Playout Buffering for Conversational Voice over IP

Qipeng Gong

Department of Electrical & Computer Engineering
McGill University
Montreal, Canada

October 2012

A thesis submitted to McGill University in partial fulfilment of the requirements of the degree of Doctor of Philosophy.

2012/10/11

Abstract

In Voice over IP, the quality of interactive conversation is important to users. Major factors affecting perceived quality are delay, delay jitter, and missing packets. For conversational VoIP, a conversational delay also plays an important role for perceived quality. Large conversational delay can result in double talk, echo or even the termination of the conversation. In practice, a playout buffer is introduced at the receiver's side to remove delay jitter, so that the voice information carried on packets can be available at regular intervals for decoding. A longer buffer reduces the possibility of late packet loss at the expense of increasing conversational delays. Since the time delay of a playout buffer is a major addition to conversational delay, to keep conversational interactivity, it is desirable to design a playout buffer to be short but capable of protecting packets against late packet loss.

In this thesis, we will explore playout buffering algorithms with improved conversational quality. We propose a quality-based adaptive playout buffering algorithm with improved voice quality and reduced conversational delays. We use the E-Model R factor as the cost index to obtain playout delays which adapt for each talkspurt. Special steps are taken to reduce conversational delay : (1) immediately play out stretched speech carried on the first packet of a talkspurt when received (stretching provides additional buffer delay for following packets) ; (2) compress the speech segment carried on the packets in the playout buffer at the end of a talkspurt (compression reduces the playout delay for the packets).

As other quality-based algorithms, our scheme is subject to burst losses. To improve perceived quality further, we use sender-driven repair algorithms, in which a sender sends redundancy information, to mitigate the impact of the missing packets due to network (lost packets) and buffer underflow (late packets) without increasing buffer delays. In this thesis, we develop a new adaptive forward error correction (FEC) scheme to provide redundancy without additional delay and apply it to our adaptive playout buffering algorithm for improved perceived quality. As an alternative sender-based technique to send redundancy information, a path diversity scheme uses multiple paths (here we consider two paths). Redundant information is sent on a second path. We consider four different path diversity schemes (two of them are proposed based on E-model in this work), and design corresponding playout buffering algorithms based on conversational quality including both calling quality and interactivity.

Sommaire

Dans le domaine de la voix sur IP (VOIP), la qualité de conversation interactive est importante pour les utilisateurs. Les principaux facteurs qui affectent la qualité à la réception sont le retard, la désynchronisation et les pertes de paquets. Lors d'une conversation par voix sur IP, le retard joue un rôle important pour la qualité à la réception. Les délais prolongés de conversation peuvent provoquer de la double parole, de l'écho ou encore la fin de la conversation. En pratique, un jeu de buffers (ou mémoirer tampon) est introduit du coté du récepteur pour supprimer les délais indésirables. Ainsi, l'information de la voix contenue dans les paquets peut être disponible à des intervalles de temps réguliers pour le décodage. Un buffer plus grande réduit la possibilité de perdre des paquets en retard aux dépens de voir augmenter les délais de conversation. Comme la capacité du jeu de buffers est une somme de retards de conversations, pour garder l'interactivité de la conversation, il est préférable de concevoir un jeu qui soit court mais également capable de protéger les paquets contre les pertes dues aux retards.

Dans cette thèse, nous explorons les algorithmes de jeux de buffers améliorant la qualité de conversation. Nous proposons un algorithme adaptatif en termes de qualité et qui a pour but d'augmenter la qualité de la voix et de réduire les retards dans la conversation. Nous utilisons le facteur R du modèle-E comme indice de coût pour obtenir les délais du jeu qui s'adaptent à chaque "talkspurt" (segment continu de parole inséré entre les temps de silence). Des étapes spécifiques sont entreprises pour réduire le retard de conversation : (1) traiter immédiatement la parole étirée qui est contenue dans le premier paquet lors de la réception d'un "talkspurt" (l'étirement fournit des délais de buffers supplémentaires pour suivre les paquets) ; (2) compresser le segment de parole contenu dans les paquets au niveau du jeu de buffers à la fin du "talkspurt" (la compression réduit les délais du jeu pour les paquets).

Comme les autres algorithmes portant sur la qualité, notre montage est sujet aux pertes de signaux en rafale (burst loss). Pour améliorer encore plus la qualité à la réception, nous utilisons des algorithmes de réparation par émetteur. L'émetteur envoie de l'information redondante pour atténuer l'impact des paquets manquants causés par le réseau (pertes de paquets) et par le manque de capacité des buffers (paquets retardés), tout ceci sans augmenter les délais de buffer. Dans cette thèse, nous développons un nouveau schéma adaptatif de correction d'erreurs sans voie de retour (FEC) pour fournir de la redondance sans ajouter

de délais. Nous appliquons ce dispositif de correction à notre algorithme adaptatif de jeu de buffers pour améliorer la qualité perçue. En tant que technique alternative d'envoie d'information de redondance, un schéma fournissant de la diversité de trajet utilise plusieurs chemins (nous en considérons deux ici). L'information de redondance est envoyée vers le deuxième chemin. Nous considérons quatre schémas de diversité de trajet (deux d'entre eux sont proposés par rapport au modèle-E utilisé dans ce travail). Nous concevons également des algorithmes de jeu de buffers avec comme critères de qualité de conversation : la qualité d'appel et l'interactivité.

Acknowledgments

I am heartily thankful to my supervisor, Peter Kabal, for his commitment, motivation, time, and support throughout my studies. This thesis would not have been possible without his encouragement, guidance and understanding. Many thanks go to Joachim Thiemann, who is always ready to offer help and advice. Thank Fabien Sacuto for his terrific work on my French abstract. Also appreciate valuable suggestions from Prof. F. Labeau and Prof. B. Champagne.

Special thanks to my family – my parents, my husband, and beloved Gavin, for their love and understanding.

Contents

List of Figures

List of Tables

Acronyms

ACELP	Algebraic Codebook Excitation Linear Prediction (Speech Codec)
ACR	Absolute Category Rating
AMR	Adaptive Multi-Rate (Speech Codec)
AR	Autoregressive
ATA	Analog Telephone Adaptor
CCR	Comparison Category Rating
CDF	Cumulative Distribution Function
CELP	Codebook Excited Linear Prediction (Speech Codec)
CN	Comfort Noise
CNG	Comfort Noise Generation
DCR	Degradation Category Rating
DSP	Digital Signal Processing
DTX	Discontinuous Transmission
EFR	Enhanced Full Rate
EGP	External Gateway Protocol (Network Protocol)
EVRC	Enhanced Variable Rate Codec (Speech Codec)
FEC	Forward Error Correction
FIFO	First-in First-out
FR	Full Rate
GSM	Global System for Mobile communications (Speech Codec)
HNM	Harmonic Plus Noise Model
HR	Half Rate
ICMP	Internet Control Message Protocol (Network Protocol)
IETF	Internet Engineering Task Force
IGP	Interior Gateway Protocol (Network Protocol)
iLBC	Internet Low Bit Rate Codec (Speech Codec)
IP	Internet Protocol (Network Protocol)
ISP	Internet Service Provider
ITU	International Telecommunications Union
LAN	Local Area Network
LP	Linear Prediction

LPC	Linear Predictive Coding
LSP	Linear Spectral Frequencies
MCU	Multipoint Control Units
MG	Media Gateway
MGC	Media Gateway Controller
Megaco	Media Gateway Control Protocol (Network Protocol)
MGCP	Cisco Media Gateway Control Protocol (Network Protocol)
MOS	Mean Opinion Score
MSE	Mean Square Error
NLMS	Normalized Least Mean Squares
OLA	Overlap Add
PBX	Private Branch Exchange
PCM	Pulse Code Modulation
PDF	Possibility Distribution Function
PESQ	Perceptual Evaluation of Speech Quality
PLC	Packet Loss Concealment
PLR	Packet Loss Rate
PSTN	Public Switched Telephone Network
PWSOLA	Packet-based WSOLA
QoS	Quality of Service
RAS	Registration, Admission, and Status
RMSE	Root Mean Square Error
RSVP	Resource Reservation Protocol (Network Protocol)
RTP	Real-Time Protocol (Network Protocol)
RTCP	Real-Time Control Protocol (Network Protocol)
SBC	Sub-band Coding
SID	Silence Insertion Descriptor
SIP	Session Initiation Protocol (Network Protocol)
SNR	Signal-Noise-Ratio
SOLA	Synchronized Overlap-and-add
SON	Service Overlay network
SQ	Scalar Quantization
TCP	Transmission Control Protocol (Network Protocol)

TDHS	Time-domain Harmonic Scaling
TSM	Time Scale Modification
UAC	User Agent Client
UAS	User Agent Server
UDP	User Datagram Protocol (Network Protocol)
VoIP	Voice over IP
VPN	Virtual Private Network
VQ	Vector Quantization
WSOLA	Waveform Similarity Overlap-and-add

Chapter 1

Introduction

1.1 VoIP

The Internet has changed the way that people communicate with each other. More people have become used to sending messages by email instead of writing in mail. Twenty years ago, telephones were majorally used for voice communications. These days, voice conversations can be delivered through an Internet connection as long as an Internet service with reasonable quality is available. People talking with friends using a laptop is a very common scene on a campus, in a cafe, or at an airport. With the growth of the Internet and the development of Internet Protocol (IP) based applications, the term VoIP has become increasingly popular.

VoIP is an acronym for Voice over Internet Protocol. It means that voice traffic is transmitted in packets over IP networks. Other terms frequently used synonymously with VoIP are Internet telephony, IP telephony, packet-voice, and packetized voice. There are many well known computer applications that make it easy to use VoIP such as Skype, MSN Messenger, Yahoo Messenger, Google Talk and so on. In VoIP applications, a family of voice processing and transmission techniques are involved to provide voice communications by utilizing packet-switched IP networks, instead of using a traditional circuit-switched network, e.g., Public Switched Telephone Network (PSTN).

In VoIP, the following technologies and protocols are involved to provide a basic telephone functionality:

– Session control protocols or media control protocols to control the set-up and tear-down of calls

- Digital signal processing (DSP) techniques for conversion between analog voice signal and digital voice signals
- Speech coding techniques to compress/decompress voice signals
- Transport protocols over IP networks to deliver voice packets

1.2 How does VoIP work?

In VoIP, voice communications are delivered by IP networks. In order to use VoIP, the parties of a conversation need to be connected to IP networks. For VoIP calls, there are three types of endpoints which can be connected to an Internet connection: a computer, an IP phone, and a normal phone with an analog telephone adaptor (ATA). Computer-to-computer connection is the original and simplest form of VoIP. To make a VoIP call, the parties are required to be online simultaneously and run the same VoIP software on their computers. Most software packages, e.g., Skype, GoogleTalk, Yahoo Messenger, are free; furthermore, the cost of a microphone/headset is very low. Hence, this type of VoIP is the most economical for long-distance/international calls. An IP phone looks like a normal phone with a handset, cradle and buttons. Instead of having the standard RJ-11 phone connector, an IP phone has an RJ-45 Ethernet connector which connects directly to a router. With IP phones, there is no need for computers: IP phones have all the hardware and software applications necessary built-in to handle VoIP calls. An ordinary telephone can be connected to a router or a modem via an ATA. An ATA is an analog-to-digital converter, which converts the analog signal received from a traditional phone into digital data for transmission over IP networks. When using IP phones or phones with ATA, an access fee is generally charged by VoIP service providers.

To make a VoIP call, signaling protocols, i.e., session/media control protocols, are used to establish, hold on and terminate call sessions or media connections. Signaling protocols are integrated into the VoIP softwares running on computers or IP phones. If a conventional telephone is used, the ATA converts the analog control signals (e.g., touch tone and hold request) into digital signals and sends them over IP networks.

IP networks are packet-switched, meaning that the unit of digital data exchanged between network nodes is in form of "a packet". Therefore, after the call has been established, the analog human voice needs to be digitalized and packetized for transmission. In most cases, the voice signal is sampled, segmented into frames of 10–30 ms length. Frame by

frame, a speech codec[1], e.g., ITU-T G.711, GSM-EFR, etc, is applied to compress and quantize the discrete-time samples into a stream of digital data. The digital data is then encapsulated into IP packets. With a specified signalling protocol (e.g., ITU-T H.323), these voice packets travel to the receiver side as individual network packets across the IP network. Routers in the transmission path simply forward packets to the next nodes according to routing protocols, e.g., interior gateway protocols (IGPs) or external gateway protocols (EGPs).

Once a voice packet arrives at its destination, the digital data which it carries is converted back to an analog signal. Again, the conversion can be done by VoIP software on a computer/an IP phone, or an ATA.

Figure 1.1 shows the basic architecture of VoIP processing.

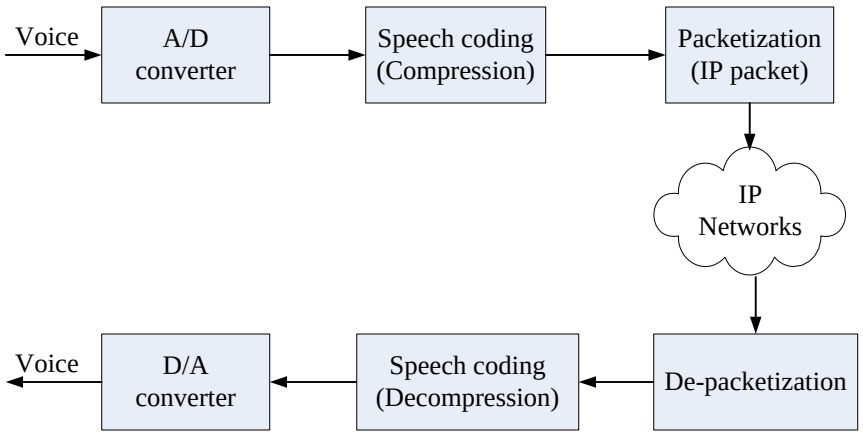


Figure 1.1 Basic architecture of VoIP

1.3 Why VoIP?

As a technology that transmits voice data over IP networks and is fully integrated into the contemporay digital lifestyle, VoIP has been gaining widespread acceptance since 2005 [7]. It is widely accepted that VoIP will dominate the field of voice communications, due to various advantages:

1. A speech codec is an algorithm for the compression/decompression of speech [6].

low cost One key advantage of VoIP is saving money. For customers, VoIP service costs much less money than conventional telephone service does, especially for long distance and international calls. Running over existing IP networks, VoIP does not require a dedicated connection. Packet-switched transmission allows more efficient use of the available bandwidth, resulting in lower costs [8]. Another reason for the low cost of VoIP is that the VoIP service is free of government regulation. Hence, VoIP service providers are not subject to the special fees and taxes that apply to most traditional phone companies [8]. These savings, which in effect cut the cost of doing business, can be passed along to customers in the form of lower prices.

VoIP users benefit from low prices if they have Internet access available. Note that most VoIP users pay for Internet service for multiple purposes, e.g., online research, email, digital TV, etc. Many PC-to-PC VoIP softwares are free, for example, Skype. Without a PC and a Internet connection, it is still possible to get free VoIP service, for example, Google Voice provides US customers free Phone-to-Phone calls within US & to Canada with existing phone lines.

Easy to use Making a call from a computer is very simple: just open the VoIP software, and click the "call" button. When using an IP phone or a home phone with a VoIP adapter for voice communication, the call is made in the same way as a conventional telephone: pick up the phone, wait for the dial tone, and dial the number.

New features VoIP introduces new ideas for voice communications. Besides making a call, VoIP services cover other features of traditional telephones in a simple way. For example, an answering machine on a normal phone records the incoming calls that you can not pick up. VoIP has similar functionality called voicemail. With voicemail, missed calls are emailed to a specified mailbox as audio file attachments. You can call someone back with a "click".

VoIP uses IP networks as a backbone, which are converged networks of voice and data. When used on a computer, a VoIP software works just like other applications running on the computer; as such, it shares the internet connection with the other applications as well. For example, you can check your email or download software while you talk. Based on the same reasoning, VoIP also provides new services such as video calls, i.e., talking "face-to-face" with a web camera; VoIP conferencing, which connects multiple parties in different places by a single VoIP call; and more. While VoIP can be augmented with video transmission, in this work we focus on voice

transmission.

Moreover, with an Internet connection, a personal VoIP account can be accessed any place in the world and retain features like Caller ID, Contact lists, extra-virtual numbers, and so on.

Secure conversation Although various security problems exist when using the Internet, VoIP can provide secure voice communication for businesses or institutes by using Virtual Private Networks (VPN) or their own managed private networks.

Promising better quality The bandwidth limitation of 300–3400 Hz makes traditional telephone speech sound weak, unnatural, and lack crispness. Analog distortion due to poor connections or old circuits are significant sources of quality degradation in conventional telephony. Since voice information is transmitted in digital form over IP networks, VoIP avoids analog degradation and potentially provides improved quality by extending the bandwidth. Besides that, most VoIP applications support wideband speech codecs, e.g., ITU-T G.722.2 (AMR-WB), which provide more natural voice than narrowband codecs.

1.4 How VoIP differs from traditional telephony

VoIP differs from traditional telephony primarily in terms of infrastructure: VoIP works on packet-switched IP networks whereas conventional telephone service is based on circuit-switched networks, e.g., PSTN.

PSTN is based on landline networks and provides a dedicated circuit, over which narrowband analog voice data is carried by copper wires, between two endpoints for the duration of a call. Once a circuit is set up, the channel is reserved for the conversation. Hence, PSTN can provide good calling quality with constant delay. But a dedicated circuit results in the underutilization of capacity, since the components used in the circuit are not available for use until the call ends [9, Chapter 1].

In contrast, the bandwidth of IP networks is shared by all packets traversing across the network, i.e., multiple traffic is carried over one IP connection. Therefore, VoIP achieves low costs and high bandwidth utilization. The voice information in IP packets is digital, and not restricted to being narrowband. Various speech codecs, including narrowband and wideband codecs, can be used in VoIP.

However, IP networks were not initially designed with voice conversation in mind. Network delay is time-varying due to various queueing delays which are effected by the load of links and nodes. For example, when a node is overloaded, packets are held in a queuing buffer at the node, and these packets are forwarded to the next hop until the load is at its normal level. In this case, these packets have longer delays than those forwarded when the burden of the node is not heavy. Moreover, it is possible for packets to be discarded due to collision when competing with other packets for sharing the same transmission links. In addition, unlike PSTN, which is a managed network (via SS7 signaling and central office processing), IP networks have a noticeable lack of management intelligence to handle the traffic. Hence, IP networks only provide "best effort" delivery and unreliable services.

Even though PSTN is optimized for voice traffic, the performance of other data traffic, such as multimedia and video conferencing, is poor if carried on the PSTN. IP networks support integrated voice and data service by simultaneously carrying voice and data packets.

1.5 Quality of Service for VoIP

Even though low price/cost is the main reason for customers to choose VoIP, high-quality VoIP service is expected. The expectation is based on the fact that customers have become accustomed to consistently high conversational quality provided by the circuit-switched public telephone network, e.g., PSTN. However, IP networks were not designed for real-time applications like VoIP. There is no central monitoring or performance measurement to track or maintain the state of the network [10]. Therefore, quality of service (QoS) can not be guaranteed for current VoIP services.

As other real-time applications, VoIP is sensitive to bandwidth and delay. Over IP networks, bandwidth pipes are shared, and voice packets have no priority over other kinds of network traffic. IP networks are unmanaged and prone to congestion and malicious attacks, e.g., DoS attacks, [11]. For example, during peak usage time, the load on a link can grow so quickly that routers are significantly overburdened, resulting in queue overflows. This network congestion can cause some voice packets to be dropped or introduce excessive latencies to the packets. The dropped packets never arrive at the destination, and the late-arriving packets with large delay are discarded if they are too "old" to be played out. These missing packets degrade the perceived quality of VoIP dramatically, because the

voice information carried by these packets is not available for the decoder at the receiver side to reconstruct speech signals.

Packets are delayed due to the queuing and processing of network equipments, and delay is not constant. IP routing is dynamic for every packet and the network maintains no state of the path of prior packets. Every time a packet arrives at an IP router, the router makes an individual decision about where to send the packet. Therefore, it is possible that packets are routed on different paths to their destination, resulting in different delays. Since the availability of links and nodes is dynamic to each packet, delay varies over time. Variation in delay is called delay jitter, and it is particularly problematic for VoIP services. Delay jitter can occur due to network congestion, timing drift, or route changes. It breaks the time regularity in which packets are packetized at the sender side, and hence complicates the decoding process at the receiver side to reconstruct the speech signal, resulting in choppy voice or temporary glitches.

All these impairments, i.e., packet missing, delay, jitter, stem from the unmanaged design of IP networks. QoS mechanisms, e.g., capacity management, packet prioritization and network monitoring, can ensure that VoIP applications achieve a high quality of service. Capacity management is related to the connection to a wide-area network or other access links [12, page 313]. The idea of prioritization techniques is to set higher priority for voice packets over less time-critical data packets. This type of mechanisms reduce undesirable packet dropping, delay and jitter without a noticeable effect on the data traffic. Many standards and protocols are available to implement differentiated services, including IEEE 802.1D [13], DiffServ [14], and the resource reservation protocol (RSVP) [15], [12, page 313]. As the state of networks changes over time, it is important to implement ongoing monitoring and management of networks [12, page 313]. Various protocols have been developed to report QoS for VoIP, including RTCP extended report (RFC 3611) [16], SIP RTCP summary reports, H.460.9 Annex B (for H.323) [17], and H.248.30 [18]. For example, RFC 3611 VoIP metrics reports are intended to provide real-time QoS feedback. The reports are exchanged during a call between endpoints for call quality calculation. The metrics carried in RFC 3611 VoIP Metrics block include packet missing (packet dropping and late-arrival discarding) metrics, delay metrics, analog metrics (signal/ noise/ echo level) and voice quality metrics (Mean Opinion Scores (MOS) and R factor) [16, Section 4.7].

Obviously, it is easy to implement QoS managements in isolated or private networks. When the traffic is routed through an unmanaged network, how to apply QoS mechanisms

to improve perceived quality is still an open and challenging area.

1.6 Playout buffering in VoIP

Much effort has been paid to improve perceived quality of VoIP. For example, delay can be reduced by marking voice packets using DiffServ [19]. In [20], multi-path routing is proposed to alleviate the impact of temporary failures on perceived quality by transmitting the streams of packets across different routes. Note that all these proposals are for a future version of the Internet, and not available over the current standard Internet. Besides, various playout buffering algorithms at the receiver side have been developed to compensate for the jitter effect.

As VoIP packets experience delay jitter, a playout scheme without a sufficient playout buffer leads to an increasing rate of late packet loss, which dramatically impacts on the perceived quality. A playout buffering algorithm schedules playout deadlines for incoming packets by managing a playout buffer at the receiver side. The incoming packets are held in the buffer when waiting for their playout time. In this way, network delay jitter is removed. To determine the size of a playout buffer, end-to-end delay is traded against late packet loss. Both a too long buffer and a too short buffer degrade perceived conversational quality of VoIP. If the buffer is designed with a small size, e.g., less than network delay jitter, the packets with large latency are discarded at the destination as if they never arrived. This case is called buffer underflow. On the other hand, a long buffer increases end-to-end delay, resulting in a large conversational delay which impedes the interactivity of conversations. The interactivity of a conversation is considered to be transparent if the end-to-end delay is less than 150 ms [21] (corresponding to 300 ms of conversational delay). The ITU-T recommends that the upper limit of end-to-end delay is 400 ms in ITU-T G.114 [21]. For applications which experience long network delays, it is desirable to keep the size of playout buffers short to avoid introducing too much additional delay.

Therefore, the goal of a playout buffering algorithm is to protect VoIP packets against late arrival with a reasonably small buffering delay. The traditional approach is to first hold the incoming packets in a packet buffer for a short time and then send the packets to a speech decoder in desired spaced intervals. A playout buffer is not simply a strict first-in first-out (FIFO) buffer. For example, some packets are in a long queue due to congestion, and the following packets are routed to the destination through another path with shorter

transmission delay. In this case, packets arrive at the destination out of order and the playout buffering scheme should be able to reorder the packets. Some speech codecs, e.g., ITU-G.711, ITU-G.729, etc, apply discontinuous transmission (DTX) techniques to reduce the overall bit rate, and hence the voice packets are sent in unevenly spaced intervals. A playout buffering scheme needs to align packets using their timestamps (carried in the RTP (Real-Time Protocol) header) so that the decoder can decode them at the same regularity of time intervals as they were sent at the sender side.

Over the past 30 years, many playout buffering schemes to handle playout buffers have been developed in literature. The most straightforward approach is to design the buffer with a fixed size, e.g., [22], [23], and [24]. This results in a constant end-to-end delay for all VoIP packets during a call. This approach requires no computations and provides minimum complexity [12, page 316]. However, as the state of IP networks changes dynamically, this fixed design requires the buffer size to be sufficiently large so that the worst case delay jitter can be accommodated. Otherwise, perceived quality would be degraded because some late-arrival packets are discarded due to buffer underflow. Therefore, a fixed playout buffer may lead to large additional buffering delays, which impair conversational quality.

Accordingly, to avoid unnecessary additional delay during low-congestion conditions and to achieve the best trade-off between end-to-end delay and late packet loss, it is important for a playout buffering algorithm to adjust its buffer size to be consistent with dynamic network conditions. A playout buffer with a changing size is called an adaptive playout buffer [25]. In an ideal case, the best perceived quality of VoIP applications can be achieved when the size of an adaptive buffer is exactly equal to network delay jitter. However, the amount of delay jitter for a packet is not available until it arrives at the receiver side due to the unknown variability of IP networks. To playout speech signals continuously, the playout buffering algorithm needs to allocate the buffer size for each packet prior to its arrival. Therefore, it is currently impractical to set the buffer size for a packet according to its network delay jitter. Most existing adaptive algorithms adapt playout buffer size based on the estimation of playout delay for packets.

Human speech consists of silence and one or more talkspurts. For an adaptive playout buffering algorithm, adaptation can be “intra-talkspurt” or “between-talkspurt”. Intra-talkspurt adaptation techniques adjust the buffer size by setting different playout delays for each packet. Playout delay is estimated based on monitoring network conditions, e.g., previous end-to-end delay, delay jitter, packet loss rate, etc. The buffer size is tuned

by scaling previous packet(s) using time-scale modification (TSM) techniques. Examples can be found in [26], [27], and [28]. On the other side, a between-talkspurt adaptation adjusts buffer size during silence periods by simply inserting/shortening silence duration or scaling packets at the beginning of a talkspurt (see [29]). Compared with intra-talkspurt adaptations, a between-talkspurt method is characterized by low complex implementation, and also avoids updating the buffer size too often in case of a connection with mild or low delay variations. However, between-talkspurt algorithms are more vulnerable to high-delay spikes [2] than intra-talkspurt ones. When delay spikes happen, intra-talkspurt approaches can adapt the buffer size almost immediately when large values of delay are detected, whereas between-talkspurt algorithms have to wait for the end of the talkspurt. To alleviate the impact of resulting burst loss on quality, many algorithms, e.g., [5], [30], [31], integrate a spike detection algorithm so that the buffer can be adapted to the delay spikes.

Early adaptive playout buffering algorithms aim to track network delay variation and adjust the playout delay on the basis of network delay statistics: mean and variation. Playout delay for incoming packets is obtained by the estimated mean plus β times the estimated variation. Various algorithms were proposed to estimate mean and variation. The classic one is in [31], which is based on an autoregressive (AR) average of the running delay. The approach in [32] generalizes the basic algorithm in [31] by adaptively adjusting the weighting factor to an optimal value for the specified loss rate [30]. Instead of reacting to a fluctuation of network delay, adaptive filter based algorithms, e.g., [33] and [30], intend to predict network delay. In these approaches, a normalized least mean squares (NLMS) filter is used to estimate the network delay from a window of previous packet delays. The basic idea is to minimize the expected mean square error between the actual delay and the estimated delay. The mean square error is then used to adjust the tap weights of the adaptive filter. Both AR based and adaptive filter based algorithms are simple to implement. They work very well under stable (low jitter) network conditions, but tend to provide high playout delays in case of high network delay jitter. Most algorithms involve a spike detection technique for the sudden delay variation.

Instead of directly using the statistics of network delay, some playout buffering algorithms, for example, [34], [35], link the network delay with packet loss rate (PLR) using delay distribution. Given a fixed threshold of PLR, the playout delay is set to the smallest

2. A *spike delay* is defined as a sudden, large increase of network delay followed by a series of packets arriving almost simultaneously (at a high frequency), leading to the completion of the spike [31].

possible delay so that the corresponding PLR is lower than the threshold. In [35], a histogram of network delay of past packets is used for estimation. In [34], delay distribution is estimated by statistical models with a heavy tail. The Pareto distribution was claimed to provide better results than other functions, i.e., Logonormal, Exponential, Normal, using a chi-squared test. In [36], a probability density function (PDF) of delay is estimated from past packets. The playout delay is set to the percentile point q in the PDF [37]. In [27], delay distribution is estimated by order statistics. All these algorithms bound playout delay with a predetermined PLR, and accordingly avoid excessive additional delay.

Since 2003, several quality-based algorithms have been developed which consider both losses and delays, for example, [5], [38], [39], [29]. The basic idea is to seek an optimum balancing of delay versus late packet loss based on the ITU-T E-Model quality measurement. The E-Model is standardized in ITU-T recommend G.107 [1] for the objective quality assessment of conversational quality. In E-Model, the overall quality index, R factor, is calculated from additive impairments which are the underlying causes of speech quality problems. The impairment metrics include delay distribution, loss rate, frame erasure rate, loss-concealment techniques, architecture choices such as de-jitter (playout) buffer, and packet and codec frame size [37]. In E-Model based playout buffering algorithms, the playout delay is set by the end-to-end delay, which achieves optimum value of the R factor. This type of playout buffering is more generalized than the approaches using only delay distribution.

In E-Model based approaches, it is also important to estimate delay distribution. The accuracy of the estimation affects the effectiveness of the subsequent optimization [37]. In [5], a priori distribution, Weibull, is compared with Pareto and Exponential distributions and is found to provide the best fit according to the RMSE (Root Mean Square Error) measure. Instead of making any assumptions on the shape of the distribution, [38] and [29] use a statistical model based on histograms.

Even though E-Model based playout buffering algorithms are supposed to achieve optimum balance between delay and late packet loss, perceived quality is degraded when burst loss occurs. A burst period is defined as an interval of time during which high packet missing happens [37]. The bursts are separated by gaps, which are characterized by sporadic loss events [37]. There are two main sources of burst loss: consecutive packet dropping by network equipments, e.g., due to link failure or network congestion, and delay spikes, resulting in successive packets arriving too late to be played out. Most PLC algorithms, which

are built in most speech codecs, fail to fill in the gap if the length of burst loss exceeds 60 ms. Some algorithms, e.g., [5], include spike detection techniques to deal with the sudden increments/decrements of network delay. A spike can be with a sudden or gradual increase [5]. For example, in [5], the playout buffering algorithm uses spike detection proposed in [36], which uses two thresholds to determine the state (no-spike or spike) for each coming packet. At the beginning of a talkspurt, if a spike is detected, the delay of the first packet is used as the playout delay for the talkspurt. Otherwise, the playout delay is estimated based on maximizing the R factor.

For conversational VoIP, conversational delay also plays a very important role for call quality. Large conversational delay can result in double talk, echo or even the termination of the conversation. Not much work has addressed perceived conversational delay in playout buffering schemes. Some algorithms, e.g., E-Model based algorithms, bound conversational delay by penalizing long end-to-end delay. But this bound is somehow loose. Therefore, perceived quality can be improved if a playout buffering design involves explicit steps of reducing conversational delay.

1.7 Motivations and aim

The motivation of this work is as follows:

> Since perceived quality is important to users of conversational VoIP, it is desirable to design playout buffering algorithms based on perceived conversational quality.

In quality-based playout buffering algorithms, perceived quality is the key metric used as a cost function for optimization. The E-Model is suitable for designing a playout buffering algorithm, since it links perceived quality directly to network impairments.

According to the definition in [40], conversational delay consists of 3 parts: end-to-end delay of User 1's ending packet of the last talk-spurt, end-to-end delay of User 2's first packet of the first talk-spurt and response delay. Response delays are the time intervals used for User 2's thinking, which vary for different people. For simplicity and without losing generality, response delays are set to be zero in this work. Due to this definition, conversational delay can be reduced by compressing the "ending" packets. Note that this compression just "speeds up" the speech segment in buffer without changing timbers.

Therefore, perceived conversational quality can be improved if E-Model based playout buffering algorithms incorporate with the process of reducing conversational delay. However, as we discussed in Section 1.6, the performance of an E-Model based algorithm is impacted by missing packets, especially burst loss. Hence, it is desirable to further improve perceived quality by reducing the number of missing packets without adding further delay.

Overall, our aim is to develop a series of playout buffering algorithms with improved conversational quality. To reach this objective, the following approaches are used:

- E-Model

  Concerning packet missing and end-to-end delays, the R factor in the E-Model [1] is used as the cost index for the maximization of voice quality. E-Model based algorithms require the estimation of delay distribution. We noticed that the playout delay may be very sensitive to the type of distribution used [38]. Therefore, the statistical model based on the histogram is used in this work.
- G.729 VAD

  According to the definition of "hangover" in [41], a "hangover" packet is a non-speech packet which is sent as a speech packet to avoid speech clipping. In this work, our "hangover" detection algorithm is based on ITU-T G.729 VAD algorithm (details in Appendix A).
- Conversational delay is reduced by gradually decreasing playout buffer depth at the end of the "talk-spurt".

  In [2], conversational delay is reduced by playing out the first packet of a conversational turn when it is received and compressing the speech segment decoded from the "hangover" packets. The buffer size is accumulated gradually up to a predetermined value. "Hangover" packets are used to trigger compression processing to reduce playout buffer size. During "hangover", the system resets itself to a small or zero buffer size. In this way, the buffering delays at the start and end of a conversational turn are small, but most packets during talkspurts are protected by a buffer with a certain size.
- Packet-based Waveform Similarity Overlap-Add (PWSOLA)

  PWSOLA [26] is used to stretch and compress speech packets, which is explained in Chapter 2.
- Redundant information sent by Forward Error Correction (FEC) to improve conversational quality.

FEC [42] is used to mitigate the impact of packet losses by sending redundant information. There are two types of FEC schemes: *media-independent FEC* and *media-dependent FEC*. *Media-independent FEC* uses block codes to provide redundant information, while *media-dependent FEC* piggybacks the redundant information onto the subsequent packets.

To use the redundant information in *media-dependent FEC*, the decoder must implement a delay. However, since a playout buffer is already present at the receiver, there needs to be no additional delays if *media-dependent FEC* is integrated with the jitter protection algorithm. The redundant information in any FEC scheme implies an increase in bit rate. To keep the rate down, the redundant information can be encoded compactly, perhaps entailing a small loss in quality in the case of packet loss. To lower the overall bit rate, separate *primary* and *redundant* encoding can be used to code the redundant information using a lower rate-compression method, resulting in a lower quality for the recovered packet [12, Chapter 15]. For example, G.711 (64 kb/s) as the main payload can be combined with GSM (13 kb/s) or G.729 (8 kb/s) for the redundant information. It is to be noted that the speech payload of VoIP is small, and so even for G.711, the packet overhead for a 20 ms IP packet is 25%. For the lower rate coders, the overhead is a much larger fraction. Doubling the payload does not double the packet length.

– Path diversity schemes to improve conversational quality by sending redundant information on a second path.

For some network conditions, FEC schemes fail to provide redundant information, especially in the case of a long burst loss. A path diversity scheme is an alternative technique, in which redundant information is sent on a second path. If the loss and delay characteristics of the two paths are uncorrelated, path diversity schemes are robust to burst losses.

Either IP source routing or Relay approaches can be used to implement path diversity schemes. With IP source routing, special configurations are required for all nodes that a packet should visit on route to its destination [38]. Therefore, all ISPs (Internet service providers) should support source routing, and it is currently difficult to achieve. Relay approaches use relays placed at a number of nodes to forward a packet to its destination. In this work, Relay approaches are used.

1.8 Thesis contributions

1. A new adaptive playout buffering based on estimated quality and conversational delay (published in [29])

   The algorithm is designed by two parts. R factor in the E-Model [1] is used as cost index. The static playout delay for a talkspurt is set based on optimizing R factor. The conversational delay is reduced by using "hangover" detection to trigger the compression of perceived voice to reduce the playout delay at the end of a talkspurt. As other quality based algorithms, the estimation of delay distribution is required in our scheme. We use a histogram based model due to the fact that the playout delay may be very sensitive to the particular type of distribution used in [5].

2. A practical way to calculate conversational delay based on human perception (published in [29])

   According to the definition of conversational delay, it is important for its calculation to detect the start and the end of a talkspurt. VoIP packets travel on Real-Time Protocol (RTP) over User Datagram Protocol (UDP). The voice information is carried in a payload with a IP/UDP/RTP header. In RFC 3550 [43], the first voice packet after a silence period, is discriminated from other packets by setting its M field as 1. We use this property to detect the start of a talkspurt. We also developed a scheme to detect the end of a talkspurt based on the "hangover" in Section 3.2.4.

3. A new buffer aware *media-dependent* scheme (published in [44])

   At the sender's side, m previous voiced packets are added to the packet. The piggybacking stops whenever "hangover" is detected. The value of m is specified by an RTCP packet sent from the receiver. This buffer-aware FEC scheme avoids sending redundant information which cannot be used by the receiver.

4. A quality-based playout buffering algorithm with this new *media-dependent* scheme (published in [44])

   E-Model based algorithms are still subject to packet losses under certain network conditions. Burst losses cause more degradation on perceived quality than isolate losses, since most packet loss concealment (PLC) algorithms can not efficiently fill in the lost frames. Most PLC schemes are designed to gradually mute the output when consecutive frames are erased. Perceived quality can be improved if packet losses,

especially burst losses, are reduced without increasing the apparent conversational delay.

FEC schemes can mitigate the impact of packet missing by providing redundancy information. However, additional delay needs to be implemented at the receiver side for utilizing redundancy information. The new playout buffering algorithm incorporates previously developed E-Model based algorithm with the new buffer aware *media-dependent* scheme, so as to achieve improved conversational perceived quality without adding delay for redundancy.

5. Quality-base playout buffering algorithms with different path diversity schemes

   With conversational interactivity in mind, the algorithms incorporate our E-Model based algorithm with path diversity schemes for improved conversational perceived quality. It is complicated to integrate the impairments of two path into one cost function, especially for estimation of network delay distribution. For the first two path diversity schemes, we process the packets on both paths in two ways:
   - Scheme 1: The processed delay for each packet is the network delay of the packet which arrives first.
   - Scheme 2: One path is selected as the primary path and the other as the redundant path. For each packet, the processed delay is the network delay of the packet received from the primary path. If the packet on the primary path fails to arrive before its scheduled time, the network delay of the corresponding packet from the redundant path is used as the processed delay.

   The processed delay is then used in the adaptive playout buffering algorithm instead of network delay.

   Instead of treating the two paths as one, we developed two schemes which estimate playout delays separately on both paths. The new schemes are based on full redundancy. When used with E-Model based playout buffering algorithms, this scheme evaluates the performance for both paths. Each path is treated as the primary path with the other as the redundancy path, as in Scheme 2. Two playout delays are calculated based on maximizing R factors on two paths. One path diversity scheme chooses the minimum one as the static playout delay for a talkspurt (this scheme is published in [45]. The other scheme uses the playout delay which achieves higher R value as the static playout delay for a talkspurt.

6. Investigation of the robustness to burst losses among playout buffering algorithms

The impact of the packet missing problem can be alleviated by using redundancy information. FEC and path diversity are two classes of sender-driven techniques to provide redundancy information for improved quality. With different expectations of burst loss length, we compared the performances among the algorithms developed in this work: our E-Model based adaptive playout buffering algorithm, our E-Model based playout buffering algorithm with our new *media-dependent* scheme, our E-Model base playout buffering algorithms with different path diversity schemes. The performances of these algorithms are evaluated by PESQ (ITU-T P.862).

1.9 Thesis structure

The outline of this thesis is as follows:

Chapter 1 is the introduction of this thesis. In this chapter, we give a brief overview of VoIP. We also state the motivations and objective of this thesis in Section 1.7. The contributions of this work are presented in Section 1.8.

Chapter 2 gives brief background information on VoIP and some techniques used in this work. The VoIP protocol stack, system, packet form, and signaling are presented in Section 2.1. Speech coding technology, speech properties and main codecs used in VoIP applications are overviewed in Section 2.2. In Section 2.3, we discuss the network components affecting the perceived quality of VoIP applications, e.g., delay, jitter, and packet missing. The calculation of conversational delay is presented in Section 2.3.2. Speech quality assessments, including the PESQ and the E-Model, are discussed in Section 2.4. In playout buffering, time scale modification (TSM) techniques are used to change buffer size by scaling speech, so that continuous speech can be played out. In Section 2.5, we give a brief introduction of TSM, and present a modified WSOLA algorithm, PWSOLA, which is used in our playout buffering algorithms.

Chapter 3 investigates playout buffering techniques in VoIP. We start from the simplest approach – fixed playout buffering in Section 3.1. Several adaptive buffering algorithms are presented in Section 3.2, including E-Model based algorithm proposed by Sun and Ifeachor in [5]. In Section 3.3, we introduce a semi-fixed buffering algorithm proposed in [2], which reduces conversational delay by compressing "ending" packets. In Section 3.4, we present our new E-Model adaptive buffering algorithm for

conversational VoIP, which considers both voice quality and conversational delay.

Chapter 4 introduces methodologies to improve conversation quality by redundancy information. We investigate the effect of burst loss on perceived quality using PESQ in Section 4.1. Forward error correction (FEC) are introduce in Section 4.2. Our new FEC scheme is presented in Section 4.3.1 and the quality-based playout buffering algorithm with this new scheme is presented in Section 4.3.2. Path diversity schemes, including our new path diversity scheme, are discussed in Section 4.4. In Section 4.5, the robustness of our playout buffering algorithms to burst loss is investigated by comparing the performances of these algorithms, i.e., E-Model based adaptive playout buffering algorithm (without redundancy), and E-Model based playout buffering algorithms with redundancy, incorporating with the new *media-dependent* scheme, and path diversity schemes.

Chapter 5 presents experiments and results to show the efficacy of our playout buffering schemes.

Chapter 6 reviews the work done, presents the conclusions and suggests future work.

Chapter 2

VoIP Background

Voice over IP (VoIP) delivers voice communications over Internet Protocol (IP) networks. In VoIP, the analog voice signal is sampled and encoded into digital stream. The digital data is encapsulated into IP packets and then transmitted over the Internet. As a widely accepted alternative of traditional telephone service, VoIP provides telephone service without using completely separate systems and communications infrastructure. However IP networks were not originally developed for real-time applications such as VoIP. To provide real-time services, e.g., telephony, using VoIP, Real-Time Protocol (RTP) is used in conjunction with a signaling protocol, e.g., H.323, SIP, which sets up and terminates calls, carries information required to locate users and negotiates capabilities.

With VoIP, users can make a call from a PC, an IP phone, or a normal phone instrument with an analog telephone adaptor (ATA). The architecture of VoIP is shown in Figure 2.1. An IP phone looks like a normal telephone and allows calls to be made over IP networks instead of PSTN. Codecs and signaling protocols are integrated in an IP phone and for some business applications, a compatible local private branch exchange (PBX) is used to handle calls to and from outside the local area network (LAN). If an ordinary phone involved, a gateway (i.e., ATA) is deployed before the Internet. In a gateway, analog voice received by telephone is digitalized and packetized into IP packets. The packets are sent over the IP network to the entrance of the destination gateway, where the process is reversed.

VoIP is a merged technology of speech processing, network protocols, and transmission technologies. Therefore, speech processing and VoIP networks are two major parts of VoIP. We will describe the VoIP network, including protocols, VoIP system, VoIP packets and signaling, in Section 2.1. Speech coding and commonly used codecs are introduced in

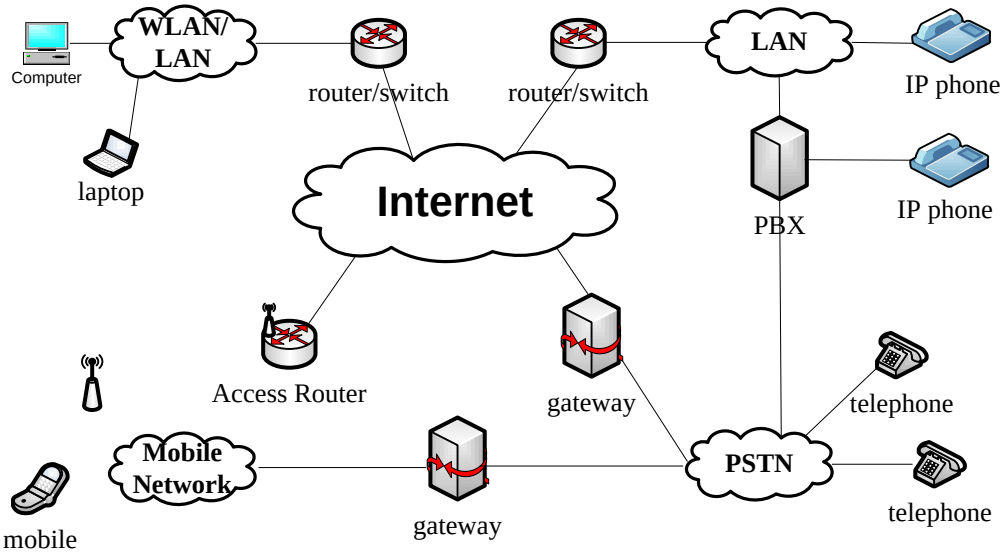


Figure 2.1 VoIP Architecture.

Section 2.2. Quality is another important issue in VoIP. Hence, the network components which affect perceived quality and quality measurements are discussed in Section 2.3 and Section 2.4 respectively. We also introduce a time scale modification (TSM) technique in Section 2.5, which is used in this work to implement VoIP playout buffering.

2.1 VoIP networks

2.1.1 VoIP protocol stack

As its name implies, VoIP runs over the network layer, and uses IP as its basic transmission protocol. Therefore, VoIP can work with other protocols supporting IP, and accordingly it is convenient for users to involve VoIP systems to their existing infrastructure. VoIP also uses the transport layer which consists of the Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). Associated with real-time applications, VoIP transmits encoded voice stream using Real-time Transport Protocol (RTP) which is sent over UDP. In VoIP, TCP, the reliable transmission protocol, is used for signaling (e.g., H.323). Figure 2.2 illustrates the VoIP protocol stack which uses H.323 as the signaling protocol. Session Initiation Protocol (SIP), another signaling protocol, is an application

layer protocol which is independent of the underlying transport layer. In Section 2.1.4, we will give a brief overview of the commonly used signaling protocols, such as, H.323, SIP, Media Gateway Control Protocol (Megaco), and Cisco Media Gateway Control Protocol (MGCP).

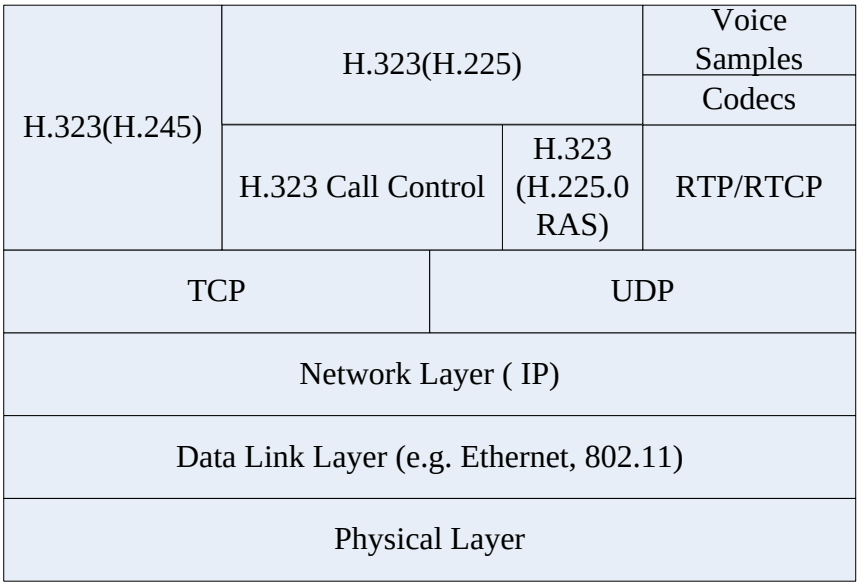


Figure 2.2 VoIP Protocol Stack.

2.1.2 VoIP system

The VoIP system used in this work is shown in Figure 2.3. At the sender side, speech from a user end is first digitalized and compressed by a speech coder, and then the encoded bit stream is encapsulated into IP/UDP/RTP packets, whose format follows RFC 3550 [43]. Packets carrying speech information are sent to IP networks. Passing through IP networks, packets can be dropped or delayed due to various schemes running in network equipments, e.g., routers, switches, etc. Two consecutive packets can arrive at the receiver side with different delays (we will explain the part in details later in Section 2.3.1). Such variation of delay is called delay jitter, and hence the regularity of time interval of these packets is disrupted by the time when they reach to the destination.

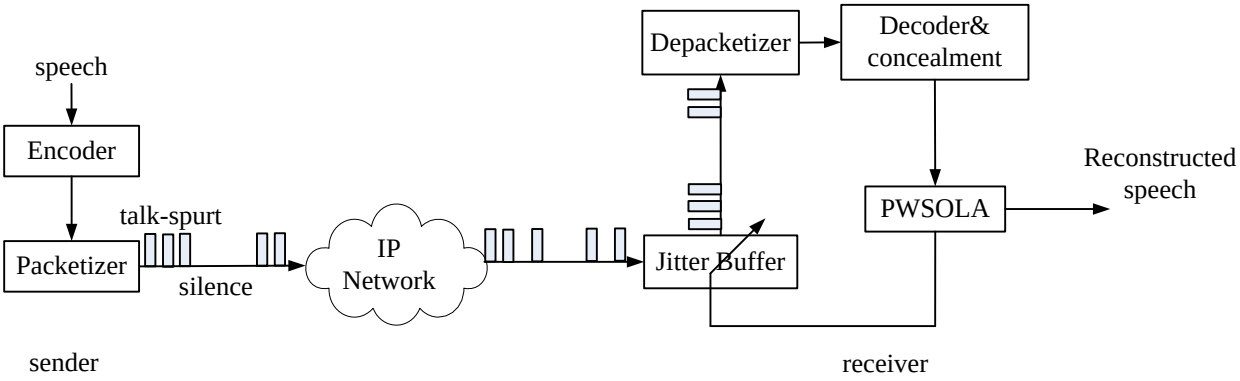


Figure 2.3 VoIP System.

To compensate the delay jitter caused by various delays over the network, a playout buffer is used at the receiver side. A playout buffering algorithm schedules arrival packets to be played out with proper buffer delay, such that the original time interval between two consecutive packets can be kept. If a packet arrives later than the scheduled time, it is called a late packet. Late packets are treated in the same way as the lost packets which are dropped during transmission due to link failure, transmission error or dropping strategies of network equipments. The system cannot use the information carried by late packets to reconstruct speech. Inherently included in speech codecs, packet loss concealment (PLC) techniques are typically used to handle missing packets (late and/or lost packets). For our adaptive playout buffering presented in this work, time-scale modification (TSM) is used to change the length of played-out speech signal so that the size of a playout buffer is varied by $(\alpha-1)\times T_F$, where α is the scaling factor and T_F is the original length of speech segment carried in a packet. When $\alpha > 1$, the speech segment carried in packets is stretched without affecting timber (pitch, tone, etc.). This stretched speech segment sounds like "slowing down" the original speech and does not affect the understanding of the speech. Since the played-out speech is longer than the original one, more time ($(\alpha - 1)\times T_F$) is available for waiting for the next packet, i.e., the playout buffer increases by $(\alpha - 1)\times T_F$. When $\alpha < 1$, the speech segment is "speeded up" without affecting timber. The playout buffer decreases by $|(\alpha - 1)\times T_F|$, since the speech segment is compressed to a shorter segment.

Note that for conversational applications, a playout buffer is used at both ends for the received packets.

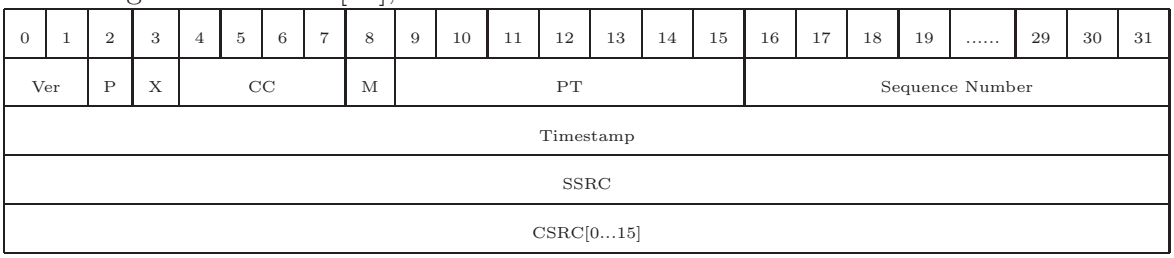
2.1.3 VoIP packets

In VoIP, a speech signal is delivered in packet format over IP networks. RTP is used for end-to-end, real-time, transfer of multimedia data. The encoded speech stream forms the payload part in a RTP packet. In RTP payload, RFC 3551 defines the payload for different codecs. The size of a packets depends on the length of speech segment and the speech codec used. RTP information is then encapsulated in a UDP packet. UDP information is then encapsulated in a IP packet before sent to the network. A IP/UDP/RTP packet is illustrated in Figure 2.4. The IPv4 header contains 20 bytes of data, UDP header contains

Figure 2.4 VoIP Packet.

8 bytes, and RTP header has minimum size of 12 bytes. Hence, the total length of a IP/UDP/RTP packet's header is at a minimum 40 bytes. If speech data is coded by G.711 (A-law/μ-law PCM, 64kbits/s), the overhead for a 20 ms IP/UDP/RTP packet is 25%.

According to RFC 3550 [43], the RTP header is



Ver: Version. 2 bits. RTP version number. Always set to 2.

P: Padding. 1 bit. If set, this packet contains one or more additional padding bytes at the end which are not part of the payload.

X: Extension. 1 bit. If set, the fixed header is followed by exactly one header extension.

CC: CSRC count. 4 bits. The number of contributing source (CSRC) identifiers that follow the fixed header.

M: Marker. 1 bit. The marker is set to 1 in the case that the packet is the first speech packet after a silence period. In our playout schemes, it is used to detect the start of a talkspurt.

PT: Payload Type. 7 bits. The value is different for different codecs. For example, when used with G.711, PT is 13 for comfort noise, 8 for a speech signal using A-law, and 0 for a speech signal using μ-law.

Sequence Number. 16 bits. The sequence number increments by one for each RTP data packet sent, and may be used by the receiver to detect missing packets and to restore packet sequence. The initial value of sequence number is picked randomly.

Timestamp. 32 bits. The timestamp reflects the sampling instant of the first octet in the RTP data packet. The sampling instant must be derived from a clock that increments monotonically and linearly in time to allow synchronization and jitter calculations.

Each RTP stream has a random initial value of timestamp. In practice, the timestamp for a voice packet is incremented by the packetization interval times the sampling rate. For example, for packets containing 20 ms of speech sampled at 8,000 Hz, the timestamp for each block of speech increases by 160, even if the block is not sent due to silence suppression [46].

SSRC: Synchronization source. 32 bits. Identifies the synchronization source.

CSRC: Contributing source. 32 bits. An array of 0 to 15 CSRC elements identifying the contributing sources for the payload contained in this packet.

2.1.4 Signaling protocols for VoIP

VoIP signaling protocols are used to set up and tear down calls, carry information required to locate users, and negotiate capabilities. There are two classes of VoIP signaling protocols: **session control** protocols and **media control** protocols. **Session control** protocols are responsible for establishing, holding on and terminating call sessions. They also carry the negotiation information of session parameters such as codec, tones, bandwidth capabilities, etc. The main session control protocols in the IP network are H.323 and Session Initiation Protocol (SIP) [47]. **Media control** protocols set up and tear down media connections by controlling media gateway controller on VoIP gateways and processing notifications from other gateways. Gateways transport media flows between the IP and PSTN networks. Media control protocols, e.g., Cisco Media Gateway Control Protocol (MGCP) and Media Gateway Control Protocol (Megaco), are used for communication to the Media Gateways (MGs) [48, Page 220]. They instruct the MG to connect streams coming from outside a packet network to a packet stream such as RTP [48, Page 220].

In this subsection, we introduce two **session control** protocols – H.323 and SIP, and two **media control** protocols – Cisco MGCP and Megaco (H.248). H.323 and SIP are peer-to-peer protocols, while MGCP and Megaco are master/slave protocols. Different from others, the current standardized SIP has no architecture that describes the decomposition

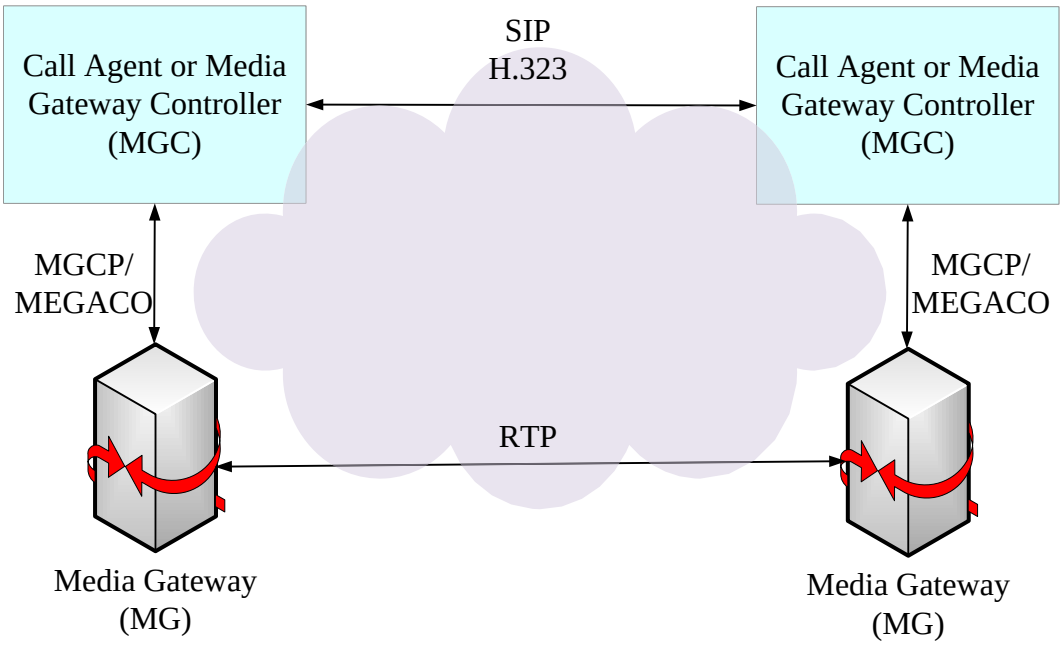


Figure 2.5 MGCP and Megaco (H.248).

of the gateway into the Media Gateway Controller and the Media Gateways [49]. It was designed as a generic transaction protocol for session initiation not bound to any specific media such as audio or video [50]. H.323 and Megaco are designed to accommodate video conferencing as well as basic telephony [51]. Assuming that more intelligence resides in a separate MGC, H.323 gateways have more call control function than the media gateways using MGCP/Megaco [51]. The goal of the Megaco architecture is to couple PSTN networks and their services (based on non-intelligent terminals) with the Internet. In the current Internet, Megaco and MGCP are used as complementary protocols for H.323 and SIP. Figure 2.5 shows the basic diagram of Cisco MGCP and Megco. For a complete signaling, a session initiation protocol (SIP/H.323) is required for communication between Media Gateway Controllers (MGCs).

H.323

H.323 is the first VoIP standard defined by the International Telecommunications Union Telecommunication Standardization Sector (ITU-T). It was originally developed for multimedia conferencing on local area networks (LANs), but was later extended to cover VoIP

[47]. H.323 was the first VoIP standard to adopt the Internet Engineering Task Force (IETF) standard RTP to transport audio and video over IP networks. The standard encompasses both point to point communications and multipoint conferences [47]. H.323 defines four logical components: Terminals, Gateways, Gatekeepers and Multipoint Control Units (MCUs) [52]. Gatekeepers provide admission control and address translation services [52]. Terminals, gateways and MCUs are known as endpoints. H.323 is made up of H.245, and H.225.0, H.225 RAS.

- H.245: A media control to establish a logical channel among the endpoints involved in a call. During H.245 negotiation, endpoints exchange the information such as capabilities, preference and the choice of codec.
- H.225.0: Call Signaling. It is used between any two H.323 entities to establish communication. Signaling messages include setup, alerting, connect, call proceeding, release complete, and facility messages (based on ITU-T Q.931).
- H.225 RAS (registration, admission, and status): The protocol establishes a logical channel between IP phone and the gatekeeper. Without appropriate RAS communication, an IP phone can not make or receive calls.

A basic H.323 call setup is shown in Figure 2.6.

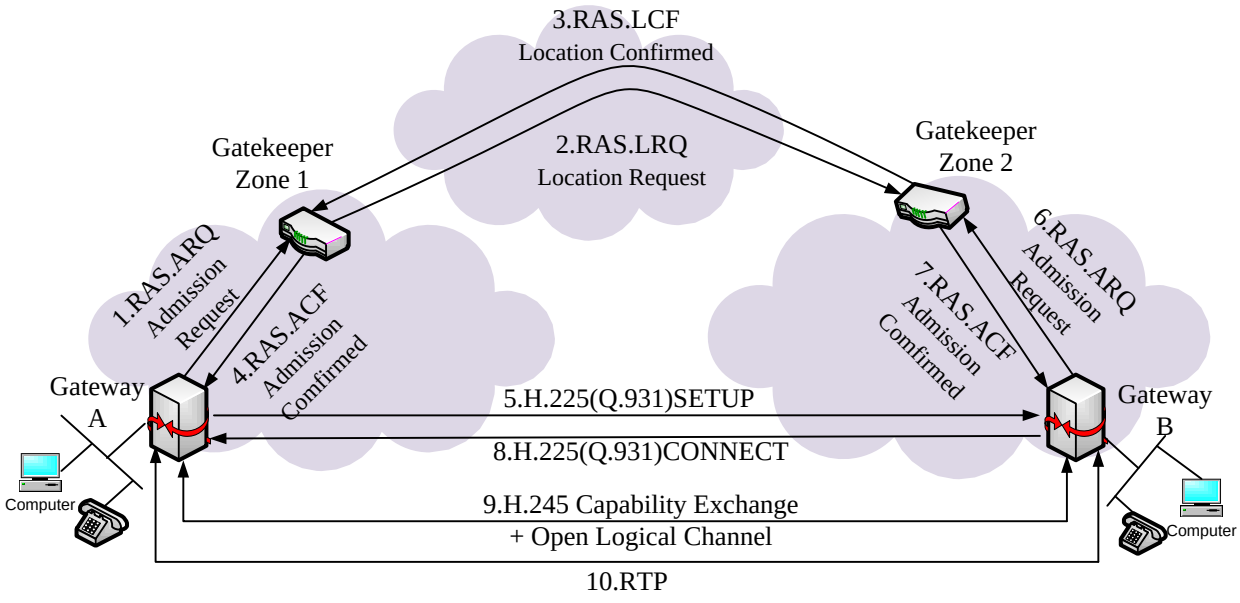


Figure 2.6 a basic H.323 call setup.

SIP

Session Initiation Protocol (SIP) is the IETF's standard published as RFC 2543 [53] for establishing VoIP connections. It was designed to initialize interactive sessions over IP networks, and it is a session-layer transaction protocol that provides advanced signaling and control functionality for a large range of multimedia communications [50]. A SIP system consists of user agents and network servers. The main entities of SIP are shown in Table 2.1. SIP is a client-to-server based protocol in which the client sends requests to the server and the server processes the requests and then sends a response to the client. A simple SIP setup is shown in Figure 2.7.

Table 2.1 SIP Entities

Components	function	
User Agent	User Agent Client (UAC)	Caller application that initiates and sends SIP requests
	User Agent Server (UAS)	Receives and responds to SIP requests: accepts, redirects, or refuses
Network Servers	Proxy Server	Contacts one or more clients or next-hop servers and passes the call requests further. It contains UAC and UAS
	Redirect Server	Does not initiate SIP requests or accept calls. Accepts SIP requests, maps the address into new addresses and returns those addresses to the client

Megaco (H.248)

The Media Gateway Control Protocol (Megaco) is a result of joint efforts of the IETF and ITU (ITU-T Recommendation H.248). It controls elements in a physically decomposed multimedia gateway, which enables separation of call control from media conversion [47]. Megaco/H.248 defines the relationship between a Media Gateway (MG) and a Media Gateway Controller (MGC). A media gateway provides a bridge to transit media between PSTN and VoIP networks, i.e., converts circuit-switched voice to packet-based traffic (RTP flow). A media gateway controller, called a call agent or softswitch, controls MGs using a media gateway control protocol. Unlike peer-to-peer in H.323 and SIP, a media gateway

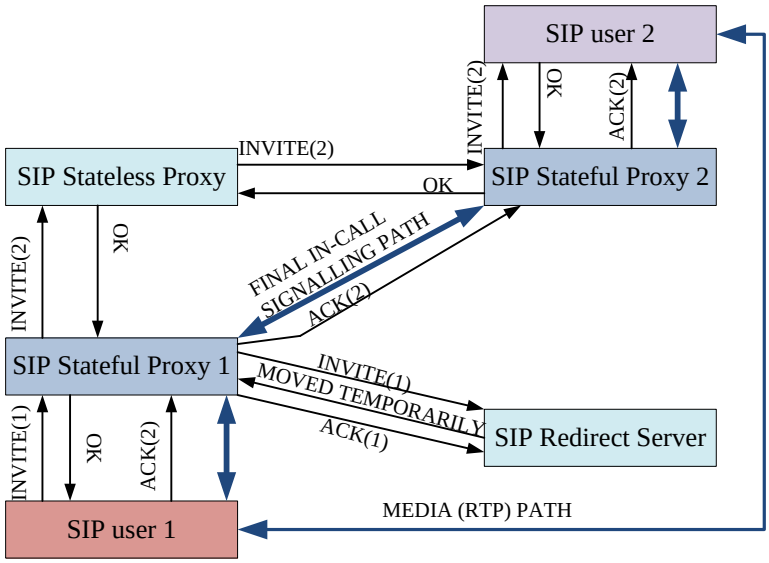


Figure 2.7 SIP Session Setup.

control protocol uses a master/slave architecture for decomposed gateways. The MGC is the master server and one or more MGs are the slave clients.

Cisco MGCP

Media Gateway Control Protocol (MGCP) is a VOIP protocol proposed by Cisco and Telcordia, which defines communication between call control elements (Call Agents or Media Gateway) and telephony gateways. An MGCP system consists of a set of IP phones, a call controller, and gatekeepers (signaling gateways and media gateways). All call connections and call control are handled by call agents, also called media gateway controller. The call agents instruct media gateways to establish and control VoIP calls. Therefore, although holding all information for making and terminating a VoIP call, media gateways have little intelligence and cannot function without the orders from call agents. Signaling gateways handle the exchange between traditional PSTN signaling and signaling protocols over IP networks to and from PSTN network.

2.2 Speech coding

As an important component in VoIP, speech coding is the process of obtaining a compact representation of speech signals for efficient transmission over wired or/and wireless

networks. Speech coding involves sampling and quantization. A continuous-time speech signal is converted to a discrete-time signal at a sampling rate equal to or greater than twice the bandwidth of the speech. A quantizer is then used to code the speech samples into bit stream, which is transmitted over networks. Speech coding can be waveform, parametric or hybrid. Waveform methods quantize the speech samples, whereas parametric methods quantize the parametric representations based on signal transformations (often unitary) or signal models (often linear source system). In this section, we first discuss some of the important speech properties which most speech coders are based on, and then briefly introduce the speech codecs and coding techniques, which are commonly used in VoIP.

2.2.1 Speech properties

Human speech signals have time varying statistics but can be considered as quasi-stationary for short segments (5–20 ms) [54]. Since both temporal and spectral analysis of the speech signal are performed in human's hearing system, the speech signal can be analyzed in frequency-domain and time-domain [55, Chapter 1]. The frequency-domain characteristics can be derived by analyzing the speech signal on a short-time basis (20–30ms). Smooth windows (such as Hann, Hamming, etc) are commonly used. According to the autocorrelation theorem [56] (Wiener-Khintchine theorem), the autocorrelation function of a signal is the inverse Fourier transform of its power spectrum. The relation exists between the autocorrelation and power-spectral domains: the fine structure of the power spectrum corresponds to the "long-term" autocorrelation of the time-domain signal, and the power-spectral envelope corresponds to the "short-time" autocorrelation.

Speech signals can be voiced (e.g., /a/,/e/), unvoiced (e.g., /sh/), or mixed. For voiced speech segments the fine structure of the power spectrum displays a harmonic structure, that is, sharp peaks in the power spectrum occur at regularly spaced frequency intervals of 75 to 400 Hz [55, Chapter 1]. The interval between the harmonics is called the fundamental frequency, which is dependent on the speaker and the utterance. Generally, the fundamental frequency of children is the highest and that of female speakers is higher than that of male speakers. Since the harmonic structure is associated to a periodic time-domain signal, voiced speech segments have a nearly harmonic frequency-domain structure and a nearly periodic time-domain sequence in [55, Chapter 1]. Unvoiced speech segments do not have the harmonic structure in the power spectrum and exhibit a noise-like structure. For natural speech, many regions of spectrum display a combination of a harmonic and a noise

structure over a large bandwidth. Generally, if the spectrum contains both harmonic and noise components, the harmonic components are more prominent at the lower frequencies, while the noise components are more prominent at the higher frequencies in [55, Chapter 1].

For speech coding, the goal is to transmit speech signals at low bit-rate with high perceived quality. Bit-rate can be reduced by removing redundant information contained in speech. Both the spectral envelope and the structure of the power spectrum contain redundant information: spectral envelope (short-time correlation) implies the redundant information contained in adjacent samples and harmonic structure of the power spectrum (periodicity) implies similarity between sequential cycles. Although unvoiced speech segments can not remove redundancy by exploit harmonic power spectrum, the bit rate required to maintain a certain perceptual quality is lower than that of voiced speech [12, Principles of Speech Coding,pages 283-305]. The reason is the fact that the human auditory system cannot distinguish between noise-like signals with identical spectral envelope and signal power contour [12, Principles of Speech Coding,pages 283-305]. Therefore, if the unvoiced speech segments are replaced with noise-like fine structure of power spectrum and similar spectral envelope, the perceived quality does not drop significantly [57]. As a result, the bit rate required for perceptually accurate reconstruction of unvoiced speech is low [12, Principles of Speech Coding,pages 283-305].

2.2.2 Speech coding

For Voice over IP transmission, the speech signal is split into frames of usually 20–30ms length. Depending on the available transmission bandwidth, i.e., the data rate on the access and core network, each speech frame is first processed by a respective speech codec for compression. Commonly, these codecs encode single speech frames at data rates of, e.g., 5.6 kbit/s (ITU-T G.723.1), 8 kbit/s (ITU-T G.729), 12.2 kbit/s (GSM-EFR), and up to 64 kbit/s (ITU-T G.711). Besides these narrow-band speech codecs (300 Hz–3.4 kHz audio bandwidth), the use of wide-band speech codecs (50 Hz–7 kHz audio bandwidth) with superior speech quality is of particular interest for VoIP. Due to the flexible IP transmission, such codecs can be easily introduced without the need of changing the network's infrastructure.

Quantization

Quantization can be memoryless or with memory depending upon whether the encoding rules depend on past inputs or outputs [54]. The quantization is a lossy process and the different sequence between the original speech sequence and the representation is referred as quantization noise. The reduced quantization noise means improved quality. The quantization methods can be classified as scalar quantization (SQ) and vector quantization (VQ). In SQ, the inputs of a quantizer are scalar, and each codeword represent a single sample. In VQ, speech sequence is encoded in block or vector form. Given a bit-rate, VQ results in a lower distortion than SQ. With the recent efficient methods for encoding high-dimensionality blocks, VQ can achieve high-quality speech coding at low rates [54].

Uniform pulse code modulation (PCM) is the simplest SQ. It is a memoryless process and quantizes the amplitudes of sampled speech signals by rounding off each sample to a level between maximum (X_{max}) and minimum (X_{min}) of the amplitudes. The step size δ, i.e., the difference between adjacent levels, is constant. Uniform PCM is simple and can be used for any bandlimited signal. It is widely used in speech coding to quantize the speech samples or the coefficients of speech models. The quantization error power of uniform PCM is largely independent of signal power, that is, high-power signals are quantized with the same resolution as low-power signals [58].

Speech coding techniques

Speech coding techniques can be roughly classified into three categories: waveform coders, parametric coders (vocoders) and hybrid coders. Waveform coders are high-bit-rates coders and with high quality, i.e., G.711 PCM (64 kbits/s). Parametric coders work on low bit rate with less perceived quality, e.g., LPC vocoder (2.0–4.8 kbits/s). And hybrid coders achieve good quality at medium bit rates, e.g., G.729 (8 kbits/s).

Waveform coding focuses on the speech waveform instead of the underlying speech model. Without using any assumption about the speech signals, waveform coders can be used for both speech and non-speech signals and keep high perceived quality. However, these algorithms also result in a higher data rate than the vocoders based on specified speech models. There are two types of waveform coding: time-domain waveform coding, e.g., companded PCM, and spectral coding (waveform coding in frequency domain), e.g., sub-band coding (SBC).

Companded PCM compresses a speech signal to get uniform distribution at the encoder, and inverses expand at the decoder. The compression reduces the dynamic range of a speech signal. This can increase the signal-to-noise ratio (SNR) in analog systems, and in the digital domain, it can reduce the quantization error (hence increasing signal to quantization noise ratio) [59].

A commonly used companded PCM is log-PCM, e.g., μ-law PCM and A-law PCM, which are standardized in G.711 [60]. It uses 7–8 bits per sample for a non-uniform PCM with 34 dB SNR to achieve the performance of a 12-bit uniform PCM, which is widely accepted as "toll quality". The μ-law PCM companding function is

$$\hat{x}(m) = X_{max} \frac{\log(1 + \mu|\frac{x(m)}{X_{max}}|)}{\log(1 + \mu)}, \tag{2.1}$$

where μ is compression parameter (255 in the North American and Japanese standards [59]).

Parametric coders describe a speech signal based on speech production procedure: forcing air first through an elastic opening, the vocal cords, and then through the laryngeal, oral, nasal and pharynx passages, and finally through the mouth and the nasal cavity [61]. Then the speech signal is represented by a model controlled by a set of parameters, which are quantized and transmitted to the decoder for speech synthesis. The most successful and commonly used algorithms are linear prediction based techniques.

Linear predictive coding (LPC) vocoders are examples of parametric speech coding. In LPC vocoder, the vocal tract is modelled as a single linear filter, which can be represented as

$$H(z) = \frac{g}{1 - A(z)} = \frac{g}{1 - \sum_{k=1}^{p} a_k}, \tag{2.2}$$

where g is the gain. Most of the poles of $H(z)$ is related to formant frequencies [58]. The coefficients of $H(z)$ are obtained by Levinson-Durbin algorithm from each speech segment (see [62] for details).

Thus, the speech sequence is modelled as a excitation sequence passing through the filter $H(z)$ in LPC. The excitation sequence can be impulse for voiced speech and white noise for unvoiced speech. Therefore, the parameters describing characteristics of a speech segment include g, voicing decision, pitch period in the case of a voice segment and the parameters of

$H(z)$. Once obtained by analyzing the windowed speech, these parameters are quantized and transmitted to the decoder. The decoder generates the excitation sequence using voicing decision and pitch period: a quasi-periodic signal with discrete pulse (1–8 per pitch period) for voiced frames and pseudo-random noise sequence for unvoice frames, or some combination of the two. The speech is synthesized by passing the excitation through the reconstructed $\hat{H}(z)$ using reflection coefficients.

Hybrid coders incorporate aspects of both waveform coders and parametric coders. As in LPC vocoders, the vocal tract is modelled by a linear prediction filter. The excitation is chosen from a codebook with a variety of excitation signals such that the reconstructed speech is as close as possible to the original speech in time domain. The closeness is measured by a perceptually weighted error signal. Many hybrid coders exist in literature, the most widely used in VoIP with high quality is CELP (codebook excited linear prediction) coder.

The main components of a CELP coder include the LPC analysis, the excitation codebook, and the perceptual weighting filter [61]. LP filter coefficients are encoded and transmitted to the decoder. CELP coders use a so-called stochastic codebook which in early works was generated by a Gaussian process. The codebook is divided into two parts: an "adaptive" codebook representing the pitch periodicity and a "fixed" codebook for unpredictable innovations, which are used to get an excitation signal by adding a pitch signal. A gain factor scales the excitation vectors before they are filtered by the long- and short-synthesis filters. For each entry in the codebook, the synthesized speech is compared with the original speech, and the best match entry which minimizes the perceptually weighted MSE is selected. The LP parameters, including linear spectral frequencies (LSF) coefficients, indices of adaptive and fixed codebooks and gains, are quantized using VQ and transmitted to the decoder. LSF is an alternative representation of LPC [63]. It can be converted to LPC and vice versa. The most attractive parts of using LSF in speech coding reside in: the minimum-phase property of the associated synthesis filter is preserved after quantization, as long as the interlacing condition is satisfied; the values of the LSFs directly control the property of the signal in the frequency domain, and changes of one parameter have a local effect on the spectrum [63].

The algebraic CELP (ACELP) generates an excitation by choosing one vector from an adaptive codebook and one vector from a fixed codebook. The entries in fixed codebook of ACELP are generated from an algebraic codebook a_k and a shaping matrix F which shapes

the excitation vectors in the frequency domain so that their energies are concentrated in the important frequency bands [64]. The advantage of this structure is that the codebook search is decoupled from the codebook properties [64]. The algebraic codebook is a set of interleaved permutation codes with few nonzero elements, and can be very efficiently searched without running into storage or complexity problems. Some VoIP codecs use ACELP, for example, G.729, G.723.1, AMR (adaptive multi-rate codec), GMS-EFR, etc.

2.2.3 Speech codecs in VoIP

In VoIP, different codecs are applied to digitalize the analog speech received from end-points. These codecs are intergraded in IP phones or gateways. Table. 2.2 shows the most commonly used codecs in VoIP.

Table 2.2 Codecs in VoIP

Codec	Algorithm	Bit Rates (Kbits/s)	Frame Length (ms)	look ahead (ms)	licensing
G.711	A-law/μ-law PCM	64	10	no	free
G.729	CS-ACELP	8	10	5	licensed
G.723.1	MP-MLQ	5.3	30	7.5	licensed
	ACELP	6.3	30		
AMR	MR-ACELP	4.75–12.2	20	5	licensed
GSM-EFR	ACELP	12.2	20	no	licensed
iLBC	Block Independent	15.2	20	5	free
	LPC	13.33	30	10	

G.711 is an ITU-T standard developed for digital telephony. It uses log-PCM to compress a 16-bit sample to 8 bits, and the resulting bit rate is 64 kbit/s. In G.711, A-law is being used in Europe and in international telephone links, and μ-law is used in the US and Japan. It is simple to implement with low computation complexity and processing delay. G.711 also achieves the highest perceived quality among all speech codecs due to the use of PCM. Therefore, G.711 is also used as a reference when quality issues of other speech coders are explored. In this work, we use G.711/μ-law in our experiments (Chapter 5).

G.729 is also an ITU-T standard and is mostly used in VoIP due to its low bandwidth requirements, low computation complexity and good perceived quality. Standard G.729 operates at 8 kbit/s with 10 ms per frame and also provides the rates of 6.4 kbit/s (Annex

D, F, H, I, C+) and 11.8 kbit/s (Annex E, G, H, I, C+) for the applications with different requirements of speech quality. The algorithm used in G.729 is CS-ACELP (conjugate structure algebraic codebook excited linear prediction), which uses a two-stage codebook structure for LSP coefficients. As other ACELP-based codecs, G.729 reconstructs speech sequences by passing an excitation sequence through a synthesis filter.

G.723.1, an ITU-T standard, supports the compression rates of 5.3 kbit/s and 6.3 kbit/s with a 30-ms frame. Both coding algorithms in G.723.1, providing different rates, are based on CELP (introduced in Section 2.2.2). The difference between these two algorithms relies on the excitation codebook. For the high bit rate (6.3 kbit/s), Multi-Pulse Maximum Likelihood Quantization (MP-MLQ) excitation is used, and for the low bit rate (5.3 kbit/s), an ACELP (Section 2.2.2) is used.

GSM-EFR (GSM enhanced full rate) is a codec developed by the ETSI (European Telecommunications Standards Institute) for GSM networks. It is an enhanced version of GSM-Full Rate (GSM-FR), and provides wirelike quality in noise free or any background noise conditions. GSM-EFR is a very robust speech coder with a bit rate of 12.2 kbits/s for a 20 ms speech frame. ACELP (see Section 2.2.2) is used as the coding algorithm.

AMR (Adaptive Multi-Rate) was standardized by 3GPP. It consists of 8 different bit rates for link adaption in Global System for Mobile Communications (GSM) network: 4.75, 5.15, 5.9, 6.7, 7.4, 7.95, 10.2 and 12.2 kbits/s. In AMR, there are 14 codec modes: 8 for a full rate (FR) channel and 6 for half rate (HR) channel, and the mode switch can occur at any time based on an optimized link adaption, which chooses the best codec mode for current network conditions. AMR was originally developed for radio channels, and when radio conditions get worse, source coding is reduced and channel coding is increased. This idea can also be adopted by VoIP applications, that is, use a full rate codec for good network conditions and a half rate one for worse network conditions, e.g., a congestion network. The frame size is 20 ms (160 samples).

iLBC (internet Low Bit Rate Codec) was developed by Global IP Solutions and is defined in RFC3951 [65]. iLBC has two variants: 15.2 kbit/s for a 20 ms frame and 13.33 kbit/s for a 30 ms. iLBC uses a block-independent linear-predictive coding (LPC) algorithm. Each frame is divided into consecutive sub-blocks with the length of 40 samples. For each sub-block, a set of LP coefficients and the residual signal are obtained by LP analysis. The two consecutive sub-blocks of the residual is divided into two parts: dominant part and remaining part. The dominant part is the segment with 57/58 (20 ms/30 ms)

samples, which has the maximal weighted energy. A scalar quantizer (i.e., DPCM) is applied to the dominant part and the location of the start state. A dynamic codebook is used to code the remaining part of the residual.

2.2.4 Packet loss concealment

In VoIP, packets are not guaranteed to reach the receiver on time to be decoded. All the codecs discussed in Section 2.2.3 have built-in packet loss concealment (PLC) algorithms for handling frame erasure.

According to G.711 Appendix I, the packet loss concealment(PLC) algorithm is based on overlap add (OLA). A segment of previous decoded speech (48.7 ms/390 samples) is stored in a circular history buffer. For the first 10 ms of the erasure, the last 20 ms of speech in the history buffer is used as the reference signal, and a 20 ms window slides back at taps from 5 ms (40 samples) to 15 ms (120 samples). The normalized cross-correlation of the reference signal and the windowed signal is calculated, and then the peak is used to detect the pitch period. With the estimated pitch period, the most recent 1.25 pitch periods segment before the erasure is selected to generate the substitution of the lost frame. To insure a smooth transition, the overlap add (OLA) is performed on the selected segment using a triangular window, which is 0.25 pitch period in length. Then the concealed frame is generated by repeating the OLAed segment as many times as needed. For the first 10 ms erased frame, no attenuation is applied. From the second 10 ms, the concealed signal is linearly attenuated with a ramp at the rate of 20% per 10 ms. After 60 ms, the concealed signal is zero.

In G.729, if frame erasure occurs, the linear prediction (LP) parameters of the last good frame are used to form the synthesis filter, and the excitation sequence is generated by the periodicity classification of the previous frame. If the last frame is classified as period (active voiced), the pitch information of the last frame is repeated. Otherwise, the fixed codebook vector is chosen at random.

Both algorithms of G.723.1 have a built-in PLC algorithm. Similar to G.729 PLC, the LP coefficients and residual sequence are interpolated independently for reconstructing an erased frame. For residual interpolation, the last previous good frame is checked with a voiced/unvoiced classifier, which is based on a cross-correlation maximization function. If the previous frame is classified as unvoiced, the excitation is generated using a uniform random number generator. For voiced-classified frames, periodic excitation is reconstructed

with the period provided by the classifier. If the frame erasure state continues for the next two frames, the regenerated vector is attenuated by an additional 2.5 dB for each frame. After 3 interpolated frames, the output is muted completely [66].

AMR and GSM-EFR have the same built-in PLC algorithm. AMR uses GSM-EFR for the mode running at 12.2 kbit/s. For an erased SID frame, the last valid SID frame information is used to generate the substitution. Similar to other ACELP based codecs, the lost speech frames are generated by passing the excitation sequence through a synthesis filter which is based on previous good frames. Random fixed codebook indices are used for generating excitation sequence. The LSF coefficients of the synthesis filter are obtained from the past LSF values by Equation (2.3)

$$\omega_{q1}(i) = \omega_{q2}(i) = \alpha\omega_q(i-1) + (1-\alpha)\bar{\omega}(i), i = 0...9. \tag{2.3}$$

where

- $\alpha = 0.95$
- ω_{q1} and ω_{q2} are two set of LSF-vectors for erased frame
- $\omega_q(i-1)$ is ω_{q2} from the previous frame
- $\bar{\omega}$ is the average LSF-vector

Note that two sets of LSF vectors are available only in AMR 12.2 mode (GSM-EFR).

In iLBC, the encoded frames are de-correlated, to some extent, by using the blocked-based coding of the residual signal. Therefore, unlike other low-bit codecs, such as G.729, G.723, AMR, the PLC algorithm in iLBC does not exploit dependencies between adjacent frames. This avoids the error propagation and hence makes the codec robust against packet loss. The built-in PLC algorithm operates on LP filters, pitch, and excitation signals. During good frames, the decoder stores the information of the current block, LP filter coefficients for each sub-block, and the entire decoded excitation signal. If a block is erased, the excitation signal is generated from the excitation in the previous block based on a pitch-synchronous repetition, and the LP filter uses the last LP filter of the previous block. A correlation analysis is performed on the previous block's excitation signal in order to detect the amount of pitch periodicity and pitch value [65]. Then the concealed block can be obtained by passing the newly constructed excitation signal through the LP filter.

2.2.5 VAD/DTX

Human speech consists of one or more talkspurts and a silence period between two consecutive talkspurts. To reduce the overall bit rate, discontinuous transmission (DTX) is widely used for silence compression in speech codecs, e.g., the codecs in Table 2.2. The idea is to provide continuous and smooth information about the non-active voice periods, while keeping a low average bit rate [41]. In this subsection, we use G.729 annex B [41], which is used in both G.711 and G.729, as an example to show how DTX works. Voice activity detection (VAD) is firstly applied to differentiate between active voice frames and non-active voice frames. For non-active voice frames, the ambient noise is described by the comfort noise generation (CNG) algorithm. Then the noise information is encoded into a silence insertion descriptor (SID) frame which is packed into a CN payload of a RTP packet (RFC 3389). A DTX algorithm determines when an SID frame is transmitted: periodically or only when there is a significant change in the background noise characteristics. When an SID packet arrives at receiver's side, the CNG algorithm updates the noise generation model using the information in the SID packet, and then generates comfort noise using the model.

Besides that, G.729 annex B [41] also defines "hangover" frames, which generally occur at the end of a talkspurt. Even thought they are classified as non-active voice frames by the VAD algorithm, "hangover" frames are encoded as active voice frames. In this way, an active voice decision can be smoothly switched to a non-active voice decision, and hence avoid voice clipping. In this work, we take advantage of these frames to design our playout buffering algorithms (Chapter 3 and Chapter 4).

2.3 Network components impacting on perceived quality

In VoIP, major factors associated with perceived quality are delay, jitter, and missing packets. All these factors stem from the "best effort" model used in IP networks, that is, the network tries its best to deliver packets to their destinations but makes no guarantee that every packet will arrive or will arrive on time. Unlike traditional telephony, all these parameters are time-varying. Therefore, high-quality VoIP should benefit from efficient packet loss concealment and jitter buffer/delay management. The inter-relationship of these impact is shown in Figure 2.8.

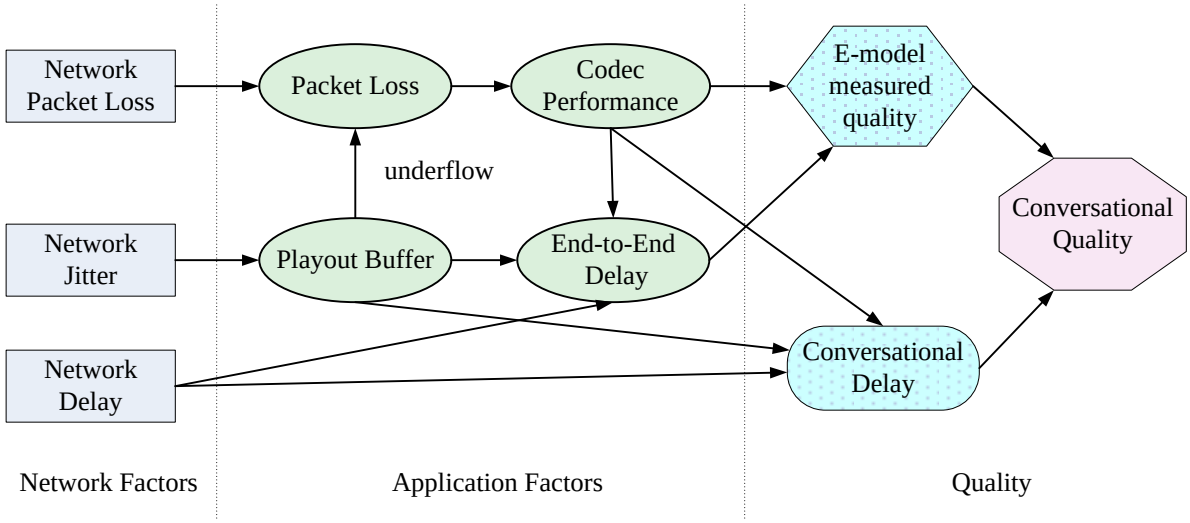


Figure 2.8 Inter-relationship of network impact factors on conversational quality.

2.3.1 Delay jitter

In packet-based networks, the transmission delay is time-varying. Delay variations are called delay jitter. There are many reasons for delay jitter. For example, when two consecutive packets arrive at a router, it is normal for them to be sent to different routers/switches according to the route selection algorithm of the router, and hence they arrive at their destination with two different routes. The network equipments on these two routes are generally different in type and in number. Therefore, the accumulated queuing and processing delays on the two routes are different, and the end-to-end delays of these two consecutive packets are different accordingly. Delay jitter presented in packet networks makes it complicated for a decoder to produce continuous speech at the receiver's side, because a decoder needs to have speech data available at regular time intervals [12].

To compensate for delay jitter, a playout buffer is introduced at the receiver side. There are two groups of approaches for playout buffering: fixed schemes and adaptive schemes. If a buffer is designed with a fixed size, all voice packets are kept to a constant end-to-end delay in a session [37]. In an adaptive scheme, a playout buffer is designed based on the monitored network behavior. Therefore, adaptive schemes can catch the temporal variability of networks by adjusting their playout buffers and hence outperform fixed approaches in most cases.

2.3.2 End-to-end delay & conversational delay

End-to-end delay in VoIP is an important performance parameter. This delay includes *processing delay* associated with speech coding and packetization, *transmission delay* between the sender and receiver, and *buffering delay* caused by a playout buffer at the receiver side. Note that the processing delay also includes delays introduced by other DSP features, e.g., echo cancelation, noise reduction, and packet loss concealment [67, Chapter 1].

Mathematically, the end-to-end delay d can be expressed as

$$d = d_{proc} + d_{net} + d_{buff}. \tag{2.4}$$

where

- d_{proc} is processing delay
- d_{net} is transmission delay
- d_{buff} is buffering delay

In some cases, the delay has a "spike" nature, that is, the sudden onset of a large increase. Although subsequent packets usually experience declining network delays, the delay values are large [30]. The spike ends when network delays return to average values [30]. The packets with spike delays are not lost and arrive at the receiver side with long delays. If the delay values exceed the size of the playout buffer, the speech information carried in packets cannot be used for speech reconstruction, resulting in burst loss which degrades perceived quality.

For conversational VoIP, conversational delay plays an important role on perceived quality. Conversational interactivity is broken up when a conversational delay is too long. In [40], conversational delay is defined by: the time interval between when User 1 stops speaking and when User 1 hears the User 2's response. Figure 2.9 visually represents the definition

Mathematically, we formulate it as:

$$D_c = t_{play}(i_{user2}) - t_{send}(j_{user1}) + d_{proc}, \tag{2.5}$$

where

- D_c is the conversational delay
- d_{proc} is processing delay

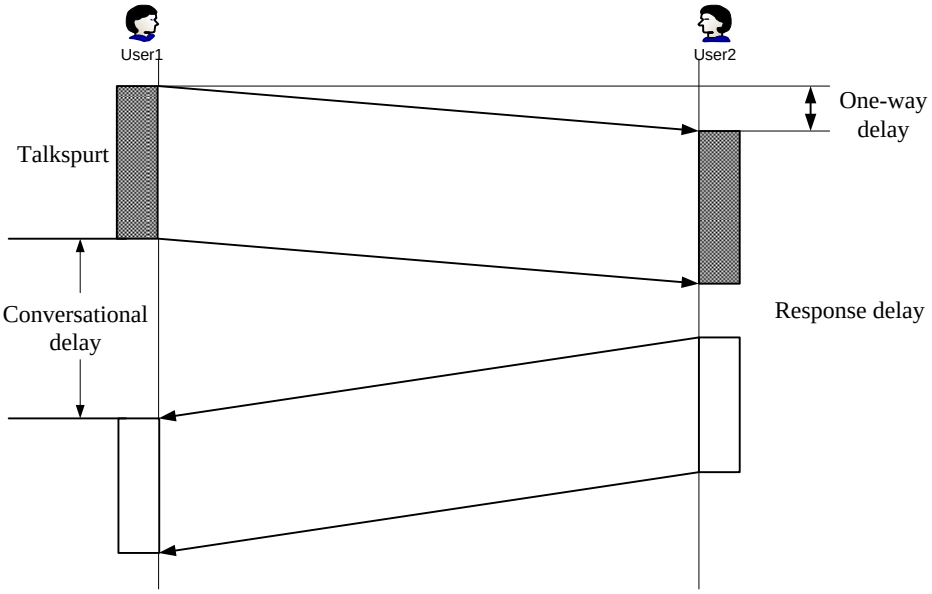


Figure 2.9 Conversational Delay.

- $t_{play}(\cdot)$ is the time scheduled to play
- $t_{send}(\cdot)$ is the sending time
- i_{user2} is User 2's first packet of the first talk-spurt
- j_{user1} is User 1's last packet of the last talk-spurt

Obviously, to calculate D_c, the key is how to detect the start and the end of a talk-spurt. The first packet of a talk-spurt can be recognized by the M field of RTP (Real-Time Transport Protocol) header, which is 1 for the first speech packet after a silence period [43]. The problem is how to define the end of the talkspurt. From our observation on listening tests, we noticed that people tend to start replying when the "hangover" packets are received. Based on the definition of "hangover" in [41], it is reasonable to define the end of talkspurt when the first "hangover" packet is perceived. In our system described in Section 2.1.2, the latest version of VAD/DTX from the G.729 [41] is used on a receiver's side to detect the "hangover" packet to calculate D_c. We will present our algorithms for detecting the start and the end of a talkspurt in Section 3.4.

2.3.3 Missing packets

The missing packets include both network losses (packets that never arrive or are dropped due to errors) and late packets (buffer underflow). Network losses can be caused by: link failure, heavy network load which causes packets in the queues in routers to be dropped, configuration errors, and collisions, for example, in Ethernet or wireless networks. Network losses can be alleviated by using a retransmission scheme, which is generally provided by a transport layer protocol, e.g., Transmission Control Protocol (TCP). The scheme retransmits the packets which are detected to be lost. However, retransmission schemes cost time to wait for acknowledgement from the destination, resulting large end-to-end delay. As a result, they are not suitable for real-time applications as VoIP. Late packets occur when packets arrive at the receiver after they are scheduled to be played out, and they are of concern for the design of playout buffer at the receiver side.

In many speech codecs, packet loss concealment (PLC) algorithms (see Section 2.2.4) are used to fill in the missing speech frames. However, PLC techniques are not very effective at concealing packet losses when packets are lost successively in a long burst. Moreover, in some cases, to save transmission bandwidth, multiple speech frames are packetized into one packet, and one single loss of a packet may result in a burst loss of speech frames [12]. If packet losses exhibit burstness, the degradation on perceived quality is more than that caused by the same number of isolated losses. In Chapter 4, we will discuss the methodologies to reduce packet loss by redundancy information.

It has been proved that losses in real voice trace over IP networks is not random [37]. Statistic models can be applied to describe the loss occupancies. In [68], it is shown that loss process is better described by a two-state Gilbert model than a Bernoulli one. In the simple Gilbert model (Figure 2.10), p is the transition possibility between a "0" (received) state to an "1" (lost) state, and q is the transition possibility between a "1" (lost) state to an "0" (received) state.

The probability of $i-th$ state S_i being "0" can be expressed as

$$Pr(S_i = 0) = qPr(S_{i-1} = 1) + (1-p)Pr(S_{i-1} = 0), \tag{2.6}$$

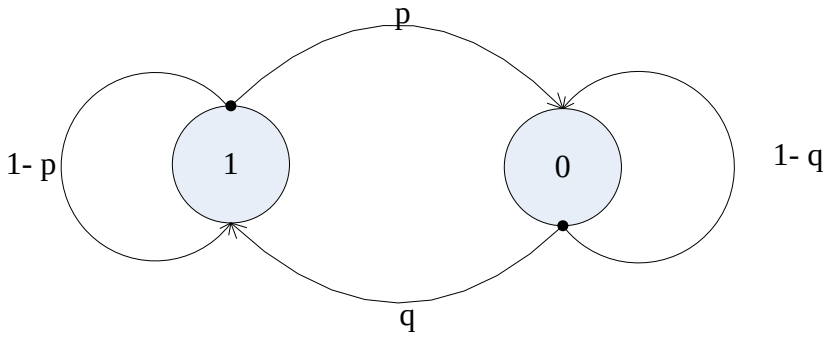


Figure 2.10 two-state Gilbert model.

if stationary

$$\begin{aligned} Pr(S_i = 0) = Pr(S_{i-1} = 0) = \rho, \\ Pr(S_i = 1) = Pr(S_{i-1} = 1) = 1 - \rho, \end{aligned} \tag{2.7}$$

with ρ is the packet loss rate.

From Equation (2.6) and Equation (2.7), ρ can be calculated as

$$\begin{aligned} \rho &= q(1-\rho) + (1-p)\rho \\ &= q - q\rho + (1-p)\rho \\ &\Longrightarrow \\ \rho &= \frac{q}{p+q}. \end{aligned} \tag{2.8}$$

When packet loss is bursty, the probability of burst loss ($Pr[BL]$) can be obtained by

$$Pr(BL) = q(1-q)^{BL-1}, \tag{2.9}$$

where BL is the length of burst losses.

Since $\sum_{j=0}^{\infty} A^{j-1} j = \frac{1}{(1-A)^2}$, then the expected burst loss length $E[BL]$ can be ex-

pressed as

$$E[BL] = \sum_{BL=0}^{\infty} (BL \cdot q(1-q)^{BL-1}) \\ = q \cdot \frac{1}{q^2} = \frac{1}{q}. \tag{2.10}$$

2.4 Speech quality measurements

In VoIP applications, perceived quality is important. Mean Opinion Score (MOS) provides a direct link to the voice quality as perceived by end users. MOS values can be obtained by subjective testing or objective models. Subjective testing is closer to the goal of speech quality testing, i.e., get a measure of the perceived quality by humans [12]. The problem of subjective testing lies in its time-consuming, costly, lack of repeatability and impractical for on-line applications. Alternatively, objective methods are simple, easy to compute and are more suitable for network planning and on-line voice quality monitoring. However, they are sensitive to processing effects and other impairments [12], e.g., delay, interactivity, echo, etc.

Objective measurements can be intrusive or non-intrusive. The main difference between this two classes is whether a reference signal is used. Intrusive method are more accurate, but not suitable for real-time application because a reference speech is required. Perceptual Evaluation of Speech Quality (PESQ) is the most popular intrusive objective method. Non-intrusive methods measure perceived speech quality without reference data, and is appropriate for network monitoring. The E-Model is the most widely used parametric non-intrusive measure, which assesses the effect on quality of network impairments, e.g., packet loss, jitter, delay.

2.4.1 Mean Opinion Score

In ITU-T P.10 [69], MOS is defined as:

> *The mean of opinion scores, i.e., of the values on a predefined scale that subjects assign to their opinion of the performance of the telephone transmission system used either for conversation or for listening to spoken material.*

Opinion scores are obtained from subjective testing. Absolute Category Rating (ACR) is one of the most popular subjective test methodologies. In an ACR test, a pool of listeners rate a series of audio files using five-point category-judgement scales recommended by CCITT:

Excellent = 5 Good = 4 Fair = 3 Poor = 2 Bad = 1

Apart from ACR, Degradation Category Rating (DCR) and Comparison Category Rating (CCR) are also used as subjective tests methodologies. DCR is associated to a DMOS score which presents the level of degradation for the degraded speech files. The CCR test produces a CMOS score to compare pairs of speech files.

Except for subjective opinions, *MOS* is also used for scores that are obtained from objective model or network planning models [70]. To distinguish the area of applications, ITU-T introduced sub-categories: **LQ** for Listening Quality, **CQ** for Conversational Quality, **S** for Subjective, **O** for Objective, and **E** for Estimated (the score is calculated using a network planning model, e.g., E-model) [70]. These are shown in Table 2.3

Table 2.3 *MOS* Terms for Different Applications

	Listening-only Quality	Conversational Quality
Subjective	MOS-LQS	MOS-CQS
Objective	MOS-LQO	MOS-CQO
Estimated	MOS-LQE	MOS-CQE

2.4.2 Perceptual evaluation of speech quality

PESQ [1]is standardized by ITU-T P.862 [71], which is an objective method to predict perceived quality. It only measures the effects of one-way speech distortion and noise on perceived quality, and does not include the impairments due to loudness loss, delay, sidetone, echo, interaction [71]. PESQ can identify constant delay offset and variable delay jitter [72]. Constant delays are not considered when calculating MOS value, whereas delay jitters change the rating of the speech quality [72]. It does not measure conversation factors, such as conversational delay. In this work, the conversational perceived quality of different playout buffering algorithms is evaluated by MOS calculated by PESQ and conversational

delay calculated by Equation (2.5).

In P.862, PESQ scores are obtained by following steps:

1. apply time alignment algorithm on degraded signals
2. divide original and aligned degraded signals into short overlapping blocks of samples
3. calculate Fourier Transform coefficients for each block and compare the sets of these coefficients
4. calculate a PESQ score

Note that PESQ scores are not an exact mapping for subjective MOS. The latest version of PESQ-LQ score is closely aligned with MOS-LQ (listening quality MOS).

2.4.3 E-Model

The E-Model is a computational model provided by ITU-T (see [1]) which assesses the combined effects of variations in several parameters. The parameters link to metrics: speech coding, delays and loss, loss-concealment algorithms, packet and codec frame size, and frame erasure distribution, etc. Figure 2.11 shows the transmission parameters used as an input to the computation model. Appendix C lists the Default values and permitted ranges for the parameters.

With the assumption of additivity property among these impairments, all the aspects impacting on perceived quality can be expressed by summing up

$$R = 100 - I_s - I_d - I_{e,eff} + A \tag{2.11}$$

with

I_s the distortions introduced by the circuit-switched part of the transmission path, with the default value of 6.8

I_d delay impairment, which is related to mouth-to-ear (end-to-end) delays

$I_{e,eff}$ the impairment associated with signal distortion, caused by low-bit-rate codec and packet losses

1. The PESQ source code used in this work is based on Recommendation P.862 (2001) Amendment 2 (11/05) with reference software. An updated version P.862 (2001) Corrigendum 1 (10/07) was later published without reference software.

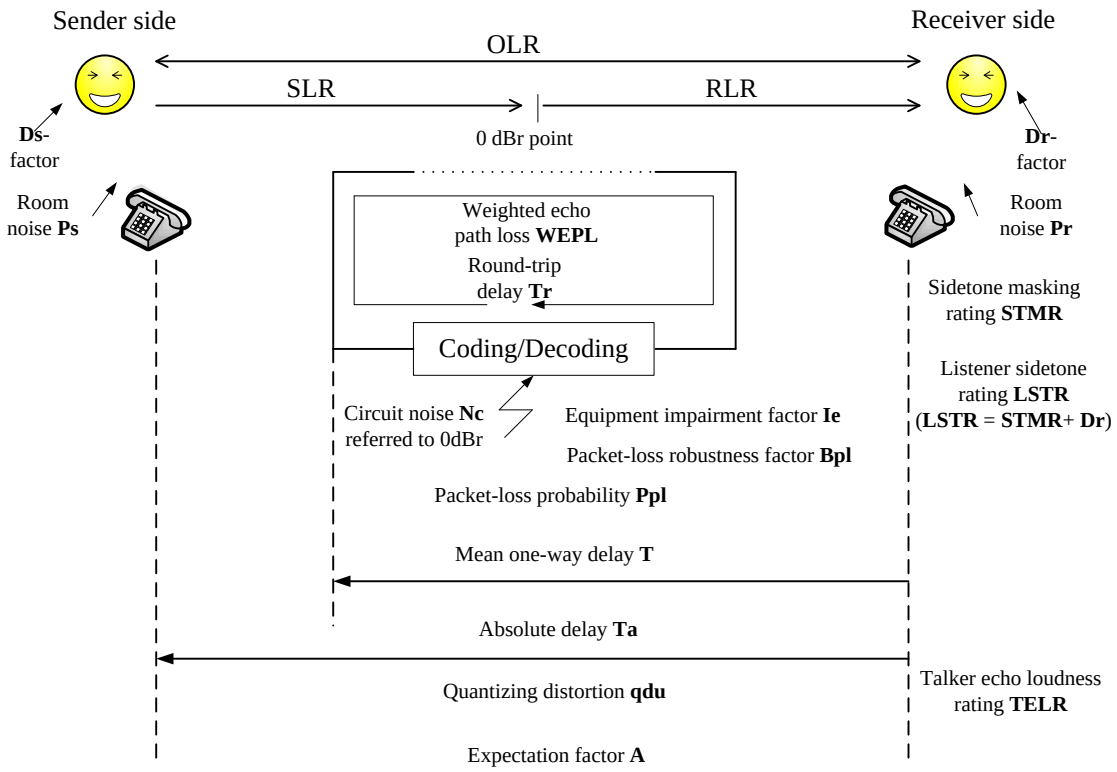


Figure 2.11 G.107 – Reference connection of the E-Model [1]

A the advantage factor which considers the end-users' acceptance of distortion (some provisional values are given in Appendix B)

According to [1], the R factor can be simplified as

$$R = 93.2 - I_e - I_d, \tag{2.12}$$

where I_d is the delay impairment factor, and I_e is the equipment impairment factor.

I_d can be derived by a simplified fitting process from [5],

$$I_d = 0.024d + 0.11(d - 177.3)\, H(d - 177.3), \tag{2.13}$$

where

$$H(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0. \end{cases}$$

d is an end-to-end delay in ms.

The equipment impairment factor is codec dependent. For G.711 with PLC, it can be approximated as [38]

$$I_e = I_{ec} + I_\rho, \tag{2.14}$$

where I_{ec} is the impairment caused by encoder and ρ is the packet loss in percentage, including network loss and the loss caused by jitter buffer.

According to ITU-T G.107 Annex B [1], MOS score can be obtained from the R-factor by

$$MOS = \begin{cases} 1 & R < 0 \\ 1 + 0.035R + R(R-60)(100-R) \times 7 \times 10^{-6} & 0 < R < 100 \\ 4.5 & R > 100 \end{cases} \tag{2.15}$$

A commonly accepted mapping among R factor, MOS and quality of voice rating is reported in [73], which is shown in Table 2.4. Figure 2.12 shows the relationship among R factor, MOS and quality of voice rating.

Table 2.4 R factor, MOS, and Quality of Voice Rating

R factor	MOS	Quality of Voice Rating
$90 < R \leq 100$	4.34 – 4.50	Best
$80 < R \leq 90$	4.03 – 4.34	High
$70 < R \leq 80$	3.60 – 4.03	Medium
$60 < R \leq 70$	3.10 – 3.60	Low
$50 < R \leq 60$	2.58 – 3.10	Poor

The main drawbacks of the E-Model are as following:

– the overall additivity property of the model is applicable only to a certain extent [37].
– many assumptions on the configuration of the network and service have to be introduced to make the model applicable in a playout buffering context [37].
– the E-model is only applicable to a limited number of codecs and network conditions [5].
– the E-model requires subjective tests to derive model parameters, which are time-consuming and often impractical. It is also inevitable that discontinuities exist in subjective results because only a limited range of scenarios can be tested for [5].

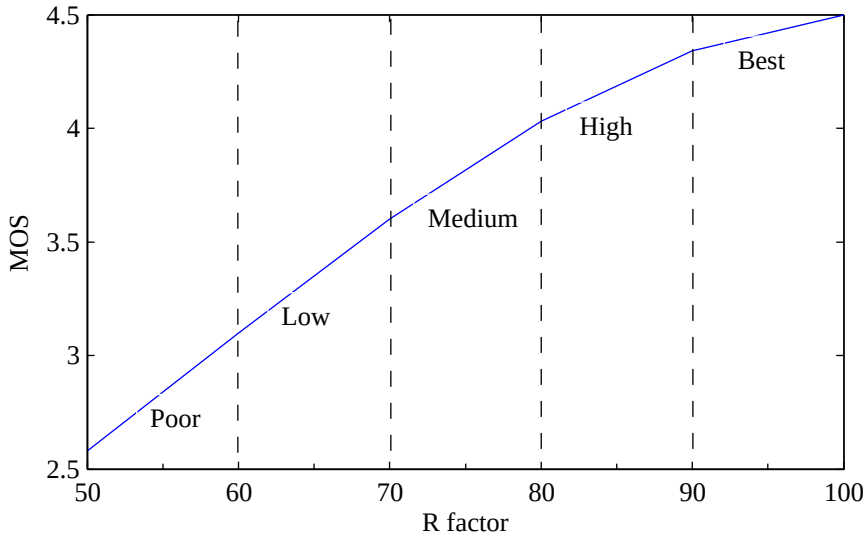


Figure 2.12 *R* factor, *MOS* and quality of voice rating

2.5 Time scale modification

To get continuous playout speech, the depth of a playout buffer is changed by scaling decoded speech. The scaling (stretch/compress) of the speech packets is realized by time scale modification (TSM). The goal of TSM is to change the rate of speech with maintaining the perceived naturalness, i.e., keeping the same pitch and timbre. The basic idea of TSM algorithms is to insert/drop one or multiple speech frame with one pitch duration according to the scaling rate needed. TSM can be formulated as a $1:1$ mapping between the original n and the modified time-scale n'

$$n \rightarrow n' = \tau(n), \tag{2.16}$$

where $\tau(\cdot)$ is called as a time-scaling/time-warping function. Equation (2.16) specifies that the sound occurring at time n in the original signal should be played at n' in the time-scaled signal [37].

There are many high-quality TSM algorithms proposed in the past three decades, for example, time-domain harmonic scaling (TDHS) in [74], harmonic plus noise model (HNM) algorithms (e.g., [75]), synchronized overlap-and-add (SOLA) algorithms (e.g., [76]) etc. As other algorithms in SOLA family, waveform similarity overlap-and-add (WSOLA) algorithms work in time domain and achieve high quality with low computational cost. Thus

WSOLA algorithms are suitable for real-time applications. The basic idea is to decompose the speech into overlapping segments of equal length, which are then realigned and superimposed to form the output with equal and fixed overlap [26]. The realignment leads to modified output length [26]. In WSOLA algorithms, a time-shifting window (controlled by tolerance Δ_k) is applied on the original speech signal, and select the position of the best segment by maximizing a similarity measurement, and then overlap-add the selected segment to the previous scaled segment. The tolerance Δ_k ensures continuity at segment joins and phase continuity of the original speech.

The conventional WSOLA algorithms, e.g., [77], [78], a delay of 2–3 packet times is introduced in expanding the packets [26]. It is not feasible for a playout buffering design which aims to cut down delay. In this work, the playout buffer size is changed gradually. For the first several packets in one talk-spurt, the system can not provide the following packets for using WSOLA. Hence the realignment of the similar segment might break the continuity between blocks, resulting in noisy reconstruction. In [26], Liang et al. modified the WSOLA algorithm to work on only one packet, and the algorithm is named packet-based WSOLA (PWSOLA). The algorithm can be summarized as

- keep a packet in the local memory, it might be used for searching similar segment
- define a template segment of fixed length of input
- find a maximal similar segment to the template segment
- weight the template segment by a falling window and the similar segment by a raising window, and then add them to generate the first part of output
- shift the remaining segment following the similar segment to form the last part of output

The important feature of this algorithm is that the beginning and ending samples of the output are the same as those in input [26]. Therefore, the continuity among speech packets is preserved. Figure 2.13 shows the stretch and compression of the speech packet using PWSOLA.

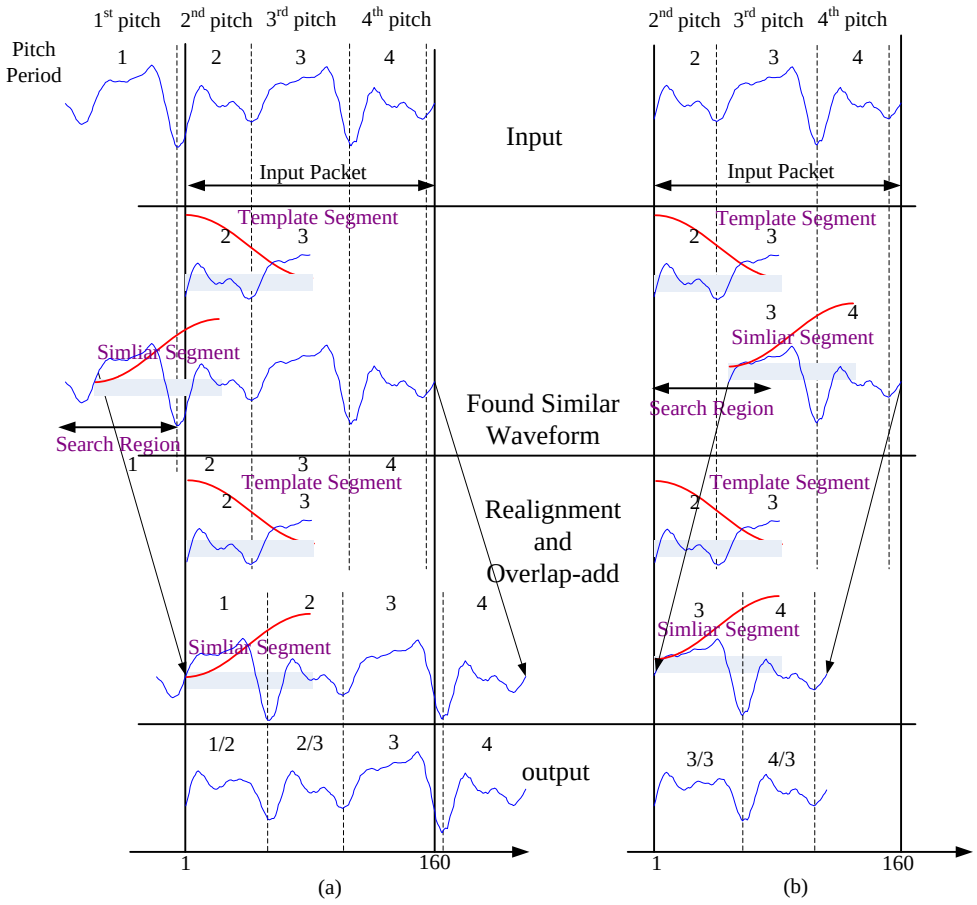


Figure 2.13 PWSOLA: (a) stretch (b) compress.

Chapter 3

Playout Buffering for VoIP

In VoIP, transmission delay (i.e., network delay) for speech packets over IP networks is time-varying due to various queueing and processing delays of network equipments, e.g., routers, switches. The variation in delay, called delay jitter, depends on the nature of the network (e.g., distance, bandwidth, hops, etc), the traffic on the network and the speed of the network facilities. Hence, when packets arrive at the receiver end, the regularity of time intervals between adjacent packets is absent. This irregularity compromises the ability for the decoder to reconstruct continuous speech.

In practice, a playout buffer is introduced at the receiver's side to remove delay jitter, so that the voice information carried on packets can be available at regular intervals for decoding. Figure 3.1 shows how to compensate for delay jitter with playout buffering. With no buffer, there would be gaps in the playout; with playout buffering, all packets in the example are available for continuous playout. The design of playout buffering impacts perceived quality. Inadequate playout buffering leads to the increases in the rate of late packet losses (buffer underflow), which degrade the voice quality dramatically. Most playout buffering designs trade delay against late packets. Such a design would allow a few packets to arrive after these scheduled playout timelines. The packet loss concealment (PLC) scheme in special coding system is used to mitigate the effect of these gaps. Since the time delay of a playout buffer is a major addition to end-to-end delay, to keep conversational interactivity, a playout buffer must be designed to be short but capable of protecting packets against late packet loss.

The main task in a playout buffer design is to estimate the playout deadlines for packets. The late packets, failing to arrive before the designed playout deadlines, are dropped as

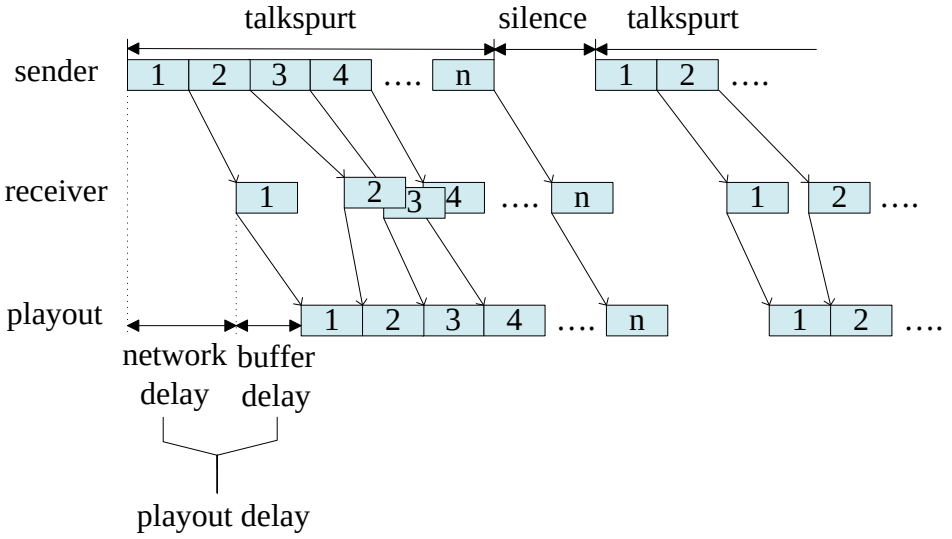


Figure 3.1 playout buffering for VoIP packets

if they had never arrived[1]. The size of the playout buffer can be fixed or adaptive. A fixed approach introduces an additional delay at the receiver side at the beginning of the conversation, and the end-to-end delay is kept constant for all packets. Adaptive approaches adjust the playout buffer size based on the changing network conditions so as to prevent long-size buffers during low-congestion conditions and vice versa.

In this chapter, we elaborate several playout buffering algorithms to represent different categories. We start from the simplest design, fixed playout buffering, in Section 3.1. Then adaptive approaches are discussed in Section 3.2. We firstly introduce two main adaptations – intra-talkspurt and between-talkspurt in Section 3.2.1. In Section 3.2.2, we review the four classic algorithms which were presented in [31]. These algorithms design the playout buffering based on estimated network delay statistics: mean and variation. They are widely used for comparison with other playout buffering algorithms. In Section 3.2.3, we discuss packet loss rate (PLR) based approaches which find optimal playout deadlines for packets to achieve a predetermined PLR. The algorithm developed by Liang et al. in [27] is elaborated as an example of PLR-based approaches. In Section 3.2.4, we illustrate quality-based adaptive buffering by the algorithm developed by Sun and Ifeachor in [5].

1. In practical applications, the late arriving packets are not discarded as if they were lost during transmission. In most speech coding systems, they can be used to synchronize the states of coefficients for decoding the following packets.

In Section 3.3, we introduce a semi-fixed algorithm developed by Lee et al. for conversational VoIP [2] over wireless networks. This algorithm is different from traditional fixed algorithms. Instead of setting a constant end-to-end delay for each packet, it increases the buffer delays for the packets at the beginning of a conversation turn and decreases buffer space at the end of each talkspurt. The aim of this strategy is to reduce the perceived conversational delay. Only a small number of packets use a dynamic-size buffer, and the packets in the middle of a talkspurt are protected by a fixed-size buffer. This novel processing of reducing perceived conversational delay is adopted by our playout buffering algorithm which is presented in Section 3.4.

Our E-Model based adaptive algorithm for conversational VoIP is presented in Section 3.4, which was published in [29]. In our optimization criterion, we consider both maximizing voice quality (estimated by the E-Model R factor) and reducing conversational delay (by a strategy related to the semi-fixed algorithm in [2]).

3.1 Fixed playout buffering

The first studies in playout buffering started in the early 1980s, e.g., [22] and [23], which were proposed for synchronization between sender and receiver. The implementation of a fixed-size playout buffer is straight-forward: add initial playout latency to the first packet received. The end-to-end delay for all packets is constant. To achieve acceptable perceived quality, the size of a playout buffer is typically designed long enough to allow most packets to be available for playout.

3.2 Adaptive playout buffering

Although a fixed buffering method is easy to implement, it can result in unsatisfactory perceived quality. The reason is because no optimal playout deadline can be predetermined for fluctuating network delay. If the buffer is set too large, the total latency may reach a level at which users are annoyed. On the other hand, when the buffer is set too small, packets may be dropped due to their late arrivals. Therefore it is desirable for a playout buffering algorithm to adjust its buffer size according to the temporal variability of the network behavior.

For an adaptive buffering design, the goal is to find a good compromise between buffer

delay and late packet loss [79]. Many approaches have been developed, for example, [31], [34], [35], [27], [29], [5], etc. Playout buffering algorithms in literature can be roughly classified as statistics-based, PLR-based, and quality-based approaches. In this section, we will discuss several algorithms in different categories.

3.2.1 Intra-talkspurt adaptation vs. between-talkspurt adaptation

In most adaptive playout buffering algorithms, the delay information (e.g., network delay, timestamp) of VoIP packets is stored and used to predict the playout time for the following packets. The adaptation can be intra-talkspurt or between-talkspurt, based on when to adjust the playout buffer.

Intra-talkspurt methods adapt playout delay at any instant during the conversation, i.e., during both talkspurts and silence periods. The change in playout buffering is implemented by scaling current decoded packet using Time Scale Modification (TSM) techniques. That is, the speech contained in a packet is stretched/compressed by TSM to fill in the gap caused by different playout deadlines, which guarantees the continuous playout speech. In the case that the distribution of network delays has many delay "spikes", the system can benefit from an intra-talkspurt approach due to its quick reaction to the network variation. However, if the connection is characterized by mild or low delay variations, it is not necessary for a system to update the buffer size so frequently.

Between-talkspurt adaptations adjust playout delays at the beginning of a talkspurt by TSM, e.g., [29] or prolonging silence duration, e.g., [5]. Compared with intra-talkspurt counterparts, these approaches are simpler to implement and computation complexity is lower. However, if the network exhibit burst high delays within a talkspurt, the playout buffering can not provide enough lag time to wait packets to play out and accordingly cause the increment of packet loss rate, which degrades the perceived quality. Therefore, many algorithms use a longer buffer during the "spike" period and packet loss concealment (PLC) algorithms to compensate the missing packets. Another problem of between-talkspurt adaptations can fail to update the playout buffer for some cases. For example, the VoIP call is made in a very noisy background. In this case, no adjustment of playout delays can be made since no silence is detected. We will discuss this problem and propose possible solutions in Section 3.4.1.

3.2.2 Statistics-based approaches

Statistics-based approaches try to set playout deadlines based on the history of past packet delays. In early work of adaptive playout buffering, variance and mean are used to describe fluctuation of network delay, e.g., algorithms in [31]. Hence the playout deadlines, which are calculated from estimated variation and mean, can be adaptive to the changing of network delay. In these approaches, the mean and variance are estimated by using an autoregressive (AR) algorithm. Another group of approaches use adaptive filtering technique to estimate network delay from a window of past packets, e.g., [33] and [30]. Instead of using *estimated mean* to calculate playout deadlines as in AR-based algorithms, adaptive filter based methods use *estimated mean network delay* which is obtained by an adaptive predictor. The network delay of past packets is firstly stored and then passes through a finite-impulse response filter (FIR) to estimate current network delay. The the tap weights of the adaptive filter is adjusted based on mean square error (MSE) between actual network delay and estimated network delay.

In this subsection, we focus on the four algorithms for estimating playout deadline in [31], which are widely accepted as classic methods and used for performance comparison.

For every packet i received, store its network delay d^i_{net} and then calculate playout delay. If packet i is the first packet of a talkspurt, its playout time t^i_p is calculated as

$$\hat{t}^i_p = t^i_s + \hat{d}^i_{av} + \mu\hat{v}^i, \tag{3.1}$$

where

- $\hat{d}^i_{av}$ is the estimated mean of network delay
- $\hat{v}^i$ is the estimated variation of network delay
- t^i_s is the send time
- μ is a constant.

Exponential-Average (Exp-Avg) In [31], the mean delay is estimated based on the RFC793 algorithm [80]. The calculation of mean delay and variation in delays is as follows:

$$\begin{aligned} \hat{d}^i_{av} &= \alpha\hat{d}^{i-1}_{av} + (1-\alpha)d^i_{net}, \\ \hat{v}^i &= \alpha\hat{v}^{i-1} + (1-\alpha)|\hat{d}^i_{av} - d^i_{net}| \end{aligned} \tag{3.2}$$

where

– $d^i_{net}$is the network delay of i-th packet

– α is a weighting factor which determines how much weight is given to the old value. According to [31], it is set to 0.998002.

Fast Exponential-Average This algorithm is a modification to **Exp-Avg**. The idea is that different weighing factors are used for estimating network delay: one for increasing trends in the delay (α) and the other for decreasing trends (β) [31]. The estimation adapts more quickly to delay spikes [31]. The algorithm is described as

$$\hat{d}^i_{av} = \begin{cases} \beta\hat{d}^{i-1}_{av} + (1-\beta)d^i_{net} & \text{if } d^i_n > \hat{d}^i_{av}, \\ \alpha\hat{d}^{i-1}_{av} + (1-\alpha)d^i_{net} & \text{otherwise,} \end{cases} \tag{3.3}$$

where α and β are weighting factors, satisfying $0 < \beta < \alpha < 1$.

Min-delay

$$\hat{d}^i_{av} = \min j \in S_i d^j_{net} \tag{3.4}$$

where S_i is the set of all packets received in previous talkspurts

Spike Detection This algorithm focuses on a delay "spike" which represents a sudden and large change in delays over a sequence number of packets.

$$\hat{d}^i_{av} = \begin{cases} 0.125d^i_{net} + 0.875\hat{d}^i - 1_{av} & \text{mode is "Normal",} \\ \hat{d}^{i-1}_{av} + d^i_{net} - d^{i-1}_{net} & \text{mode is "Spike".} \end{cases} \tag{3.5}$$

The spike detection is illustrated by the pseudocode in Algorithm 1. In this algorithm, all times are measured in bytes, with 20 ms (the voice packetization interval) of time corresponding to 160 bytes [31].

Statistics based approaches are straightforward and simple to implement. However, they do not directly take late packet loss into consideration. They work very well in case of low jitter, but tend to result in high playout delay if jitter is high.

3.2.3 PLR-based approaches

Human's perceptual system can tolerate a small amount of packet missing. Most speech codecs used in VoIP have built-in Packet Loss Concealment (PLC) algorithms to fill in the

Algorithm 1 Spike Detection

d^i_{net} is the network delay of packet i
if $mode == NORMAL$ **then**
  if $abs(d^i_{net} - d^{i-1}_{net}) > abs(\hat{v^i}) * 2 + 800$ **then**
    $var = 0$; %Detected beginning of spike
    $mode = IMPULSE$;
  end if
else
  $var = var/2 + abs((2 * d^i_{net} - d^{i-1}_{net} - d^{i-2}_{net})/8)$;
  if $var \leq 63$ **then**
    $mode = NORMAL$; %end of spike
    $d^{i-2}_{net} = d^{i-1}_{net}$;
    $d^{i-1}_{net} = d^i_{net}$;
    return
  end if
end if
if $mode == NORMAL$ **then**
  $\hat{d}^i_{av} = \hat{d}^{i-1}_{av} + d^i_{net} - d^{i-1}_{net}$;
  $\hat{v}^i = 0.125 * abs(d^i_{net} - \hat{d}^i_{av}) + 0.875 * \hat{v}^{i-1}$;
  $d^{i-2}_{net} = d^{i-1}_{net}$;
  $d^{i-1}_{net} = d^i_{net}$;
  return
end if

absent speech. These facts motivated some algorithms based on a user-specified PLR (Packet Loss Rate), e.g., [34],[35], [27]. In these algorithms, PLR is used as the cost index for designing a buffer size. In this subsection, we cite the adaptive algorithm developed by Liang et al. in [27] as an example of PLR-based approaches. Algorithm 2 is the pseudocode for the algorithm in [26].

Algorithm 2 Liang's PLR-based Algorithm

Receive packet i;
Estimate and set the playout time for the $(i+1)$-th packet, $\hat{t}_{av}^{i+1}$;
Calculate the desired length of packet i
$\hat{L}^i = \hat{t}_p^{i+1} - \hat{t}_p^i$;
if $\hat{L}^i - L_0 >$ expansion threshold **then**
  Scale packet i with target length $min(\hat{L}^i, L_{max})$;
else if $\hat{L}^i - L_0 <$ -compression threshold **then**
  Scale packet i with target length $max(\hat{L}^i, L_{min})$;
else
  Keep packet i without modification;
end if
Output packet i with actual length L^i;
Update the playout time for the $(i+1)$-th packet
$t_p^{i+1} = t_p^i + L^i$;

In Algorithm 2, L_0 is the original speech length in packet i, L_{max} is the maximum target speech length of a packet, which is set as $L_{max} = 2.3L_0$, and L_{min} is the minimum target speech length of a packet, which is set as $L_{min} = 0.3L_0$ [27].

In this algorithm, it is crucial to estimate $\hat{t}_p^{i+1}$, the estimated playout deadline for $(i+1)$-th packet. $\hat{t}_p^{i+1}$ determines how to scale the speech segment represented in i-th packet, and accordingly the actual playout time $(i+1)$-th packet, t_p^{i+1}, based on the length of scaled speech segment. To estimate $\hat{t}_p^{i+1}$, a threshold of the end-to-end delay for the $(i+1)$-th packet, $\hat{d}_{max}^{i+1}$, is firstly estimated based on order statistics of a sliding window of the past w packets.

For i-th packet, the network delay of the past w packets is denoted as $d_n^{i-w+1}, d_n^{i-w+2}, ..., d_n^i$. Its order statistics are $D^1, D^2, ..., D^w$ where

$$D^1 \leq D^2 \leq ... \leq D^w. \tag{3.6}$$

The probability that the network delay $\vec{d_n}$ is no greater than the r-th order statistic D^r is

$$F(D^r) = P(d \leq D^r), r = 1, 2, ..., w. \tag{3.7}$$

The expected probability that a packet with the same delay statistics can be received by D^r is [27]

$$\epsilon(F(D^r)) = \frac{r}{w+1}, r = 1, 2, ..., w. \tag{3.8}$$

In Equation (3.8), the expected probability $\epsilon(F(D^r))$ cannot reach beyond $\frac{w}{w+1}$ or below $\frac{1}{w+1}$. To solve this problem, [27] extends the order statistics in Equation (3.6) by adding

$$\begin{aligned} D^0 &= \max(D^1 - 2s_d, 0) \\ D^{w+1} &= D^w + 2s_d. \end{aligned} \tag{3.9}$$

where s_d is the standard deviation of $d_n^{i-w+1}, d_n^{i-w+2}, ..., d_n^i$. Then the extended order statistics is

$$\begin{aligned} D^0 \leq D^1 \leq D^2 \leq ... \leq D^w \leq D^{w+1}, \\ \epsilon(F(D^r)) = \frac{r}{w+1}, r = 0, 1, 2, ..., w, w+1. \end{aligned} \tag{3.10}$$

Given a user-specified loss rate ε_l, a threshold of the end-to-end delay for the $(i+1)$-th packet, $\hat{d}_{max}^{i+1}$, can be estimated by searching for the smallest possible delay $D^{\hat{r}}$ to achieve ε_l. In other words, $\hat{d}_{max}^{i+1}$ is the greatest $D^{\hat{r}}$ which makes $(F(D^r)) \leq 1 - \varepsilon_l$. According to Equation (3.10), the index $\hat{r}$ can be obtained by

$$\hat{r} = \lfloor (w+1)(1-\varepsilon_l) \rfloor. \tag{3.11}$$

Then $\hat{d}_{max}^{i+1}$ can be approximated by the interpolation between $D^{\hat{r}}$ and $D^{\hat{r}+1}$ as

$$\hat{d}_{max}^{i+1} = D^{\hat{r}} + (D^{\hat{r}+1} - D^{\hat{r}})[(w+1)(1-\varepsilon_l) - \hat{r}]. \tag{3.12}$$

with $\hat{d}_{max}^{i+1}$, $\hat{t}_p^{i+1}$ can be obtained by

$$\begin{aligned} \hat{t}_p^{i+1} &= t_s^{i+1} + \hat{d}_{max}^{i+1} \\ &= t_s^i + L_0 + \hat{d}_{max}^{i+1}. \end{aligned} \tag{3.13}$$

where t_s^i is the time when i-th packet was sent.

For the "spike" case, where the current delay exceeds the previous ones by an amount over a threshold value, the scheme in [27] switches to *rapid adaptation mode*. In *rapid adaptation mode*, the first packet with unpredictable high delay is discarded [27]. After that, $\hat{d}_{max}^{i+1}$ is set to the last "spike delay" without considering or further updating the order statistics. *Rapid adaptation mode* is switched off when the delays drop down to the level before the mode is in force and the algorithm returns to its normal operation reusing the state of order statistics before the spike occurred [27].

This algorithm uses intra-talkspurt adaptation to adjust its playout buffer. It is low complexity. PLR is used as a quality index for estimating playout deadline. However, a low PLR does not guarantee high quality, especially in the case that a PLR is reduced by choosing a large buffer to protect more late packets. And the large buffer results in long conversational delay which might break up the conversational interactivities.

3.2.4 Quality-based approaches

In current quality-based buffer design/optimization for VoIP, voice quality is used as the key metric since it links directly to end-user perceived quality. Many methods (see [37]) use the E-Model [1] to estimate voice quality.

Since 2003, the E-Model [1] is widely used in VoIP playout buffering since it provides a link between quality and network impairments (packet loss and delays). An early work of using the E-Model in playout buffering framework is [81], in which the E-Model is used to evaluate the four algorithms in Section 3.2.2 with real network traffic traces. Even though the work is not for designing a playout buffer, it presented the relationship between perceived VoIP quality and buffer designs, using the E-Model.

Later in [5], Sun and Ifeachor proposed a playout buffering algorithm based on the E-Model optimization. In [5], the R factor is expressed as a function of packet loss ρ and end-to-end delay d, and the relationship between ρ and d is described by delay cumulative distribution function (CDF). Accordingly, the R factor as a function of d can be

obtained. To estimate the CDF of delays, three logarithmic expressions are exploited, i.e., Exponential, Pareto, and Weibull.

There are other E-Model based playout buffering algorithms in literature. For example, in [39], a simplified E-Model expression is used particularly for G.729A codec. The loss rate as a function of delays is obtained by Chebyshev's inequality. Another example is [82], in which the parameters of forward error correction (FEC) and playout delays are optimized based on perceived quality (estimated by the E-Model). In this subsection, we elaborate the algorithm in [5], which is used for performance comparison in our experiments.

In [5], an overall impairment function I_m is defined as

$$I_m = f(d, \rho) = I_d + I_{e\rho}. \tag{3.14}$$

where $I_{e\rho}$ is the impairment caused by packet loss. Packet loss rate ρ consists of two parts: network loss ρ_n and the buffer loss caused by buffer underflow ρ_b.

Considering that the equipment impairment in Equation (2.12) consists of two components $I_e = I_{ec} + I_{e\rho}$, where I_{ec} is the impairment without loss and $I_{e\rho}$ is the impairment with loss, the R factor in Equation (2.12) can be further simplified as

$$R = 93.2 - I_d - I_e = (93.2 - I_{ec}) - I_m. \tag{3.15}$$

Thus the optimization criterion is changed from maximization of R factor to minimization of overall impairment I_m.

In [5], $I_{e\rho}$ is derived using a non-linear regression model, and the form of $I_{e\rho}$ is

$$I_{e\rho} = a \times \ln(1 + b\rho) + c. \tag{3.16}$$

The parameters a, b, c are different for different codecs under PESQ and PESQ_LQ.

Recalling Equation (2.13), the I_m can be expressed as

$$\begin{aligned} I_m &= I_{e\rho} + I_d = a \times \ln(1 + b\rho) + c + 0.024d + 0.11(d - 177.3)\, H(d - 177.3), \\ &= a \times \ln(1 + b(\rho_n + \rho_b) + c + 0.024d + 0.11(d - 177.3)\, H(d - 177.3), \end{aligned} \tag{3.17}$$

with

$$H(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0. \end{cases}$$

Hence, the optimization criterion for playout buffering is defined as

> Given d, ρ_n and codec type (determine a, b, c), estimate an optimized playout delay d_{opt} such that I_m is minimized.

Given d, ρ_n, the late packet loss caused by buffer underflow, ρ_b, can be calculated as

$$\begin{aligned} \rho_b &= (1 - \rho_n/100)P(X \geq d) \\ &= (1 - \rho_n/100)(1 - F(d)). \end{aligned} \tag{3.18}$$

where $P(\cdot)$ is the possibility distribution function (PDF) and $F(\cdot)$ is the CDF. To estimate CDF of delays, [5] investigated three distribution candidates (details in Table 3.1), and proved Weibull distribution to be the best fitting function for the collected Internet trace. Then replacing $F(\cdot)$ by Weibull function in Equation (3.18), then the overall impairment factor I_m in Equation (3.17) can be expressed as

$$\begin{aligned} I_m &= a \times ln(1 + b(\rho_n + \rho_b) + c + \\ &\quad 0.024d + 0.11(d - 177.3)\, H(d - 177.3) \\ &\quad = 0.024d + 0.11(d - 177.3)\, H(d - 177.3) + \\ &\quad a \times ln(1 + b(\rho_n + (1 - \rho_n/100)e^{-((d-\mu)/\alpha)^\gamma})) + c. \end{aligned} \tag{3.19}$$

In this way, I_m is expressed as a function of d. An optimization process can be applied to get d_{opt}.

Table 3.1 Definition of Candidate Cumulative Probability Distributions [5]

Distribution	Exponential	Pareto	Weibull
CDF: $F(x)$	$1 - e^{-(x-\mu/\beta)}$	$1 - (k/x)^\alpha$	$1 - e^{-((x-\mu)/\alpha)^\gamma}$

In [5], steps were also taken for delay "spike", the case that a number of packets have significantly higher delays than the previous ones. The pseudo-code of the algorithm in [5] is shown in Algorithm 3

3.3 Semi-fixed playout buffering

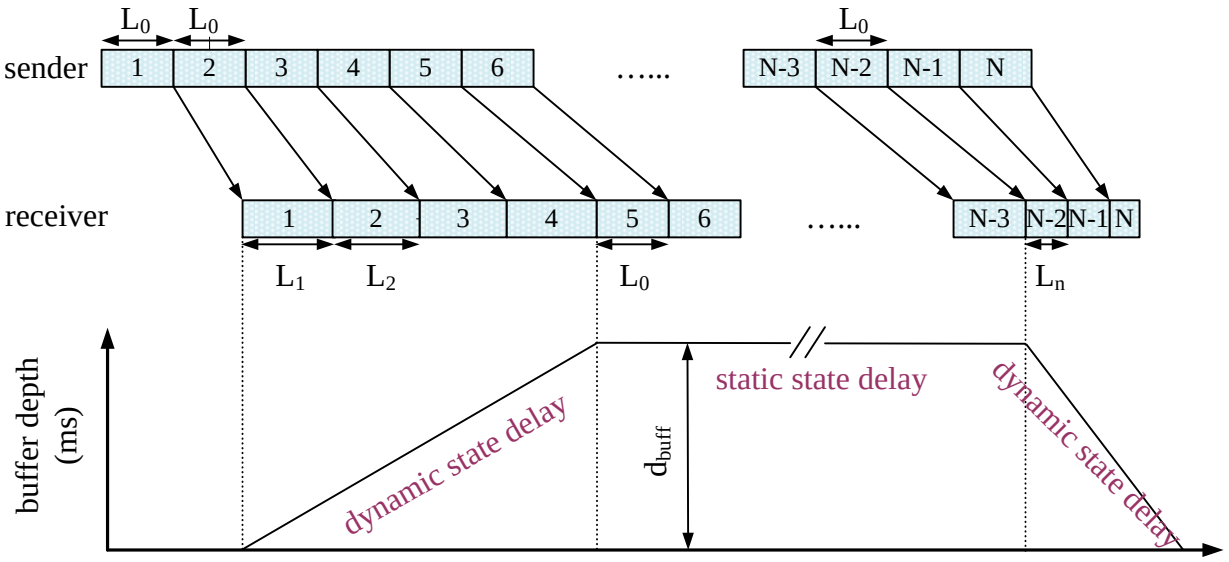
A different approach to buffering appears in [2]. This algorithm aims to provide adequate buffering for most speech packets while reducing the apparent conversational delay.

Algorithm 3 Adaptive Buffering Algorithm by Sun and Ifeachor [5]

For every packet i received, calculate network delay d_n^i
if $mode == SPIKE$ **then**
  if $d_n^i \leq tail \times old\_d$ **then**
    $mode = NORMAL$; % the end of a spike
  end if
else if $n_i \geq head \times d_i$ **then**
  $mode == SPIKE$; % the beginning of a spike
  $old\_d = d_i$;
else
  update delay records for the past W packets
end if
At the beginning of a talkspurt
if $mode == SPIKE$ **then**
  $d_i = d_n^i$;
else
  obtain (μ, α, γ) in Weibull distribution
  search playout delay d for $d = d_{opt}$ which meets $\min(I_m)$
end if

The playout scheduling algorithm continuously accumulates the buffer size to a fixed value at the beginning of a conversation turn and continuously decreases the buffer size at the end of each talkspurt. Figure 3.2 shows the buffer management for the first talkspurt in one conversation turn. When the first packet of the first talkspurt arrives, the decoded speech is stretched using TSM and then is played out without additional delay. Since the length of speech segment is extended from L_0 to L_1, the size of playout buffer for the following packets becomes $L_1 - L_0$. Consequently, the second packet gets $L_1 - L_0$ buffer protection, i.e., the packet would not miss its playout deadline as long as its relative delay [2]is less than $L_1 - L_0$. The speech segment decoded from the second packet is also stretch and hence the playout buffer size increases $L_2 - L_0$. This stretching strategy continues until the buffer size reaches a predefined maximum (we call it as "static delay"). The following packets are played out at original length (L_0). At the end of the talkspurt, the decoded speech from "hangover" packets are contracted and the buffer size accordingly decreases until it is zero. This gives zero additional delay for next talkspurt. Therefore, the conversational delay which is defined in Section 2.3.2 is reduced by: immediately playing out the speech represented in the first packet of the first talkspurt and compressing the speech segment

decoded from "hangover" packets.



L_0 – original length of speech carried in one voice packet (ms)
L_1 – length of stretched speech segment carried in Packet 1
L_2 – length of stretched speech segment carried in Packet 2
L_n – length of compressed speech segment carried in Packet N
Note that L_1 and L_2 might be different due to the PWSOLA algorithm.

Figure 3.2 playout buffering scheme for one talkspurt in [2]

A conversation turn consists of at least one talkspurt, and the playout buffering for one conversation turn is shown in Figure 3.3. In this design, the beginning part of the first talkspurt is expanded whereas no stretching is performed for the following talkspurts, and the buffer size is set to be the static state depth d_{static} by adding silence intervals. For real-time applications, it is hard to discriminate the end of a conversation turn or the end of a talkspurt, and therefore compression is triggered by the first "hangover" packet in each talkspurt. According to the definition of "hangover" in G.729 [41], a "hangover" packet is actually a non-active voice packet sent as an active voice packet to avoid speech clipping. So the compression of speech segment represented in "hangover" packets and the following silence period has little effect on the perceived quality and understanding of the conversation. In [2], an enhanced variable rate codec (EVRC) vocoder is used for coding/decoding the speech transmitted. There are three rates in EVRC: full rate for

2. A relative delay is the difference between a packet's network delay and the base delay. In this work, we use the network delay of the first packet received as the base delay.

active speech, 1/2 rate for "hangover" segment, and 1/8 rate for silence duration. Then "hangover" can be detected by receiving an 1/2 rate packet following a full rate packet. As shown in Figure 3.3, almost all active voice packets, except the beginning part of the first talkspurt, are protected by a playout buffer with the size of d_{static} [2].

"Hangover" can happen in the middle of a talkspurt, that is, the silence interval is relatively short, for example, the silence gap between one word. In this case, the decoded speech from following active voice packets (full rate) are stretched as if they were the beginning of a conversation turn. This prevents a noticeable silence gap being pushed into the word (e.g., "foot-ball" instead of "foot.ball") [2]. We present this processing in Figure 3.3.

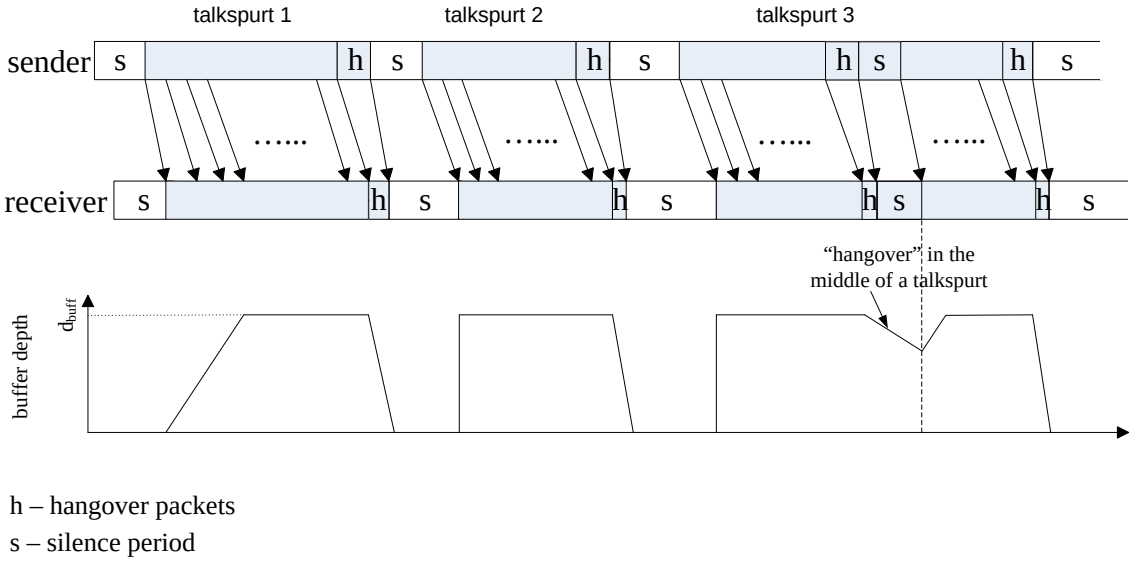


Figure 3.3 playout buffering scheme for one conversation turn in [2]

The design of playout buffering in [2] is low-complexity and efficient to reduce conversational delay with protection of most active voice packets. The problem of this design is that the static buffer depth d_{static} is predetermined and fixed for all talk-spurts. Since the propagation delay distribution is unknown, it is difficult to choose a proper d_{static}. In [2], it is claimed that $d_{\text{static}} = 60ms$ is enough achieve an acceptable level of packet loss for most wireless applications, but in wired IP network, packets can experience longer jitter delay than $60ms$. Although a large value can be chosen for d_{static} to reduce the probability of packet erasures, it increases the mouth-to-ear delay. This increases the conversational de-

lay and accordingly increases the risk of disrupting the conversation interactivity. Another problem is that the speech coder used is restricted to EVRC vocoder because "hangover", which triggers the compression process to reduce conversational delay. The "hangover" is detected by the packets using 1/2 rate, which is not available for other speech coders. In Section 3.4, we propose a quality-based adaptive playout buffering algorithm to overcome these problems.

3.4 Quality-based playout buffering for conversational VoIP

For conversational VoIP, conversational delay is one of main factors which impact on perceived quality. A large conversational delay can lead to double talk, echo or even the termination of the conversation. For some applications, quick response is required, for example, in a highly interactive business negotiation. Therefore, for conversational VoIP, it is essential to keep conversational delay low while providing enough playout buffering for required quality, since high interactivity is important.

Hence, it is reasonable to take conversational delay into account when design a playout buffering algorithm for conversational VoIP. E-Model based playout buffering algorithms consider end-to-end delay. In the E-Model, the factor I_d in Equation (2.13) reflects the penalty to long end-to-end delay, and accordingly somehow protects the design against too long conversational delay. Therefore, conversational quality of E-Model based algorithms can be improved if special steps are taken to reduce conversational delay. Thanks to Lee et al., they proposed a processing to further reduce conversational delay in [2] (see Section 3.3 for details).

Motivated by these, our quality-based playout scheme is accordingly designed by two parts. We use the R factor in the E-Model [1] as the cost index to obtain playout delays which adapt for each talkspurt. To reduce conversational delay, we adopt the idea in [2] which takes advantage of using "hangover" frames: trigger the compression of the decoded voice to decrease the playout buffer depth at the end of the "talk-spurt". The algorithm is described as following:

– During a silence period, comfort noise is played out every 20 ms, using SID information received. If an SID packet does not arrive before the playout time scheduled, the most recent received SID information is used to generate the comfort noise. The playout buffer size is zero. Information about occurred packet losses and transmission

delay are stored.

- When the first voiced packet of the first talk-spurt arrives, PWSOLA (Chapter 2 Section 2.5) is applied to stretch the decoded speech before it is played out. The playout buffer size increases by $(\alpha - 1) \times T_F$ [3]. The d_{static} is estimated based on previously stored information (window size is 1000 packets).
- When the estimated d_{static} is achieved, the decoded speech is not stretched any further. The depth of playout buffer keeps the maximum value d_{static} and α is set to 1.
- At the end of a talk-spurt, when the hangover is detected, PWSOLA is applied to compress the decoded speech before it is played out. The playout buffer size decreases by $(1 - \alpha) \times T_F$. Compression stops when the buffer depth is decreased to zero. It is possible for "hangover" to happen in the middle of the talk-spurt (see Figure 3.7), for example, the silence gap within a word. In this case, we stretched the subsequent voiced packet as if it were the beginning of the talk-spurt. A noticeable silence gap can be avoided [2] (details in Section 3.3).
- For the "spike" case, we follow the same steps in [5] (see Section 3.2.4 for details).

In this algorithm, two values, α and d_{static}, need to be designed. α is chosen as follows: $\alpha \geq 1 + T_p/T_F$ (T_p is one pitch period) during the stretching process; $\alpha \leq 1 - T_p/T_F$ during the compression process; $\alpha = 1$ during a silence period or when the estimated d_{static} is reached. For each talk-spurt, d_{static} is estimated based on maximizing the expected voice quality. According to Equation (3.15), the maximization of the R factor is equal to minimizing I_m, which is the function of playout delay given the network loss.

To determine d_{static}, we use Equation (3.17) as the cost index. In Equation (3.17), $a \times \ln(1 + b\rho) + c$ is used to calculate the impairment caused by packet loss. To determine a, b, c, we use the curve fit of the measured data for G.711 from ITU-T G.113. Figure 3.4 shows the data and result, and the following equation is obtained:

$$I_e = 53.64 \times \ln(1 + 0.063516\rho) \tag{3.20}$$

where I_e is the impairment caused by packet loss and ρ is the packet loss rate in percentage.

3. T_F is the original length of speech segment carried in a packet.

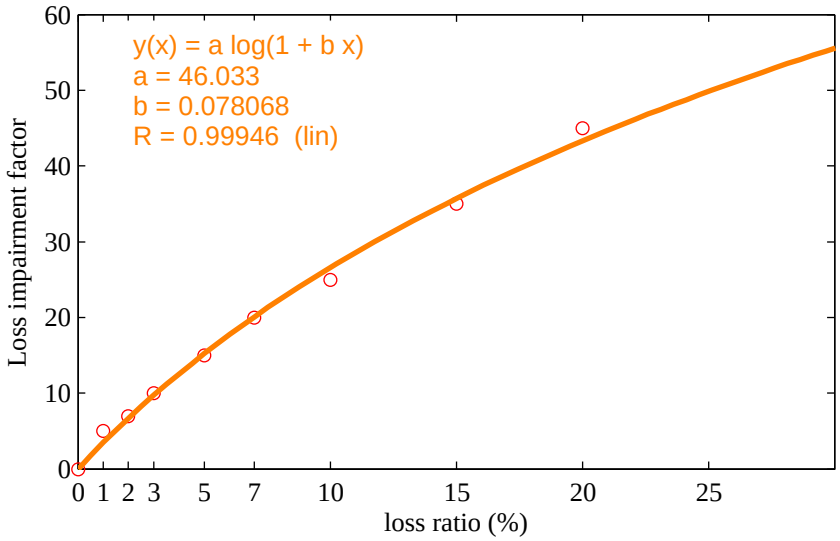


Figure 3.4 Loss impairment factor I_ρ vs. Packet loss rate ρ and a curve fit of the measured data from ITU-T G.113 [3].

Then Equation (3.17) can be rewritten as:

$$\begin{aligned} I_m &= I_d + I_\rho \\ &= 0.024d + 0.11(d - 177.3)\, H(d - 177.3) + 53.64 \ln\big(1 + 0.064(\rho_n + \rho_b)\big), \end{aligned} \tag{3.21}$$

with ρ_n is the network loss and ρ_b is the loss caused by buffer, which depends on the playout delay. The factor ρ_b can be calculated as

$$\begin{aligned} \rho_b &= (1 - \rho_n)P(X > d) = (1 - \rho_n)(1 - P(X \leq d)) \\ &= (1 - \rho_n)(1 - F(d)), \end{aligned} \tag{3.22}$$

in which $F(d)$ is the cumulative distribution function (CDF) of delay. Assuming the shape of the distribution tail is known, many works use *a priori* selected distributions to estimate the CDF of the delay distribution, for examples, Exponential in [79], Pareto in [83], Weibull in [5]. However, it was noticed that the playout delay may be very sensitive to the type of distribution used [38]. It is observed that the distribution of network delay is long-tail, which can be better presented by histogram. Therefore, we choose the statistical model based on the histogram which is more general and makes no assumption on the delay distribution. In this work, we use $w = 1000$ the most recent packets to obtain the

histogram. Assuming network packet loss rate is 2%, Figure 3.5 shows I_m vs. d for our trace from Canada to China. With the optimum playout delay d_{opt} that minimizes I_m, it is easy to get an optimum d_{static} for each talk-spurt. For a conversational application, this design is used on the both sides of sender and receiver.

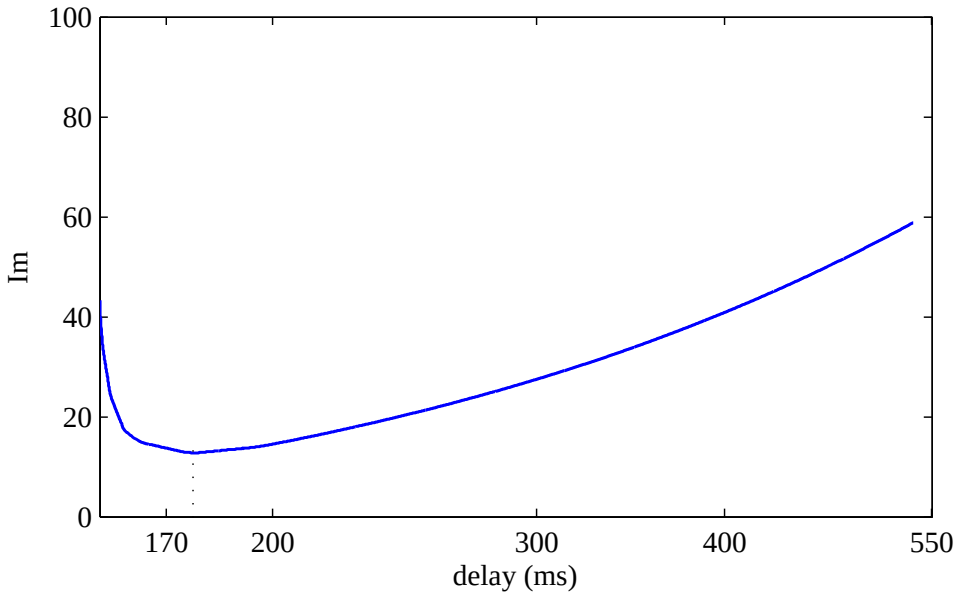


Figure 3.5 I_m vs. d.

In our playout scheme, conversational delay is reduced by two steps:

– stretch the first packet of a talk-spurt and play it out as soon as it arrives (the stretching increases the buffer depth)

– compress the voiced packets in a playout buffer whenever a "hangover" packet is detected (the compressing decreases the buffer depth)

Therefore, it is essential to detect the beginning of a talkspurt and the "hangover" packets, since they trigger the stretch and compress operations. The way to detect the beginning packet of a talkspurt is straight forward. As discussed in Chapter 2, the speech signal is packed into RTP packet to be transmitted over IP networks. In a RTP packet, the "M" field of RTP header (see Chapter 2) is set 1 for the first voice packet after a silence period. Therefore, it is natural to use this "M" field to determine when a talkspurt starts. The pseudocode is shown in Algorithm 4

In G.711 Appendix I, G.729 VAD is used to discriminate active voice frame and non-active voice frame. In VAD algorithm (see Appendix A for details), v_flag is used to

Algorithm 4 Detect the beginning of a talkspurt

```
for each packet received
CHECK_RTP_HEADER(packet)
if packet_header.M == 1&&talkspurt == 0 then
   % the first packet of a talk-spurt
   talkspurt = 1;
else if packet_header.M == 0&&talkspurt == 0&&packet_header.PT == 8 then
   % the first packet is lost
   talkspurt = 1;
end if
```

indicate when a hangover occurs. Thus, at the receiver side, the compression of speech is triggered as soon as a packet containing at least one frame with $v\_flag = 1$ arrives. Figure 3.6 shows the simulation of VAD and hangover, the red circle is the beginning of the hangover (v_flag is 1).

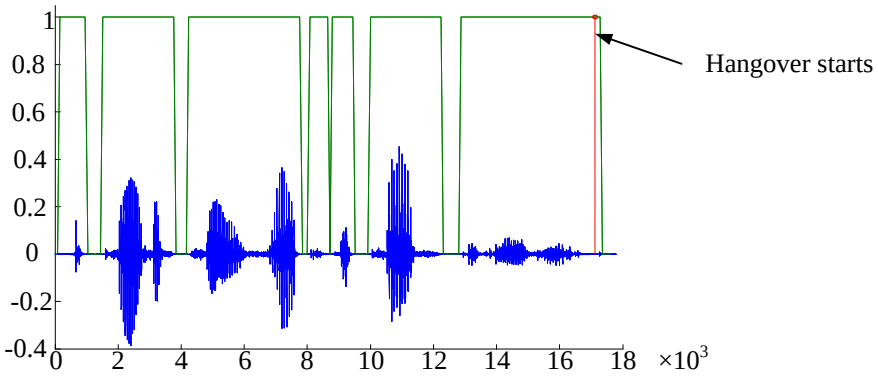


Figure 3.6 Hangover Detection using G.729 VAD

Based on the VAD algorithm in G.711 Appendix I, the pseudocode is shown in Algorithm 5.

Since the hangover detection scheme is based on the inherent VAD algorithm in the codec used (e.g., G.711 in this work), our adaptive playout buffering can be extended to other codecs which have built-in VAD algorithm.

In semi-fixed algorithm in [2] which was discussed in Section 3.3, the static buffer depth d_{static} is predetermined and fixed for all talk-spurts, the value of which is recommended to be more than 60ms to achieve an acceptable level of packet loss. The problem is that

Algorithm 5 Hangover Detection

```
for each packet received
DePacket(packet);
Decode(Packet.data, Speech_Seg);
VAD(Speech_Seg, v_flag);
if v_flag == 1 then
  % hangover packet
  hangover = 1;
end if
```

the propagation delay distribution is unknown, which makes it hard to choose a proper d_{static}. Although a large value can be chosen for d_{static} to reduce the probability of packet erasures, this increases the mouth-to-ear delay and degrades voice quality (see Figure 3.5). Moreover, a large d_{static} also increases the conversational delay and accordingly increases the risk of disrupting the conversation interactivity.

Figure 3.7 shows the playout buffering for one conversation turn (the "asking" turn), which contains only one talk-spurt. The first packet of talk-spurt is stretched and played out immediately when it is received. The speech compression begins when "hangover" is detected. In our experiment, the 81-st packet is the first "hangover" packet. For the proposed algorithm, compression starts at the 80-th packet, and for the case of $d_{\text{static}} = 60ms$, compression starts at the 78-th packet, because the latter case has larger buffer and the 81-st packet is stored in buffer before the 78-th packet is played out. Although more packets are contracted, more buffer delay remains for the buffer with $d_{\text{static}} = 60ms$, which increases the conversational delay accordingly.

Besides adaptive static buffer depth d_{static} and "hangover" detection, our playout buffering algorithm differs from semi-fixed algorithm on stretching processing. In semi-fixed algorithm, the stretching processing is applied to the active voice packets only at the beginning a conversational turn, while in our algorithm, we stretch the active voice packets at the start of each talkspurt. Instead of using fixed d_{static} in semi-fixed algorithm, our playout buffering scheme adapts d_{static} for each talkspurt. For some cases, e.g., the d_{static} for one talkspurt is much higher than the previous d_{static}, "pushing silence" as in semi-fixed algorithm might cause noticeable gap. To overcome it, we stretch the active voice packets at the beginning of each talkspurt.

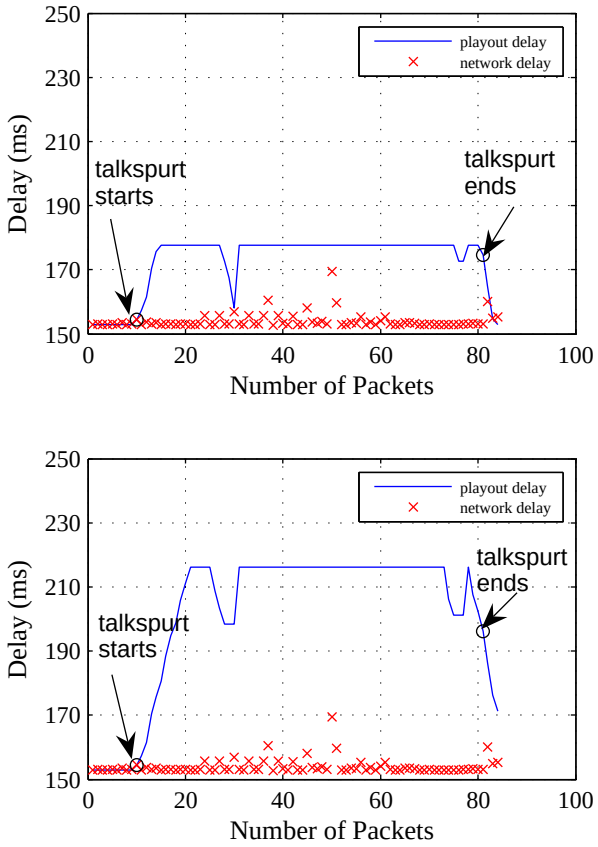


Figure 3.7 Playout Buffering : upper is our adaptive buffering algorithm; bottom is semi-fixed algorithm with fixed depth of $60ms$

3.4.1 Discussion

In our adaptive playout buffering algorithm, the static buffer depth d_{static} determines when to stop the stretching processing. d_{static} is calculated by $(d_{opt} - d_{base})$, where d_{opt} is the optimum playout delay which maximizes R factor and d_{base} is the base delay. In this work, the network delay of the first packet received is used as the base delay. If the base delay is relative high, e.g., the first packet is in a delay "spike", d_{opt} might be less than d_{base}. Accordingly d_{static} is negative. In this case, there is no adaptation, no stretching and no compression. As the result, the algorithm works as a fixed method discussed in Section 3.1. To tackle this problem, we use a threshold based on the history of past packet delays to avoid a too high base delay.

if $d_{base} > d_{hmax}$ then

$$d_{base} = d_{hav}$$

where

- d_{base} is the base delay
- d_{hmax} is the maximum network delay of the most recent $w = 1000$ packets
- d_{hav} is the average network delay of the most recent $w = 1000$ packets

As other playout buffering algorithms using between-talkspurt adaptation, our algorithm is subject to long talkspurts and very noisy background. In these cases, the algorithm fails to adjust buffer size. A possible solution is to set several thresholds based on R factor. In this way, the algorithm can adapt d_{static} according to observed changes of network conditions.

Our adaptive playout buffering algorithm uses PWSOLA to change playout buffer size by stretching/compressing decoded speech segments. The question is how perceived quality is affected by stretching/compressing speech segments. To study this issue, we randomly select 20 speech files (10 male, 10 female) from a speech database. Each speech file is 2–3 seconds in duration, and the speech signal is segmented every 20 ms (2–3seconds correspond to 100–150 segments) before PWSOLA [26] is applied.

For stretching case, we start stretching when the first talkspurt begins with stretching rate set to 1.2. For each file, we change the number of segments affected by stretching to achieve different amounts of buffer change. We then calculate the average MOS-LQO using PESQ [71] and average buffer increment [4]. When applying PESQ, the decoded speech signal (in this work, we use the G.711 decoder) is used as the reference signal. For the compression case, we compress $(n-1)$ segments previous to the first detected "hangover" segment and the "hangover" segment with compressing rate set to 0.5. The value n varies to achieve different buffer changes and Figure 3.8 shows the average MOS-LQO and average buffer decrement.

From our results, we observed that the processing of stretching and compression decreases the MOS-LQO when increasing the number of segments used in PWSOLA (to increase/decrease buffer size). Figure 3.8 shows that a high MOS-LQO can be kept during stretching process and achieve a relative large buffer change. For example, based on our results, MOS-LQO is above 4.0 with 42.06 ms buffer increment. Moreover, for the applications with long talkspurts (longer than the 2–3 seconds), the stretched speech is a small

4. Due to the individual packet-based nature of PWSOLA, modifying n packets does not achieve exactly the same buffer change for each of the 20 speech files.

proportion and hence the effect of stretching on quality is less than our result in Figure 3.8.

Figure 3.8 shows that the compression degrades MOS-LQO faster than stretching. To keep MOS-LQO above 4.0, no more than 19.20 ms of buffer decrement can be achieved. However, Figure 3.8 does not reflect the exact operation of our algorithm. In our algorithm, the compression starts when receiving a "hangover" packet. The number of packets to be compressed depends on the buffer depth accumulated by stretching process at the beginning of a talkspurt. For example, if the buffer depth is 40 ms, at most 2 packets are available in the buffer, one of which is the "hangover" packets. The compression continues reducing the buffer depth by compressing the following "hangover" packets and silence duration until the buffer size is zero. Compressing these packets does not degrade perceived quality as much as compressing the packets in talkspurts, which is shown in Figure 3.8.

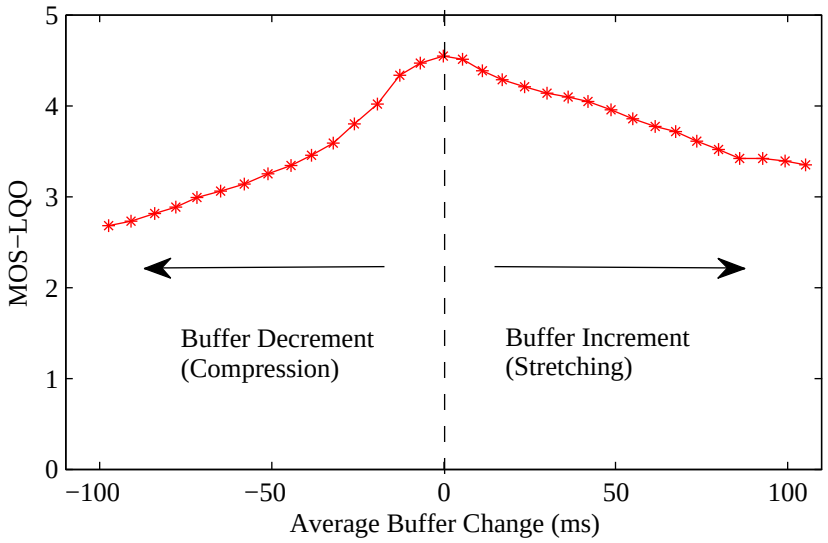


Figure 3.8 Effect of stretching on PESQ MOS-LQO scores

3.5 Summary

In this Chapter, playout buffering in VoIP was introduced. We discussed several algorithms including Lee's algorithm [2] in semi-fixed buffering and E-Model based adaptive buffering algorithm by Sun and Ifeachor in adaptive buffering. The attractive part of Lee's

algorithm (semi-fixed algorithm) lies in the procedure of reducing conversational delay by continuously changing the buffer size. Although the E-Model based adaptive algorithms take care of end-to-end delay, conversational delay can be further reduced by Lee's method (semi-fixed algorithm). Based on these, we develop a new adaptive playout buffering algorithm for conversational VoIP (in Section 3.4), which considers both voice quality (estimated by the E-Model R factor) and conversational delay. Without the limitation of using AMR codec like in [2], our algorithm can be extended to other codecs, e.g., G.729, G.723.1, iBLC, etc, with built-in a VAD algorithm, besides G.711.

Chapter 4

Playout Buffering using Redundancy Information

As we discussed in previous two chapters (Chapter 2 and Chapter 3), missing packets impair perceived quality. Besides network loss (packets are dropped during transmission), buffer underflow also results in missing packets. Causing late packets to be dropped as if they did not arrive, buffer underflow is of concern for the design of playout buffer at the receiver side. Even though, a long buffer reduces the number of late packets, conversational delay is increased accordingly, with a consequent impact on interactivity. If the packet loss is bursty, degradation on perceived quality is more than that caused by isolated losses [44]. Therefore, it is desirable to improve perceived quality by reducing packet missing without adding further delay.

Many speech codecs, e.g., G.711, G.729, AMR, etc, have built-in packet loss concealment (PLC) algorithms to reconstruct the missing speech frames at receiver side (in Section 2.2.4). In some applications, multiple speech frames are packetized into one packet to save transmission bandwidth. Therefore, a single packet loss can result in a consecutive loss of two or more speech frames, which may not be filled in by PLC techniques successfully. Most PLC schemes are designed to gradually mute the output when successive frames are erased. For example, in G.711 Appendix I, the concealed speech is linearly attenuated with a ramp at the rate of 20% per 10 ms after the first erased 10 ms frame and is muted after 60 ms.

Another solution is sender-driven repair, in which the sender sends redundancy information to mitigate the impact of packet loss. In VoIP, forward error correction (FEC) [42]

is commonly used to provide additional information for missing packets. There are two classes of FEC: *media-independent FEC* which is independent of the contents of packets, and *media-dependent FEC* which uses knowledge of the stream in packets. With the additional information provided by FEC schemes, missing packets may be recovered. However, if the length burst loss is too long, FEC schemes may fail to recover the missing frames, especially in the case that the packet(s) containing redundancy information is lost. There is a trade-off between the efficiency of recovering lost packets and additional delay: the more packets protected by FEC, the more additional delay required. Therefore, FEC is very efficient to recover isolate packet loss and also can shorten the length of burst loss. The perceived quality can be improved accordingly.

A path diversity scheme is an alternative sender-based technique which uses multiple paths (here we consider two paths). Redundant information is sent on a second path. If the loss and delay characteristics of the two paths are uncorrelated, path diversity schemes are robust to burst losses. The information on a second path can be full redundancy or partial redundancy. In a full redundancy scheme, packets are 100% duplicated. In a partial redundancy scheme, only important packets (those which have a significant effect on perceived quality if lost), are duplicated and sent on a second path. In this way, network loading is reduced. However, importance detection at the sender's side would increase complexity, and for some applications, the increase in bit rate for fully duplication is preferred to an increase in complexity.

In this Chapter, we will focus on playout buffering with redundancy information for the purpose of quality improvement. We will firstly show the impact of bursty loss on perceived quality by an experiment in Section 4.1. In Section 4.2, FEC is briefly overviewed, and then we present our new FEC scheme without introducing additional delay and a quality-based adaptive playout buffering algorithm based on. In Section 4.4, path diversity methodology is discussed and four path diversity schemes are introduced. The bursty robustness of quality-based adaptive playout buffering algorithms with different ways of providing redundancy is compared in Section 4.5.

4.1 Burst loss vs. quality

In this section, we show the effect of burst packet loss on perceived quality. Perceived quality is calculated objectively using the ITU-T PESQ (Perceptual Evaluation of Speech

Quality) which is standardized in [71] (Section 2.4.2).

The experiment is set as following: We randomly select 20 speech files (10 male, 10 female) from a speech database. Each speech file is 2–3 seconds in duration. A 2-state Gilbert model is used for the packet loss process (Section 2.3.3). The transition probabilities are set such that the packet loss is 5% and that an expected burst length ($E[BL]$) is achieved. The $E[BL]$ is varied from 1 frame to 8 frames (10 ms for each frame). When $E[BL] = 1\times$frame, the packet loss is random, with no burst loss. For each $E[BL]$, we generate losses for each file and calculate the MOS-LQO scores using PESQ [71], and then average the scores. Figure 4.1 shows that the average MOS-LQO scores decline with the increment of $E[BL]$, i.e., quality can be improved if steps are taken to reduce burst losses.

In VoIP, packet loss concealment (PLC) techniques are widely used to fill in missing packets. In Figure 4.1, the lower values are for silence substitution of missing packets, and higher values are obtained by G.711 PLC algorithm. The G.711 PLC algorithm improves perceived quality, but the quality still drops down when $E[BL]$ increases. Therefore, when packets are lost successively in a long burst or when network delays suddenly increases for a period of time, PLC techniques are not entirely effective.

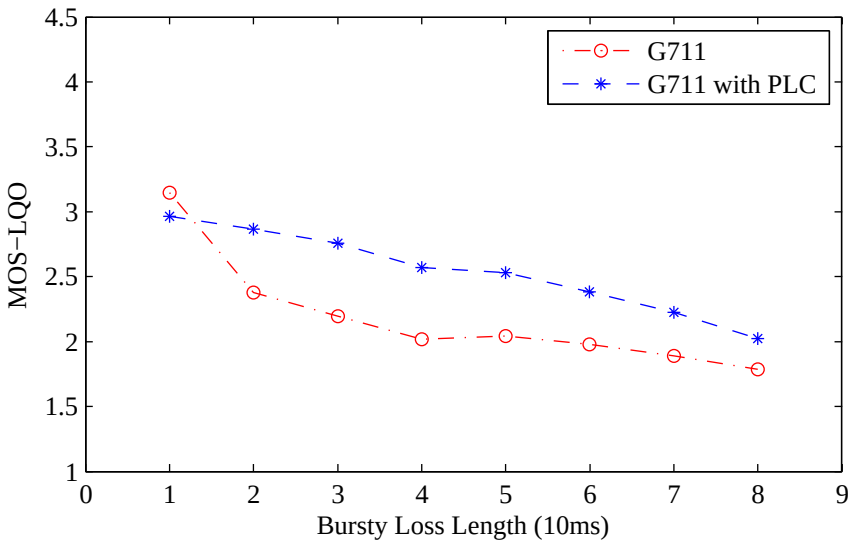


Figure 4.1 Expected burst length vs. MOS-LQO.

4.2 Forward error correction

Forward error correction (FEC) was originally developed for channel coding to control errors in data transmission over unreliable or noisy communication channels. The basic idea of FEC is to send redundancy information about the transmitted data so that this additional information can be used to recover the erased data. Therefore, quality can be improved with no requirement of retransmission. Note that in VoIP applications, it is desirable to avoid retransmission because the delay introduced by the procedure of retransmission degrades the perceived quality. There are a number of FEC techniques developed in literature, which can be classified as *media-independent FEC* and *media-dependent FEC*, depending on how they produce redundant information. *Media-independent FEC* applies algebraic codes to generate redundant information, which can be used for any type of data. *Media-dependent FEC* uses encoded signal as redundant information and requires low bit rate coding to avoid generating large packets, which is very suitable for speech applications like VoIP.

In *media-independent FEC*, additional packets, called FEC packets, are generated with algebraic codes. These FEC packets contain the redundancy information of previous packets, which can be used for packet recovery. For n packets, $n-k$ FEC packets are needed if each FEC packet contains redundancy of previous k packets. Parity codes are the simplest approaches, which utilize exclusive-or (XOR) operations to generate corresponding FEC packets. One simple example is illustrated in Figure 4.2, which is implemented by Rosenberg [84] for the case of isolated packet loss. The data in missing packet (Packet 2) is recovered by the FEC packet and Packet 1 and Packet 3. Supposing n-th packet as an FEC packet, it contains bitwise XOR on the previous $n-1$ packets. It is possible to recover a small number of consecutive packet losses by increasing the amount of redundancy and delay [12, Chapter15]. Another example is based on Reed-Solomon (RS) coding. It is a more powerful technique to recover packet erasure than the parity scheme. For k packets of data, RS(n,k) coding generates n packets with $n-k$ packets containing redundancy information for reconstructing lost packets. The redundancy packets can be simply generated by using parity symbols, called parity packets. Considering the overhead problem mentioned in Chapter 2 (Section 2.1.3), the parity packets (i.e., redundancy packets) can piggyback onto the original packets. Figure 4.3 shows an example of piggybacking RS(5,3) code [12, Chapter15]. The redundancy information for three original packets is coded into two parity packets which are piggybacked onto the original packets. For VoIP application,

RFC 2733 defines the format of an RTP payload using *media-independent FEC*.

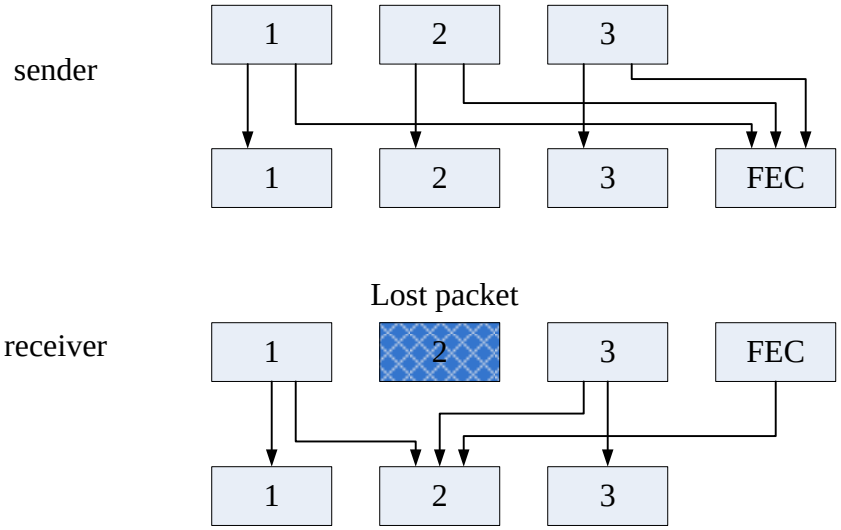


Figure 4.2 Parity coding in FEC.

Instead of using bitwise operation in *media-independent FEC*, *media-dependent FEC* generates redundancy packets by using encoded bitstream of previous packets. In the simplest scheme, a redundancy packet is formed by the exact copy of the original packet. Then the redundancy packet is piggybacked onto the subsequent packets and is used to reconstruct erased speech in the case that the original packet is lost. To lower the overall bit rate, *primary and redundant encoding* can be used to encode the redundant information using a lower rate-compression method, resulting in a lower quality for the recovered packet [12]. For example, G.711 (64 kb/s) as the main payload can be combined with GSM (13 kb/s) or G.729 (8 kb/s). Multiple redundancy packets can be piggybacked on a packet to safeguard against burst errors. That is, for packet n, it contains the primary encoding of the n-th segment of speech and redundant encodings of segments $n-1$, $n-2$, ..., $n-m$. Figure 4.4 shows the approach in which m redundant encodings ($B(i-1)$ – $B(i-m)$) are piggybacked on the subsequent primary encoding $A(i)$ in the following packets. Therefore, given a codec for primary coding, the size of a VoIP packet depends on the codec used as redundant encoding and the number of redundancy packets contained (m). It is to be noted that the speech payload of VoIP is small and so even for G.711 (64 kbits/s), the overhead for a 20 ms IP packet is 25%. For the lower rate coders, the overhead is much larger. Doubling the payload does not double the packet length. For VoIP application,

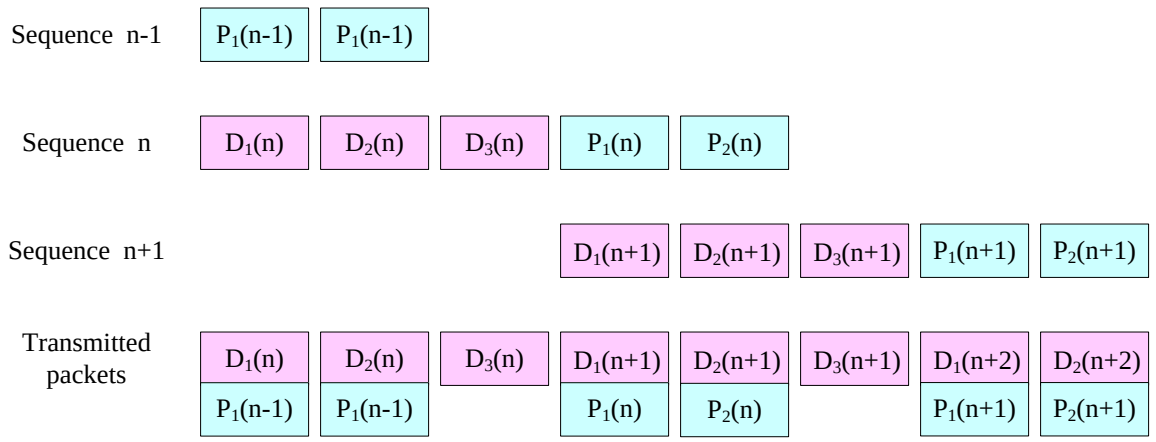


Figure 4.3 Piggybacking in an RS FEC.

RFC 2198 defines the format of an RTP payload using *media-dependent FEC*.

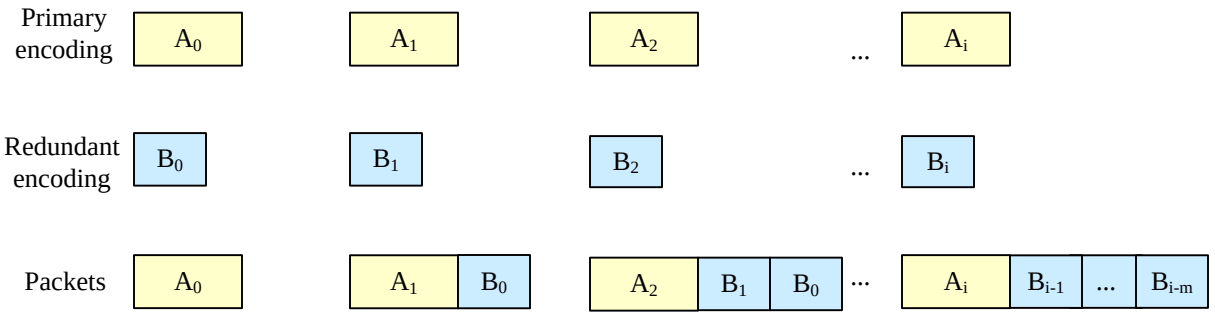


Figure 4.4 *Media-dependent FEC* with m redundancy packets.

4.3 Playout scheduling algorithm using FEC

For VoIP applications, the call quality is of the most concern. For conversational VoIP, conversational delay plays an important role on perceived quality. A long conversational delay breaks up the interactivity of a conversation. With conversational interactivity in mind, we developed a new adaptive playout buffering algorithm for conversational VoIP in Chapter 3, which takes into account both voice quality and conversational delay. To

improve quality further, we develop a new *media-dependent FEC* to mitigate the effect of missing packet on perceived quality.

4.3.1 A new *media-dependent FEC* scheme

Media-dependent FEC works at a sender side to send redundant information to recover the possible packet losses. To use the redundant information in *media-dependent FEC*, the decoder must implement a delay. For VoIP applications, a playout buffer is generally applied at the receiver to alleviate delay jitter. Therefore, since a playout buffer is already present at the receiver, no additional delays are needed if *media-dependent FEC* is integrated with playout buffering algorithms.

When *media-dependent FEC* works with playout buffering at the receiver's side, the size of a playout buffer influence the efficiency of recovering missing packets. The reconstruction from redundant information is possible only when the buffer size is greater than the time interval between the missing packet and the packet containing the corresponding redundant packet. In other words, the redundancy packet must arrive before the playout time scheduled for the associated missing packet. For example, if n-th packet is lost, it may be recovered with the piggybacked packet in the $(n+1)$-th packet only if the playout buffer size is greater than T_p (T_p is the duration of speech segment packetized in one active voice packet), and can be recovered using $(n+2)$-th packet only if playout buffer size is greater than $2 \times T_p$, etc. Therefore, it is reasonable to vary the number of redundancy packets at the send side according to the playout buffer size at the receiver's side. We only piggyback the voiced packets in talkspurts, and stop piggybacking whenever the "hangover" is detected (VAD/DTX from the G.729 [41]).

Shown in Figure 4.5, our *media-dependent FEC* scheme is

- At the sender's side: at the beginning of a talkspurt, piggyback previous m voiced packets, m is calculated according to the latest RTCP packet which contains the information of playout buffer at the receiver's side, stop piggybacking whenever the "hangover" packet is detected.
- At the receiver's side: at the beginning of a talkspurt, send RTCP packet if current playout buffer is changed greater than one packet length compared to the buffer size for the previous talkspurt.

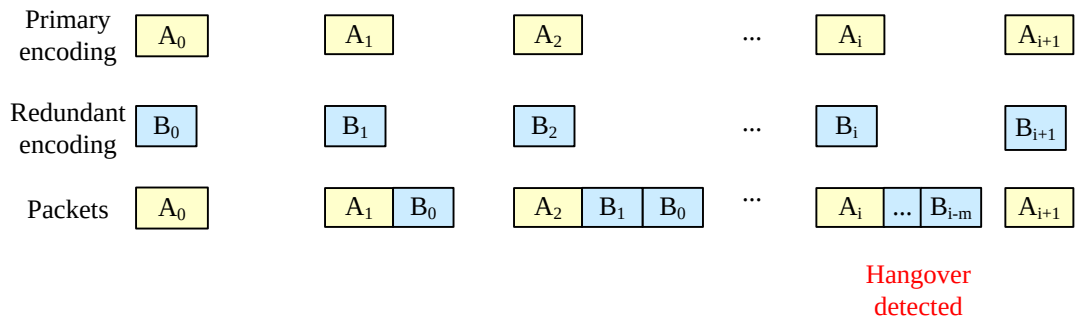


Figure 4.5 *Media-dependent FEC* scheme

4.3.2 Adaptive playout buffering with FEC

To apply the new *media-dependent FEC* to our quality-based adaptive playout buffering algorithm (details in Section 3.4), the static buffer depth is sent to the sender for determining the number of redundancy packets. Here, we elaborate how to calculate it.

According to Equation (3.15), Equation (3.17), and Equation (3.22), the relation between the E-Model factor R and playout delay d is as following

$$\begin{aligned} R(d) &= (93.2 - I_{ec}) - I_m \\ &= (93.2 - I_{ec}) - (I_d + I_\rho) \end{aligned} \tag{4.1}$$

with

$$I_d = 0.024d + 0.11(d - 177.3)\, H(d - 177.3) \tag{4.2a}$$

$$I_\rho = 53.64 \ln\bigl(1 + 0.064(\rho_n + \rho_b)\bigr) \tag{4.2b}$$

$$\rho_b = (1 - \rho_n/100)(1 - F(d)). \tag{4.2c}$$

where ρ_n is network loss, and $F(d)$ is the cumulative distribution function (CDF) of delay.

In our scheme, a static buffer depth d_{jb} is calculated for each talkspurt from the playout delay d_{opt}, which maximizes the R factor in Equation (4.1). The relation between playout buffer size d_{buff} and playout delay d is shown in Figure 4.6. Obviously, the relationship

between two delays is

$$\begin{aligned} d_{buff}(i) + TimeInterval(i) + d_{net}(1) &= d(i) + TimeInterval(i) \\ \Longrightarrow d_{buff}(i) &= d(i) - d_{net}(1). \end{aligned} \tag{4.3}$$

where $d_{net}(1)$ is the network delay of the first arrival packet, which is used in this work as the base delay. We have discussed how to choose base delay in Section 3.4.1. Accordingly, the static buffer depth d_{jb} can be calculated as

$$d_{\text{static}} = d_{\text{opt}} - d_{net}(1). \tag{4.4}$$

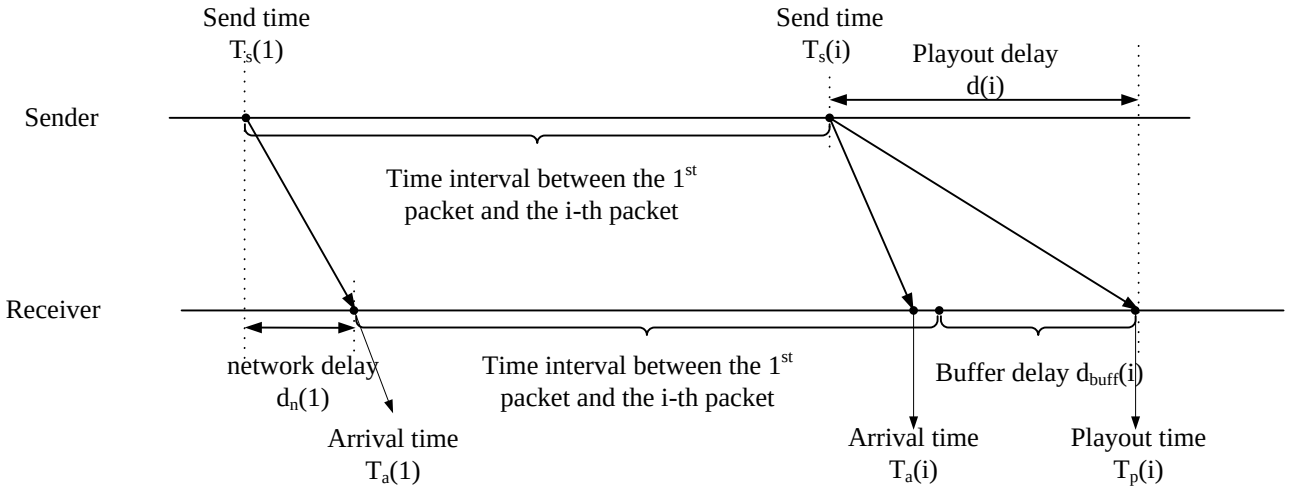


Figure 4.6 relation between playout buffer delay d_{buff} and playout delay d.

With the new *media-dependent FEC* and our adaptive playout buffering algorithm in Section 3.4, the new adaptive playout scheduling with FEC is described as following:

- During a silence period, comfort noise is played out every 10 ms, no matter whether the SID packet arrives or not. The playout buffer size is zero. Information about occurred packet losses and transmission delay are stored. SID packets are used to update the comfort noise parameters.
- When the first voiced packet of the first talk-spurt arrives, PWSOLA [26] is applied to stretch the decoded speech before it is played out. The playout buffer size increases by $(\alpha - 1) \times T_F$ (α is the stretch factor, and T_F is the payload length of a packet).

The d_{static} parameter is estimated based on previously stored packet delay information (window size is 1000 packets), send an RTCP packet with the information of d_{static} if the absolute difference between d_{static} and previous d_{static} is greater than T_p.

- When the estimated d_{static} is achieved, the decoded speech is not stretched any further. The depth of playout buffer keeps the steady-state value d_{static} and $\alpha = 1$.
- At the end of a conversation turn, when the hangover is detected, PWSOLA is applied to compress the decoded speech before it is played out. The playout buffer size decreases by $(1-\alpha) \times T_F$. Compression stops when buffer depth is decreased to zero. It is possible for "hangover" to happen in the middle of the talk-spurt, for example, during the silence gap within a word. In this case, we stretch the subsequent voiced packet as if it were the beginning of the talk-spurt.

For simplicity, we use the same G.711 encoder for original and redundant descriptions. Then a packet is played out either the packet itself or the following packets piggybacked with it are received.

4.4 Path diversity

Unlike FEC schemes in which the redundant information is sent along original packets using a same route, path diversity allows redundancy packets to be carried by a second path. If the loss and delay characteristics of the two paths are uncorrelated, the possibility of packet missing on both paths is very low. Hence, if packet loss happens on one path, the receiver can use the counterpart packets on the other path to reconstruct speech signal. Therefore, path diversity schemes are robust to burst losses if two paths are chosen as uncorrelated as possible. Moreover, with proper playout buffering, additional delays can be avoided with path diversity.

The overall bit rate can be lowered by either sending only important packets or using a lower rate-compression encoder (with lower attendant quality and higher complexity). Recalling Section 4.2, in *media-dependent FEC*, bit rate can be reduced by using a lower rate-compression to encode the redundancy information. We can also adopt this idea in path diversity schemes to keep low bit rate, that is, use lower rate-compression to encode the redundancy packets on a second path. For example, use G.711 (64 kb/s) on the "default" path (path 1), and GSM coding (13 kb/s) or G.729 coding (8 kb/s) on the second path (path 2).

When using path diversity, packets are required to be explicitly sent over different routes. One straightforward approach is IP source routing. The name derives from the fact that the source host (sender) can partially or completely specify the route that a packet should take across the network. However, special configurations are required for all nodes that a packet might visit on route to its destination [38]. Therefore, it is currently difficult to deploy in the Internet since not all ISPs (Internet Service Providers) support source routing.

Another solution is to use service overlay networks (SONs) which are developed to provide flexible routing process for VoIP [85]. The advantage of using SON is that an overlay network can be incrementally deployed on end-hosts running the overlay protocol software, without cooperation from ISPs. In SONs, service gateways act as overlay nodes and perform service-dependent routing at the boundaries of the Internet. In this way, a packet can be sent to the destination by a chosen route. Figure 4.7 shows a basic SON architecture. A commercial example is the technique used in Skype, which uses SONs with a peer-to-peer system (see details in [86]).

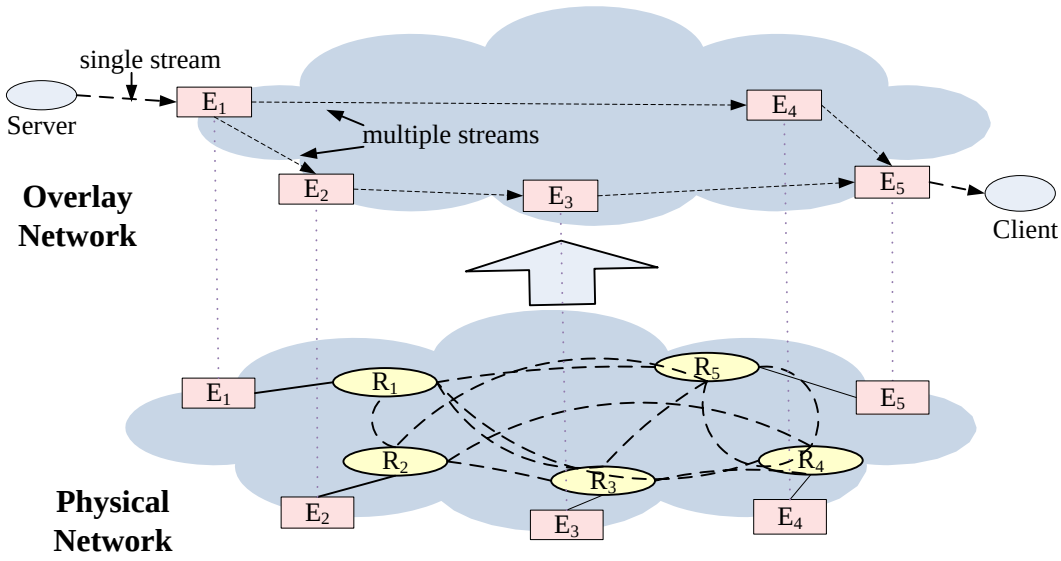


Figure 4.7 Service overlay network

In [87], relays were placed at a number of strategic nodes to forward a packet to its destination. A packet is sent to a relay first and the relay forwards it to the receiver with a specified routing. The basic idea is to encapsulate an original IP packet into a new packet

with the relay's address (RA) as the destination address (DA). In this way, the packet is sent to the relay before passing over the Internet. When the relay receives the packet, it strips off the RA and sends the packet back to the network with the packet's original destination address such that the packet can be routed to the final destination. Figure 4.8 shows a simple scheme by which packets are forwarded to destination over different paths using relays. In this work, we use a relay to simulate two paths diversity.

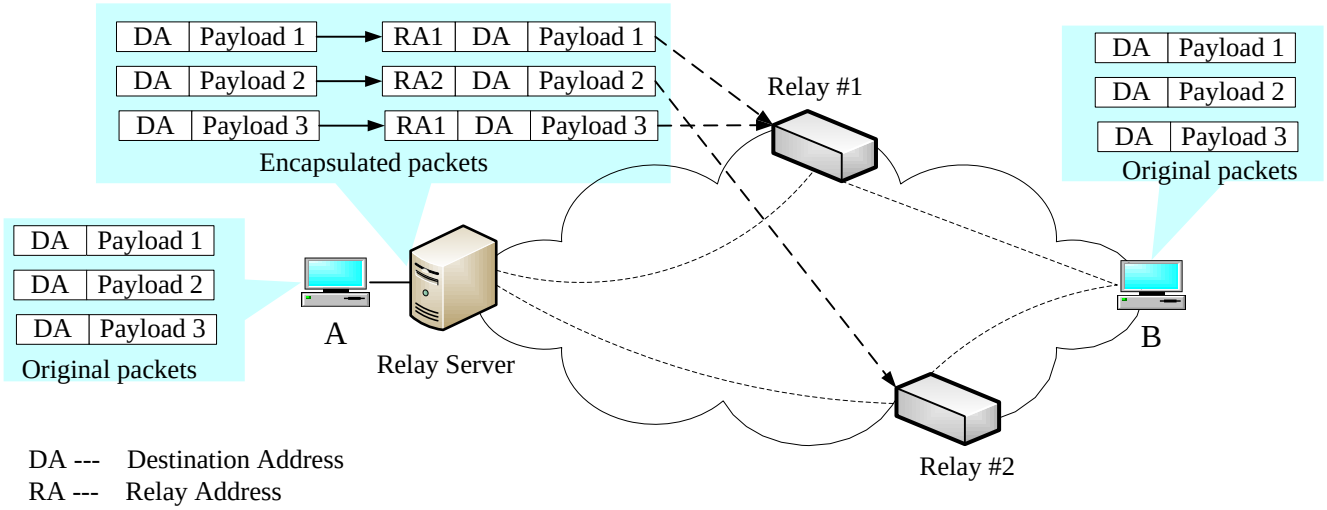


Figure 4.8 Path diversity using relays

4.4.1 Playout scheduling algorithms using path diversity

When used with quality-based playout algorithm, the packets carrying redundancy can also used to estimate the playout buffer size or playout delay.

In [88], playout delay for every packet is set by searching the optimal $d_{play}(i)$ which minimizes a Lagrange cost function:

$$\begin{aligned} C(i) =& d_{play}(i) + \lambda_1 \varepsilon_1(i)\varepsilon_2(i) \\ & \lambda_2\big(\varepsilon_1(i)(1-\varepsilon_2(i)) + \varepsilon_2(i)(1-\varepsilon_1(i))\big) \end{aligned} \tag{4.5}$$

where

- $\varepsilon_1(i)$ and $\varepsilon_2(i)$ are the estimated loss possibilities of path1 and path2
- λ_1 and λ_2 are Lagrange multipliers which are predefined to trade off delay and the two possibilities

The problem of this algorithm is that only packet loss rate is used to measure perceived quality. However, for some applications, low packet loss rate cannot guarantee the improved quality, especially in the case of costing long delay to achieve required packet loss rate.

In the work of Ghanassi [38], path diversity was used to improve the perceived quality for a E-Model based playout algorithm. In [38], both fully redundancy and partial redundancy schemes are considered. In a full redundancy scheme, the receiver selects the first arriving packet to reconstruct the speech signal and the minimum of the current delays for the two paths is used to estimate the playout delay for following packets.

Although Ghanassi's method guarantees acceptable conversational delay, delay can be further reduced by two steps that we use in this work. That is, first, the first packet of a talk-spurt is stretched and played out as soon as it arrives. This stretching process increases the buffer depth. Second, at the end of a talkspurt, compress the voiced packets in the playout buffer whenever the "hangover" packet is detected.

In our playout scheme, the speech signal is encoded and packetized before it is sent to path 1 and path 2. For simplicity, we use G.711 as encoder for both paths. At receiver side, redundancy packets on path 2 are used to recover missing packets from path 1, that is, if a packet is lost or arrives after it is scheduled to play out, the corresponding redundancy packet received from path 2 is used to reconstruct the speech. Figure 4.9 illustrates the scheduling process using path diversity with our adaptive buffering. Note that the packets at the start of a talkspurt are stretched (shown as being longer) to build up the buffer delay, and packets are compressed when "hangover" is detected.

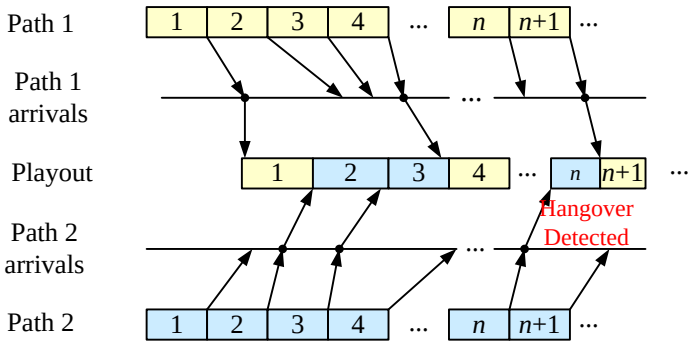


Figure 4.9 Path diversity scheme

When use path diversity with our quality-based playout algorithm in Chapter 3 (Section 3.4), the E-Model based cost function Equation (4.1) is rewritten as

$$\begin{aligned} R(p_{d1}, p_{d2}) &= (93.2 - I_{ec}) - I_m \\ &= (93.2 - I_{ec}) - (I_d(p_{d1}, p_{d2}) + I_\rho(p_{d1}, p_{d2}) \end{aligned} \tag{4.6}$$

where , and.

- p_{d1} is end-to-end delay on path 1
- p_{d2} is end-to-end delay on path 2

The steady-state buffer depth d_{static} is estimated by optimizing Equation (4.6). It is hard to handle Equation (4.6) directly. Here, we introduce 4 schemes to estimate d_{static} using path diversity.

Scheme 1 In this scheme, the first arrived packet on both paths is used. If the performances of the two paths are highly correlated, there is no significant gain in improving quality [38]. The operation is shown in Figure 4.10. The following is performed:

For every packet, use $d = \min(p_{d1}, p_{d2})$ *to update the delay window.*

At the beginning of a talkspurt, find the d_{opt} *which maximizes Equation* (4.1) $\rightarrow d_{\text{static}}$.

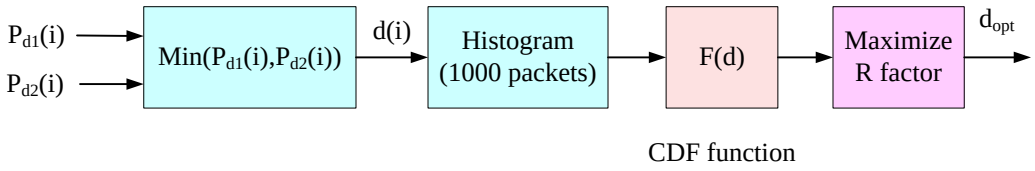


Figure 4.10 Path diversity scheme 1

Scheme 2 Path 2 is used only for reducing the packet loss on path 1, i.e., the packets on path 2 are used only when the corresponding packets are lost on path 1. Figure 4.11 shows the algorithm. This scheme is supposed to achieve high performance when the packet loss distributions of two paths are uncorrelated. The following is performed:

For every packet, use

$$d = \begin{cases} p_{d1}, & \text{packet on path 1 arrives} \\ p_{d2}, & \text{packet on path 1 is lost.} \end{cases}$$

to update the delay window.

At the beginning of a talkspurt, find d_{opt} which maximizes Equation (4.1) $\rightarrow d_{\text{static}}$.

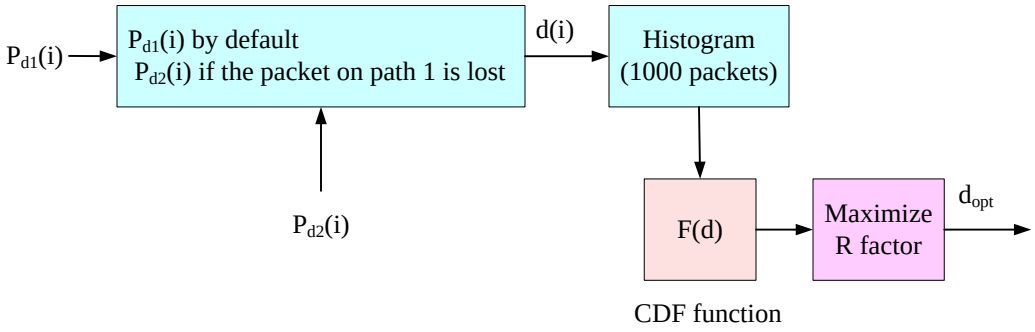


Figure 4.11 Path diversity scheme 2

Scheme 3 We proposed this scheme in [45]. In this scheme, the performance for both paths is estimated using the E-Model, and choose the minimum d_{static} for the current talkspurt. Figure 4.12 shows the algorithm. The following is performed:

For every packet, use

$$d^{(1)} = \begin{cases} p_{d1}, & \text{packet on path 1 arrives} \\ p_{d2}, & \text{packet on path 1 is lost.} \end{cases}$$

and

$$d^{(2)} = \begin{cases} p_{d2}, & \text{packet on path 2 arrives} \\ p_{d1}, & \text{packet on path 2 is lost.} \end{cases}$$

to update the delay windows for path 1 and path 2 respectively.

At the beginning of a talkspurt, search $d_{opt}^{(1)}$ and $d_{opt}^{(2)}$ which maximize R in Equation (4.1), $d_{opt} = \min(d_{opt}^{(1)}, d_{opt}^{(2)}) \rightarrow d_{\text{static}}$.

Scheme 4 This is a new scheme proposed in this work. The basic idea is to predict the performance using the E-Model on both paths, and choose the d_{static} which achieves higher R value for the current talkspurt. The operation is shown in Figure 4.13. The algorithm is explained as following:

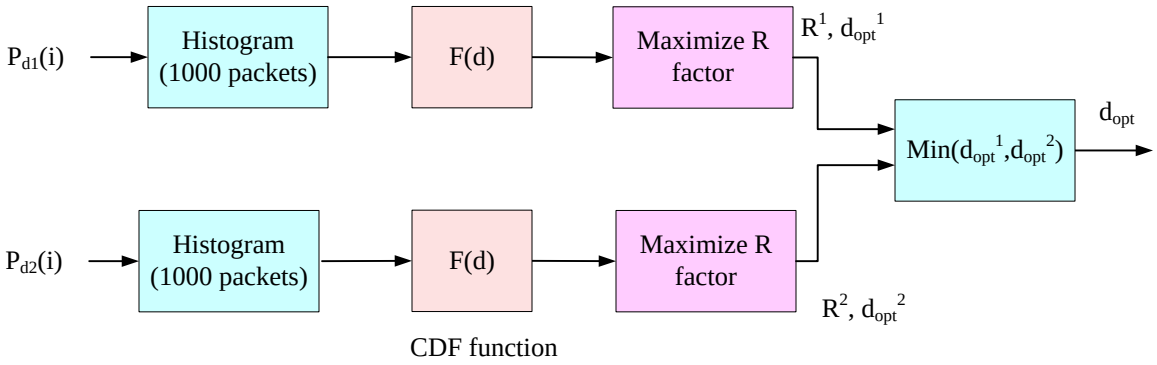


Figure 4.12 Path diversity scheme 3

For every packet, use

$$d^{(1)} = \begin{cases} p_{d1}, & \text{packet on path 1 arrives} \\ p_{d2}, & \text{packet on path 1 is lost.} \end{cases}$$

and

$$d^{(2)} = \begin{cases} p_{d2}, & \text{packet on path 2 arrives} \\ p_{d1}, & \text{packet on path 2 is lost.} \end{cases}$$

to update the delay windows for path 1 and path 2 respectively.

At the beginning of a talkspurt, search $d_{opt}^{(1)}$ *and* $d_{opt}^{(2)}$ *which maximize* $R^{(1)}$,$R^{(2)}$ *in Equation* (4.1)*,* d_{opt} *is the one which achieves* $\max(R^{(1)}, R^{(2)}) \rightarrow d_{\text{static}}$.

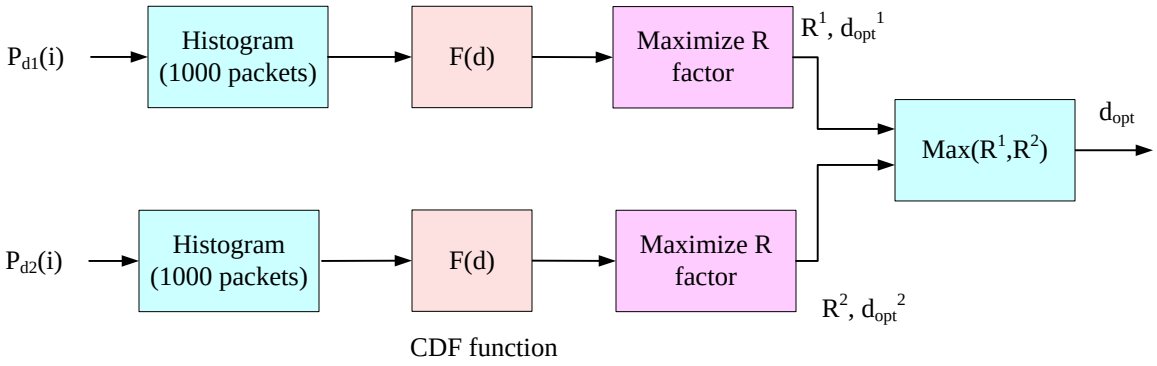


Figure 4.13 Path diversity scheme 4

Scheme 1 gives the smallest d_{static} in the case that two paths are uncorrelated. The d_{static} obtained from *Scheme 2* is slightly different from that estimated on a single path (path 1), because p_{d2} is used when a packet is lost on path 1, which changes CDF in Equation (4.2a). Indeed, *Scheme 2* can achieve high performance when $p_{d1} > p_{d2}$, and hence gives the highest d_{static}. In *Scheme 3* and *Scheme 4*, the d_{static} is between those from *Scheme 1* and *Scheme 2*, which keeps the playout buffer reasonable short.

4.5 Robustness to burst losses

Packet loss is a main factor influencing perceived quality. Burst losses degrade perceived quality. In this section, we investigate the robustness of our algorithms to burst losses. We randomly select 20 speech files (10 male, 10 female) from a speech database. Each speech file is 2–3 seconds in duration. To evaluate algorithms, the network channel is modelled from our Internet trace file from Canada to China (we will explain it later in Chapter 5), and a 2-state Gilbert Model (details in Chapter 2) is superposed to generate network losses. The transition probabilities are set such that the network loss is 5% and that an expected burst length ($E[BL]$) is achieved. The $E[BL]$ is varied from 1 packet to 19 packets (20 ms per packet). When $E[BL] = 1\times$packet, the packet loss is random, with no burst loss. The two paths for path diversity are simulated by randomly choosing two uncorrelated segments from the trace file.

Figure 4.14 shows the performances of different playout algorithms to different $E[BL]$: *Algorithm 1*, *Algorithm 2*, *Algorithm 3* and *Algorithm 4*, are the adaptive algorithm described in Section 4.4.1 with *Scheme 1*, *Scheme 2*, *Scheme 3*, and *Scheme 4* respectively. *FEC* is the algorithm illustrated in Section 4.3.1, which uses adaptive *media-dependent FEC* scheme to send redundancy. *Adaptive* is the adaptive algorithm in Chapter 3 without redundancy transmission. Perceived quality is calculated objectively using PESQ [71], whose output is MOS-LQO score. Note that PESQ only measures one-way quality and does not take conversational delay into account. *FEC* algorithm fails to improve perceived quality when $E[BL] \geq 7$ while the four algorithms with path diversity keep the MOS-LQO score hight when $E[BL]$ increases. Therefore, path diversity schemes are more robust to the burst losses, and can improve perceived quality than the other two algorithms. Among the four algorithms using path diversity, *Algorithm 2* achieves highest performance with highest MOS-LQO score because *Algorithm 2* gives the highest d_{static} for a talkspurt, which

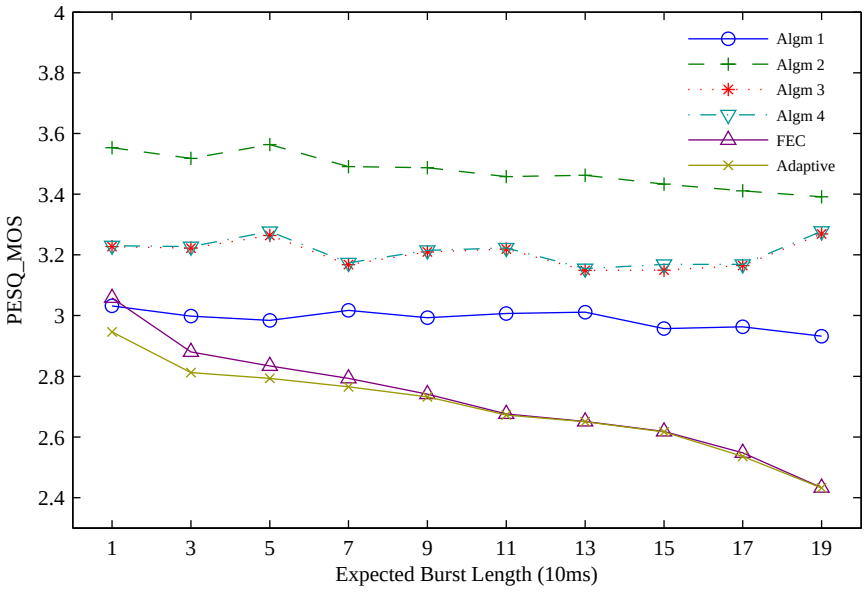


Figure 4.14 Performance Comparison with different E[BL].

reduces the impact of buffer underflow (late packets). *Algorithm 3* and *Algorithm 4* perform between *Algorithm 1* and *Algorithm 2*. Compared with *Algorithm 3*, *Algorithm 4* does not benefit too much by picking up higher R value (higher performance) between two paths: just a little bit improvement on MOS-LQO score. The reason is that the two paths used in this experiment have similar predicted quality based on the E-Model. For some cases, for example, the network condition (delay and packet loss) on either path turns to worse for a period of time, *Algorithm 4* can achieve better performance than *Algorithm 3*, since the algorithm is based on higher predicted quality of two paths.

4.6 Summary

When design playout buffering for VoIP applications, quality is the most important to be concerned. In VoIP, conversational quality includes perceived quality and interactivity (measured by conversational delay). Perceived quality can be improved by redundancy information. In this Chapter, we discussed two classes of sender-driven methodologies to improve perceived quality by using redundancy: FEC and path diversity. FEC schemes

are easier to implement than path diversity ones, whereas path diversity schemes achieve more improved perceived quality.

In this Chapter, we developed a new adaptive *media-dependent FEC* scheme and two new path diversity schemes which are based on the E-Model. Several E-Model based playout buffering algorithms are developed and compared based on the adaptive *media-dependent FEC* scheme and different path diversity schemes. Without increasing conversational delay, our quality-based algorithms with redundancy information improve the conversational quality for conversational VoIP.

Chapter 5

Experiments and Results

In Chapter 3, we discussed playout buffering for VoIP and explained our algorithm for conversational VoIP which is driven by both voice quality and conversational interactivity considerations. In Chapter 4, we developed several methods to improve perceived conversational quality by utilizing redundancy information. In this chapter, we will show the efficacy of these algorithms by experiments.

5.1 Experimental settings

In Chapter 2, we have explained the VoIP system used in this work (see Figure 2.3). Figure 5.1 shows the simulation of the VoIP system. Six speech files are from recordings of real dialogs with noisy background. Each speech file has two speakers (12 different speakers: 7 males and 5 females) and contains 2 conversational turns. Log-PCM, standardized by ITU-T G.711 [60], is used for encoder and decoder. The bitstream, output of the G.711 encoder, is encapsulated into IP/UDP/RTP packets (details in Section 2.1.3). These packets pass through a "Network simulator" which consists of two parts: a network delay simulator driven by a network trace file to simulate network delay and a loss model for network packet loss. Network trace files are collected from current IP network and contain round-trip/end-to-end delay and sequence numbers. The sequence numbers can be used to derive network loss distribution, i.e., the absent sequence numbers indicate that the corresponding packets are lost during transmission. Although network trace files contain the information of network packet loss [1], it is hard to manage network packet loss when investigating the impact of different packet losses on quality, e.g., burst network losses with

different lengths. Therefore, we remove the network losses from the trace files and superimpose a loss model based on 2-state Gilbert Model to generate desired network packet loss for our experiments. For each trace file, the six speech files share the same network delay and loss distribution.

The playout buffering algorithms used in this chapter are as follows:

- Exponential-average (**Exp-Avg**) in [31] described in Section 3.2.2
- Fast exponential average (**Fast-Exp**) in [83] described in Section 3.2.2
- **Adaptive_IS** in [5] described in Section 3.2.4
- **Adaptive_Gong** in Section 3.4
- **Adaptive_FEC** in Section 4.3
- Path Diversity **Algorithm 1** in Section 4.4.1.
- Path Diversity **Algorithm 2** in Section 4.4.1.
- Path Diversity **Algorithm 3** in Section 4.4.1.
- Path Diversity **Algorithm 4** in Section 4.4.1.

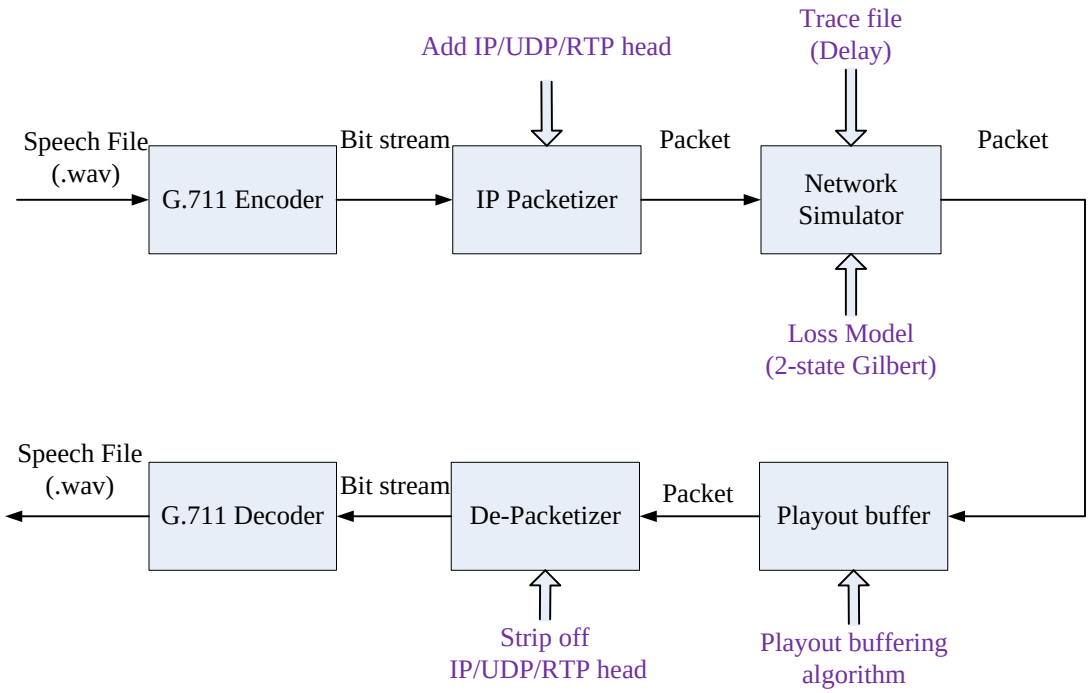


Figure 5.1 VoIP Simulation

1. The network packet loss rate for the three trace files used in this work are: 1.8% for Trace 1, 1.8% for Trace 2 and 14.3% for Trace 3.

The performance of different algorithms are evaluated by average PESQ scores [71] and average conversational delays, since PESQ algorithm [71] is a one-way measurement which does not consider conversational delay. The reference speech for each speech file used in PESQ algorithm [71] is the decoded speech (G.711 decoder) with 0% packet loss.

5.1.1 Network trace files

Data collection

In this work, the trace files used were obtained in two different ways: ICMP (Internet Control Message Protocol) based and UDP based. The former collection operates by sending "ICMP Echo Request" packets to the target host and waiting for the matching "Echo response" packets. ICMP tools, e.g., *ping* and *hrping* [89], can be used to measures round-trip delay and records packet loss. UDP based collection uses a UDP/IP probe tool to collect and measure network parameters, e.g., packet loss and end-to-end delay. A UDP/IP probe tool typically sends probe packets to the remote host by using UDP/IP over Internet. The hosts/servers at the two ends (Sender and Receiver) build the data files with the information carried by these probe packets.

Hrping is a ICMP tool which can be downloaded from [89]. The reason that we choose *hrping* instead of *ping* is because *hrping* has useful features suitable for measuring VoIP packets: timing the round trip delay in microseconds and sending out packet every x milliseconds. Following the same assumption for IP-based transport and VoIP application [90], the one way network delay is half of the round-trip delay in the trace file.

The utility of *hrping* is as following:

$$hrping < options >< host >$$

$< host >$ can be the IP address or the hostname. If the hostname is used, it will be resolved to its address at the beginning of the PING loop [89]. There are a couple of selected options:

$-t$ Ping the specified host until stopped. *Hrping* can be aborted any time with CTRL-C or CTRL-Break. Unlike Windows PING, *hrping* will still print the statistics gathered so far when aborted.

$-n$ *count* Specify the number of PING packets to send.

$-l$ *size* Send buffer size (ICMP payload size). This option can be used to specify the payload size.

$-L$ *size* Total IP datagram size (ICMP payload size + 28). Each packet is of the form: IP header (20 bytes) + ICMP header (8 bytes) + payload.

$-s$ *time* Interval in milliseconds between packets.

$-r$ Switch to traceroute mode. This mode works almost the same as Windows TRACERT, except that it only does one test per host, not three.

$-T$ Print timestamp in front of each line

$-i$ TTL Time To Live.

$-o$ Don't do overlapped send/receive.

$-E$ $file$ Stop pinging when $< file >$ exists

$-F$ $file$ write the result into $< file >$

An example of using *hrping* is

$$hrping - t - F./tracefile - T - L348 - s0.02 \quad 202.120.36.176$$

In this example, the trace result is stored in file "tracefile", and the following is a segment of the information in "tracefile"

```
This is hrPING v2.38 by cFos Software GmbH -- http://www.cfos.de

Using source IP address 142.157.11.162 to send packets
Pinging 202.120.36.176
with 320 bytes data (348 bytes IP):

2009-01-07 11:28:56.015: Reply from 202.120.36.176: seq=0000 time=305.977ms TTL=32 ID=e0e5
2009-01-07 11:28:56.031: Reply from 202.120.36.176: seq=0001 time=306.249ms TTL=32 ID=e0e6
2009-01-07 11:28:56.062: Reply from 202.120.36.176: seq=0002 time=305.958ms TTL=32 ID=e0e7
2009-01-07 11:28:56.078: Reply from 202.120.36.176: seq=0003 time=305.886ms TTL=32 ID=e0e8
2009-01-07 11:28:56.093: Reply from 202.120.36.176: seq=0004 time=305.650ms TTL=32 ID=e0e9
2009-01-07 11:28:56.109: Reply from 202.120.36.176: seq=0005 time=305.926ms TTL=32 ID=e0ea
2009-01-07 11:28:56.140: Reply from 202.120.36.176: seq=0006 time=305.604ms TTL=32 ID=e0eb
2009-01-07 11:28:56.156: Reply from 202.120.36.176: seq=0007 time=305.730ms TTL=32 ID=e0ec
2009-01-07 11:28:56.171: Reply from 202.120.36.176: seq=0008 time=305.664ms TTL=32 ID=e0ed
2009-01-07 11:28:56.203: Reply from 202.120.36.176: seq=0009 time=305.628ms TTL=32 ID=e0ee
2009-01-07 11:28:56.218: Reply from 202.120.36.176: seq=000a time=306.170ms TTL=32 ID=e0ef
```

Sequence Number

Round-trip Delay

As shown above, sequence numbers which infer the information of packet losses and round-trip delay are collected in "tracefile".

A UDP/IP probe tool can be downloaded from [4]. In this work, we use two trace files (Plymouth's trace files [2]) in [91], which were collected using the UDP/IP probe tool. The structure of UDP/IP trace data collection is shown in Figure 5.2. The system consists of four processes: Source, Echo, Sink and Logger. Typically, the Source and the Sink

processes run on the same local host (Sender Host), while the Echo process runs on a remote host (Receiver Host) [91]. The Logger process may run on any host and in [91], it runs on the Sender Host. The speech packets are generated by Source process and are sent to the Echo process via the Internet. The Echo process on Receiver Host sends the packets back when it receives them.

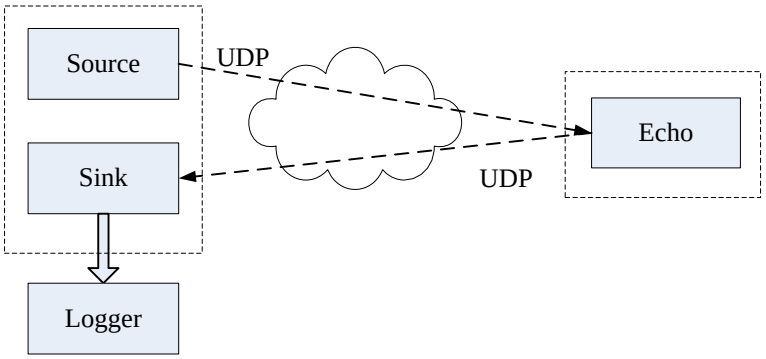


Figure 5.2 UDP/IP probe Tool in [4]

Description for network trace files

To simulate the transmission over the internet, we use three trace files from our database to obtain the network delay for each packet. These three trace files are typical ones presenting most features of the traces in our database for long distance VoIP applications. The trace files used in this work were collected from wired networks without involving any mobile networks.

Trace 1 was collected in January, 2009 between McGill University (Canada) and Shanghai Jiao Tong University (SHJTU) in China. SHJTU is inside the China Education Network. In Trace 1, packets with 320-byte payload were sent every 20ms for half an hour. The min/avg/max of round-trip delays is $305ms/308ms/442ms$, and $153ms/154ms/221ms$ for one-way delays accordingly. Figure 5.3 shows IP packet delay in Trace 1 and statistics (CDF and PDF) of the network delay.

Trace 2 is from Plymouth's trace file from Beijing University of Posts & Telecommunications (BUPT) in China (Northern China) to University of Plymouth (UoP) in UK. The size of the probe packets is set to 32 byte and the interval between successive packets is 30

2. Thank L. Sun and E. Ifeachor for providing the delay traces.

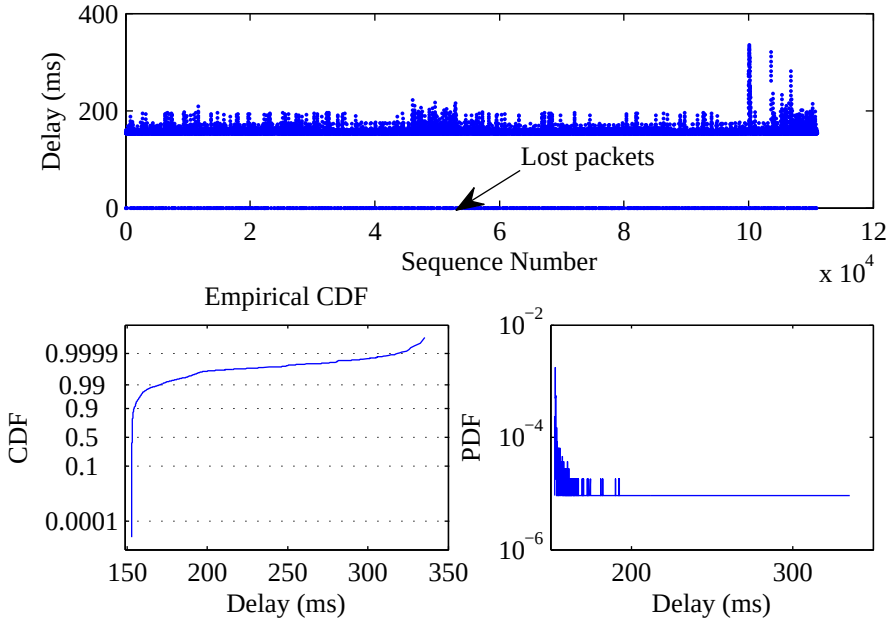


Figure 5.3 Trace 1: Network Delay from Canada to China ($153ms/154ms/221ms$).

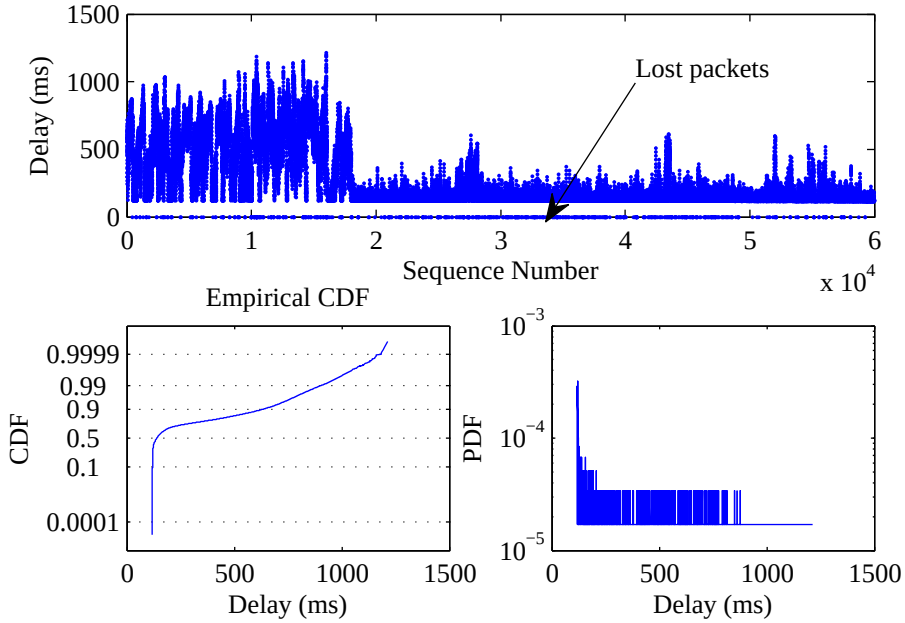


Figure 5.4 Trace 2: Network Delay from China to UK (from Plymouth's trace files).

ms. A linear regression method in [92] was used to calculate a drift rate and then removed the drift from the one-way trace data [91]. Figure 5.4 shows the collected data and its statistics (CDF and PDF).

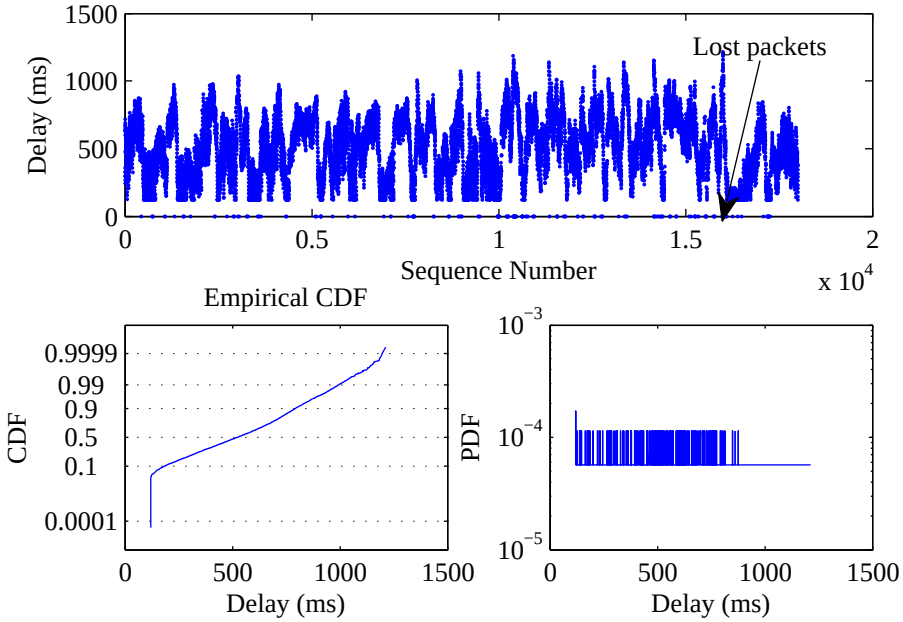


Figure 5.5 First Part of Trace 2.

The first part of Trace 2 , shown in Figure 5.5, is very different from the second part. This phenomena can happen in the current Internet. For example, one or more links were down during the first part of time and then became up during the second part of time. The Internet traffic got lighter due to more links available, and thus packets experienced less queueing delays on the way to their destinations. To give a better view of Trace 2, Figure 5.5 and Figure 5.6 are drawn to provide more details.

Trace 3 is also from Plymouth's trace files. It was collected and processed in the same way as Trace 2. The direction of Trace 3 is from UoP (UK) to BUPT (China). Figure 5.7 shows the distribution of the data, CDF and PDF of the delay.

The basic information of the three trace files is shown in Table 5.1.

5.1.2 Packet loss model

In Chapter 2, we have introduced a 2-state Gilbert Model for describing the loss occurrences. We simulate simple 2-state Gilbert Model using Matlab. The transition matrix is

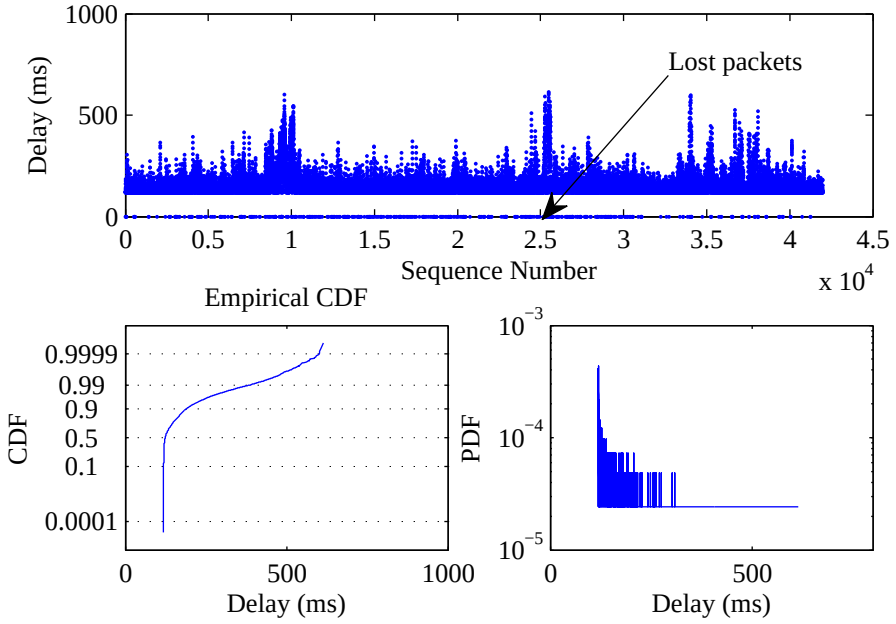


Figure 5.6 Second Part of Trace 2.

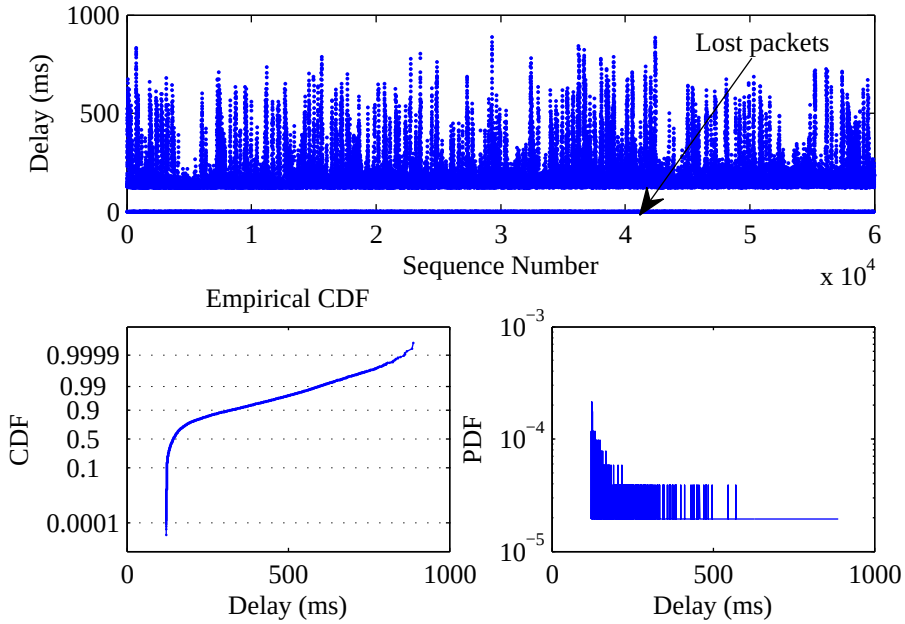


Figure 5.7 Trace 3: Network Delay from UK to China (from Plymouth's trace files) .

Table 5.1 Network Delay Traces

Trace	Min (ms)	Average (ms)	Max (ms)	Trace Description	Collection date
Trace1	153	154	221	McGill $\longrightarrow$ SHJTU	$2009-01-07$
Trace2	118	255	1212	UoP $\longrightarrow$ BUPT	$2002-06-11$
Trace3	122	186	888	BUPT $\longrightarrow$ UoP	$2002-06-07$

$\begin{vmatrix} 1-p & p \\ q & 1-q \end{vmatrix}$ with

– p as the transition possibility between a "0" (received) state to an "1" (lost) state

– q as the transition possibility between a "1" (lost) state to an "0" (received) state.

With specified packet loss rate and average burst length, the parameters p and q are set based on 2-state Gilbert Model (details in Section 2.3.3).

5.2 Experiments and results

In this section, we present two experiments. *Experiment 1* shows the efficacy of our quality-based playout buffering algorithm which is explained in Section 3.4 and improved quality of using new adaptive *media-dependent FEC* in Section 4.3. *Experiment 2* compares the performance of playout buffering algorithms with different schemes of sending redundancy information: adaptive *media-dependent FEC* in Section 4.3 and path diversity schemes in Section 4.4.1.

5.2.1 Experiment 1: Quality-based adaptive playout buffering

In *Experiment 1*, the loss model in "Network Simulator" generates network loss rate as 2% and the expected length of network burst loss is 2 packets. Performance is compared among five algorithms:

– Exponential-average (**Exp-Avg**) in [31] (Section 3.2.2), which estimates playout delay based on mean and variance of network delay. We set $\alpha = 0.998002$ as in [31].

– Fast exponential average (**Fast-Exp**) in [83] (Section 3.2.2), which is a modified **Exp-Avg**. It can adapt more quickly to short burst of packets incurring long delays [31]. We set $\alpha = 0.998002$ and $\beta = 0.75$ following [83].

- **Adaptive_IS** (Section 3.2.4): quality-based algorithm developed by Sun and Ifeachor in [5], which uses the E-Model R factor as cost function for optimization.
- **Adaptive_Gong** in Section 3.4, which is E-Model based playout buffering algorithm with two steps of reducing conversational delay.
- **Adaptive_FEC** in Section 4.3, which combines **Adaptive_Gong** with new *media-dependent FEC* in Section 4.3.

The results are shown in Table 5.2. In this table, the average packet loss rate (PLR) includes network loss(2%) and late packet loss caused by buffer underflow. Since all playout buffering algorithms use the same loss distribution for each trace file, the variation of the average PLR is mainly caused by late packet loss due to different playout buffering schemes. According to the definition in Section 2.3.2, conversational delay is majorly affected by network delay and the playout buffer depth. Besides these, we also observed that conversational delay is also affected if the first packet of the first talkspurt in a conversation turn is lost due to network loss or late arrival[3]. The reason is that the buffer adaption happens when the first packet of a talkspurt is received. If the first packet is lost, the playout buffer algorithm fails to set up predicted buffer delay to the following packets, and hence may cause the loss of the following packets. Suppose that the n-th packet is the first packet of the first talkspurt of a conversation turn and it is lost with other m packets during transmission. Hence, the $(n+m)$-th packet is the first packet in the talkspurt which arrives at the receiver side. Then the time interval $m \times 20$ ms needs to be counted into the conversational delay.

Based on our results, **Fast-Exp** gets relative high MOS-LQO score, but suffers from very a high conversational delay, especially in *Trace 3*. From our observation, the playout buffer provided by **Fast-Exp** is the longest. Note that the high conversation delay obtained in *Trace 2* and *Trace 3* is not acceptable according to ITU-T G.114 [21], in which the upper limit of end-to-end delay is 400 ms and hence the conversational delay should be no more than 800 ms.

Exp-Avg obtains lowest conversational delay in *Trace 1*, but the lowest quality because more late packets are dropped due to buffer underflow. In *Trace 3*, **Exp-Avg** gets relative high MOS-LQO score at expense of a large playout buffer delay, and hence the

3. In our experiments, we consider the first packet of a talkspurt is lost if it arrived later than when the playout buffer of the previous talkspurt is applied on this packet (the base delay plus buffer delay of the previous talkpurst).

conversational delay is higher than quality-based algorithms.

On the other hand, the quality-based techniques achieve a good balancing between packet losses and delays. In this experiment, **Adaptive_IS** performs better in MOS-LQO score than **Adaptive_Gong** but with longer conversational delays. One reason is because two steps (see Section 3.4 for details) to reduce conversational delays are used in **Adaptive_Gong** besides optimization of the E-Model. Another reason is that **Adaptive_IS** use Weibull distribution to estimate the delay distribution which results in a larger playout buffer than the one in **Adaptive_Gong**, which is estimated based on histogram. Applying *media-dependent FEC* to reconstruct missing packets, **Adaptive_FEC** obtains higher MOS-LQO scores than **Adaptive_IS** and **Adaptive_Gong** with the same conversational delay as **Adaptive_Gong**.

From the results in Table 5.2, **Adaptive_FEC** performs better than **Adaptive_Gong**, with the same conversational delay and higher MOS-LQO score. The reason is because the redundancy information is used to reconstruct the erased packets.

5.2.2 Experiment 2: Playout buffering with redundancy information

In this experiment, we compare six playout buffering algorithms as follows:

- **Adaptive_Gong**: E-Model based playout buffering algorithm with two steps of reducing conversational delay, which is described in Section 3.4.
- **Algorithm 1**: **Adaptive_Gong** with path diversity scheme 1 in Section 4.4.1.
- **Algorithm 2**: **Adaptive_Gong** with path diversity scheme 2 in Section 4.4.1.
- **Algorithm 3**: **Adaptive_Gong** with path diversity scheme 3 in Section 4.4.1.
- **Algorithm 4**: **Adaptive_Gong** with path diversity scheme 4 in Section 4.4.1.
- **Adaptive_FEC**: **Adaptive_Gong** with *media-dependent FEC* which is described in Section 4.3.

For path diversity case (two paths in this work), our "Network simulator" simulates these two paths by randomly selecting two pieces of delay data from trace files in Section 5.1.1. The parameters used in "Network simulator" are shown in Table 5.3 and Table 5.4. For **Adaptive_Gong** and **Adaptive_FEC**, which have no redundancy path, path 1 is used for transmission. Results are shown in Table 5.5. As in the last experiment, the average packet loss rate in Table 5.5 includes network loss(e.g.,5%) and late packet loss caused by buffer underflow.

Table 5.2 Performance Comparison of Playout Buffering Algorithms

Trace	Buffering algorithms	Average conversational delay (ms)	Average MOS-LQO	Average PLR (%)
Trace 1	Exp-Avg	336.2	3.41	6.3
	Fast-Exp	518.1	3.89	3.0
	Adaptive_IS	369.7	3.97	2.8
	Adaptive_Gong	345.1	3.92	3.4
	Adaptive_FEC	345.1	3.99	2.8
Trace 2	Exp-Avg	416.3	2.95	10.2
	Fast-Exp	989.6	3.77	3.1
	Adaptive_IS	444.7	3.34	6.5
	Adaptive_Gong	390.7	3.15	7.5
	Adaptive_FEC	390.7	3.42	5.6
Trace 3	Exp-Avg	506.4	3.52	6.1
	Fast-Exp	1437.5	3.67	4.5
	Adaptive_IS	369.0	3.52	6.1
	Adaptive_Gong	314.4	3.44	6.9
	Adaptive_FEC	314.4	3.60	5.3

Table 5.3 End-to-end Delay in Experiment 2

	path 1		path 2		
	source	statistics (min/avg/max)	source	statistics (min/avg/max)	correlation matrix
Simu1	Trace1	152.7/154.3/216.8	Trace1	152.7/154.5/214.5	$\begin{bmatrix} 0.86 & -0.02 \\ -0.01 & 1.16 \end{bmatrix}$
Simu2	Trace1	152.7/154.3/216.8	Trace2	118.9/138.7/394.2	$\begin{bmatrix} 0.16 & -0.02 \\ 0.02 & 6.23 \end{bmatrix}$
Simu3	Trace3	122.1/185.6/787.9	Trace3	122.9/142.6/602.5	$\begin{bmatrix} 2.25 & 0.15 \\ 0.15 & 0.44 \end{bmatrix}$
Simu4	Trace2	118.9/138.7/394.2	Trace3	122.1/185.6/787.9	$\begin{bmatrix} 0.27 & 0.03 \\ 0.03 & 3.65 \end{bmatrix}$

Table 5.4 Network Loss in Experiment 2

	path 1		path 2	
	packet loss rate	$E[BL]$	packet loss rate	$E[BL]$
Simu1	5%	5 packets	5%	5 packets
Simu2	5%	5 packets	5%	2 packets
Simu3	5%	5 packets	5%	5 packets
Simu4	5%	2 packets	5%	5 packets

Table 5.5 Performance Comparison of Playout Buffering Algorithms using redundancy

Trace	Buffering algorithms	Average conversational delay (ms)	Average MOS-LQO	Average PLR (%)
Simu1	**Algorithm 1**	310.2	3.28	5.9
	Algorithm 2	320.3	3.88	3.1
	Algorithm 3	319.0	3.87	3.1
	Algorithm 4	319.5	3.86	3.1
	Adaptive_FEC	340.3	2.78	10.3
	Adaptive_Gong	340.3	2.63	11.8
Simu2	**Algorithm 1**	299.9	3.89	2.3
	Algorithm 2	318.0	4.04	1.8
	Algorithm 3	306.5	4.04	1.8
	Algorithm 4	306.5	4.04	1.8
	Adaptive_FEC	346.6	3.41	5.6
	Adaptive_Gong	346.6	3.21	6.7
Simu3	**Algorithm 1**	320.0	3.75	4.1
	Algorithm 2	325.3	4.02	2.0
	Algorithm 3	322.8	4.02	2.0
	Algorithm 4	322.8	4.02	2.0
	Adaptive_FEC	330.2	3.02	8.3
	Adaptive_Gong	330.2	2.70	10.2
Simu4	Algorithm 1	283.2	3.96	2.1
	Algorithm 2	320.1	4.10	1.3
	Algorithm 3	288.6	4.10	1.3
	Algorithm 4	293.2	4.10	1.1
	Adaptive_FEC	323.8	2.10	15.7
	Adaptive_Gong	323.8	1.83	20.5

According to our results in Table 5.5, the algorithms with path diversity improve the perceived quality without increasing conversational delay. The results also show that path diversity schemes work better than the *media-dependent FEC* scheme for reducing packet loss.

Algorithm 1 achieves the lowest conversational delay because the minimum delay of two paths is used to design playout buffer, and hence the playout buffer size is shorter than other path diversity algorithms. For the same reason, the late packets are more likely to be dropped and accordingly PLR for speech packets is higher than other three path diversity algorithms.

Among four path diversity algorithms, **Algorithm 2** achieves highest MOS-LQO score for all experiments with relatively high conversational delay. Since the second path (path 2) is used only for provide redundancy information, **Algorithm 2** provides much improvement on quality by reducing network packet loss, resulting high MOS-LQO score. Since the size of the playout buffer, d_{static}, is estimated from primary path (path 1), the conversational delay of **Algorithm 2** is mainly depends on the end-to-end delay on path 1.

In Table 5.5, the MOS-LQO scores achieved by **Algorithm 3** and **Algorithm 4** is slightly lower than (*Simu1*) or as high as (*Simu2* – *Simu4*) that of **Algorithm 2**. However, these two algorithms obtain lower conversational delays than **Algorithm 2** because **Algorithm 3** and **Algorithm 4** consider estimated quality (R factor) on both paths instead of on path 1 in **Algorithm 2**.

In *Simu1* and *Simu2*, the conversational delays of **Adaptive_FEC** and **Adaptive_Gong** are much higher than path diversity algorithms. The main reason is that several packets at the beginning of the first talkspurt are missing due to network loss.

5.3 Summary

For conversational VoIP, conversational quality is the most important for playout buffering design. As we discussed in previous chapters, conversational quality includes perceived quality and interactivity which is measured by conversational delay. In this chapter, we present efficacy of our playout algorithms by performance comparison. The result of Experiment 1 shows that our E-Model based playout buffering algorithm (**Adaptive_Gong**) not only keep high MOS-LQO value (high perceived quality), but also obtains low conversational delay (high interactivity). Experiment 1 also shows that perceived quality can be

improved by redundancy information using adaptive *media-dependent FEC*. The adaptive *media-dependent FEC* scheme can reduce PLR (packet loss rate) and shorten the length of burst loss.

Path diversity is alternative methodology to provide redundancy information. In Experiment 2, we compared the performance among different path diversity schemes together with adaptive *media-dependent FEC* scheme. The results show that path diversity schemes achieve higher performances than adaptive *media-dependent FEC* scheme.

Chapter 6

Conclusions and Future Work

Playout buffering plays an very important role in VoIP applications. A poor design of playout buffering impacts the perceived quality. In this work, we have presented several playout algorithms based on preserving perceived conversational quality. In our schemes, perceived conversational quality involves both voice quality and interactivity.

The sender-driven technologies, e.g., Forward Error Correction (FEC) and path diversity, are used to achieve better QoS.

In this chapter, we will review our algorithms in Section 6.1. The limitations of this work are discussed in Section 6.2. The conclusions are drawn in Section 6.3. Our future research topics would be discussed in Section 6.4.

6.1 Review of the work

In this work, our goal is to design playout buffering algorithms with improved perceived conversational quality. In Section 3.4, we presented our quality-based adaptive algorithm which takes conversational delay into account. In our design, the E-Model R factor is used as a cost index to estimate static playout buffer delay for each talkspurt. Even though the use of the E-Model can guarantee a proper balance between missing packets and end-to-end delay, conversational delay can be reduced further. Two steps are taken to lessen the conversational delay by tuning the playout buffer:

– at the beginning of a talkspurt, a short playout buffer is gradually increased by stretching received speech in the voice packets. Note that the first packet is played out immediately after stretched.

– at the end of a talkspurt (when "hangover" is detected), the playout buffer is gradually decreased by compressing the packets in buffer.

Thus, the beginning part and the ending part of a talkspurt have no/little buffer delays, and these buffer delays contribute to conversational delay. Therefore, the algorithm obtains less conversational delay than other E-Model based adaptive playout buffering algorithms. The start of a talkspurt is determined by utilizing the information in RTP header (M field). In this work, we define the end of a talkspurt as when the first "hangover" frame is detected. Most existing speech codecs have built-in voice activity detection (VAD) algorithms which can be used for "hangover" detection. Therefore, it is easy to incorporate our algorithm with speech codecs with VAD.

Even though protecting most voice packets during a talkspurt, our adaptive playout buffering algorithm is still sensitive to burst losses. In Section 4.5, Fig. 4.14 shows that the perceived voice quality measured by the PESQ goes down when the expected burst length increases. This fact motivated us to incorporate our adaptive playout buffering scheme with redundancy information to alleviate the effect of burst losses. Since large conversational delay degrades perceived conversational quality, it is desirable that the utilization of redundancy information does not introduce extra delay. In Section 4.3.1, we presented a new *media-dependent FEC* scheme which introduces no additional delay when used with playout buffering algorithms. The idea is that the sender gradually increases redundancy packets based on the size of the playout buffer at the receiver. The buffer delay at the receiver side is used to wait for redundancy packets.

Forward error correction (FEC) schemes are very efficient to compensate for isolated missing packets and shorten the length of burst losses. However, FEC fails to work in case of a long gap (see Fig. 4.14). Path diversity is an alternative to provide redundancy using multiple paths between the conversation parties. In this work, we introduced our new path diversity schemes which are based on the E-Model R factors calculated on both paths. In Section 4.4, we presented four path diversity schemes which are used with our adaptive playout buffering algorithm. If network packet loss and delay distribution are largely independent between two paths, path diversity schemes are robust to both isolate and burst losses.

6.2 Limitations

The limitations of this work are as follows

– Limited trace data collection

The Internet trace files used in this work are of limited duration and not up to date. The latest trace file was collected in 2009, from Canada to China. To reflect the network characteristics, the trace data should cover more local, continental, and international links with various transmission mediums, e.g., modem, ADSL links, cable links and different ISPs. The trace data should be collected periodically, for example, every two or three months, to track the changes of networks [91].

– Simple form E-Model

In our algorithms, the E-Model R factor is used to estimate static playout delay for a talkspurt. Recalling Equation (4.1), R is expressed as

$$\begin{aligned} R &= 93.2 - I_d - I_e \\ &= (93.2 - I_{ec}) - (I_d + I_\rho). \end{aligned} \tag{6.1}$$

In this equation, only the most important impairments considered mainly stem from IP networks (e.g., packet missing, delay and delay jitter), and speech coding (e.g., codec). The impairments relevant in hybrid networks, e.g., echo, sidetone, background noise, double talk, are not taken into account. These impairments exist in practical VoIP applications and have impact on perceived quality. We will discuss a possible solution in Section 6.4, which would be one of our future focuses.

– perceived conversational quality assessment

In this work, perceived conversational quality is evaluated by a objective measurement – PESQ and conversational delay. More accurate measurement can be achieved by a subjective test, in which a group of people are asked to hear and rank the conversational quality at the receiver's side.

Besides, the impact of PWSOLA is not involved in the perceived conversational quality. A subjective quality test is needed to measure the human's attitude to the stretched/compressed speech.

6.3 Conclusions

In Chapter 5, the results of our experiments show the efficiency of our playout buffering algorithms. The main conclusion of this work is as following

> Even though E-Model based playout buffering algorithms achieve optimal balance between packet missing and delay, perceived conversational quality can be further improved by
> – two steps of reducing conversational delay (Section 3.4, Section 6.1)
> – redundancy information provided by sender's side (FEC and path diversity schemes)

Through the experiments, our adaptive playout buffering algorithm achieves shortened conversational delay and at the same time, protects most voice packets with enough buffer delay to achieve required quality. Sender-driven repair schemes, e.g., FEC and path diversity schemes, are also proven to be efficient at mitigating packet missing, especially burst loss, by our experiments.

The novelty of this work is in

– a new adaptive quality-based playout buffering algorithm
– a method to calculate conversational delay
– a new *media-dependent FEC* scheme
– two new path diversity schemes
– playout buffering algorithms with redundancy schemes (*media-dependent FEC* scheme and three path diversity schemes)

All these algorithms are developed based on published standards or protocols. Therefore, they can be directly applied in VoIP applications.

6.4 Future focuses

Our future research would be focused on the following topics

1. Measuring the impact of stretching/compressing processes on the perceived quality

   We will set up a subjective test to measure the impact of PWSOLA on perceived quality. With the MOS scores obtained from the test, we will find some mappings like quality vs. length of processed speech, quality vs. stretching/compressing rate, etc.

2. Extended E-Model

   In this work, sender-driven repair techniques, e.g., FEC, path diversity, are used to alleviate the impact of burst loss. In [93], another solution dealing with burst loss was proposed using extended E-Model. In [93], the E-Model is extended by incorporating ETSI Tiphon [94] so as to intergrade the effects of burst loss on the perceived quality. According to [93], I_e in Equation (6.1) is extended as

$$I_e = I_e(av) + (k \cdot (I_1 - I_e(av))) \cdot \exp(-y/t_3), \tag{6.2}$$

   where
   - I_1 is the exist value of the equipment impairment factor from the last burst
   - k is a constant value typically set to 0.7
   - y is the time interval in seconds since the last burst
   - t_3 is exponential time constant representing the user memory about the last occur burst event
   - $I_e(av)$ is the average distortion

   The average distortion $I_e(av)$ is calculated by

$$I_{av} = \frac{b \cdot I_{eb} + g \cdot I_{eg} - t_1 \cdot (I_{eb} - I_{2g})(1 - e^{-b/t_1}) + t_2 \cdot (I_{1b} - I_{eg})(1 - e^{-b/t_2})}{b + g}, \tag{6.3}$$

   which
   - g is average gap
   - b is burst duration (in sec)
   - I_{eg} is impairment of the gap
   - I_{eb} is impairment of the gap
   - t_1 and t_2 are two smoothing parameters
   - I_{2g} is the impairment level perceived at the end of the gap
   - I_{1b} is the impairment level at the burst periods

   Our future research would involve this extended form of the E-Model to estimate static buffer size for each talkspurt.

3. Partial redundancy

   In this work, the redundancy packets in path diversity schemes are 100% copy of original packets, i.e., full redundancy. The problem of full redundancy schemes is high

bit rates. In partial redundancy schemes, only important packets are transmitted to receiver via a second path. Thus, the overall bit rate of a partial redundancy scheme is lower than that of a full redundancy scheme. Many algorithms have been proposed in literature to mark important packets, e.g., [38], [95].

Overall bit rate can be further reduced if important packets are encoded by using low-bit-rate algorithms, e.g., G.723.1, AMR, etc. For example, G.711, a high quality encoder, is used on the primary path, and AMR is used to encode the important packets on the secondary path. The challenging part of this design is how to consistently update LP parameters for decoding. That would be our next step of research.

4. Echo cancelation

   In this thesis, we assume a VoIP system with ideal echo cancelation. Hence, we do not take any special steps to reduce echo effects. Echo is the phenomenon that delayed voice of VoIP users is reflected back to their ears via the handset or headset speakers. Echo is in fact a transmission impairment, and the impact of echo on perceived quality depends on the loudness of the voice and how many milliseconds the voice is delayed. For example, an echo with a few milliseconds is bearable whereas the conversation might be interrupted if the voice is delayed by several hundred milliseconds. Echo is a severe distraction if the round trip delay is longer than 30–40ms [96]. In VoIP applications, the loudness of echo is no worse than in PSTN. However, the overall latency introduced by VoIP system, including transmission delay of IP networks, packetization intervals and jitter buffers,etc, is much higher than that of traditional telecommunication systems, and accordingly makes echo more noticeable to users.

   Echo in VoIP can be introduced by network echo and acoustic echo. Network echo is the dominant source of echo in telephone networks [12, Chapter 15]. It is mainly caused by unbalance hybrid circuits in PSTN-VoIP gateways, where the signal coming into the two wire side (local line) is sent back on the return path of the the four wire side (long-hual trunk).

   Another source of echo is acoustic echo, which is a nonlinear combination of multiple reflections of the speaker's sound back to the microphone through different paths, e.g., from walls, floor, ceiling, etc. It is mainly caused by poor acoustic isolation between the speaker and microphone of a speaker's device, e.g., handset, headset, IP softphone, speakerphone.

   Echo cancelation algorithms eliminate the echo by correctly removing a portion of the

transmitted signal from the return signal. To achieve good voice quality, accurate estimation of echo is required. It is a challenging research area and would be our future focus.

5. VoIP over wireless local area network (WLAN)

   In the past few years, WLAN has become increasingly popular for Internet access in proximity of an access point (AP). The main advantages of WLAN are its simplicity, flexibility and cost effectiveness [97]. The combination of WLAN and VoIP has been proved to be successful to provide wireless voice service with cost efficiency.

   However, wireless networks are harder to manage than wired networks. Jitter and packet missing are significantly higher in WLAN than in a wired network. Therefore, more effort is needed to obtain high perceived quality for VoIP over WLAN. Approved by the IEEE in late 2005, 802.11e defines a set of QoS mechanisms for WLAN through alterations to the media access control (MAC) layer. It also provides essential services to support delay-sensitive and bandwidth-sensitive applications applications, such as streaming video and VoIP.

   In 802.11e, traffic is classified as 8 categories with different priority for transmission (see Table 6.1 for details).

Table 6.1 QoS Level in 802.11e

Priority	Access Category	Designation
0	0	Best Effort
1	0	Best Effort
2	0	Best Effort
3	1	Video Probe
4	2	Video
5	2	Video
6	3	Voice
7	3	Voice

   Our adaptive playout buffering algorithm (Section 3.4) can be definitely used in VoIP over WLAN, because it works at the application layer without any assumption on physical and MAC layers. The interesting part is that we can assign higher priority

for the packets with dynamic playout buffer (upside and downside in Figure 3.2) to reduce the chance of packet missing. These packets, especially the first few packets of a talkspurt, are somehow important to the receiver. For example, if the first packet of a talkspurt is lost, the stretching process to increase playout buffer depth cannot be trigger, resulting in the missing of the following packets. Therefore, it is promising to achieve improved quality when our algorithm incorporates with 802.11e.

Appendix A

G.729 VAD

In this appendix, we introduce voice active detection (VAD) algorithm ITU-T G.729. In this work, we use this algorithm to detect "hangover" packets, which trigger the compression processing in our playout buffering algorithms. All the following sections are from ITU-T G.729 Annex B [41].

A.1 General description of the VAD algorithm

The VAD algorithm makes a voice activity decision every 10 ms in accordance with the frame size of the G.729 speech coder. A set of difference parameters is extracted and used for an initial decision. The parameters are the full-band energy, the low-band energy, the zero-crossing rate and a spectral measure. The long-term averages of the parameters during non-active voice segments follow the changing nature of the background noise. A set of differential parameters is obtained at each frame. These are a difference measure between each parameter and its respective long-term average. The initial voice activity decision is obtained using a piecewise linear decision boundary between each pair of differential parameters. A final voice activity decision is obtained by smoothing the initial decision.

The output of the VAD module is either 1 or 0, indicating the presence or absence of voice activity respectively. If the VAD output is 1, the G.729 speech codec is invoked to code/decode the active voice frames. However, if the VAD output is 0, the DTX/CNG algorithms described herein are used to code/decode the non-active voice frames. Traditional speech coders and decoders use comfort noise to simulate the background noise in the non-active voice frames. If the background noise is not stationary, a mere comfort noise

insertion does not provide the naturalness of the original background noise. Therefore it is desirable to intermittently send some information about the background noise in order to obtain a better quality when non-active voice frames are detected. The coding efficiency of the non-active voice frames can be achieved by coding the energy of the frame and its spectrum with as few as fifteen bits. These bits are not automatically transmitted whenever there is a non-active voice detection. Rather, the bits are transmitted only when an appreciable change has been detected with respect to the last transmitted non-active voice frame.

At the decoder side, the received bit stream is decoded. If the VAD output is 1, the G.729 decoder is invoked to synthesize the reconstructed active voice frames. If the VAD output is 0, the CNG module is called to reproduce the non-active voice frames.

A.2 Detailed description of the VAD algorithm

A flowchart of the VAD operation is given in Figure A.1. The VAD operates on frames of digitized speech. The frames are processed in time order and are consecutively numbered from the beginning of each conversation/recording.

At the first stage, four parametric features are extracted from the input signal. Extraction of the parameters is shared with the active voice encoder module and the non-active voice encoder for computational efficiency. The parameters are the full- and low-band frame energies, the set of line spectral frequencies (LSF) and the frame zero crossing rate.

If the frame number is less than N_i, an initialization stage of the long-term averages takes place, and the voice activity decision is forced to 1 if the frame energy from the LPC analysis is above 15 dB (see Equation (A.1)). Otherwise, the voice activity decision is forced to 0. If the frame number is equal to N_i, an initialization stage for the characteristic energies of the background noise occurs.

At the next stage, a set of difference parameters are calculated. This set is generated as a difference measure between the current frame parameters and running averages of the background noise characteristics. Four difference measures are calculated:

- a spectral distortion;
- an energy difference;
- a low-band energy difference; and
- a zero-crossing difference.

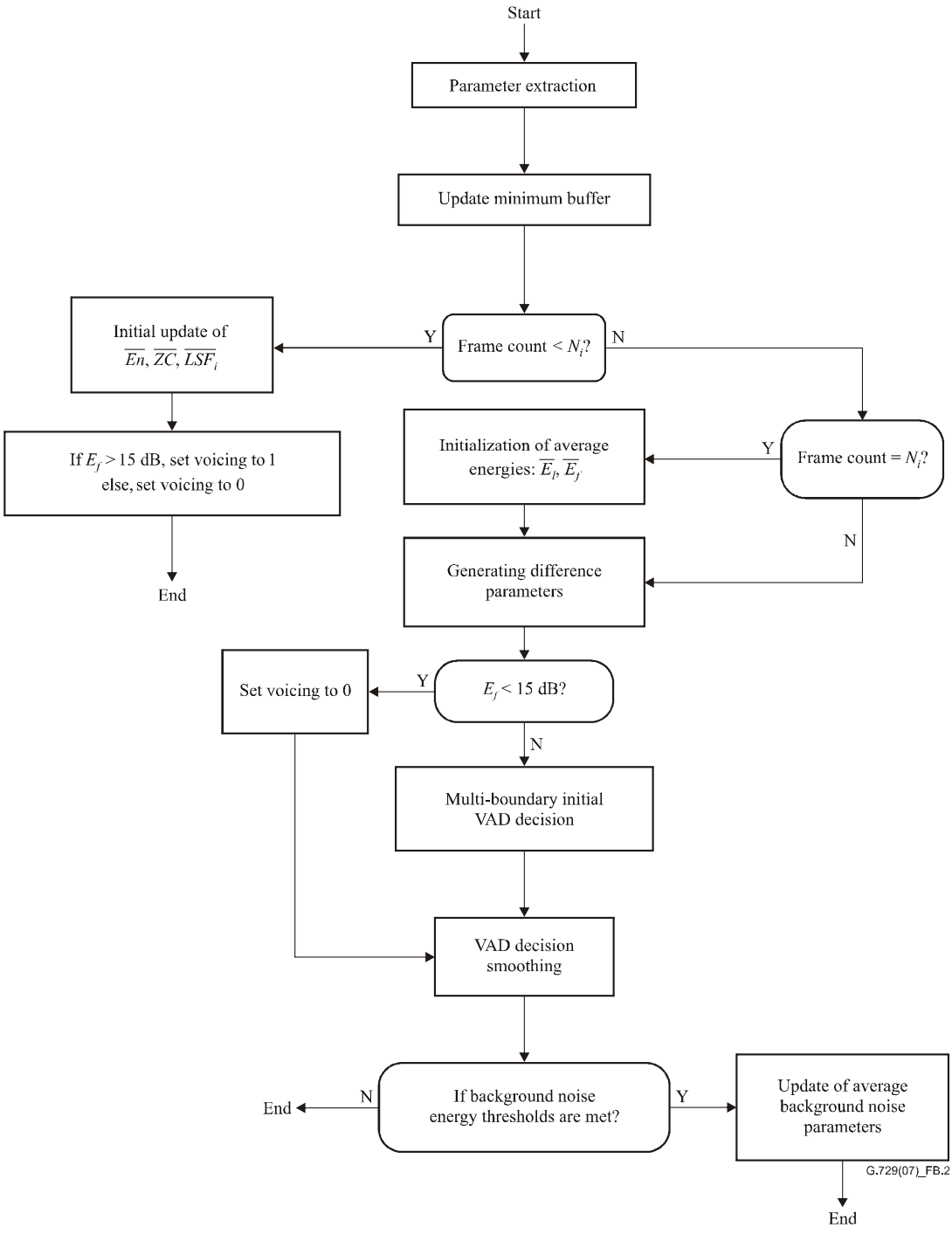


Figure A.1 VAD flowchart

The initial voice activity decision is made at the next stage, using multi-boundary decision regions in the space of the four difference measures. The active voice decision is given as the union of the decision regions and the non-active voice decision is its complementary logical decision. Energy considerations, together with neighboring past frames decisions, are used for decision smoothing.

The running averages have to be updated only in the presence of background noise, and not in the presence of speech. An adaptive threshold is tested, and the update takes place only if the threshold criterion is met.

A.2.1 Parameter extraction

For each frame, a set of parameters is extracted from the speech signal. The parameters extraction module can be shared between the VAD, the active voice encoder and the non-active voice encoder. The basic set of parameters is the set of autocorrelation coefficients, which is derived similarly to the full version of G.729 (see clause 3.2.1). The set of autocorrelation coefficients will be denoted by:

$$R(i)_{i=0}^{q}, \text{ where } q = 12.$$

Line spectral frequencies (LSF)

A set of linear prediction coefficients is derived from the autocorrelation and a set of $LSF_{i=1}^{p}$, where $p = 10$, is derived from the set of linear prediction coefficients.

Full-band energy

The full-band energy E_f is the logarithm of the normalized first autocorrelation coefficient $R(0)$:

$$E_f = 10 \cdot \log_{10} \left[\frac{1}{N} R(0) \right], \tag{A.1}$$

where $N = 240$ is the LPC analysis window size in speech samples.

Low-band energy

The low-band energy E_l measured on 0 to F_lHz band, is computed as follows:

$$E_l = 10 \cdot \log_{10} \left[\frac{1}{N} \mathbf{h^T R h} \right], \tag{A.2}$$

where $\mathbf{h}$ is the impulse response of an FIR filter with cut-off frequency at F_lHz, $\mathbf{R}$ is the Toeplitz autocorrelation matrix with the autocorrelation coefficients on each diagonal.

Zero-crossing rate

Normalized zero-crossing rate ZC for each frame is calculated by:

$$ZC = \frac{1}{2M} \sum_{i=0}^{M-1} \left[|sgn[x(i)] - sgn[x(i-1)]| \right], \tag{A.3}$$

where $x(i)$ is the preprocessed input signal and $M = 80$.

A.2.2 Initialization of the running averages of the background noise characteristics

For the first N_i frames, the spectral parameters of the background noise, denoted by $\overline{LSF}^p_{i i=1}$ are initialized as an average of the $LSF^p_{i i=1}$ of the frames. The average of the background noise zero-crossings, denoted by $\overline{ZC}$ is initialized as an average of the zero-crossing rate ZC of the frames. The running averages of the background noise energy, denoted by $\overline{E_f}$, and the background noise low-band energy, denoted by $\overline{E_l}$ are initialized as follows. First, the initialization procedure uses $\overline{E_n}$, defined as the average of the frame energy E_f over the first N_i frames. These three averaging ($\overline{E_n}$, $\overline{ZC}$, and $\overline{LSF}^p_{i i=1}$) include only the frames that have an energy E greater than 15 dB. Second, the initialization procedure continues as follows:

if $\overline{E_n} \leq T1$ then
  $\overline{E_f} = \overline{E_n} + K0$
  $\overline{E_l} = \overline{E_n} + K1$
elseif $T1 < \overline{E_n} < T2$ then
  $\overline{E_f} = \overline{E_n} + K2$

$\overline{E_l} = \overline{E_n} + K3$

else

$\overline{E_f} = \overline{E_n} + K4$

$\overline{E_l} = \overline{E_n} + K5$

See Table A.1 for constant values.

A.2.3 Generating the long-term minimum energy

A long-term minimum energy parameter, E_{min}, is calculated as the minimum of E_f over N_0 previous frames. Since N_0 is relatively large, E_{min} is calculated using stored values of the minimum of E_f over short segments of the past.

A.2.4 Generating the difference parameters

Four difference measures are generated from the current frame parameters and the running averages of the background noise.

The spectral distortion ΔS

The spectral distortion measure is generated as the sum of squares of the difference between the current frame $LSF_{i=1}^{p}$ vector and the running averages of the background noise $\overline{LSF}_{i=1}^{p}$:

$$\Delta S = \sum_{i=1}^{p} (LSF_i - \overline{LSF_i}). \tag{A.4}$$

The full-band energy difference ΔE_f

The full-band energy difference measure is generated as the difference between the current frame energy, E_f, and the running average of the background noise energy, $\overline{E_f}$:

$$\Delta E_f = \overline{E_f} - E_f. \tag{A.5}$$

The low-band energy difference ΔE_l

The low-band energy difference measure is generated as the difference between the current frame low-band energy, E_l, and the running average of the background noise low-

band energy, $\overline{E_l}$:

$$\Delta E_l = \overline{E_l} - E_l. \tag{A.6}$$

The zero-crossing difference ΔZC

The zero-crossing difference measure is generated as the difference between the current frame zero-crossing rate, ZC, and the running average of the background noise zero-crossing rate, $\overline{ZC}$:

$$\Delta ZC = \overline{ZC} - ZC. \tag{A.7}$$

Multi-boundary initial voice activity decision

The initial voice activity decision is denoted by I_{VD}, and is set to 0 ("FALSE") if the vector of difference parameters lies within the non-active voice region. Otherwise, the initial voice activity decision is set to 1 ("TRUE"). The fourteen boundary decisions in the four-dimensional space are defined as follows:

1. if $\Delta S > a_1 \cdot \Delta ZC + b_1$ then $I_{VD} = 1$
2. if $\Delta S > a_2 \cdot \Delta ZC + b_2$ then $I_{VD} = 1$
3. if $\Delta E_f < a_3 \cdot \Delta ZC + b_3$ then $I_{VD} = 1$
4. if $\Delta E_f < a_4 \cdot \Delta ZC + b_4$ then $I_{VD} = 1$
5. if $\Delta E_f < b_5$ then $I_{VD} = 1$
6. if $\Delta E_f < a_6 \cdot \Delta ZC + b_6$ then $I_{VD} = 1$
7. if $\Delta S > b_7$ then $I_{VD} = 1$
8. if $\Delta E_f < a_8 \cdot \Delta ZC + b_8$ then $I_{VD} = 1$
9. if $\Delta E_f < a_9 \cdot \Delta ZC + b_9$ then $I_{VD} = 1$
10. if $\Delta E_f < b_{10}$ then $I_{VD} = 1$
11. if $\Delta E_l < a_{11} \cdot \Delta ZC + b_{11}$ then $I_{VD} = 1$
12. if $\Delta E_l > a_{12} \cdot \Delta E_f + b_{12}$ then $I_{VD} = 1$
13. if $\Delta E_l > a_{13} \cdot \Delta E_f + b_{13}$ then $I_{VD} = 1$
14. if $\Delta E_l > a_{14} \cdot \Delta E_f + b_{14}$ then $I_{VD} = 1$

If none of the fourteen conditions is "TRUE" $I_{VD} = 0$. See Table A.1 for constant values.

Table A.1 G.729 VAD – Table of Constants

Name	Constant	Name	Constant
N_i	32	N_1	4
N_0	128	N_2	10
K_0	0	T_1	671088640
K_1	−53687091	T_2	738197504
K_2	−67108864	T_3	26843546
K_3	−93952410	T_4	40265318
K_4	−134217728	T_5	40265318
K_5	−161061274	T_6	40265318
a_1	23488	b_1	28521
a_2	−30504	b_2	19446
a_3	−32768	b_3	−32768
a_4	26214	b_4	−19661
a_5	0	b_5	−30802
a_6	28160	b_6	−19661
a_7	0	b_7	30199
a_8	16384	b_8	−22938
a_9	−19065	b_9	−31576
a_{10}	0	b_{10}	−17367
a_{11}	22400	b_{11}	−27034
a_{12}	30427	b_{12}	29959
a_{13}	−24576	b_{13}	−29491
a_{14}	23406	b_{14}	−28087

A.2.5 Voice activity decision smoothing

The initial voice activity decision is smoothed (hangover) to reflect the long-term stationarity nature of the speech signal. The smoothing is done in four stages.

A flag indicating that hangover has occurred is defined as v_flag. It is set to zero each time before the voice activity decision smoothing is performed. Denote the smoothed voice activity decision of the frame, the previous frame and frame before the previous frame by S_{VD}^{0}, S_{VD}^{-1} and S_{VD}^{-2}, respectively. S_{VD}^{-1} is initialized to 1, and S_{VD}^{-2} is initialized to 1. For start $S_{VD}^{0} = I_{VD}$. The first smoothing stage is:

$$\text{if } (I_{VD} = 0) \text{ and } (S_{VD}^{-1} = 1) \text{ and } (E > \overline{E_f} + T_3) \text{ then } S_{VD}^{0} = 1 \text{ and } v\_flag = 1$$

For the second smoothing stage define a Boolean parameter F_{VD}^{-1} and a smoothing counter C_e. F_{VD}^{-1} is initialized to 1 and C_e is initialized to 0. Denote the energy of the previous frame by E_{-1}. The second smoothing stage is:

if $(F_{VD}^{-1} = 1)$ and $(I_{VD} = 0)$ and $(S_{VD}^{-1} = 1)$ and $(|E_f - E{-}1| \leq T_4)$ then
  $S_{VD}^{0} = 1$
  $v\_flag = 1$
  $C_e = C_e + 1$
  if $(C_e \leq N_1)$ then
    $F_{VD}^{-1} = 1$
  else
    $F_{VD}^{-1} = 0$
    $C_e = 0$
else
$F_{VD}^{-1} = 1$

For the third smoothing stage define a noise continuity counter C_s, which is initialized to 0. If $S_{VD}^{0} = 0$, then C_s is incremented. The third smoothing stage is:

if $(S_{VD}^{0} = 1)$ and $(C_s > N_2)$ and $(|E_f - E{-}1| \leq T_5)$ then
  $S_{VD}^{0} = 0$
  $C_s = 0$
if $(S_{VD}^{0} = 1)$ then $C_s = 0$

In the fourth stage, a voice activity decision is made if the following condition is satisfied:

if $(E_f < \overline{E_f} + T_6)$ and $(frm_count > N_0)$ and $(v\_flag = 0)$ then

$S_{VD}^{0} = 0$

A.2.6 Updating the running averages of the background noise characteristics

The running averages of the background noise characteristics are updated at the last stage of the VAD module. At this stage, the following condition is tested and the updating takes place if the following condition is met:

if $(E_f < \overline{E_f} + T_6)$ then update

The running averages of the background noise characteristics are updated using a first order auto-regressive (AR) scheme. Different AR coefficients are used for different parameters, and different sets of coefficients are used at the beginning of the recording/conversation or when a large change of the noise characteristics is detected.

Let β_{E_f} be the AR coefficient for the update of $\overline{E_f}$, β_{E_l} be the AR coefficient for the update of $\overline{E_l}$, β_{ZC} be the AR coefficient for the update of $\overline{ZC}$ and β_{LSF} be the AR coefficient for the update of $\overline{LSF_{i}}_{i=1}^{p}$. The total number of frames where the update condition was satisfied is counted by C_n. Different set of the coefficients β_{E_f}, β_{E_l}, β_{ZC} and β_{LSF} is used according to the value of C_n

The AR update is done according to:

$$\begin{aligned}
\overline{E_f} &= \beta_{E_f} \cdot \overline{E_f} + (1 - \beta_{E_f}) \cdot E_f \\
\overline{E_l} &= \beta_{E_l} \cdot \overline{E_l} + (1 - \beta_{E_l}) \cdot E_l \\
\overline{ZC} &= \beta_{ZC} \cdot \overline{ZC} + (1 - \beta_{ZC}) \cdot ZC \\
\overline{LSF_i} &= \beta_{LSF} \cdot \overline{LSF_i} + (1 - \beta_{LSF}) \cdot LSF_i
\end{aligned} \tag{A.8}$$

$\overline{E_f}$ and C_n are further updated according to:

if $(frm_count > N_0)$ and $(\overline{E_f} < E_{min})$ then
$\overline{E_f} = E_{min}$
$C_n = 0$

Appendix B

E-Model: Advantage Factor, A

Due to the specified meaning of the advantage factor A, there is – consequently – no relation to all other transmission parameters [1]. Some provisional values are given in Table B.1.

Table B.1 G.107-Provisional Examples for the Advantage Factor A [1]

Communication system example	Maximum value of A
Conventional (wirebound)	0
Mobility by cellular networks in a building	5
Mobility in a geographical area or moving in a vehicle	10
Access to hard-to-reach locations, e.g., via multi-hop satellite connections	20

Appendix C

E-Model: Default Values and Permitted Ranges

For all input parameters used in the algorithm of the E-model, the default values are listed in Table C.1. It is strongly recommended to use these default values for all parameters which are not varied during planning calculation. If all parameters are set to the default values, the calculation results in a very high quality with a rating factor of $R = 93.2$. [1]

Table C.1 G.107– Default Values and Permitted Ranges [1]

Parameter	Abbr.	Unit	Default value	Permitted range	Remark
Send Loudness Rating	SLR	dB	+8	0.... + 18	(Note 1)
Receive Loudness Rating	RLR	dB	+2	−5... + 14	(Note 1)
Sidetone Masking Rating	STMR	dB	15	10...20	(Note 2)
Listener Sidetone Rating	LSTR	dB	18	13...23	(Note 2)
D-Value of Telephone, Send Side	Ds	–	3	−3... + 3	(Note 2)
D-Value of Telephone, Receiver Side	Dr	–	3	−3... + 3	(Note 2)
Talker Echo Loudness Rating	TELR	dB	65	5...65	
Weighted Echo Path Loss	WEPL	dB	110	5...110	
Mean one-way Delay of the Echo Path	T	ms	0	0...500	
Round-Trip Delay in a 4-wire Loop	Tr	ms	0	0...1000	
Absolute Delay in echo-free Connections	Ta	ms	0	0...500	
Number of Quantization Distortion Units	qdu	–	1	1...14	
Equipment Impairment Factor	Ie	–	0	0...40	
Packet-loss Robustness Factor	Bpl	–	1	1...40	(Note 3)
Random Packet-loss Probability	Ppl	%	0	0...20	(Note 3)
Burst Ratio	BurstR	–	1	1...2	(Note 3)
Circuit Noise referred to 0 dBr-point	Nc	dBm0p	−70	−80... − 40	
Noise Floor at the Receive Side	Nfor	dBmp	−64	–	(Note 3)
Room Noise at the Send Side	Ps	dB(A)	35	35...85	
Room Noise at the Receive Side	Ps	dB(A)	35	35...85	
Advantage Factor	A	–	035	0...20	

Note 1 – Total values between microphone or receiver and 0 dBr-point

Note 2 – $Fixedrelation : LSTR = STMR + D$

Note 3 – Currently under study

Bibliography

[1] ITU-T, *Recomendation G.107: The E-Model, A Computational Model for Use in Transmission Planning*. ITU Recomendation G.107, Mar. 2003.

[2] M. Lee, J. McGowan, and M. C. Recchione, "Enabling Wireless VoIP," *Bell Labs Technical Journal*, vol. 11, pp. 201–215, Nov. 2007.

[3] ITU-T, *Recomendation G.113: Transmission Impairments due to Speech Processing*. ITU Recomendation G.113, Feb. 2001.

[4] W. Jiang, "UDP trace tool." on Website, 2002. `http://www.cs.columbia.edu/~wenyu/`, Retrieved 2012-04-06.

[5] L. Sun and E. Ifeachor, "New Models for Perceived Voice Quality Prediction and Their Applications in Playout Buffer Optimization for VoIP Networks," in *Proc. IEEE ICC*, (Paris, France), pp. 1478–1483, June 2004.

[6] Speech Codecs: Pros & Cons, "Understanding Various Speech Codecs." on Website. `http://speechcodecs.wordpress.com`, Retrieved 2012-04-06.

[7] Sandvine, "Using Network Intelligence to Provide Carrier-grade VoIP (White Paper)." on Website, 2005. `http://www.sandvine.com`, Retrieved 2005-05-31.

[8] Choose Your VoIP, "Cost-Saving Advantages of VoIP Service." on Website. `http://www.chooseyourvoip.com/articles/cost-advantages-of-voip`, Retrieved 2012-04-06.

[9] S. Ganguly and S. Bhatnagar, *VoIP: wireless, P2P and New Enterprise Voice Over IP*. John Wiley & Son Ltd., 2008.

[10] Wikipedia, "Internet Protocol." on Website. `http://en.wikipedia.org/wiki/Internet_Protocol`, Retrieved 2012-04-06.

[11] Continuity Central, "VOIP – Vulnerability over Internet Protocol." on Website. `http://www.continuitycentral.com/telecomscontinuityarticles.htm`, Retrieved 2012-04-08.

[12] J. Benesty, M. M. Sondhi, and Y. Huang (Eds.), *Springer Handbook of Speech Processing*. Springer-Verlag, 2008.

[13] IEEE, *802.1D Media Access Control (MAC) Bridges*. IEEE standards, Feb. 2004.

[14] D. Grossman, "RFC3260: New Terminology and Clarifications for DiffServ." IETF, on Website, Apr. 2002. `http://tools.ietf.org/pdf/rfc3260.pdf`, Retrieved 2012-04-06.

[15] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "RFC2205: Resource Reservation Protocol (RSVP) – Version 1 Functional Specification." IETF, on Website, Sept. 1997. `http://tools.ietf.org/pdf/rfc2205.pdf`, Retrieved 2012-04-06.

[16] T. Friedman, R. Caceres, and A. Clark, "RFC3611: RTP Control Protocol Extended Reports (RTCP XR)." IETF, on Website, Nov. 2003. `http://tools.ietf.org/pdf/rfc3611.pdf`, Retrieved 2012-04-06.

[17] ITU-T, *Recommendation H.460.9: Support for Real-Time QoS Monitoring*. ITU-T Recommendation H.460.9, Mar. 2004.

[18] ITU-T, *Recommendation H.248.30: RTCP Extended Performance Metrics Packages*. ITU-T Recommendation H.248.30, Jan. 2007.

[19] Cisco, "Quality of Service for Voice over IP." on Website. `http://www.cisco.com/en/US/docs/ios/solutions_docs/qos_solutions/QoSVoIP/QoSVoIP.html#wp1015329`, Retrieved 2012-04-06.

[20] H. Li and L. Mason, "Multipath Routing with Adaptive Playback Scheduling for Voice over IP in Service Overlay Networks," *Sarnoff Symp., IEEE*, pp. 1–5, Apr. 2008.

[21] ITU-T, *Recomendation G.114: One-way Transmission Time*. ITU Recomendation G.114, May 2003.

[22] J. G. Gruber, "Delay-related Issues in Integradted Voice and Data Networks," *IEEE Trans., Commun.*, vol. 29, pp. 786–800, June 1981.

[23] W. Montgomery, "Techniques for Packet Voice Synchronization," *IEEE JSAC*, vol. 1, pp. 1022–1028, Dec. 1983.

[24] G. Barberis and D. Pazzaglia, "Analysis, Design and Buffer sizing of a Packet Voice Receiver," *IEEE Trans. Commun.*, vol. 28, pp. 217–227, Feb. 1980.

[25] S. B. Moon, J .F. Kurose, and D. Towsley, "Packet Audio Playout Delay Adjustment: Performance Bounds and Algorithms," in *Multimedia Syst.*, vol. 6, pp. 17–28, 1998.

[26] Y. J. Liang, N. Farber, and B. Girod, "Adaptive Playout Scheduling using Time-Scale Modification in Packet Voice Communications," in *IEEE ICASSP'01*, (Salt Lake City, USA), pp. 1445–1448, Jan. 2001.

[27] Y. J. Liang, N. Farber, and B. Girod, "Adaptive Playout Scheduling and Loss Concealment for Voice Communication over IP Networks," *IEEE Trans. Multimedia*, vol. 5, pp. 532–543, Dec. 2003.

[28] Y. Xie, C. Liu, and T. N. Saadawi, "Adaptive Multimedia Synchronization in Teleconference System," in *Proc. IEEE ICC'96*, (Dallas, USA), pp. 1355–1359, June 1996.

[29] Q. Gong and P. Kabal, "A New Optimum Jitter Protection for Conversational IP," in *IEEE Intl. Conf. on Wireless Commun. & Signal Processing*, (Nanjing, China), pp. 1–5, Nov. 2009.

[30] A. Shallwani and P. Kabal, "An Adaptive Playout Algorithm with Delay Spike Detection for Real-time VoIP," in *Proc. IEEE Canadian Conf. Electrical, Computer Engineering*, (Montreal, QC), pp. 997–1000, May 2003.

[31] R. Ramjee, J. Kurose, D. Towsely, and H. Schulzrinne, "Adaptive Playout Mechanisns for Packetized Audio Applications in Wide-area Networks," in *Proc. IEEE INFOCOM'94*, (Toronto, Ontario), pp. 680–688, June 1994.

[32] A. Kansal and A. Karandikar, "Adaptive Delay Estimation for Low Jitter Aaudio over Intenet," in *Proc. IEEE Global Telecommun. Conf.*, (San Antonio, USA), pp. 2591–2595, Nov. 2001.

[33] P. DeLeon and C. Sreenan, "An Adaptive Predictor for Media Playout Buffering," in *Proc. ICASSP*, (Phoenix, USA), pp. 3097–3100, Mar. 1999.

[34] K. Fujimoto, S. Ata, and M. Murata, "Playout Control for Streaming Applications by Statistical Delay Analysis," in *Proc. IEEE ICC'01*, (Helsinki, Finland), pp. 2337–2342, June 2001.

[35] C. J. Sreenan, J. Chen, P. Agrawal, and B. Narendran, "Delay Reduction Techniques for Playout Buffering," *IEEE Trans. Multimedia*, vol. 2, pp. 28–34, July 2005.

[36] S. B. Moon, I. Kuruse, and D. Towslcy, "Packet Audio Playout Delay Adjustment: Performance Bounds and Algorithms," *ACM/Springer Multimedia Syst.*, vol. 6, pp. 17–28, Jan. 1998.

[37] L. Atzori and M. L. Lobina, "Playout Buffering in IP Telephony: A Survey Discussing Problems and Approaches," *IEEE Commun. Surveys & Tutorials*, vol. 8, no. 3, pp. 36–46, 2006.

[38] M. Ghanassi and P. Kabal, "Optimizing Voice-over-IP Speech Quality Using Path Diversity," in *Proc. IEEE Workshop on Multimedia Signal Processing*, (Victoria, CA), pp. 155–160, Oct. 2006.

[39] L. Atzori and M. L. Lobina, "Speech Playout Buffering Based on a Simplified Version of the ITU-T E-Model," *IEEE Signal Processing Letters*, vol. 11, pp. 382–385, Mar. 2004.

[40] M. Yavuz, S. Diaz, R. Kapoor, M. Grob, P. Black, Y. Tokgoz, C. Lott, S. Guha, N. Daswani, and R. Jain, "VoIP over CDMA2000 1xEV-DO Revision A," *IEEE Commun. Magazine*, vol. 44, pp. 88–95, Feb. 2006.

[41] ITU-T, *Recommendation G.729 Annex B: A Silence Compression Scheme for G.729 Optimized for Terminals Conforming to Recommendation V.70*. ITU Recomendation G.729, Jan. 2007.

[42] J-C. Bolot, S. Fosse Parisis, and D. Towsley, "Adaptive FEC-based Error Control for Internet Telephony," in *Proc. IEEE INFOCOM'99*, vol. 3, (New York, USA), pp. 1453 – 1460, Mar. 1999.

[43] H. Schulzrinne, S. Casner, and V. Jacobson, "RFC3550: RTP: A Transport Protocol for Real-Time Applications." IETF, on Website, July 1997. `http://tools.ietf.org/pdf/rfc3550.pdf`, Retrieved 2012-04-06.

[44] Q. Gong and P. Kabal, "Quality-Based Playout Buffering with FEC for Conversational VoIP," in *Proc. Interspeech*, (Makuhari, Japan), pp. 2402–2405, Sept. 2010.

[45] Q. Gong and P. Kabal, "Improved Quality for Conversational VoIP using Path Diversity," in *Proc. Interspeech*, (Florence, Italy), pp. 2549–2552, Aug. 2011.

[46] H. Schulzrinne, "Some Frequently Asked Questions about RTP." on Website. `http://www.cs.columbia.edu/~hgs/rtp/faq.html`, Retrieved 2012-04-06.

[47] S. Antoniou, "VoIP Signaling Protocols." on Website, June 2010. `http://www.trainsignaltraining.com/voip-signaling-protocols`, Retrieved 2012-04-06.

[48] D. Minoli and E. Minoli, *Delivering Voice over IP Networks*. Wiley Publishing, Inc, 2002.

[49] Packetizer, "H.323 versus SIP: A Comparison." on Website. `http://www.packetizer.com/ipmc/h323_vs_sip/`, Retrieved 2012-04-09.

[50] J. Glasmann, W. Kellerer, and H. Muller, "Service Architectures in H.323 and SIP: A Comparison ," *IEEE Commun. Surveys & Tutorials*, vol. 5, pp. 32–47, 2003.

[51] B. Goode, "Voice Over Internet Protocol (VoIP)," *Proceedings of the IEEE*, vol. 90, pp. 1495–1517, Sept. 2002.

[52] ITU-T, *Recomendation H.323: Packet-based Multimedia Communications Systems*. ITU Recomendation H.323, July 2003.

[53] M. Handley et al., "SIP: Session Initiation Protocol." on Website, Mar. 1999. `http://www.ietf.org/rfc/rfc2543.txt`, Retrieved 2012-04-06.

[54] A. S. Spanias, "Speech Coding: A Tutorial Review," *Proceedings of the IEEE*, vol. 82, pp. 1541–1582, Oct. 1994.

[55] W. B. Kleijn and K. K. Paliwal, *Speech Coding and Synthesis*. Elsevier, 1995.

[56] R. Bracewell, *The Fourier Transform at Its Applications*. McGraw Hill, 1986.

[57] G. Kubin, B. S. Atal, and W. B. Kleijn, "Performance of Noise Excitation for Unvoiced Speech," in *Proc. IEEE Workshop on Speech Coding for Telecomm.*, (Sainte-Adele, Quebec), pp. 35–36, Oct. 1993.

[58] M. Hasegawa-Johnson and A. Alwan, *Encyclopedia of Telecommunications*. John Wiley & Sons, 2003.

[59] Wikipedia, "μ-law algorithm." on Website. `http://en.wikipedia.org/wiki/%CE%9C-law_algorithm`, Retrieved 2012-04-06.

[60] ITU-T, *Recomendation G.711: Pulse Code Modulation (PCM) of Voice Frequencies*. ITU Recomendation G.711, Feb. 2002.

[61] K. Sayood, *Introduction to Data Compression*. Morgan Kaufmann, 2000.

[62] S. Haykin, *Adaptive Filter Theory*. Prentice-Hall, 2002.

[63] W. C. Chu, *Speech Coding Algorithm*. John Wiley & Son, Inc, 2003.

[64] C. Laflamme, J. P. Adoul, R. Salami, S. Morissette, and P. Mabilleau, "16 kbps Wideband Speech Coding Technique Based on Algebraic CELP ," in *Proc. ICCASP 1991*, (Toronto, Ontario), pp. 13–16, Apr. 1991.

[65] S. Andersen, *RFC3951: Internet Low Bit Rate Codec (iLBC)* . Network Working Group, Dec. 2004.

[66] ITU-T, *Recomendation G.723.1: Dual Rate Speech Coder for Multimedia Communications Transmitting at 5.3 and 6.3 kbit/s*. ITU Recomendation G.723.1, May 2006.

[67] Aziz Shallwani, "An Adaptive Playout Algorithm with Delay Spike Detection for Real-Time VoIP," Master's thesis, McGill University, 2003.

[68] W. Jiang and H. Schulzrinne, "Modeling of Packet Loss and Delay and Their Effect on Real-Time Multimedia Service Quality," in *Proc. NOSSDAV*, (Chapel Hill, USA), June 2000. 4pp, paper 27.

[69] ITU-T, *Recomendation P.10: Vocabulary for Performance and Quality of Service*. ITU Recomendation P.10, July 2006.

[70] ITU-T, *Recomendation P.800.1: Methods for Subjective determination of Transmission Quality*. ITU Recomendation P.800.1, Mar. 2003.

[71] ITU-T, *Recomendation P.862: Perceptual Evaluation of Speech Quality (PESQ), an objective method for end-to-end speech quality assessment of narrowband telephone networks and speech codecs*. ITU Recomendation P.862, Nov. 2005.

[72] C. Hoene, H. Karl, and A. Wolisz, "A Perceptual Quality Model for Adaptive VoIP Applications," in *Proc. Intl. Symp. on Performance Evaluation of Computer and Telecommun. Syst. (SPECTS'04)*, (San Jose, USA), July 2004.

[73] R. G. Cole and J. H. Rosenbluth, "Voice over IP Performance Monitoring," *SIGCOMM Computer Commun. Review*, pp. 9–24, Apr. 2001.

[74] D. Malah, "Time-domain Algorithms for Harmonic Bandwidth Reduction and Time Scaling of Speech Signals," *IEEE Trans. Acoustics, Speech, Signal Processing*, vol. ASSP-27, pp. 121–133, Apr. 1979.

[75] J. Laroche, "Autocorrelation Method for High Quality Time/Pitch Scaling," in *Proc. IEEE Workshop Application Signal Processing Audio Acoustisc*, (New Paltz, USA), pp. 131–134, Oct. 1993.

[76] W. Verhelst, "Overlap-add Methods for Time-scaling of Speech," *Speech Commun.*, vol. 30, no. 4, pp. 207–221, 2000.

[77] W. Verhelst and M. Roelands, "An Overlap-add Technique Based on Waveform Similarity (WSOLA) for High Quality Time-scale Modification of Speech," in *IEEE ICASSP'93*, (Minneapolis, USA), pp. 554–557, Apr. 1993.

[78] H. Sanneck, A. Stenger, K. Ben Younes, and B. Girod, "A New Technique for Audio Packet Loss Concealment," in *IEEE GLOBECOM*, (London, UK), pp. 48–52, Nov. 1996.

[79] M. Narbutt and L. Murphy, "VoIP Playout Buffer Adjustment Using Adaptive Estimation of Network Delays," in *Proc. 18th Intl. Teletraffic Congress (ITC-18)*, (Berlin, Germany), pp. 1171–1180, Sept. 2003.

[80] IETF, "RFC793: Transmission Control Protocol." IETF, on Website. `http://www.ietf.org/rfc/rfc793.txt`, Retrieved 2012-04-06.

[81] L. Sun and E. Ifeachor, "Prediction of Perceived Conversational Speech Quality and Effects of Playout Buffer Algorithms," in *Proc. IEEE ICC'03*, (Anchorage, USA), pp. 1–6, May 2003.

[82] C. Boutremans and J-Y. Le Boudec, "Adaptive Joint Playout Buffer and FEC Adjustement for Internet Telephony," in *Proc. IEEE INFOCOM'03*, vol. 1, (San Francisco, USA), pp. 652–662, Mar. 2003.

[83] K. Fujimoto, S. Ata, and Murata, "Adaptive Playout Buffer Algorithm for Enhancing Perceived Quality of Streaming Applications," *Springer Telecommun. Syst.*, vol. 25, pp. 259–271, Mar. 2004.

[84] C. Perkins, O. Hodson, and V. Hardman, "A Survey of Packet-Loss Recovery Techniques for Streaming Audio," *IEEE Network*, vol. 12, pp. 40–48, Sept. 1998.

[85] Z. Duan, Z. L. Zhang, and Y. T. Hou, "Service Overlay Networks: SLAs, QoS, and Bandwidth Provisioning," *IEEE/ACM Trans. Networking*, vol. 11, pp. 870–883, Dec. 2003.

[86] N. D. S. Guha and R. Jain, "An Experimental Study of the Skype Peer-to-Peer VoIP System," in *Proc. of The 5th Intl. Workshop on Peer-to-Peer Syst. (IPTPS'06)*, (Santa Barbara, CA), Feb. 2006.

[87] J. G. Apostolopoulos, "Reliable Video Communication Over Lossy Packet Networks using Multiple State Encoding and Path Diversity," in *Proc. Visual Commun. and Image Processing 2001*, (San Jose, USA), pp. 392–409, Jan. 2001.

[88] Y. J. Liang, E. G. Steinbach, and B. Girod, "Real-time Voice Communication over the Internet using Packet Path Diversity," in *Proc. the Ninth ACM Intl. Conf. on Multimedia*, (Ottawa, CA), pp. 431–440, Oct. 2001.

[89] Cfos, "Ping Utility hrPING v3.13." on Website. `http://www.cfos.de/en/ping/ping.htm`, Retrieved 2012-04-06.

[90] R. G. Cole and J. Rosenbluth, "Voice over IP Performance Monitoring," *Journal on Computer Commun. Review*, vol. 31, pp. 9–24, Apr. 2001.

[91] L. Sun, *Speech Quality Prediction for Voice over Internet Protocol Networks*. PhD thesis, University of Plymouth, 2004.

[92] S. B. Moon, P. S. Kelly, and D. Towsley, "Estimation and Removal of Clock Skew from Network Delay Measurements," in *Proc. IEEE INFOCOM*, (NewYork, USA), pp. 21–25, Mar. 1999.

[93] L. Atzori, M. L. Lobina, and M. Corona, "Playout Buffering of Speech Packets based on a Quality Maximization Approach," *IEEE Trans. Multimedia*, vol. 10, pp. 420–426, Apr. 2006.

[94] A. Clark, *Extensions to the E Model to Incorporate the Effects of Time Varying Packet Loss and Recency*. ETSI T1A1.1/2001-037, Apr. 2001.

[95] L. Tosun and P. Kabal, "Dynamically Adding Redundancy for Improved Error Concealment in Packet Voice Coding," in *Proc. European Signal Processing Conf.*, (Antalya, Turkey), pp. 4–8, Sept. 2005.

[96] ITU-T, *Recomendation G.131: Talker Echo and Its Control*. ITU Recomendation G.131, Mar. 2003.

[97] L. Cai, Y. Xiao, X. Shen, L. Cai, and J. W. Mark, "VoIP over WLAN: Voice Capacity, Addmission Control, QoS, and MAC," *Intl. Journal of Commun. Syst.*, vol. 19, pp. 491–508, 2006.