The AudioGraph recorder has been well designed and under Silgar's classification of multimedia authoring tools, the media it supports are compressed audio, images and vector graphics; and it generates presentations for web-based delivery. In order to achieve these design goals, which promise to provide simple and easy to use tool for the user, two compromises were made to the system. Video was not one of the media elements, and neither was a complex programming model supported. The compromise of removing the support for video media in the system resulted in reducing the bandwidth requirements of the system. The other compromise was done to keep the user interface simple. This is achieved by providing only a single time-line (strict sequence only) to a presentation, so as to reduce the complexity of the programming model. In contrast, the system does support audio media and we must ask if the design goal of small footprint and modem-speed streaming can still be satisfied? The answer is yes, because it uses the GSM 06.10 RPE-LTP (regular pulse excitation long-term predictor) full-rate speech transcoder (This will be described in section 2.2), which is used by the European Global Systems Mobile digital telecommunications system [42][43]. The current version of the AudioGraph recorder works quite well and meets the design goal mentioned above.

To date, technologies have been moving forward rapidly and the current version of the AudioGraph recorder, which is an asynchronous one, does not satisfy the new demands that have arisen, such as real-time tutoring via the web, etc. Recently, a project called Technology Integrated Learning Environments for education at distance is being developed at Massey University (www-tile.massey.ac.nz). The AudioGraph tools will be integrated into the whole project and a need has arisen that means the tools must be improved towards synchronous version in order to satisfy the new demands of on-line educators, namely real-time tutoring via the web. This is the focus of this project, to explore the technologies required to implement this.

The objectives of the synchronous version of AudioGraph as follows:

- to keep the same low-bandwidth requirements

- voice would be transmitted over the internet in real-time
- transmit only the voice activity but not the silence
- to provide an audio link in both directions
- create GSM compressed speech on the fly using a small memory buffer and minimising any latency

In order to meet these objectives, we must optimise the existing system and will need to explore new techniques. Actually, the way the current version of the recorder works, it would be hard to meet the new real-time based demands. Therefore there are five aspects that must be changed or at least considered so as to improve the functionality of the system.

Firstly, the voice patch should not be of a fixed duration (currently limited by memory size to a few minutes of PCM recording) and redundant silence should be removed. As C. Jesshope mentioned in 1998 [41], there are two real-time events within the AudioGraph, which are a voice patch with a fixed maximum duration and pauses used to provide timing control over all other events. As is well known, usually over 50% of a conversation is silence. That means, that there are spaces that exist within the speech, which can be used to reduce the bandwidth requirements. A VAD is a good way to get rid of redundant silence within voice patch. After having removed redundant silence, only the conversational speech (or active speech) will remain in the voice. But this means the continuous speech has been divided into a list of small active packets transmitted in real-time. Obviously, the size of each packet is unpredictable, and this is the reason why we could not guarantee that each voice patch has an equal duration. However, we can set a maximum length of those packets. If the size of active speech packet is larger than the maximum length of a packet, we will simply break it. Up to this point, you can see that only the useful information will be handled and be transmitted across the network later.

Secondly a pause providing timing control should simply be replaced by a pause in transmission and a comfort noise generator may be required in order to smooth the sound, when it is reconstructed for playback. Another problem is that the transmission delay is

hard to predict. Let us assume the network is working normally, if the delay is small enough to be ignored, comfort noise should be produced when we start playback, otherwise, there is no need to invoke the comfort noise generator because the pause will have been replaced by timed pause. To understand this problem, let us look at the following diagrams. At first, figure 1 shows the waveform of the original speech signal. As can be seen, there is noise within the speech signal, which is continuous, and the speech is mixed in with it. Figure 1.1.3-1 shows the packetised result, where all noise and silence between two successive active speech packet has been removed, and where only three active speech packets will be transmitted across the network. At the other end of the network—the receiver end, if the time pause delay between two packetised syllables is much shorter than the natural silence period between the two syllables in the original speech signal, we should invoke the comfort noise generator to smooth the speech during playback. Otherwise, the timed pause does everything for us.

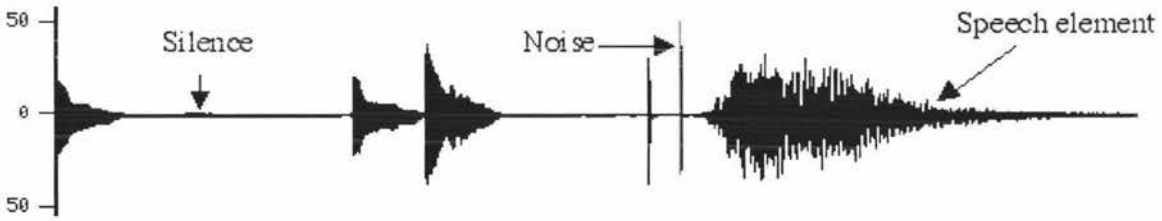


Figure 1.1.3-1. Original speech signal

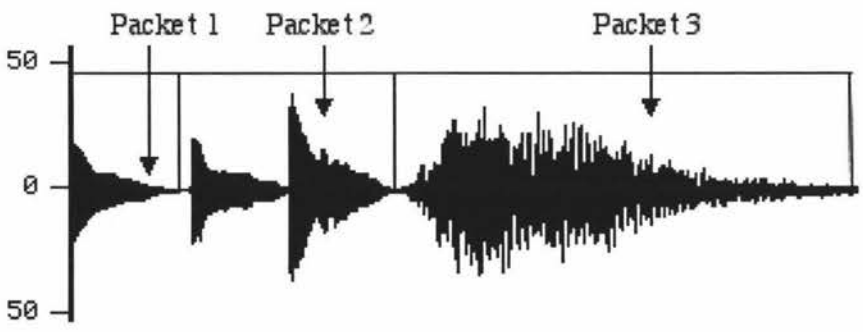


Figure 1.1.3-2. Output signal where three speech elements remained and packetised.

Thirdly when creating a voice patch in GSM format it should be performed on the fly, in order to reduce the memory requirements of the recorder and to speed up the system. In the current version of the recorder, all the voice patches are kept in PCM format as they are recorded, and they are only converted into GSM when a web site is produced. This results in large memory requirements when the user is producing his or her presentations. As is well known, the size requirement of GSM is approximately one tenth of that of PCM. Thus this is a tremendous saving in file size, especially when we are producing quite large presentations. It will also speed up our system because we hide the conversion delay during the sound recording process.

The final requirement, in order to support real-time tutoring, is that a peer-to-peer server must be set up and voice packets sent across the network.

A trial version of the synchronous AudioGraph recorder is being implemented at Massey University to demonstrate the techniques mentioned above.

1.1.4 Voice Activity Detection

Voice activity detection (VAD) is the process of separating conversational speech and silence [44]. As is well known, speech can happen in different environments in which several types of acoustic noise exist. More accurately, VAD is one kind of silence suppression techniques implemented by distinguishing between speech activity and inactivity, no matter what kind of environments the speech exists in.

VAD is being widely applied to speech communication applications such as speech recognition, speech coding, hands-free telephony and echo cancellation, and to

multimedia communication applications such as video conferencing [44]. Since it was investigated for use on the Time Assignment Speech Interpolation (TASI) system in the late nineteen fifty's, which was developed by scientists at Bell Labs [46], various types of VAD algorithms have been developed and applied to commercial products. VAD seems to have become one of the cores of speech compression algorithm standards. For example, there is a VAD algorithm within the G.729, which is a toll-quality 8-Kbps standard proposed by the International Telecommunication Union-Telecommunications Sector (ITU-T), and the G.729A, which is designed for simultaneous voice and data in multimedia communications [45].

The basic way a VAD algorithm works is described as follows [44]:

- Extract some measured features or quantities from the input signal.
- Compare these values with thresholds.
- The voice-active decision is made if the measured values exceed the thresholds.
- Repeat from 1 until we reach the end of the input signal.

Solutions that have provided VAD have varied from the early classical methods such as energy threshold (ET), zero crossing rate (ZCR) [48] and least-square periodicity estimator (LSPE) [49] to recently developed probability distribution based method such as geometrically adaptive energy threshold method (GAET) [50] and neural network method [47].

Algorithms Review

Before looking at algorithms, we had better define some terms, which will be helpful to understand the algorithms.

Signal-to-noise ratio (SNR) [53]: The ratio of the amplitude of the desired signal to the amplitude of noise signals at a given point in time. Both the signal and noise should be characterized, e.g., peak-signal-to-peak-noise ratio, so as to avoid ambiguity. SNR is usually express in dB, which is 10 times the base 10 logarithm of the power ratio or 20 times the base 10 logarithm of the amplitude ratio.

Window: "When we analyze signals we tend to do so on only small portions at a time" [54]. To window a speech signal before analysis means that we cut out a small section.

ET method

The classical energy threshold algorithm is very straightforward, simple, and linear. With this method, the energy of the signal is monitored and compared with the threshold value, which will usually be re-calculated for each analysis window [54] when noise level is varying. As a result of the threshold value being re-calculated, clearly, the algorithm can not work well in a non-stationary noise environment. Meanwhile, there is a condition to be met in order to detect the active speech region, which is that the energy of the total signal in the presence of speech must be sufficiently larger than that of the background noise. That is to say, the algorithm only works well with a high SNR value. Additionally, problems such as abrupt transitions at threshold causing noise may exist. Solutions, such as adding hangover period (consider several successive frames or a period of time before triggering), must be provided in the ET method, otherwise the algorithm can not work well in this case.

A typical example of VAD algorithm is the European Telecommunications Standards Institute (ETSI) VAD for the GSM system [45]. F. Beritelli et al has given a brief review of the example [45]. The following discussion is referenced from F. Beritelli's paper.

Actually, the ETSI VAD algorithm is essentially an energy detector based on adaptive threshold mechanism. Even though the ETSI VAD algorithm is not a classical ET algorithm, it is ET based and it does provide a clear example of an ET algorithm. That is the reason why it is discussed here. A solution to the abrupt transition problem is also provided within the ETSI VAD algorithm.

Firstly, the activity/inactivity classification's input consists of some parameters calculated by the GSM codec on frames of 160 samples, which come to 20 ms in duration. Secondly, the input signal will pass through an adaptive analysis filter so as to reduce the amount of background noise, the coefficients of which are calculated from the auto-correlation coefficients of the input signal averaged over four consecutive frames. This average operation gives a more accurate voice/noise discrimination. Thirdly, updating the threshold and the adaptive filter coefficients are controlled by two conditions. If at least one of the conditions is met, they are updated. The first condition is that the energy of the signal is very low. That represents a case where there is no speech, i.e., during absolute noise periods. The second condition is that the speech signal spectrum is stationary, and the signal does not contain a periodic component or sinusoids relating to network information tones.

"Signal stationarity is calculated by the likelihood ratio (LHR) between the coefficients of the current linear predictive coding (LPC) filter and the average over the last four frames. When the LHR spectral distortion is lower than a fixed threshold, the signal is considered to be stationary. The presence of a periodic component is determined using the pitch values calculated by the GSM speech coder very 5 ms" [45].

In order to prevent pauses between syllables from being misclassified as periods of inactivity, which causes speech clipping, the active voice decision is maintained for a period of five frames (100 ms). "This hangover period is only added when the VAD has detected active speech for at least three consecutive frames, thus excluding the likelihood that occasional disturbances present in silent periods will prolong the period that has been

misdetected as being active. The main feature of this VAD is that it is a fail-safe system, i.e., one that in doubtful cases indicate active speech rather than silence" [45].

ZCR method

"The zero crossing rate measures the number of times a signal crosses the zero line per unit of time. This gives us a measure of the dominant frequency in a signal -- the frequency with the largest amplitude" [52]. ZCR can be useful in differentiating between speech signal and noise signal since noise signal tend to have a large ZCR while in speech signal it is smaller.

With ZCR based VAD algorithm, the zero crossing rates for each analysis window are calculated and compared with the preset threshold value. An assumption of which the algorithm may work well is that the zero crossing rate of noise must be considerably larger than that of the speech signal. The assumption is precise at high SNR value. However, in the low SNR case, a problem arises. For example, when the speech signal is in the presence of periodic noise and the speech has a high zero crossing rate [44].

LSPE method

Tucker [49] designed a VAD algorithm based on periodicity. The difficulty that arises from periodicity measurement method is that it is sensitive to any periodic signal, which may well be interference, or a background signal. False triggering is easy to happen especially on non-speech periodic signals. When the speech signal contains non-periodic components, false triggering may happen [44].

GAET method

The GAET method is designed to set the threshold value adaptively instead of re-calculating it at each voice-inactive segment, by using the amplitude probability distributions (APD) of the speech signal [44]. The GAET method is very good for non-stationary noise but false triggering often happens when noise has short bursts.

H. Ozer et al [50] [44] gave us a briefly summary of the basic APD algorithm. Let us assume that the total signal can be written in the form

$$s(t) = c(t) + n(t) \tag{1}$$

Where $c(t)$ and $n(t)$ are the clear speech and noise signals, respectively. The APD function $F_s(s)$ and the amplitude probability density (apd) function $f_s(s)$ of a continuous time random variable $s(t)$ are related by

$$F_s(s) = \int_{-\infty}^{s} f_s(\xi) d\xi = \int_{0}^{s} f_s(\xi) d\xi \tag{2}$$

Let us denote the elements of the stochastic process on discrete time signal by $s[k]$ which are the (N+1) samples of $s(t)$ at $t = k\Delta t$ in the analysis window $(T_1 \le t \le T_2)$, i.e.,

$$s[k] = s(T_1 + k\Delta t) \tag{3}$$

for $k = 0,1,...,N$ and where

$$\Delta t = \frac{(T_2 - T_1)}{N} \tag{4}$$

Note that the sampling rate does not need to obey the Nyquist theorem [55] and N is assumed to be sufficiently large. The APD, $F_s[m]$, and apd, $f_s[m]$ of discrete time random variable $s[k]$ can be defined as

$$F_s[m] = \sum_{i=0}^{m} f_s[i] \tag{5}$$

and where $f_s[i]$ is the number of samples of $s[k]$ satisfying

$$i\Delta s \le |s[k]| < (i+1)\Delta s \tag{6}$$

normalized by the total number of samples N, and where Δs is the resolution parameter. Note that for $\Delta t \to 0$, $\Delta s \to 0, (T_1 \to -\infty)$, $(T_2 \to \infty)$ and $(N \to \infty)$, $F_s[m]$ and $f_s[m]$ converge to $F_s(s)$ and $f_s(s)$, respectively. If the APD of the signal and noise, $F_s(s)$ and $f_s(s)$, are different then, $F_s[m]$ and $f_s[m]$ are expected to be different.

For a corrupted signal, the signal and noise will partially occupy different regions on the APD.

Neural network method

The VAD algorithm based on a neural network is one of the most effective methods in speech activity/inactivity separation. It can achieve better performance at any background noise level. The disadvantage is its high computational cost.

In 1998, J. Ikedo [47] proposed a VAD using neural network. Let us take a look at how it works. The neural network gets three parameters as input, and outputs only one scalar value related to voice activity, which is used to compare with the threshold value to see whether there is a trigger point of activity or inactivity. The three parameters are short-term power, pitch stability parameters, and spectrum envelope, used widely by CELP (Code Excited Linear Prediction) (discussed in section 2.2) based Codecs. Within a voice slice, the short-term power is high; the pitch is stable and the spectrum envelope has some characteristic. At first, these parameters are calculated by analysing windows of the speech signal. Then they are input to the neural network that is trained to output one scalar value, and the output of the neural network is compared to a threshold to determine whether there is trigger point.

Schemes	Good performance under environments		Computational Cost	Good performance Min SNR (dB)
	Non-stationary	stationary		
ET	No	Yes	low	>>5
ZCR	No	Yes	low	5 [44]
LSPE	No	Yes	low	-5 [44]
GAET	Yes	Yes	High	-5 [44]
Neural network	Yes	Yes	high	-26 [47]

Table 1.1.4-1. Comparison of VAD methods

Within table 1.1.4-1, environment means that the background may include non-stationary noise or stationary noise. The minimum SNR value, here, that guarantees good performance, is extracted from reference [44] and [47].

As a consequence of the above comparison, it is not difficult to conclude that, most of the VAD algorithms are based on a compromise because different VAD algorithms trade off delay, sensitivity, accuracy and computational cost.

Noise Environments of the AudioGraph

The environments where the AudioGraph system will be used also consist of stationary noise and non-stationary noise. As a result of that the AudioGraph's users are academics, the environment noise can be further categorised as the stationary noise and the easily controlled non-stationary noise. On the one hand, the regular noise of academics' working environment mostly comes from computer fan, air conditioner fan, and etc. They are stationary noise. On the other hand, there are some strong non-stationary noises in the

working environment, such as doors banging, door knocking, and the phone ringing, and they are not regular ones. When they come, the users of the AudioGraph might stop their work in most of these cases. Even if they do trigger a response, it is not important. Therefore, the strong non-stationary noise case should not be considered. As for the non-stationary noise such as mouse clicking and keyboard typing, we can control them very easy during recording by positioning of microphone, or adjusting the input level control of the microphone.

VAD algorithm in the AudioGraph

We have decided to choose the simple energy threshold method as a VAD algorithm targeted in the AudioGraph. As described above, the simple energy threshold algorithm is straightforward, simple, has a low computational cost, and can have good performance when the SNR value is large and the background noise is stationary. There are three reasons to support our decision. First, the SNR value will be large because users are speaking in a quiet environment. Second, a simple and low computational cost algorithm meets the goal of the AudioGraph. Third, as regards the non-stationary noise, some simple and easy ways such as positioning a microphone can provide very good control to them without costing one penny. As a result of that, there is no reason why we do not use low cost scheme, the simple energy threshold algorithm, in the AudioGraph. The algorithm used will be described in more detail in section 2.6.

1.2 Definition of problems

The specification of the project is to implement the following:

- On-the-fly recording and compression of GSM sound.
- Implement 2-way voice over IP connection to be incorporated at a later stage into the asynchronous AudioGraph.
- Ensure protocols of voice over IP transfer can handle echo suppression by using a simplex channel.

As can be seen, there are three major problems to be solved during the implementation. Let us describe them respectively. To begin with, we had better define two terms (used throughout this paper): System-A, which is that the GSM compressed sound is recording on the fly, and System-B, which provides 2-way voice over IP connection based on the System-A.

The first problem is that of recording GSM compressed sound on the fly and then preparing these recorded sounds for later playback in sequence. As described in the AudioGraph tool section, the asynchronous AudioGraph recorder takes two steps to generate GSM compressed sound. In the first step the recorder records sound into PCM format, and in the last step it will convert the recorded sound into GSM compressed sound. When the user pushes down the record button, the system starts recording and stores the recorded data into a buffer as PCM format. It will not stop recording until the record button is released or the buffer is full. If so, the data within the buffer will be saved and kept in PCM format. In the final state of the presentation creation, all recorded sound media will be converted into GSM compressed sound. Obviously, it is very slow and consumes a lot of memory. In order to overcome these drawback, we must consider several sub-problems, which are choosing reasonable buffering mechanism, making sure continuous recording possible while data is processing, VAD processing, and PCM-to-GSM conversion, so as to provide the solution.

These sub-problems are the key issues, which will guarantee that the System-A works well, and solutions of those sub-problems constitute the final solution to the System-A.

The buffering mechanism chosen has great impact on the System-A, especially if the

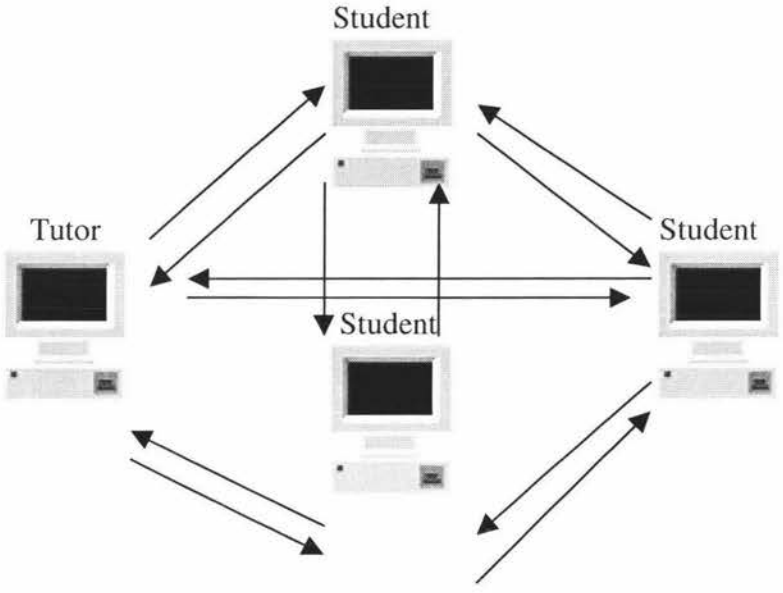


Figure 1.2-1 Multicasting tutoring

system is time-sensitive. We must keep the memory usage as low as possible.

In addition to continuous recording, the Macintosh does support asynchronous recording so that we can do recording and sound processing at the same time. The Macintosh operating system gives us a chance to call those low-level routines that provide developers more flexible and powerful functionality. Otherwise, multi-threading can be used to achieve the same thing, especially on other platforms where the continuous recording is not supported.

With respect to the VAD processing, this is the core of the System-A. It makes System-A record active speech and ignore inactive speech such as noise and silence. Only the active speech will be transferred to the PCM-GSM converter. Here we apply a voice activity detector, which has been implemented in simple energy threshold algorithm (this will be discussed in details in next section).

As regards the PCM-GSM conversion, we use GSM 6.10 library to convert the PCM-based packetized sound into GSM-based packetized sound. The details of GSM 6.10 and the details of PCM will be discussed in the next chapter. As the output of the VAD processing, active speech has been packetized as PCM format. Each of these packets will be converted into GSM compressed sound, actually, GSM variant WAV#49 format, in sequence by frame-to-frame conversion.

The second problem is to implement 2-way voice over IP connection to be incorporated at a later stage into the asynchronous AudioGraph. As can be seen, System-A can only playback packetized sound in sequence. System-B based on the System-A should also provide 2-way voice over IP connection. Obviously, System-B is a VoIP system, which has two major components, a gateway and a gatekeeper. It also is a peer-to-peer server.

The functionality of the System-A is quite similar to a gateway except it does not provide the mechanism to decompress the data received from the network. Hence a task has arisen, which is to expand the functionality of System-A to meet the needs of a gateway. In order to playback the decompressed sound, we should build a real-time player within the gateway as well. To ensure real-time playback successfully, a compromise must be made, which trades off delay and echo. This will be discussed in the last problem.

In addition, a gatekeeper that provides communication within the network should be added. Within System-B, the approach we have chosen is to distribute gatekeepers within the network. More clearly, each peer-to-peer server running on the network has its own gatekeeper. The gatekeeper can provide address translation, bandwidth control and zone management. As we are currently working on a private network, the admission control has not been included yet; it will be left to the later stage of the synchronous AudioGraph. Furthermore, the gatekeeper supports multi-cast communication shown in figure 1.2-1. This guarantees that a tutor can communicate with several students simultaneously.

The last problem is to ensure protocols of voice over IP transfer can handle echo suppression by using simplex channel. As described in the previous problem, a real-time player working well is based on a compromise. Figure 1.2-2 shows how an echo is generated. The input signal S comes in from left-hand side, then across the packet network, it is output from the right-hand side. But the story has not ended, it goes back to the microphone as input signal again, it repeats and repeats. No need to say this will cause echo. So the echo suppression is a significant factor that should be considered carefully.

An echo inhibition protocol provides a simple way to break the echo in the System-B. The protocol ensures echo suppression by using a simplex channel. Specifically, when there is no recording event, the playback can be activated; when there is no playback event, the recording can be activated. That is to say neither event can occur simultaneously. In order to implement the protocol, we must trade off delay and echo suppression, by means of providing buffer to store those sounds waiting to be played back.

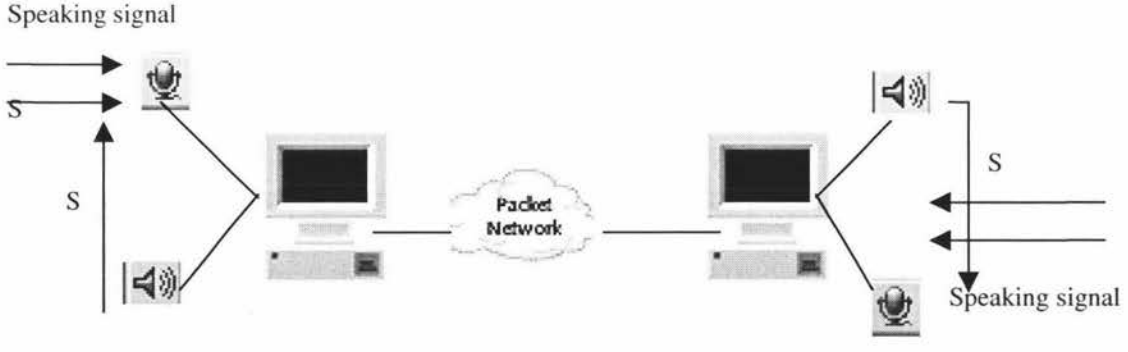


Figure 1.2-2 Echo generation.

Within this section, three major problems have been discussed, and some sub-problems have been identified in these major problems. This section has presented a clear view of this project's tasks.

Chapter 2 Research Techniques

In this chapter we outline all relevant techniques such as speech coding schemes, sound file formats, and etc., including the VAD algorithm used in the synchronous AudioGraph.

2.1 Sound Representation

Sound carries information and is the ubiquitous nature of mechanical radiation, produced when an object (the source) vibrates and causes the air around it to move. Obviously it is a means of transmitting information. As a sense, human being can produce sound – speech—that allows us communicate. And speech consists of active and inactive voice.

To date, sound has become one of the most important media in reality. We can hear someone's speech simultaneously 10,000 miles away or even further. This can be achieved by data communication. Data communication is sending the coded voice signal from one location to another. The voice signal will be sent out by a transmitter, will traverse the network by a path, and will end up with a receiver. In reality sound could not naturally go that far. But we did grasp sound and the miracle comes true. So let us look at how to represent sound –that is—the analog and digital representations.

2.1.1 Analog

Sound can be represented as an analog signal. An analog signal is a continuously varying voltage (or current) representation of a sound wave. Before 1980's, most of the telephone systems were analog-based. So a telephone can produce an analog signal very easily. When people speak, the voice goes into the transmitter of a telephone set and changes the air pressure (these are sound waves). The diaphragm responds to changing air pressure

and changes a circuit's resistance by compressing carbon in the transmitter. The change in resistance causes the current flow to fluctuate, creating an electrical wave analogous to the sound wave [56].

An analog signal is composed of different attributes such as frequency and amplitude, which define the sound wave that an analog signal represents. The rate at which the source oscillates is the frequency of the sound wave, and is quoted in hertz (Hz) or cycles per second (cps). The amount of compression and rarefaction of the air, which results from the source's motion, is the amplitude of the sound wave, and is related to the loudness of the sound. The frequency and amplitude are two characteristics of the analog signal that can be varied to convey information.

But the analog signal has negative phenomena—impairments. It affects the transmission of analog signals. The impairments consist of loss (Attenuation), Noise (Unwanted electrical signals) and Distortion (Frequency characteristic changes).

When the analog signal progresses along the transmission medium, it is continuously attenuated, or weakened. To overcome this, amplifiers should be placed at the intervals to compensate for the attenuation.

The analog signal picks up noise as it travels through the network. The noise and distortion change the shape of the analog signal. Meanwhile, using amplifiers causes that the effects of noise and distortion to be cumulative, because the amplifiers reproduce all of the analog signal without distinguishing noise, voice and distortion components of the analog signal [56].

This kind of cumulative nature of transmission impairments is the major disadvantage of an analog transmission system. To thoroughly overcome this analog problem, the digital signal was invented.

2.1.2 Digital

Sound can be represented as digital signal as well. A digital signal is a discontinuous signal in the form of pulses and is composed of a sequence of binary integers. The digital signal can provide very high quality, which is independent of the storage or transmission medium and is determined instead by the accuracy of conversion between the analog and digital domain.

To represent an analog signal in digital form is achieved by analog-to-digital conversion. This comprises the three steps of sampling, quantising, and encoding. Within the conversion, the input to the process is a continuous-time, continuous-voltage waveform, and this is changed into a discrete-time, discrete-voltage format by a combination of sampling and quantising [58]. Finally, encoding generates binary integers.

With respect to the sampling, portions of a signal are used to represent the whole signal. As shown in figure 2.1.2-1, Pulse Amplitude Modulation (PAM) results in a sequence of pulses whose amplitude is proportional to the amplitude of the sampled analog signal at

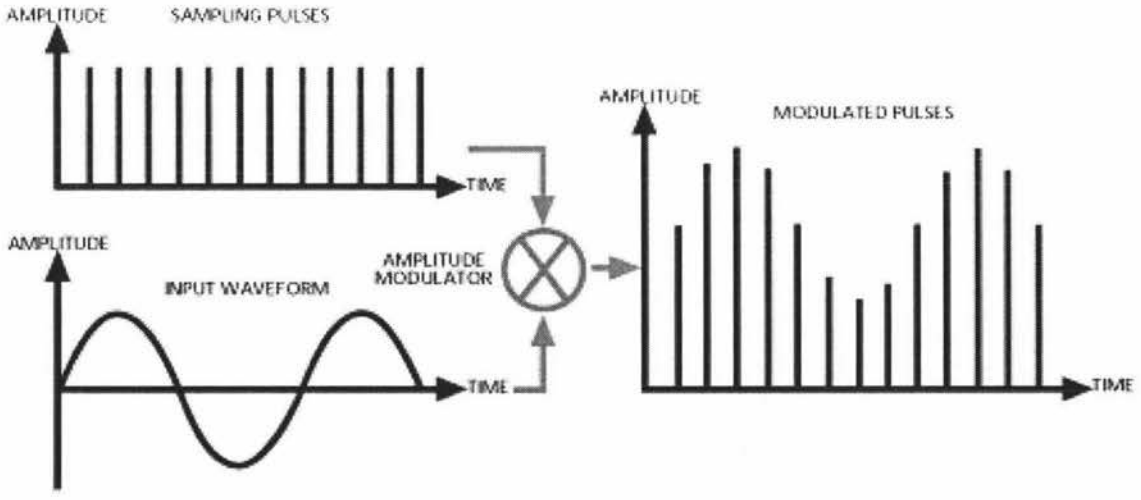


Figure 2.1.2-1. The sampling process results in PAM [57]

the sampling instant. Each time the signal is sampled, a PAM signal is generated. In order to reproduce the analog signal faithfully, the sampling process must obey the Nyquist's theorem (the sampling rate must be at least twice the highest input frequency). Whilst, on the input, anti-aliasing filters must be provided to prevent frequencies, which are more than half the sampling rate, from reaching the sampling stage [58].

Quantising is the process of expressing some infinitely variable quantity by discrete or stepped values. During the quantisation, the PAM signal is measured and coded. The amplitude or height of the PAM is measured to derive a number that represents its amplitude level.

As for the encoding, the decimal number derived in the quantising step is then converted, for example, into its equivalent 8 bit binary number. The output is an 8 bit "word" in which each bit may be either a "1" (for pulse) or a "0" (for no pulse) [56].

Within a computer, digitising speech is mostly done by hardware—A/D (analog/digital converter). And D/A (digital/analog converter) does reverse job.

2.2 Speech coding

To date, speech-coding schemes can be broadly divided into three main categories: Waveform coders, Vocoders, and Hybrid Coders. The general function of these coding schemes is to analyse the signal, remove the redundancies and efficiently code the non-redundant parts of the signal in a perceptually acceptable manner. The waveform coding is capable of taking an appropriate number of samples and encoding the value of each sample so its reconstructed amplitude did not significantly differ from its original amplitude. And the vocoding is based on modelling speech. The speech signal can be consider to be stationary during short period of time such as tens of milliseconds, so that sound is predictable and can be synthesised. That is the reason why the vocoding method works well. The hybrid coding was developed based on waveform coding and vocoding and is as straightforward as its name [60]. We will discuss these coding schemes respectively.

In order to understand these coding schemes easily, we have better say something about human speech and the three classes of speech based on their mode of excitation.

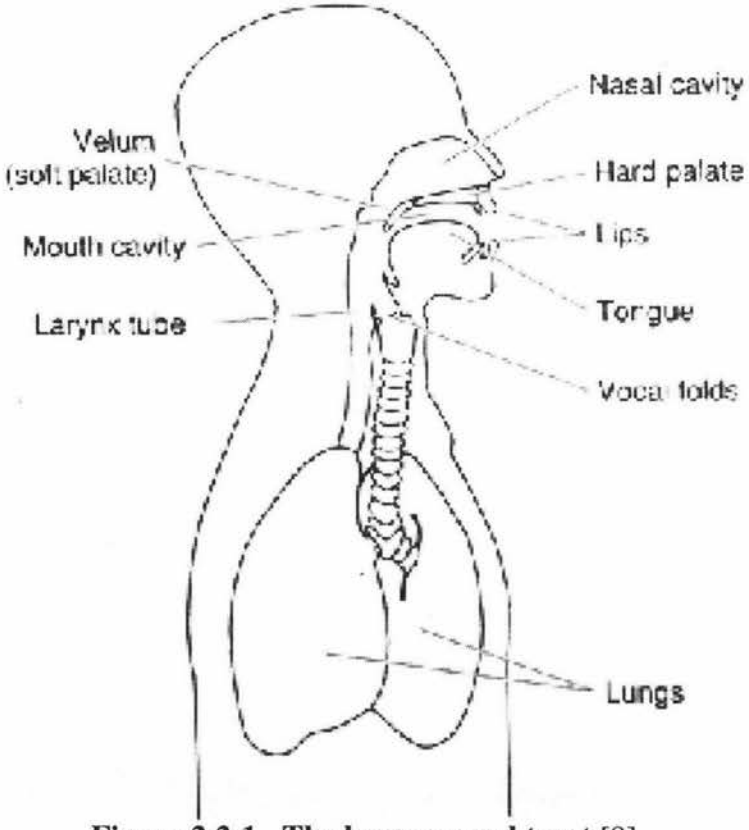


Figure 2.2-1. The human vocal tract [8]

When speaking, air is forced from a human's lungs through human's two vocal folds and along the human's vocal tract. Tension in the vocal folds (or cords) is increased until they close. After the folds have closed, air from their lungs forces the glottis open again.

The cycle repeats between 50 to 500 times per second, depending on the physical construction of their larynx and how strong they pull on their vocal cords. This is the process of how to produce vibrations in the vocal cords. Next, these vibrations pass through the throat, over the tongue, against the roof of the mouth, and out through the teeth and lips, and finally, reach human's ear. This is the transmission of speech in a natural way. During transitions, and for some "mixed" phonemes, people use the same airstream twice--first to make a low-frequency hum with their vocal cords, then to make a high frequency, noisy hiss in their mouth. Figure 2.2-1 illustrates the human's vocal tract and several parts of the body that effect the production of speech.

Normally, speech signals can be categorised into one of three classes based on their mode of excitation, which are voiced sounds, unvoiced sound, and a mixture of the two [60]. A voiced sound segment is known by its relatively high energy content. It contains periodicity that is called the pitch of the voiced sound. In addition, in the unvoiced sound segment, there is no periodicity any more, and what it looks like is like a random noise. Because the voice will attenuate in time, there is a transition region containing voice and unvoice sounds, where there is a change either from voice to unvoice or vice versa. Now let us turn our attention back to speech coding schemes.

2.2.1 PCM (G.711 Recommendation)

Pulse code modulation (PCM) [8][60] is a waveform coding technique where an analog signal is digitised. It's the foundational method of voice digitisation used by communication carriers for transmission on the PSTN, consisting of three-step process: sampling, quantisation, and coding.

In the sampling process of PCM, an analog signal is sampled at 8,000 times per second, or once every 125 μs. The Nyquist theorem specifies that frequency must be at least twice as high as the highest frequency in the wave and this must be considered when the

sample rate is chosen. This guarantees that the analog signal can be reconstructed faithfully. Some might ask why this sample rate is selected? Normally, the frequency range of human speech is between 300 Hz and 3,400 Hz. In order to represent a sound wave digitally, without loosing any essential information, a sample rate at least 6,800 times a second is required (according to the Nyquist's theorem). But PCM's filters do not work instantaneously and allow some lower-power speech to pass below 300 Hz and beyond 3,400 Hz. As a result, the passband can extend to near 4,000 Hz. This results in the sample rate 8,000 times per second being selected as shown in Figure 2.2-2.

The sampling process constructs a PAM wave as shown on figure 2.1.2-1. The PAM wave represents a series of samples. These samples can have an infinite number of voltages because the samples represent analog heights and not discrete digitally encoded signal values. Therefore, to obtain a limited number of discrete amplitude values, PAM wave must be quantised.

In quantisation process of PCM, the transmission rate becomes the crucial factor that affects the decision of choosing how many bits are in the binary code to represent each sample value. The larger the number of bits we use, the more accurate the signal we reconstruct. But larger number of bits results in higher transmission rate. Usually, the PCM we used is general telephony PCM, which uses an 8 bit binary code and a sample rate at 8 KHz, and this requires a transmission rate of 8000 samples

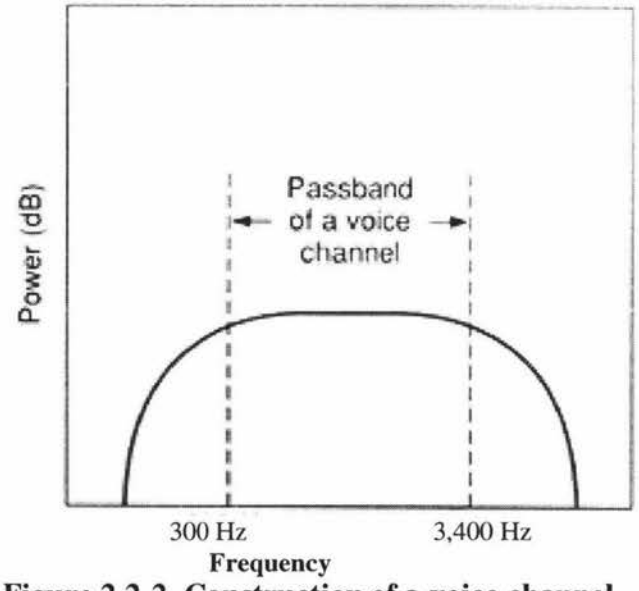


Figure 2.2-2. Construction of a voice channel. Through the use of filters, a passband between 300 and 3,400 Hz is established for use as voice channel. [8]

per second x 8 bits/sample, or 64,000 bps.

However, using an 8 bit binary code and linear quantisation could not satisfy all cases. For example, to represent a 60 dB power range completely, 12 bit per sample would be required if linear quantisation was used. This also results in a bit rate of 96,000 kpbs. So non-linear quantisation techniques must be introduced so that the excess amount of bandwidth (in 96,000 kpbs) can be reduced. The kind of technique used is to compress or compand the signal prior to quantisation, followed by uniform quantisation.

Within a PCM system, the compandor (compressor-expandor) enlarges the weak signal so that they can be transmitted above the noise and crosstalk level associated with a typical communications channel while attenuating very high signals to minimise the possibility of crosstalk that affects other communications channels [8]. Compandors are logarithmic and follow one of two laws: the A-law (employed primarily in Europe) and the $\mu-law$ (used primarily in North America).

The A-law compression is defined by

$$ALaw(x) = \frac{Ax}{1+\log_{10}(A)} \qquad \text{for } 0 \le x \le \frac{1}{A} \tag{1}$$

$$ALaw(x) = \frac{1+\log_{10}(Ax)}{1+\log_{10}(A)} \qquad \text{for } \frac{1}{A} \le x \le 1 \tag{2}$$

where A is the compression parameter with typical values of 86 for 7 bit (North America PCM) and 87.56 for 8 bit (Europe PCM) speech quantisers.

The $\mu-law$ compression is defined by

$$\mu Law(x) = sign(x)\frac{V_0 \log_{10}[1+\frac{\mu|x|}{V_0}]}{\log_{10}[1+\mu]} \tag{3}$$

where V_0 is given by $V_0 = L\sigma_x$, in which L is the loading factor and σ_x is the root mean square value of the input speech signal. A typical value of the compression factor μ is 255 [60].

Before turning to the encoding of the PCM system, we had better define two terms (under $\mu-law$): chords and steps. The chords are spaced logarithmically, with each succeeding chord larger than the preceding one. Within each chord, there are 16 steps spaced linearly. Therefor the steps are larger in larger chords. However, in the A-law, 13 segments are used to replace the use of 16 chords under $\mu-law$ [8].

Each PCM word consists of three parts: a polarity bit, 3 bits for the chord value, and 4 bits to represent one of the 16 possible steps within a chord. So encoding is not simply encoding a step to represent the power level of a sample.

PCM was standardised by ITU as recommendation G.711. It is not only the most widely used waveform scheme thus far, but also the fundamental method broadly used on PSTN. The A- and $\mu-law$ companding can both provide high quality of speech reconstruction. The different between the two methods is so trivial that the difference between the reconstructed speeches by the two methods is hard to distinguish by human's ear. PCM was developed during the 1960s. Nowadays lower-bit-rate-coding technologies have been developed very quickly because of the computer technologies revolution.

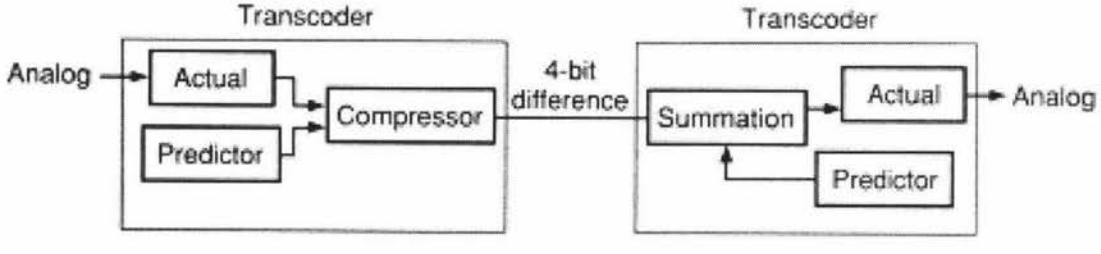


Figure 2.2-3. Block diagram of ADPCM [8]

2.2.2 ADPCM (G.721 Recommendation)

Adaptive differential PCM (ADPCM) [8] is also a waveform coding technique, using the same sample rate, 8,000 samples per second, with that used in PCM. As for the quantisation, a transcoder, which includes an adaptive predictor and compressor, is used to replace the quantising of the actual signal samples. The compressor subtracting the predicted value from the actual value of the sample generates the difference. Finally the difference is encoding into 4-bits word. The block diagram of the ADPCM system is shown on figure 2.2-3. The ADPCM results in the transmission rate of 8,000 samples per second x 4 bits/sample, or 32,000 bps. This scheme was standardised by ITU as recommendation G.721.

2.2.3 GSM 06.10 RPE-LTP

The GSM 06.10 RPE-LTP (regular pulse excitation long-term predictor) is a full-rate speech transcoder [60][61][62], which is defined as a standard by ESTI. It falls into the hybrid Coders category, which overcome the deficiencies of pure waveform and vocoding schemes, and incorporate the advantages offered by each of the pure schemes. GSM 06.10 RPE-LTP is a time domain hybrid coder that employs forms of linear prediction, and it is, more accurate, a hybrid of RELP (Residual excited linear predictive coding) [65][66] and MPLPC (Multi-pulse LPC) [63][64]. In the former ancestor, the RELP is based on analysis-and-synthesis procedure, which is that the speech signal is analysed to extract the parameters that are used to remove speech redundancies and the remaining signal is then quantised. Only the residual information felt in the low frequency regions --the baseband—of the speech is significant and encoded. In the latter ancestor, the MPLPC is based on an analysis-by-synthesis (ABS) technique, which is that each sub-system is jointly optimised such that the overall synthetic speech introduces minimum distortion. This is achieved by having a local decoder at the transmitter end such that the synthetic speech is available for analysis. But this coder has a disadvantage in that is has a relatively high computational load. The GSM 06.10 RPE-LTP transcoder

adopts a solution that is a compromise of its ancestors. The optimal excitation is constrained to be equally spaced pulses of different amplitudes, and by using a time-variant block filter, which is the equivalent for the low-pass filter in RELP, as an approximation to the full analysis-by-synthesis procedure [60]. "By selecting the highest energy baseband after the block filtering, the RPELPC becomes effectively a RELP coder with variable baseband grid selection" [60].

As can be seen from the figure 2.2-4, the GSM RPE-LPC coder consists of two main blocks: encoder and decoder.

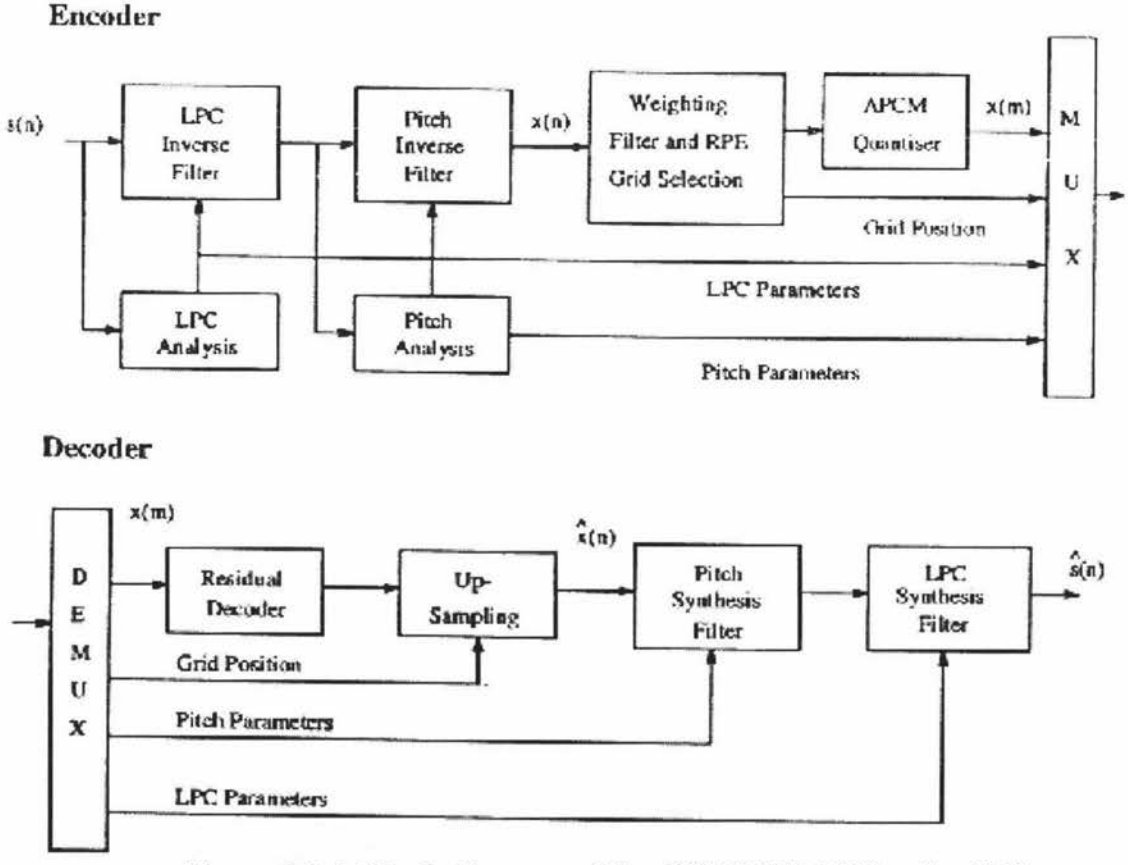


Figure 2.2-4. Block diagram of the GSM RPE-LPC coder [60]

Encoder

Within the encoder, the computation can be broken down into three blocks: short-term analysis (LPC analysis), long-term analysis (pitch analysis), and remaining residual pulse; then it quantises and encodes the residual pulse; and finally it parameters for the two predictors.

To model the speech production, the LPC analysis is performed on its input, a 20-ms frame, which is extracted from the original speech signal and contains 160 PCM samples. The LPC analysis that is often referred to as short-term analysis calculates the short-term

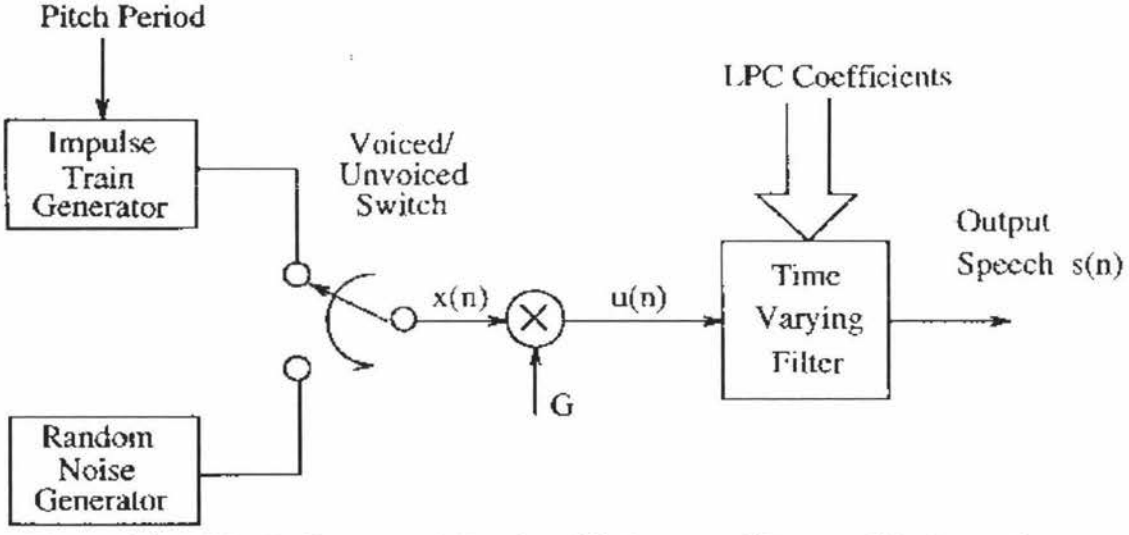


Figure 2.2-5. Block diagram of the simplified source filter model of speech Production [60]

residual signal that will excite the short-term synthesis stage in the decoder.

A source filter is designed to model the speech production so as to model the effects of the vocal and nasal tracts. This is shown on the figure 2.2-5. Within this model the driving input, or excitation signal, is modelled as either an impulse train (for voiced speech) or random noise (for unvoiced speech). The LPC coefficients represent the source filter model. So the first computation task within the encoder is to compute a set

of LPC coefficients. And what is the number of LPC coefficients here? As a result of low-bit-rate requirements, speech coding only models the useful speech signal, whose frequency ranges from 300 Hz to 3,400 Hz. The speech coding is usually sampled at 8 kHz, thus giving a 4 kHz spectrum for analysis. Within the 4 kHz spectrums, the maximum number of formants displayed is usually four, thus indicating that the number of coefficients needs to be at least eight. In GSM, eight is chosen.

Let us denote that an array $lpc[p]$ represents the LPC coefficients, where p is 8, and that every frame of speech samples is in array $s[n]$. The computed $lpc[p]$ must satisfy that $s[n]$ is as close as possible to $lpc[0]\times s[n-1]+lpc[1]\times s[n-2]+...+lpc[p-1]\times s[n-p]$ for values of all samples $s[n]$.

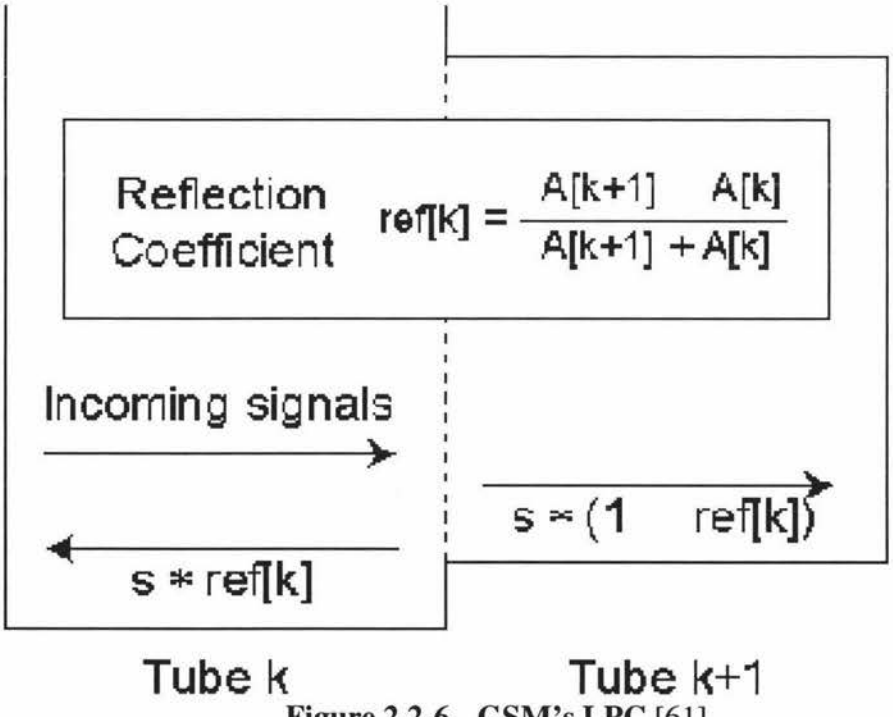


Figure 2.2-6. GSM's LPC [61]

But the $lpc[]$ coefficients computed above are not exactly the same with the one used on GSM's LPC—closely related to "reflection coefficients"—shown on figure 2.2-6. This is

a physical model of the GSM's filter: a system of connected, hard-walled, loss-less tubes through which a wave travels in one dimension. A reflection coefficient calculated from the cross-sectional areas of both tubes expresses the rate of reflection [61].

After having finished the LPC analysis, the long-term analysis usually referred as pitch analysis is preformed. And the just used frame is further subdivided into four 5-ms subframes. For each subframe, two parameters for the long-term predictor, which are a gain and a delay, are computed.

On the one hand, a gain (the LTP gain), which is the scaling factor, is the maximum correlation divided by the energy of the reconstructed short-term residual signal. The energy of a discrete signal is the sum of its squared values--its correlation with itself for a lag of zero. The *lpc[]* array in the LPC analysis plays a similar role to the LTP gain.

On the other hand, a delay (the LTP lag) describes the source of the copy in time. In order to compute the delay, the encoder looks for the segment of the past that most resembles the present, regardless of scaling. How is resemblance computed? The answer is the correlation of the two sequences whose resemblance we want to know. The correlation of two sequences, *x[]* and *y[]*, is the sum of the products *x[n]*y[n--lag]* for all *n*. It is a function of the lag, which shows the time between every two samples that are multiplied. The lag between 40 and 120 with the maximum correlation becomes the LTP lag. Ideally, the computed LTP lag will be the distance between two glottal waves, the inverse of the speech's pitch.

Next, the encoder uses 40 residual signal samples (eight per subframe) and converts those samples into three groups of excitation samples, each 13 samples in length. The aim of computing the excitation samples is to reconstruct a waveform, which is matched against the original waveform, with the selected coefficients used by a synthesis filter. Therefore the difference between waveforms is obtained. This process is repeated using a new set of coefficients until the error between synthesised speech and the original waveform reaches

a minimal level. Meanwhile, three samples are selected when synthesised. These samples produce a minimal difference between the original and synthesised waveform.

Then the encoder produces four evenly spaced 13-value subsequences to choose from, starting with samples 1,2,3, and 4. The sequence, which has the highest energy level, is selected as the best representation of the excitation sequence. A 2-bit grid-selection index transmits the choice to the decoder. That results in the PCM encoding being turned into an Adaptive PCM (APCM) (the algorithm adapts to the overall amplitude by increasing or decreasing the scaling factor), by using 13 3-bit sample values and a 6-bit scaling factor [61].

Finally, the short-term residual must be reconstructed, and the next LTP analysis is prepared by updating its remembered previous output. And the transcoder must maintain consistency in that the encoder and decoder must use the same residual signal, by using the decoder's grainy approximation of the past instead of its own more accurate residual signal.

Decoder

The reverse of the encoder is performed. To obtain the residual signal that is the input to the long-term synthesis filter, the compressed signal will be multiplied by the 13 3-bit samples of the scaling factor and be expanded back into 40 samples. If necessary, a zero-padding technique is used to cover the gaps. Next, produce the new estimated short-term residual signal, by cutting out a 40-sample segment from the old estimated short-term residual signal, scaling it by the LTP gain, and adding it to the incoming pulse. Last, the new estimated short-term residual signal passes through the short-term synthesis filter, and the speech is reconstructed.

2.2.4 CELP

Code-excited linear predictive coding is a variant of analysis-by-synthesis LPC schemes, first reported by Atal in 1982 [60], and providing bit-rates below 8kbps.

Compared with RPE-LPC, the CELP differs in that the excitation signal is vector-quantised after speech is passed through a vocal tract and pitch predictor, and an index from a codebook is used in place of the actual quantisation of the excitation signal. By using a 10-bit index, a codebook of 1024 entries can substantially reduce the bit rate required to transmit the excitation signal that best represents the original waveform.

The computation of the CELP can be broken down into the following blocks: LPC analysis, long-term prediction (LTP) analysis, and codebook search. The block diagram of the standard CELP algorithm is shown by figure 2.2-7. Let us examine how this algorithm operates [68].

The original speech $s(n)$, as the input, is first divided into analysis frames of around 20-30 ms. The LPC analysis results in a set of LPC coefficients being computed. These LPC coefficients are used to model the spectral envelope of the speech by the short-term predictor (STP). And the memory of the STP (initial condition) is removed from the reference to give a memory-less STP for subsequent analysis.

Next, subdivide the LPC frame into subframes, e.g. 5-10 ms. The LTP analysis is formed on those subframes and can be performed on either the residual generated by an inverse filter (with the derived LPC coefficients) or the original speech (with the memory of the STP from previous blocks removed). This results in two parameters being computed: a delay and a gain.

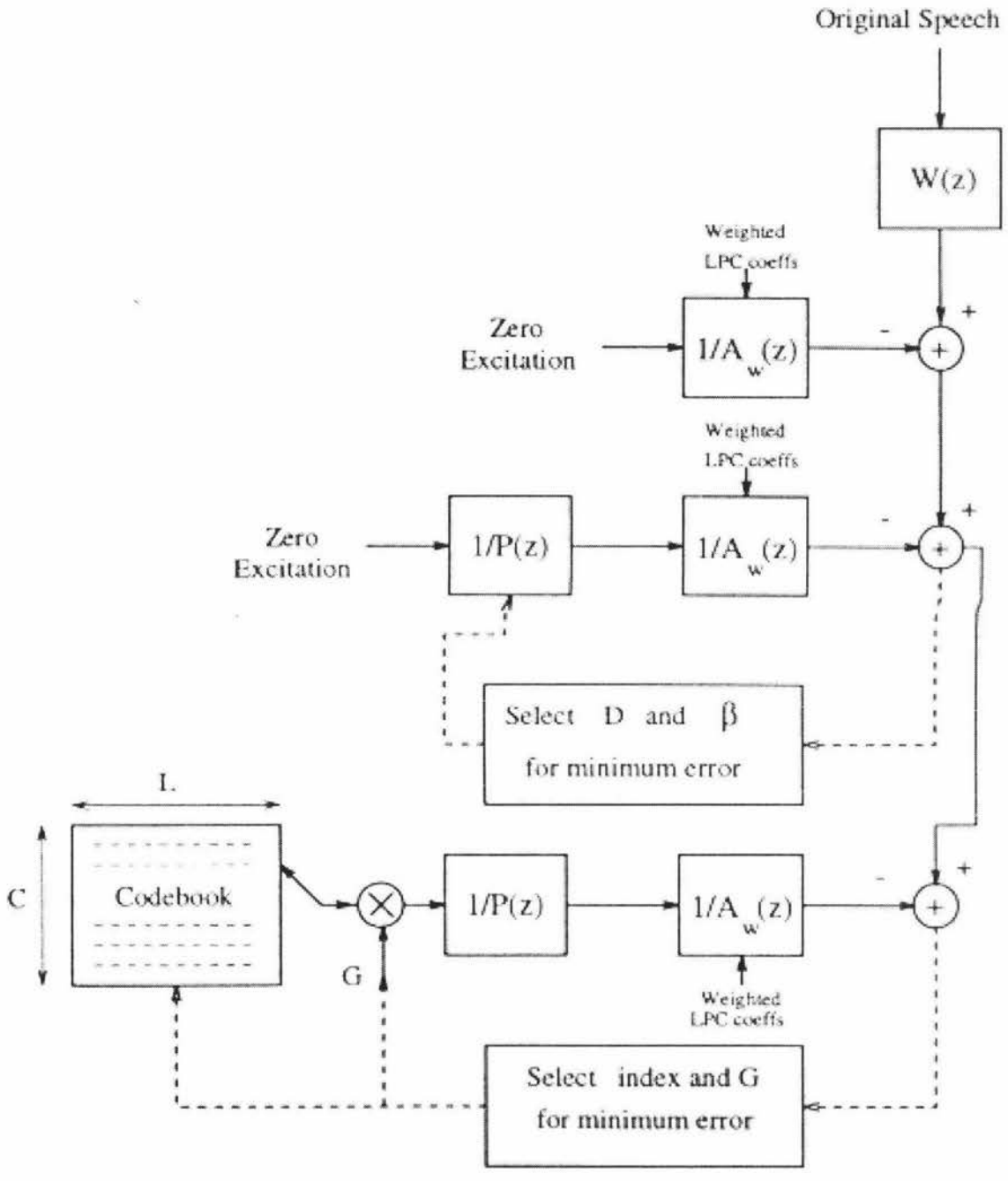


Figure 2.2-7 The block diagram of CELP [60]

After that, the excitation can be determined. Again, based on those subframes mentioned above, the excitation is updated. In standard CELP, the excitation $x(n)$ is selected from a codebook of random white Gaussian sequences. In order to obtain the best $x(n)$, the search procedure must be performed based on the following steps:

- To generate an ensemble of filtered Gaussian sequences (synthetic speech).
- To compute the objective error for each sequence.
- To select the sequence with minimum error.

As for the synthesiser, the initial conditions of the filters are restored. And by filtering the scaled optimum codebook sequence through the filters without any perceptual weighting, the synthetic speech is generated.

To date, some variants of CELP coders have been standardised by ITU, such as G.728 recommendation, G.729 recommendation, etc. The key difference between various CELP coders is in the adaptability of codebook entries and the delay associated with selecting and transmitting a codebook index that represents the excitation signal.

2.2.5 G.728 Recommendation

The G.728 CELP recommendation is called Low-Delay Code Excited Linear Prediction, developed at AT&T Bell Laboratories, having a bit-rate of 16 kbps, and being standardised by ITU in 1992. The aim of this scheme was to overcome the significant delay of the FS 1016 CELP coding (FS 1016, a version of CELP, was standardised by the U.S Department of Defense and operates at 4.8 kbps.) and to provide toll-quality speech comparable to ITU's G.721 recommendation. The FS 1016 CELP is broadly used for secure telephone communications via the encryption of its digitised data stream. But the drawback of this scheme is that, an end-to-end transmission delay can be introduced resulting in a system delay of up to approximately 100 ms.

The G.728 CELP recommendation uses a backward-adaptive method to calculate the LPC coefficients, as the LPC coefficients are computed from past reconstructed speech rather than from a 20-30 ms frame. This results in a shorter frame length, a frame length of five samples, is used instead of that used by other CELP coding techniques. As a result of that, a delay of proximately 20-ms is produced. Furthermore, the scheme uses a high-order short-term predictor so as to eliminate the need for a long-term predictor. And it uses 10 bits for per sample (7 bits representing a fixed codebook index and 3 bits representing an excitation gain).

2.2.6 G.729 Recommendation

The G.729 recommendation is called Conjugate-Structure Algebraic Code Excited Linear Prediction (CS-ACELP) approved originally by ITU in 1995. It operates at 8 kbps coding rates. Alternatively, another version of the CS-ACELP that reduces complexity is also standardised as Annex A.

The CS-ACELP uses a 10-ms frame plus a 5-ms lookahead that results in a 15-ms algorithm delay. The delay is lower than that of the low-delay coding G.728 recommendation. As for the quality of the speech, the CS-ACELP provides a near-toll quality roughly compatible to the G.721 recommendation.

2.2.7 G.723.1 Recommendation

The G.723.1 recommendation is called Dual Rate Speech Coder for Multimedia Communication Transmitting, approved by ITU in 1996 and later recommended by the International Multimedia Teleconferencing Consortium's Voice over IP Forum as the

default low-bit-rate audio coder for the ITU H.323 standard. It operates at both 5.3- and 6.3 kbps coding rates.

When compared to the other standard recommendations of ITU, the G.723.1 recommendation has a distinct difference with an explanation that frame erasure ability is supported. This is also the reason why the G.723.1 recommendation is robust in handling lost packets during the transmission of packetised voice over the network [8].

Also there is an option within the coder for the developer to further reduce the bit rates by using the VAD technique. If the VAD technique is used, the coder can provide data rate at 2.65 and 3.15 Kbps.

As regards the quality of reconstructed speeches, the coder provides near-toll quality. So it is not hard to conclude that the G.723.1 recommendation is good for Internet videoconferencing, Internet telephony, and multimedia applications. However, everything has two sides. The G.723.1 recommendation is good but not free. The latest version is 5.1 of the G.723.1 (from November 1996), and its source code can be obtained from the ITU Sales Organisation.

In sum, the different ITU recommendations, no matter mentioned above or not, reflect different trade offs between speech quality, bit rate, computer power, and signal delay. And a selection of speech coding schemes is very important for a multimedia application, and so is it for this project.

2.2.8 Speech coding selection for asynchronous AudioGraph

The selection involves a number of areas, and this can be summarised as criteria. Let us briefly look at criteria for selecting an appropriate speech-coding scheme, which can be addressed in six questions (summarised by G. Held) as follows [8]:

- What coding bandwidth does the algorithm require?
- Is the algorithm standardised?
- Do other vendor products support interoperability based on the algorithm being considered?
- Does the algorithm generate high-quality voice or just intelligent speech?
- What is the end-to-end delay associated with the algorithm?
- Is the algorithm suitable for use on packet networks?

In this project, the G.723.1 recommendation looks like the best candidate if the project was a new one to be built from the ground. But it's not the case. There is an existing constraint, which must incorporate the old version AudioGraph into this project. What we should do is to expand the current version of the AudioGraph tool into a new version that supports real-time operations. The old version AudioGraph supported GSM 6.10 speech coding scheme. And another reason is that the G.723.1 is not free but GSM's source code can be obtained from the web site (http://kbs.cs.tu-berlin.de/~jutta/gsm/degener.htm) without costing a penny. Hence it's not hard to conclude that GSM 6.10 scheme would be a candidate. Actually I was told explicitly to use the GSM 6.10 scheme.

2.3 IP and related protocols

Any project relating the transmission of voice over IP networks associates with the Internet Protocols (IP). So that having a degree of knowledge concerning the IP is one of the requirements of this project.

Before 1990, the dominant model concerning data communication and networking was the Open Systems Interconnection (OSI) model standardised by International Standards Organization (ISO). Unfortunately, the TCP/IP protocol suite dominates the commercial architecture because it was used and tested throughout in the Internet [67]. But the TCP/IP protocol suite has a similar structure to the OSI model. Thus, prior to exploring the TCP/IP protocol suite, let us briefly review the OSI model first.

2.3.1 OSI model

Layer Number	Layer
7	Application
6	Presentation
5	Session
4	Transport
3	Network
2	Data link
1	Physical

Figure 2.3.1-1. The 7-layer OSI Reference Model.

The OSI model is a layered framework for the design of network systems that allows communication between all types of computer systems, composed of seven ordered layers shown on figure 2.3.1-1. Each layer defines a segment of the process of moving information across a network.

Within a single machine, each layer calls on the service of the layer just below it. On the other hand, between machines, a layer on one machine communicates with the same layer on another machine.

According to the functionality, seven layers can be thought of as belonging to three subgroups. The first subgroup, the network support layers, is composed of the layers 1, 2, and 3. The second subgroup, the transport layer, is the layer 4. And the third subgroup, the user support layers, is composed of the layers 5, 6, and 7. Let us briefly describe the functions of each layer as follows:

The physical layer. The physical layer deals with the mechanical and electrical mechanisms required for the transmission media, also defining the procedural and functional mechanisms performed for transmission.

The data link layer. The data link layer transforms the physical layer to a reliable link so that a device can gain access to the transmission media. It is also responsible for error control to make sure that the physical layer is error free for the network layer.

The network layer. The network layer is responsible for the source-to-destination delivery of a packet possibly across multiple networks (or links). To accomplish this task, the network layer performs logical addressing, routing, switching, sequencing of data packets, and flow control.

The transport layer. The transport layer is responsible for source-to-destination (end-to-end) delivery of the entire information. The network layer gets each packet to the correct computer, but the transport layer gets the entire information to the correct process on that computer.

The session layer. The session layer establishes, maintains, and synchronises the interaction between two stations on a network. It is also responsible for name-to-station address translation.

The presentation layer. The presentation layer is concerning with the syntax and semantics of the information exchanged between two systems. It provides data transformation, formatting, syntax conversion, and encryption.

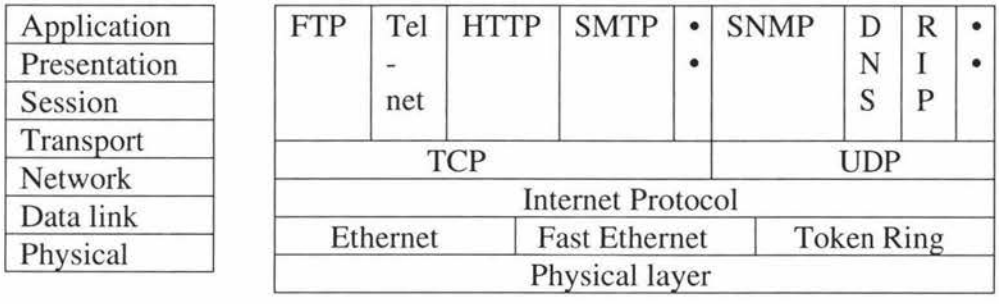
The application layer. The application layer enables the user, no matter human or software, gain the access to all of the services of the model.

This is what the OSI model accomplishes. Even though the OSI model has not been fully implemented, its clear structure still helps people to understand the TCP/IP protocol suite.

2.3.2 TCP/IP protocol suite

The Transmission Control Protocol (TCP) [8] was developed prior to the OSI model. Therefore, logical subgroups in the TCP/IP protocol suite are not exactly the same as those in the OSI model. Actually, there are only five layers within the TCP/IP protocol suite, which are physical, data link, network, transport, and application. Figure 2.3.2-1 gives a clear view of the general correspondence between the TCP/IP protocol suite and the OSI model. As can be seen, the first four layers of TCP/IP correspond to the first four ones of OSI model respectively, but the top-most layer of TCP/IP roughly corresponds to the three top-most layers in OSI model.

The TCP/IP protocol suite is made up of protocols, each of which provides a specific functionality, such as FTP, SMTP, and etc. At the network layer, it defines the main protocol—Internet Protocol (IP). And at the transport layer, it defines two protocols: Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). In the following paragraph, we will examine the IP first and then look at these two protocols in transport layer.



Legend:

FTP	:	File Transfer Protocol
SMTP	:	Simple Mail Transport Protocol
http	:	HyperText Transmission Protocol
SNMP	:	Simple Network Management Protocol
DNS	:	Domain Name Service

Figure 2.3.2-1. Correspondence between TCP/IP and OSI model

IP

The Internet Protocol [8][67][68] is the transmission mechanism used by the TCP/IP protocol suite. It is an unreliable and connectionless datagram (packet) protocol. In another words, it provides a best-effort delivery service, i.e. IP does its best to send packets to the destination, but with no guarantees (does not provide error checking and tracking). Also each datagram is handled independently and may get a different path to reach the destination. So, there is no guarantee that each datagram will arrive at the destination in the order that it was sent.

Actually, the IP protocol provides general functions: basic data transfer, addressing, routing, and fragmentation of datagrams. The IP protocol only sends datagrams from host-to-host, but TCP and UDP do it from process-to-process. And IP addressing

mechanism provides a distinct identity to each device interface, such as host, router, gateway, etc., connected on the Internet, so that each device interface must have their unique identity. Also the IP protocol should provide the route for a datagram. In order to pass through different physical network, datagrams might need to be fragmented.

Now let us briefly look at its header structure shown on figure 2.3.2-2.

VER field. This field stands for version and consists of four bits, defining the version of the IP protocol. The current version is 4, and version 6 that is being testing in part of the Internet will replace version 4 shortly. Version 4 uses 32-bit addresses in each datagram, but version 6 uses 128-bit addresses.

HLEN field. This field stands for the total length of the datagram header and consists of four bits. The length of the header is variable, which ranges in from 20 to 60 bytes. When there is no options, the length of the header is 20 bytes, and the value of this field is 5 ($4\times 5=20$). When the option field is at its maximum size, the value of this field is 15 ($4\times 15=60$).

Service type field. This eight-bit field defines how the routers handle the datagram through the priority mechanism. Three of the eight bits in this field are used to denote the precedence that defines the priority of the datagram in issues such as congestion. If there is congestion and some least important data should be discarded, the datagrams with the lowest precedence are discarded first. Other bits in this field are used to denote the service type.

Total length field. This 16-bit field defines the total length of the datagram.

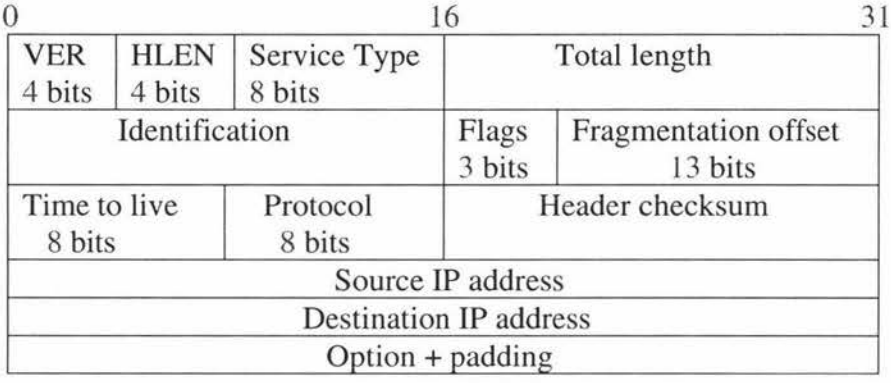


Figure 2.3.2-2 IP Header

Identification and fragment offset fields. The 16-bit identification field enables each datagram or fragmented datagram to be identified. When a datagram is fragmented into small pieces, the 13-bit fragment offset field specifies the offset in the original datagram of the data being transported.

Flags field. This 3-bit field shown on figure 2.3.2-3 only represents 2 flags, and one bit is not used currently. If flag D is set, the datagram will not be fragmented; and if the datagram cannot be passed through available physical network, it will be discarded. If flag M is reset, the datagram is the last or only fragment; otherwise, more fragments are coming.

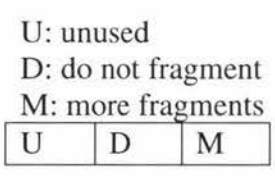


Figure 2.3.2-3. Flags field of IP header

Time-to-live field. This eight-bit field specifies the maximum limited time-life of a datagram when the datagram travels through an Internet. Each router, which processes the datagram, decrements the value of this field by one. When the value of this field is zero, the datagram is discarded. The purpose of using this field is to prevent that a datagram from endlessly wandering the network. For example,

when routers in the Internet became corrupted, a datagram may travel between two or more routers for a long time. This results in resources being tied up without benefit.

Protocol field. This eight-bit field defines the higher level protocol used to create the message carried in the datagram.

Source address and **destination address field**. Each field consists of 32 bits to represent the IP address of source and destination respectively. The address fields must remain unchanged during the time the IP datagram travels from the source to destination host.

The IP protocol gives no guarantee of service, and it only provide transmission between host-to-host. However, to date, most of the computer systems support multitasking, and that means that multiple processes can be running simultaneously. The datagram delivered by IP protocol can only reach host-to-host level not process-to-process level. Obviously this results in the data transmission being incomplete. Hence some protocols should take over the incomplete transmission so as to make sure that the transmission can reach the process-to-process level. Fortunately, the transport layer of the TCP/IP protocol suite supports this. Let us turn our attention to the transport layer.

TCP

TCP [8][67][68] is called a connection-oriented, reliable transport protocol, and it could be thought of as creating a virtual circuit for the transfer of information. Creating the virtual circuit represents a connection establishment. And the transfer of information via TCP is reliable. Prior to introducing the TCP header, let us briefly examine how a connection is set up or terminated between the source and destination host.

TCP is working in full-duplex mode, which means that data can flow in both directions at the same time. Therefore, the connection establishment involves both ends' (client and server) action. Foremost, the server should be ready to accept a connection. Then, a client that wishes to connect to a server makes a request called active open. This starts to build the virtual circuit.

A client sends a request to a server to ask for a connection. The request includes initialisation information about the traffic from the client to server, such as source and destination port numbers, initialisation sequence number, and etc. Next, the server sends back both an acknowledgement to confirm the request of the client and initialisation information about the traffic from the server to client. Finally, the client sends back an acknowledgement to confirm the server's request.

It is very clear that it looks like a three-way handshaking operation. But for the termination of the connection, TCP uses a four-way handshaking approach. To begin with, the client sends a request to the server for terminating a connection. When the server has received that, it sends back a confirmation message first; then if no more data needs to be sent to the client, the server sends a request to the client to ask for termination. Finally, the client sends back a confirmation message. Up to this far, the virtual circuit has been terminated completely.

TCP is a reliable transport protocol and can deliver a stream of data to the transport layer of a destination through the TCP connection, without error, and without any part lost or duplicated. TCP achieves this by using error control mechanisms such as detecting corrupt packets, lost packets, out-of-order packets, and etc.

Finally, let us turn back to the TCP header shown on the figure 2.3.2-4.

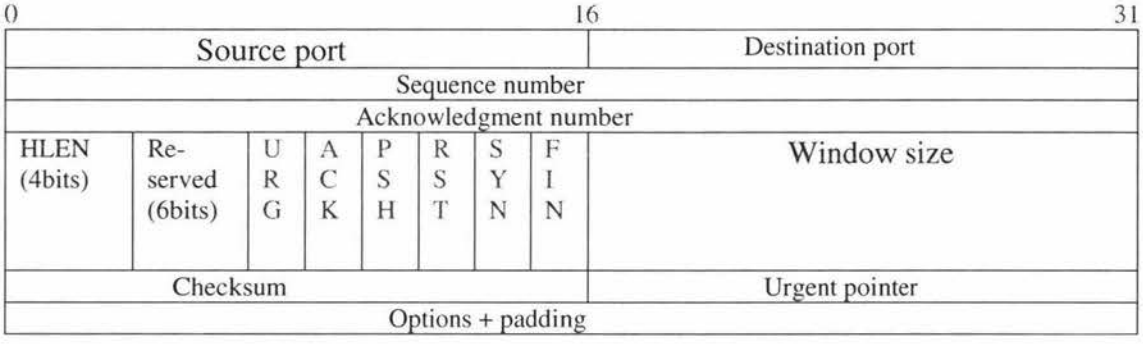


Figure 2.3.2-4 The TCP header

Source and destination port fields. The source and destination port field are each 16 bits in length. Each field specifies a user process or application. The source port field is optional and, when not used, is set to zero. Port numbers at or below 255 are reserved as well-known port number for layer protocols or processes.

Sequence number field. This 32-bit field defines a number assigned to a TCP datagram to indicate the first byte number of a packet. If the SYN bit is set, then this field is ignored.

Acknowledgment number field. This 32-bit field define a number sent from destination to source. The number indicates an acknowledgment of a previously received packet or packets, and also denotes the next sequence number the destination expects to receive.

Data offset field. This 4-bit field denotes how long the TCP header is, and it also indicates where the data begins and the TCP header stops.

Reserved field. This 6-bit field is reserved for future use. Must be 0.

Control bits field. This 6-bit field contains 6 flags. Their function will be shown on the following table.

Flag	Description
URG (urgent)	The value of the urgent pointer field is valid
ACK (acknowledgment)	The value of the acknowledgment field is valid
PSH (push)	Push the data
RST (reset the)	The connection must reset
SYN (synchronisation)	Synchronise sequence numbers during connection
FIN (finish)	Terminate the connection

Table 2.3.2-1 Description of each control bit

Window size field. This 16-bit field denotes the maximum number of blocks of data the receiving device can accept. A large value can significantly improve the TCP performance. The maximum value is 64 k.

Checksum field. This 16-bit field contains an error detection number.

Urgent pointer field. This 16-bit field, which is valid only if the urgent flag is set, is used when the packet contains urgent data.

UDP

UDP is a connectionless, unreliable transport protocol, which adds nothing to the services of IP except for providing

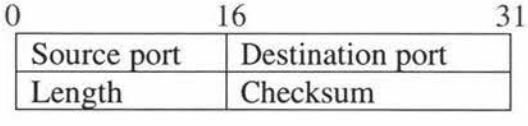


Figure 2.3.2-5 The UDP header

process-to-process communication. It does not provide flow control, and hence no windowing mechanism. At the receiver end, incoming messages might cause overflow. Also there is no error control mechanism in UDP except for the checksum.

Compared with TCP, the UDP does not need to establish a connection before transferring data. So all connection-related functions and flow of control procedures are removed and that removes a considerable amount of overhead that is normally paid for obtaining the reliable capability in TCP. It does add a degree of flexibility to the protocol family.

Let us turn our attention to the UDP header. Actually, it is very simple.

Source and Destination port field. These field works in a similar manner with those in the TCP header.

Length field. The length field, which is 16 bits in length, denotes the length of the UPD datagram including header and user data.

Checksum field. This field is 16 bits in length and is optional; when not used, it is set to zero.

Because the UDP does not provide a reliable service, thus errors such as receiving duplicate data packets might occur. If reliability is required, protocols in a higher layer, such as application layer, can be used to ensure the reliability.

RTP

The Real-Time Transport Protocol (RTP) [68] provides end-to-end data delivery for real-time applications such as interactive multimedia applications. It accomplishes this through payload type identification, sequence numbering, timestamping, and delivery

monitoring. But there is no guarantee for QoS because the actual monitoring of the QoS is performed by the RTP Control Protocol (RTCP), which is a control mechanism.

The RTP is as an application of UDP in IP version 4 but is placed at the transport layer in IP version 6. Usually, an application that employs RTP will run it on the top of UDP or similar protocols. This gives rise to the use of an even port number being compulsory and that the next higher odd port number is reserved for the corresponding RTCP.

Within the RTP, there are two vital services: Translators and Mixers. The translator's function is simple with an explanation that it translates one payload format to another. It enables workstations with different capabilities to participate in real-time delivery of data without requiring all workstations to adjust themselves to work at the lowest common

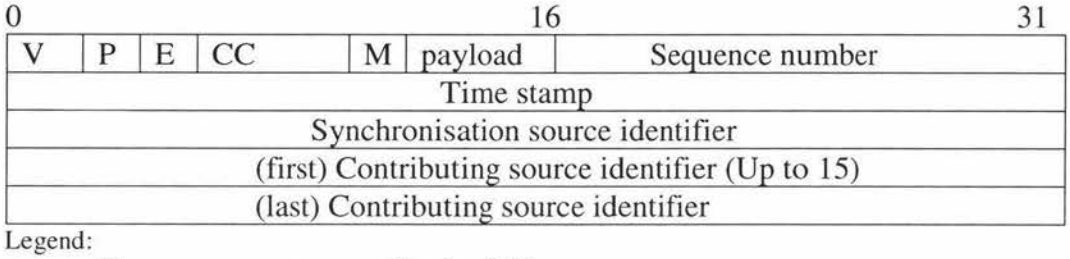
PTV	Audio	PTV	Audio	PTV	Audio
0	PCMU	6	DBI4 (16kHz)	12	TPS0
1	1016	7	LPC	13	VSC
2	G.721	8	PCMA	14	MPA
3	GSM	9	G.722	15	G.728
4	Unassigned	10	LP16 (stereo)	16...22	Unassigned
5	DVI4 (4kHz)	11	LP16 (mono)		

Table 2.3.2-2. Payload type value (PTV) and its corresponding audio format.

denominator in terms of characteristics that govern a real-time communication, such as bandwidth, speech coding schemes, bit rates, etc. The mixers perform a resynchronisation with functions that enable multiple source streams to be combined into one single stream and the original format being preserved. These two services provide a mechanism that enables workstations to receive real-time data streams independently with their capabilities without affecting each other as different capabilities.

To explore the RTP, let us look at its payload types and header structure. The payload type used in RTP may be one of the values shown in the table 2.3.2-2, only the audio-related values are shown in the table. And the header is shown in the figure 2.3.2-6.

Version field. This field, which is 2 bits in length, identifies the version of RTP. A value 2 stands for the current version, 1 stands for the draft RTP version, and 0 stands for the public domain Visual Audio Tool (VAT) [51].



Legend:

V	:	Version field
P	:	Padding field
E	:	Extension field
CC	:	Contributor count field
M	:	Marker field
Payload :	Payload type field	

Figure 2.3.2-6. The RTP Header

Padding field. This is a 1-bit flag that indicates whether padding is used at the end of the datagram. This flag is designed to accommodate the use of encryption that requires a fixed block length or for transporting several RTP packets in a lower-layer protocol data unit.

Extension field. This is also a 1-bit flag that indicates whether the fixed header is followed by an extension header or not. The extension mechanism allows developers to experiment with new payload-format-independent functions that require additional information to be carried in the RTP data packet header.

Contributor count field. This field is 4 bits in length and indicates the number of contributing source identifiers the message contains. There is a maximum of 15 contributors allowed to use in the mixing operation.

Maker field. This field represents a flag, which is 1 bit in length. It is used by applications to mark information in the data that they are transmitting. When audio is transported it would mark the beginning of the speech occurring between two silent periods.

Payload type field. This is a 7-bit field that identifies the format of the RTP payload type. Some of the payload type values are shown in table 2.3.2-2.

Sequence number field. This field is 2 octets in length and indicates a sequence number of each packet. It increases by 1 after a packet has been sent. This provides a mechanism that allows a receiver to detect whether there is a lost packet.

Timestamp field. This field is 32 bits in length and is used to restore the exact timing of a packet as sent from the source. This field provides information to help remove the jitter delay.

Synchronisation source identifier field. This field is 4 octets in length and identifies the synchronisation source. A random identifier will be generated by RTP so that there are no two synchronisation sources within the same RTP session that will have the same identifier.

Contributing source identifier field. This 4-octet field identifies the contributing sources for the payload contained in the packet. There is a maximum of 15 contributing sources that can be added by mixers.

To sum up, RTP supports sequencing and time stamps, synchronises different data streams, and also incorporates additional information in extended header to describe the payload type. That information enables RTP to support multiple compression schemes, such as GSM, G.711, etc. But the RTP does not provide mechanisms that are used to compensate for the loss of a packet or for jitter.

Summary

Currently, most VoIP products use TCP to establish communications between network devices, while UDP is used for the flow of packetised data. Also RTP is widely applied in real-time sensitive applications because it provides mechanisms that can detect the loss of packets and track the order of sent packets to help remove jitter.

2.4 Sound File formats

This project is an audio-relevant application, so all file formats used are capable of storing audio data. Generally file formats used for audio relevant applications can be categorised as two forms: pure audio and non-pure audio file format. The pure audio file formats, as its name suggests, are used to store pure audio information and audio data. However the non-pure file formats can store other information as well, such as video information and data, etc. We have encountered both of these two formats in the project (AIFF and AIFC are pure formats, and AEP is non-pure).

2.4.1 Pure audio format

Historically, almost every type of computer used its own file format for audio data. There are two types of file formats: self-describing formats, where the relevant information and encoding are made explicit in some form of header, and raw formats, where the relevant information and encoding are fixed. The header of self-describing formats usually contains parameters (e.g. a sample rate, number of channels, etc) and other information (e.g. a copyright notice, etc). But in some formats, chunks of data intermingled with chunks of encoding information may be contained as well.

AIFF and AIFC

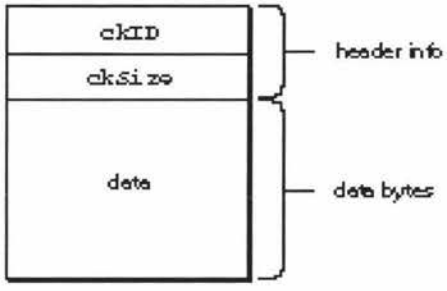


Figure 2.4.1-1. The general structure of a chunk.

AIFF and AIFC [15] stand for Audio Interchange File Format and Audio Interchange File Compressed respectively. Both are originally used for Macintosh. AIFC is the same thing as AIFF except it has built in compression capability. An AIFF file consists of a collection of chunks that define characteristics of the sampled sound or other relevant data about the sound. There are several types of chunk used in the AIFF and AIFC thus far, and table 2.4-1 shows all of them.

Chunk type	Description
Form	Contains information about the format of an AIFF or AIFC file and contains all the other chunks of such a file.
Format Version	Contains an indication of the version of the AIFC specification according to which this file is structured.
Common	Contains information about the sampled sound such as the sampling rate and sample size.
Sound Data	Contains the sample frames that comprise the sampled sound.
Marker	Contains markers that point to positions in the sound data.
Comments	Contains comments about markers in the file.
Sound Accelerator	Contains information intended to allow applications to accelerate the decompression of compressed audio data.
Instrument	Defines basic parameters that an instrument (such as a sampling keyboard) can use to play back the sound data.
MIDI Data	Contains MIDI data.
Audio Recording	Contains information pertaining to audio recording devices.
Application Specific	Contains application-specific information.
Name	Contains the name of the sampled sound.
Author	Contains one or more names of the authors (or creators) of the sampled sound.
Copyright	Contains a copyright notice for the sampled sound.

Annotation	Contains a comment.

Table 2.4.1-1. Chunk types and its description

Before going any further, we had better understand how a chunk is organised. In general, a chunk consists of some header information followed by some data. The header information consists of a chunk ID number and a number that specifies the size of the chunk data. Figure 2.4-1 gives a clear view of a chunk's structure.

The header of a chunk can be represented as a C-like language as follows:

```
struc chunkheader =
{
        ckID:           ID;
        ckSize:         long;
};
```

where ID is a 32-bit concatenation of any four printable ASCII characters in the range ' ' (space character) through '~'. The ckSize field specifies the size of the data portion of a chunk and does not include the length of the chunk header information.

After having understood chunk's structure, let us look at some principal chunks used in the AIFF or AIFC. First one is called the form chunk that defines the characteristics of a sampled sound and contains the actual sound data are grouped together into a container chunk. It defines the type and size of the file and holds all remaining chunks in the file. The chunk ID for this container chunk is 'FORM'. The following is its structure.

```
struc FormChunk =
{
        ckID:           ID;     //'FORM'
        ckSize:         long;
        formType:       ID;     //type of file
```

```
};
```

Where, we should pay attention to the ckSize field, which contains the size of the data portion of this chunk. The data portion comprises two parts: formType and the rest of the chunks of the file, which follow the formType field and is referred as a local chunk. The local chunks can occur in any order in a sound file. And the formType field specifies the format of the file. Its possible value is either 'AIFF' or 'AIFC'.

The second one is called the format version chunk, which is an AIFC-only chunk. Every AIFC file must contain one and only one Format Version Chunk. Its structure is:

```
struc FormatVersionChunk =
{
        ckID:           ID;     //'FVER'
        ckSize:         long;   // constant 4
        timeStamp:      long;   //date of format version
};
```

The third one is call Common Chunk, which defines some fundamental characteristics of the sampled sound contained in the file. Both AIFF and AIFC file must contain a Common Chunk. But the structure of a Common Chunk is different for AIFF and AIFC file.

Common Chunk structure for AIFF file is:

```
struc CommonChunk =
{
        ckID:                   ID;     //'FVER'
        ckSize:                 long;   // size of chunk data
        numChannels:    short;  // number of channels
        numSampleFrames:        long;   // number of sample frames
        sampleSize:             short;  // number of bits per sample
        sampleRate:             long;   // number of frame per second
```

```
};
```

Common Chunk structure for AIFC file is:

```
struc ExtCommonChunk =
{
        ckID:                   ID;     //'FVER'
        ckSize:                 long;   // size of chunk data
        numChannels:    short;  // number of channels
        numSampleFrames:        long;   // number of sample frames
        sampleSize:             short;  // number of bits per sample
        sampleRate:             long;   // number of frame per second
        compressionType:        ID;     // compression type ID
        compressionName:        char[]; //compression type name
};
```

The fourth one is called the sound data chunk, which contains the actual sample frames that make up the sampled sound. The sound data chunk has this structure:

```
struc SoundDataChunk =
{
        ckID:           ID;     //'SSND'
        ckSize:         long;   // size of chunk data
        offset:         long;   //offset to sound data
        blockSize:      long;   //size of alignment blocks
};
```

For details of those fields that exist in the structures, you can find them on Apple's web site (http://developer.apple.com/techpubs/quicktime/qt4beta/INMAC/SOUND/imsoundmgr.30.htm), so we will not discuss them here. But we seem to forget to talk about sample-sound data, let us turn to it.

Sampled-sound data is made up of a series of sample frames, which are stored contiguously in order of increasing time. Each sample frame contains either one or more sample points or packets, depending upon whether the sound data are non-compressed or compressed.

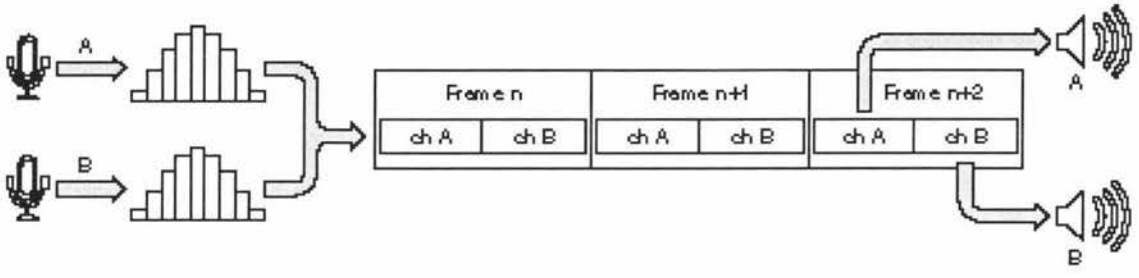


Figure 2.4.1-2. Interleaving stereo sample points [1]

Figure 2.4.1-2 gives us a clear example of interleaved sample points in 2 channels. So for multi-channel sounds, a sample frame is an interleaved set of sample points or packets. And what is the value of each sample point of non-compressed sound data in a sample frame for sound files? It is a two's complement value. Currently Apple only supports 8-bit and 16-bit sample points. So the value of each sample point is between 127 and –128 or between 32767 and –32768.

WAV#49

As a subtype of RIFF files defined by Microsoft, WAV files are broadly adopted by many different operating systems because it can store many different audio formats. There is an interesting WAV format, WAV#49, which stores audio files in the shape of GSM parameters, but its structure of parameter arrangement within each frame is different from GSM. Each frame of WAV#49 contains 65 bytes, but in GSM, there are 33 bytes of which only 32.5 are used. WAV#49 represents 40 ms for each frame of 65 bytes and GSM represents only 20 ms for each frame of 33 bytes. This WAV#49 format is

being used on the AudioGraph project because it provides compatibility with PC computers. This format provides a very good compression rate of around 10:1 and also good quality for speech.

MP3

MPEG Audio Layer-3 (MP3) is an audio compression file format, invented by the Fraunhofer IIS-A in 1987 [69]. Using MP3, CD quality sound data can be compressed by a factor of 12, without losing sound quality. That means proximately 120 Kbit of audio data can represent one second of stereo music in CD quality. MPEG audio use perceptual audio coding and psycho-acoustic compression to remove all superfluous information so that data reduction is achieved without sacrificing the sound quality [70].

In the MEPG audio family compression levels have increased over time. Layer 1 used a factor 4, layer 2 used a factor from 6...8, and now layer 3 uses a factor from 10...12. These compression factors used in the different layers make the compress the sound but still maintain the original CD quality. Moreover, MP3 even can produce a reasonable sound quality for stereo signals using a reduction of 1:24. Table 2.4.1-2 shows qualities MP3 provided with a relationship between bitrate, mode and reduction rate. The ITU recommends MP3 (ITU doc. BS.1115) as a low bit-rate audio coding schemes in broadcast applications at bitrates of 60 kbit/s per audio channel [69].

Quality	Bit rate (kbps)	mode	Reduction rate
Telephone sound	8	Mono	96:1
Better than AM radio	32	Mono	24
Similar to FM radio	56...64	Stereo	26...24:1
Near-CD	96	Stereo	16:1
CD	112...128	Stereo	14...12:1

Table 2.4.1-2. MP3 qualities

Let us look at its details now. Figure 2.4.1-3 shows a diagram of MP3.

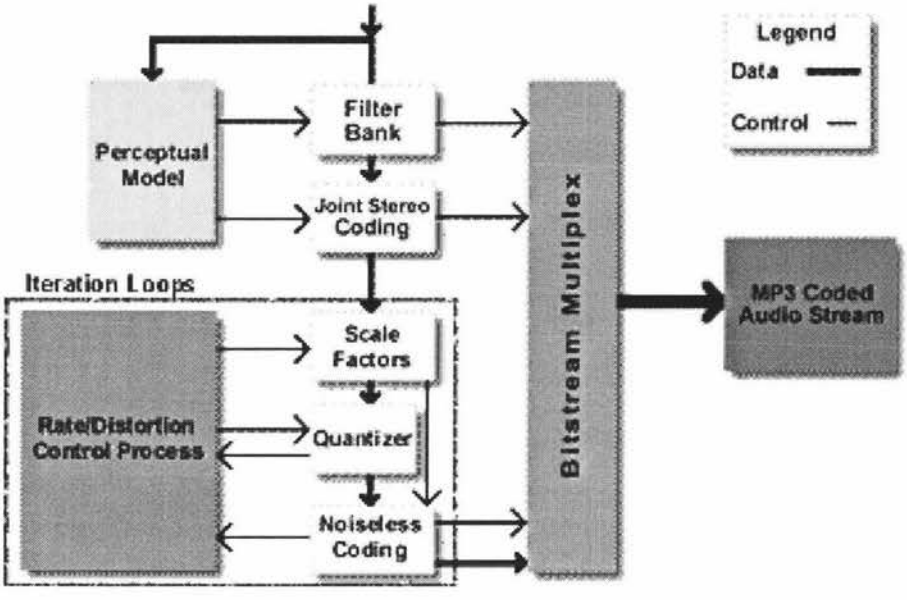


Figure 2.4.1-3. Diagram for MP3 [69].

The input of the coding system, the PCM audio signal, is converted from the time into a time-frequency domain, and this is done by a polyphase filter bank, which consists of 32 subbands [71].

The filter bank in layer 3 is a hybrid filter bank made up of polyphase filters and a Modified Discrete Cosine Transform (MDCT). The MDCT is used to split every subband into higher resolution frequency lines, operating on 18 sub band samples in each subband [71].

The perceptual model dominates the quality of a given encoder implementation, using either a separate filter bank or combining the calculation of energy values and the main filter bank. The values for the masking thresholds or the allowed noise for each coder partition are generated by the perceptual model. The coder partitions are frequency ranges and the thresholds are based on human perception limits for different frequencies.

Joint stereo coding takes advantage of the fact that both channels of a stereo channel pair contain significant information that is the same. "These stereophonic irrelevancies and redundancies are exploited to reduce the total bitrate. Joint stereo is used in cases where only low bitrates are available but stereo signals are desired [69]."

Quantization is done via a power-law quantizer. In order to provide better coding efficiency, non-uniform quantisation, adaptive segmentation, and entropy coding of the quantised values are employed. And the quantised values are encoded by a lossless scheme—Huffman encoding.

To find the optimum gain and scalefactors for a given frame, bit-rate and outputs from the perceptual model, the analysis-by-synthesis (ABS) technique is applied by two nested iteration loops: inner iteration loop (rate loop) and outer iteration loop (noise control/distortion loop). The inner loop works in the following manner: "The Huffman code tables assign shorter code words to (more frequent) smaller quantized values. If the number of bits resulting from the coding operation exceeds the number of bits available to code a given block of data, this can be corrected by adjusting the global gain to result in a larger quantization step size, leading to smaller quantized values. This operation is repeated with different quantization step sizes until the resulting bit demand for Huffman coding is small enough. [69] " The loop modifies the overall coder rate until it is small enough.

And the outer loop works in the following manner: "To shape the quantization noise according to the masking threshold, scalefactors are applied to each scalefactor band. The system starts with a default factor of 1.0 for each band. If the quantization noise in a given band is found to exceed the masking threshold (allowed noise) as supplied by the perceptual model, the scalefactor for this band is adjusted to reduce the quantization noise. Since achieving a smaller quantization noise requires a larger number of quantization steps and thus a higher bitrate, the rate adjustment loop has to be repeated every time new scalefactors are used. In other words, the rate loop is nested within the noise control loop. The outer (noise control) loop is executed until the actual noise

(computed from the difference of the original spectral values minus the quantized spectral values) is below the masking threshold for every scalefactor band (i.e. critical band). [69]"

MP3 files are small. They are currently widely used on the Internet to transfer music data because MP3 provides CD quality.

2.4.2 Non-pure audio formats

AEP format

The AudioGraph Episode Presentation (AEP) format is originally used for the AudioGraph tool. Within this format, each AudioGraph Presentation is a series of episodes, each of which is a separate file. An episode file consists of a header and a body, followed by the control sum. Both the header and body are sequences of records where all records share the same structure:

- Total record length in bytes, all fields inclusive (four bytes)
- Record type (byte)
- Data fields

Record type	Data fields
'L' (length of episode)	• *Number of header records (4 bytes)* • *Number of body records (4 bytes)*
'W' (window aatributes)	*1. Width (2 bytes)* *2. Height (2 bytes)* *3. Colour (4 bytes)*
'S' (Audio stream descriptor)	*1. Stream ID (2 bytes)*

	2. *Audio format (byte):* • *Linear samples* • *MuLaw* • *GSM* 3. *Sample rate (Hz) (4 bytes)* 4. *Number of bits per sample (byte).* *Ignored for the GSM format.*

Table 2.4.2-1. Header records relevant with this project

Record type	Data fields
'z' (Pause)	*Duration in milliseconds (4 bytes)*
'a' (Audio)	1. *Audio stream ID (2 bytes)* 2. *Normalisation factor (byte)* 3. *Duration in milliseconds (4 bytes)* 4. *Audio data*

Table 2.4.2-2. Body records relevant with this project

Within an AEP file, there can be several virtual audio streams in an episode. Each audio stream has its own properties: compression method, sampling rate, and etc. If an audio patch belongs to a steam, it means the patch has the same properties as the stream.

The relevant header and body types with this project are shown on table 2.4.2-1 and 2.4.2-2. The AEP file format also supports images and vector graphics as well, but it is out of scope for our interests.

2.5 Macintosh programming and PowerPlant

We intend to implement this project in Macintosh platform that is because the current version of the AudioGraph tool is implemented in Macintosh. Hence understanding Macintosh programming [36] is very important. However this topic involves a large number of issues so it is difficult to discuss in detail. What we will discuss here are only those issues that are relevant with this project's implementation.

To begin with, we would like to talk about the memory management issue. Most of us might have programming experiences with PC or Unix platform. However, in Macintosh, the way that the operating system organises and manages memory is different with that of those platforms.

When the Macintosh operating system has been set up, it divides the memory into two partitions: a system partition, which is remained for the use of the operating system, and an application partition, which is remained for the use of applications. When an application is launched, the operating system allocates for it a partition of memory (application partition). This is shown in figure 2.5-1. The application partition is divided into three major blocks: a stack, a heap, and the application variables and A5 world. The arrows in different colour show the direction of which the stack or the heap is expanding respectively. In general, the application only uses memory contained in its own application partition.

The second thing we would like to mention is the event handling. It is somewhat difference than that in Windows. Conceptually, the Windows operating system acts in an "active" role that pushes events into each application's message queue, but the Macintosh operating system acts in a "passive" role-- that is—each Macintosh application's event loop must pull its events from the operating system.

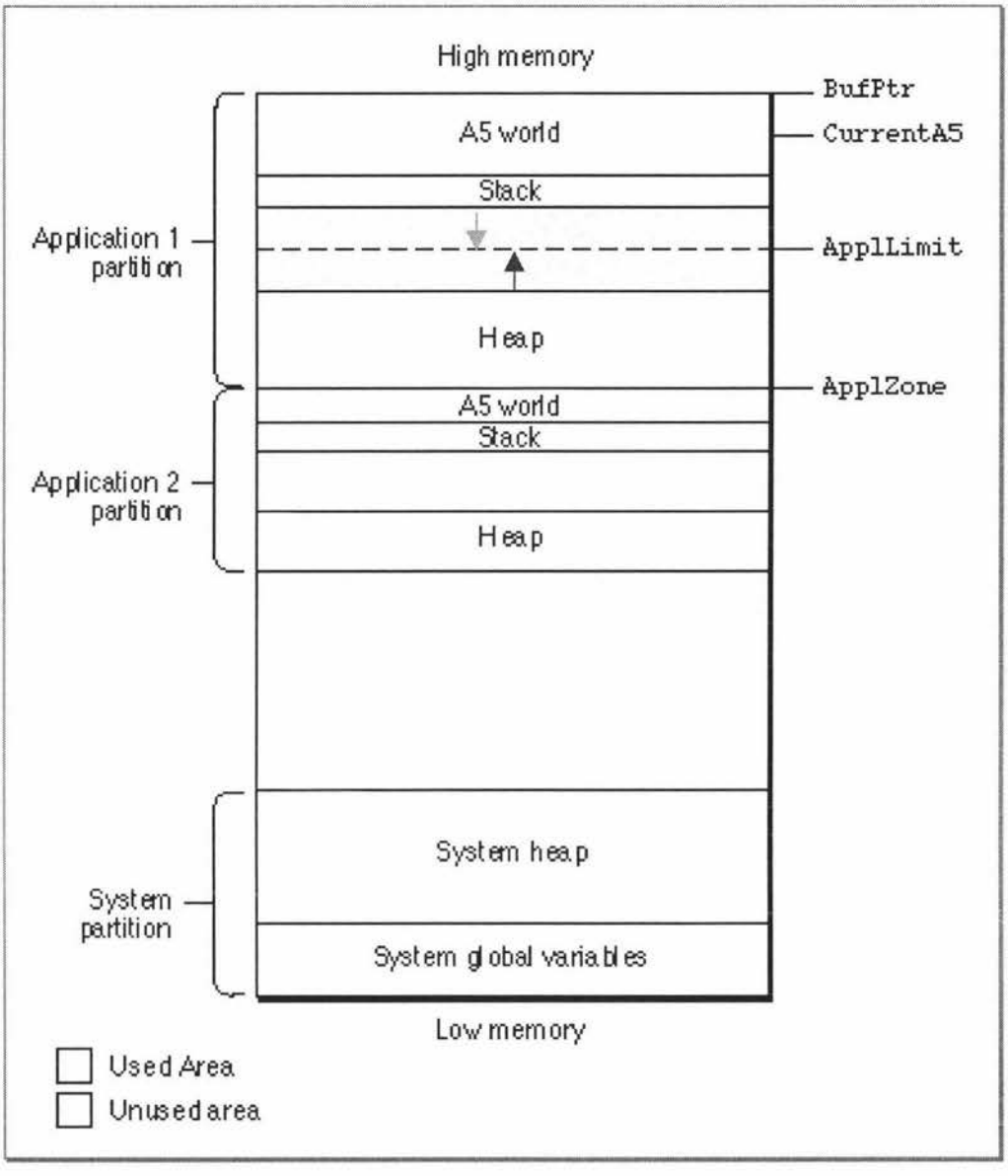


Figure 2.5-1. Memory organisation with two applications open

In Macintosh, every application has an event loop, which is that piece of code in an application that processes and responds to user actions and other events. Through the event loop, the developer can let his/her application communicate with other applications in order to request services or information from another application or to provide services to other applications. Macintosh operating system provides an event manager to help the developer to retrieve information about the actions of the event [59].

The last thing we would like to say is the cooperative multi-tasking. The Macintosh operating system provides a cooperative multi-tasking environment that allows users to switch between many open applications and that allows applications to receive available processing time when other applications are not using the processor. For example, an application, which is not busy currently, can yield control of the CPU or yield process time, to other processes at very specific times. This is done when you call the Event Manager functions WaitNextEvent or EventAvail. As long as any events are pending for an application, it continues to receive processing time.

The cooperative multi-tasking in Macintosh is somewhat different than in other operating system like Windows or POSIX. These operating system support timeslicing, which means that the operating system will switch the control of CPU between processes or threads in a period of time. But in Macintosh (especially in PowerPlant) the event loop and cooperative multi-tasking environment are mechanisms that are used to devote processor time to threads (or processes, or tasks).

No matter what kind of operating systems it is, switching control of CPU causes that a context switch occurs. A cost of context switch depends upon what kind of executing unit is running. In Macintosh, there are two types of context switch: a major switch and minor switch. A major switch occurs when the Process Manager switches the context of the foreground process with the context of a background process and brings the background process to the front, sending the previous foreground process to the background. But a minor switch occurs when the Process Manager switches the context of a process to give time to a background process without bringing the background process to the front.

However, in Macintosh, there is a special kind of task, which is an interrupt task. An interrupt is a form of exception, an error or special condition detected by the processor in the course of program execution. Interrupt tasks are those routines that are executed as a

result of an interrupt. The execution of an interrupt task is not tied to the normal execution of the application, that means that the interrupt task might continue to be executed even when the application is not itself executing. Therefore, when invoking interrupt tasks, the developer must pay considerable attention to the follows:

- Before having determined the application's context is valid, the interrupt code must avoid calling traps that access application-specific system global variables.
- The interrupt tasks should never directly or indirectly cause memory to be allocated, moved, or purged.

Interrupt tasking is invoked in the implementation. To understand that is not difficult but important. Actually interrupt tasking has no difference with other tasks except it depends upon an entry point (a callback function), which specifies the target task, to achieve switching the control of CPU.

Taking advantages of the features of Macintosh operating system significantly increases programming productivity. The Macintosh operating system programming framework has already provided lots of interfaces that allow the designer to use the operating system's utilities, such as Sound Manager, Memory Manager, Process Manager, and so forth, without building everything from the ground. Now let us turn our attention to the C++ framework PowerPlant.

PowerPlant [29] is a Metrowerks' object-oriented, pure C++ language application framework for Macintosh, and it also takes advantages of the features of Macintosh operating system. What is an application framework? An application framework is an object-based framework whose problem domain is application programming, providing generic solutions to the interface elements of most applications, which are fairly standard, and to the flow of control in an application. So to do programming in PowerPlant is very simple because what a designer needs to do is only to add the content to a complete,

powerful, ready-to-run program. Having said this there is a steep learning curve to overcome before any programming can be done.

PowerPlant provides families of classes that cover most of the object-oriented techniques. One of the most important features of PowerPlant is that multiple inheritance has been widely used. A large set of classes have been designed to satisfy one purpose, that is to be mixed into other classes by multiple inheritance, such as LAttachment (describes an object that –typically-- alters the run-time behavior of another object), LPane (describes a rectangular object that can display graphics), LPeriodical (describes an object that receives time on a regular basis), and so forth. This makes the entire framework easier to understand. Multiple inheritance is one of the design principals of PowerPlant.

PowerPlant provides a set of thread-related classes. This facilitates the use of the Macintosh's cooperative multi-tasking. Actually, they are not difficult to use and for further detail the reader is referred PowerPlant advance topic documentation.

In PowerPlant, PowerPlant code runs the main event loop, gets each event, figures out what kind of event it is, and dispatches it to the appropriate PowerPlant object. What the developer should do is simply to structure the application so that it can respond to events and should yield time to other applications when it is not busy. This enables the user can perform tasks in any order.

In additon, PowerPlant provides timing control, which is achieved by periodical. An object that receives time on a regular basis is called a periodical in PowerPlant terminology. More specifically, PowerPlant deals with two kinds of periodicals: repeaters that get time after every event, and idlers that get time at every idle event. The periodical leaves chances to threads themselves to do whatever they want at specific time without depending upon the operating system to devote time. The periodical is one of the PowerPlant's distinct characteristics.

2.6 VAD algorithm used on synchronous AudioGraph

The VAD algorithm used on the synchronous AudioGraph is a simple energy threshold scheme. The algorithm provides a way to detect whether each point in the speech is active speech or inactive speech. At any time, we can only consider that a point is active or inactive. If the speech has m samples, there are m time comparisons. But the quantity, which determines whether there is a triggering point, needs to be recalculated each time. Fortunately, we have found a simple way to do it.

Let us make assumptions as follows:

- The speech contains m samples, and its duration is $t_i = t_{i-1} + \Delta t$, where $0 \le i \le m-1$ and $\Delta t = \frac{t_m}{m}$.
- The speech can be represented as an amplitude array $s[k]$, where $k = 0,1,...,m\text{-}1$.
- ω samples will be considered at a time, these ω samples constitute a calculating window called F or we can say the length of the window is ω, where $0 \le \omega \le m$.
- The sum of the energy of the window F at time t_0-- its initial state -- is zero, and all $s[j] = 0$, where $0 \le j \le \omega$.
- And the threshold percentage is ρ.

Before looking at the algorithm, we need to say something about how to calculate the size of the calculating window. Within the implementation, we used a parameter based on time to represent the length of the calculating window; and we will normalise speech first before applying the voice activity detector, i.e., convert the speech into 8kHz, 16 bit, and mono. Let us denote the size of the window is S_{cf} in samples, and a time-based-parameter window length, is called L_{time} in milliseconds. This is the relationship between L_{time} and S_{cf}.

$$S_{cf} = L_{time} * 8000 \quad (1)$$

Ok, it is time to look at the algorithm.

Given a window F at time t_i, let us denote the sum of the energy of the window F at time t_i is SUM_{t_i},

$$SUM_{t_i} = \sum_{i=i_0}^{i_0+\omega} |s[i]| \quad (2)$$

and (2) could be expressed as (3),

$$SUM_{t_1} = |s[i_0]| + \sum_{i=i_1}^{i_1+\omega} |s[i]| \quad (3)$$

given a window F at time $t_{i+1} = t_i + \Delta t$; let us denote the sum of the energy of the window F at time t_{i+1} is $SUM_{t_{i+1}}$,

$$SUM_{t_{i+1}} = \sum_{i=i_1}^{i_1+\omega+1} |s[i]| \quad (4)$$

and (4) is equivalent with (5),

$$SUM_{t_{i+1}} = \sum_{i=i_1}^{i_1+\omega} |s[i]| + |s[i_{\omega+1}]| \quad (5)$$

when time varies from t_i to t_{i+1}, we can simply obtain the sum of the energy of the window F at time t_2 by subtract (5) and (3).

$$SUM_{t_{i+1}} - SUM_{t_1} = |s[i_{\omega+1}]| - |s[i_0]| \quad (6)$$

from (6), we have got (7),

$$SUM_{t_{i+1}} = |s[i_{\omega+1}]| - |s[i_0]| + SUM_{t_1} \tag{7}$$

if $t_{i+1} \leq \omega$, according to the previous assumption, we have got (8),

$$SUM_{t_{i+1}} = |s[i_{\omega+1}]| + SUM_{t_1} \tag{8}$$

from (7) and (8), we can see that it is very easy to recalculate next quantity, which determines the point at $i_{\omega+1}$ is active or inactive.

For instance, let us assume the length of the calculating window is 101. The 101-th quantity calculation can be showed by figure 2. So 102-th quantity is calculated by formula $|s[102]| + SUM_{t_{101}} - |s[1]|$; the 103-th quantity is $|s[103]| + SUM_{t_{102}} - |s[2]|$; and so on.

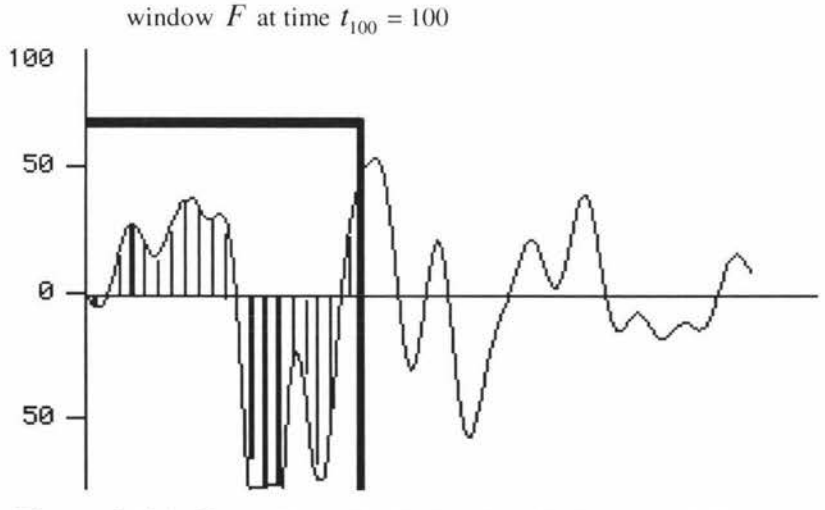


Figure 2.6-1. Quantity calculating in window F at time t_{100} = 100.

Thus far, the determining quantity for each point can be computed in a very simple way.

However, it is important that we should use a buffering technique to maintain the speech before and after the active and inactive triggering points. These situations are shown in figure 2.6-2.

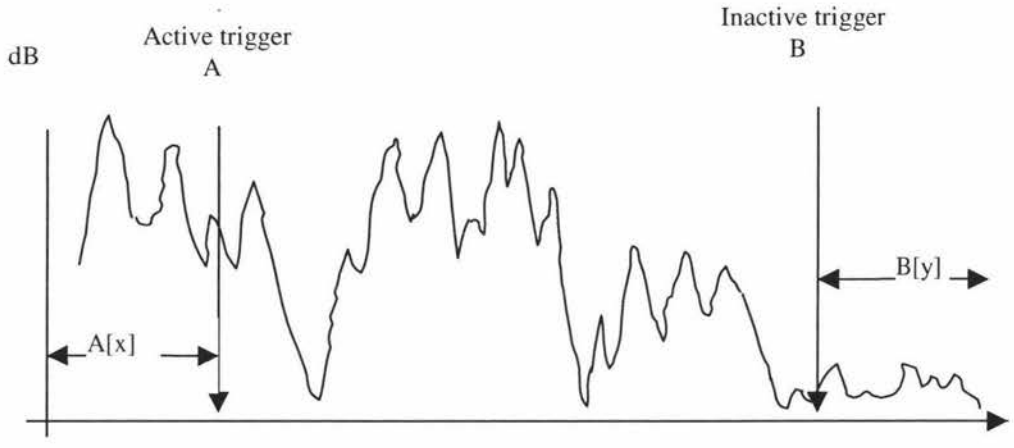


Figure 2.6-2. Situations where buffer is required

In figure 2.6-2, we assume that, an active trigger point would appear at point A, an inactive trigger point would appear at point B. A buffer array A[x] represents sample points have not been considered as active speech because active trigger point A has not been reached. And another buffer array B[y] represents on-coming samples after an inactive trigger point B has been reached. Therefore, we used a buffer called would-be-speech buffer to keep track of array A[x]. On the other hand, in order to obtain complete speech, we should not chop the on-coming active speech immediately after the inactive trigger point has been reached. So we used another buffer called would-be-speech-dribbles buffer to keep track of on-coming speech samples B[y].

And the length of those buffers is calculated as the same as the length of the window. According to formula (1), it is very easy to represent these two buffers by time-based parameters.

To summarise this algorithm, the way of re-calculating of the energy quantity is simple and straightforward, and it is linear without high computational cost. Finally a buffering mechanism must be adopted in order to separate active/inactive speech successfully. Clearly, varying the size of those buffers might affect the qualitative performance of the VAD.

Chapter 3 Implementation Issues

The experimental work of this project consists of two parts (two subsystems): System-A and System-B (defined in section 1.2). System-A focuses on capturing the voice on the fly, and generates an AEP file, which is composed of GSM-compressed sound components and pauses, for later playback by the current version of the AudioGraph player. System-B focuses on an IP connection and provides two-way voice over IP. Specifically, System-B reuses System-A's automatic voice capturing functional blocks to capture speech, then transfers the recorded speech across the network, and provides a real-time player to playback the data received. These two subsystems are implemented in C++ on the Macintosh.

3.1 System-A

3.1.1 Objectives

The primary purpose of the System-A is to record GSM-compressed sound on the fly and to further reduce the bit rate using the VAD technique. After having recorded a speech signal in PCM form, System-A applies the VAD scheme to produce only the active speech in GSM format. The active speech will be packetised into a number of sound components separated by pauses representing the non-active period. The other purpose of the System-A is to store the packetised speech into an AEP file for later playback. Within that AEP file, each sound component is a segment of the active speech, and the pauses have been added to represent the real silence period in the original speech. Hence, the System-A must maintain information, which is used for the time-pause parameters.

The VAD scheme we have developed is the core of this part of the project, and packetising the voice also provides preparation for the system B.

3.1.2 Architecture

System-A's architecture is a multi-tasking architecture as shown in figure 3.1.2-1. Two threads (thread-A and thread-B) are invoked. Wherein, thread-A handles the entire recording operation, and thread-B does the speech processing and serialisation. The communication between these two threads is only one direction, which means that thread-A will trigger thread-B. A restriction is that thread-B must finish earlier than thread-A does for a given segment of speech. This happens if the computer used has a high enough processing speed. After thread-B has finished, thread-A is still recording. Therefore what thread-B can do at this stage is just to wait for the next input from thread-A.

In the general case (platform independent), this architecture can be explained as the following scenario:

Thread-A starts recording for a specific amount of time then it yields the control to thread-B. If thread-B has data to be processed, thread-B works for a specific amount of time then yields; otherwise (thread-B has no data available yet or has finished), thread-B just simple yields or sleeps a little while.

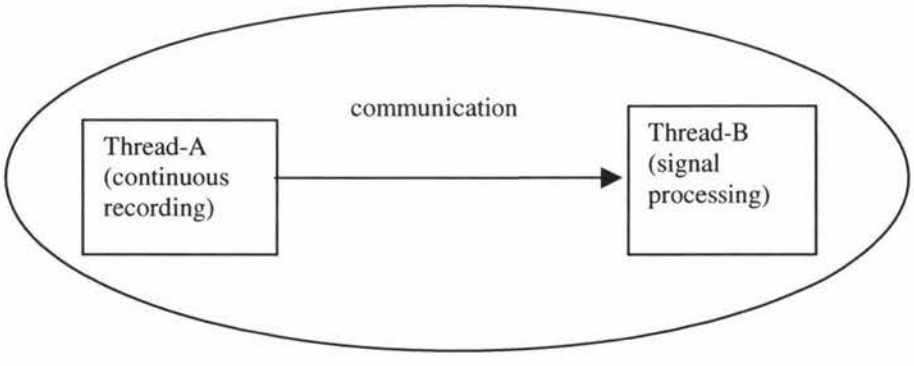


Figure 3.1.2-1 An architecture of system-A

As regards the actual implementation (on the Macintosh platform), it is a little bit different to the general case. Communication between thread-A and thread-B is achieved by an interrupt task. When the recording buffer has been filled, thread-A switches buffers. Then it starts the next recording action and calls thread-B. This is all done within a recording interrupt callback routine, **InterruptCompletionProc**, which is a completion callback routine (it is an entry point to be called when a recording has filled the specified buffer). Within thread-A the recording process is handled by the peripheral equipment and thread-B consequently has most of the processor resource. As shown in figure 3.1.2-2, the peripheral equipment records the speech into its internal small buffer. When this is done, it provides an interrupt callback routine for us to capture or modify this data. If we do not, it stores this data into the buffer we have specified when we started the recording. When the specified buffer has been filled up, the completion callback routine will be called so as to trigger thread-B to start the signal processing. In the

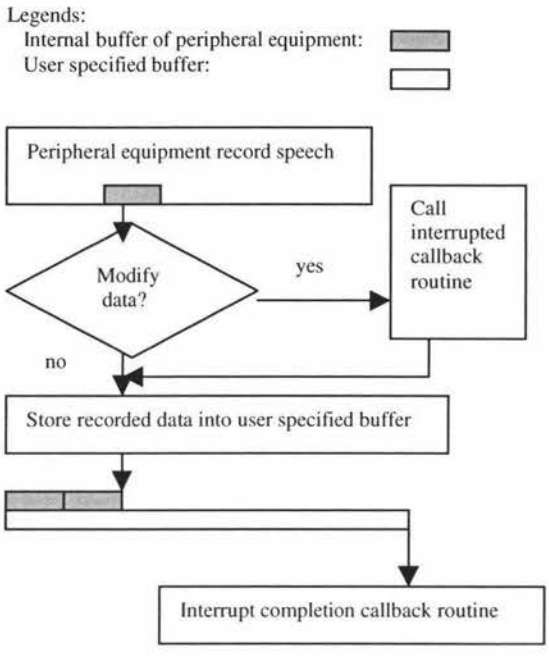


Figure 3.1.2-2. Detail of recording process

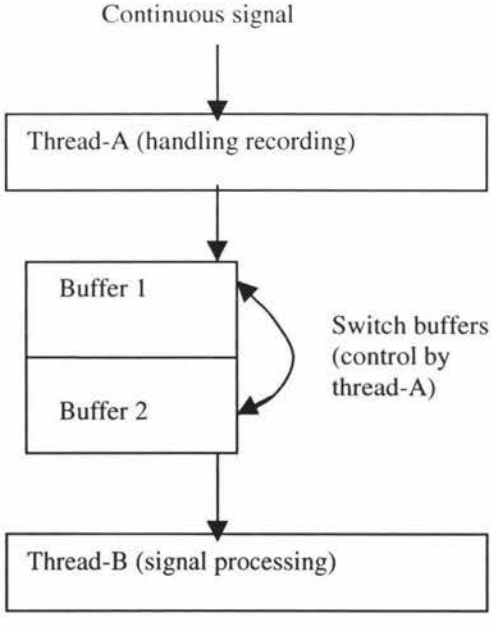


Figure 3.1.2-3 The relationship between buffers and threads

implementation, we make the buffer ten times larger than the internal buffer of the peripheral equipment.

Even though two threads are working together, they do not share any data. Hence we do not need any synchronisation mechanism here. The reason is that we devote two buffers for two threads, so that each of threads can only use one buffer at a time, and that there is no case for two threads sharing a buffer. As shown in figure 3.1.2-3, we can see the relationship between buffers and threads. Clearly there is no need to provide a synchronisation mechanism. We just have the constraint that thread-B must finish earlier than thread-A does. This suits for the general case as well providing the microprocessor of the computer used has a high enough processing speed.

3.1.3 Application

In this section, we would like to introduce the operations of System-A.

As have been seen in figure 3.1.2-3, thread-A only does two things: one is to record the original speech signal as digital signal (PCM form), and two is to prepare buffers for the next recording and for subsequent signal processing. The buffer used here is an internal ring buffer, which guarantees that very limited size is required. The buffer is designed to record one-second of speech (sometimes the buffer is a little bigger or smaller than one second buffer because we make it a multiple of the internal device buffer). For example, if the user is willing to record speech at 22 kHz, 16 bits, and stereo, the total size of the internal ring buffer is 160 kbytes. The size of the buffer is determined by the interests of the user (the higher the sample rate the user chooses, the bigger the buffer size is required). In the implementation, the ring buffer is divided equally into two small buffers. In the previous example, each small buffer is 80 kbytes. Therefore, when one small buffer is being filled by the recording the other is being processed to generate GSM compressed sound of the active speech only.

One thing we would like to make it clear here is that, thread-A is doing its best to capture the input signal, and the signal processing is done within thread-B. The buffers (the ring buffer) are used to store recorded signal and to provide the input to thread-B. We have provided other buffers within the thread-B to handle the signal data, which are before an active trigger point or after a non-active trigger point. This will be discussed later in this section.

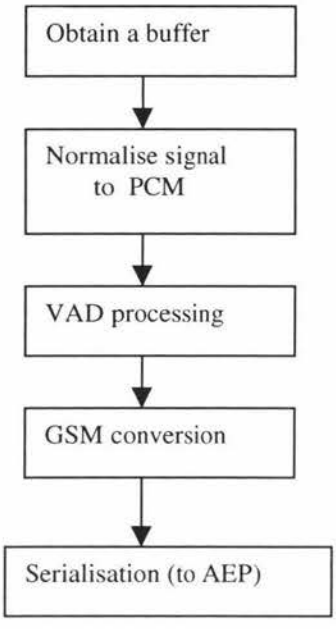


Figure 3.1.2-4. The block diagram of thread-B

The block diagram of the thread-B is shown in figure 3.1.2-4. As can be seen, it does a little bit more than the thread-A. Thread-B takes a smaller time to process the buffer than thread-A does to fill it (on a Macintosh G3 computer with a maximum speed at 400 MHz).

The input to the thread-B is the recorded signal of PCM form stored in a small buffer. The normalisation process is required because the original speech might be recorded at any user specific rates. What we should do here is to convert the original speech into 8 KHz, 16 bit, and mono PCM form, for conversion to GSM.

The VAD process has been described in previous chapter (In section 2.6). It removes all redundant silences within the speech signal. The output of this process is a list of packets if there has been any active speech within the original speech; otherwise, the list is an empty list. Information such as the length of the active speech packet, and the length of silence period is maintained in the list.

The VAD algorithm used in the implementation is presented in figure 3.1.3-1 and figure 3.1.3-2.

Definitions:

Last: Represents time Last, when signal s[0],s[1],...s[Last] has been processed. (Let us assume s[] represents the signal.)

m: The size of the window.

n: The size of the input buffer.

RecordingState: This flag indicates whether we should record or not.

OutputBuffer: This buffer contains all active speech samples, which are come from the input buffer.

Window: Represents a portion of signal in a buffer, which is being processing. Initialised as a zero-filled buffer. At any time k, the sum of the energy of the window is calculated by the following:

$$SumWindow_k = \begin{cases} \sum_{i=0}^{k} |s[i]| & (\text{if } 0 \le k < m) \\ \sum_{i=0}^{m-1} |s[k-i]| & (\text{if } m \le k) \end{cases}$$

$Window_0$: ($s'[]$ represents the signal as an array of samples in the window.) Represents the first signal sample in the window and is presented by the following:

$$Window_0 = \begin{cases} s'[m-k] & (\text{if } 0 \le k < m) \\ s'[0] & (\text{if } m \le k) \end{cases}$$

WouldbeSpeechBuffer0: Stores samples before the active trigger point.

WouldbeSpeechBuffer1: Stores samples after the non-active trigger point.

CurrentActiveList: Preallocated active List; It has never been used before.

Figure 3.1.3-1. Definitions for the VAD algorithm

In the VAD algorithm, the CurrentActiveList only contains packet relevant information such as the pause length, length of the active speech, etc., but it does not include any signal data. So it is only a partial list. As shown in figure 3.1.3-3, the actual signal is still in the OutputBuffer as a whole continuous signal without inactive speech. This provides convenience for the implementation because there is a memory issue that should be solved. If you want to make things clear and do not care about doing more operations, you can add a signal into CurrentActiveList directly.

```
SignalProcessing (SumWindow_Last, CurrentActiveList, s[])
/* s[] represents the input signal in the input buffer */
{
 /* since LastSum will be updated after it  has been used each time*/
 Set LastSum as SumWindow_Last;
 Set the first node of the CurrentActiveList to Node;
 /* consider each sample  */
 For p from 0 to n do
 {
        Get Window_0;
        Add the current sample s[p] into the Window;
        Add the current sample s[p] into the WouldbeSpeechBuffer0;

        if (RecordingState is recording){
                Output the current sample to the OutputBuffer;
                Count the active speech length in current Node;

                if (Not IsActiveTriggerPoint (LastSum, Window_0, s[p]) ){
                        Add s[p] into WouldbeSpeechBuffer1;
                        if (WouldbeSpeechBuffer1 is full)
                        {
                                Set RecordingState to non-recording;
                                Add all samples in WouldbeSpeechBuffer1 to OutputBuffer, update the active
                                speech length information in Node and reset this buffer to its initial state;

                                Update CurrentActiveList information; /* such as pause length in millisecond,
                                Containing valid information, and etc */
                                Reset pause counting;
                                Set Node point to next node of the CurrentActiveList;
                        }
                }else
                        /* if we suddenly have active trigger point  again */
                        Reset WouldbeSpeechBuffer1;
        }
        else
        {
                if (IsActiveTriggerPoint (LastSum, Window_0, s[p]) )
                {
                        Set RecordingState to recording;
                        Add all samples in WouldbeSpeechBuffer0 to OutputBuffer;
                        Reset WouldbeSpeechBuffer0;
                        Update Node information; /* such as the length of the active speech */
                }else {
                        Update pause counting information;
        }
        }
 }
 if (Has pause counting information not been updated yet)
 {
        Update pause information to Node;
 }
}
```

Figure 3.1.3-2. Seudo code for the VAD algorithm

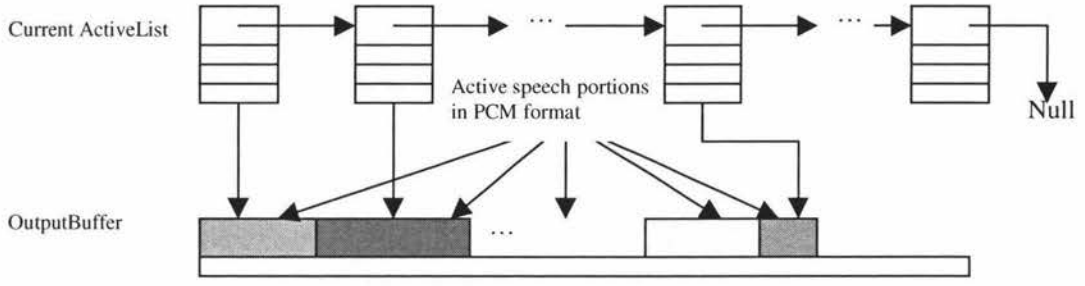


Figure 3.1.3-3. Each node of the current active list contains information corresponding to the active speech portion.

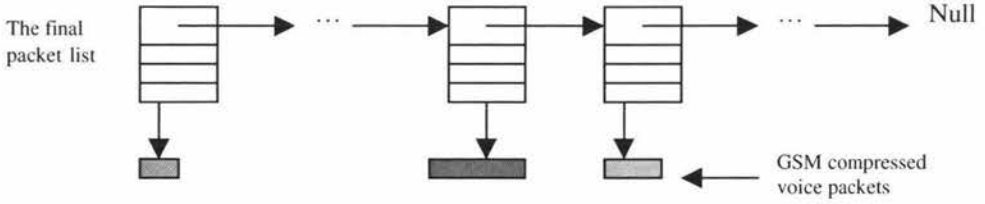


Figure 3.1.3-4. After the GSM conversion the CurrentActiveList is merged into the final packet list, each node of which contains a portion of GSM compressed active speeches.

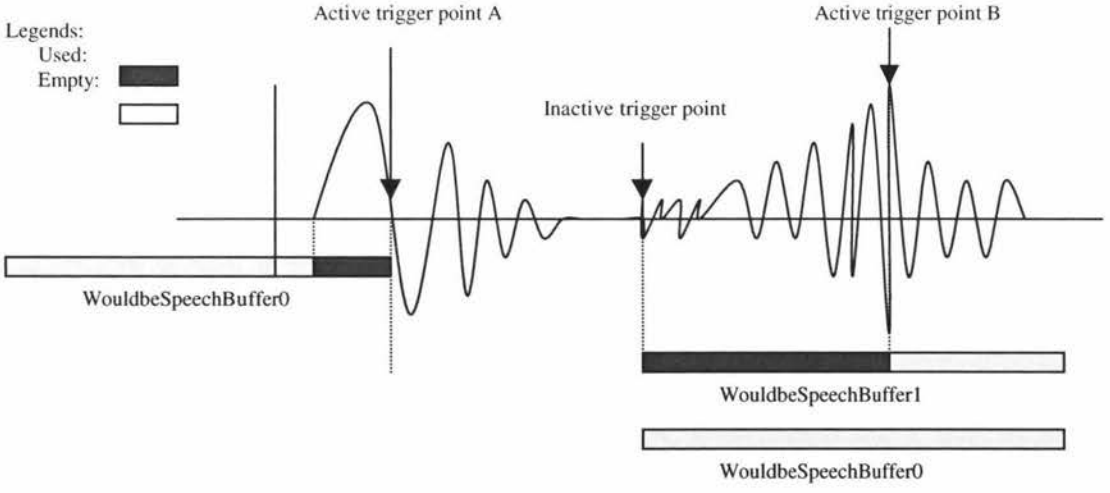


Figure 3.1.3-5. Using buffers to record active speech signals in either before an active trigger point or after an inactive trigger point.

After having been passed the GSM conversion block, the actual signal is packetised into the CurrentActiveList. After that the CurrentActiveList will be added into a final speech packet linked list as shown in figure 3.1.3-4.

We have seen that two buffers, each of which has the equal size, are used to keep track of those active speech signals either before an active trigger point or after an inactive trigger point. This is shown in figure 3.1.3-5. Also there is a special case --that is—if there is another active trigger point but the WouldbeSpeechBuffer1 (this keeps track of the active speech signals after an inactive trigger point) has not been filled up, we let the recording continue. In this case the WouldbeSpeechBuffer0 (this keeps track of the active speech signals before an active trigger point) is not invoked because WouldbeSpeechBuffer1 has done its job already.

The GSM conversion process converts the PCM speech packets into GSM compressed signal (actually WAV#49), and the active speeches have been packetised completely.

The serialisation process constructs an AEP file using the active speech packets. It also sets appropriate pauses according to the corresponding periods of silences.

3.1.4 Problems encountered and solutions used

Problem 1: Memory management issue

In Macintosh programming, memory management should be used with considerably caution. Because Macintosh OS does not allow a program to move, purge, merge, or allocate memory during an interrupt routine. As has been mentioned in the previous section, we must invoke interrupt tasking in the thread-A. When a buffer has been filled up the thread-A will call the thread-B via an interrupt routine

(**InterruptCompletionProc**). Also we have seen that dynamic memory allocation cannot be avoided in the implementation. There is a possible conflict here and it must be solved.

Solution 1:

Therefore, we must pre-allocate any memory we require and pass it into the interrupt routine. In the implementation, there are several places where dynamic memory allocation is involved.

As has been mentioned in section 3.1.3, that an internal ring buffer is used. The memory allocation and initialisation of the internal ring buffer must be performed before the recording starts.

In addition, the normalisation involves signal conversion, and this is done through a set of conversion routines. Macintosh OS provides this utility. For instance, if we want to convert voice at 22 kHz, 16 bit, and stereo into 8 kHz, 16 bit, and mono, we should open a converter first with this information. Then the converter does the conversion for us and finally we should close the converter. However this conversion process involves in the memory issue mentioned above. What we should do is to open a converter with the explicit recording information before an interrupt routine is called (specifically before recording starts). This results in the memory issue being completely solved during the normalisation.

Finally, within the VAD functional block, we need some temporary buffers for GSM conversion. They must also follow this rule.

Problem 2: Trigger point

As we have discussed in the VAD scheme section (section 2.6), the trigger point is determined by the result of a comparison between the threshold value and the sum of energy of a given size window. Let us assume the threshold is a constant value such as 20-30% of the sum of peak amplitude values. If the size of the window were set too small, we might end up with a trigger point happening between two syllables within the active speech. If the size were set too large such as one or two second, we might end up with no trigger point for a short speech. So the size of the window determines whether we have the correct trigger point or not.

Solution 2:

In order to figure out the correct size of the window, we have allowed the parameterisation of the window size, using the time-based scale as a measurement. The actual relationship between the parameter for the size of the window and numbers of samples has been mentioned in section 2.6. Therefore, a user can fully test the system so that the best parameter value can be found.

Problem 3: Cut point

In the GSM conversion block, we convert each PCM packet into a GSM compressed signal. That means, each time, 160 normalised PCM samples (one frame of 8 kHz, 16 bit, and mono) will be converted into 33 GSM bytes (one GSM frame). But for the WAV#49

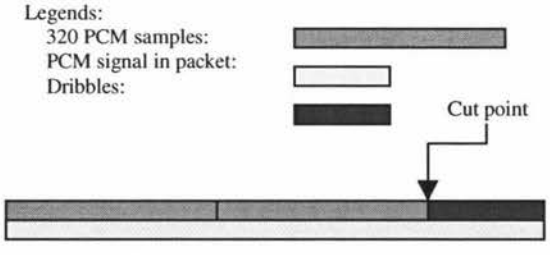


Figure 3.1.4-1. Cut point generated during the GSM conversion.

form, two frames of normalised PCM samples are converted into one WAV#49 frame (65 bytes). So the input for the conversion must be 320 samples.

However, we cannot know what the actual length of the active speech in a packet (let us denote it as length of packet) is. If the length of a packet is not a multiple of 320 bytes, a problem arises. That means there is a cut point within these 320 bytes. This is called the cut point problem as shown in figure 3.1.4-1.

Clearly, straightforward solutions are to ignore the dribbles or to use zero fill mechanism to fill the dribbles with zeroes, since we process the signal frame by frame, and each frame is very small (40 milliseconds). So in theory, the straightforward solutions should work well. However it is out of our expectation. If we use zero-fill technique for every frame, we cannot succeed. That is because we must expand the dribbles into 320 samples all filled with zero. This might result in the stepping in the reconstructed speech signal. For example, if there are lots consecutive packets containing dribbles and the size of dribbles is very small such as only 1 or 2 sample; if we use the zero-fill mechanism, a problem will arise as shown in figure 3.1.4-2. Also to ignore or delete the dribbles might cause a bad result since we might end up with losing too much valid data. For example, if the size of the dribbles in each packet is close to the 320 samples like 315 samples, if we use the dribble ignoring mechanism, we also will end up with a stepping because of losing too much valid data. This is shown in figure 3.1.4-3.

Hence we must look for another solutions, which can cope with these problems.

Solution 3:

The simplest way (called the multiple method) to solve the cut point problem is to make the length of every packet to be a multiple of 320 bytes. Needless to say there is not an easy to achieve because the length of each active speech portion is non-deterministic.

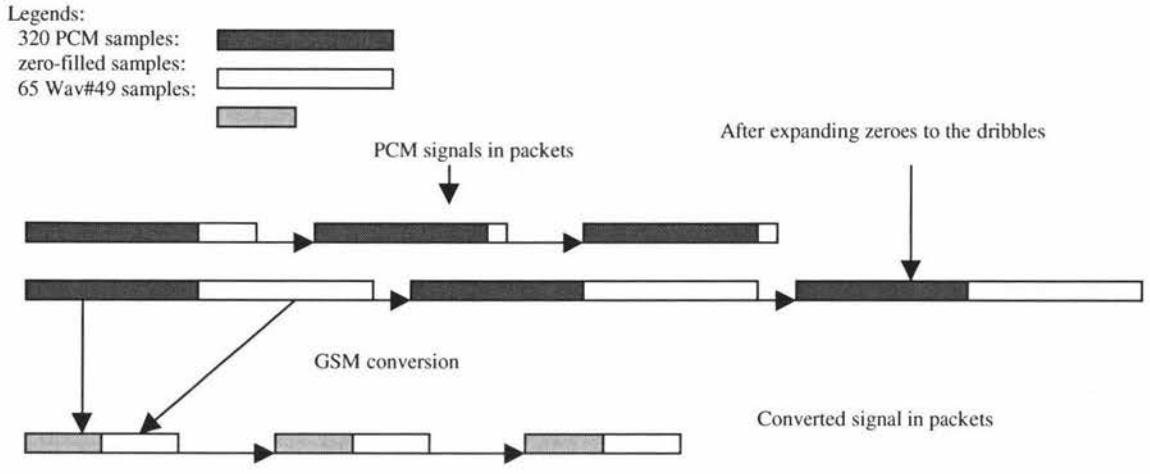


Figure 3.1.4-2. The zero-fill causes stepping in the GSM compressed signals.

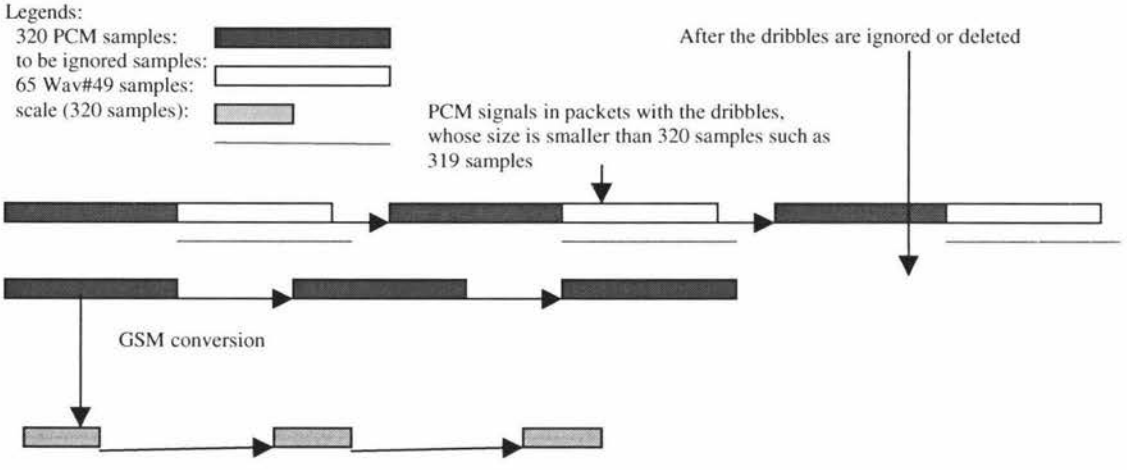


Figure 3.1.4-3. The situation where too much valid signals have lost after the GSM conversion results from the dribbles deleting.

Another way (called the buffering method) is to store the active speech signal altogether in a big buffer before the conversion. Obviously, this cannot satisfy the time-sensitive applications, but for the non-time-sensitive applications it works well. Hence the buffering method can be a candidate for the System-A. However System-A is going to be a part of the System-B that is a time-sensitive application. So the buffering method is also rejected.

The method used in the final implementation is to leave the dribbles of the current converted packet into the following packet. More specifically, if there are less than 320 bytes to be converted to GSM, all of those bytes are considered as dribbles. We add the dribbles to the start of the following packet. This will be continued until the end of the active speech packet is reached. In the last active speech packet, we use zero-fill mechanism to deal with the last dribbles if there are any.

Does this solution have any side effect? Yes, but it has but can be ignored. For instance, if the current frame (or packet) is irrelevant to its following frame (or packet), and there is cut point in the current packet, those dribbles will be added to an irrelevant frame (or packet). However those dribbles only take a maximum of 40 milliseconds in the active speech. Clearly a human cannot distinguish it. So the solution is acceptable. It also maintains all speech signals as complete as possible. This also provides a chance for us to adjust the buffering method by using a reasonable buffer size plus the cut point technique. When we need to faithfully reconstruct the original speech signal, the adjusted buffering method provides a good performance.

Problem 4: time-pause (Packetising signal)

The AudioGraph stores an asynchronous representation of the recorded speech, using time pauses instead of recording the actual silence. This is one of the goals of the AudioGraph project in order to reduce to size of AudioGraph files. So after having removed unwanted silence from the speech signal, we must provide information for constructing time-pauses.

Within System-A, if we want to faithfully reconstruct the original speech signal, we can use the buffering method with the time-pause information stored. But for time-sensitive applications, we cannot use the buffering method to reconstruct the signal. The

information for time-pause must be obtained when the signal of a packet is reconstructed or played back.

Solution 4:

Packetising signal is a good way to solve this problem. It guarantees that an active speech

Packet fields	Length (byte)	Description
packetLen	4	*length of raw data stored; after the raw data have been converted into GSM, this field will be discarded.*
duration	4	*GSM duration*
uncompletedflag	1	*if set, an active speech portion is complete in this packet; otherwise, the portion is not complete.*
pauseLen	4	*the length of pause (in millisecond) from the previous completed packet. If uncompletedflag is set to zero;*

Figure 3.1.4-4. Structure of a voice packet

packet contains all its relevant information. So whenever we want to reconstruct the original signal, we always have enough information to do it. The structure of the packet node used in the implementation is shown in figure 3.1.4-4.

However this solution involves dynamic memory allocation and there is conflict with the memory management! We must also follow the rule for memory management. That means we must de-allocate unwanted packets and reallocate new packet outside the interrupt routine, **InterruptCompletionProc** (thread-A calls it to trigger the thread-B). Actually, in the implementation, the system is an event-driven application. As we have introduced in the section 2.5, there is an event loop in the system. We do the memory allocation at the periodical's (an idler) **SpendTime**, which is an object that can receive time on a regular basic. When there is no event to process, the system will call the idlers' **SpendTime** routine.

Although we can solve this conflict, it does have a drawback. In a sense, the linked list is very flexible, and without the memory problem, we could allocate any memory block to store a packet when we need it. But with this constraint, we cannot. Hence, what we should do is to pre-allocate a linked list outside the interrupt routine **InterruptCompletionProc**, and then pass it into the interrupt routine. When finished processing a frame, discard the unused nodes of the linked list in the periodical; then add the remaining list to the speech's packet list, and so on.

Problem 5: the size of linked list

In problem 4, we have seen that a linked list must be created and passed into the interrupt routine. The length of the linked list is one of the key factors, which determines the overall speed of the system. What is the size for that list? Fortunately, the size need not be very large.

Solution 5:

We have recorded one-second of speech into the buffer, and then we start another recording. Then we process the recorded speech as the recording continues. All of the speech signal must be converted into PCM at 8 kHz, 16 bit, and mono before it can be converted into GSM. The speech processing produces a linked list, each node of which contains one active speech portion.

We have preset the internal recording buffer to record one-second of speech. For the normalised signals, one-second of speech signal contains the 8,000 samples x 16 bits per sample / 8 bits per byte = 16,000 bytes. After the recorded speech has been processed, the

total number of bytes of the active speech ranges in between 0 and 16 kilobytes (kbytes). This number of nodes determines the size of the linked list, let us denote it as $L_{oflinkedlist}$.

Let us denote that $P_{cf}, P_{wbs}, P_{wbsd}$ are parameters based on time in millisecond, each of which represents the size of the window, the size of the would-be-speech buffer, and the size of the would-be-speech-dribbles buffer respectively. We can derive a formula $Min_{speechportion}$, which represents the minimum active speech portion in bytes based on these parameters.

$$Min_{speechportion} = \frac{(P_{cf} + P_{wbs} + P_{wbsd})millisecond \times 8{,}000samples \times 16bitspersample}{1{,}000millisecond \times 8bitsperbyte} \quad (1)$$

$$Min_{speechportion} = 16 \times (P_{cf} + P_{wbs} + P_{wbsd}) \quad \text{bytes} \quad (2)$$

so we have got:

$$L_{oflinkedlist} = \frac{16{,}000}{Min_{speechportion}} = \frac{1{,}000}{P_{cf} + P_{wbs} + P_{wbsd}} \quad \text{nodes} \quad (3)$$

From the formula (2), we can see that $P_{cf}, P_{wbs}, P_{wbsd}$ parameters determine the size of the linked list. For example, In order to avoid false triggering P_{cf} must be greater than 100 ms, and we set that P_{wbs} and P_{wbsd} are 50 ms as minimum respectively. Therefore we have

$L_{oflinkedlist} \leq 5$ nodes

That means the maximum size of the linked list is 5 nodes in this case.

3.2 System B

3.2.1 Objectives

The purpose of the System-B is to provide a two-way voice over IP connection based on system A, and also to provide echo cancellation by using an echo inhibition protocol. System-B is similar to an IP phone, so providing quality of service is the primary task.

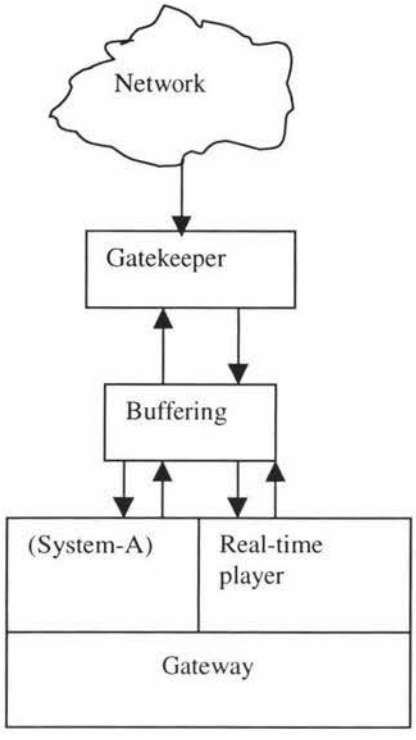


Figure 3.2.2-1. Architecture of the System-B

3.2.2 architecture

The architecture of the System-B is shown in figure 3.2.2-1. Clearly it is a typical VoIP application with the extra component being the real-time player. System-A with a real-time player acts in the role of a gateway for the entire System-B. This architecture is also a multi-threaded architecture because each sub-component of the gatekeeper and gateway is implemented as a thread. As for the gatekeeper, we embed this functionality into each peer-to-peer server.

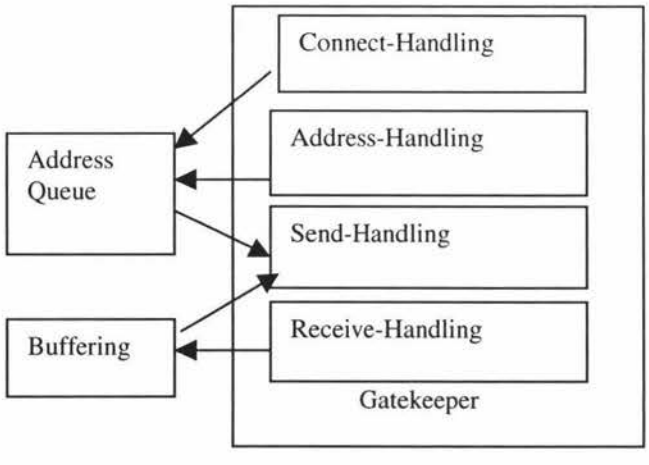


Figure 3.2.3-1. A group of threads in the gatekeeper

3.2.3 Application

In this section, we will examine those functional blocks within the gatekeeper and the gateway.

As shown in figure 3.2.3-1, the gatekeeper is composed of a group of threads. Some threads provide the server functionality and other threads provide the client functionality.

The connect-handling thread connects to the other gateways by invoking TCP protocol. The address-handling thread collects new connection addresses, to put it into an address queue for multicast purpose. The send-handling thread sends the recorded speech to all of the destinations by UDP protocol. The last thread is a receive-handling thread that receives voice packets from the network. Those packets can come from any location.

As can be seen from the figure 3.2.2-1, the gateway consists of two components: one is System-A, and another is the real-time player. The real-time player decodes the GSM compressed voice, and then plays it back while ensuring a simplex channel. The simplex channel ensures the playback and recording cannot happen at the same time so that echo cancellation can be achieved. If we are recording, the player only performs decoding then stores the decompressed data into an internal buffer without playback.

In operation, the speech signal has already been converted into GSM format within the gateway, and also it has been packetised and stored in an internal buffer (a shared queue). The shared queue is used to provide synchronisation between the gateway and the send-handling thread. The send-handling thread picks up a packet from the shared queue and then sends it to all connected client one by one. For example, if we are talking with 2 people (one is in Auckland, the other is in Wellington) at the same time, what we are saying is sent to them simultaneously. When a packet is inbound, it will be placed into a shared queue, which provides synchronisation between the receive-handling thread and the real-time player thread. The real-time player takes packets from the shared queue and decodes it. If playback can be performed, the player plays it back immediately; otherwise, the player leaves it into a circular buffer for later playback.

3.2.4 Problems encountered and solutions used

Problem 1: Echo cancellation

Echo cancellation is one of the key factors, which affects the QoS of the VoIP products. Because System-B is designed to provide two-way speech over IP connection, echo is generated by playback when recording. We have described this in the first chapter (section 1.2).

Solution 1:

Our solution to this problem is to use a simplex channel to provide two-way speech. This is achieved by using an echo-inhibiting protocol. The protocol ensures that recording and playback cannot occur at the same time. When playback is performed, a recording request will be discarded or prohibited. In our implementation, we disable the record button when the player is playing back. When recording is being performed, the player will still decode those packets that have arrived. In order not to incur data loss, we must buffer this decoded data. What we have done is to provide a 30 second circular buffer (ring buffer) to store that data for later play back. If no playback is started within the 30 seconds, some data loss might occur (because we override it in the buffer).

Problem 2: Simplex channel busy

Because we use a simplex channel to provide the echo cancellation, this gives rise to the question of how long one event (recording or playback) should use the simplex channel at

one time? If an event were occupying the channel and was never to release it, the other event would not have a chance to use it.

Solution 2:

In our implementation, we let the user control the trigger of the recording event. That means the user can click on the button to start or stop recording. However the user cannot do anything about playback because the real-time player controls everything. As we have

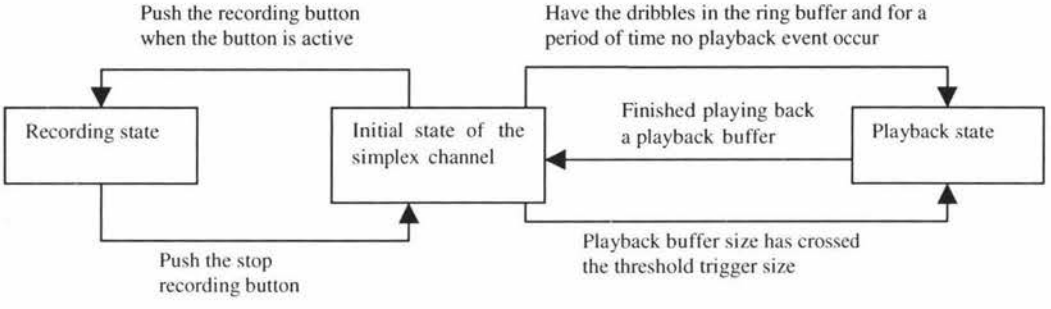


Figure 3.2.4-1. The state transition diagram of the echo-inhibiting protocol

mentioned above, we use an internal circular buffer to store data that has been decoded. The buffer can contain 30 seconds data at maximum. But we cannot use this buffer as a playback buffer in the normal situation, as it is too large and would cause unacceptable delay in interaction. So our solution is to provide a variable playback ranging from 250 to 1000 milliseconds. The user may choose whatever size required.

One may ask a question how it handles silence? Actually we have not provided any mechanism here to handle silence, instead we use the network delay between two packets as the pause to represent silence and assume the network delay is uniform. However, if the data can not be played back on time, this mechanism can not provide silence. This is one drawback of the system.

When the real-time player has finished a buffer of data, it will reset the simplex channel to its initial state. This allows the user have a chance to start speak if s/he wants to do that. But during the playback, we disable the recording button so as to ensure that the user is not allowed to start speaking. The transition diagram of the echo-inhibiting protocol is shown in figure 3.2.4-1.

We also provide a graphical interface to show how much data there is still in the circular buffer. While data is accumulating in the circular buffer, this graphical interface gives an indication to the user to decide whether s/he should stop speaking.

Problem 3: Voice dribbles

As have mentioned above, the real-time player plays a buffer of data. However, if a short packet of speech does not cross the threshold buffer size it cannot be played back in time. It must await the next packet. For example, some signals might exist in the graphical interface, but if no more signals were coming in, they would not be played back.

Solution 3:

To solve this voice dribbles problem we must trigger the player to playback those dribbles if there is no signal coming in over a period of time. We set this period to a value, which is the minimum of the playback buffer size, i.e. 250 milliseconds.

Chapter 4 Results

In this chapter, to begin with, we would like to discuss the two systems implemented. One is the VAD recorder system (System-A) and another is the VAD network recorder (System-B). Two brief user documentations will be given in the first section. These two systems are implemented in Macintosh (specifically iMac with G3 processor).

Moreover, we will discuss the results of testing. The software implemented has been tested in the multimedia laboratory, which is a typical working environment of the AudioGraph. The network environment for System-B is a local area network (Ethernet, 100 Mbps, half-duplex). The results are mostly tested by one person, and the quality (Bad, acceptable, and good) of results presented in this report is given by the tester. For example, the tester listens to the original speech (recorded) and compares it to the results, so that an appropriate rating value for the result can be given.

Also the parameters will be analysed so as to produce a set of default parameters.

Figure 4.1.1-1 The graphical interface of the System-A.

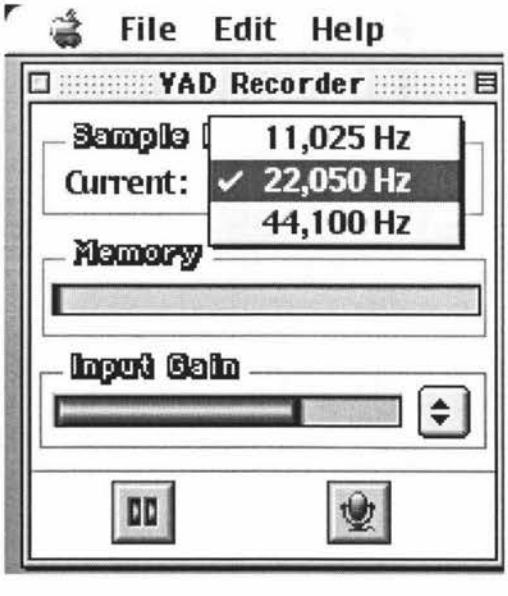


Figure 4.1.1-2. Select a sample rate

4.1 VAD Recorder

4.1.1 Documentation

Figure 4.1.1-1 shows the interface of System-A. Here we give a brief user documentation for this system as the follows:

To select a sample rate: click the pop-up menu button. A pop-up menu that contains all available sample rates will appear. Move the mouse to select the rate you like. This is shown in figure 4.1.1-2.

To change an input gain: click the pop-up menu button. A pop-up menu that contains all available input gain values ranging from 0.5 to 1.5 will appear. Move the mouse to select the value you like. This is shown in figure 4.1.1-3.

To set a preference setting: click on Edit then the menu will appear. Select the preference tab. This is shown in figure 4.1.1-4. Then the preference

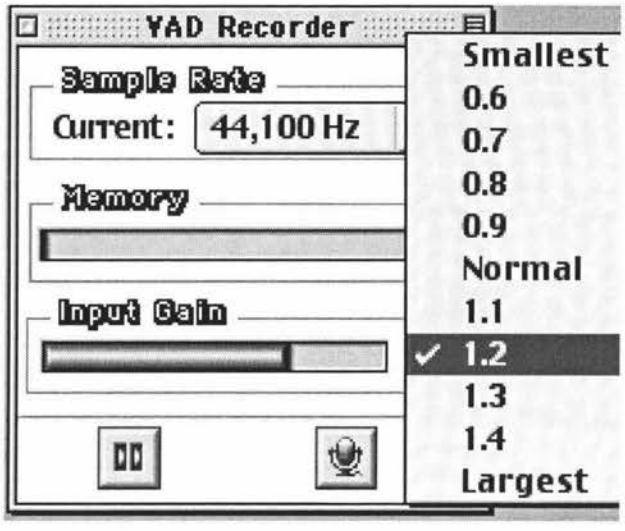


Figure 4.1.1-3. Select an input gain value

Figure 4.1.1-4. Select preference setting

window appears as shown in figure 4.1.1-5.

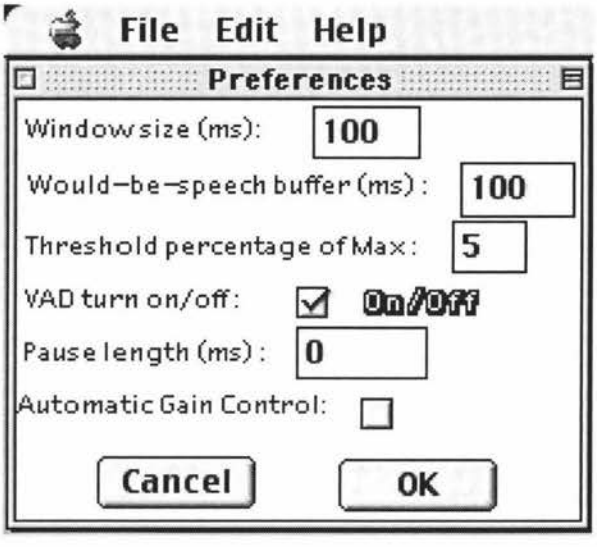


Figure 4.1.1-5. The preference settings

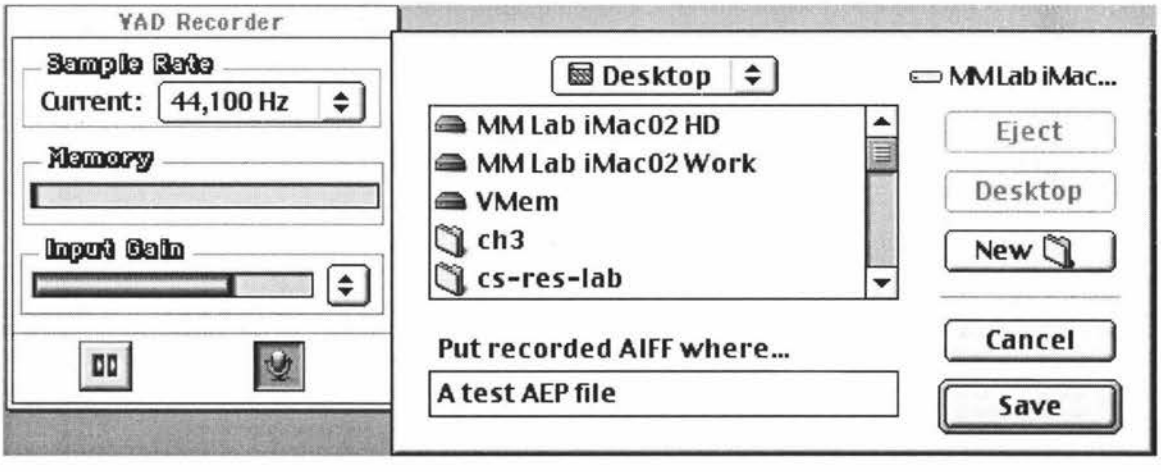


Figure 4.1.1-6. File chooser window

To record an AEP file: click on the record button (the microphone icon) then a pop-up window appears. The user can choose the location and any file name for the output. This is shown in figure 4.1.1-6.

To stop recording: click on the stop recording button, which is besides the record button.

4.1.2 Application

As can be seen through those snapshots above, the VAD recorder lets the user select any recording sample rates that the Macintosh operating system (OS) supports. This will vary from one Macintosh to another. The sample rate option can let us test the speech recorded in different sample rates.

There is also a memory bar that indicates the memory usage. The memory bar in full range denotes four Mbytes. This tells us how much GSM compressed data has been generated. And also the recorder provides an input gain setting; the input gain affects the SNR value of the speech we want to record, which determines the performance of the VAD system.

Moreover, the recorder provides a preference setting, which allows the user (mainly for the tester) to vary the set of parameters. Actually, this provides convenience, because, in the system a number of parameters need to be tested and we often need to test a parameter based on others parameters' best values. As can be shown in figure 4.1.1-5, these parameters are the window size, the size of the would-be-speech buffer, the threshold percentage, and etc. The default settings are as the initial values.

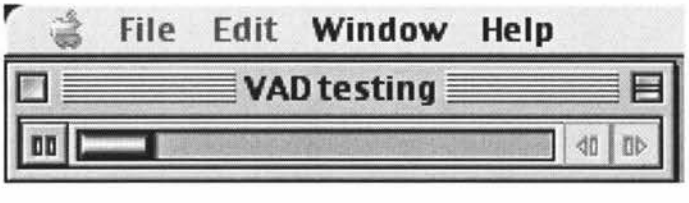


Figure 4.1.2-1 . The AudioGraph player is playing an AEP file produced by the recorder

In addition, the output of the recorder is an AEP file, which

contains the GSM compressed data as the first stream and the original speech signal as the second stream. This provides convenience for the tester to rate the VAD performance. Figure 4.1.2-1 shows an example of the AudioGraph player playing an AEP file recorded using System-A.

When the VAD recorder is launched, all default parameter are set up, such as the sample rate, the input gain, and others that exist in the preference setting. The most important parameter is the size of the window, and this should be preset correctly in order to produce reasonable results. A speech segment usually lasts 100 milliseconds so that we set this parameter to 100 millisecond as its original value. According to this parameter, we can test the recorder and find the best values for other parameters. And then according to the best values of other parameters, we come back and find out what it is the best value for the window size parameter.

Actually this application is only intended to demonstrate the feasibility of doing this in the AudioGraph not as a particularly useful application in its own right. Because the results were successful this application is now being integrated into the production AudioGraph recorder application.

4.2 VAD network recorder (System-B)

4.2.1 Documentation

The interface for the VAD network recorder is shown in figure 4.2.1-1. The documentation for this system is given as the follows:

To set preference setting: click on the Edit menu, then a sub menu appears. Select the preference menu, and a preference setting window appears. This is already shown in figure 4.1.1-5.

To select an input gain: click on the pop-up menu tab, a pop-up menu appears, select the gain value. This is shown in figure 4.1.1-3.

To set up a network connection: click on the connection button (or click on File menu, then select Connect Remote Host as shown in C of figure 4.2.1-2), then a connection window appears. You can type in the IP address of the host you want to connect to. This is shown in B of figure 4.2.1-2.

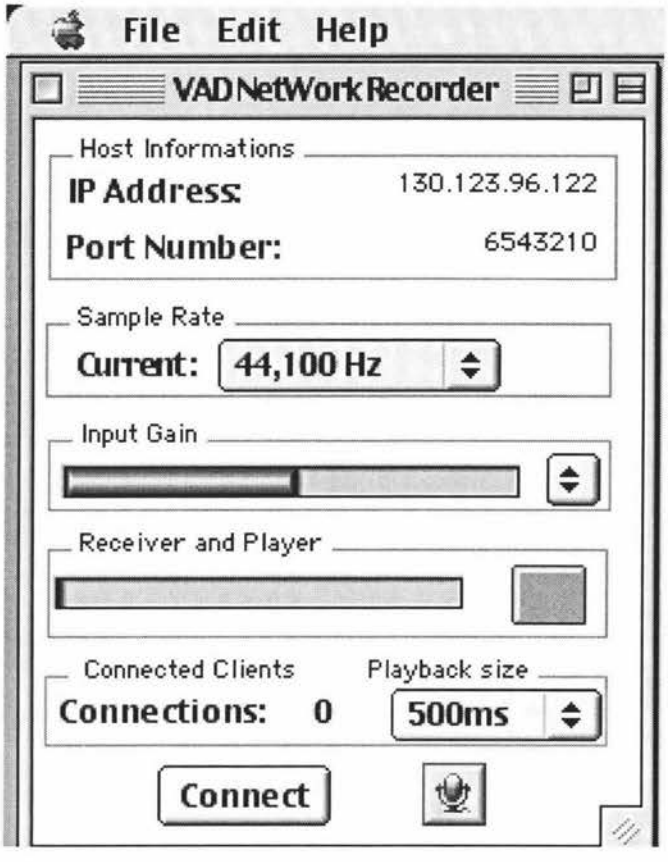


Figure 4.2.1-1. The graphical interface of the VAD network recorder application.

To start recording: simply click on the record button if the button is active. If recording is started, the indicator will become red. This is shown in D of figure 4. 2.1-2.

To select a playback buffer size: click on the pop-up menu tab, then a pop-up menu appears. Select the size you want. This is shown in E of figure 4. 2.1-2.

To select a sample rate: click on the pop-up menu tab, a pop-up menu appears. Select the sample rate. This is shown in A of figure 4. 2.1-2.

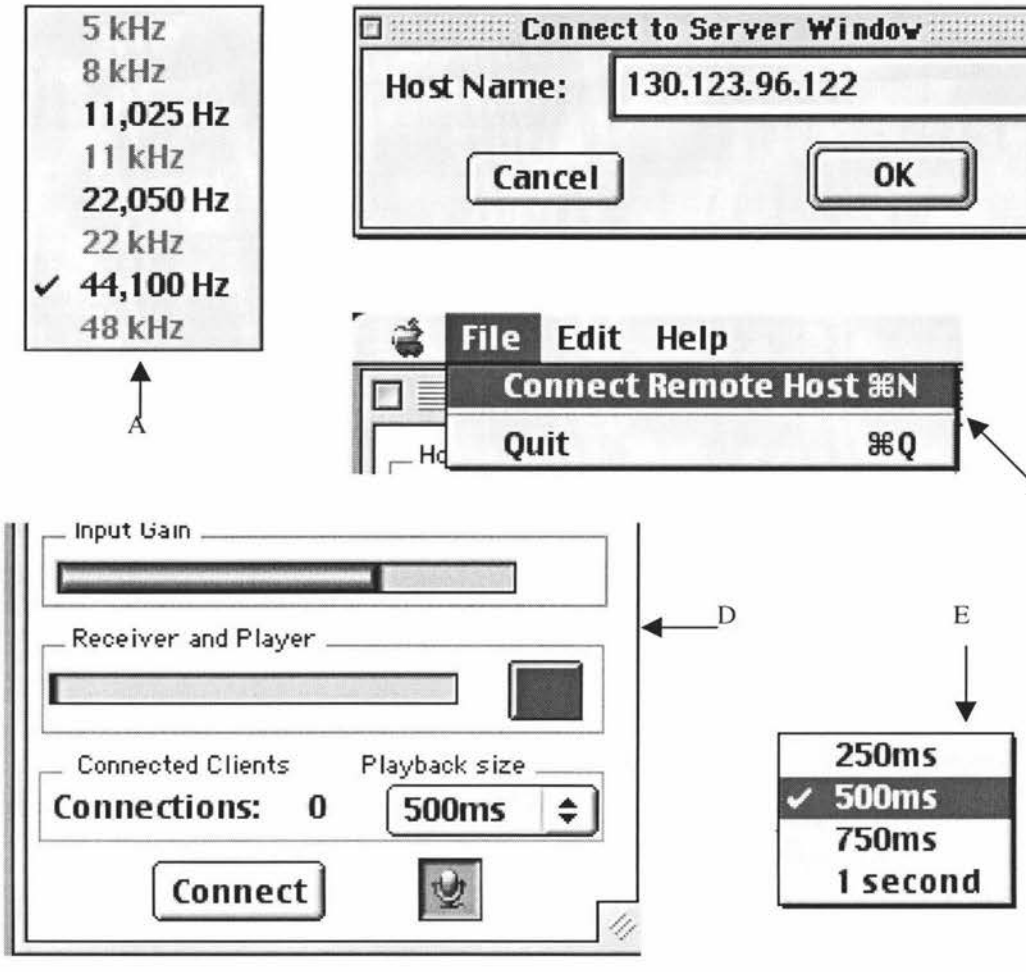


Figure 4.2.1-2. Snapshots for the VAD network recorder

4.2.2 Application

Compared to the interface of the VAD recorder, this system is quite similar except it also involves network connection. Actually, this system is a time-sensitive application but the VAD recorder is not.

As shown in figure 4.2.1-1, the sample rate and input gain provide the same functionality as those in the System-A. But the sample rate pop-up menu is not actually the same. For

example, when the sample rate's pop-up menu appears, we can see a list of sample rates that are used frequently; and some sample rates are disabled because they are not supported by the operating system (or the machine).

Also we can see the host information (IP address and Port Number) of the current machine through the interface of the system.

The progress bar of the Receiver and Player is a buffer indicator. When there are packets arriving, the buffer increases; when the real-time player plays back, the buffer decreases. This buffer is a ring buffer but the indicator only denotes how many portions of the buffer have been used. Obviously, when the buffer is full the new inbound packets will override the oldest packets in the ring buffer regardless of those packets having not been played back yet. However, in this case, the system beeps automatically so as to inform the user that the buffer is full. This lets the user to decide whether s/he should continue speaking or give the chance to the "real-time" player to playback.

The Connected Clients shows the number of clients currently connected with the system. The colour indicator tells us the following things:

- In red, the system is playing back, and the recording is disabled.
- In green, the user can start recording; or if some packets had arrived, the player can start playing.

The Connect button lets the user start connecting with other servers using the TCP protocol. If no connection has been set up and you start recording, the recorded speech is simply ignored.

When the real-time player is not playing, the user can change the playback buffer size. This parameter determines how many samples (in time scale) can trigger the player to start playing back; otherwise the user is not allowed to do that. We have preset a set of values of the playback buffer in the pop-up menu as shown in E of figure 4.2.1-2.

However, some might ask why these values of the buffer size given look like a little bit large? As we have stated earlier, the record button is disabled when the player starts playing back. If the buffer size were too small, the user would not have any chance to click that button to trigger the recording.

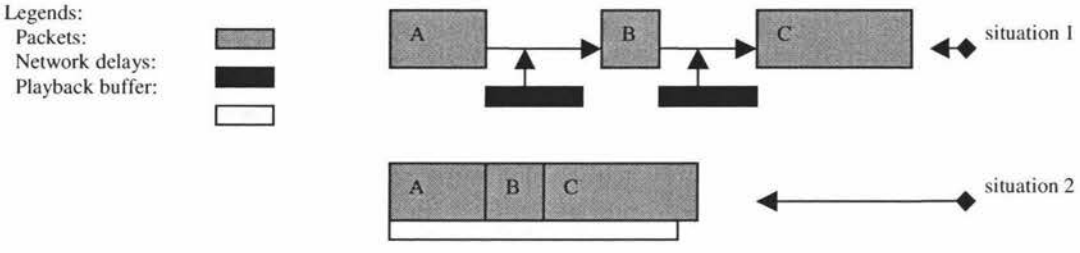


Figure 4.2.2-1. The size of the playback buffer removes the delay representing the pause is what we expect

When the system is launched it is already capable of receiving voice packets and playing back if data is available. All inbound packets are pause-less because we expect the delay in the network to define the pause in this application. However, the larger the playback buffer size is, the more the delayed pause is affected. This can be shown in figure 4.2.2-1. In situation 1, what we expect is that receive packet A and playback, over a period of time, receive packet B and playback, over a period of time again, receive packet C and playback and so forth. But something might happen as situation 2 shown in the figure 4.2.2-1. The packets A, B and C are stored in to the buffer, when the threshold value of the playback buffer size is reached, we play it back but without delays between these packets. This is one of the drawbacks using a simplex channel. Therefore, we must balance the playback buffer size and the performance of the simplex channel. Thus far, we can understand a quite large value of the playback buffer size the system chosen diminishes the quality of service the system provided.

When the user has pushed down the record button and the network connection has been set up as well, the system starts recording and transmits GSM compressed voice packets to all connected ends through the network.

The user also can change the parameter settings through the preference setting window as the same as System-A.

4.3 Parameters analysis

In this section, the results produced by qualitative evaluation will be shown in this chapter, and parameters are analysed. Prior to doing so, we would like to introduce a rating system and the experimental set up for the results.

- **Rating system**

  In order to make the results more understandable a rating system used to represent the performance of the VAD algorithm is employed. The rating system is divided into three areas: Bad, Acceptable, and Good. Bad means that the result is a total failure, such as, in the VAD, if no silence has been detected or the result is unrecognisable at all. Acceptable means that the result is recognisable but it still can be improved, e.g. silence has been removed from the result or it is not smooth enough. The last one Good means that the result has not much difference with the original speech.

- **Experimental set up**

  All results presented here are tested on the Macintosh G3 (iMac, PowerPC G3 400 MHz) machine. The network environment is a LAN (Ethernet, 100 Mbps, half-duplex). The microphone used is an external microphone. Location is in the multimedia laboratory of the Computer Science department at Massey University. Let us look at what the primary parameters are.

  As we have known, the VAD algorithm can not work in a low SNR value condition. Hence we turn off the automatic gain control first then set the input gain to a medium value like a 1.2 or 1.1. Next, we set the size of the window to 100 milliseconds and

the size of would-be-speech buffer to 50 milliseconds. Next, we set the threshold percentage to 10%. We start testing the system under the primary parameters, and we will test the parameters one by one so as to find its optimal setting.

By using the rating system, the tester gives a rating to the result s/he has evaluated. One might ask how a tester evaluates the results? Actually the tester listens the result and its original speech, and then s/he will give a rating to the result according to the result of comparison. I was the main tester, and a couple of students helped to test the results. Let us turn our attention to those results now.

4.3.1 Input Gain and VAD performance

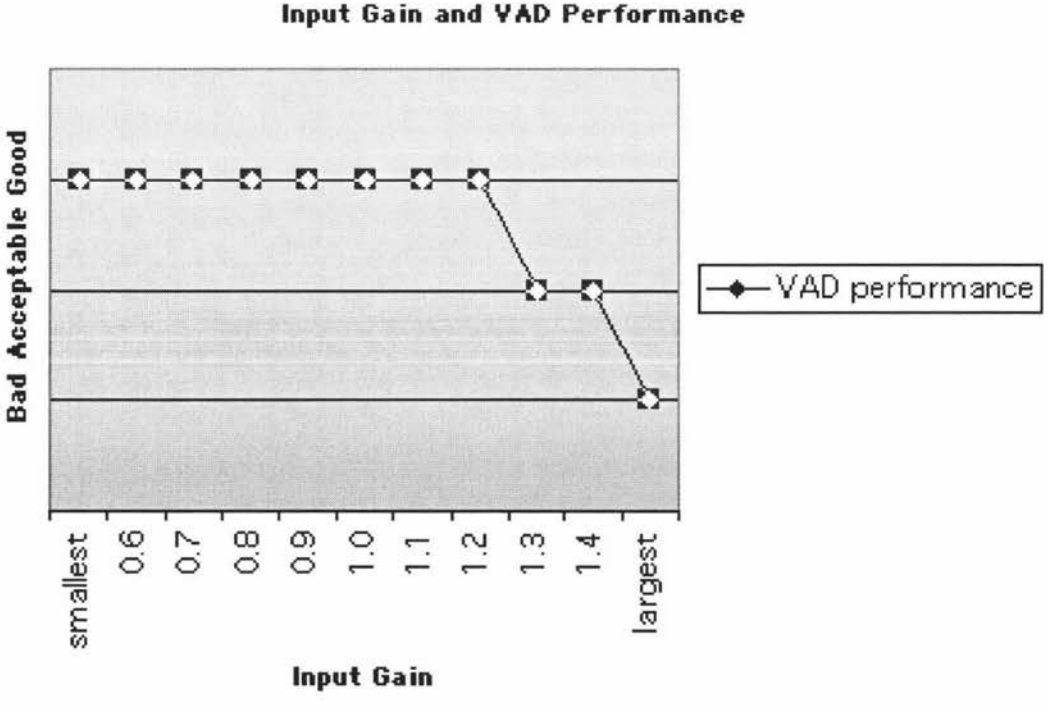


Figure 4.3.1-1. The relationship between the input gain and the VAD system's performance.

The figure 4.3.1-1 shows the relationship between the input gain and the VAD system's performance. The input gain value affects the SNR value. For example, when the gain is 1.3 or 1.4, the SNR value is very small. As been discussed in the VAD section (section 2.6), we have known that the energy threshold VAD scheme cannot work well when SNR value is quite small during silence. Our result illustrates this effect. Although working with a small input gain value still makes the System-A work well, the microphone must be positioned as close to the sound source as possible in order to capture the speech.
This result produced is based on the following parameter setting:

- Microphone position 30 cm to the sound source
- Window size: 100 ms
- Would-be-speech buffer 100 ms

However, the input gain affects the threshold percentage setting. Hence we cannot give a specific input gain and threshold percentage here. What we can do is to, choose a particular input gain value, and then find out a full range of threshold percentage values, which ensures the VAD recorder has good performance. One thing we would like to make clear is that the position of the microphone has never been changed during the production of those values shown in the table 4.3.1-1. The smaller the input gain value is, the smaller the threshold percentage value is.

Input gain value	The range of threshold percentage
Smallest	0.5%
0.6	0.5-0.8%
0.7	0.8-1.2%
0.8	1.2-1.5%
0.9	1.5-2.5%
Normal	2.5-5%
1.1	5-10%
1.2	10-15%
1.3	15-25%
1.4	25-35%

Table 4.3.1-1. The correspondence between the Input gain value and threshold percentage that ensures the VAD recorder has a good performance.

4.3.2 Threshold and the VAD performance

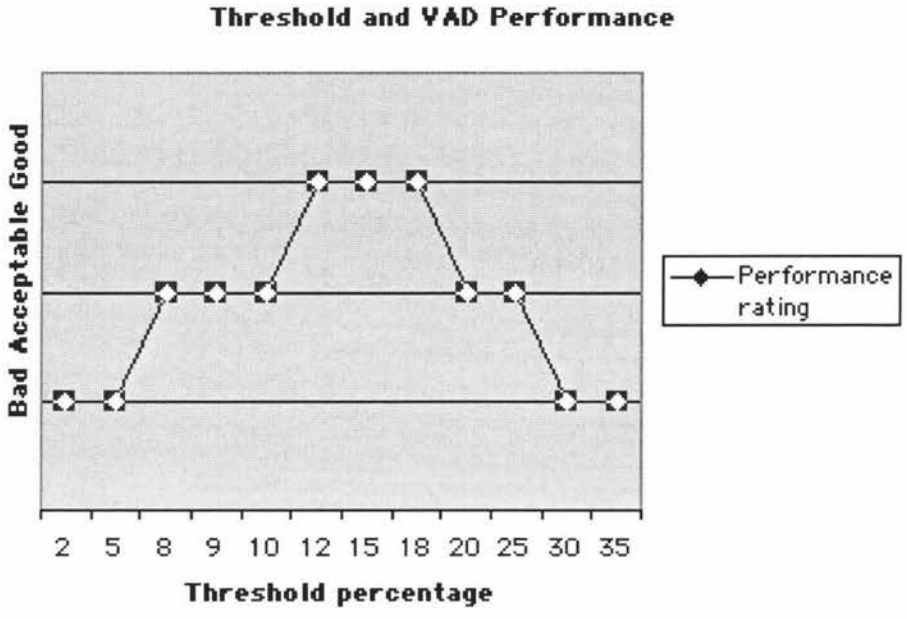


Figure 4.3.2-1 The threshold percentage and VAD performance

This graph (in figure 4.3.2-1) shows the relationship between the threshold percentage and the performance of the VAD system. In our implementation, the threshold value is calculated by the threshold percentage times the peak amplitude of a sample (if sample were16 bit, the peak amplitude is 32767) then times the number of samples in the window. It can be expressed as the following formula:

Let us denote the sample is n bits, the window size is S_w ms, the threshold value is V_t, and the threshold percentage is P_t. And we have known that the recorded signal is normalised to PCM (8 khz, 16 bit, and mono).

$$V_t = P_t \times S_w \times \frac{8000}{1000} \times (2^{n-1} - 1) \qquad (ms \times \frac{samples}{ms} \times \frac{peakamplitudevalue}{sample})$$

$$V_t = P_t \times S_w \times \frac{8000}{1000} \times (2^{16-1} - 1) \qquad (peakthreshold)$$

$$V_t = P_t \times S_w \times 262136$$

The above formula shows that the actual recording sample rate has no relationship with the threshold value. That is because we will normalise the recorded signal before it is processed. Also we have seen that the threshold value is based on the peak amplitude value of a sample.

The results shown in this figure are made under a condition that other parameters are set to the follows:

1 The size of the window: 100 ms
2 The would-be-speech buffer: 100 ms
3 The input gain: 1.2

According to the above values, we prerecord a period of speech, and then we change the threshold percentage value each time to produce an output. Then we listen to this output AEP file and give a rating to this result. Finally we produce this graph. Actually this result illustrates the relationship between the threshold percentage and the VAD performance under the most common input gain.

4.3.3 Window length and the VAD performance

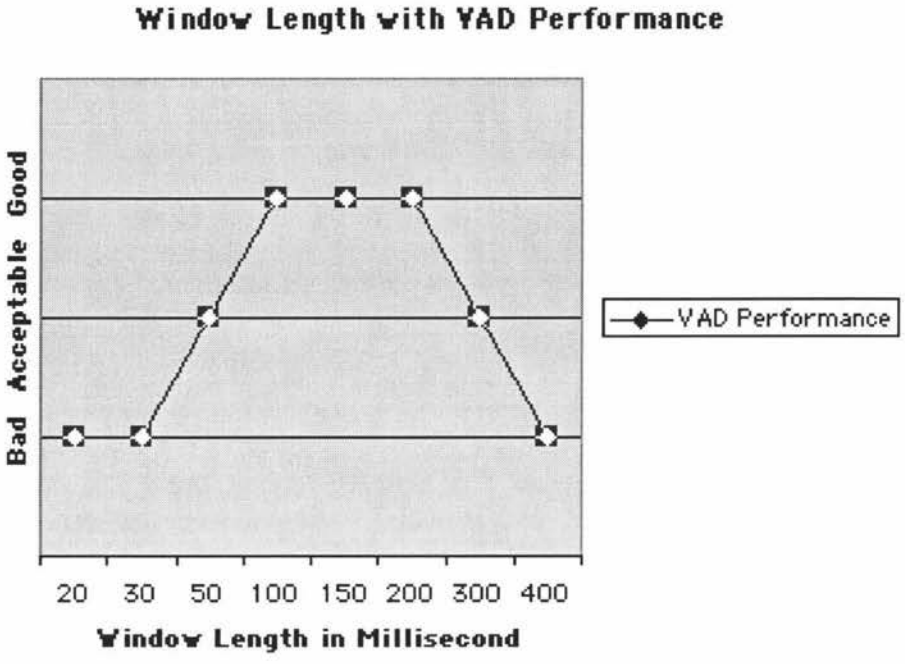


Figure 4.3.3-1. The relationship between the size of window and VAD performance

From the figure 4.3.3-1, we can see that the size of window also affects the VAD performance. The parameter settings used to produce this graph are shown in table 4.3.3-1.

Parameters	Data group1	Data group2
Input gain	1.1	1.2
Threshold percentage	10%	15%
The would-be-speech buffer	100, 150, 200 ms	100, 150, 200 ms

Table 4.3.3-1. Parameter settings

We tested the window size parameter based on two groups of data. We prerecorded a period of speech and then played it back as the sound source for every recording. We varied the window size from 20 to 400 ms (we increase it by 20 each time). Next we listen to those results and give a rating to every result then produce the graph.

4.3.4 Would-be-speech buffer and VAD performance

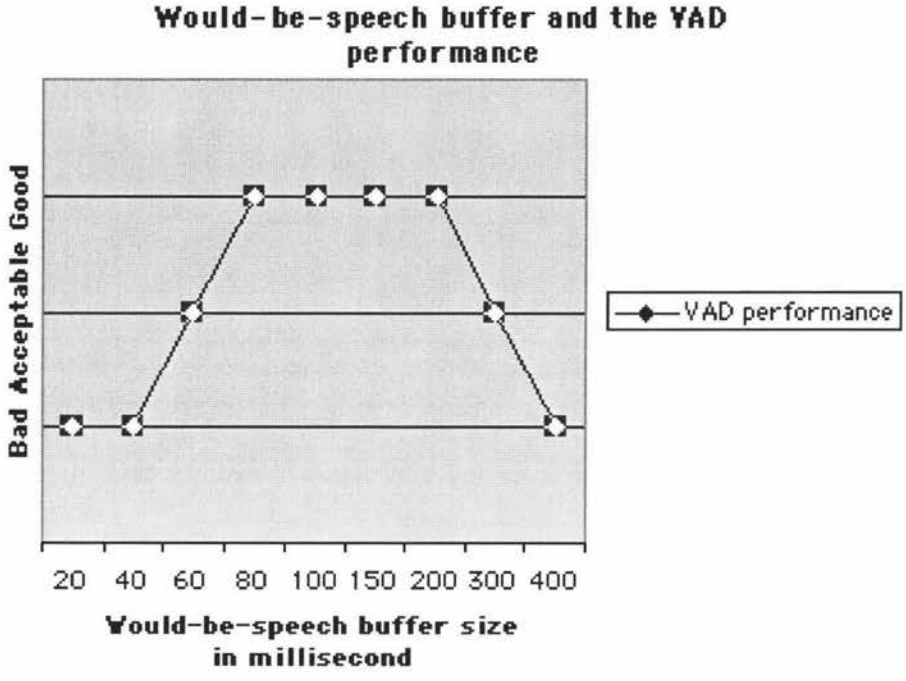


Figure 4.3.4-1. The relationship between the size of would-be-speech buffer and VAD performance.

The result shown in the graph (in figure 4.3.4-1) is produced as the following settings:

1. Input gain 1.1
2. Threshold percentage 10%
3. Window size 100 ms

The size of would-be-speech buffer affects the performance of the VAD system, especially on the bit-rate reduction. When the buffer is too large, the quality of service (QoS) of the VAD system provided is good but the bit-rate reduction is poor. So we should keep this buffer as small as possible so it can provide a good QoS.

4.3.5 Playback trigger interval and System-B performance

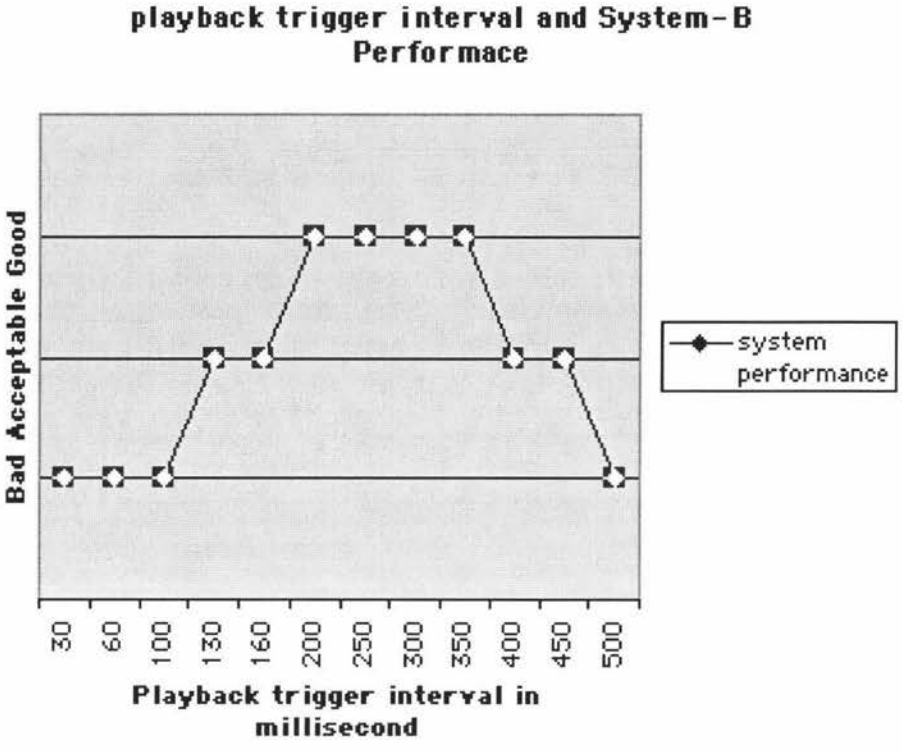


Figure 4.3.5-1. The playback trigger interval and System-B's performance

The playback trigger interval is a time-based parameter, used to give the real-time player a trigger on the interval to playback remaining packets in the buffer. For example, when there is no recording event but there are some packets in the buffer waiting to be played; however those packets cannot fill a playback buffer up. When the interval has expired and the system is idle, then the player needs a trigger to start playing those remaining dribbles. The result is based on the following parameter settings shown in table 4.3.5-1:

Type	Parameters
Input gain	1.1
Window size	100 ms
Would-be-speech buffer	100 ms
Threshold percentage	10 %
LAN	Ethernet 100 Mbps
Computers (client and server)	IMac G3 400 MHz

Table 4.3.5-1 Parameters

4.3.6 Playback buffer and smoothing the voice

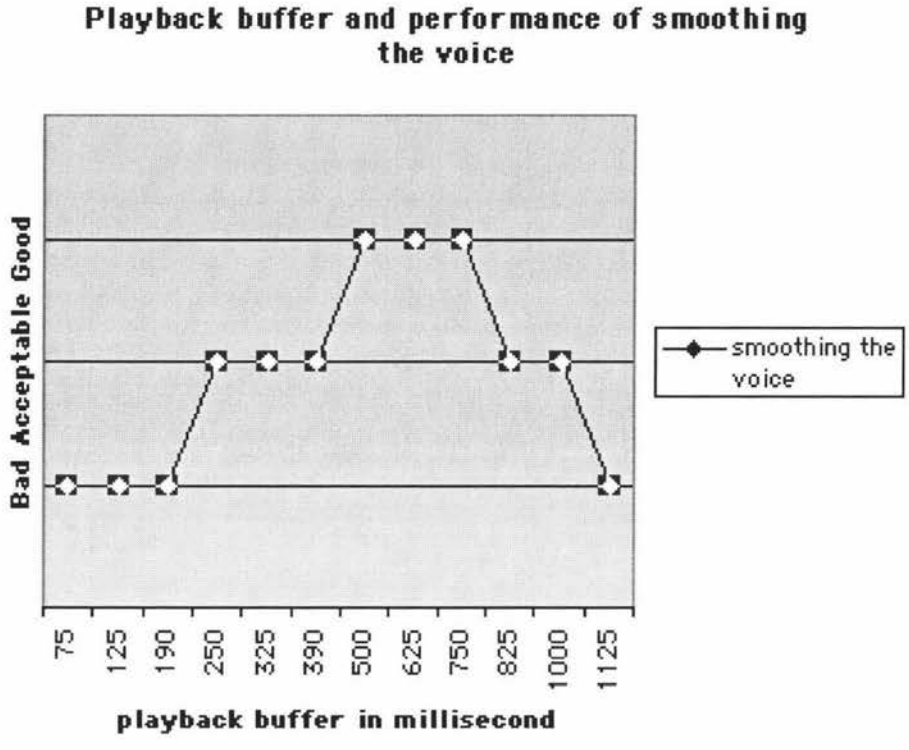


Figure 4.3.6-1. The playback buffer size and the performance of smoothing the voice reconstructed

The figure up front shows the relationship between the size of the playback buffer and the performance of the System-B trying to smooth the voice reconstructed. This buffer increase the total delay of the System-B, but reasonable buffer size smooths the voice reconstructed while playing back. We recommend the size of the playback buffer should be set to 750 millisecond as a default value. This value seems too large because it causes an 1.5 second end-to-end delay. However callers must make some adjustments on a simplex channel. Actually, when the playback buffer size is set to 250 millisecond, System-B should still provide acceptable performance. The reason why this situation happens is because of only a unique channel used to playback the packets and the unique channel having a delay between two soundplay buffers. This phenomenon exists in the

current version of the AudioGraph tool. To overcome this, we might try to use multiple channels to playback. But this might cause a problem--that is—the playback order should be maintained the same with the packet order.

4.3.7 Default parameters recommendation

After having fully tested System-A and System-B, we recommend the parameters used in the two applications should set to the following values shown in table 4.3.7-1:

Parameter types	Parameter
Window size	100 ms
Would-be-speech buffer	100 ms
Input gain	1.2
Threshold percentage	15%
Playback buffer	750 ms
Trigger interval	250 ms

Table 4.3.7-1. Default setting for the two applications

Summary

The results are generated from the Macintosh platform where our system has been implemented. The recommended parameters can be used as the default values of the synchronous version's AudioGraph, and they have been fully tested in our implementation. The VAD algorithm used provides a good performance in the AudioGraph's working environment. And the two-way speech over the IP (we only tested our system in the LAN) have an at least acceptable performance, but it can still be improved to minimise the playback buffer size so as to reduce the end-to-end delay.

Chapter 5 Conclusion and future work

5.1 Conclusion

This section gives a general discussion of the two systems that we have implemented, and then we will conclude with the techniques we have learnt through the work for this project. To begin with, we will discuss these two systems respectively.

System-A focuses on the VAD algorithm testing. This work was a feasibility study for evaluating GSM compression and VAD speech segmentation for the AudioGraph tool. Under the environment of the AudioGraph, the simple energy threshold VAD algorithm chosen has a good performance. This low cost and simple algorithm satisfies the AudioGraph's system requirements and results in the bit-rate requirement of the AudioGraph system being reduced from 13 kbps down to around 7 kbps. This also provides more flexibility for the synchronous AudioGraph to achieve its goals of distance education.

The parameters used in the System-A are fully tested and can be set as the default values for the synchronous AudioGraph tool. Most of the parameters are platform independent, so they can be directly ported to the other platforms such as PC and Unix. But input gain is platform dependent. Moreover there is a multi-tasking issue in this implementation, which might cause a problem during the porting of this system to another platform. The platform independence of the VAD algorithm and its most parameters can be reused without making any changes.

System-A packetises the active speech into small packets and stores them into an AEP file for later playback. The packetising functionality enriches the voice-editing feature of the AudioGraph tool. Any mistakes made during the recording can be eliminated through the voice editing without redoing the entire recording again.

Another advantage of System-A compared to the current version of the AudioGraph tool is that it speeds up the system and reduces the memory requirement. This occurs in loading and storing files and also in expanding web sites. These are achieved by storing the GSM compressed sound instead of the PCM compressed sound directly. This eliminates a big drawback of the current version of the AudioGraph tool.

In conclusion, System-A successfully achieves one the first goal set for this project, which was to use a reasonable VAD algorithm to improve the AudioGraph tool's functionality for real-time requirements.

A voice over IP connection system (System-B), which provides real-time two-way speech, has been implemented as well. This gives us a chance to experiment the VoIP technology.

From the designer's point of views, System-B is a typical VoIP system and takes the direction that the gatekeeper is embedded into the gateway. Although we have still separated these two components in the design phrase, they are really hard to be separated in the implementation phrase.

System-B solves one of the biggest problems in VOIP applications, that of the echo cancellation. It achieves this by using a simplex channel, and this makes a simple two-way voice over IP connection provide a good performance. This solution is very simple and straightforward and has a low cost, based on the trade off time sensitivity (or between the real-time applications and partial real-time applications). Although the simplex channel diminishes time sensitivity of System-B, it provides a good performance. However, the echo cancellation problem of the VoIP has not been completely solved nowadays by even the commercial vendors, so the standard solution to the problem remains open.

With respect to TCP/IP, actually in the implementation, we have employed TCP protocol to establish the network connection and UDP protocol to transmit voice packets.

However, in order to satisfy the real-time demand of the system completely a higher level protocol is required like RTP.

In addition, multicast is supported by System-B, and this provides an easy interaction between tutors and students. This real-time interaction allows distance education tools to provide a more powerful functionality.

Also the system solves the problem of smoothing voice by using comfort noise generation. The time pauses will be added into the playback buffer virtually, and this is all controlled by the size of playback buffer.

As for the portability, most components of System-B are platform dependent, except for the gatekeeper. The gateway component that is System-A has been discussed above, but the real-time player heavily depends upon Macintosh OS. However if the platform supports multi-tasking everything can be solved.

Finally, these two implemented systems solve all the tasks set for this project. During the problem solving process, we have encountered and seen some relevant problems, and this give us details of the future work of this project.

However, the aims of doing this project do not only focus on the implementation of these two systems but also on the techniques and skills learnt. Let us see what we have learnt throughout this project.

The first technical aspect is about the AudioGraph tool. Because this project has closer relationship with the AudioGraph tool, we must study the tool in order to have a deeper understanding (especially, we must be familiar with those components involving in sound handling) as a prerequisite. Hence, we have studied the AudioGraph tool's design goal, architecture, and implementation details (only those parts are relevant to this project). This enables us to be able to advise people who will follow our work to expand the functionality of the AudioGraph tool (detail is given in the future work section). Also we

have analysed the working environments of the AudioGraph tool so that we can choose an appropriate VAD scheme for the project.

The second aspect is the VoIP techniques. This one of the hottest fields in the world is widely being researched by the organisations no matter whether they are commercial vendors or scientists. After having experienced this project, we have understood the VoIP in a deeper manner. We have learnt the VoIP's system architecture and its benefits, and we have also examined seven issues that affect the performance of VoIP system. However, there are still many standards open in VoIP because the issues involving quality of service have not yet been resolved. Also we have seen, via experiment, how hard it is to improve the quality of service during doing this project.

The third aspect is the speech coding techniques. From the analog signal to digital signal, we have understood how speech is represented and how speech is encoded. The speech coding technique is one of the most important steps in the evolution of which human communication. The evolution of speech-coding techniques has focused on QoS as well as low bit-rate. This results from the data transmission growing faster than the voice transmission. In this project, we have a chance to look at the speech coding schemes from PCM to G.723.1 recommendation (this order is according to the bit rate from large to small). This drives us to study different speech coding schemes (waveform coder, vocoder, or hybrid coder). We have understood the theories and principles behind those schemes and have known the differences between those schemes. Actually we have focused our attention to PCM and GSM schemes in this report because they are the techniques required by the project specification.

The fourth aspect is VAD technique. Nowadays this technique has been a part of speech coding technique in some ITU's recommendations. The VAD technique provides further functionality to remove the redundant information in the speech so as to achieve a lower bit rate without affecting QoS. We have examined most of the VAD techniques used and have made a comparison between different VAD schemes. Therefore, we have a clear

point of view--that is—most of the VAD schemes are based on a compromise because different schemes trade off delay, sensitivity, accuracy and computational cost.

The fifth aspect is the Internet relevant protocols. The current network infrastructure of the VoIP is the Internet protocol. We have examined its structure and functionality in a little bit deeper manner because it is the prerequisite for this project. Through comparing the different protocols, we have understood why network transmission is non-deterministic and have seen that researchers are putting their efforts manage this. As the power of computers has grown rapidly, we have seen that more sophisticated techniques have been used to improve the network infrastructure and this solves the drawbacks of the current network infrastructure. Also understanding the TCP/IP protocol helps us to achieve a better comprehension of those issues in the VoIP.

The last aspect is the final conclusion of this project—that is—we have learnt how to tackle a task and how to solve a problem. By this project, we have known how to do the research, how to absorb the knowledge from other people's work and how to break down the task into smaller tasks so as to be able to solve them one by one. Likewise, we have understood one thing that, having a correct attitude to knowledge is one of the most important things to doing a research.

5.2 Future work

The VAD algorithm used in this project is simple and has a low cost. Also it has good performance in the working environment like that of the AudioGraph. It is not hard to surmise the most obvious and urgent task would be to incorporate the VAD recorder into AudioGraph. Actually this is very simple. In order to do this, we have already separated the interface relevant code and other codes of the VAD recorder, which has been bundled

into a library VADlib. This makes the porting very easily, because the interface for every system is different but the core of the VAD algorithm should remain the same.

Therefore, to add the VAD algorithm into AudioGraph involves in the following steps:

- Add the interface elements into AudioGraph, including the VAD recorder and its preference setting.
- Map variables in preference setting to those static variables in the VAD recorder.
- Add variables declaration. Define a global variable to keep track of the recorder.
- Add recording handling code. What we suggest is that, every time, when the user clicks on the record button, initialise the recorder--simply call the recorder initialisation code. Then start the recording by calling the recorder's **StartRecording** function. When the user clicks on the stop-recording button, stop recording and destroy the recorder.
- The speech packet list that is used to maintain all packets (including packets' relevant information like pause) has been defined in the VAD recorder as a static member. So, even though we destroyed the VAD recorder, it still exists. Please do not forget to delete it when we exit the system or after we have done the serialisation, by calling a piece of code like CVADRecorder::mQueue. **EmptyQueue**().
- Maintain graphical views. There must be a graphical view corresponding to the speech packet list in AudioGraph. The user can select or delete any packet of which s/he does not want or satisfy.
- Modify the class AsynchronousSound of the AudioGraph, which provides playback functionality. The new player should only playback a packet of which the user has selected at a time. Please refer to the class CVADPlayer in System-B.
- Add serialisation code into where AudioGraph handles saving the presentation.

As for the memory issues, we must pre-allocate one megabytes memory for the use of the VAD recorder. This should always be reserved for the recorder. However, according to presentation of different length, the actual memory size used in the AudioGraph should be reconsidered.

The VAD algorithm has some constraints however, for example, it can not work well in a low SNR value environment. However, as the power of the computer increases, this gives us a chance to choose a more sophisticated scheme such as ANN (Artificial Neural Network) scheme in the future.

As for using a simplex channel to handle the echo cancellation, we can completely resolve the drawback of this method by using a more sophisticated echo cancellation method. The echo generated follows this formula:

$$\kappa_{mixedsignal} = f_{speech} + \chi_{playbacksignal} \qquad (1)$$

where the right hand side of formula (1) is the signal we have recorded, which comprises of the clear speech signal (without noise) f_{speech} and the echo source $\chi_{playbacksignal}$. Because we have all $\chi_{playbacksignal}$ data in buffer, if we can remove $\chi_{playbacksignal}$ signal, we will get clear speech signal only. ANN is a good way to achieve this but requires a lot of power from the computer. When the power of computer is no longer a problem, ANN will be a good candidate.

Our System-B provides multicast support, but it is achieved in the application layer. This does not reduce the bandwidth. Hence we must introduce the RTP protocol to add into the future system. Also the RTP supports multicast and saves bandwidth, by letting packets go to all its destinations from the network directly.

Also the RTP provides a lot of information to remove the jitter such as time-stamping information, etc. We have not provided any jitter handling technique in our system because we only test it under a local LAN. Adding RTP protocol would also to expand the jitter handling function into the future system.

Reference:

[1] "Inside Macintosh: Sound/ Chapter 2 – Sound Manager", retrieved from the world-wide web on 25/2/2001, http://developer.apple.com/techpubs/mac/Sound/Sound-46.html

[2] "What is VoIP", 2000, retrieved from the world-wide web on 25/2/2001, http://www.innomedia.com/ip_telephony/voip/index.htm

[3] Yashvant Jani (2000), "Network Talk: Voice over IP", retrieved from the world-wide web on 25/2/2001, http://www.embedded.com/internet/0008/0008ia2.htm

[4] Vinodkrishnan Kulathumani, Voice over IP: Products, Services and Issues, retrieved from the world-wide web on 25/2/2001, http://www.cis.ohio-state.edu/~jain/cis788-99/voip_products/index.html (17 of 20) [2/7/2000 10:47:16 AM]

[5] "H.323: Gatekeeper functionality", retrieved from the world-wide web on 25/2/2001, http://support.wpine.com/MP4/doc/help/basic7.htm

[6] "Voice-over-IP Overview", 1999, retrieved from the world-wide web on 25/2/2001, http://www.ieng.com/univercd/cc/td/doc/product/access/acs_mod/1750/cnfg_gds/voip_cnf/intro.htm

[7] R. V. Cox, and P. Kroon, "Low bit-rate speech coders for multimedia communication," IEEE Comm. Mag., pp. 34-40, Dec., 1996.

[8] G. Held, "Voice over data networks," McGraw-Hill, 1998, ISBN 0-07-028135-1.

[9] M. Handley etc. (1999), "SIP: Session Initiation Protocol", retrieved from the world-wide web on 25/2/2001, http://www.landfield.com/rfcs/rfc2543.html

[10] M. Arango etc. (October, 1999), "Media Gateway Control Protocol (MGCP)", retrieved from the world-wide web on 25/2/2001, http://www.landfield.com/rfcs/rfc2705.html

[11] M. Arango etc (July 1998), "Simple Gateway Control Protocol", retrieved from the world-wide web on 25/2/2001, http://www.globecom.net/ietf/draft/draft-huitema-sgcp-v1-02.html

[12] W. Yeong etc. (1995), "Lightweight Directory Access Protocol", retrieved from the world-wide web on 25/2/2001, http://www.landfield.com/rfcs/rfc1777.html

[13] J. Case etc. (May 1990), "A Simple Network Management Protocol (SNMP)", retrieved from the world-wide web on 25/2/2001, http://www.landfield.com/rfcs/rfc1157.html

[14] Texas Instruments Inc. (2000), "Texas Instruments Honored with 'Product of the Year' Awards for IP Phone Chipset", retrieved from the world-wide web on 25/2/2001, http://www.ti.com/sc/docs/news/2000/00013.htm

[15] "Quick Time 4 API Documentation: Inside Macintosh: Sound", retrieved from the world-wide web on 26/11/2000, http://developer.apple.com/techpubs/quicktime/qt4beta/INMAC/SOUND/imsoundmgr.30.htm

[16] "Cameras & Conferencing", retrieved from the world-wide web on 25/2/2001, http://support.intel.com/support/videophone/trial21/h323_wpr.htm

[17] Vanguard 6560, retrieved from the world-wide web on 25/2/2001, http://www.motorola.com/networking/products/vanguard_6560/faqs.html#3

[18] Cisco System Inc. (1999), "Weighted Random Early Detection on the Cisco 12000 Series Router", retrieved from the world-wide web on 25/2/2001, http://www.ieng.com/univercd/cc/td/doc/product/software/ios112/ios112p/gsr/wred_gs.htm#xtocid94550

[19] M. A. Labrador and S. Banerjee : "Packet Dropping Policies for ATM and IP Networks," IEEE Comm. Surv. Third Quar. Vol. 2 no. 3 1999.
retrieved from the world-wide web on 25/2/2001,
http://www.comsoc.org/pubs/surveys/3q99issue/banerjee.html

[20] A. Demers, S. Keshav, and S. Shenker: "Analysis and Simulation of a Fair Queuing Algorithm," In Proceedings of ACM SIGCOMM, pp. 1-12, Sept.1989.

[21] R. Braden, etc. (September, 1997), "Resource ReSerVation Protocol (RSVP)", retrieved from the world-wide web on 25/2/2001, http://www.landfield.com/rfcs/rfc2205.html

[22] Cisco System Inc. (2000), "Data Sheet Cisco IP Phone 7960", retrieved from the world-wide web on 25/2/2001, http://www.cisco.com/warp/public/cc/pd/tlhw/prodlit/7960_ds.htm

[23] "New Solution from TI and Telogy Networks Decreases IP Phone Manufacturing Costs to Standard PBX Phone Levels", retrieved from the world-wide web on 25/2/2001, http://www.telephonyworld.com/roundup/telogyip.htm

[24] "Netmeeting Zone", retrieved from the world-wide web on 25/2/2001, http://www.netmeet.net/nm3_faq.asp

[25] "Desktop Video Collaboration Software", retrieved from the world-wide web on 25/2/2001, http://www.cuseeme.com/software/pro_bus.htm#features

[26] S. Kotha, "Deploying H.323 Applications in Cisco Networks",retrieved from the world-wide web on 25/2/2001, http://www.cisco.com/warp/public/cc/pd/iosw/ioft/mmcm/tech/h323_wp.htm

[27] "CWSApps Listing (with download) for Microsoft NetMeeting", retrieved from the world-wide web on 25/2/2001, http://cws.internet.com/reviews/iphone-netmeet2.html

[28] S. Glass (February, 1997), "Telecommunications Systems: An Introductory Guide", retrieved from the world-wide web on 25/2/2001, http://www.gtlaw.com.au/pubs/telcosysintroguide.html

[29] Metrowerks Corp., 1998, "CodeWarrior the PowerPlant book", retrieved from the world-wide web on 25/2/2001, http://cs.beloit.edu/shields/Documentation/CodeWarrior%20Documentation/HTML/PowerPlant_Book_html/PPB000_Front.fm.html

[30] "VocalTec Gatekeeper" , retrieved from the world-wide web on 25/2/2001, http://www.vocaltec.com/iptelephony/products/vgk/vgk_pdf.htm

[31] D. Essex, "IP telephony: what does the future hold," retrieved from the world-wide web on 25/2/2001, http://courses.washington.edu/is470red/iptelephonyfuture_NetworkMag.html

[32] "Industry-Leading Voice Over IP Gateway Selected for Voice Quality and Ease of Implementation", retrieved from the world-wide web on 25/2/2001, http://www.ieng.com/warp/public/146/pressroom/2000/may00/sp_050800.htm

[33] "Interoperability is the clear winner", retrieved from the world-wide web on 25/2/2001, http://www.tundo.com/news/coverage/1999coverage/pdf007.pdf

[34] P. Bernier (May, 1999), "Quicknet Embraces Linux for IP", retrieved from the world-wide web on 25/2/2001, http://www.soundingboardmag.com/articles/951whats.html

[35] "An Essential Tool For The Definition, Control And Management Of H.323 IP Telephony And Multimedia Networks", retrieved from the world-wide web on 25/2/2001, http://www.viewcom.com/gatekeep.html

[36] "Mac OS 8 and 9 Developer Documentation", retrieved from the world-wide web on 25/2/2001, http://developer.apple.com/techpubs/macos8/mac8.html

[37] "Multimedia Web Tools", retrieved from the World Wide Web on 25/2/2001, http://www.nzedsoft.com/audiographhomepa.html

[38] "Multimedia Authoring Systems FAQ version 2.23, April 1999, retrieved from the World Wide Web on 25/2/2001, http://www.tiac.net/users/jasiglar/MMASFAQ.HTML

[39] R. Gehne and C. Jesshope: "Tools for the production of small-footprint, low-bandwidth, streaming multi-media for distance education," retrieved from the world-wide web on 25/2/2001, http://www.nzedsoft.com/links.html

[40] "Features of the AudioGraph Tools", retrieved from the World Wide Web on 25/2/2001, http://www.nzedsoft.com/infoaudiograph.html

[41] C. Jesshope . et al: " Low-bandwidth, multi-media tools for web-based lecture publishing," IEE Engineering Science and Educational Journal, 7 (4), pp148-154. retrieved from the world-wide web on 25/2/2001, http://www.nzedsoft.com/links.html

[42] European Telecommunication Standard ETS 300 580-6, May 1994.

[43] D.K. Freeman, et al, "The voice activity detector for the Pan-European digital cellular mobile telephone service," International Conference on, vol.1, pp. 369 –372,1989.

[44] S.G. Tanyer and H. Ozer, "Voice activity detection in non-stationary noise," *IEEE Transactions on Speech & Audio Processing, pp.478-82, July 2000.*

[45] F. Beritelli, S. Casale and A. Cavallaero, "A robust voice activity detector for wireless communications using soft computing," *IEEE Journal on Selected Areas in Communications, pp.1818-29, Dec. 1998.*

[46] K. Bullington and J. M. Fraser, "Engineering aspects of TASI," Bell Syst. Tech. J., pp. 353-364, Mar. 1959.

[47] J. Ikedo, "Voice activity detection using neural network", IEICE Trans. Commun., VOL. E81-B. NO. 12 Dec., 1998.

[48] J. C. Junqua, et al, "A study of endpoint detection algorithms in adverse conditions: Incidence on a DTW and HMM recognize," in Proc. Eurospeech' 91, 1991, pp. 1371-1374.

[49] R. Tucker, "Voice activity detection using a periodicity measure," Proc. Inst. Elect. Eng., vol. 139, pp. 277-380, Aug., 1992.

[50] H. Ozer and S. G. Tanyer, "A geometric algorithm for voice activity detection in nonstationary Gaussian noise," Proc. EUSIPCO'98, Rhodes, Greece, Sep. 1998.

[51] "User Guide to vat", retrieved from the world-wide web on 25/2/2001, http://ivs.cs.uni-magdeburg.de/~elkner/Mbone/tools/user-vat.html

[52] S. Cassidy (2000), "SLP801: Introduction to Speech Processing", chapter 5, retrieved from the world-wide web on 25/2/2001, http://www.shlrc.mq.edu.au/masters/801/x289.html

[53] "signal-to-noise ratio (SNR)", August 1996, retrieved from the world-wide web on 25/2/2001, http://www.its.bldrdoc.gov/fs-1037/dir-033/_4849.htm

[54] S. Cassidy (2000), "SLP801: Introduction to Speech Processing", chapter 5, retrieved from the world-wide web on 25/2/2001, http://www.shlrc.mq.edu.au/masters/801/time.html#AEN277

[55] M. R. Wski, and W. R. Young, " Consequences of Nyquist Theorem for Acoustic Signals Stored in Digital Format," Proceedings, Acous.Wk. Canada, 1991. retrieved from the world-wide web on 25/2/2001, http://www.digital-recordings.com/publ/pubneq.html#theorem

[56] E. Turbo (Fall, 1993), "INTRODUCTION TO BASIC DIGITAL TECHNOLOGY", Xenon Foundation, retrieved from the world-wide web on 25/2/2001, http://www.etext.org/Zines/ASCII/Xenon/xf0001.txt

[57] M. Robin (August, 2000), "Sampling," retrieved from the world-wide web on 25/2/2001, http://www.broadcastengineering.com/html/2000/august/digitalHandbook/transitionToDigital_0800.htm

[58] F. Rumsey and T. McCormick, "Sound and Recording: An Introduction," Focal Press, 1992, ISBN 0-2405-1313-4.

[59] "An introduction to Macintosh programming for windows programmers", retrieved from the world-wide web on 25/2/2001, http://devworld.apple.com/macos/macoverview/prog06.html

[60] A.M. Kondoz, "Digital speech," John Wiley & Sons Publishing 1995, ISBN 0-471-95064-5.

[61] J. Degener, "Digital Speech Compression," retrieved from the world-wide web on 12/12/2000, http://kbs.cs.tu-berlin.de/~jutta/gsm/degener.htm

[62] R. John, et al, "Discrete-Time Processing of Speech Signals," New York, Macmillan Publishing, 1993, ISBN 0-7803-5386-2.

[63] B. Atal and J. Remde, "A new model of LPC excitation for producing natural sounding speech at low bit rates," Proc. Of ICASSP, pp. 614-617, 1982.

[64] S. Singhal and B. Atal, "Amplitude optimisation and pitch prediction in multipulse coders," IEEE Trans. On Assp, pp.317-327, March 1989.

[65] B. Fete, "High quality secure voice communication," Speech Technology, pp. 40-48, Oct.,1989.

[66] A. M. Kondoz, and et al, "CELP base-band coder for high quality speech coding at 9.6 and 2.4 kb/s," Proc. Of ICASSP pp.159-162, NY,USA, May 1988.

[67] B. A. Forouzan, "TCP/IP protocol suite," McGraw Hill Publishing, 2000, ISBN 0-256-24166-X, http://www.mhhe.com

[68] M. G. Naugle, "Network Protocols," McGraw-Hill Publishing1998, ISBN 0-07-046603-3.

[69] "MPEG Audio Layer-3", Fraunhofer IIS-A, 1998 retrieved from the world-wide web on 25/2/2001, http://www.iis.fhg.de/amm/techinf/layer3/index.html

[70] "MP3", retrieved from the world-wide web on 25/2/2001, http://webopedia.internet.com/TERM/M/MP3.html

[71] "STA013 MPEG 2.5 LAYER III (MP3) SOURCE DECODER", June, 1999, retrieved from the world-wide web on 25/2/2001, http://www.st.com/stonline/books/ascii/docs/6526.htm