THE DESIGN OF A MULTI-PARTY VOIP CONFERENCING SYSTEM

BY

ZIXIA HUANG

B.Eng., Shanghai Jiao Tong University, 2006

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2009

Urbana, Illinois

Adviser:

Professor Benjamin W. Wah

ABSTRACT

This thesis identifies the problems and trade-offs of a multi-party VoIP conferencing system implemented over the Internet and proposes approaches to solve these problems. The current Internet is unreliable, and it degrades the conversational quality of real-time multi-party conferencing. Delay disparities may cause unbalanced silence periods, and losses and jitters may affect the intelligibility of speech segments received. We collect real Internet traces from the PlanetLab and classify them into different categories according to the traffic behavior. After studying the conversational dynamics in the multi-party system, we identify user-observable metrics that affect the perception of conversational quality and study their trade-offs. Based on the dynamics and the Internet traces, we design the transmission topology to reduce delay variations and to avoid links with high losses and jitters. We propose loss concealment schemes for reducing the packet drop rate and play-out scheduling algorithms for equalizing silence periods and smooth jitters. We also discuss issues and solutions in a practical multi-party VoIP system design. We compare the performance of our system and that of Skype (Version 3.5.0.214) using repeatable experiments that simulate human participants and network conditions in a multi-party conferencing scenario. Our limited, subjective tests show that we can improve the perceptual quality when network connections are lossy and have large delay disparities. Because it is impossible to conduct subjective tests under all possible conditions, we have developed a classifier that learns to select the best equalization algorithm using learning examples derived from subjective tests under limited network and conversational conditions. Experimental results show that our classifier can consistently pick the best algorithm with the highest subjective conversational quality under unseen conditions.

To mom and dad

ACKNOWLEDGMENTS

I would like to give thanks to my research advisor, Prof. Benjamin W. Wah, for all his stimulating discussions and meaningful advice throughout my graduate study. I have learnt a lot from working with him and have benefited from his professionalism.

I would also like to thank Batu Sat, Chih-Wei Hsu, Hang Yu, and other members in my research group for their friendly assistance, interesting ideas, and all useful group meetings.

Special thanks go to my parents, Shen Huang and Jianping Xia, who consistently give me encouragement, support, and love.

This research was supported in part by National Science Foundation Grant CNS 08-41336.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Motivations

The current Internet is unreliable and provides only best-effort delivery. For interactive multimedia transmitted over the Internet, as in a Voice over Internet Protocol (VoIP) system, jitters and losses may degrade the intelligibility of the multimedia contents, while latency incurred during transmissions is directly related to the interactivity. VoIP users like a conversation that is comparable to what they usually experience in face-to-face communications. They care about good speech signal quality and appropriate response time. But when the VoIP performance is suboptimal, different people may have a preference on different aspects, depending on each subjective evaluation. Existing objective metrics can evaluate each aspect separately; none of those, however, is able to describe the combined subjective human perception.

A multi-party VoIP system includes at least three users in the conference, so both the conversational order and network condition are different from those of two-party systems. Internet traffic between any two users in the conference is not uniform, and the diversity of network patterns may influence listening quality. We are interested in designing a multi-party VoIP conferencing system that can provide the best user experience, but a huge variety of conversational order and Internet traffic combinations pose a potential problem for generalization. All these considerations motivate us to investigate the design problem of the multi-party VoIP system and issues of subjective evaluations.

1.2 Problems Studied

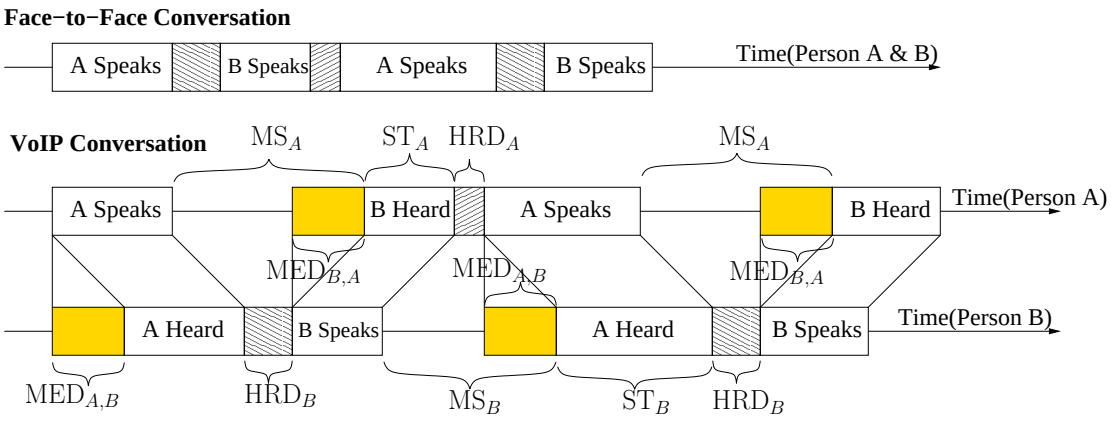
1.2.1 Conversational quality in VoIP conferencing

Voice conversation is the most natural form of interpersonal communication. In a conversation of two or more participants, each person takes turns in uttering his/her thoughts and listens to others, and everyone perceives a silence duration (called *mutual silence* or MS) in between turns (*speech segments*).

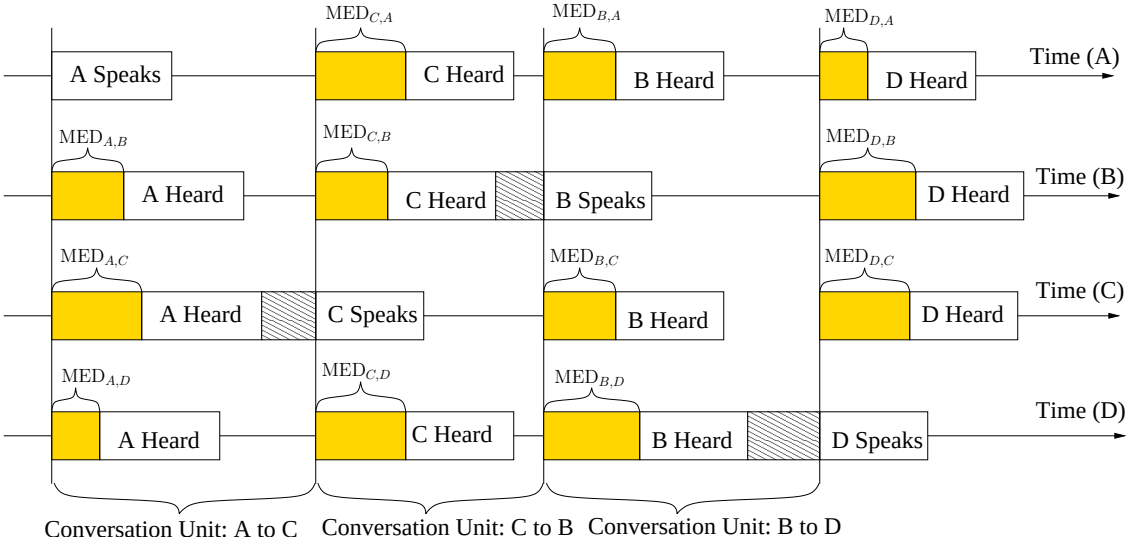
In a two-party face-to-face conversation, in which two participants reside in the same physical location, such as a meeting room, both clients have a common perspective of the conversation and experience approximately the same durations of mutual silence (see Figure 1.1(a)). This gives participants a sense of interactivity during the conversation.

However, when a two-party conversation is carried out over a network, a speech packet from a speaker may experience a latency before it arrives at a listener. This is called the *mouth-to-ear delay* or MED, as it describes a delay from the mouth of a speaker to the ear of a listener. MED is usually incurred at three places: the sender, the network and the receiver. The mutual silences now are perceived as alternating short and long silence durations between turns (Figure 1.1(a)). This asymmetry is caused by the fact that after A speaks, the silence period experienced by A is governed by the time for A's speech to travel to B ($\mathrm{MED}_{A,B}$), the time for B to construct a response (called the *human response delay* of B or HRD_B), and the time for B's response to travel to A ($\mathrm{MED}_{B,A}$). In contrast, after A receives the response from B, the silence period experienced before A speaks is only governed by his/her HRD_A.

As a VoIP user usually compares the conversation with a face-to-face communication, he/she waits for the next speech segment in a pre-estimated time range. If the MS is longer than this range, the listener may feel impatient and assume that the other person is not responding timely or that the speech packets may have been dropped in the Internet. This asymmetry leads to a degraded perception of interactivity in the two-party conversation. A larger MED causes a longer MS and may reduce the satisfaction rate [1, 2] and the *conversational quality* of a VoIP listener. Note that an excessively long MS can result in one client starting to talk before the other client completes. This can cause double-talks, leading to confusion and further degradation of



(a) Face-to-face and 2-party conversations



(b) A 4-party conversation

Figure 1.1: The dynamics of VoIP conversations.

interactivity and conversational quality.

The extension of a VoIP system from two-party to multi-party is not straightforward, as the perceived effects of delays in multi-party VoIP is more complex (Figure 1.1(b)). The network latency is not uniform across the links sent from the same client and shows a greater diversity (e.g., the intercontinental latency is usually much larger than that of an intracontinental link). As a result, the speech uttered by one client can arrive at different clients with different delays. Unlike a two-party conversation, where both participants take turns to speak, the order in which participants speak is not predetermined in a multi-party conversation, as there are multiple possibilities of who is the speaker in the next turn. Hence, each client may experience different mutual silences and a different perspective from the other clients. Figure 1.2 shows the MS variations in a multi-party VoIP conversation using Skype[1] software.

The dynamic, unreliable nature of the Internet may also degrade the conversational quality of VoIP users. Packets may be lost, either in isolation or in batches [3], and may experience sudden delay increases (jitters [4, 5]). This behavior causes packets to be unavailable at the receiver at their scheduled play-out times, and has a direct impact on the understanding of speech contents. We call it the *listening-only speech quality* (LOSQ) [6], as it is solely related to the intelligibility of speech heard (though a VoIP user may lack a reference of the original speeches) and does not describe any issues of interactivity.

To smooth the irregular arrival of packets, receivers commonly employ jitter buffers [7, 8, 9] for storing received packets and play-out schedulers for playing the speech signals. Loss concealment [10, 11, 12] techniques are implemented to recover lost speech frames. However, the fraction of those frames that cannot be correctly received (called *unconcealed frame rate* or UCFR) depends on the buffering time at the receiver. A larger receiver buffer will increase MED and reduce UCFR. Thus LOSQ is improved accordingly.

Note that degradations in LOSQ may also depend on the codec used in a VoIP system. A high bit-rate codec tends to include more information of the original speeches than a low bit-rate codec and thus provides clearer sound and improves LOSQ after the encoding/decoding process. A higher bit-rate codec is also more robust to losses, as it usually includes more redundant bits that

[1] Skype is a trademark of Skype Technologies S.A. in Luxembourg. URL: `http://www.skype.com`.

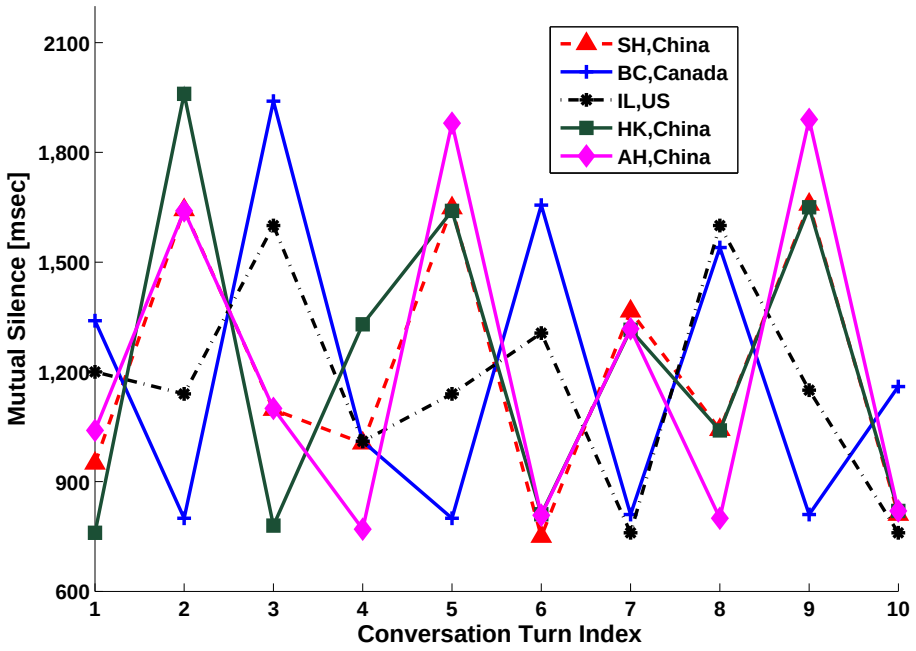


Figure 1.2: MS variations from a multi-party VoIP conversation using Skype.

can facilitate the loss concealment at the codec level using techniques such as linear prediction or code-book reconstruction.

MS variations and LOSQ influence the conversational quality of VoIP users, but both aspects cannot be improved simultaneously. A longer MED can improve LOSQ, but it will also increase MS at the same time. Their trade-off is shown in Figure 1.3, and the intersection of the two curves in the figure depicts the best operating point that can strike a balance between MS and LOSQ. Note that for different network and conversational conditions, the curves and the best operating point to achieve the optimal conversational quality are different.

1.2.2 Subjective and objective evaluations

Users evaluate conversational quality based on subjective perception that is affected by the tradeoff between MS and LOSQ. Several objective metrics are proposed to evaluate each aspect separately. For example, PESQ (*Perceptual Evaluation of Speech Quality*, defined in ITU P.862 [13]) is used to measure the LOSQ. We can also directly measure and record the MS durations. However, none of these objective metrics by itself can evaluate the combined effect that

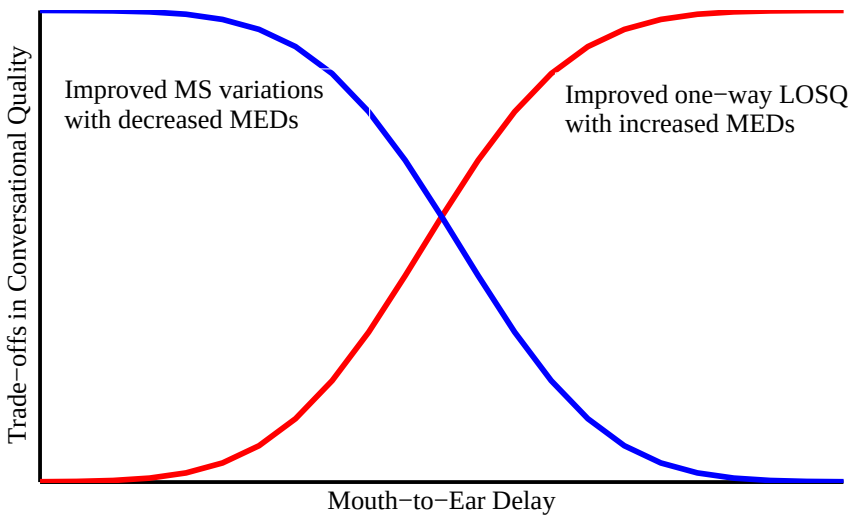


Figure 1.3: Trade-offs between MS variations and LOSQ.

can describe human perceptions. Therefore, subjective tests are still needed to study the impact of MS variations and LOSQ on conversational quality. There are several shortcomings for subjective evaluations, however. First, it is nearly impossible to do subjective tests on-line, and they are expensive if done off-line. Second, the results of subjective tests may depend on the expertise of VoIP users and are very difficult to repeat, even under the same conversational and network conditions. Third, it is very hard to give a score to a single conversation without providing a reference of the subjective evaluations. Fourth, two outputs may be *incomparable* because of the trade-offs between MS and LOSQ and the fact that different people may have different preferences. Hence, subjective evaluations are used only off-line to guide the system designs, and only objective metrics can be run on-line.

Based on the study of off-line subjective evaluations and objective metrics that are computed on-line, a good design of a multi-party VoIP system should be able to dynamically find the best operating point that can achieve the optimal conversational quality and adjust its MEDs using control algorithms. However, because there are a large number of network and conversational conditions, a VoIP system may meet an unseen condition on-line. Given this problem, a good model is to learn the trade-offs among the objective metrics that affect the subjective quality and generalize the results to unseen situations.

1.2.3 A study of MS variations on conversational quality

The impact of LOSQ on conversational quality is easy to understand, and there are numerous studies related to it [6, 13, 14]. In this section, we study only the impact of MS variations on conversational quality. There are several previous studies related to the effect of MED on listeners [1, 2, 15]. However, none of these studies takes into account the loss effect and MS variations, and none is conducted in a multi-party conferencing scenario.

In our study, we construct multiple real multi-party conferences using the same set of speech segments. In each conference, the silence durations between the speech segments are carefully selected so that all conference outputs in the experiment show a variety of MS ranges. We ask nine people to compare any of the two conference outputs and provide a preference. We assume no losses in our study in order to reduce the LOSQ effects on the conversational quality. The results are shown in Table 1.1.

Table 1.1 shows that people have a preference for small MS variations (ratios of the maximum MS over the minimum in the test). We can conclude from the study and the user experiences of multi-party VoIP conferences that MS variations may lead to three potential problems.

1. Each listener will have a slightly different perception of the same conversation in a conference call. This may cause double-talks, when more than one persons start speaking at the same time and the listeners perceive the double-talk at different points in time.

2. From a listener's perspective, there is asymmetry in the silence durations in between different speakers' speeches. This means that some speakers may appear to be more distant than others, or some respond slower than others.

3. When the same speech is delivered at different quality to different listeners, it is possible that one listener cannot understand an utterance and request the speaker to repeat it. This leads to significant inefficiency to all participants.

Note that there are no incomparable results in the study, because of the no loss assumption.

This study provides a direct guideline to our multi-party VoIP system design. We need to design a play-out scheduling (POS) algorithm that can equalize the variations of MS while

Table 1.1: A study of MS variations on conversational quality.

$maxMS/minMS$	1.00	1.25	1.50	1.75	2.00	2.25
1.00	(0,9,0)	(1,7,1)	(7,2,0)	(9,0,0)	(9,0,0)	(9,0,0)
1.25	(1,7,1)	(1,8,0)	(1,7,1)	(6,3,0)	(8,1,0)	(9,0,0)
1.50	(0,2,7)	(1,7,1)	(1,7,1)	(2,6,1)	(5,4,0)	(8,1,0)
1.75	(0,0,9)	(0,3,6)	(1,6,2)	(0,8,1)	(1,6,2)	(5,4,0)
2.00	(0,0,9)	(0,1,8)	(0,4,5)	(2,6,1)	(1,8,0)	(1,6,2)
2.25	(0,0,9)	(0,0,9)	(0,1,8)	(0,4,5)	(2,6,1)	(1,7,1)

Note: the MS variations are described as the maximum over the minimum. Results are presented in terms of the number of respondents in (better than, about the same as, worse than). Nine people were invited to participate in the test.

guaranteeing LOSQ. A topology also needs to be carefully selected in order to reduce the diversity of MEDs and consequently jitters and losses.

1.2.4 Problem statement

The goal of this study is to design a multi-party VoIP conferencing system that can achieve better subjective conversational quality with reduced MS variations and improved LOSQ and be consistent across time and participants. We study the conversational dynamics and investigate the trade-offs among various components of network control schemes that cover transmission topology, loss concealment, and play-out scheduling in VoIP conferencing. We develop an automated learning model that can be generalized to unseen conversational and network conditions and can select the best learned algorithms at run time. We evaluate and verify the performance of our system and learning model using speech outputs from our multi-party conferencing prototype and under different network and conversational conditions.

1.3 Our Approaches

The following procedures illustrate our approaches of solving the problems of multi-party VoIP conference.

1. We collect Internet traces from PlanetLab,[2] study the traffic behavior, and classify the traces

[2]PlanetLab - An open platform for developing, deploying, and accessing planetary-scale services. URL: `http://www.planet-lab.org/`.

into different categories so that repeatable experiments can be conducted and analyzed in each category.

2. We discuss the multi-party conversational dynamics and identify user-observable objective and subjective metrics that affect conversational quality. The objective metrics can capture the effects of MS variations and LOSQ. The subjective counterparts can describe the human subjective perception of VoIP conferencing. We study the impacts, interrelations, and trade-offs among these metrics.

3. We present a detailed design of the multi-party VoIP system. Special focus is given to the design of the transmission topology and the play-out scheduling scheme. Our proposed topology minimizes the network latency diversity so that MS variations can be reduced. It also avoids paths with high jitters and losses, using network traffic data collected in real time. Our play-out scheduling algorithms are designed to equalize the silence periods and smooth the jitters effectively. We also implement a Visual C version of the multi-party VoIP system under Microsoft Windows so that practical issues can be analyzed and solved.

4. We develop an automated learning model that can find a mapping from objective metrics to subjective test results. It is generated from learning examples derived from subjective tests under different network and conversational conditions. By employing the mappings at run time, this classifier can dynamically select the best among several play-out scheduling algorithms that can achieve good conversational quality, even under unseen conditions.

5. We verify our VoIP system using traces collected under various conditions. We also compare our system with Skype. The resulting speech output and subjective tests are used to generate the classifier. We test the prediction accuracy of the learning model using unseen network and conversational conditions.

1.4 Contribution of Our Work

The contribution of our work is twofold. First, we propose a quasi-optimal transmission topology that can reduce latency diversity and MS variations and avoid links with high jitters and

losses. The topology determination is based on probed network traffic. A novel play-out scheduling scheme is also presented that can equalize silence periods between two speech turns and can smooth jitters effectively. Second, we propose a classifier that can learn to choose the best play-out scheduling algorithm at run time dynamically.

1.5 Thesis Outline

Chapter 2 presents a complete study of previous related work that has been done on conversational quality, multi-party VoIP system design and Internet speech codecs. Skype's commercial VoIP conferencing for multi-party clients is also studied. Chapter 3 analyzes the Internet traffic patterns in terms of delays, jitters and losses. Chapter 4 identifies user-observable metrics that affect the perception of conversational quality and studies their trade-offs. Based on these metrics, we propose new transmission topology, loss concealment, and play-out scheduling schemes. Implementation issues for practical systems are discussed. A classifier that can learn to select the best equalization algorithm is also proposed. Chapter 5 discusses the experiment setup and results. Chapter 6 summarizes the work of this thesis.

CHAPTER 2

RELATED STUDY

This chapter surveys the standard metrics and related study on conversational quality. Their applications and limitations are analyzed. Different techniques for the core components of a VoIP design are also illustrated and compared. At the end of chapter, we present a complete study of Skype, a popular VoIP system that supports both two-party and multi-party communications.

2.1 Related Study on Evaluating Conversational Quality

There have been several studies and standard metrics related to the conversational quality of a VoIP system. We briefly illustrate these studies and standards and discuss their pros and cons in this section.

2.1.1 Effects of delays on conversational quality

Kiatawaki and Itoh at Nippon Telegraph and Telephone Corporation (NTT) [15] studied the pure delay effect on speech quality, and their results showed that one-way delays are detectable and influence listeners subjective assessment. Their studies were based on the transmissions of signals over a telephone switched network, where delay variations were small and no losses assumed.

Brady [2] and Richards [1] conducted several subjective evaluations of delay effects on satellite communication and concluded that longer delays could decrease the satisfaction rate and increase the likelihood of double-talks. However, they only tested on several constant delays in their study; thus, their results could not be directly applied to the VoIP system, where there are frequent losses and delay variations.

ITU G.114 [16] prescribes that in a two-party conversation, a one-way delay of less than 150 msec is desirable and a delay of more than 400 msec is unacceptable. However, the standard does

not prescribe any multi-party conversation scenario.

All these studies are related to the pure delay effect. Without considering the loss effect on the speech quality and the trade-offs between loss and delay, however, it is difficult to evaluate the conversational quality over VoIP. Hence, in our study, we combine the loss effects with delays and study their interactions and trade-offs on VoIP conferencing quality.

2.1.2 Subjective standard measures

ITU P.800 [17] and P.800.1 [18] prescribe subjective measures that can evaluate the speech quality when using a VoIP system. These measures can primarily be divided into two categories: *absolute category rating* (ACR) and *comparative category rating* (CCR).

In ACR, users are asked to give assessment of VoIP quality in $(1, 2, 3, 4, 5)$ corresponding to (*Bad*, *Poor*, *Fair*, *Good*, *Excellent*) based on their subjective perceptions, and the final ranking is based on the average of all the scores called the *mean opinion score* (MOS). Three different situations where ACR can be applied have to distinguished:

1. Listening only situation ($\mathrm{MOS}_{\mathrm{LQS}}$): MOS scoring is applied to a listening-only situation.
2. Conversational situation ($\mathrm{MOS}_{\mathrm{CQS}}$): MOS scoring is applied to a conversational situation.
3. Talking only situation ($\mathrm{MOS}_{\mathrm{TQS}}$): MOS scoring is applied to the quality of a phone call only as perceived by the talking party.

In CCR, users are asked to compare the quality between two output in $(-3, -2, -1, 0, 1, 2, 3)$ corresponding to (*Much Worse*, *Worse*, *Slightly Worse*, *About the Same*, *Slightly Better*, *Better*, *Much Better*), and the final ranking is also based on the average of these scores, called the *comparative mean opinion score* (CMOS). Only a listening situation is defined in the ITU P.800 standard.

Because there are trade-offs between LOSQ and delay on conversational quality, sometimes two conversations are *incomparable* because different people may focus on different ones of the two aspects and evaluate the conversations solely based on their preferences. Also, it is difficult to assess a conversation using a grade in ACR without a standard that defines assessment guidelines because of the trade-offs.

The pros of subjective metrics are that they can better reflect users' preferences. The cons are that subjective assessments cannot be conducted at run time, and it is very costly to conduct subjective ratings off-line. The results are sometimes very hard to repeat, even for the same network and conversational situations. The results are also affected by the level of expertise of the users.

Because subjective measures can best describe user experiences of a multi-party VoIP conferencing system, we conduct subjective tests in our study evaluating conversational quality. However, because of the limitations of subjective measures discussed, we need to combine them with other types of metrics (say, objective counterparts) or find ways to use other metrics to model subjective metrics in our research.

2.1.3 Objective standard measures

ITU P.862 [13] prescribes an objective measure called *perceptual evaluation of speech quality* (PESQ), which can evaluate LOSQ of a degraded speech frame with a reference to the original frame. Its value is highly correlated to subjective MOS assessment and can be transformed to $\mathrm{MOS_{LQO}}$ using a mapping equation (Eq. (2.1)) defined in ITU P.862.1 [19]. The maximum value of PESQ is 4.5 instead of 5 as defined in MOS. This is reasonable because people usually tend to be cautious when grading a speech output.

$$\mathrm{MOS_{LQO}} = 0.999 + \frac{4.999 - 0.999}{1 + e^{-1.4945 \cdot \mathrm{PESQ} + 4.6607}} \tag{2.1}$$

PESQ can be evaluated only off-line, because it requires the original speech frames as references. Also PESQ does not consider any delay effect, so it has to be used to evaluate VoIP conversational quality in conjunction with other objective measures.

The E-model (ITU G.107 [20]) is designed to use a linear model (Eq. (2.2)) to estimate conversational quality. The primary output is a scalar quality called the *Transmission Rating Factor* (R), which can also be transformed into MOS using a mapping function. The model considers several factors affecting conversational quality, including the basic signal-to-noise ratio (R_0); all impairments that happen simultaneously with voices, such as quantization noise and too-loud speech level (I_s); impairments due to delay and echo effects (I_d); impairments due to low bit-rate

codecs and losses ($I_{e,\text{eff}}$); and an *advantage factor* that can compensate for the impairments (A).

$$R = R_0 - I_s - I_d - I_{e,\text{eff}} + A \tag{2.2}$$

Some studies have tried to utilize the E-model for improving the VoIP quality [21, 22]. However, the E-model oversimplifies the problem by assuming independence and simple additivity of various factors. Because of the trade-offs between delays and LOSQ, this independence may not stand and their combined effects cannot be linearly added by individual ones. Therefore, it is difficult to use this model for evaluating the conversational quality in a real-time VoIP system.

The pros of the objective metrics are that most of them (except PESQ) can be extracted and evaluated at run time and can lead to repeatable results. The cons are that each objective metric can evaluate solely an individual factor and cannot capture the combined, complex trade-offs under all network and conversational conditions.

ITU-T Study Group 12 [23] has recognized the lack of metrics that capture the trade-offs between LOSQ and delays and is working on defining new metrics for measuring conversational quality under delays. The group proposed an updated version of the E-model in the 2005–2008 study period, but it had not been released by the time we finished this thesis. As of the 2009–2012 study period, the group is trying to apply the E-model to VoIP quality monitoring and to reflect the influences of pure delays on interactivity. They are also working on proposing a model for objective conversational voice quality assessment, and the work is expected to be finished by 2011. However, we are not sure whether their metrics will lead to any useful results that can help design a VoIP conferencing system.

In our study, we extract objective metrics for different factors that may influence conversational quality both on-line and off-line. By combining these metrics with subjective counterparts, we can study the effects of objective factors, either separately or as a whole, on subjective assessments. Based on the results, we can develop control algorithms to enhance each objective aspect and improve final subjective evaluations.

2.2 Related Study on VoIP System Design

The design of a multi-party VoIP conferencing system includes the determination of the conferencing topology, a choice of the proper jitter buffer and play-out scheduling algorithm to smooth jitters, loss concealment techniques for dropped-frame recovery, and selecting an appropriate Internet speech codec. In this section, we discuss and analyze previous studies of each component.

2.2.1 Topology design

The determination of a conferencing topology is a key component of multi-party VoIP system design. There are several factors that have to be considered, including the CPU and network bandwidth usage of each client, as well as end-to-end delay between any two clients.

Decentralized topology

A *decentralized* scheme [24] requires each client to send packets to every listener, either directly via uni-casts or via multi-casts if supported by the underlying network. The most common architecture is a full-mesh topology (Figure 2.1), where each of the N speaking clients sends its data to each of the $M-1$ listening clients via uni-casts. Although the *maximum end-to-end delay* (ME2ED) is the shortest in this topology, the scheme may be bottlenecked at a client (in terms of both CPU usage and network conditions), especially when the number of clients is large. Each client maintains $M-1$ jitter buffers and decoders, $N(t)$ of which are active at time t.

Centralized topology

In a *centralized* scheme [25], shown in Figure 2.2, all the clients communicate with one of the VoIP clients, called the *central host*, through which all speech packets are relayed. The number of clients in the centralized scheme has to be limited, because this topology will cause tremendous CPU and bandwidth burdens on the central host. Degradations of speech quality can be propagated to all clients in the conference if the network condition at the central host is poor. Another disadvantage is that ME2ED in the topology can be very large if the central host is not selected appropriately. For example, assume that there are five clients in the conference, of which the

central host is in India and the rest are in the United States. The latencies between India and the United States can be as large as 400 msec, whereas the latencies among the clients in the United States are less than 50 msec. Huge ME2ED can increase the asymmetry in the multi-party conversation and MS variations.

Overlay network

Figure 2.3 shows an *overlay network* [26, 27, 28]. Each speaking client communicates with the nearest node in the overlay network, whereas nodes in the overlay network (called *parent nodes*) send packets to each other using either uni-casts or multi-casts. The burdens on the parent nodes are significantly reduced as compared to a centralized topology, because the number of parent nodes is now determined by M in the conference. All the other nodes in this topology are called *child nodes*. There have been several studies on overlay network designs, both in general and in the context of VoIP conferencing applications [24, 29], using different optimization criteria. However, in [24], the authors only compare the pros and cons of several topologies and do not provide a method for determining the best overlay network. In [29], the paper combines the losses into latencies, but their assumption and derivation are based on the retransmission of lost packets, which is impossible in time-sensitive VoIP systems. Neither does the paper consider the diversity of MEDs and the effects on MS variations. Hence, in designing a new overlay network, reducing MED diversities will be the top priority in the topology design.

Topology with dedicated servers

As the name suggests, this topology (Figure 2.4) [30] employs some dedicated servers as gateways to transmit packets. There have been several studies presenting different communication strategies among these dedicated servers, as well as the communications among the clients sharing the same dedicated server for reducing the bandwidth burden. The pro of using dedicated servers is that the number of clients each dedicated server supports can be very large. The cons are that (1) it will create additional costs running a VoIP system; (2) the dedicated servers may not be flexibly and optimally located if clients in the conference are scattered geographically; (3) if a dedicated server fails, the whole multi-party conference system fails.

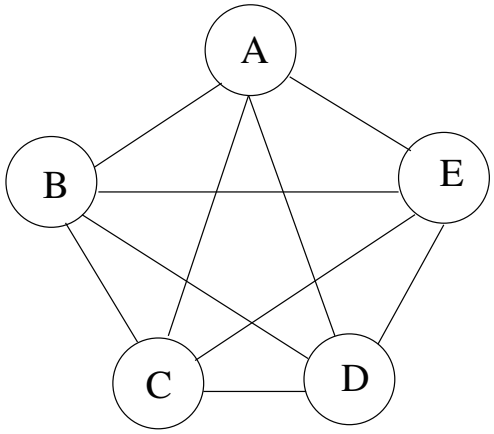


Figure 2.1: A decentralized scheme: full mesh topology with $M = 5$.

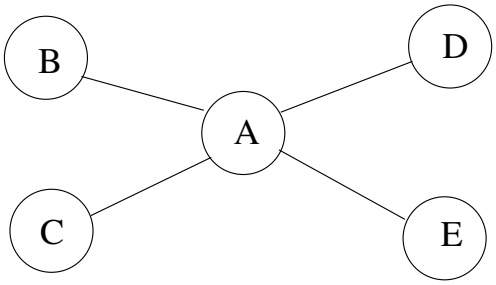


Figure 2.2: A centralized topology with $M = 5$.

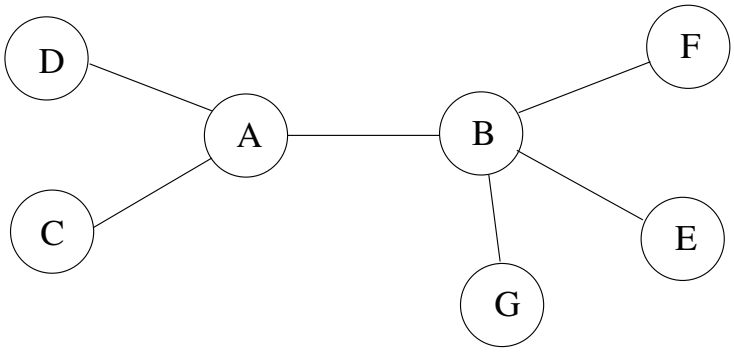


Figure 2.3: An overlay topology with $M = 7$ and two parents.

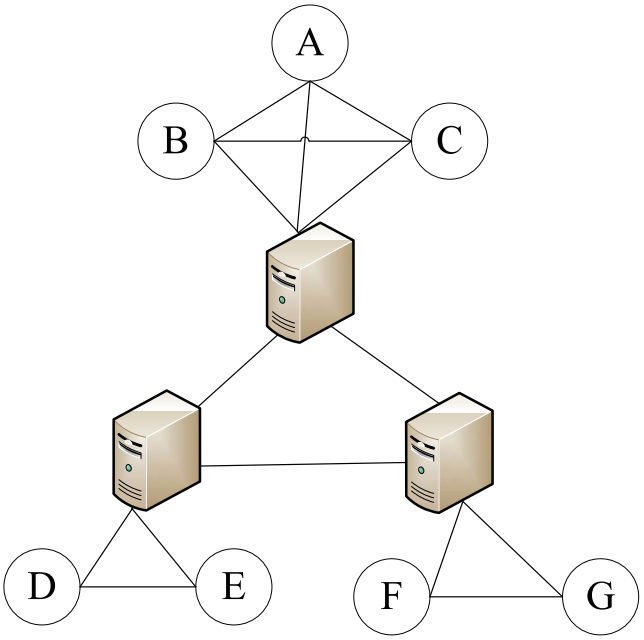


Figure 2.4: A topology with a dedicated server.

We do not want to utilize additional dedicated servers in our study. Hence, the overlay network becomes our top choice on account of its flexibility. When we design a new overlay network, we consider a topology that can minimize MS variations, reduce jitters and loss probability, and limit the network and CPU burden on the parent nodes.

2.2.2 Jitter buffer and play-out scheduling

In order to smooth the irregular arrivals of packets, play-out scheduling with a proper jitter control algorithm has to be used in a VoIP system design. There have been many studies related to this topic for a two-party VoIP system, which can broadly be divided into three categories.

Non-adaptive play-out scheduling

In this simple algorithm [31], each VoIP client sends probing packets to each VoIP client participating in the multi-party conversations during the establishment of a call (e.g., first 3 seconds). Each client then calculates the mean *end-to-end delay* (EED) of packets received from each other client. During the entire call, each client plays the received packets from a particular speaker by maintaining the MED of α sending periods more than the initially calculated network

delay that corresponds to the current speaker.

$$\mathrm{MED} = \mathrm{EED} + \alpha * T_{\mathrm{period}} \tag{2.3}$$

If all VoIP clients are within a local area and there are seldom any jitters between two nodes, this algorithm is perfect in terms of simplicity and practicality. However, for clients scattered throughout the world, jitters are more likely to exist. A fixed jitter buffer size cannot adapt to large variations of delays and jitter sizes. If α is set to be too small, packets may arrive after their scheduled play-out time. On the contrary, if α is very large, each VoIP user may have to wait a long time before he receives the utterance, which will still degrade the perceptual listening speech quality.

Adaptive play-out scheduling

Since network delay conditions can change during a call, two-party VoIP systems commonly employ adaptive play-out scheduling schemes [32, 33, 34]. A commonly used approach is to collect previous network delay statistics to decide on the MED; therefore, it is also called histogram-based adaptive play-out scheduling.

In multi-party applications, a simple extension of this approach is to collect statistics at each client with regard to packets sent from each of the speakers. Let F be the CDF of the network delay between a speaker-listener pair in the past 10 seconds, MED can be calculated from the history window:

$$\mathrm{MED} = F(\beta) + \alpha * T_{\mathrm{period}} \tag{2.4}$$

We have evaluated different β, and our results showed that $\beta = 0.98$ could best fit the dynamic network conditions if we used the 10-second history window.

Note that in a real situation, the jitter buffer size has to be adjusted during a silence period, not within a talk spurt, in order to reduce the distortion of speech segments.

Time scale modification (TSM)

The primary purpose of the TSM scheme [35, 36] is to further increase or decrease the jitter buffer size without changing the MED. The technique stretches or compresses speech frames, while its pitch period remains unchanged; therefore, it requires additional computational resources. The speech frames that TSM changes are usually located at the very beginning and the end of a speech segment, and the number of these frames cannot be very large (usually the first and last four speech frames according to the paper) to reduce listeners' perception annoyance. Therefore, it has small effects on the jitter buffer size and LOSQ.

Based on our discussions above, we implement adaptive play-out scheduling in our design because of its effectiveness in dynamic network conditions. In a multi-party VoIP system, however, the goal of minimizing the MED for each individual path may not be so crucial as in a two-party system, because the overall MED is governed by the bottleneck path with the largest delay and jitter size. Instead, MS variations plays a key role in the design. If play-out scheduling is conducted non-cooperatively at each client, MS variations may not be reduced to a large extent. Hence, we consider appropriate cooperative techniques in our study so that not only can MS variations be reduced, but jitters can also be effectively smoothed.

2.2.3 Loss concealment

There exists a number of techniques to conceal losses over the network. Basically, all methods can be divided into two categories: sender-based and receiver-based. Note that almost all sender-based schemes require a matching process at the receiver, so they are also called sender-receiver-based schemes. Figures 2.5 and 2.6 summarizes all techniques in these two categories.

The sender-receiver-based schemes include four types.

1. *Retransmission.* This technique is commonly employed in TCP [37] transmission. The receiver asks a sender to resend a lost packet. Now MED is at least three times the one-way delay T_{EED} plus the buffering time, because it takes one T_{EED} to find a lost packet, one for the receiver to inform the sender, and another for retransmission. Given the time-sensitive

nature, it is impossible to employ this technique in a real-time VoIP system.

2. *Non-redundancy-based*. This scheme commonly does interleaving [38, 39] at the senders and the receiver needs to reconstruct the packets from the interleaved frames. It exploits the fact that shorter distortions are less likely to be perceived by listeners. Therefore, if a packet is lost, the missing subframes in this packet are not consecutive and, thus, improve listeners' perception. Strictly speaking, this is actually not a loss concealment technique, because it does not recover losses. This scheme will also incur a delay of one packet-sending period before the receiver can reconstruct the interleaved packet.

3. *Redundancy-based (for partial protection)* [40, 41, 42]. In [41] and [42], layered coding is employed to divide a speech frame into several layers according to the significance of the information, and uses more bits to encode the more important layers. This scheme requires additional computation resources and will possibly distort the original speech frames.

4. *Redundancy-based (for full protection)* [14, 43, 44, 45]. [43] and [44] use forward error correction (FEC), which adds redundant information for recovering losses at the receiver. Another method [14], called piggy-backing, attaches previous frames in the current packet, so that the receiver can conceal a lost frame. It has been developed because the network bandwidth nowadays is becoming increasingly larger, and the bit rate of speech packets is no longer a bottleneck. Piggy-backing (Figure 2.7) is easy to realize at the transport layer, does not require any computational resources, and can provide best quality without the need to distort the original frames. Therefore, it is commonly used now for loss concealment at the transport layer in a VoIP system. We use several traces to evaluate its effectiveness. In a trace whose loss rate is up to 17%, a piggy-backing scheme can conceal losses and lead to a loss rate at the receiver of only 6% for two-way piggy-backing, 3% for three-way, and 2% for four-way.

The receiver-based schemes can primarily be divided into two categories.

1. *Sample-based.* In a sample-based scheme, the receiver conceals losses without the need for codec supports. Losses can be concealed using *insertion* techniques, which insert silence or

comfort noise frames [46] to replace the original frame. The previous and the next frames that are correctly received can also be used as a replacement [47]. Losses can also be recovered by *interpolation* techniques, which employ waveform substitution [48], or time scale modification [35, 36].

2. *Model-based.* This scheme employs codec parameters for repetition [49] or interpolation [50]. Usually model-based schemes provide better quality than sample-based concealments.

All receiver-based schemes are used in codec-level loss concealments, because such concealment requires information of the decoded speech frames. For packet-level loss concealment, we use the piggy-backing scheme, because of its effectiveness, shown from our analysis above. The issue of using this scheme now becomes the determination of the piggy-backing degree. This determination is based on the packet size transmitted over the Internet and the frame rate of the Internet speech codec, as is discussed in the following section.

2.3 Internet Speech Codec

The choice of speech codecs is an important part in the design of a multi-party VoIP conferencing system. Because of the unreliable nature of Internet traffic behavior, a good speech codec should adapt to dynamic network conditions and be able to recover the original waveforms with good quality, even under lossy Internet situations. A good speech codec should also compress speech at a reasonable bit rate so as to save bandwidth, both at the sender and the receiver.

Over the years, many speech codecs have been proposed in the ITU [51] and IETF [52] standards. Some of these standards, such as G.711 [53] and G.726 [54], achieve good MOS at the expense of high bit rate (64 kbps for A-law and u-law in G.711 and up to 32 kbps for G.726). Other standards, such as G.723.1 [55], requires only a very narrow bandwidth (5.3 - 6.3 kbps). These codecs were developed 10 years ago, when Internet bandwidth resources were precious. In recent years, they have gradually become obsolete in the Internet community, because the network bandwidth is no longer a critical issue.

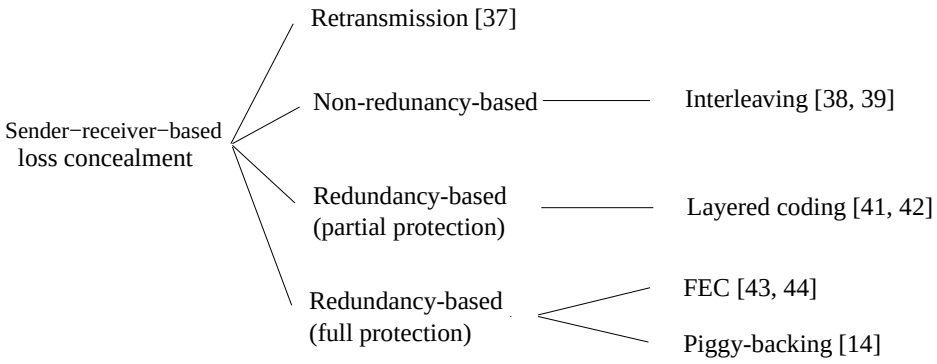


Figure 2.5: Sender-receiver-based loss concealments.

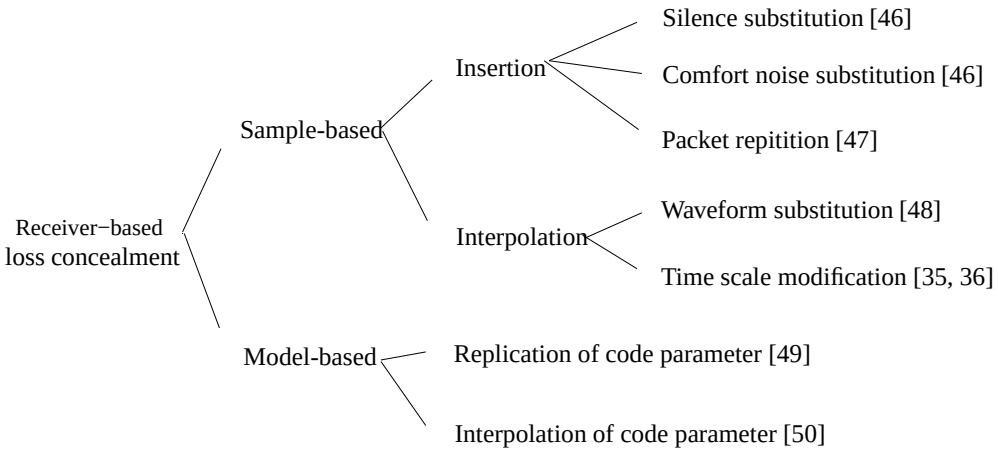


Figure 2.6: Receiver-based loss concealments.

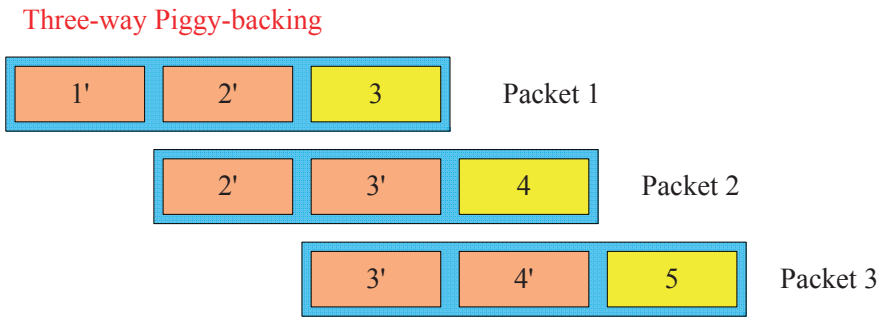


Figure 2.7: A 3-way piggy-backing algorithm for concealing losses.

In this section, we compare four popular speech codecs that are widely used in the current VoIP market. They are ITU G.729 [56] and ITU G.722.2 [57], the IETF Internet Low Bit-rate Codec (iLBC) [58] and Internet Speech Audio Codec (iSAC) [59], and the free codec for free speech, Speex [60].

2.3.1 Open-source and proprietary codecs

G.729, iLBC, and Speex are open-source codecs. They are free to users who are simply using them for educational or research purposes. G.722.2 is a proprietary codec and provides only its encoder and decoder interface for users' testing. G.729, G.722.2, iLBC, and Speex all have published user manuals. So far, we have had no access to the source code, interface, or user manual of iSAC.

2.3.2 Narrow-band and wide-band codecs

A narrow-band codec can usually encode sound whose frequencies range from 200 Hz to 3400 Hz, and a wide-band codec can encode sound ranging from 50 Hz to 7000 Hz. The Nyquist-Shannon theorem prescribes that the minimum sampling rate (f_s) is two times the bandwidth (B) of band-limited signal to avoid aliasing, as shown below:

$$f_s \geq 2 \cdot B \tag{2.5}$$

Therefore, a narrow-band codec can use a sampling rate of 8000 Hz and a wide-band codec must use a rate of 16,000 Hz. Because of this difference, a wide-band codec requires a higher bit rate (bandwidth) if the compression ratio is the same. It is still overwhelmingly preferred over a narrow-band codec because it can provide clearer sound and less distortion.

G.729 and iLBC are narrow-band codecs, while G.722.2, iSAC, and Speex are wide-band codecs. Moreover, Speex can support a sample rate of up to 32,000 Hz.

2.3.3 Packetization and frame rate

G.729 uses a fixed bit rate of 8 kbps and a frame size of 10 msec (10 bytes). iLBC has two options to select according to the network condition (15.2 or 13.3 kbps), and a frame size of 20

msec (38 bytes) or 30 msec (50 bytes) accordingly. Both G.722.2 and iSAC use adaptive bit rate for encoding and decoding. The bit rate of G.722.2 ranges from 6.6 kbps to 23.85 kbps and its frame size is 20 msec (16-61 bytes), while the rate of iSAC ranges from 10 kbps to 32 kbps and its frame size is 30$\tilde{6}$0 msec (38-120 bytes). Speex uses a larger bit rate that ranges from 4.4 kbps to 44 kbps, and a frame size of 20 msec (11-110 bytes). Because an adaptive bit rate can dynamically adjust to diverse network conditions and bandwidth, it is preferred over a fixed bit rate.

As we need to add redundant information for loss concealment at the packet level, each speech packet should be below the Maximum Transmission Unit (MTU) so as not be fragmented. ITU defines that the MTU in the Internet should be at least 576 bytes for IPv4 (normally 1500 bytes for a broadband network) and 1280 bytes for IPv6. All five codecs can satisfy this requirement, even for some degrees of piggy-backing.

2.3.4 Speech coding

G.729 and G.722.2 use algebraic code-excited linear prediction (ACELP) for speech coding. Both utilize a fixed code-book (algebraic code-book), an adaptive code-book, and a synthesis filter to complete the encoding. G.729 and G.722.2 require a 5-msec look-ahead of the next frame for coding the current frame. Because the original speech waveform is reconstructed by filtering the excitation signal through the linear prediction synthesis filter, a missing frame at the receiver may affect the overall decoder state.

iLBC, however, encodes each frame independently. It divides a frame into 6/4 subframes (30/20 msec, each 40 samples) and does linear prediction for each subframe. iLBC finds two subframes with the highest energy from the LP residues, removes the first or the last (depending on which energy is lower) 23/22 samples (30/20 msec), and accepts the remaining 57/58 samples as the initial state of the adaptive code-book. Because the coding state is solely determined by an individual frame, a missing frame at the receiver has little impact on the overall decoder state.

Speex has a 10-msec look-ahead for a narrow-band codec and a 14-msec look-ahead for wide-band version. We do not yet know how iSAC codes the speech frame, as there is no detailed documentation for this codec.

2.3.5 Loss concealment schemes in codecs

Because Internet speech packets are prone to loss, loss concealments should also be done at the codec level.

In G.729, the replacement excitation depends on the periodicity of the last reconstructed frame. If the last frame is classified as periodic, the current frame is also considered to be periodic and the adaptive code-book is used. If it is nonperiodic, the current frame is also nonperiodic and only the fixed code-book is used by randomly selecting a code-book index and a sign index. The random function is:

$$seed = seed \cdot 31821 + 13849 \tag{2.6}$$

The initial value of $seed$ is 21,845.

In iLBC, the replacement frame is generated from the pitch-synchronous repetition of the excitation signal filtered by the last linear prediction filter. For several consecutive lost frames, its result will lead to a dampened speech.

For G.722.2, a special bit in the frame should be set to indicate RX_TYPE to be SPEECH_BAD or RX_SPEECH_LOST. However, according to G.722.2 Appendix I [61], it will possibly lead to an unpleasant noise effect. A better way is to replace or interpolate previous correctly received speech frames.

According to Speex and iSAC, both codecs have the loss concealment mechanism at the codec level. We do not know the detailed implementations, though.

2.3.6 Overall considerations

To achieve the best perceptual quality, different factors need to be evaluated and balanced. The wide-band codecs G.722.2, iSAC, and Speex are preferred over narrow-band as they can provide much clearer sound. For speech coding, iLBC and iSAC are preferred because they encode each frame individually without any look ahead. We have also done an unofficial survey of the performance of G.722.2, iSAC, and Speex output at different loss rates using PESQ, and our results show that under a loss rate of less than 6%, the PESQ differences for most speech segments

are less than 0.1–0.2, which can be hardly differentiated subjectively. Moreover, Speex also offers a free preprocessing library, which can provide speech enhancement, automatic gain control, and echo cancellation mechanisms. Balancing different trade-offs, we chose to use Speex in our study and software.

2.4 A Study of the Skype Multi-Party VoIP Conferencing System

There are several existing commercial multi-party VoIP conferencing systems on the current commercial market. Several companies, like Vodafone, provide services through proprietary network and require specific hardware support. Although the proprietary network can provide more reliable voice transmission than the public Internet, the hardware limitations and high expenses restrict its popularity. The Luxembourg-based company Skype and the China-based company QQ,[1] however, allow users to conduct a multi-party VoIP conference through the public Internet, and the cost of using their software is free. Hence, they have won large popularity in the current VoIP market. In this section, we analyze the performance of Skype in multiple aspects.

2.4.1 Speech codecs and packetization

Skype employs iSAC, developed by GIPS, as its Internet speech codec. Its maximum speech frame size (corresponding to 60 msec) is no larger than 120 bytes. The codec encodes a frame without look-ahead; therefore, the effect of a lost frame will not propagate to other correctly received frames.

Our study shows that Skype adopts four framing options in multi-party conferencing: 60 ms, 45 ms, 30 ms, and 15 ms, with payloads of 246–255 bytes, 196–205 bytes, 136–170 bytes, and 96–110 bytes respectively. Our measurements indicate that, when the network has low loss and low jitters (regardless of delays), all nodes progressively increase from an initial period of around 60 ms and 32 kbps to around a 15-ms period and 50 kbps. Further, each node adaptively adjusts its rate according to the network condition. For instance, if one of the links has higher jitter, then its packet period may stay at 30 ms.

[1] QQ is a trademark of Tencent Corporation in China. URL: `http://www.qq.com`.

Skype uses an asymmetic topology in its conference. In Figure 2.8, the links between A and D are high-jitter, low-loss traces. We find that Skype sends 30-msec packets between A and D, and 15-msec packets in all the other low-jitter, low-loss links.

Our study also indicates that the clients in Skype employ silence suppression and send silence packets of around 16–21 bytes every 50 ms, as shown in Figure 2.9.

2.4.2 Topology

In Skype, the central host receives and decodes all the incoming UDP speech streams from other clients, mixes them with its own stream, and re-encodes the waveform to be sent to the clients in the conference. This is evidenced by our observation that, under no loss and jitter, the packet size and packet rate to each client are not increased when the number of simultaneous speakers is increased as shown in Figure 2.10. Another item of evidence is that the central host is generally more loaded than the other clients. The central host excludes the speech waveforms of one particular client from the mixed streams destined to this client; hence the central host has to do different mixing for all the clients.

Because of the CPU burden and the bandwidth limitation on the central client, the total number of clients in the conference cannot exceed 9 in Skype. The system is not flexible, because the client who starts the conference acts as the central host. The poor network conditions at the central host and diversity of MED will likely degrade the conversational quality of a multi-party conference.

2.4.3 Jitter buffer and play-out scheduling

We were not able to identify the POS algorithm used in Skype because its voice packets are encrypted and the source codes of the clients are not available. But we have noticed that under high jitter, Skype gradually increases the sending period in an effort to reduce network congestions. Meanwhile, it also doubles its packet size so that losses can be concealed.

2.4.4 Loss concealment

We consider two situations to measure the loss concealment in Skype. In Figure 2.11, packets sent from D to A experience high losses, and the packet payload size from D to A (central host)

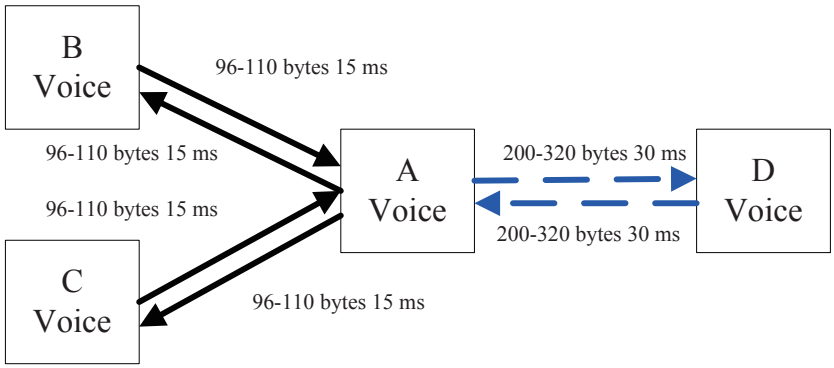


Figure 2.8: Skype: four nodes speaking simultaneously. Link A↔D: high-jitter, low-loss; all other links are low-loss, low-jitter traces.

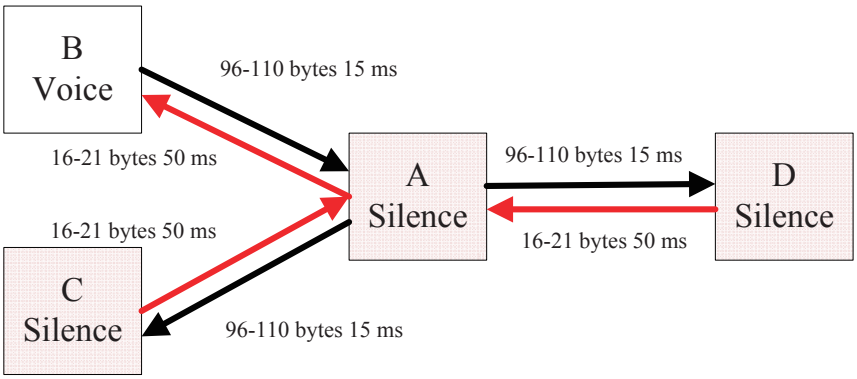


Figure 2.9: Skype: only B speaking. All links are low-loss, low-jitter traces.

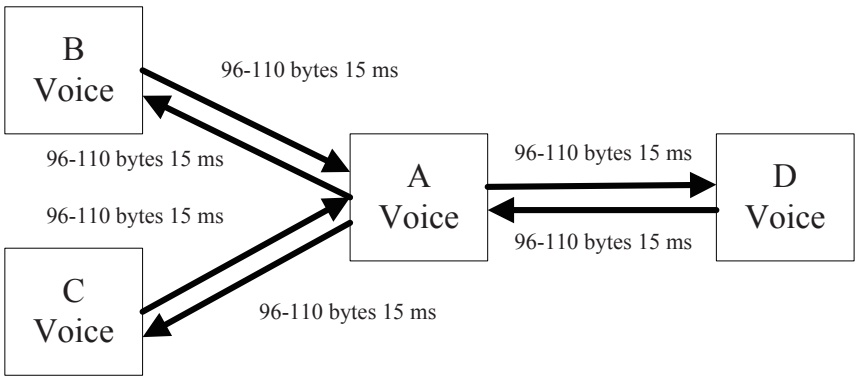


Figure 2.10: Skype: four nodes speaking simultaneously. All links are low-loss, low-jitter traces. The numbers show the average packet length (bytes) and sending rate (millisecond per packet).

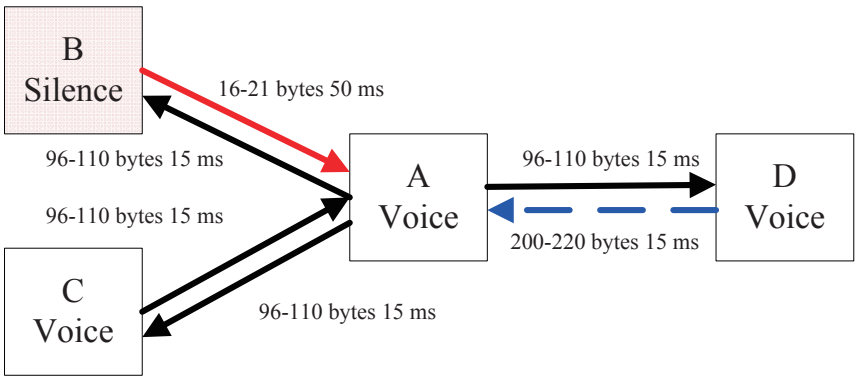


Figure 2.11: Loss concealment in Skype. Case 1: A,C,D speaking simultaneously. Link D→A: high-loss, low-jitter. All other links are low-loss, low-jitter traces.

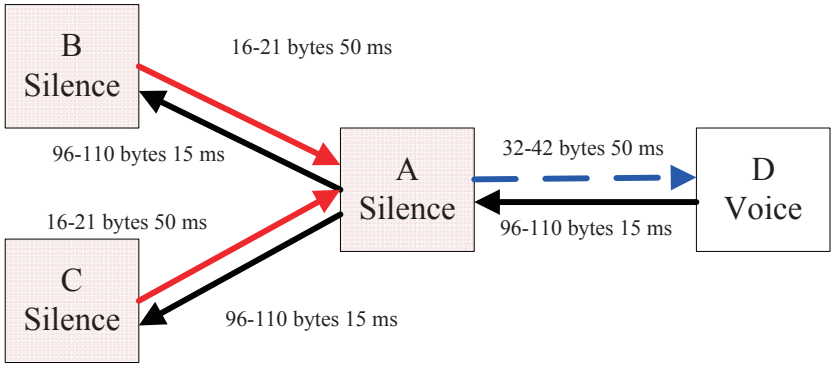


Figure 2.12: Loss concealment in Skype. Case 2: Only D speaking. Link A→D: high-loss, low-jitter. All other links are low-loss, low-jitter traces.

doubles (regardless of the average delay). But A still sends packets of the normal size to all the non-central nodes, including D. We can postulate from the payload size changes that only one-way redundancy is applied on the packets sent, and we conclude that the central host does the loss concealment before it sends the processed voices, on the basis of the changes of payload size of its incoming and outgoing packets. In Figure 2.12, only D speaks, and high losses exist from A to D. Our experiment shows that Skype does the piggy-backing also in noncentral clients. For both conditions, the packet sending rate does not change. As Skype adopts the centralized approach, it will introduce an additional delay of at least one sending period at the central host. Because of the dynamic network conditions, using only one-way redundancy may be too conservative.

2.5 Summary

In this chapter, we have presented a complete survey of related work on multi-party VoIP conferencing systems. We have discussed various studies and standard measures by focusing on the conversational quality, and we have analyzed their limitations when dealing with trade-offs between LOSQ and delays. We have also provided an all-around analysis of different schemes used in multi-party VoIP designs and their pros and cons. Four Internet speech codecs have been compared, as the speech codec is one of the key factors affecting LOSQ. At the end of this chapter, we studied Skype's behavior and its strategies under different network conditions. Understanding all these related studies and works will help better design a multi-party VoIP system.

CHAPTER 3

ANALYSIS OF INTERNET BEHAVIOR

In this chapter, we analyze Internet traffic behavior in terms of end-to-end delays, jitters (the variations of packet arrivals from the average delay) and losses using real traffic data collected from PlanetLab. Various impacts on the multi-party VoIP conferencing system are studied. Our implementations of Linux kernel modification for the Internet are discussed.

3.1 Objective

The current Internet has a significantly unreliable nature. Speech packets may be delayed and dropped because of dynamic changes of Internet conditions. As there are multiple clients in a VoIP conference, network traffic exhibits more diversities than a two-party call does. The Transmission Control Protocol (TCP) can provide reliable and in-order service by employing packet re-delivery and congestion control mechanisms to overcome the Internet problems. In other words, its quality of service (QoS) in terms of receiving complete voice streams is guaranteed. TCP implementation, however, makes it possible to wait an indefinitely long time for some packets to arrive, which severely violates the real-time nature and deadline restriction of a VoIP system; thus, TCP is not used in our design. Rather, User Datagram Protocol (UDP) is adopted in our VoIP conferencing system because it delivers packets at its best effort and does not guarantee reliability at the cost of creating more delays. Since dynamic delays, jitters, and losses over the Internet degrade the performance of VoIP clients, it is worthwhile to collect and look at the traffic data. These data are later used in our VoIP test beds to simulate a real Internet environment.

This chapter presents the Internet traces by grouping them into similar traffic patterns. Its main goal is to study the impact of different patterns and relate them to the multi-party VoIP conferencing system design. Descriptions of kernel modifications are also presented for our

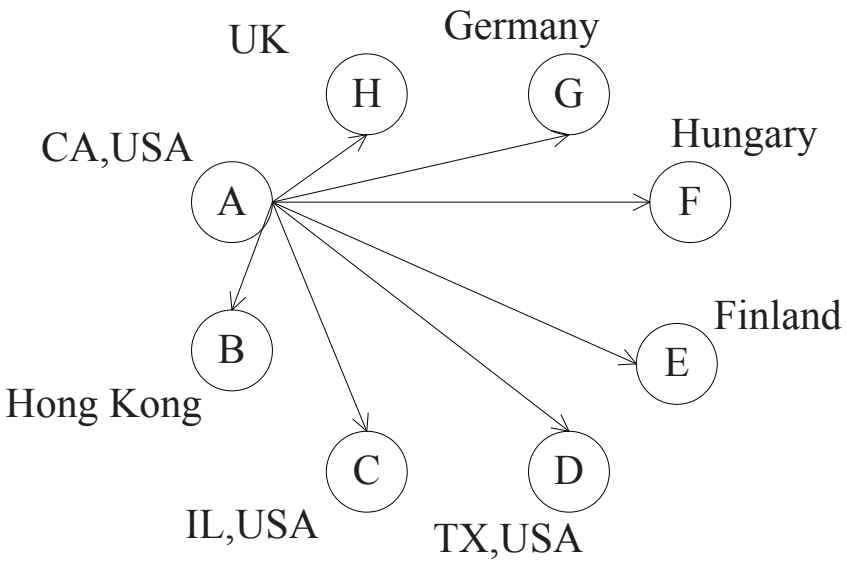


Figure 3.1: Topology of Internet traffic collection in PlanetLab.

Internet environment simulation.

3.2 Collection of Internet Traffic

Internet traffic data were collected through PlanetLab, where there are consistently around 200-300 active nodes in the world-wide overlay. These nodes are scattered over the five continents served: North and South America, Asia, Europe, and Australia. In order to collect diverse traffic patterns, we choose multiple nodes in all continents (except Australia, where there were no more than two active nodes during the time we conducted our experiment). Table 3.1 list all 60 nodes (20 in Asia, 20 in the Americas, 18 in Europe, and 2 in Australia) that we have used for the traffic collection. Both intracontinental and intercontinental traces are taken into account for diversity purposes.

In our experiments, only one-way end-to-end traffic was collected. Every eight nodes formed a trace set, and in each set, packets were sent from one node to all the other nodes simultaneously using point-to-point UDP packets every hour over a 24-hour period, as shown in Figure 3.1. To avoid network congestion and traffic disturbances, no other nodes in this group were sending packets at the same time.

Table 3.1: PlanetLab nodes used to collect Internet traffic in 2007 and 2008.

Continent	Country	Host Name
Asia	China	planetlab1.iin-bit.com.cn thu1.6planetlab.edu.cn pku1.6planetlab.edu.cn uestc1.6planetlab.edu.cn dlut1.6planetlab.edu.cn ustc1.6planetlab.edu.cn ustc2.6planetlab.edu.cn sjtu2.6planetlab.edu.cn tongji1.6planetlab.edu.cn xmu2.6planetlab.edu.cn xjtu1.6planetlab.edu.cn
	Hong Kong	planetlab3.ie.cuhk.edu.hk
	Taiwan	planetlab1.ntu.nodes.planet-lab.org pads23.cs.nthu.edu.tw
	Japan	planet1.jaist.ac.jp pl1-higashi.ics.es.osaka-u.ac.jp
	Korea	pl2.snu.ac.kr
	Israel	ds-pl2.technion.ac.il planet1.cs.huji.ac.il
	India	planetlab1.iitr.ernet.in
Americas	United States	planet1.scs.stanford.edu planetlab1.cs.ucla.edu planetlab12.millennium.berkeley.edu planetlab13.millennium.berkeley.edu planetslug1.cse.ucsc.edu planetlab1.cs.uchicago.edu planetlab1.mnlab.cti.depaul.edu planetlab1.cs.umass.edu planetlab2.csail.mit.edu planetlab1.cnds.jhu.edu planetlab1.cs.dartmouth.edu planetlab2.cs.columbia.edu planetlab1.cs.uoregon.edu vn1.cs.wustl.edu planetlab01.cs.washington.edu planetlab1.cs.wisc.edu
	Canada	cs-planetlab1.cs.surrey.sfu.ca planetlab2.win.trlabs.ca
	Brazil	planetlab1.lsd.ufcg.edu.br
	Uruguay	planetlab-1.fing.edu.uy
Europe	Czech	planetlab1.cesnet.cz planetlab2.cesnet.cz
	Denmark	planetlab1.diku.dk
	Finland	planetlab1.hiit.fi
	Germany	mars.planetlab.haw-hamburg.de planet1.zib.de planet2.zib.de planetlab1.itwm.fhg.de

Continued on next page

Table 3.1: Continued.

Continent	Country	Host Name
Europe	Hungary	planetlab1.tmit.bme.hu planetlab2.tmit.bme.hu
	Netherlands	planetlab1.cs.vu.nl
	Portugal	planetlab-1.di.fc.ul.pt
	Switzerland	planetlab2.inf.ethz.ch planetlab01.ethz.ch
	United Kindom	planetlab1.xeno.cl.cam.ac.uk planetlab2.xeno.cl.cam.ac.uk planetlab-1.imperial.ac.uk planetlab2.aston.ac.uk
Australia	Australia	plnode01.cs.mu.oz.au plnode02.cs.mu.oz.au

Table 3.2: NTP servers used for synchronization.

Location	IP Address	Host Name
Americas	time.nist.gov	192.43.244.18
Asia	ntp.time.ac.cn	210.72.145.44
Europe	ntp2.npl.co.uk	139.143.5.31

We used both 20-ms and 30-ms packet periods in order to match the sending rate in VoIP transmissions. As it was important to measure the latencies packets took to travel from the sender to the destinations, each packet carried in its payload a local timestamp that was synchronized every 10 minutes by a nearby NTP time server. We used three local NTP servers, one in each continent, in our experiments as shown in Table 3.2.

Let t_1 and t_2 be the local time of the sender and the receiver, Δt_1 be the offset of the sender from its nearby NTP server, and Δt_2 be that of the receiver. The one-way delay between these two nodes is:

$$\mathrm{DL} = (t_2 - \Delta t_2) - (t_1 - \Delta t_1) \tag{3.1}$$

Our scheme assumes that the various NTP servers are synchronized to within some small tolerance (usually within 10 msec according to the statistics we obtained in our previous experiments) and that each client has compensated for round-trip delays between itself and the nearby NTP server. Although this scheme does not guarantee that all local clocks are perfectly synchronized, the errors incurred are small enough, compared to the one-way delay between two clients.

The errors are also expected to be smaller than those of a simple scheme that computes the one-way delay as half of the round-trip time (RTT) between two nodes, because two-way delays are usually not symmetric. Jitters and losses on one trip are usually not related to the reverse trip, because the packets in two directions travel through different paths.

Another reason of using one-way latency is that delay variations of different traces from the same node at the same time may not be correct. Let $\delta_1, \ldots, \delta_n$ be the deviations from the average delays of different traces from the same node to different nodes, and $\delta_1^{'}, \ldots, \delta_n^{'}$ be the deviations of the reverse traces at the same time. One-way deviations obtained from RTT are $(\delta_1+\delta_1^{'})/2, \ldots, (\delta_n+\delta_n^{'})/2$. Using these data may disturb the disparities among the traces and thus probably lead to a wrong classification (as shown in the next section).

3.3 Classifications of Internet Traces

We use two methodologies to classify Internet traces to facilitate analysis and simulation. One classification is based on the traffic patterns from one particular source, and the other is based on the patterns from all nodes in the trace sets. This first one can help understand the diversity and correlation of Internet traffic sent from one node at the same time to other nodes. The latter can help the overall consideration of our conferencing system topology and play-out scheduling algorithm.

Table 3.3 shows the statistics of 11 sample trace sets (one trace set in each category) collected from one particular source to seven destinations as shown in Figure 3.1. For each trace set, we list the minimum and the maximum average delays, jitter sizes, and loss rates.

Table 3.4 shows the complete statistics of seven trace sets that fall into five categories. It includes traffic sent from every node to all other nodes in a set. The minimum and the maximum average delays, jitter sizes, and loss rates are also listed. We use these seven trace sets in our system design and repeatable measurements of multi-party VoIP prototypes under different network conditions.

There are several observations on the data we have collected.

First, the traces have large variations in their delays, jitters, and losses that depend on the time they were collected. This is understandable, as nodes may experience high traffic during business

Table 3.3: Internet traces collected in July and August 2007 from one source to seven destinations (duration 10 min; packet period 30 ms).

Set	Type	DL JT LR (L/H/M)			Hour (CST)	Source Location	Dest. (S,A,U)	Mean DL (ms)		JT30 (%)		JT60 (%)		LR (%)	
								Min	Max	Min	Max	Min	Max	Min	Max
A1	Uniform	L	L	L	20:00	CA,USA	(1,2,4)	42.2	94.6	0.00	0.23	0.00	0.15	0.00	0.00
A2	Uniform	H	L	L	18:00	China	(0,3,4)	107.3	190.4	0.03	4.2	0.00	3.5	0.00	0.01
A3	Uniform	H	L	H	23:00	Hong Kong	(0,3,4)	101.2	204.3	0.02	1.8	0.00	1.64	14.7	22.7
A4	Uniform	H	H	L	22:00	Taiwan	(1,3,3)	198.0	280.4	74.7	76.5	68.3	72.2	0.14	0.22
A5	Non-unif	M	L	L	20:00	Czech	(2,3,2)	56.0	158.4	1.8	2.3	0.45	0.97	0.00	3.39
A6	Non-unif	M	H	L	17:00	CA,USA	(2,2,3)	74.9	170.9	27.8	48.2	5.2	6.2	0.00	4.33
A7	Non-unif	M	L	H	1:00	Hong Kong	(1,3,3)	85.4	195.9	0.01	1.9	0.00	1.6	15.3	22.8
A8	Non-unif	M	L	M	11:00	Canada	(2,2,3)	52.4	147.3	0.00	0.86	0.00	0.83	0.00	16.9
A9	Non-unif	M	M	L	5:00	UK	(2,3,2)	26.5	139.9	0.01	8.11	0.00	8.10	0.00	3.2
A10	Non-unif	H	M	M	1:00	China	(0,4,3)	103.7	198.9	2.7	12.6	1.2	6.6	1.9	8.6
A11	Non-unif	M	M	M	8:00	Hungary	(3,2,2)	22.6	190.6	0.02	79.8	0.00	79.0	0.00	25.1

Abbreviations: DL: delay; JT: jitter; JT30: jitters larger than 30 ms with respect to mean delay; JT60: jitters larger than 60 ms with respect to mean delay; and LR: loss rate). Delays are classified into low (less than 100 ms), high (larger than 100 ms), and mixed (a combination of both). Similarly, jitters are classified into low (less than 5% in JT60), high (greater than 5% in JT60), and mixed; and losses into low (less than 5%), high (greater than 5%) and mixed. The delay, jitter and loss behavior of the different receivers is characterized by Type into uniform and non-uniform. The destination nodes are listed using a triplet of three numbers (number in aSia, number in the Americas, number in eUrope).

hours, especially in the afternoon, but be idle in the middle of the night.

Second, there may be large disparities in delays, jitters, and losses across the destinations for packets sent from a source. The behavior tends to be more uniform across destinations in the same continent but have larger disparities across continents. For example, packets in Trace A11 from Hungary to nodes in Europe have less than 100 ms of average delay and little jitters. However, the same stream to Asia has over 120 ms of average delay and has jitters and losses. The complete statistics in A11 are shown in Table 3.5.

Third, the behavior of packets sent from one source to multiple destinations may be correlated. Figure 3.2 shows that the delays of packets sent from Taiwan to five destinations in Asia, the Americas, and Europe in Trace Set A4 are strongly correlated. Such correlations were likely caused by congestion in the vicinity of the source node. All of the traces from Taiwan in this set are experiencing high jitters. The jitter sizes as well as the time these jitters took place, were strongly correlated. This is possibly because there were congestions at the source nodes, so that jitter and loss patterns were correlated in all the traces from these nodes. In contrast, Table 3.5

Table 3.4: Internet traces collected in 2007 and 2008.

#	Loc	DL/JT/LR (L/H/M)	Avg DL(ms)		JT60(%)		LR(%)	
			Min	Max	Min	Max	Min	Max
B1	CA,US	L/L/L	45	92	0.2	3.6	0.0	0.1
	IL,US		45	63	0.0	2.4	0.0	0.0
	Germany		28	92	0.0	2.4	0.0	0.2
	MD,US		58	90	2.4	2.6	0.0	0.0
	UK		29	88	0.0	2.5	0.0	0.2
B2	NY,US	L/L/L	26	52	0.0	0.0	0.0	0.0
	OR,US		25	60	0.0	0.0	0.0	0.0
	TX,US		26	31	0.0	0.0	0.0	0.0
	CA,US		11	39	0.0	0.0	0.0	0.0
	MO,US		17	54	0.0	0.0	0.0	0.0
B3	BJ,CN	M/L/L	50	284	0.4	0.6	0.0	0.0
	IL,US		120	219	0.0	0.2	0.0	0.0
	Hungary		120	290	0.4	0.7	0.0	0.0
	SH,CN		83	301	0.1	2.8	0.0	0.1
	Taiwan		131	319	0.0	7.5	0.2	0.3
B4	SD,CN	M/L/L	22	242	0.0	0.9	0.1	1.4
	Japan		70	226	0.0	0.0	0.0	0.5
	TJ,CN		27	244	0.0	0.0	0.0	1.1
	TX,CN		124	165	0.0	0.0	0.0	0.0
	Uruguay		121	242	0.0	0.1	0.0	0.0
B5	CA,US	L/L/M	42	178	0.0	0.1	0.0	0.0
	Canada		53	148	0.0	0.0	0.0	3.6
	HK		101	131	0.0	1.3	14.3	17.1
	NH,US		49	129	0.0	0.1	0.0	0.2
	AH,CN		97	194	0.0	0.0	0.0	0.1
B6	BJ,CN	L/M/M	104	199	0.1	5.3	1.9	8.6
	UK		88	132	0.0	0.1	0.0	0.4
	TX,US		88	163	0.0	2.9	0.0	2.6
	Canada		64	199	0.0	1.4	0.0	1.1
	SX,CN		107	190	0.0	2.8	0.0	0.0
B7	Canada	L/M/L	58	202	0.0	2.2	0.0	0.7
	India		248	352	12.2	12.9	3.7	4.2
	CA,US		32	185	0.0	0.8	0.0	0.4
	SC,CN		46	301	0.0	0.0	0.0	0.5
	AH,CN		33	296	0.0	0.0	0.0	0.5

Abbreviations: DL: delay; JT: jitter; JT60: jitters larger than 60 ms with respect to mean delay; and LR: loss rate. Delays are classified into Low (less than 100 ms), High (larger than 100 ms), and Mixed (a combination of both). Similarly, jitters are classified into Low (less than 5% in JT60), High (greater than 5% in JT60), and Mixed; and losses into Low (less than 5%), High (greater than 5%) and Mixed.).

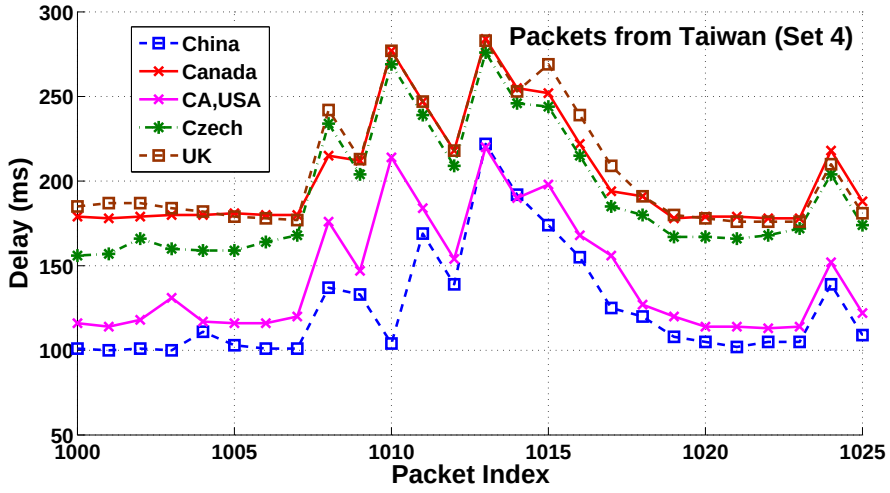


Figure 3.2: Delay behavior of packets collected from Taiwan to Xian (China), Canada, California (United States) and Czech Republic at 1:00 CST in August 2007 (Trace Set A4).

Table 3.5: Traffic behavior of packets collected from Hungary in Trace Set A11 at 1:00 CST in July, 2007.

Destination	Min DL	Avg DL	Max DL	JT30	JT60	LR
Hong Kong	133 ms	190.6 ms	1529 ms	79.8%	79.0%	0.00%
China 1	121 ms	150.3 ms	1495 ms	77.4%	76.1%	0.00%
China 2	117 ms	147.4 ms	1483 ms	76.6%	75.4%	0.00%
Berkeley	90 ms	90.8 ms	126 ms	0.02%	0.00%	0.00%
Canada	60 ms	61.0 ms	100 ms	0.00%	0.00%	0.00%
Finland	24 ms	25.7 ms	64 ms	0.03%	0.00%	0.00%
Portugal	21 ms	22.6 ms	193 ms	0.00%	0.00%	25.14%

Abbreviations: DL: delay; JT: jitter; JT30: jitters larger than 30 ms wrt mean delay; JT60: jitters larger than 60 ms with respect to mean delay; and LR: loss rate.).

illustrates that packets sent from Hungary in Trace set A11 experienced high jitters to destinations in Asia. Such correlations were likely caused by congestions in the links between Europe and Asia.

3.4 Linux Kernel Modifications for Simulation

In order to measure the performance of any VoIP software easily, systematically, and repeatably, we have created an intermediate router to simulate the real Internet environment for the purpose of measuring the performance of VoIP clients. Through Linux kernel modifications (Linux version 2.4.26), this router creates exactly the same traffic patterns as the Internet data

collected from PlanetLab. Whenever a packet passes through the router, this packet will be delayed or dropped.

There are several ways to build up such a router. One simple way is to directly create a link list in the kernel, which can be implemented by modifying the *ip_forward()* function at the link layer. When a packet arrives at the intermediate router, the kernel either drops it or creates delay according to the input Internet traffic patterns. If the kernel is scheduled to delay the packet, it calculates the estimated release time by adding the arrival time and expected delay together. The kernel temporary holds the packet in the link list, with the estimated release time added to the fake header. This link list is triggered again when the next packet arrives. All packets that have passed the estimated release time will be removed from the link list and sent to the remote destination. There are several problems, however, with this implementation. First, the kernel may be overburdened, especially if a huge number of multimedia packets arrive during a very short amount of time. The kernel may not have enough memory space to allocate for these packets, and additional packets will be dropped immediately. Second, the link list in the kernel is triggered only by newly-arrived packets. If no new packets arrive, all the old packets will remain in the link list forever. The advantage of this implementation is that the router can delay and drop packets of any transport-layer protocols.

In this section, we propose an alternative implementation that creates delay and loss patterns at the application-level software, which overrides the limitation of the kernel. The only restriction of this new method is that only UDP packets can be delayed and dropped in the router. As most VoIP systems, such as Skype, QQ, and our VoIP client, use UDP for packet transmissions, this alternative implementation works well in our experiment.

Figure 3.3 shows the overall kernel design of our router. It can be divided into three stages:

- Kernel level: processing incoming packets
- Application level: dropping and delaying packets
- Kernel level: processing outgoing packets

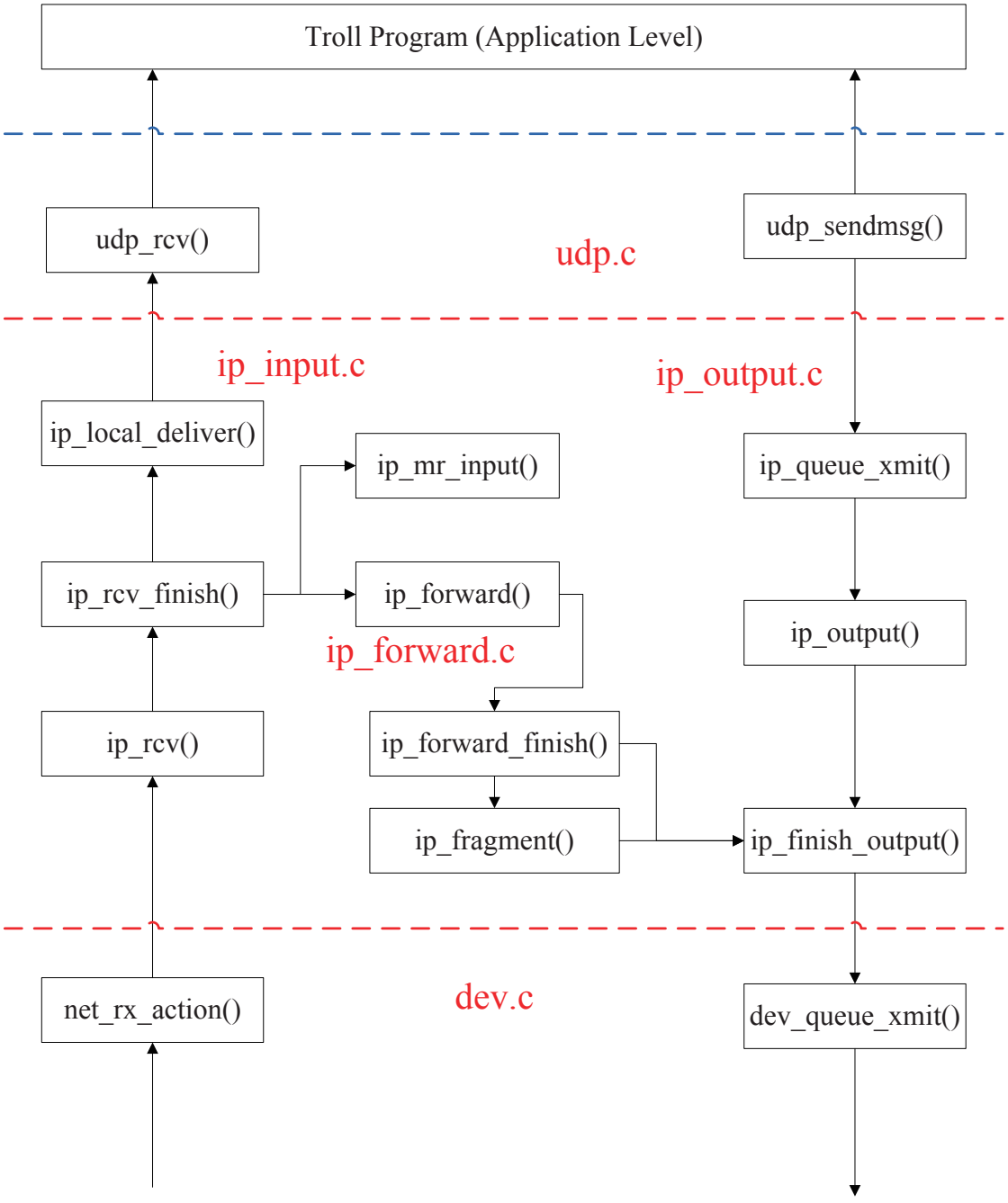


Figure 3.3: Overall kernel design of router.

Table 3.6: Mapping table implemented in the intermediate router.

Source		Destination	
IP Addr	Port	IP Addr	Port
130.126.142.56	10080	130.126.142.55	10023
130.126.142.56	10080	130.126.142.52	10076
130.126.142.51	10123	130.126.142.52	10076
...	...	...	...

3.4.1 Kernel level: Processing incoming packets

The main purpose of this stage is to change the destination IP address of a valid speech packet and to pass it to the local application-level software that creates delay and loss patterns. Specifically, a mapping table that saves the IP addresses of all valid VoIP clients is preloaded into the kernel through the *proc* entry. When a speech packet arrives, the kernel checks to see whether it is from a valid client in the mapping table. If it is, the kernel creates a fake IP header for the packet with the destination IP and port number set to its local address and listening port number. The kernel also passes the packet to the application-level software.

The mapping table is implemented in such a way that one pair of source and destination IP addresses and port numbers form a map as shown in Table 3.6. Since VoIP clients usually use only one port to receive speech packets from all other nodes, the port numbers of two source clients cannot be the same when the source IP address is looked up successfully.

The pseudocode of this part is shown in Code 1.

As shown in Figure 3.3, the functions in this step have the following usage:

- **net_rx_action()** receives a packet from the device and delivers it to **ip_rcv()**.
- **ip_rcv()** rejects irrelevant packets and examines the IP header and checksum of the packets. After it is done, it delivers the packets to **ip_rcv_finish()**.
- **ip_rcv_finish()** forces a valid VoIP UDP packet, which would otherwise be passed to **ip_forward()**, to be delivered to **ip_local_deliver()**. **ip_local_deliver()** simply forwards the packet to **udp_rcv()**.
- **udp_rcv()** keeps a VoIP table that maps from the source IP address and port number to the

Code 1 Kernel level: processing incoming packets

```
1:  if Packet whose destination IP is not the local address then
2:     if UDP Packet from a valid VoIP client then
3:        Forward the packet to udp_rcv() through ip_local_deliver()
4:        Look up the mapping table using source IP address of the VoIP packet
5:        if First VoIP packet from this IP address then
6:           Add source and destination IP address and port to the mapping table
7:        else
8:           if Source port is different from returned port number in mapping table then
9:              Issue a conflict warning
10:          end if
11:       end if
12:       Reserve space for fake UDP header using skb_push()
13:       Set destination IP to local IP address
14:       Calculate the listening port number of application-level troll program
15:       Set destination port number to the calculated result
16:       Forward the packet to the application
17:    else
18:       Forward the packet using ip_forward()
19:    end if
20: else
21:    Forward the packet using ip_forward()
22: end if
```

destination IP address and port number. It adds a fake header to the packet and forwards it to the application-level software.

3.4.2 Application level: Dropping and delaying packets

We have modified an existing troll program that was originally designed to automatically generate delay and loss patterns for a specific end-to-end link. The application-level troll program now reads a collection of traffic data collected from PlanetLab and simulates the Internet accordingly. The software validates each speech packet, pulls out the original destination IP address and port number, and uses a timer for the packet delay. It also sends the released packet using the same port number as the original source port.

The pseudocode of this part is shown in Code 2.

Code 2 Application level: dropping and delaying packets

```
1: Read real traffic pattern (delay and loss) from the saved file to the array
2: Create listening socket for incoming packets
3:
4: Thread 1
5: for each new incoming packet do
6:     Read current value from traffic array
7:     if It is a loss (indicated by −1) then
8:         Drop the packet
9:     else
10:        Set the timer to the delay number (indicated in milliseconds)
11:        Add the VoIP packet to the timer list
12:    end if
13: end for
14:
15: Thread 2
16: for each packet whose timer has expired do
17:    Pull out the VoIP packet from timer list
18:    Remove the original UDP header of the packet
19:    Set sending socket port to be the same as original source port of the packet
20:    Set destination IP address and port to be the same as original
21:    Send out the VoIP packet
22: end for
```

3.4.3 Kernel level: Processing outgoing packets

This stage looks up the mapping table using the destination IP address and port number as well as source port number. Since the source and destination pair is unique in the table, the original source IP address can be found. The kernel then changes the source IP address in the header back to the original address.

The pseudocode of this part is shown in Code 3.

Code 3 Kernel level: processing outgoing packets

```
1: for each VoIP packet from the application-level software do
2:     Look up the mapping table using the destination IP address and port number as well as source
       port number
3:     Change the source IP address from local address to the original address
4:     Pass the packet to ip_queue_xmit()
5: end for
```

As shown in Figure 3.3, the functions in this step have the following usage:

- **udp_sendmsg()** adds the source and destination IP addresses and port numbers to the packet

header and passes the packet to **ip_build_xmit()**, which is a fast path for nonfragmented packets.

- **ip_build_xmit()** looks up the packet information in the mapping table and change the packet back to the original source IP address.
- **ip_finish_output()** initializes the last tasks of the Internet protocol and specifies the output network device.
- **dev_queue_xmit()** sends packets out using the selected network device.

3.5 Summary

In this chapter, we have studied the impact of the Internet behavior on a multi-party VoIP conferencing system. We collected real Internet traffic from the PlanetLab by sending UDP streams from one node to all other nodes simultaneously. The Internet traffic is classified into 11 categories according to delay, loss and jitter statistics.

We have analyzed the different categories of Internet traces. Our findings show that the Internet behavior has diversity and disparity across both time and destinations. However, the traffic patterns may show some correlations for packets sent from the same node at the same time. Such behaviors will help us determine the optimized topology, play-out scheduling, and loss concealment strategy for a multi-party VoIP conferencing system.

We propose the implementation of an intermediate router to create losses, delays, and jitters. It is used to simulate a real Internet environment and facilitate the measurement of real-time VoIP clients in a systematic and repeatable manner. We will use this router to evaluate our system and Skype in the next chapters.

CHAPTER 4

DESIGN AND EVALUATION OF A MULTI-PARTY VOIP CONFERENCING SYSTEM

In this chapter, we study the multi-party conversational model and propose several objective metrics that describe the MS variations. Based on the model, we design our multi-party VoIP system. We present a new method to determine the optimized conferencing topology and distributed equalization algorithm to reduce MS variations. We also illustrate several issues on the implementation of our prototype. At the end of the chapter, we propose a classifier approach for generalizing the result to unseen condition. The approach can be used to select the algorithm for achieving the best perceptual conversational quality.

4.1 Roadmap of This Chapter

The roadmap of this chapter is shown in Figure 4.1. We propose several objective metrics from the multi-party conversational model, including *conversational symmetry* (CS), *conversational efficiency* (CE), and *consecutive mutual silence ratio* (CMSR), that can capture the MS variations and impact the subjective conversational quality. Based on the understanding of these metrics, we study our VoIP conferencing system design. Two core components are the optimized overlay topology, for reducing the MED diversity, and the play-out scheduling algorithm, for smoothing jitters and reducing MS variations. Other practical issues include silence detection, loss concealment strategy and mixing policy. Since there are trade-offs between LOSQ and MS variations, it is hard to determine their relations to the subjective quality using a simple model. We extract all objective metrics that may impact the subjective opinions. Along with the subjective results we have collected so far, the experiment data are trained using a *support vector machine* (SVM) learning classifier [62] which can effectively find a mapping from the input features (objective metrics) to the output (subjective ratings). The learning model can later be used to

generalize the data to unseen network and conversational conditions.

4.2 Multi-Party Conversational Model

We begin with the two-party conversational model. As has been illustrated in Chapter 1, MS can be perceived as alternating short and long silence durations between turns in a two-party VoIP conversation. Generally, they can be divided into two categories as shown in Figure 4.2.

- ***Human Response Delay (HRD).*** After hearing the previous speech turn from A in a two-party conversation, the other party, B, waits for HRD_B before he gives a response to A. This duration is specified as $\mathrm{MS}_B^{A\rightarrow B}$, where $A \rightarrow B$ means that the speech turn shifts from A to B, and MS_B means the MS from B's perspective.

- ***Response Mutual Silence (RMS).*** After A in a two-party conversation gives an utterance, he needs to wait for a RMS, indicated by $\mathrm{MS}_A^{A\rightarrow B}$ before he receives a response from the other party B. $\mathrm{MS}_A^{A\rightarrow B} = \mathrm{MED}_{A,B} + \mathrm{HRD}_B + \mathrm{MED}_{B,A}$, where $\mathrm{MED}_{A,B}$ is the delay from the mouth of the speaker A to the ear of the listener B. MED usually includes three parts: the sender processing delay, Internet propagation delay, and the receiver buffering delay.

Note that because of MED, the longer and shorter MSs have led to an asymmetry in a two-party conversation.

The extension of a VoIP system from two-party to multi-party is not straightforward. A multi-party conversation (Figure 4.3) not only includes the speaker-and-response pair (the same as a two-party conversation); it also includes a third type of clients who are simply listening to the speaker-and-response pair. We call these *passive listeners*. The MS incurred on passive listeners is named *listener mutual silence* (LMS). The three types of MSs in a multi-party conversation are illustrated below.

- ***Human Response Delay (HRD).*** $\mathrm{MS}_C^{A\rightarrow C} = \mathrm{HRD}_C$.

- ***Response Mutual Silence (RMS).*** $\mathrm{MS}_A^{A\rightarrow C} = \mathrm{MED}_{A,C} + \mathrm{HRD}_C + \mathrm{MED}_{C,A}$.

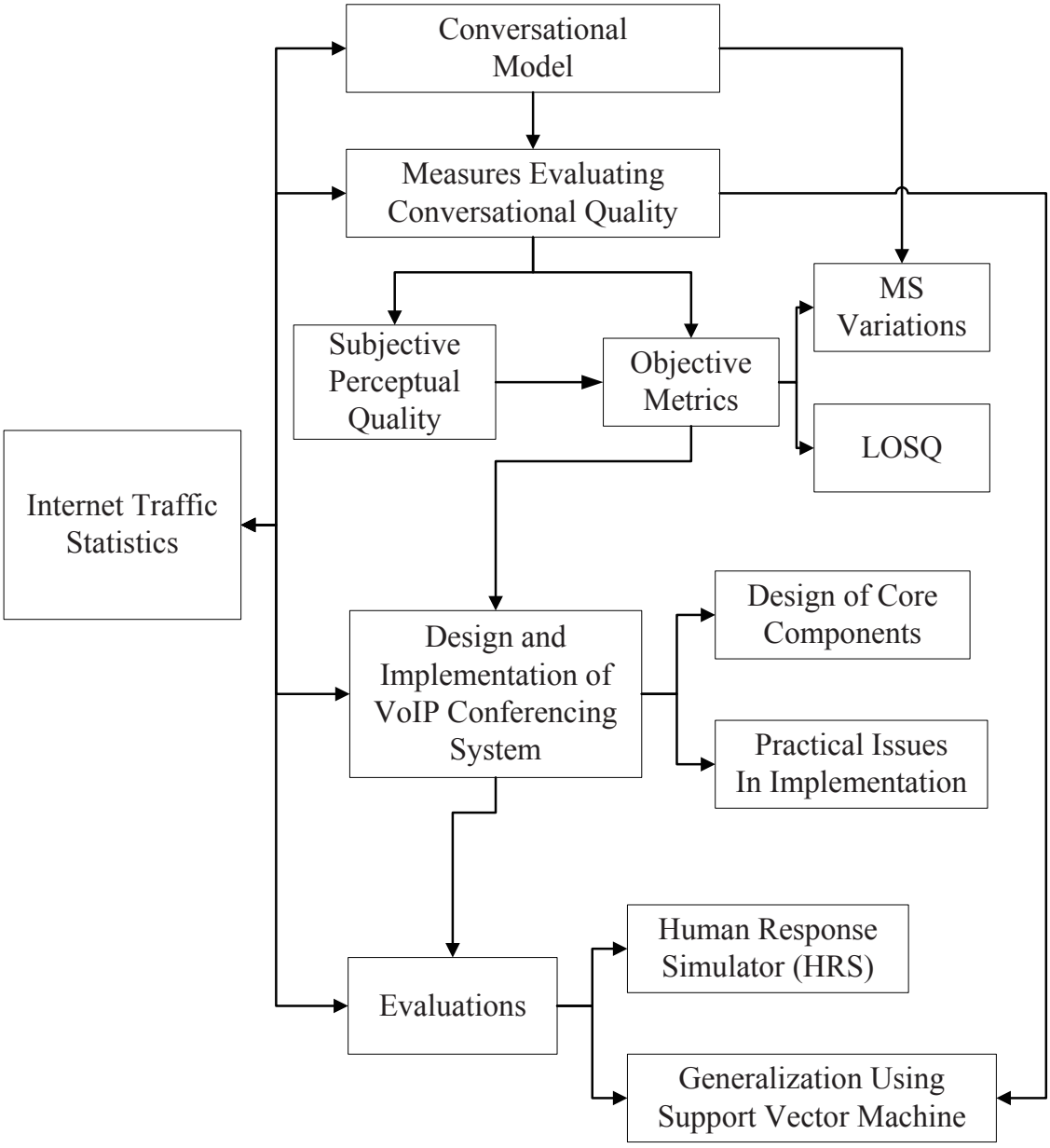


Figure 4.1: A roadmap of Chapter 4.

Two−party VoIP Conversation

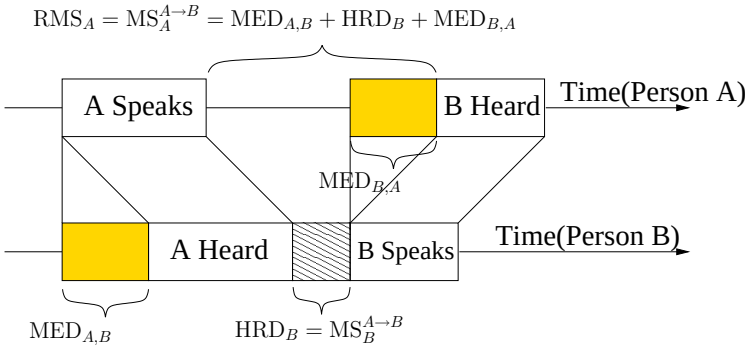


Figure 4.2: A two-party conversational model.

Multi−party VoIP Conversation

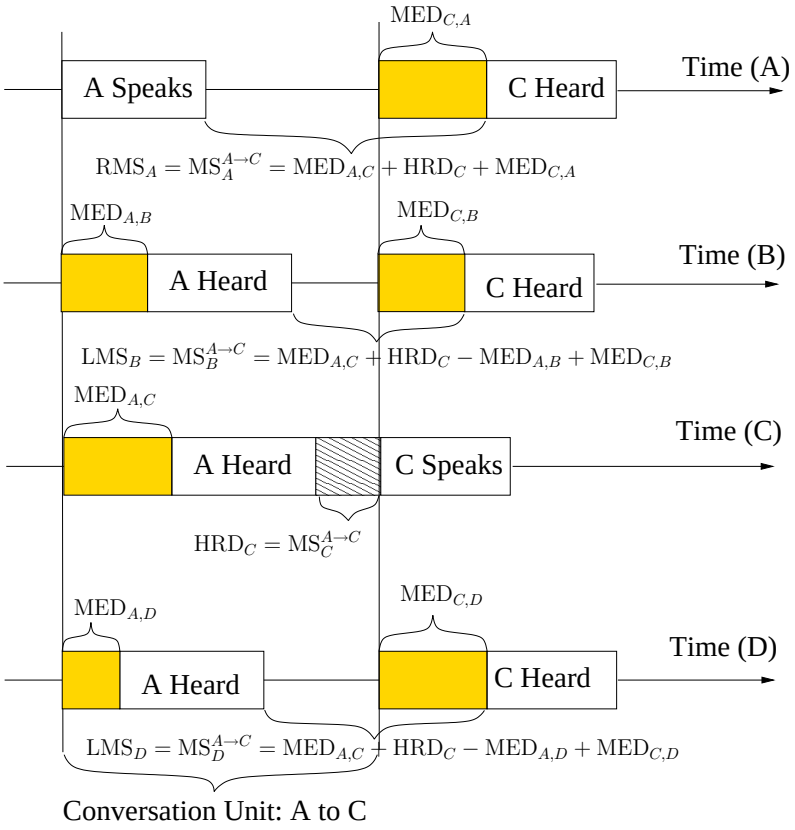


Figure 4.3: A multi-party conversational model.

- ***Listener Mutual Silence (LMS).*** A different client in a multi-party conversation (say k), is just listening to A and B speaking, indicated by $\mathrm{MS}_k^{A\rightarrow C}$ as the mutual silence experienced by listener k on the switch from A to C. $\mathrm{MS}_{k\notin\{A,C\}}^{A\rightarrow C} = \mathrm{MED}_{A,C} + \mathrm{HRD}_C - \mathrm{MED}_{A,k} + \mathrm{MED}_{C,k}$.

The current speaker experiences HRD (usually the shortest MS) when switching from the last speaker, and RMS (usually the longest MS, as it covers twice the MED) when switching to the next speaker. These correspond to the short and long MSs that are similarly observed in the two-party case and cannot be reduced without further compromising the perceptual quality. This pair of speakers at a particular turn is called the *bottleneck pair*, as it usually decides the maximum variations of mutual silences in the multi-party VoIP conference. In contrast, the remaining listeners perceive LMS that do not contribute to the bottleneck. Each passive listener belongs to a *non-bottleneck pair* with respect to the speaker in a given turn.

For the purpose of analysis, a conversation can be divided into segments called *conversational units* (CU), each of which is identified by the start and the end time of the segment in absolute time. For example, a CU from X to Y is denoted by the start of X's speech until the start of the next speaker Y's speech. Its duration is represented as in Eq. (4.1) (Figure 4.3) where SS_X is the speech segment uttered by X.

$$\mathrm{CU}^{X\rightarrow Y} = \mathrm{MED}_{X,Y} + \mathrm{SS}_X + \mathrm{HRD}_Y \tag{4.1}$$

4.3 Measures for Evaluating Conversational Quality

In this section, we propose objective and subjective metrics that are related to conversational quality. These metrics can be measured either on-line or off-line (or both) as indicators of the performance of a multi-party VoIP conferencing system.

4.3.1 Objective measures for evaluating MS variations

In order to capture the effects of MS variations, two types of objective metrics are proposed.

Conversational Symmetry (CS)

As each participant perceives variations of MSs with respect to others, he or she tends to perceive a degradation in the naturalness of the conversation because it does not resemble a face-to-face conversation with small and uniform delays. To capture the symmetry perceived by k, we define CS to be the ratio of the maximum MS experienced by k and the minimum MS experienced by k (excluding HRD) recently (say in the last minute):

$$\mathrm{CS}_k \;=\; \frac{\max_j \mathrm{MS}_k^{i\to j}}{\min_{j,j\neq k} \mathrm{MS}_k^{i\to j}}. \tag{4.2}$$

Intuitively, the numerator represents the maximum of the silence duration experienced by k, whereas the denominator is the minimum while discounting the minimum term of HRD. Note that CS_k for client k should be approximately equal to 1 in a face-to-face conversation.

Conversational Efficiency (CE)

CE measures the extension in time to accomplish a VoIP conversation when there are communication delays. It is defined as the ratio of the time a user speaks or actively listens to others to the total duration of the call:

$$\mathrm{CE} \;=\; \frac{\text{Speaking Time} + \text{Listening Time}}{\text{Total Time of Call}}. \tag{4.3}$$

Since a conversation over a network is charged according to its duration, the same conversation may cost more for a network with longer MEDs. This effect is especially pronounced in international and mobile calls, when both the network delay and the per-minute price are higher. Each participant perceives the same CE during the conversation.

Since the delays from a speaker to listeners may vary significantly, each listener can perceive different silence periods. A short silence followed by a long one will make the listener think that someone is not responding or that the listener is not receiving speech packets. On the other hand, a long silence followed by a shorter one will make the listener think that someone is responding too abruptly or even trying to interrupt others. Either case will degrade the listening quality of the multi-party conversation. These degradations may also depend on the the ratio of two consecutive

MS (CMSR). However, there is a huge volume of numbers for each conference, which is a disadvantage to defining such a term. In real practice, we may select the maximum, minimum, and average of CMSR in our study. For each person k, CMSR can be expressed as:

$$\mathrm{CMSR}_k(t) = \frac{\max\{\mathrm{MS}_k(t), \mathrm{MS}_k(t-1)\}}{\min\{\mathrm{MS}_k(t), \mathrm{MS}_k(t-1)\}}. \tag{4.4}$$

The degradations due to delays may also depend on the conversational condition, such as the type of the conversation being carried out and the conversational switching frequency [6]. For example, in a conversation with less frequent switches between the parties, the degradations due to longer MEDs will be perceived less severely. In contrast, in a conversation with a higher switching frequency, there is an increased need for face-to-face-like interactivity. For simplicity, we do not consider this factor in our evaluations.

Note that during a VoIP session, a user does not have an absolute perception of MEDs because the user does not know who will speak next and when that person will start talking. However, by perceiving the indirect effects of MED, such as MS and CE, the participant can deduce the existence of MED. For this reason, a participant cannot estimate exactly the duration of a CU but knows that it is closely related to CE. In short, MS, CMSR, CE, and CS are user-perceptible metrics that are intimately affected by MED.

4.3.2 Objective measures for evaluating LOSQ

LOSQ is determined by the percentage of speech packets that will arrive before the scheduled play-out time as well as the speech codec used in the system. The speech codec is no longer a big issue nowadays, because most of the wide-band codecs such as G.722.2 and iSAC can provide a comparable quality to the uncompressed sound. Hence, the LOSQ depends largely on how completely a VoIP client receives the speech packets. Higher jitters and losses over the Internet often lead to a degraded speech quality. Longer MEDs will improve LOSQ, because more packets will arrive before the scheduled play-out time.

At each VoIP client, speech segments can be extracted from the received audio streams. By comparing these segments with the original one, LOSQ can be evaluated using PESQ in an off-line analysis. A higher PESQ score means a better LOSQ.

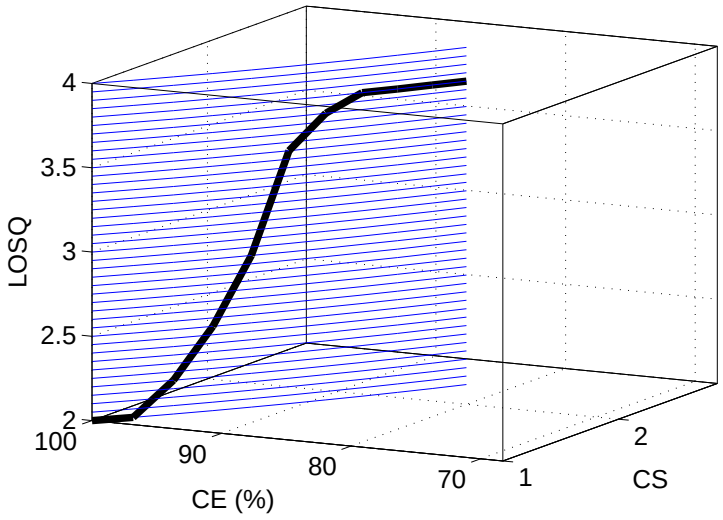


Figure 4.4: A 3-D representation of an operating curve under a conversational condition.

For online analysis, we are unable to evaluate PESQ scores at run time because of a lack of original speech segments. Sat and Wah have proposed a learning algorithm to solve this problem [63]. The idea is based on the fact that CS, CE and LOSQ can be represented by an operating curve in a three-dimensional space for a given set of network and conversational conditions (Figure 4.4). Because CS and CE can be obtained online, LOSQ can be inferred if the curve is known. In Sat and Wah's paper, they conduct off-line analysis, and CS, CE, and PESQ under different network and conversational conditions, and use a classifier to learn these curves in the three-dimensional space. When a VoIP system is run on-line, it determines the operating conditions and uses the learned classifier to locate the curve. LOSQ can be inferred from these curves using the CS and CE data collected at run time.

4.3.3 Subjective perceptual quality

The evaluations of a multi-party VoIP conferencing system depend largely on humans' subjective ratings. As was illustrated in Chapter 2, ITU P.800 Annex E [17] defines CMOS to compare two conversations based on subjective opinions using a score from -3 to 3. In our study, we are only interested in which output is better, but not how better is an output. Moreover, there are incomparable situations due to trade-offs between MS and LOSQ (also discussed in Chapter 2). Therefore, we propose a new subjective preference metric to evaluate a multi-party VoIP

conversation. Specifically, this measure defines only four scores (-1, 0, 1, 2) corresponding to (*worse*, *about the same*, *better*, *incomparable*). People are invited to do the subjective comparisons in our study and give their ratings in one of these four outcomes. In our current study and experiments, incomparable situations are not considered in subjective evaluations. We will consider this category in our future research.

4.4 Design of a Multi-Party VoIP System

In this section, we propose optimized approaches to determine the conferencing topology as well as the play-out scheduling algorithms to smooth jitters and minimize MS variations.

4.4.1 Overlay conferencing topology design

A good VoIP conferencing topology should be able to reduce the latency variations and avoid links with high jitters and losses in order to provide better LOSQ. It also needs to take into account the network burden at each client. Neither a full-mesh network nor Skype's centralized topology can achieve this. In our study, we use an overlay topology because it can better provide flexibility when the number of clients in the conference is large and when dedicated servers are not available. Its design depends on trade-offs between P, the maximum number of packets transmitted or relayed by any node in one period, and ME2ED, the maximum end-to-end delay observed by any speaker-listener pair. The quality of a multi-party VoIP system is affected by P because sending packets too frequently may lead to congestion and loss. It is also affected by ME2ED which captures the worst-case one-way delay.

The computational complexity of the algorithms is a secondary issues, though more advanced hardware is available. If there are too many packets processed (either relayed or mixed) at a parent node in an overlay, it will exert a heavy network and CPU burden at this node, and the quality of speech segments can be degraded.

In this subsection, we propose a greedy method that can iteratively decide the optimized topology faster without enumerating all possibilities (see Code 4). We define *I-parent topology* is a topology with I parent nodes. We also define $\mathrm{ME2ED}_I$ to be the ME2ED in I-parent topology. Assuming the simple case in which clients do not join or leave during a call, our approach can

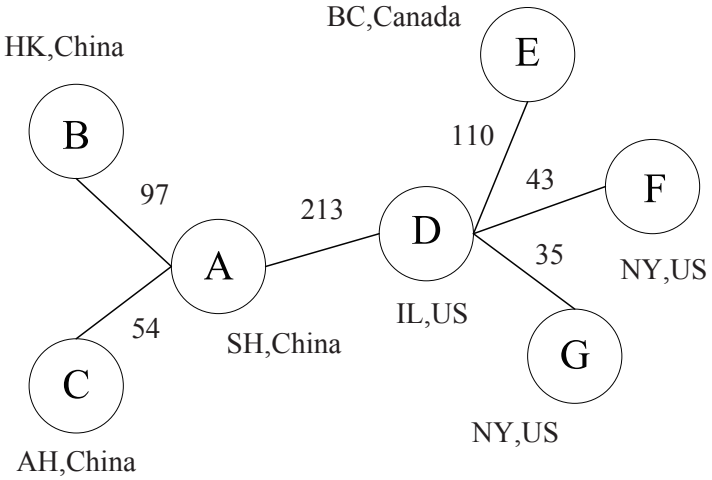


Figure 4.5: An overlay topology determined by our algorithm with two parent nodes and five child nodes.

balance a good trade-off between ME2ED and P. Figure 4.5 shows one of the topologies generated using our approach. Because the choice of topology depends on network conditions, we collect Internet data (delay and loss) during the initialization session. The topology can be redetermined if there is a significant change of the network condition. Note that, in order to reduce the processing time and computational cost, the parents simply forward the received packets through multiplexing instead of mixing.

Code 4 Determining the overlay topology

```
1: I ← 1
2: Determine ME2ED in zero-parent topology ME2ED_0
3: Determine minimum ME2ED in one-parent topology ME2ED_1
4: Save the one-parent topology TP_1
5:
6: while (ME2ED_I − ME2ED_{I−1}) > THRES do
7:    I ← I + 1
8:    Determine minimum ME2ED in I-parent topology ME2ED_I if one parent is added to TP_{I−1}
9:    Save the I-parent topology TP_I
10: end while
11:
12: Set best topology TP_I
```

4.4.2 Play-out scheduling

Common approaches in a two-party VoIP system based on adaptive jitter buffering or time-scale modification focus only on smoothing jitters, and scheduling is subject to a trade-off between MS variations and LOSQ. In a multi-party system, MS variations are more common than in a two-party version, but their trade-offs are incurred only on the speaker-listener pair. Hence, we can design a play-out scheduling algorithm that is able to reduce MED diversity while LOSQ is optimally satisfied. Two approaches are proposed.

Cooperative histogram-based adaptive play-out scheduling

In order to reduce the diversity of MEDs and better adapt to the bottleneck path in a multi-party conference, a cooperative histogram-based adaptive play-out scheduling algorithm is proposed by utilizing global network statistics.

Assuming that the end-to-end-delay statistics between the current speaker and all clients are periodically broadcast to all participants, $\mathrm{node}_{\mathrm{BN}}(t)$ (the bottleneck node, or the listening client that experiences the highest delay from the current speaker at time t) as well as the bottleneck path and its estimated MED are known to each client. The bottleneck node then uses the adaptive jitter buffering common in the two-party version and adapts its MED according to this delay statistics. The nonbottleneck nodes adapt that MEDs based on both the statistics as well as the most recent MED estimate of the bottleneck node:

$$\begin{aligned} \mathrm{MED}_{\mathrm{BN}} &= F(\beta) \\ \mathrm{MED}_{\mathrm{nonBN}} &= \gamma \cdot F(\beta) + (1-\gamma) \cdot \mathrm{MED}_{\mathrm{BN}}, \end{aligned} \tag{4.5}$$

Here, γ adjusts how symmetric the MEDs would be for different clients listening to the same speaker. For $\gamma = 0$, all listening nodes use the recent estimate of the bottleneck MED, which can improve CS at the expense of causing unnecessary waiting time for the nonbottleneck nodes. In contrast, $\gamma = 1$ reduces the scheme to a noncooperating scheme by choosing the optimal MED for each speaker-listener pair, which is equivalent to adaptive jitter buffering. In our study, we use $\gamma = 0.3$ for simplicity. However, the fixed value of γ may not best adapt to different network conditions. Another disadvantage of this algorithm is that sometimes there are high jitters on the

non-bottleneck paths, and $\mathrm{MED}_{\mathrm{nonBN}}$ will be even larger than $\mathrm{MED}_{\mathrm{BN}}$. The cooperative histogram-based adaptive play-out scheduling algorithm cannot address this case.

Distributed Equalization

To solve the issues of cooperative histogram-based adaptive play-out scheduling, we propose a distributed equalization algorithm.

For a bottleneck pair during a conversation, the RMS of the corresponding listener client can be reduced either by decreasing the HRD of the speaker or by reducing the jitter delay of the listener. The HRD of the speaker cannot be reduced because it is not under the control of the VoIP system. In most cases, the listener in the bottleneck pair is the speaker in the last turn. It is undesirable to reduce the jitter delay of this listener because it may incur losses and results in worse quality. (We set the minimum jitter delay of this listener to 60 msec according to Table 3.3.) Hence, a feasible way to reduce fluctuations in MS in each turn is to equalize the LMSs of those clients who are not speakers in the past and the current turns. This can be done by delaying voice packets played at these clients. A side effect of a longer LMS is a larger jitter delay, which accommodates more jitters and leads to better quality of the received sequence at these clients.

However, it is not possible to increase LMSs indefinitely in order to minimize the variations of MSs. The reason is that the passive listeners will have a lower perceptual quality when they have to wait for a long time before hearing the utterances from the next speaker. On the other hand, when variations are large and CS is much larger than 1, the listeners experience a conversation with unbalanced silence periods, again leading to lower perceptual quality. To this end, there is a suitable LMS that results in the best perceptual quality. Our results and user feedbacks show that the maximum MS should be less than 1300-1500 ms.

Our equalization algorithm dynamically adjusts the MS of each listener based on the history of MSs. To tolerate fluctuations in MSs, we define estimated MSs (EMS) range, $[\mathrm{EMS}_{\mathrm{min}}, \mathrm{EMS}_{\mathrm{max}}]$, as a reference that covers most of the MSs in the actual conversation. There are three cases considered in our algorithm.

1. If the MS of a listener client in the last turn is the same as RMS and is very large, then its current LMS is usually small as compared to RMS, and we set it to $\mathrm{EMS}_{\mathrm{max}}$. This allows

this client to sense less abrupt changes in his/her MS from the last turn to the current.

2. If the MS in the last turn is less than $\mathrm{EMS}_{\min}$ and the current LMS without adjustment is also less than $\mathrm{EMS}_{\min}$, then we set the LMS to $\mathrm{EMS}_{\min}$.

3. If the previous MS is within the predefined tolerable MS range, we use the moving average of the previous several MSs that are also within the range.

Our results have shown that changes of the moving window size has limited influence on perceptual quality. In this paper, we heuristically set the window size to 3. Note that our method does not depend on the specific HRD in each turn.

The equalization algorithm described above can be applied in a non-cooperative or a cooperative fashion. In a non-cooperative strategy, each client applies the algorithm without considering the MSs used by the other clients. This may result in one client setting its MS to be unnecessarily large. To address this issue, a cooperative strategy requires each client to broadcast its history of MSs to other clients at the end of a turn. Based on the the listener's estimated MS and assuming that this client is the next speaker, the strategy predicts the MSs of all listeners in the next turn. In this step, we set $\mathrm{MED}_{i,j}$ to be the average end-to-end delay from i to j plus 60-ms jitter delay at the receiver. If the equalized MS in the current turn causes any MS in the next turn to be larger than $\mathrm{EMS}_{\max}$, we reduce the current MS to a reasonable level according to the current delay statistics. The pseudocode for the cooperative equalization algorithm is shown in Code 5.

Figures 4.6 and 4.7 show a comparison of MS variations using the fixed jitter buffer and our distributed equalization. We can see that the MS variations are reduced because of LMS adjustment in the equalization algorithm, while HRD and RMS are unchanged. The cooperative strategy takes effect at turn 7 for both AH (China) and HK (China) where LMS cannot be increased to the EMS range in order to prevent over-adjustment.

4.4.3 Loss concealment

From our comparison and analysis in Chapter 2, we adopt the piggy-backing algorithm in our design for its simplicity and effectiveness.

The IETF defines that the minimum MTU (maximum transmission unit) that all hosts are

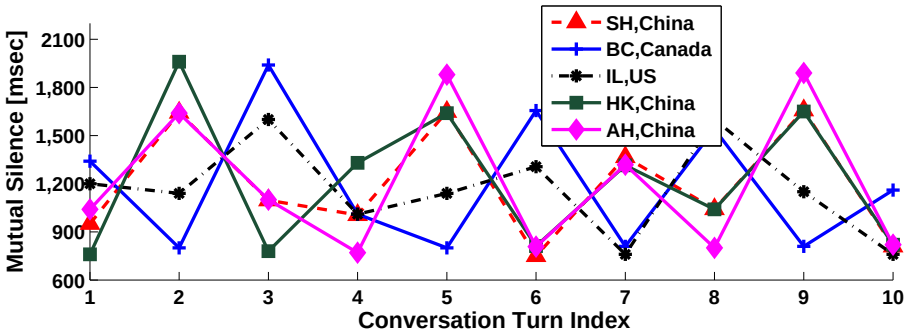


Figure 4.6: Non-cooperative POS with fixed jitter delays.

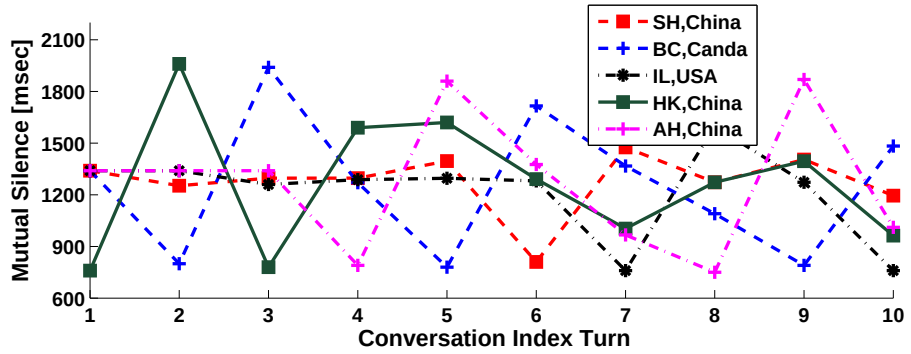


Figure 4.7: Cooperative POS with equalized MEDs for non-bottleneck pairs.

required to support is 576 bytes as defined in RFC879 [64], and the maximum length of the data field of a packet sent over an Ethernet is 1500 bytes as defined in RFC894 [65]. Usually in a broadband network, MTU is set to be 1500 bytes. Therefore, the maximum piggy-backing redundancy degree should be restricted so that the multiplexed packet should not exceed MTU. The degree depends on the codec bit rate as well as the number of voice streams for multiplexing. We have decided from Chapter 2 that G.722.2 is to be used in our implementation. Given that the maximum frame size of G.722.2 is 61 bytes (20 msec), and the number of streams for multiplexing is 6, the MTU can still support a redundancy degree up to 4.

4.4.4 Trade-offs

There are trade-offs among conferencing topology design, play-out scheduling, and loss concealment. The primary goal of topology design is to reduce latency variations and limit the network burdens on each VoIP client. However, by increasing the buffering delay at the receiver,

Code 5 Distributed equalization

```
1: {Initialization}
2: Collect initial network statistics
3: Determine estimated MS range
4:
5: {Dynamic equalization of MS for each node}
6: for all listener i do
7:    Obtain MS of last turn
8:    Calculate estimated optimal MS for the 3 cases
9: end for
10:
11: {Incorporation of cooperative strategy}
12: for all listener i in the current turn do
13:    Assume listener i is the next speaker
14:    for all listener j in the next turn do
15:       Predict MS_NEXT in the next turn in response to i
16:       Reduce MS if MS_NEXT is too high
17:    end for
18: end for
```

MS variations can be adversely affected, although larger jitters can be smoothed using the play-out scheduling algorithm. The piggy-backing loss concealment can also affect the MSs. The larger the piggy-backing degree, the larger the buffering time at the receiver and the larger the network burdens at each clients are. Our proposed algorithms on topology design, distributed equalization, and loss concealment can balance these trade-offs that can lead to best multi-party VoIP conversational quality.

4.5 Practical Issues of Multi-Party VoIP System Implementation

In this section, we present a detailed description of the practical implementation of our multi-party VoIP conferencing system. Our system was developed under Microsoft Windows XP operating system[1] and Microsoft Visual C++ 2005 platform[2].

[1] Windows XP is a product of Microsoft Corporation.
[2] Visual C++ 2005 is a product of Microsoft Corporation.

4.5.1 Overall design

Figure 4.8 shows the overall design of our VoIP conferencing system. The following procedures need to be conducted for each client.

1. ***Initial Setup.*** Each VoIP client gets the IP address and port information of all the participants in the conference, allocates memory for relay or jitter buffers, sets up communication sockets, and creates threads for sending and receiving different types of packets. Hardware support for audio wave-in and out also needs to be examined and initialized at this stage.

2. ***Determination of Topology.*** VoIP clients send time-stamped UDP probe packets to each other in the conference simultaneously using a full-mesh topology. When probe packets are received, a receiver attaches its own information and sends back these probe packets immediately. One-way delay is derived from half of the round-trip time (RTT). Based on the average delay, jitter, and loss statistics, the whole topology of the multi-party conferencing system is determined, and each VoIP client is set up for this topology.

3. ***Sending and Receiving Speech Packets.*** A VoIP client needs to start the audio wave-in thread to collect real-time audio samples. The client encodes the raw speech packets, sends the encoded packet out, and saves it in the sender buffer for piggy-backing. The receiver thread receives the speech packet and puts it into its own jitter buffer. For a parent node, it also needs to relay recently received speech packets using multiplexing.

4. ***Playing Speech Waveforms.*** When received speech packets are stored in the jitter buffer, the VoIP client needs to schedule the play-out time of the packets by using different play-out scheduling algorithms as discussed in the previous sections. A client also needs to select audio waveforms for playing and mixes the decoded streams. The client then feeds the mixed streams into the wave-out sound card device.

We assume in this section that N is the total number of clients in the multi-party VoIP conference and P is the number of parent nodes.

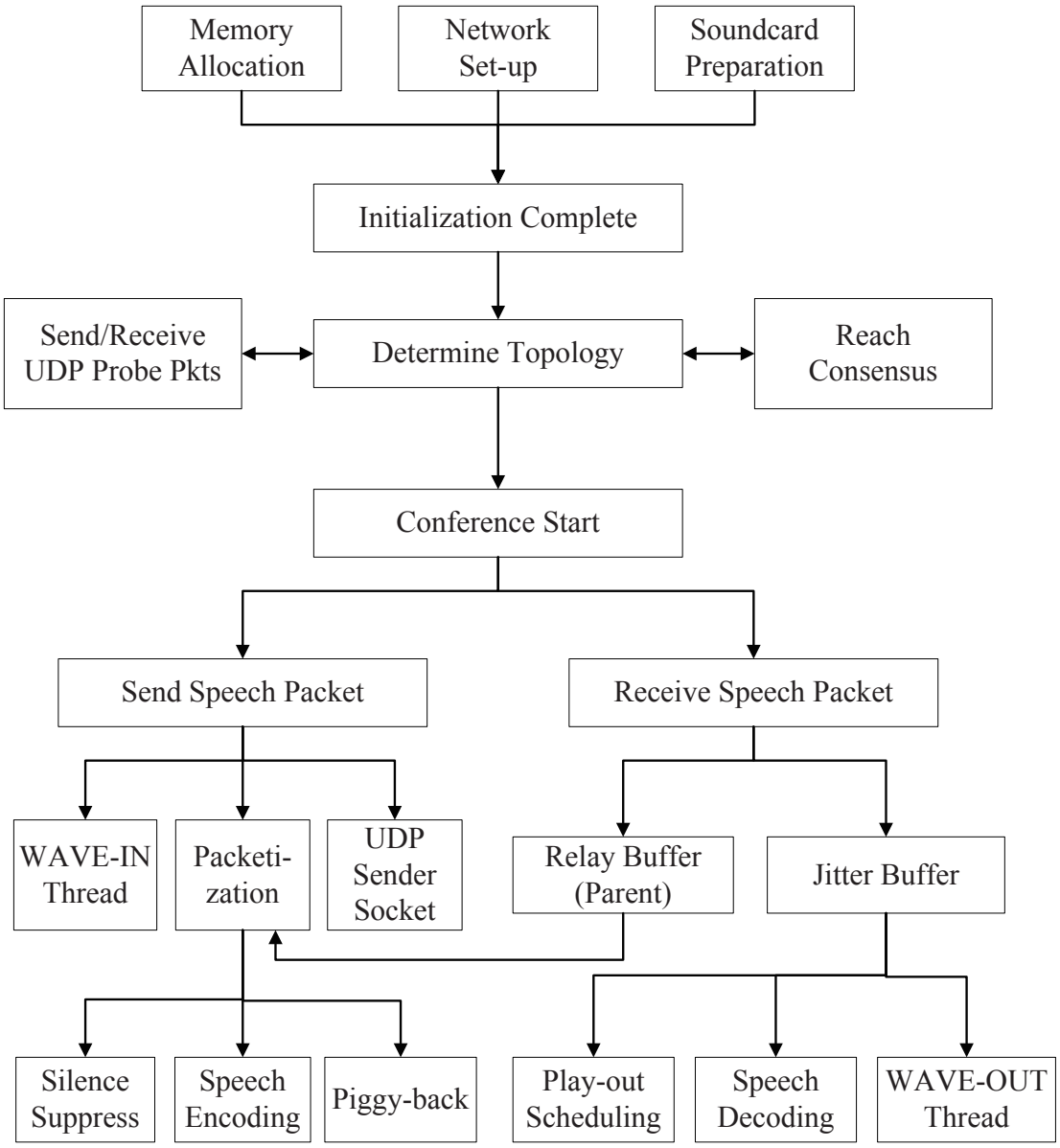


Figure 4.8: A flowchart of our multi-party VoIP system implementation.

4.5.2 Initial setup

In the initial setup, the client reads the IP address, port number and virtual ID (VID) information of all clients in the multi-party conference from a file. Note that we assign $\mathrm{VID} = 0$ to the client who first starts the conference, and this client later behaves as the virtual server to collect all network statistics data and decide the best topology.

The client also allocates memory for its own sender buffer, $(N-1)$ receiver jitter buffers, and at most $(N-1)$ relay buffers if the client is one of the relay nodes. For each client, both TCP and UDP sockets are opened in our design. TCP sockets are used to communicate topology-related control messages to guarantee reliable communications. UDP sockets are opened to send and receive network-related probe packets and speech packets.

The initialization of the sound card is done at this stage. The client should select the sound card, set the sampling rate and number of bits per sample. It also needs to allocate memory for wave-in and wave-out buffers.

4.5.3 Packetization

There are totally four different types of packets in the design of our multi-party VoIP system: UDP network probe packets, UDP speech packets, TCP network statistics packets, and TCP topology information packets. The network statistics packets and topology information packets are communicated using TCP for reliable transmission. There is a unique sequence of data at the beginning of all four types of packets for packet validation.

The structure of each UDP network probe packet is as follows:

- UNIQUE SEQUENCE : 4 bytes
- PACKET TYPE (1000) : 4 bytes
- SENDER ID : 4 bytes
- RECEIVER ID : 4 bytes
- PACKET SENDING TIME : 2 bytes

The structure of a TCP network statistics packet is as follows:

- UNIQUE SEQUENCE : 4 bytes
- PACKET TYPE (1001) : 4 bytes

- SENDER ID : 4 bytes
- LOSS RATE TO ALL NODES FROM SENDER : 4 * N bytes
- MAXIMUM DELAY TO ALL NODES FROM SENDER : 4 * N bytes
- AVERAGE DELAY TO ALL NODES FROM SENDER : 4 * N bytes
- MINIMUM DELAY TO ALL NODES FROM SENDER : 4 * N bytes

The structure of a TCP topology information packet is as follows. Note that the parent of a parent node is itself.

- UNIQUE SEQUENCE : 4 bytes
- PACKET TYPE (1002) : 4 bytes
- SENDER ID : 4 bytes
- TOTAL NUMBER OF PARENTS IN THE TOPOLOGY : 4 bytes
- ALL CLIENT ID IN THE TOPOLOGY : 4 * N bytes
- CORRESPONDING PARENT ID IN THE TOPOLOGY : 4 * N bytes

The structure of a UDP speech packet include the general header and the multiplexed encoded speech frames. A child node only needs to send its own voice streams to its parent. A parent node needs to multiplex multiple voice streams together and sends them to other nodes. Different speech packets are sent at the parent node for different destinations.

Letting M be the total number of streams that are used for multiplexing and P be the piggy-backing degree, the general header for a UDP speech packet is as follows:

- UNIQUE SEQUENCE : 4 bytes
- PACKET TYPE (1003) : 4 bytes
- SENDER ID : 4 bytes
- RECEIVER ID : 4 bytes
- REQUEST PIGGY-BACKING DEGREE FOR THIS RECEIVER ID : 4 bytes
- TOTAL NUMBER OF STREAMS THAT ARE USED TO MULTIPLEX (M) : 1 bytes
- SPECIAL INDICATOR FOR PIGGY-BACKING : 1 bytes
- CURRENT PIGGY-BACKING DEGREE (P) : 1 bytes

A UDP speech packet includes $M \times P$ encoded speech frames, and each is indicated by the client ID and the frame sequence number. It also indicates whether a frame is a silence frame or a speech frame by implementing *silence suppression*, discussed in the following subsection.

- STREAMING ID : 1 bytes

- SEQUENCE NUMBER : 3 bytes
- INDICATOR OF SILENCE SUPPRESSION : 2 bytes
- VOICE DATA (IF NOT SILENCE) : Size depends on the encoded rate

The system examines all the data fields and packet sizes in order to validate all four types of packets.

4.5.4 Determination of topology

A VoIP client starts with sending one or two seconds of UDP probing packets (with sending rate equal to the UDP speech packets) to all the other clients in the conferencing system. The network statistics is obtained by using half of the round-trip time. The TCP network statistics packets, which include the average loss rate, maximum, average, and minimum delays from one client, are sent to the client starting the conference, who is elected as the *leader* in the distributed system.

After the leader receives all TCP network statistics packets, it decides the best topology using our greedy approach as described in the previous section. It then sends TCP topology packets back to the other clients. Through this mechanism, all VoIP clients in the conference will reach a *consensus* on the conferencing topology.

Note that we use TCP to transmit network statistics and topology packets to guarantee successful and reliable transmissions just for simplicity. Under some cases, where a firewall exists and TCP ports may be blocked, we can use UDP packets alternatively by implementing ACK packets and a retransmission mechanism.

Another point worth considering is the case in which one client (say A) cannot reach another client (say B) by sending UDP probing packets. It is somewhat not uncommon according to the traces we have collected from PlanetLab. Sometimes A cannot find a route to B, so packets from A cannot reach B, but at the same time, packets from B can reach A. Under that circumstance, we set the delay between A and B (both ways) as infinity and try to avoid these links.

4.5.5 Sending speech packets

After all the clients in the conference reach a consensus of the topology, the WAVE-IN thread starts, and each client begins to collect audio sampled frames from the sound card periodically (20

msec in our design). The encoded frame is saved into the sender buffer, and the length of the sender buffer is $\mathrm{MAX\_PIGGY\_DEG} \times \mathrm{ENCODED\_FRAME}$. Here $\mathrm{MAX\_PIGGY\_DEG}$ is the maximum piggy-backing degree used, and $\mathrm{ENCODED\_FRAME}$ is the size (in bytes) of the encoded frame.

A child node sends its piggy-backed UDP speech packet to the its parent node. A parent node attaches its own piggy-backed encoded frames as well as relayed frames in its UDP speech packet and sends the packet to its children and other parent nodes. There are relay buffers implemented in each parent node so as to hold these relayed frames temporarily.

Note that we implement the link-based piggy-backing algorithm in our design, meaning that the piggy-backing degree depends only on the loss rate of each specific link, not on the client-to-client loss condition. A parent node use the same piggy-backing degree for its own speech frames as well as its relayed frames.

4.5.6 Receiving and playing-out speech packets

Each client uses a single UDP receiver socket for all the incoming streams. A client extracts the speech packets and puts the encoded frames into the corresponding receiver jitter buffer. A parent node, in addition, also needs to put the encoded frames into the relay buffers.

All clients use a pre-encoded silence frame to replace the silence-suppressed packets at the receiver buffer.

At the scheduled play-out time, the client decodes the received frames in the jitter buffer and mixes the voice streams before they are sent to the WAVE-OUT buffer.

4.5.7 Failure detection and recovery

If a VoIP client (say, A) cannot receive any packets from another client (say, B) for more than 10 seconds, our system assumes a failure. There are two possible reasons:

1. Client B fails.

2. The link between clients A and B fails.

In the first case, we can simply disregard the failed client. In the second case, our current system does not have a self-recovery mechanism. A further step is to implement failure recovery in the VoIP conferencing system design. We assume that the client starting the conference (say, S) is still working and all TCP connections are successful (otherwise, the only way is to restart the conference). The failure can be recovered by performing the following steps:

1. A sends a TCP emergency message to S indicating that it cannot reach B.
2. S receives the emergency message and broadcasts to all the clients in the conference that the topology needs to be redetermined.
3. UDP probe packets are sent again using the full-mesh topology while UDP speech packets are sent simultaneously.
4. Each client informs S of the new network statistics, and S calculates the new topology.
5. S informs all clients the new topology, and each newly determined parent node prepares its relay buffers for the new topology. Note that in order to avoid gaps in the received streams, the old topology and the new one require an overlapping working time.
6. Once the new topology is set and functioning well, each client releases memory for the old topology.

4.5.8 Other components implemented

The following is a list of all critical components that are implemented in our VoIP conferencing design. Implementations of these components are from existing open source libraries offered by the Speex codec and the *Simple Directmedia Layer* (SDL) library [66].

Silence suppression

From our experiences in a multi-party conferencing, usually only a small portion of people will speak at the same time, and most of time there is only one speaker. This suggests that we can reduce the network bandwidth burden to a large extent by implementing silence suppression. Note

that the goal is not to suppress the silences within a speech segment, but to suppress those between two consecutive speech segments.

In order to differentiate the voice and the silence parts, we have used the voice activity detection (VAD) mechanism in Speex, which can detect whether the audio frame being encoded is speech or silence/background noise by monitoring its envelope.

Speech enhancement

Background noises from the microphone and sound card can degrade the LOSQ of the listeners. These noises can be suppressed by reducing them from the speech spectrum using an adaptive filtering approach [67, 68]. In our implementation, we have utilized the preprocessing library provided by Speex, which can effectively suppress the background noises and enhance listening perceptual quality.

Echo cancellation

Echo in voice transmission usually occurs for two reasons: (1) analog signals transmitted through a place where two-wire cord changes to four-wire cord and the characteristic impedances of the two cords mismatch; (2) audio outputs leaked into the audio inputs [69, 70].

In our current VoIP implementation, because the speech waveforms are digitally encoded, and because digital packets are transmitted over a wired or wireless Internet, no echo due to the first reason can occur. When we use a microphone to listen and speak in the conferencing, unless two participants are sitting nearby, it is highly unlikely that the output sound can be leaked into the input of its own microphone or other microphones. If someone is using a speaker for the audio output, however, these sounds can be leaked into the microphone input, and VoIP users can hear the echoes in this case.

We have used the echo cancellation component in Speex to reduce echoes. Because it requires an initial estimation of the time shift between the echoes and the original signals, however, sometimes this component may not be very effective.

Automatic gain control

A current VoIP system such as Skype can automatically adjust the volume of its audio input in order to reach best listening quality. In our implementation, we have used the SDL library to achieve automatic input gain control. The open source library can differentiate the speech and silence/background noise component in an audio frame, and adjust only the speech component to a certain gain.

Mixing of multiple speakers

We have also used a mixing component provided by SDL library to mix speeches from multiple speakers. The mixed signals are sent to the sound card for audio output.

4.6 Generalization of Results Using SVM

Subjective quality can best describe human perception on the performance of a VoIP system; however, it is impossible to do subjective rating on-line, and subjective tests are rather costly off-line. Instead, objective metrics can be easily obtained both on-line and off-line, which motivates us to relate these metrics to subjective opinions and generalize the results. Based on the generalization and objective metrics, we can select an algorithm at run time to achieve the best subjective conversational quality.

4.6.1 Mapping from objective metrics to subjective opinions

As mentioned in the previous sections, there are too many objective metrics that influence the perceptual quality of VoIP listeners. We know that subjective quality is not a linear combination of multiple objective metrics. For example, a lower PESQ with smaller MS variations is preferred over a higher PESQ with larger MS variations. Sometimes even the absolute value of PESQ difference of two outputs may really matter. The subjective comparison of a pair of VoIP outputs depends on a complex nonlinear curve that may include all objective metrics and their difference in absolute-value form. It is difficult to determine such a curve by a simple model.

In order to address this issue, we present a *support vector machine* (SVM) classifier to do the

mapping in this chapter. The goal of the SVM in our study is to predict the comparative subjective perceptual quality based on the extracted objective measures of two systems. SVM supports multi-class classifications, which help predict subjective ratings with multiple output possibilities.

4.6.2 Overview of SVM

SVM is a powerful tool developed by Vapnik and his group at AT&T Bell Lab for data classification [71]. Its principle is to use a hyperplane to separate two classes. The determination of the hyperplane is based on the maximization of the margin and the minimization of the errors between the *training data* belonging to two classes. The resulting hyperplane can be used to predict the output of unseen *testing data*.

There are numerous SVM implementations nowadays, such as LIBSVM [72] and Light-SVM [73]. In our study, we use LIBSVM for its fast speed and user-friendly interface, as well as its ability to support multi-class classifications. LIBSVM was developed by Chih-Chung Chang and Chih-Jen Lin in the National Taiwan University.

LIBSVM is based on the C-support vector classification (C-SVC) developed by Boser et al. [74] in 1992 and Cortes and Vapnik in 1995 [71]. Let the *input feature vectors* be $\mathbf{x}_i \in R^n$, $i = 1, \ldots, l$ and the *output vector* be $\mathbf{y} \in R^l$ such that $y_i \in \{1, -1\}$. A data set that is linearly separable by using a hyperplane pair $(\mathbf{w}, b)$ can be expressed as:

$$y_i = 1 : \mathbf{w} \cdot \mathbf{x}_i + b \quad \geqslant \quad 1, \ \ \forall\, \mathbf{x}_i \in \text{ Class 1} \tag{4.6}$$

$$y_i = -1 : \mathbf{w} \cdot \mathbf{x}_i + b \quad \leqslant \quad -1, \ \ \forall\, \mathbf{x}_i \in \text{ Class 2} \tag{4.7}$$

The decision function can be expressed in Eq. (4.8). By deciding the input sample is at which side of the hyperplane, the sample can be classified into the corresponding category.

$$f_{\mathbf{w},b} = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b) \tag{4.8}$$

To be suitable for diverse tasks that may include nonlinear relations, a mapping of input variable $\mathbf{x}$ to a higher dimensional feature space $\mathbf{x} \rightarrow \phi(\mathbf{x})$ is commonly implemented by using the

Gaussian RBF kernel (Eq. (4.9)). Using this method, a linear solver can be applied to a more accurate extent that only nonlinear solvers can solve otherwise.

$$K(\mathbf{x}, \mathbf{y}) = \exp(-\|\mathbf{x} - \mathbf{y}\|^2) \tag{4.9}$$

By constructing an error function (Eq. 4.10), SVM can be solved as a quadratic programming (QP) problem. A simple linear solver can be implemented to achieve faster and better results as compared to a non-linear solver. A decomposition algorithm is often implemented to iteratively solve sub-problems and guarantee global optimality in order to reduce the memory requirement.

$$\text{Minimize}_{\ \mathbf{w},b}\ \Phi(\mathbf{w}) \quad = \quad \frac{1}{2}\|\mathbf{w}\|^2 \tag{4.10}$$

$$\text{subject to} \quad y_i(\mathbf{w} \centerdot \mathbf{x}_i + b) \quad \geqslant \quad 1, \ \ i = 1...l. \tag{4.11}$$

Note that C-SVC itself supports only two-class classifications. When there are multiple classes in the task, LIBSVM determines hyperplanes for each pair of classes first and then uses a voting approach to decide the exact class [72].

The general approach of using SVM classifier is illustrated as follows. We choose a set of data called *training data* that can evenly represent different classes. The SVM learn the training data and generate a model that can achieve a classification hyperplane with the highest accuracy. After validating the learned SVM model using training data, we use the classifier to predict the results of unseen *testing data*.

4.6.3 Statistical significance of preferences

To determine the dominant opinion between two algorithms under a given condition (with $> 50\%$ probability and a certain level of statistical significance in our study), we model the subjective opinions by a multi-nomial distribution with 4 possible outcomes $\{-1, 0, 1, 2\}$ as discussed in Section 4.4, assuming the independence of samples. Letting $x_{-1},\ x_0,\ x_1,\ x_2$ be the number of votes for each outcome, and $p_{-1},\ p_0,\ p_1,\ p_2$ be the probability of voting for these outcomes, the multi-nomial model can be expressed as the following probability mass function:

$$\frac{(x_{-1}+x_0+x_1+x_2)!}{x_{-1}!\,x_0!\,x_1!\,x_2!}\,p_{-1}^{x_{-1}}\,p_0^{x_0}\,p_1^{x_1}\,p_2^{x_2} \tag{4.12}$$

In order to make statistical decisions using experimental data, we conduct hypothesis testing by selectively combining three options and produce an equivalent binomial distribution that represents the *for* and *against* probabilities of the opinion.

Option i is *dominant* if the following hypothesis is accepted:

$$H: \left(p_i, \sum_{j\neq i} p_j\right) \text{ is drawn from } \mathrm{binomial}(N, p \geq 0.5) \tag{4.13}$$

where N is the number of samples.

To conduct hypothesis testing to determine which opinion i is *dominant*, the following terms are defined:

1. Null Hypothesis H_0: $\left(p_i, \sum_{j\neq i} p_j\right)$ is drawn randomly from $\mathrm{binomial}(N, p \geq 0.5)$.
2. Alternate Hypothesis H_1: $\left(p_i, \sum_{j\neq i} p_j\right)$ is drawn with statistical significance from $\mathrm{binomial}(N, p \geq 0.5)$.
3. Significance Level α: the significance value that the tests can rule out the null hypothesis. We set $\alpha = 20\% \sim 40\%$ in our experiments depending on the significance requirement.
4. P-value: the probability of voting for the dominant opinion from $\mathrm{binomial}(N, p \geq 0.5)$, which can be obtained by the cumulative density function (CDF).

Given α and $p \geq 0.5$, and K out of N samples need to agree on an opinion if this opinion is dominant, the relation between α, N and K satisfies:

$$\mathrm{argmin}_K \sum_{i=0}^{K} \begin{pmatrix} N \\ i \end{pmatrix} \cdot 0.5^i \cdot 0.5^{N-i} \geq 1-\alpha \tag{4.14}$$

For instance, for 90%, 80%, and 70% significance (corresponding to $\alpha = 10\%$, 20%, and 30%), we know from Eq. (4.14) that 27, 25, and 24 out of 45 samples need to agree on an opinion.

In case there is no dominant opinion, say, if the resulting percentage voting for the four scores is (33%, 33%, 33%, 0), we call this situation *inconclusive* (IC).

4.6.4 SVM methodology and example

In our study, we have multiple pairs of speech output waveforms. Each waveform contains several speech turns (segments) separated by silence periods. For two waveforms in a pair, objective metrics such as MS, CS, CE, and PESQ may be different. We want to compare in each pair which one leads to a better subjective perceptual quality. However, it is difficult to formulate a mapping curve to get the subjective results from these objective metrics, and it is rather costly to do the subjective tests for each pair. Hence, we have recourse to SVM to generalize our results. The following example illustrates the methodology and our approach.

1. Extract all objective metrics for each waveform output. In Table 4.1, we are interested in 11 metrics, including CE, CS, MS (average, minimum, maximum, and variance), CMSR (average, minimum, and maximum), and PESQ (average and minimum).

2. Conduct subjective perceptual quality tests in Section 4.3.3 to compare every pairs of waveforms. Suppose that there are N pairs of output waveforms for comparisons. We invited nine people in the test, and the comparison results can be expressed in (number voting for worse, number same, number better), representing the number of votes in each opinion correspondingly. One example of the comparison outcome is (1, 7, 1), meaning that one person thinks that the listening quality of the first waveform is better than the second, and seven people think that the listening quality of both waveforms are approximately the same, while another one person thinks that the first waveform is better than the second. The sum of votes for each opinion should be equal to the total number of people in the tests, which is 9 in this test. Note that currently we do not consider incomparable situations, just for simplicity.

3. Use J pairs as the training data to generate the SVM prediction model and K pairs as the testing data to verify the model, satisfying $J + K = N$. All 11 of the metrics extracted in Table 4.1 are employed as input features. The training data in LIBSVM is represented by J

Table 4.1: Sample SVM input features using Conversational Order 1 and Trace Set 3.

	Features	Person 1	Person 2	Person 3	Person 4	Person 5
A1	CE	0.644	0.646	0.684	0.605	0.653
	CS	3.142	2.875	2.142	2.232	2.277
	avg MS	1231	1221	1072	1216	1199
	min MS (excl. HRD)	880	880	880	885	885
	max MS	2765	2530	1885	1975	2015
	var MS	617.45	568.71	331.92	407.94	386.43
	avg CMSR	1.89	1.76	1.61	1.64	1.52
	min CMSR	1.04	1.04	1.01	1.16	1.14
	max CMSR	4.42	3.33	2.48	2.23	2.65
	avg PESQ	3.184	3.145	3.823	3.132	3.236
	min PESQ	0.077	0.101	2.336	0.101	0.101
A4	CE	0.648	0.65	0.652	0.653	0.654
	CS	2.13	1.904	1.88	1.811	1.204
	avg MS	1278	1266	1226	1266	1252
	min MS (excl. HRD)	960	1040	1045	1085	1200
	max MS	2045	1980	1965	1965	1445
	var MS	367.29	354.15	403.07	335.23	227.69
	avg CMSR	1.53	1.53	1.74	1.56	1.27
	min CMSR	1.03	1.03	1.03	1.06	1
	max CMSR	2.71	2.61	2.59	2.59	1.89
	avg PESQ	3.516	3.554	3.911	3.864	3.804
	min PESQ	0.245	1.277	3.374	3.327	3.327
	Subjective Raw Results	(1,4,4)	(0,2,7)	(1,7,1)	(0,3,6)	(1,2,6)
P1	OP	2	1	0	1	1
P2	OP 1	1	0	1	0	1
	OP 2	4	2	7	3	2
	OP 3	4	7	1	6	6

Note: A1: Non-adaptive play-out scheduling. A4: Distributed equalization. MS is represented in msec. Subjective raw results are presented by (number worse, number same, number better). P1 and P2 are the two approaches of SVM output representations in Section 4.6.4. OP is the SVM output class number.

lines (corresponding to J pairs of training data) in the format of $(\mathrm{OP}\ \ \mathrm{IP1}\ \ \mathrm{IP2}\ \ \mathrm{IP3}\ \ \mathrm{IP4}\ \ \ldots\ \ \mathrm{IP22})$, where OP is the output result and IP is the input features. There are a total of 22 input features in our test, corresponding to 11 metrics for the first waveform output and another 11 metrics for the second in a comparison pair. An example of the 22 features is listed in each column in Table 4.1.

4. The SVM output is the subjective results. When we use the SVM, there are two approaches to represent the output.

   - The first approach, indicated by P1 in Table 4.1, is to use only four classes (either class -1, 0, 1 or 2) as OP in LIBSVM. Classes -1, 0 and 1 correspond to whether an output waveform is worse than, same as, or better than the other waveform respectively. Class 2 means that the comparison will not lead to any inconclusive result, and will be explained later in an example. The goal of the SVM classifier is to predict which class is most accurate for testing data. Because the unprocessed subjective results are represented by (number worse, number same, number better), we have to apply hypothesis testing to determine whether the results are in class -1, 0, 1 or 2. Since we have nine people to do the subjective tests, for 70% significance we know from Eq. (4.14) that the output belongs to a class (or a class is dominant) if at least six people vote for this class. For example, if the comparison result is (1, 7, 1), OP is 0, meaning that the first waveform is worse than the second. If there are no more than 6 people voting for any of the three opinions (e.g. 1, 4, 4), this subjective comparison leads to an inconclusive result, and OP is 2 in this case. The advantage of this approach is that SVM only needs to be trained once before a prediction model is generated.
   - The second approach, indicated by P2 in Table 4.1, is to train the SVM data before hypothesis testing is applied. Since there are three opinions in the subjective comparisons representing $\mathrm{OP1}$, $\mathrm{OP2}$ and $\mathrm{OP3}$, the data should be trained three times for each opinion, and three SVM models are generated. For the output of each SVM model, there can be up to 10 classes (ranging from 0 to 9) to represent OP.

Because of the precision of the SVM fitting and the limited number of training data, the

second approach may lead to a problem when the sum of prediction outputs generated from the three SVM models using the the testing data is not equal to 9 (say 6, 4, 2), which poses a problem in the hypothesis testing. Hence, we adopt the first approach in our study.

5. After the SVM data is trained and the model is generated, we should verify the original training data. A good SVM model should lead to a prediction accuracy of at least 80-90%. Otherwise, the selection of the training data may not be appropriate. Two factors can affect accuracy: (1) the number of samples in each class may not be as evenly divided as possible; (2) noise causes many samples that are not easily separable.

6. Use the SVM model learned to predict the testing data and verify the prediction result with actual subjective test results.

7. If we want to compare the performance of two systems and there are C clients in the multi-party VoIP conference, we should combine all C comparisons and make an overall assessment. Suppose $C = 5$ and nine people do the subjective tests; there is a total of 45 votes. According to Eq. (4.14), if we want to achieve a significance of at least 70%, there should be at least 24 votes in favor of of one opinion. Otherwise, the two systems are called inconclusive.

4.7 Summary

In this chapter, we have studied the multi-party conversational model and have proposed new objective and subjective metrics that can better describe and evaluate the quality of a VoIP conferencing system. Based on the proposed measures and trade-offs illustrated in Chapter 1, we have designed algorithms for the core components of the system. Key issues are discussed for the implementation of a real system. We have proposed a systematic approach for the evaluation, and this approach can be generalized to any unseen network and conversational condition.

Based on experiments conducted on our system and Skype, we can extract relevant objective metrics impacting conversational quality. Subjective comparisons are also conducted so as to find a mapping from the objective results to subjective ratings. By utilizing the mapping generated from the SVM classifier, we can predict, in four play-out scheduling algorithms, the one to achieve

the best subjective perceptual quality. We present a detailed data analysis and results discussion in the next chapter.

CHAPTER 5

ANALYSIS OF EXPERIMENT RESULTS

In this chapter, we conduct multi-party VoIP conversation simulations using our VoIP system and Skype. We present complete objective and subjective results from our study and give an in-depth discussion of these results. We generalize the experimental data using LIBSVM and develop a classifier that can learn to select the best algorithm using learning examples derived from subjective tests under limited network and conversational conditions.

5.1 Experiment Setup

In this section, we conduct simulations both on our VoIP system and Skype (Version 3.5.0.214). The experiment setup is shown in Figure 5.1. The topology is determined by the proposed greedy approach (Code 4) in our design and may be different from what is shown in Figure 5.1 in terms of the number of parent nodes. In our VoIP system, we also implement four play-out scheduling algorithms:

1. Non-adaptive play-out scheduling: The fix jitter buffer size is set to be 60 msec.
2. Histogram-based adaptive play-out scheduling (non-cooperative): We dynamically adjust the jitter buffer size based on the previous 10-sec Internet delay history.
3. Histogram-based adaptive play-out scheduling (cooperative): We use Eq. (4.5) to schedule the play-out time.
4. Distributed equalization: Play-out time of each speech segment is scheduled according to the algorithm illustrated in Code 5.

We design a *Conference Human Response Simulator* (CHRS) at each computer that can communicate with any VoIP software via the Virtual Audio Cable (VAC) software, which behaves like a virtual pipe for audio transmissions. The goal of CHRS is to simulate a multi-party conversation with smooth turn-taking among participants and without double-talks.

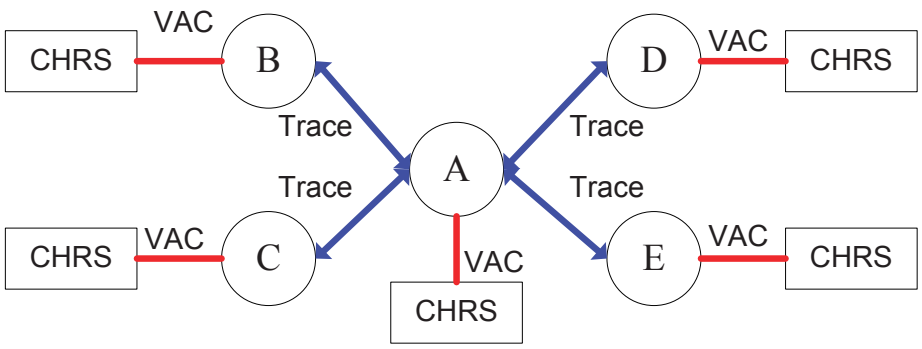


Figure 5.1: The configuration in our simulation.

By using a predefined order in which participants converse, when a particular participant's turn is up for conversation, its CHRS waits for a preset time after detecting the end of the previous speech, before sending some prerecorded speech waveforms to the VoIP software. To allow the analysis of quality, CHRS also records the spoken waveforms as well as the waveforms heard from other participants.

We also use Wireshark[1] in each node to monitor incoming and outgoing packets.

To facilitate repeatable experiments and to allow the behavior of VoIP systems under different scenarios to be examined, we use in our simulations the five categories of Internet traces collected from PlanetLab in Chapter 3 (Table 3.4) and two 5-party conversations extracted from videos. Table 5.1 summarizes the average, minimum, and maximum of the lengths of speech segments and the conversation order of two multi-party social conversations extracted from a television series. One conversation consists of fifteen turns from three females and two males, and the other has thirteen turns from two females and three males.

We process the waveforms in each of the conversations from both systems. Based on the boundaries extracted from the spoken and heard waveforms, we compute the MS perceived by each client between two segments, as well as CMSR, CS, and CE. For each segment, we also evaluate its LOSQ using PESQ. Finally, we conduct subjective tests that compare each of the conversations generated by the four play-out scheduling algorithms implemented in our system and the corresponding conversations of Skype. In our tests, each test subject was presented with

[1]TheWireshark protocol analyzer is available under the GNU General Public License version 2 `http://www.wireshark.org/`.

Table 5.1: Characteristics of speech segments in two five-party social conversations used in our experiments.

Set	Length (ms)			Conversation Order
	Avg	Min	Max	
1	2222	600	4400	A, C, A, B, C, E, D, B, C, D, B, C, D, B, C
2	1603	630	3350	B, A, C, B, D, E, C, D, B, C, B

two conversations and was asked to compare the quality of one relative to another, using the subjective perceptual quality proposed in Chapter 4. In our current comparisons, we do not consider incomparable situations.

5.2 Topology

Table 5.2 summarizes the overlay topology generated by our greedy approach using the seven trace sets presented in Table 3.4. Parent nodes are specified in shaded boxes. We can see from the table that Trace Sets 3 and 4 will lead to an overlay with two parents because of the large diversity of network latencies, while other trace sets generate only one parent. Figures 5.2 and 5.3 show the two-parent overlay topology for Trace Sets 3 and 4 in Table 3.4.

5.3 Results on Objective Metrics

Table 5.3 summarizes the extracted objective results from the four play-out scheduling algorithms implemented in our VoIP system using different network and conversational conditions. In general, a dynamic play-out scheduling algorithms can better adapt to network jitters and improve PESQ when compared to a fixed-jitter-buffer algorithm. Under the circumstance when there are large diversities of MED, the distributed equalization algorithm can greatly reduce CS by up to 25% at the expense of slightly lowering CE as compared to other algorithms, and the average CMSR also becomes apparently smaller.

Table 5.4 summarizes the extracted objective results from Skype. By comparing Skype output and the distributed equalization algorithm implemented in our VoIP system, the results show that CSs and average CMSRs from Skype are larger than those from our system, which means greater

Table 5.2: Overlay topology generated by our proposed greedy approach. Parent nodes are specified in shaded boxes.

Set 1	Set 2	Set 3	Set 4	Set 5	Set 6	Set 7
CA,US	NY,US	BJ,CN	SD,CN	CA,US	BJ,CN	Canada
IL,US	OR,US	IL,US	Japan	Canada	UK	India
Germany	TX,US	Hungary	TJ,CN	HK	TX,US	CA,US
MD,US	CA,US	SH,CN	TX,CN	NH,US	Canada	SC,CN
UK	MO,US	Taiwan	Uruguay	AH,CN	SX,CN	AH,CN

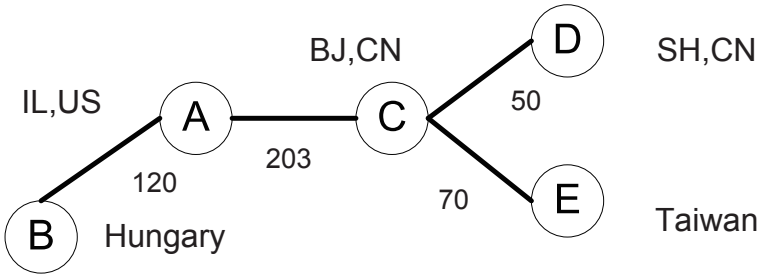


Figure 5.2: Overlay topology for Trace 3 in Table 3.4. Each number shows the delay in msec.

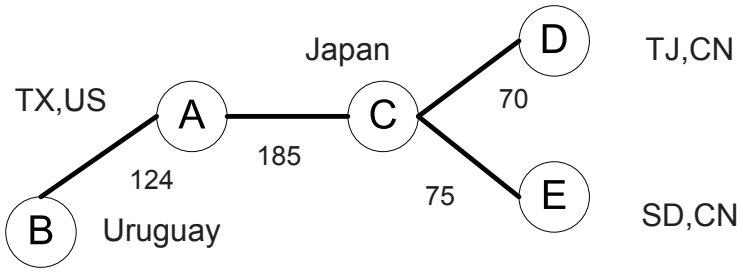


Figure 5.3: Overlay topology for Trace 4 in Table 3.4. Each number shows the delay in msec.

MS variations. The resulting PESQs from Skype are much smaller, which means poorer LOSQ.

5.4 Subjective Comparison Results

Table 5.5 summarizes the subjective comparison results between any two of four play-out scheduling algorithms implemented in our VoIP system under various network and conversational conditions. In general, we find that at low diversity of network delays, the majority of the subjects vote for approximately the same condition, regardless of whether the Internet links are lossy or not. When there is a larger diversity of network delays or jitters over the links, the distributed equalization algorithm has a preference over other two adaptive play-out scheduling algorithms, and all three adaptive algorithms are preferred over fixed jitter buffering.

Table 5.6 summarizes the results between conversation output from Skype and our system using distributed equalization. We find that except for the first two trace sets, where jitters and losses are seldom and MED diversity is small, our VoIP system is preferred over Skype on all other trace sets.

5.5 Generalization Using SVM

5.5.1 Generalization approach

Since our goal is to predict the comparative subjective perceptual quality from the extracted objective metrics of two outputs, the input features we use in SVM are all 22 related objective metrics that might impact the subjective perceptual quality: CE; CS; the minimum (excluding HRD), maximum, average, and variations of MS; the minimum, maximum, average of CMSR; and the minimum and average of PESQ of an output as well as the other 11 objective metrics. The maximum PESQ is not included in the input features, because in our simulation the speech segment of a client with the best PESQ is always the client's own utterance. Table 4.1 has already shown sample input features in comparing non-adaptive play-out scheduling and distributed equalization algorithm using conversation order set 1 in Trace Set 3, and for all the five clients in the conference.

Table 5.3: Objective results for the four play-out scheduling algorithms implemented in our system for Trace Sets 1-7 shown in Table 3.4.

Person	TS/CS/Algorithm	CE	CS	avg CMSR	min PESQ
Person 1		0.707	1.324	1.19	3.482
Person 2		0.708	1.477	1.22	3.374
Person 3	TS1/CS1/A1	0.711	1.362	1.31	3.374
Person 4		0.708	1.273	1.24	3.374
Person 5		0.707	1.364	1.22	3.374
Person 1		0.718	1.348	1.16	3.482
Person 2		0.718	1.525	1.18	3.374
Person 3	TS1/CS1/A2	0.722	1.284	1.23	3.374
Person 4		0.719	1.256	1.19	3.374
Person 5		0.718	1.286	1.15	3.374
Person 1		0.717	1.348	1.17	3.482
Person 2		0.718	1.452	1.18	3.374
Person 3	TS1/CS1/A3	0.721	1.284	1.23	3.374
Person 4		0.718	1.210	1.18	3.374
Person 5		0.717	1.286	1.15	3.374
Person 1		0.716	1.233	1.12	3.482
Person 2		0.716	1.452	1.17	3.374
Person 3	TS1/CS1/A4	0.719	1.179	1.22	3.374
Person 4		0.716	1.226	1.20	3.374
Person 5		0.715	1.227	1.13	3.374

Person	TS/CS/Algorithm	CE	CS	avg CMSR	min PESQ
Person 1		0.715	1.233	1.11	3.482
Person 2		0.715	1.386	1.15	3.374
Person 3	TS2/CS1/A1	0.718	1.233	1.25	3.374
Person 4		0.715	1.226	1.21	3.374
Person 5		0.715	1.227	1.15	3.374
Person 1		0.726	1.248	1.13	3.482
Person 2		0.725	1.375	1.14	3.374
Person 3	TS2/CS1/A2	0.728	1.136	1.15	3.374
Person 4		0.725	1.238	1.18	3.374
Person 5		0.725	1.300	1.16	3.186
Person 1		0.724	1.248	1.12	3.482
Person 2		0.724	1.425	1.16	3.374
Person 3	TS2/CS1/A3	0.727	1.136	1.15	3.374
Person 4		0.724	1.238	1.19	3.374
Person 5		0.724	1.300	1.14	3.374
Person 1		0.719	1.262	1.11	3.482
Person 2		0.72	1.250	1.11	3.374
Person 3	TS2/CS1/A4	0.723	1.130	1.18	3.374
Person 4		0.719	1.175	1.18	3.374
Person 5		0.72	1.238	1.13	3.359

Continued on next page

Note: A1: Non-adaptive play-out scheduling; A2: histogram-based adaptive play-out scheduling (non-cooperative); A3: histogram-based adaptive play-out scheduling (cooperative); A4: distributed equalization. TS: Trace set; CS: conversation set shown in Table 5.1.

Table 5.3: Continued.

Person	TS/CS/Algorithm	CE	CS	avg CMSR	min PESQ
Person 1		0.644	3.142	1.89	0.077
Person 2		0.646	2.875	1.76	0.101
Person 3	TS3/CS1/A1	0.684	2.142	1.61	2.336
Person 4		0.605	2.232	1.64	0.101
Person 5		0.653	2.277	1.52	0.101
Person 1		0.667	2.292	1.62	0.440
Person 2		0.669	2.357	1.58	0.440
Person 3	TS3/CS1/A2	0.671	2.292	1.73	2.385
Person 4		0.628	2.441	1.66	0.440
Person 5		0.673	1.805	1.38	0.440
Person 1		0.665	2.188	1.59	0.440
Person 2		0.667	2.250	1.58	1.311
Person 3	TS3/CS1/A3	0.660	2.188	1.76	2.461
Person 4		0.669	2.325	1.59	1.456
Person 5		0.670	1.678	1.36	1.704
Person 1		0.648	2.130	1.53	0.245
Person 2		0.650	1.904	1.53	1.277
Person 3	TS3/CS1/A4	0.652	1.880	1.74	3.374
Person 4		0.653	1.811	1.56	3.327
Person 5		0.654	1.204	1.27	3.327

Person	TS/CS/Algorithm	CE	CS	avg CMSR	min PESQ
Person 1		0.678	1.642	1.46	3.482
Person 2		0.678	1.717	1.34	3.374
Person 3	TS4/CS1/A1	0.683	1.909	1.63	3.374
Person 4		0.676	1.898	1.54	3.374
Person 5		0.679	1.773	1.41	3.374
Person 1		0.687	1.806	1.50	3.482
Person 2		0.686	1.875	1.34	3.374
Person 3	TS4/CS1/A2	0.691	2.050	1.60	3.374
Person 4		0.684	2.037	1.53	3.374
Person 5		0.688	1.751	1.41	3.374
Person 1		0.684	1.597	1.42	3.482
Person 2		0.683	1.705	1.29	3.374
Person 3	TS4/CS1/A3	0.687	1.952	1.59	3.374
Person 4		0.681	1.941	1.52	3.374
Person 5		0.685	1.682	1.38	3.374
Person 1		0.665	1.517	1.26	3.482
Person 2		0.665	1.493	1.27	3.374
Person 3	TS4/CS1/A4	0.671	1.708	1.62	3.374
Person 4		0.665	1.632	1.50	3.374
Person 5		0.668	1.473	1.31	3.374

Continued on next page

Table 5.3: Continued.

Person	TS/CS/Algorithm	CE	CS	avg CMSR	min PESQ
Person 1		0.678	1.430	1.28	3.265
Person 2		0.677	1.917	1.41	3.374
Person 3	TS5/CS1/A1	0.683	1.792	1.57	3.374
Person 4		0.677	1.751	1.51	3.374
Person 5		0.675	2.036	1.47	3.374
Person 1		0.689	1.405	1.24	3.068
Person 2		0.689	1.913	1.37	3.374
Person 3	TS5/CS1/A2	0.694	1.783	1.47	3.374
Person 4		0.689	1.689	1.43	3.374
Person 5		0.686	2.144	1.48	3.374
Person 1		0.688	1.449	1.25	2.659
Person 2		0.687	1.913	1.34	3.374
Person 3	TS5/CS1/A3	0.693	1.708	1.47	3.374
Person 4		0.687	1.689	1.44	3.374
Person 5		0.685	2.144	1.47	3.374
Person 1		0.673	1.571	1.25	2.555
Person 2		0.674	1.604	1.30	2.974
Person 3	TS5/CS1/A4	0.679	1.512	1.54	3.374
Person 4		0.674	1.306	1.39	2.974
Person 5		0.672	1.941	1.42	2.974

Person	TS/CS/Algorithm	CE	CS	avg CMSR	min PESQ
Person 1		0.614	1.972	1.41	2.456
Person 2		0.613	1.456	1.29	2.456
Person 3	TS5/CS2/A1	0.611	2.085	1.40	3.813
Person 4		0.617	1.500	1.39	2.456
Person 5		0.611	2.204	1.35	2.456
Person 1		0.625	1.859	1.37	1.838
Person 2		0.624	1.387	1.25	1.838
Person 3	TS5/CS2/A2	0.622	1.859	1.34	3.813
Person 4		0.627	1.409	1.34	1.838
Person 5		0.622	2.092	1.34	1.838
Person 1		0.623	1.859	1.35	1.838
Person 2		0.622	1.369	1.25	1.838
Person 3	TS5/CS2/A3	0.619	1.859	1.33	3.813
Person 4		0.625	1.409	1.38	1.838
Person 5		0.620	2.092	1.33	1.838
Person 1		0.592	1.823	1.35	1.830
Person 2		0.598	1.542	1.37	1.830
Person 3	TS5/CS2/A4	0.593	1.590	1.30	3.813
Person 4		0.601	1.393	1.47	1.830
Person 5		0.592	1.880	1.32	1.830

Continued on next page

Table 5.3: Continued.

Person	TS/CS/Algorithm	CE	CS	avg CMSR	min PESQ
Person 1		0.684	1.678	1.23	1.635
Person 2		0.677	1.288	1.13	1.635
Person 3	TS6/CS1/A1	0.681	1.818	1.55	1.635
Person 4		0.665	1.560	1.45	1.689
Person 5		0.675	1.514	1.28	1.635
Person 1		0.668	1.795	1.29	1.887
Person 2		0.663	1.423	1.17	1.887
Person 3	TS6/CS1/A2	0.670	2.400	1.72	1.887
Person 4		0.649	1.684	1.61	0.036
Person 5		0.662	2.083	1.42	1.887
Person 1		0.666	1.710	1.29	1.745
Person 2		0.662	1.423	1.17	1.935
Person 3	TS6/CS1/A3	0.668	2.286	1.72	1.745
Person 4		0.650	1.684	1.59	0.071
Person 5		0.660	1.989	1.40	1.745
Person 1		0.656	1.558	1.30	2.612
Person 2		0.649	1.589	1.22	1.637
Person 3	TS6/CS1/A4	0.655	1.770	1.77	1.637
Person 4		0.648	1.622	1.62	1.637
Person 5		0.648	1.585	1.39	1.637

Person	TS/CS/Algorithm	CE	CS	avg CMSR	min PESQ
Person 1		0.585	1.672	1.41	3.257
Person 2		0.584	1.814	1.58	2.315
Person 3	TS6/CS2/A1	0.584	1.814	1.39	2.315
Person 4		0.592	1.475	1.58	2.315
Person 5		0.586	1.153	1.14	2.315
Person 1		0.592	1.839	1.41	3.257
Person 2		0.591	1.757	1.54	2.390
Person 3	TS6/CS2/A2	0.591	1.852	1.38	2.390
Person 4		0.599	1.493	1.53	2.390
Person 5		0.593	1.153	1.14	2.390
Person 1		0.590	1.672	1.37	3.257
Person 2		0.590	1.678	1.53	2.390
Person 3	TS6/CS2/A3	0.590	1.768	1.39	2.390
Person 4		0.597	1.493	1.54	2.390
Person 5		0.591	1.204	1.14	2.390
Person 1		0.586	1.437	1.34	3.257
Person 2		0.585	1.326	1.47	2.390
Person 3	TS6/CS2/A4	0.585	1.519	1.39	2.390
Person 4		0.592	1.493	1.56	2.390
Person 5		0.585	1.364	1.18	2.390

Continued on next page

Table 5.3: Continued.

Person	TS/CS/Algorithm	CE	CS	avg CMSR	min PESQ
Person 1		0.651	4.733	1.76	0.000
Person 2		0.684	1.795	1.36	3.374
Person 3	TS7/CS1/A1	0.654	5.079	1.73	0.000
Person 4		0.649	5.252	1.94	0.000
Person 5		0.638	5.641	1.95	0.000
Person 1		0.653	3.137	1.83	3.482
Person 2		0.651	3.205	1.83	3.374
Person 3	TS7/CS1/A2	0.655	2.733	1.82	3.374
Person 4		0.651	3.256	2.06	3.374
Person 5		0.605	3.185	1.55	1.539
Person 1		0.651	3.006	1.67	3.482
Person 2		0.649	3.071	1.77	3.374
Person 3	TS7/CS1/A3	0.652	2.733	1.82	3.374
Person 4		0.649	3.101	1.98	3.374
Person 5		0.651	2.988	1.53	2.098
Person 1		0.646	3.006	1.57	3.482
Person 2		0.643	2.789	1.64	3.374
Person 3	TS7/CS1/A4	0.647	2.492	1.79	3.374
Person 4		0.643	3.036	1.81	3.374
Person 5		0.642	3.938	1.58	3.005

The output is the dominant opinion of the subjective comparison results using hypothesis testing. As mentioned in Section 5.1, a person in our study only needs to select one of the following three opinions $(-1, 0, 1)$ corresponding to (worse, about the same, better). There are nine people doing the subjective tests. As discussed in Chapter 4, an opinion is dominant with at least 70% significance if at least six people vote for this opinion. If there are no more than six people voting for any of the three opinions, this subjective comparison leads to an inconclusive result. Therefore, the output of the SVM is one of $(-1, 0, 1, 2)$, where class 2 means inconclusive.

In order to generate learning patterns for the SVM classifier, the learning sample data we use in our study are the objective measures and subjective comparison results of Trace Sets 1, 3, 5, 6, and 7 that can represent all the five classes in Table 3.4 and conversation order set 1 in Table 5.1. To examine whether the learned classifier can generalize the results to unseen similar network conditions, we test the input data generated using Trace Sets 2 and 4 and the same conversational order (Conversation Set 1). To examine whether the classifier can generalize the results to unseen conversational conditions, we test the inputs generated using Trace Sets 5 and 6 and a different conversational order (Conversation Set 2).

Table 5.4: Objective results for Skype's output.

CS1/TS1	CE	CS	MSRAvg	PESQMin
Person 1	0.628	2.237	1.36	2.222
Person 2	0.647	2.727	1.55	1.615
Person 3	0.701	1.995	1.72	1.773
Person 4	0.682	1.725	1.66	2.666
Person 5	0.663	1.674	1.32	1.822

CS2/TS1	CE	CS	MSRAvg	PESQMin
Person 1	0.652	2.128	1.32	2.378
Person 2	0.679	2.148	1.45	1.651
Person 3	0.683	1.988	1.72	2.557
Person 4	0.679	1.567	1.53	1.982
Person 5	0.691	1.242	1.25	2.385

CS3/TS1	CE	CS	MSRAvg	PESQMin
Person 1	0.596	2.917	1.72	1.528
Person 2	0.642	2.988	1.84	1.284
Person 3	0.642	3.899	2.26	1.907
Person 4	0.626	2.348	1.68	1.316
Person 5	0.646	1.534	1.37	1.882

CS4/TS1	CE	CS	MSRAvg	PESQMin
Person 1	0.587	3.219	1.71	2.352
Person 2	0.63	3.405	2.02	2.092
Person 3	0.633	3.242	2.34	1.984
Person 4	0.653	2.052	1.92	2.082
Person 5	0.637	1.484	1.47	2.190

Continued on next page

Abbreviations: TS: Trace order set in Table 3.4; CS: conversation set shown in Table 5.1.

Table 5.4: Continued.

CS5/TS1	CE	CS	MSRAvg	PESQMin
Person 1	0.61	2.843	1.52	2.381
Person 2	0.614	3.227	1.68	1.753
Person 3	0.643	2.805	2.11	1.676
Person 4	0.661	2.232	1.71	1.416
Person 5	0.644	1.260	1.37	2.026

CS5/TS2	CE	CS	MSRAvg	PESQMin
Person 1	0.527	2.081	1.89	0.790
Person 2	0.547	5.452	3.51	1.134
Person 3	0.564	1.736	2.89	0.537
Person 4	0.614	4.113	2.63	1.369
Person 5	0.571	2.007	1.64	1.869

CS6/TS1	CE	CS	MSRAvg	PESQMin
Person 1	0.477	2.177	1.70	0.883
Person 2	0.535	2.354	2.16	0.125
Person 3	0.581	2.280	1.69	0.815
Person 4	0.579	1.770	1.94	0.958
Person 5	0.552	1.338	1.30	0.661

CS6/TS2	CE	CS	MSRAvg	PESQMin
Person 1	0.526	2.056	1.79	1.806
Person 2	0.534	2.175	2.04	1.474
Person 3	0.594	2.571	1.93	2.166
Person 4	0.625	1.517	1.95	1.324
Person 5	0.569	1.431	1.35	1.746

CS7/TS1	CE	CS	MSRAvg	PESQMin
Person 1	0.587	2.867	1.82	1.832
Person 2	0.665	2.994	2.07	0.913
Person 3	0.636	2.772	2.05	1.426
Person 4	0.612	1.716	1.78	1.857
Person 5	0.639	1.338	1.37	1.224

Table 5.5: Subjective comparison results for the four play-out scheduling algorithms implemented in our system.

TS1/CS1	A1 & A2	A1 & A3	A1 & A4	A2 & A3	A2 & A4	A3 & A4
Person 1	1 8 0	0 9 0	0 8 1	1 7 1	0 9 0	1 7 1
Person 2	0 9 0	0 9 0	1 7 1	0 9 0	1 8 0	0 9 0
Person 3	1 8 0	0 9 0	0 8 1	1 7 1	0 9 0	0 9 0
Person 4	1 8 0	0 8 1	1 7 1	0 8 1	1 7 1	0 8 1
Person 5	0 9 0	0 9 0	0 8 1	1 8 0	1 8 0	1 8 0

TS2/CS1	A1 & A2	A1 & A3	A1 & A4	A2 & A3	A2 & A4	A3 & A4
Person 1	1 8 0	0 9 0	1 8 0	2 7 0	0 9 0	1 7 1
Person 2	1 8 0	0 9 0	1 7 1	1 8 0	0 8 1	0 9 0
Person 3	1 8 0	0 9 0	0 8 1	1 7 1	0 9 0	0 9 0
Person 4	2 6 1	0 8 1	1 7 1	0 9 0	1 7 1	2 7 0
Person 5	0 8 1	0 9 0	0 8 1	1 8 0	0 8 1	1 8 0

TS3/CS1	A1 & A2	A1 & A3	A1 & A4	A2 & A3	A2 & A4	A3 & A4
Person 1	6 3 0	7 2 0	7 2 0	1 6 2	1 6 2	1 7 1
Person 2	6 3 0	6 3 0	6 3 0	2 6 1	2 6 1	1 6 2
Person 3	1 7 1	1 7 1	4 5 0	2 6 1	5 4 0	4 5 0
Person 4	1 8 0	3 6 0	8 1 0	4 4 1	8 1 0	7 2 0
Person 5	4 5 0	5 4 0	8 1 0	4 4 1	7 2 0	6 3 0

TS4/CS1	A1 & A2	A1 & A3	A1 & A4	A2 & A3	A2 & A4	A3 & A4
Person 1	1 6 2	2 6 1	3 6 0	2 6 1	3 5 1	3 6 0
Person 2	2 6 1	2 6 1	3 5 1	2 5 2	3 5 1	2 6 1
Person 3	2 6 1	3 5 1	2 5 2	2 5 2	2 5 2	1 7 1
Person 4	1 6 2	0 9 0	2 6 1	2 7 0	2 5 2	2 7 0
Person 5	0 9 0	1 6 2	3 6 0	2 7 0	3 5 1	3 6 0

Continued on next page

Note: A1: Non-adaptive play-out scheduling; A2: histogram-based adaptive play-out scheduling (non-cooperative); A3: histogram-based adaptive play-out scheduling (cooperative); A4: distributed equalization. TS: Trace set shown in Table 3.4; CS: conversation set shown in Table 5.1. Results are presented in (number worse, number same, number better). In our study, there are nine people doing the subjective rating.

Table 5.5: Continued.

TS5/CS1	A1 & A2	A1 & A3	A1 & A4	A2 & A3	A2 & A4	A3 & A4
Person 1	0 6 3	3 3 3	2 4 3	1 7 1	0 9 0	1 6 2
Person 2	2 6 1	3 3 3	1 7 1	0 9 0	2 7 0	0 9 0
Person 3	2 7 0	2 6 1	3 6 0	2 6 1	3 4 2	2 6 1
Person 4	0 9 0	1 7 1	3 5 1	2 5 2	2 6 1	1 7 1
Person 5	2 6 1	2 7 0	2 7 0	1 6 2	2 6 1	1 8 0

TS5/CS2	A1 & A2	A1 & A3	A1 & A4	A2 & A3	A2 & A4	A3 & A4
Person 1	0 6 3	0 9 0	1 8 0	2 7 0	2 6 1	0 8 1
Person 2	1 6 2	0 8 1	1 8 0	4 5 0	2 6 1	1 6 2
Person 3	1 7 1	0 9 0	3 6 0	2 6 1	1 6 2	1 7 1
Person 4	1 6 2	2 5 2	0 8 1	1 6 2	2 7 0	3 6 0
Person 5	2 6 1	1 8 0	0 6 3	0 7 2	1 8 0	1 7 1

TS6/CS1	A1 & A2	A1 & A3	A1 & A4	A2 & A3	A2 & A4	A3 & A4
Person 1	2 6 1	2 7 0	6 3 0	0 9 0	4 5 0	5 4 0
Person 2	1 8 0	1 8 0	1 7 1	1 7 1	1 6 2	1 6 2
Person 3	1 8 0	2 7 0	0 9 0	0 8 1	0 9 0	0 8 1
Person 4	0 6 3	1 6 2	0 9 0	0 9 0	6 3 0	7 2 0
Person 5	2 6 1	0 9 0	1 8 0	1 8 0	0 9 0	0 8 1

TS6/CS2	A1 & A2	A1 & A3	A1 & A4	A2 & A3	A2 & A4	A3 & A4
Person 1	2 6 1	2 7 0	2 5 2	3 6 0	3 6 0	2 5 2
Person 2	4 5 0	2 4 3	1 7 1	1 7 1	3 6 0	1 7 1
Person 3	2 6 1	2 6 1	3 5 1	0 4 5	3 5 1	1 7 1
Person 4	0 8 1	1 7 1	3 5 1	1 7 1	2 5 2	2 6 1
Person 5	1 8 0	0 7 2	1 6 2	2 7 0	1 8 0	2 5 2

TS7/CS1	A1 & A2	A1 & A3	A1 & A4	A2 & A3	A2 & A4	A3 & A4
Person 1	5 4 0	6 2 1	7 1 1	3 5 1	2 3 4	3 4 2
Person 2	1 5 3	0 5 4	0 7 2	3 5 1	3 5 1	2 6 1
Person 3	3 6 0	6 3 0	4 3 2	3 3 3	3 4 2	3 6 0
Person 4	4 5 0	5 4 0	5 2 2	3 3 3	2 6 1	3 5 1
Person 5	5 4 0	7 2 0	7 2 0	5 2 2	4 3 2	3 5 1

Table 5.6: Subjective comparison results for Skype's output and the output from the distributed equalization algorithm implemented in our system.

Person	TS1/CS1	TS2/CS1	TS3/CS1	TS4/CS1
Person 1	2 7 0	3 5 1	7 2 0	7 2 0
Person 2	2 7 0	2 6 1	6 3 0	7 2 0
Person 3	1 7 1	1 8 0	7 2 0	8 1 0
Person 4	0 8 1	2 6 1	7 2 0	6 3 0
Person 5	1 8 0	0 9 0	6 3 0	6 3 0

Person	TS5/CS1	TS5/CS2	TS6/CS1	TS6/CS2	TS7/CS1
Person 1	8 1 0	6 2 1	8 1 0	7 2 0	6 3 0
Person 2	7 2 0	7 2 0	8 1 0	7 2 0	6 3 0
Person 3	8 1 0	8 1 0	8 1 0	7 2 0	7 2 0
Person 4	6 3 0	8 1 0	7 2 0	8 1 0	8 1 0
Person 5	7 2 0	7 2 0	7 2 0	8 1 0	7 2 0

Abbreviations: TS: Trace set shown in Table 3.4. CS: Conversation order set in Table 5.1. Results are presented in (number worse, number same, number better). In our study, there are nine people doing the subjective rating.

Since we have conducted VoIP conferencing simulations on Skype using the same trace sets and conversational order sets as in our own VoIP software, we can examine whether the learned classifier can generalize the results to Skype. We conduct subjective comparisons on our distributed equalization algorithm implemented in our VoIP software with Skype's output using the same network and conversational conditions. Because the input features from Skype are never seen by SVM, in order to enhance the prediction accuracy, comparison results for Trace Set 3 and Conversation Set 1 are also used as the training data. For the testing data used to predict the objective results by SVM, we compare all the waveform outputs from the four play-out scheduling algorithms with Skype's output, so as to validate the accuracy of the learned SVM model.

5.5.2 Generalization results

Table 5.7 summarizes the partial order of the four algorithms and the multi-party Skype in terms of conversational quality preference with 70% statistical significance under the seven network conditions in Table 3.4. For the subjective comparisons, Skype is not included, because we compare Skype only with the distributed equalization algorithm. The classifier outputs include

Table 5.7: The partial order of the four algorithms and the multi-party Skype.

TS & CS	Partial Order (Subjective) Algorithms 1-4				Partial Order (Objective) Algorithms 1-4 & Skype				
	A1	A2	A3	A4	A1	A2	A3	A4	Skype
TS1/CS1	1	1	1	1	1	1	1	1	1
TS2/CS1	1	1	1	1	1	1	1	1	1
TS3/CS1	2	2	2	1	3	3	2	1	3
TS4/CS1	1	1	1	1	1	1	1	1	1
TS5/CS1	1	1	1	1	1	1	1	1	2
TS5/CS2	1	1	1	1	1	1	1	1	2
TS6/CS1	1	1	1	1	1	1	1	1	2
TS6/CS2	1	1	1	1	1	1	1	1	2
TS7/CS1	2	2	1	1	3	2	1	1	3

Note: algorithms and Skype are ordered in terms of conversational quality preference with 70% statistical significance under seven network conditions in Table 3.4: A1: Non-adaptive play-out scheduling; A2: histogram-based adaptive play-out scheduling (non-cooperative); A3: histogram-based adaptive play-out scheduling (cooperative); A4: distributed equalization. TS: Trace set in Table 3.4; CS: conversation order set in Table 5.1. The partial order can be 1, 2, or 3, where 1 is the system with the highest preference, and 3 is the one with the least preference. Shaded TS/CS conditions represent the training data in SVM, while the others represent the testing data.

Skype because we compare all four algorithms with Skype using the SVM learned classifier.

The SVM results for the comparison partial order are represented as 1, 2 or 3 in Table 5.7. A system with lower order has a better conversational preference over one with higher order. Two systems having the same order means that the listening qualities are approximately the same. A system is better than another system if at least three out of five people in the VoIP conferencing have this preference.

Table 5.7 shows that the SVM output matches well with the subjective tests. For trace sets with low delay disparities, losses, and jitters (Trace Sets 1, 2, and 4), all five alternatives are statistically equal. For Trace Sets 5 and 6 and for both conversation order sets, the four algorithms are mutually equal, and each is preferred over Skype.

Figures 5.4 and 5.5 illustrate the partial order for Trace Sets 3 and 7. An arrow indicates a dominating opinion with 70% statistical significance; a line without arrows indicates that a statistically significant relation could not be established.

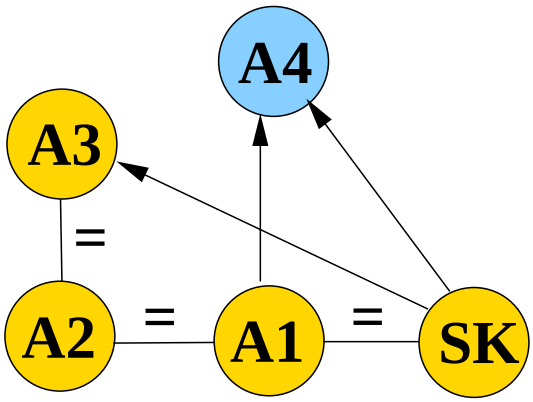


Figure 5.4: Partial orders found for Trace Set 3. An arrow indicates a dominating opinion with 70% statistical significance; a line without arrows indicates that a statistically significant relation could not be established.

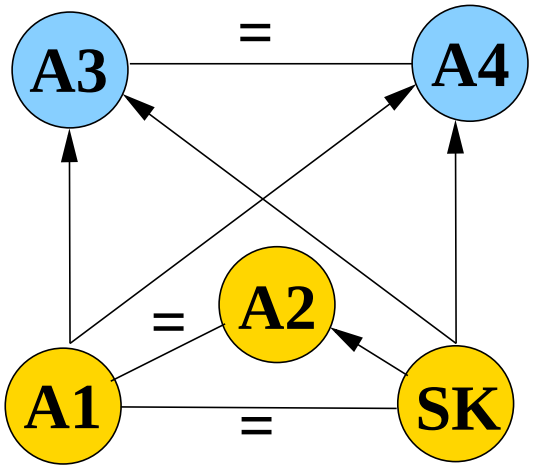


Figure 5.5: Partial orders found for Trace Set 7. An arrow indicates a dominating opinion with 70% statistical significance; a line without arrows indicates that a statistically significant relation could not be established.

5.5.3 Generalization performance

The prediction accuracy of the SVM model is related to the generalization performance. If the SVM can predict which algorithm can achieve a better conversational quality online at unseen network or conversational conditions, the VoIP software can automatically choose this best algorithm.

For the training data indicated in shaded network/conversational conditions in Table 5.7, our learned SVM model can achieve a prediction accuracy of more than 85%. However, because of the limited number of samples in class 2 (inconclusive) as compared to other classes, 60% of the prediction results in this class will mistakenly be classified into class 0 (same). But this misclassification will cause a major problem, because it will not change the partial order of the four play-out scheduling algorithms implemented in our multi-party VoIP software along with the Skype system.

The unseen data can be divided into two categories: (1) the output data for the four play-out schedulings shown in unshaded network/conversational conditions in Table 5.7; (2) comparisons of Skype's output data with output generated from our distributed equalization algorithm for all the nine TS/CS combinations listed in Table 5.7 (except TS3/CS1 used as the training data). For the data in the first category, the SVM model can achieve a prediction accuracy of more than 75%. The major part of false classifications is that our learned SVM model may classify the original class 1 (better) or -1 (worse) into class 0 (same). This is understandable, because the number of samples belonging to class 0 is about twice the number of samples belonging to class 1 or -1. If the number of samples in each class becomes more evenly divided, the prediction accuracy can increase. But right now, the 25% false classification will not pose a critical issue in the partial order of different algorithms, as long as the SVM model does not give a totally reverse classification (i.e., data in class 1 are classified into class -1, or data in class -1 are classified into class 1). For the testing data generated from Skype and our distributed equalization algorithm in the second category, because the listening conversational quality is noticeable, and because we have used Skype's output of TS3/CS1 in Table 5.7 as the training data, the prediction accuracy is more than 80%.

5.5.4 Practical issues and considerations in SVM learning

The top consideration in SVM learning is the the ability of the learning classifier to predict each class correctly. To enhance the correctness, the number of samples in each class should be as evenly divided as possible. The number of samples belonging to class 0 (same) is the largest in our study. We swap the comparison pair at some samples, so that the subjective comparison results 1 (better) or -1 (worse) can be changed to the inverse number, and the number of samples belonging to class 1 and -1 are approximately equal.

Another consideration is the low prediction accuracy of class 2 (inconclusive). The rate can be improved by increasing the weight of class 2 in SVM. However, as was discussed in the last subsection, it is not a critical issue in our current study because it will not change the partial order of the four play-out scheduling algorithms, and because we are mostly interested in finding a play-out scheduling algorithm that can lead to the best subjective results.

5.6 Summary

In this chapter, we have presented objective and subjective comparative results on four play-out scheduling algorithms implemented in our VoIP system and Skype. Our study shows that our overlay approach can adapt to different network conditions and can find a topology with the minimum ME2ED. Our distributed equalization algorithm can guarantee the effect of dynamic play-out scheduling to smooth jitters as well as reduce CS and CMSR up to 25%, while slightly increasing CE when there is a large diversity of MEDs. The subjective results also demonstrate the benefit of distributed equalization in terms of conversational quality.

We have trained an SVM classifier using part of our samples and have validated the classifier using the rest. Although some practical issues still need to be considered, our results show that SVM in general can well predict the subjective comparison result and the partial order of the four algorithms and Skype based on the extracted objective measures. Hence, an SVM classifier can be used at run time to predict which play-out scheduling algorithm can lead to the best subjective perceptual conversational quality.

CHAPTER 6

CONCLUSIONS

This thesis focuses on the problem of improving the subjective conversational quality of multi-party VoIP conferencing over the Internet. The design is affected by two factors: MS variations, caused by the diversity of network latency, and LOSQ, degraded by jitters and losses. Because there are many trade-offs among different objective metrics, it is difficult to formulate a mapping to subjective perceptual results from objective parameters. Since it is costly to do subjective comparison tests, the thesis also proposes a new SVM approach to train a model that can predict the subjective results using objective features and generalize the results to unseen data.

Chapter 1 illustrates the problems our study has addressed. Chapter 2 presents a thorough survey of related work that has been performed so far. Chapter 3 is an analysis of network traffic behavior collected from Planetlab. Chapter 4 proposes objective and subjective metrics that may affect conversational quality, detailed design of multi-party VoIP conferencing system, practical issues, and SVM generalization approaches. Chapter 5 presents the experiment setup and results.

We show in our study and experimental results that our system can improve conversational quality by reducing MS variations and improving LOSQ. We also show that the SVM model can generalize the results and effectively predict the subjective comparison results on-line based on the extracted objective metrics. Based on the SVM model, the system can dynamically adjust its control algorithms based on the objective parameter collected at run time in order to achieve the best subjective quality.

REFERENCES

[1] D. L. Richards, *Telecommunication by Speech*. London, UK: Butterworths, 1973.

[2] P. T. Brady, "Effects of transmission delay on conversational behaviour on echo-free telephone circuits," *Bell System Technical Journal*, vol. 50, no. 1, pp. 115–134, Jan. 1971.

[3] D. Lin, "Real-time voice transmissions over the Internet," M.S. thesis, University of Illinois at Urbana-Champaign, Urbana, IL, 1998.

[4] M. Claypool and J. Tanner, "The effects of jitter on the perceptual quality of video," in *Proceedings of Seventh ACM International Conference on Multimedia*, Nov. 1999, pp. 115–118.

[5] R. Landry and I. Stavrakakis, "Study of delay jitter with and without peak rate enforcement," *IEEE/ACM Transactions on Networking*, vol. 5, no. 4, pp. 543–553, Aug. 1997.

[6] B. Sat and B. W. Wah, "Playout scheduling and loss-concealments in VoIP for optimizing conversational voice communication quality," in *Proceedings of ACM Multimedia*, Sep. 2007, pp. 137–146.

[7] S. ZahirAzami, A. Yongacoglu, L. Orozco-Barbosa, and T. Aboulnasr, "Evaluating the effects of buffer management on voice transmission over packet switching networks," in *Proceedings of IEEE International Conference on Communication*, Jun. 2001, pp. 738–742.

[8] Y. Mansour and B. Patt-Shamir, "Jitter control in QoS networks," *IEEE/ACM Transactions On Networking*, vol. 9, no. 4, pp. 492–502, Aug. 2001.

[9] H. Melvin and L. Murphy, "Time synchronization for VoIP quality of service," *IEEE Internet Computing*, vol. 6, no. 3, pp. 57–63, Jun. 2002.

[10] B. W. Wah and X. Su, "Streaming video with transformation-based error concealment and reconstruction," in *Proceedings of IEEE International Conference on Multimedia Computing and Systems*, Jun. 1999, pp. 238–243.

[11] B. W. Wah, D. Lin, and X. Su, "Streaming real-time audio and video data with transformation-based error concealment and reconstruction," in *Proceedings of IEEE 1st International Conference on Web Information Systems Engineering*, Jun. 2000, pp. 2–11.

[12] C. S. Perkins, O. Hoson, and V. Hardman, "A survey of packet-loss recovery techniques for streaming audio," *IEEE Network*, vol. 12, no. 5, pp. 40–48, Sep. 1998.

[13] ITU-P.862, "Perceptual evaluation of speech quality (PESQ): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs," 2001. [Online]. Available: http://www.itu.int/rec/T-REC-P.862/en/

[14] B. Sat and B. W. Wah, “Analysis and evaluation of the Skype and Google-Talk VoIP systems,” in *Proceedings of IEEE International Conference on Multimedia and Expo*, Jul. 2006, pp. 2153–2156.

[15] N. Kiatawaki and K. Itoh, “Pure delay effect on speech quality in telecommunications,” *IEEE Journal on Selected Areas of Communication*, vol. 9, no. 4, pp. 586–593, May 1991.

[16] ITU-G.114, “One-way transmission time,” 2003. [Online]. Available: http://www.itu.int/rec/T-REC-G.114/en/

[17] ITU-P.800, “Methods for subjective determination of transmission quality,” 1996. [Online]. Available: http://www.itu.int/rec/T-REC-P.800/en/

[18] ITU-P.800.1, “Mean opinion score (MOS) terminology,” 2006. [Online]. Available: http://www.itu.int/rec/T-REC-P.800.1/en/

[19] ITU-P.862.1, “Mapping function for transforming P.862 raw result scores to MOS-LQO,” 2001. [Online]. Available: http://www.itu.int/rec/T-REC-P.862.1/en/

[20] ITU-G.107, “The E-model, a computational model for use in transmission planning,” 2008. [Online]. Available: http://www.itu.int/rec/T-REC-G.107/en/

[21] L. C. et al, “An E-model implementation for speech quality evaluation in VoIP systems,” in *Proceedings of 10th IEEE Symposium on Computers and Communications*, Jun. 2005, pp. 933–938.

[22] 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, no. 3, pp. 382–385, Mar. 2004.

[23] International Telecommunication Union, “ITU study group 12: Performance, QoS and QoE,” 2009-2012. [Online]. Available: http://www.itu.int/ITU-T/studygroups/com12/index.asp/

[24] P. J. Smith, P. Kabal, M. L. Blostein, and R. Rabipour, “Tandem-free VoIP conferencing: A bridge to next-generation networks,” *IEEE Communications Magazine*, pp. 136–145, May 2003.

[25] S. Baset and H. Schulzrinne, “An analysis of the Skype peer-to-peer Internet telephony protocol,” in *Proceedings of IEEE International Conference on Computer Communications*, Apr. 2006, pp. 1–11.

[26] M. King, “Dynamic distributed overlay networking of IP multicast,” in *Proceedings of 3rd International Conference on Peer-to-peer Computing*, Sep. 2003, pp. 10–11.

[27] Z. Duan, Z. L. Zhang, and Y. T. Hou, “Service overlay networks: SLAs, QoS, and bandwidth provisioning,” *IEEE/ACM Transactions on Networking*, vol. 11, no. 6, pp. 870–883, Dec. 2003.

[28] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, “Resilient overlay networks,” in *Proceedings of 18th ACM Symposium on Operating Systems Principles*, Dec. 2001, pp. 131–145.

[29] Y. Amir, C. Danilov, S. Goose, D. Hedqvist, and A. Terzis, “An overlay architecture for high quality VoIP streams,” *IEEE Trans. on Multimedia*, vol. 8, no. 6, pp. 1250–1262, Dec. 2006.

[30] J. Li, "Mutualcast: A serverless peer-to-peer multiparty real-time audio conferencing system," in *Proceedings of IEEE International Conference on Multimedia and Expo*, Jul. 2005, pp. 602–605.

[31] S. G. Rao and S. V. Raghavan, "Fast techniques for the optimal smoothing of stored video," *Multimedia Systems*, vol. 7, no. 3, pp. 222–233, May 1999.

[32] M. Narbutt, A. Kelly, L. Murphy, and P. Perry, "Adaptive VoIP playout scheduling: Assessing user satisfaction," *IEEE Internet Computing*, vol. 9, no. 4, pp. 28–34, Aug. 2005.

[33] C. Boutremans and J. Y. L. Boudec, "Adaptive joint playout buffer and fec adjustment for internet telephony," in *Proceedings of 22nd Annual Joint Conference of the IEEE Computer and Communications Societies*, Mar. 2003, pp. 652–662.

[34] J. Liu and Z. Niu, "An adaptive receiver buffer adjust algorithm for VoIP applications considering voice characters," in *Proceedings of 10th Asia-Pacific Conference on Communications and 5th International Symposium on Multi-Dimensional Mobile Communication*, Aug. 2004, pp. 597–601.

[35] Y. J. Liang, N. Faber, and B. Girod, "Adaptive playout scheduling and loss concealment for voice communication over IP networks," *IEEE Transaction on Multimedia*, vol. 5, no. 4, pp. 532–543, Dec. 2003.

[36] C. A. Rodbro and S. H. Jensen, "Time-scaling of sinusoids for intelligent jitter buffer in packet based telephony," in *Proceedings of IEEE Workshop on Speech Coding*, Oct. 2002, pp. 71–73.

[37] RFC-793, "Transmission control protocol (TCP)," Sep. 1981. [Online]. Available: http://www.ietf.org/rfc/rfc0793.txt

[38] B. W. Wah and D. Lin, "Transformation-based reconstruction for real-time voice transmissions over the Internet," *IEEE Transaction on Multimedia*, vol. 1, no. 4, pp. 342–351, Dec. 1999.

[39] C. Perkins, O. Hodson, and V. Hardman, "A survey of packet loss recovery techniques for streamig audio," *IEEE Network*, vol. 12, no. 5, pp. 40–48, Sep. 1998.

[40] M. Chen and M. N. Murthi, "Optimized unequal error protection for voice over IP," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, May 2004, pp. 865–868.

[41] B. Sat and B. W. Wah, "Speech-adaptive layered G.729 coder for loss concealments of real-time voice over IP," in *Proceedings of IEEE International Conference on Multimedia and Expo*, Jul. 2005, pp. 250–253.

[42] B. Sat and B. W. Wah, "Speech- and network-adaptive layered G.729 coder for loss concealments of real-time voice over IP," in *Proceedings of IEEE International Workshop on Multimedia Signal Processing*, Oct. 2005, pp. 1–4.

[43] J. C. Bolot, S. Fosse-Parisis, and D. Towsley, "Adaptive FEC-based error control for Internet telephony," in *Proceedings of IEEE Conference on Computer Communications*, Mar. 1999, pp. 1453–1460.

[44] V. R. Gandikota, B. R. Tamma, and C. R. Murthy, “Adaptive FEC-based packet loss resilience scheme for supporting voice communication over ad hoc wireless networks,” *IEEE Transactions on Mobile Computing*, vol. 7, no. 10, pp. 1184–1199, Oct. 2008.

[45] N. Degrande, D. D. Vleeschauwer, and K. Laevens, “Protecting IPTV against packet loss: Techniques and trade-offs,” *Bell Labs Technical Journal*, vol. 13, no. 1, pp. 35–51, Mar. 2008.

[46] J. Suzuki and M. Taka, “Missing packet recovery techniques for low-bit-rate coded speech,” *IEEE Journal on Selected Areas in Communications*, vol. 7, no. 5, pp. 707–717, Jun. 1989.

[47] R. C. F. Tucker and J. E. Flood, “Optimizing the performance of packet-switch speech,” in *Proceedings of IEEE Conference on Digital Processing of Signals in Communications*, Apr. 1985, pp. 227–234.

[48] N. Aoki, K. Takano, I. Ohmura, H. Arai, and T. Namerikawa, “Development of a voip system implementing a high quality packet loss concealment technique,” in *Proceedings of Canadian Conference on Electrical and Computer Engineering*, May 2005, pp. 308–311.

[49] A. K. S.A. Atungsiri and B. Evans, “Error control for low-bit-rate speech communication systems,” in *IEE Proceedings I: Communications, Speech and Vision*, Apr. 1993, pp. 97–103.

[50] D. Lin, “Loss concealments for low bit-rate packet voice,” Ph.D. dissertation, University of Illinois at Urbana-Champaign, Urbana, IL, 2002.

[51] ITU, “International Telecommunication Union.” [Online]. Available: http://www.itu.int/

[52] IETF, “Internet Engineering Task Force.” [Online]. Available: http://www.ietf.org/

[53] ITU-G.711, “Pulse code modulation (pcm) of voice frequencies,” 1988. [Online]. Available: http://www.itu.int/rec/T-REC-G.711/en/

[54] ITU-G.726, “40, 32, 24, 16 kbit/s adaptive differential pulse code modulation (ADPCM),” 1990. [Online]. Available: http://www.itu.int/rec/T-REC-G.726/en/

[55] ITU-G.723.1, “Dual rate speech coder for multimedia communications transmitting at 5.3 and 6.3 kbit/s,” 2006. [Online]. Available: http://www.itu.int/rec/T-REC-G.723.1/en/

[56] ITU-G.729, “Coding of speech at 8 kbit/s using conjugate-structure algebraic-code-excited linear prediction (CS-ACELP),” 2007. [Online]. Available: http://www.itu.int/rec/T-REC-G.729/en/

[57] ITU-G.722.2, “Wideband coding of speech at around 16 kbit/s using adaptive multi-rate wideband (AMR-WB),” 2003. [Online]. Available: http://www.itu.int/rec/T-REC-G.722.2/en/

[58] S. Andersen, A. Duric, H. Astrom, R. Hagen, W. Kleijn, and J. Linden, “Internet low bit rate codec (iLBC),” Dec. 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3951.txt

[59] iSAC Codec, “GIPS Internet Speech Audio Codec,” 2007. [Online]. Available: http://www.gipscorp.com/files/english/datasheets/iSAC.pdf

[60] Speex, “Speex: A free codec for free speech.” [Online]. Available: http://www.speex.org/

[61] ITU-G.722, “ITU-G.722 appendix I: Error concealment of erroneous or lost frames,” 2002. [Online]. Available: http://www.itu.int/rec/T-REC-G.722.2/en/

[62] E. Osuna, R. Freund, and F. Girosi, "Support vector machines: Training and applications," Massachusetts Institute of Technology, USA, 2007.

[63] B. Sat and B. W. Wah, "Statistical scheduling of offline comparative subjective evaluations for real-time multimedia," *IEEE Transaction on Multimedia*, submitted for publication.

[64] RFC-879, "TCP maximum segment size and related topics," Nov. 1983. [Online]. Available: http://tools.ietf.org/html/rfc879/

[65] RFC-894, "A standard for the transmission of IP datagrams over ethernet networks," Apr. 1984. [Online]. Available: http://tools.ietf.org/html/rfc894/

[66] SDL, "Simple directmedia layer library." [Online]. Available: http://www.libsdl.org/

[67] C. T. Lin, "Single-channel speech enhancement in variable noise-level environment," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 33, no. 1, pp. 137–143, Jan. 2003.

[68] D. Farrokhi, R. Togneri, and A. Zaknich, "Speech enhancement of non-stationary noise based on controlled forward moving average," in *Proceedings of IEEE International Symposium on Communications and Information Technologies*, Oct. 2007, pp. 1551–1555.

[69] J. Radecki, Z. Zilic, and K. Radecka, "Echo cancellation in IP networks," in *Proceedings of IEEE 45th Midwest Symposium on Circuits and Systems*, Aug. 2002, pp. 219–222.

[70] G. Periakarruppan and H. A. A. Rashid, "Packet based echo cancellation for voice over Internet protocol," in *Proceedings of IEEE International Conference on Communication Networks*, Nov. 2005, pp. 1–4.

[71] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, Feb. 1995.

[72] C. C. Chang and C. J. Lin, "A library for Support Vector Machines." [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/libsvm/

[73] LIGHT-SVM, "Light Support Vector Machine." [Online]. Available: http://svmlight.joachims.org/

[74] B. Boser, I. Guyon, and V. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of ACM 5th Annual Workshop on Computational Learning Theory*, 1992, pp. 144–152.