

**A VOICE OVER IP SOLUTION TO THE PROBLEM OF MOBILE  
RADIO INTEROPERABILITY**

By

JOHN H. MOCK

B.S. University of New Hampshire, 2001

THESIS

Submitted to the University of New Hampshire

in Partial Fulfillment of

the Requirements for the Degree of

Master of Science

in

Electrical Engineering

May, 2003

This thesis has been examined and approved.

---

Thesis Director, Dr. W. Thomas Miller, III  
Professor of Electrical Engineering

---

Dr. William Lenharth  
Research Associate Professor of Electrical  
Engineering

---

Dr. Andrew L. Kun  
Assistant Professor of Electrical Engineering

---

Date

## **DEDICATION**

This thesis is dedicated to my parents for their constant support, love and guidance during all moments of my life.

## **ACKNOWLEDGEMENTS**

First, I would like to thank my advisor, Dr. W. Thomas Miller, III, for his advice, guidance and constant availability throughout my research and the realization of my thesis. I appreciate his support and the example he has set for me.

I would like to thank Dr. William Lenharth and Dr. Andrew L. Kun for serving on my thesis committee and taking the time to review my thesis and share their ideas.

I would like to thank the CATLab team for there assistance. I would also like to thank Danielle for her love and guidance throughout this process.

Finally, I would like to thank my family and friends for their love, understanding and support throughout my entire time at UNH.

## TABLE OF CONTENTS

DEDICATION.....	iii
ACKNOWLEDGEMENTS.....	iv
LIST OF FIGURES .....	ix
LIST OF TABLES.....	xi
LIST OF ACRONYMS .....	xii
LIST OF ACRONYMS .....	xii
ABSTRACT .....	xiv
CHAPTER	PAGE
I. INTRODUCTION.....	1
II. BACKGROUND .....	4
2.1 VOIP STANDARDS.....	4
2.1.1 H.323 .....	4
2.1.2 H.323 COMPONENTS .....	6
2.1.3 TERMINAL .....	6
2.1.4 GATEWAY .....	7
2.1.5 GATEKEEPER.....	7
2.1.6 MULTIPPOINT CONTROL UNIT.....	8

2.1.7 SESSION INITIATED PROTOCOL (SIP) .....	8
2.1.8 SIP COMPONENTS .....	9
2.1.9 USER AGENTS .....	9
2.1.10 PROXY SERVER .....	9
2.1.11 REDIRECT SERVER .....	10
2.1.12 REGISTRAR SERVER .....	10
2.2 REAL-TIME TRANSPORT PROTOCOL .....	11
2.3 PREVIOUS WORK IN MOBILE RADIO INTEROPERABILITY .....	12
2.4 FACTORS AFFECTING VOIP PERFORMANCE .....	13
2.4.1 DELAY .....	13
2.4.2 JITTER, PACKET LOSS, AND BANDWIDTH LIMITATIONS .....	14
2.5 VOICE QUALITY EVALUATION .....	16
2.5.1 SUBJECTIVE EVALUATION OF VOICE QUALITY .....	16
2.5.2 OBJECTIVE EVALUATION OF VOICE QUALITY .....	18
2.6 MICROSOFT TELEPHONY APPLICATION PROGRAMMING INTERFACE .....	19
<b>III. SYSTEM DESCRIPTION .....</b>	<b>24</b>
3.1 DESIGN REQUIREMENTS .....	24
3.2 SYSTEM LEVEL DESCRIPTION .....	25
<b>IV. SYSTEM HARDWARE .....</b>	<b>27</b>
4.1 PROTOTYPE MOBILE RADIO/PC INTERFACE .....	27
<b>V. SYSTEM SOFTWARE .....</b>	<b>29</b>

5.1 FIRST VIRTUAL COMMUNICATIONS CONFERENCE SERVER .....	29
5.1.1 AUDIO MIXING.....	30
5.1.2 SILENCE DETECTION / BACKGROUND NOISE SUPPRESSION.....	30
5.1.3 AUDIO REQUIRING H.323 ENDPOINTS.....	31
5.1.5 SERVER CONFIGURATION.....	31
5.2 PROJECT54 CONFERENCE CLIENT APPLICATION .....	32
5.2.2 P54 CONFERENCE CLIENT APPLICATION SOFTWARE DESCRIPTION .....	33
5.2.3 GRAPHICAL USER INTERFACE .....	36
5.3 PROJECT54 CONFERENCE CONFIGURATION APPLICATION .....	39
5.3.1 CONFIGURATION APPLICATION SOFTWARE DESCRIPTION.....	40
5.3.2 GRAPHICAL USER INTERFACE .....	42
<b>VI. SYSTEM TESTING.....</b>	<b>46</b>
6.1 SYSTEM LEVEL DESCRIPTION .....	46
6.2 NIST NET ROUTER .....	48
6.3 TEST SETUP.....	49
6.4 SOURCE AND RECORDED FILES .....	52
6.5 TEST SCENARIO.....	52
6.6 EVALUATING VOICE QUALITY USING PESQ .....	52
6.7 TESTING FOR DELAY .....	53
6.8 PC/RADIO INTERFACE.....	55
<b>VII. RESULTS AND DISCUSSION .....</b>	<b>56</b>
7.1 RESULTS OF BANDWIDTH LIMITATION.....	56

7.2 RESULTS OF INCREASING PERCENTAGE OF DROPPED PACKETS.....	60
7.3 RESULTS OF INCREASING LEVEL OF JITTER.....	63
7.3 RESULTS OF DELAY TESTING .....	66
7.4 RESULTS OF PC/RADIO INTERFACE TESTING .....	68
7.5 DISCUSSION OF RESULTS.....	69
VIII. CONCLUSION.....	72
IX. SUGGESTIONS FOR FUTURE ENHANCEMENTS .....	74
REFERENCES .....	76
APPENDIX A.....	78
APPENDIX B .....	79
APPENDIX C .....	82
APPENDIX D.....	86
APPENDIX E .....	95
APPENDIX F .....	99
APPENDIX G.....	103



## **LIST OF FIGURES**

Figure 2.1 H.323 Architecture	5
Figure 2.2 H.323 Terminal	7
Figure 2.3 Diagram of TAPI Architecture	20
Figure 2.4 Diagram of TAPI Object Interaction	21
Figure 3.1 System Level Diagram	26
Figure 4.1 Block Diagram of PC/Radio Interface in System	27
Figure 4.2 Circuit Diagram of Mobile Radio/PC Interface	28
Figure 5.1 Screenshot of Web-Based GUI	32
Figure 5.2 Flowchart of Conference Client Application (CCA)	34
Figure 5.3 Screenshot of Full Featured CCA Graphical User Interface	36
Figure 5.4 Screenshot of CCA Graphical User Interface with Limited Features	37
Figure 5.5 Flowchart of Conference Configuration Application	40
Figure 5.6 Screenshot of Full Featured Configuration App Graphical User Interface	42
Figure 5.7 Screenshot of Configuration App Graphical User Interface with Limited Features	43
Figure 6.1 Diagram of Test System	46
Figure 6.2 Connections between Test System Soundcards	47
Figure 6.3 Screenshot of NIST GUI	48
Figure 7.1 Average PESQ Scores for Varied Bandwidth - S1	57
Figure 7.2 Average PESQ Scores for Varied Bandwidth - S2	57
Figure 7.3 Average PESQ Scores for Varied Bandwidth - S3	58
Figure 7.4 Average PESQ Scores as Bandwidth is Decreased – S1, S2, S3	59

Figure 7.5 Average PESQ Scores for Increased Drop Percentage - S1	60
Figure 7.6 Average PESQ Scores for Increased Drop Percentage - S2	61
Figure 7.7 Average PESQ Scores for Increased Drop Percentage - S3	61
Figure 7.8 Average PESQ Scores as Drop Percentage is Increased – S1, S2, S3	62
Figure 7.9 Average PESQ Scores for Increasing Jitter - S1	63
Figure 7.10 Average PESQ Scores for Increasing Jitter - S2	64
Figure 7.11 Average PESQ Scores for Increasing Jitter - S3	64
Figure 7.12 Average PESQ Scores as Level of Jitter is Increased – S1, S2, S3	65
Figure 7.13 Round Trip Times - Nashua	66
Figure 7.14 Round Trip Times – Keene	67
Figure 7.15 Round Trip Times - New London	67
Figure 7.16 Round Trip Times - Rindge	68
Figure D.1 Save the Conference Button	93
Figure E.1 Conference Configuration Application GUI	96
Figure F.1 Project54 Conference Client Application GUI	100

## LIST OF TABLES

Table 2.1 G.114 Limits for One-Way Transmission Time .....	14
Table 2.2 Bandwidth Requirements for Various Audio Codecs.....	15
Table 2.3 MOS Quality Opinion Scale .....	17
Table 5.1 Audio Codecs Supported by FVC Conference Server.....	30
Table 5.2 Common FVC Server Telnet Commands .....	31
Table 5.3 Conference Attributes .....	38
Table 6.1 Units for Values in Text Files.....	50
Table 6.2 Locations and IP Addresses Used in Delay Tests.....	54
Table 7.1 Response Time Versus Input Voltage for PC/Radio Interface .....	69
Table 7.2 Mapping Between Test Scores, PESQ Scores and MOS.....	70
Table D.1 Links to Five Main Areas of User Interface .....	91

## LIST OF ACRONYMS

<b>ACR</b>	Absolute Category Rating
<b>BSD</b>	Bark Spectral Distortion
<b>CCA</b>	Conference Client Application
<b>COM</b>	Component Object Model
<b>DLL</b>	Dynamic Link Library
<b>FVC</b>	First Virtual Communications
<b>GUI</b>	Graphical User Interface
<b>HTTP</b>	Hyper Text Transfer Protocol
<b>ICMP</b>	Internet Control Message Protocol
<b>IETF</b>	Internet Engineering Task Force
<b>IETF</b>	Internet Engineering Task Force
<b>ITU</b>	International Telecommunications Union
<b>LAN</b>	Local Area Networks
<b>MC</b>	Multipoint Controller
<b>MCU</b>	Multipoint Control Unit
<b>MNB</b>	Measuring Normalized Blocks
<b>MOS</b>	Mean Opinion Score
<b>MP</b>	Multipoint Processor
<b>MSP</b>	Media Stream Provider
<b>MSPI</b>	Media Stream Provider Interface

<b>PSQM</b>	Perceptual Quality Speech Measure
<b>PSTN</b>	Public Switched Telephone Network
<b>PTT</b>	Push-To-Talk
<b>QoS</b>	Quality of Service
<b>RAS</b>	Registration/ Admission/ Status
<b>RIP</b>	Routing Information Protocol
<b>RPC</b>	Remote Procedure Call
<b>RTP</b>	Real-time Transport Protocol
<b>SIP</b>	Session Initiated Protocol
<b>SMTP</b>	Simple Mail Transfer Protocol
<b>SNR</b>	Signal-to-Noise Ratio
<b>TAPI</b>	Telephony Application Programming Interface
<b>TSP</b>	Telephony Service Provider
<b>TSPI</b>	TAPI Service Provider Interface
<b>VoIP</b>	Voice over IP

## **ABSTRACT**

### **A VOICE OVER IP SOLUTION TO THE PROBLEM OF MOBILE RADIO INTEROPERABILITY**

by

John H. Mock  
University of New Hampshire, May, 2003

While performing their duties, emergency service personnel in mobile units from various departments need to communicate with each other. Currently, unless every department has identical radios, this is impossible due to lack of radio interoperability.

This research addresses the need for these non-interoperable mobile units to communicate with one another. It is not financially possible for these agencies to each purchase new, matching radios; instead a solution is being developed using Voice over IP (VoIP). This solution involves establishing a network consisting of a single central server and clients located at each of the departments. Interfaced to each client is a radio that is compatible with the individual department's mobile radio units. Each client runs an H.323 compliant application that was developed using Microsoft's Telephony Application Programming Interface 3.0 (TAPI 3.0).

A prototype system has been developed and tested against network impairments similar to those present across the Internet. Preliminary results show the system performs robustly when faced with these network impairments.

# **CHAPTER 1**

## **INTRODUCTION**

Project54 is a collaborative research and development effort between the University of New Hampshire and the New Hampshire Department of Safety and is supported by the U.S. Department of Justice. The goal of the project is to apply new technologies to law enforcement so that officers can perform their duties more safely and efficiently [1].

While handling emergency situations, police officers and other emergency service personnel in mobile units from various agencies have a need to communicate with each other. Currently, unless every department has identical radios, this is very difficult due to lack of radio interoperability. Mobile radios from different manufacturers operate on different frequency bands or in the case of digital radios may use different proprietary encryption techniques. It is rare that these radios will be able to communicate with each other. The motivation for the work presented here was the need for these non-interoperable mobile units to communicate with one another. It is not financially possible for these agencies to each purchase new, identical radios; instead a solution is being developed using Voice over Internet Protocol (VoIP). This solution involves establishing a VoIP network consisting of a single central server and a client located at each of the participating departments.

When a mobile unit wishes to communicate with a mobile unit from a different agency, instead of doing so directly, the transmission will traverse the established

network. Transmission will begin at the mobile unit and travel to its base station. At its base station, the transmission will be processed by the client and sent to the conference on the server. The server will then distribute the transmission to whoever has joined the conference. Each client computer at the different agencies then sends the transmission out using a radio compatible with its mobile units

The central server will host a voice conference that follows the ITU H.323 standard for audio, video, and data communications across the Internet. An H.323 [2] compliant application was developed using Microsoft's Telephony Application Programming Interface 3.0 (TAPI 3.0) [3] to run on a desktop computer at the client side. A radio that is compatible with the individual departments' mobile radio units is interfaced to the desktop computer at the client. The interface is necessary to pass audio between the PC and the radio. When sending audio from the radio to the PC, a connection can be made from the radio's speaker output to an audio input on the PC. Sending audio from the PC to the radio is more difficult because of the Push-to-Talk button on the radio. This button must only be depressed whenever there is voice present.

An in-lab prototype system has been developed and tested against network impairments similar to those present across the Internet. This testing was necessary due to the unpredictable performance characteristics and lack of Quality of Service (QoS) in most IP networks, such as the Internet.

This document is designed to describe the development and testing process of the Project54 VoIP system to allow incompatible radios to communicate with each other. Chapter 2 provides background information about key concepts in the transmission of voice over packet-based networks. Chapter 3 describes the requirements used in the



design and offers a high level description of the system. Chapter 4 describes a hardware component that was designed for the system to allow transmission over Motorola Astro digital radios without having to push the Push-To-Talk button. The system designed utilizes both off-the-shelf and custom software. Chapter 5 describes the off-the-shelf software used and details the development of the custom software designed specifically for this system. Because of the unpredictable nature of transmission over IP networks, the system had to be thoroughly tested against real-world conditions. Chapter 6 details the test setup and process used in the evaluation of this system's performance. The results of the testing performed and discussion of those results are shown in Chapter 7. Chapter 8 gives concluding remarks to this thesis and Chapter 9 suggests future enhancements to the Project54 VoIP System.

## **CHAPTER 2**

### **BACKGROUND**

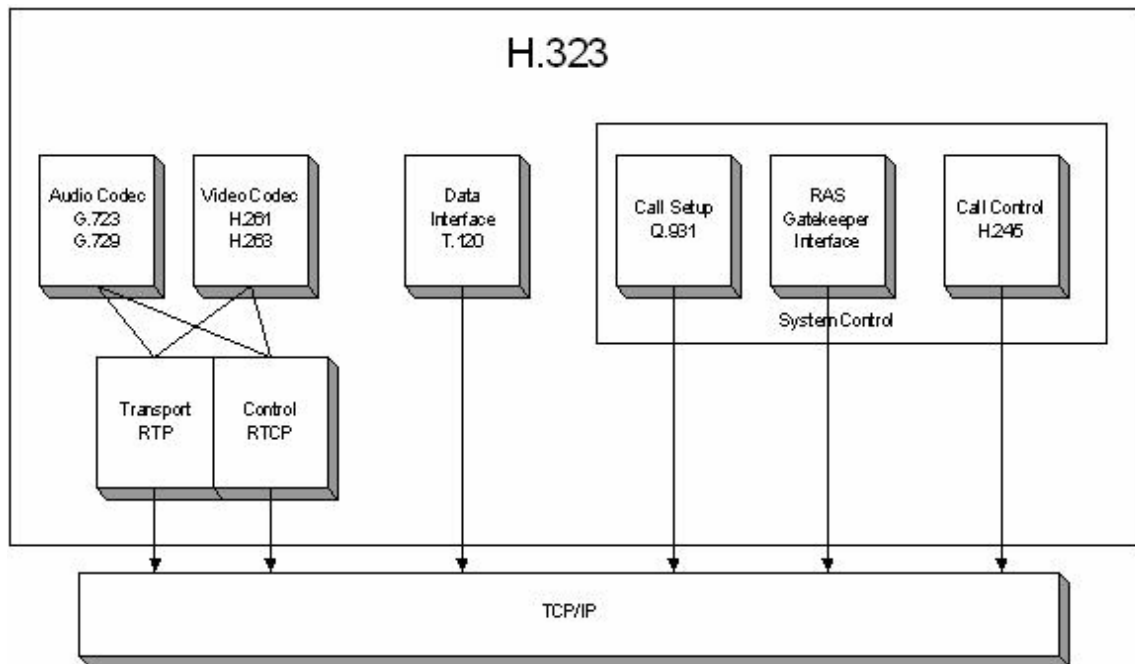
#### **2.1 VoIP Standards**

There are currently two competing standards for voice transmission over packet networks. These standards are the International Telecommunications Union's (ITU) H.323 and the Internet Engineering Task Force's (IETF) Session Initiated Protocol (SIP) [4]. The ITU is an organization who establishes standards for a wide range of telecommunications sectors, from radio to telephone to Internet communications. The IETF is an organization who establishes standards mainly for the evolution of the Internet architecture. H.323 was invented first and is currently the most popular and widely used standard. SIP is somewhat new and threatens to unseat H.323 as the standard, but currently is not widely used.

##### **2.1.1 H.323**

The ITU H.323 standard is predominately used in the VoIP industry. It is not a single standard, but a group of standards which address various aspects of multimedia communication, from control signaling to packet transport. Currently there are more products and support for H.323 than any other VoIP standard. By complying with H.323, it is possible for VoIP products and applications from various vendors to interoperate. It was originally developed for multimedia conferencing over a Local Area Network (LAN) but later extended with more support for communication over the Internet. H.323

provides a foundation for audio, video, and data communications across IP based networks which do not provide a guaranteed Quality of Service (QoS). H.323 uses the Real-time Transport Protocol (RTP) to counter the effect of LAN latency that may occur due to lack of QoS. Data communications are supported by the T.120 standard which is referenced within H.323. The H.323 standard also addresses the issues of call control, multimedia management, bandwidth management, and interfaces between LANs and other networks. Weaknesses were found with the original standard such as lack of support for communication with telephones on the Public Switched Telephone Network (PSTN). To address these weaknesses, a new version of H.323 was developed, known as H.323 version 2. Figure 2.1 shows the H.323 architecture.



**Figure 2.1 H.323 Architecture**

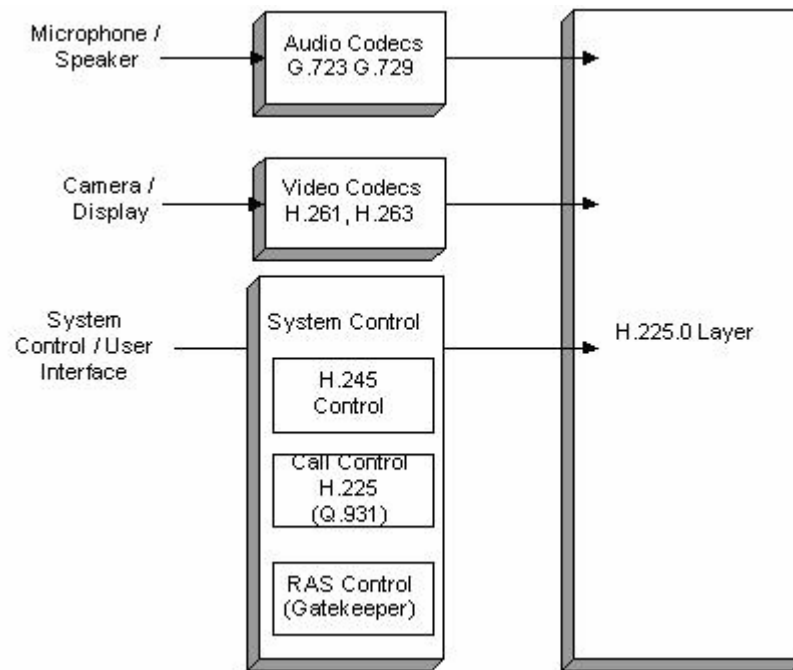
### **2.1.2 H.323 Components**

The H.323 standard specifies four entities, which when networked together, provide point-to-point and point-to-multipoint communications. These entities are:

- Terminals
- Gateways
- Gatekeepers
- Multipoint Control Units (MCUs)

### **2.1.3 Terminal**

Terminals are the client endpoints on the network. All terminals must support voice communications; video and data support is optional. The terminal may be implemented on a desktop PC or any other device which has support for the H.323 protocol stack shown in Figure 2.1. All terminals must support the H.245 standard which is used to negotiate channel usage and terminal capabilities. Three other components are also required: H.245 (also known as Q.931) for call setup, Registration/ Admission/ Status (RAS) for communication with a gatekeeper, and RTP support to overcome the effects of LAN latency. Optional support may be included for video communication and T.120 data conferencing. A diagram of an H.323 terminal is shown in Figure 2.2.



**Figure 2.2 H.323 Terminal**

#### **2.1.4 Gateway**

A gateway is used to bridge together H.323 networks with non-H.323 networks. This bridging is accomplished by translating the protocols for call setup and release and converting media streams to appropriate formats for each network. An example would be an IP/PSTN gateway. If communication is occurring strictly over an IP network, no gateway is needed.

#### **2.1.5 Gatekeeper**

A gatekeeper provides the management of an H.323 zone. An H.323 zone is the collection of all terminals, gateways, and MCUs that are under the management of a single gatekeeper. Every gatekeeper must perform the following functions: address translation (routing), admission control, minimal bandwidth control (also user bandwidth request processing), and zone management. Optional features for the gatekeeper to

implement include: call control handling (Q.931), gatekeeper management information, and directory services.

#### **2.1.6 Multipoint Control Unit**

A Multipoint Control Unit (MCU) supports conferencing between three or more H.323 endpoints. It can be a stand-alone device or integrated into a gatekeeper, gateway or terminal. MCUs typically consist of two components; a Multipoint Controller (MC) which is required and a Multipoint Processor (MP) which is optional. The MC handles conference resource management and the negotiation signaling for conference setup. The MP handles the processing of media streams. It receives the streams from endpoints, processes them, and then returns them to other endpoints in the conference.

#### **2.1.7 Session Initiated Protocol (SIP)**

SIP, like H.323 is a standard for voice transmission over packet based networks but that is where the similarities between the two end. SIP is a protocol for initializing, managing, and terminating voice and video communications sessions over packet-switched networks. It is being developed by the IETF specifically for use over the Internet. Unlike H.323, the SIP protocol was not designed to address issues such as packet transport, resource allocation, or internetworking with non-SIP entities such as the PSTN. RTP is used by SIP to counter the effects of latency over the network. SIP sessions can contain two or more users and may use unicast or multicast methods of transmission. Similar to protocols like HTTP and SMTP, SIP is text encoded which makes its implementation simple and makes it greatly extensible.

### **2.1.8 SIP Components**

A SIP network is composed of four types of logical SIP entities. Each entity has specific function and during a SIP session can take on the role of client, server, or both. One physical device can perform the functions of multiple logical entities. The four types of entities are:

- User Agents
- Proxy Servers
- Redirect Servers
- Registrars

### **2.1.9 User Agents**

The User Agent is the endpoint entity in a SIP session. It initiates and terminates sessions by exchanging responses and requests. A User Agent consists of both a User Agent client and a User Agent server application. The client application initiates requests for connections with others while the server application listens for and accepts other User Agent connection requests.

### **2.1.10 Proxy Server**

A Proxy Server is an entity that takes on the role of both a client and server for the purpose of making requests for other clients. Requests are serviced from within the Proxy Server or by passing them on to other servers. Translation of the request may be performed before passing the request on to others. A Proxy interprets, and possibly, rewrites a request message before forwarding it.

#### **2.1.11 Redirect Server**

A Redirect Server accepts requests from a client who is interested in initiating a SIP call with another client. The Server determines if the client destination address is correct. If it is correct, an acknowledgement is sent back to the source indicating so. If it is not correct, the Redirect Server then returns the correct destination address back to the source and the call is then made directly from client to client. Unlike the Proxy Server, the Redirect Server does not pass on the requests to other servers.

#### **2.1.12 Registrar Server**

A Registrar Server accepts REGISTER requests for the purpose of updating a location database. The Server then returns the contact information of the user specified in the request.



## **2.2 Real-time Transport Protocol**

The Real-time Transport Protocol (RTP) is an Internet Engineering Task Force (IETF) application layer protocol designed to handle the requirements of data with real-time characteristics, such as streaming audio and video, over the Internet [5]. It can be used over multicast or unicast networks. RTP usually works in conjunction with another protocol called the Real Time Control Protocol. RTP provides four main functions which are:

- Feedback Information: This is used to check the quality of the data distribution.
- Transport-Level Identification: This is used to keep a track of the members of a communications session.
- Transmission Interval Control: This guarantees that control traffic will be limited to at most 5% of the overall session traffic.
- Minimal Session Control: This is optional but may be used to convey a minimal amount of information to all session participants.

### **2.3 Previous Work in Mobile Radio Interoperability**

Previous efforts in the field of mobile radio interoperability have yielded a number of different solutions. One solution is to have one of each of the interoperable radios at each headquarters. At the headquarters, these radios would be interconnected and communication received from a mobile unit would then be transmitted using these interconnected radios to their respective mobile units. This solution has been suggested by JPS Communications who have implemented it in regions such as Chicago, Florida, and Montana [6]. This solution has proven successful, but costly. The product (ACU-1000 Modular Interconnect System) developed by JPS Communications to interconnect radios is between \$15,000 and \$25,000 and would be required at each headquarters. Purchasing one of each of the incompatible radios for each headquarters adds greatly to the cost of this solution.

Others have developed solutions similar to the one discussed in this thesis which route mobile radio communication over the Internet. Catalyst Communications is one company who has developed such products. Catalyst offers software products for use on desktop computers to interface with mobile radios and route their transmissions over IP networks such as the Internet [7]. While the Catalyst products do offer a viable solution to the problem of radio interoperability, they do so using proprietary technologies. A system designed using Catalyst products is very inflexible and cannot incorporate any other companies' products or technologies.

## **2.4 Factors Affecting VoIP Performance**

There are four major factors affecting the quality of voice transmission over packet networks [8]. These are:

- Delay
- Jitter (variable interpacket delay)
- Packet Loss
- Bandwidth Limitations

### **2.4.1 Delay**

End-to-end delay does not affect voice quality directly, but instead affects the flow of conversation and may result in talker overlap [8]. ITU-T Recommendation G.114 provides limits for one-way transmission time (delay) [9]. These times are shown in Table 2.1. If the delay through the network is less than 150 ms, users will not notice. When the delay is between 150 ms and 400 ms, users will notice hesitation in the conversation. This hesitation can affect how each listener perceives the mood of the conversation and interruptions are more frequent. As the delay approaches 400 ms, the conversation gets out of beat. Beyond 400 ms, the delay is very obvious to users and conversation becomes virtually impossible.

One-Way Transmission Time (Delay)	User Acceptance
0 to 150 ms	Acceptable for most users
150 to 400 ms	Acceptable, but has impact on conversation
400 ms and above	Unacceptable

**Table 2.1 G.114 Limits for One-Way Transmission Time**

### **2.4.2 Jitter, Packet Loss, and Bandwidth Limitations**

Jitter is the variability in inter-packet delay and is introduced by the variable transmission time through a network. Jitter can be reduced to some extent through the use of jitter buffers at endpoints. Jitter buffers collect and hold packets long enough to allow the slowest packets to arrive in time to be reassembled in the correct order. Collecting and holding packets results in additional delay in the system therefore jitter buffers can only compensate for limited amounts of jitter. Because delays through networks vary greatly, there is no maximum buffer size, and consequently no maximum level of jitter, that can be specified for all systems. If the level of jitter grows beyond what a jitter buffer is designed to handle, audio will be played back without the packets that have not arrived. Thus, the results of excessive jitter on voice quality are similar to those of the affects of packet loss.

Bandwidth limitations can have a severe effect on voice quality in a system. The minimum bandwidth required for the transmission of voice over an IP network is dependant on the audio codec being used. In addition to the voice information, various header information increases the minimum bandwidth requirements. Bandwidth requirements for various audio codecs are shown in Table 2.2.

Audio Codec	Encoding	Bandwidth Required (information alone)	Bandwidth Required (with IP header)
G.711	PCM	64 kbps	80 kbps
G.723	ACELP	5.6 / 6.4 kbps	16.2 / 17.1 kbps
G.729A	CS – ACELP	8 kbps	24 kbps

**Table 2.2 Bandwidth Requirements for Various Audio Codecs**

Packet loss will also have a negative effect on voice quality through the system.

A relatively low percentage (five to ten percent) of lost packets will cause a noticeable decrease in voice quality. If the percentage of lost packets rises above ten percent, quality will become unacceptable.

## **2.5 Voice Quality Evaluation**

The evaluation of voice quality across a degraded channel within a packetized network is not a trivial issue. Traditional analytical techniques, such as Signal-to-Noise Ratio (SNR), are not accurate when determining the effects of a degraded channel on voice transmissions. These techniques compare the received with the transmitted waveforms and assume any discrepancies in the received waveform were caused by errors introduced by the channel. Many of the non-linear codecs used in VoIP transmissions compress the audio so that the perceptually important information (how the audio sounds) is preserved, but not necessarily the audio waveform and frequency spectrum information. Because the waveforms are not preserved by the coding process, a direct comparison of the received and transmitted waveforms would not give a true indication to the level of degradation of voice quality through the system [10].

### **2.5.1 Subjective Evaluation of Voice Quality**

There is no absolute physical definition of voice quality; the only baseline available is subjective perception of human listeners. Subjective testing of voice quality uses humans to listen to original and degraded transmitted voice signals and has them express their opinion as to the level of degradation and whether that level is acceptable for communication [11]. The Mean Opinion Score (MOS) is one such subjective measure [12]. The basic procedure for the subjective test to obtain an MOS is as follows:

1. A panel of test subjects listens to a speech sample
2. Each subject assigns a quality score, selected from Table 2.3, to each sample
3. The average of the panel's scores is that sample's MOS

Quality of Speech	MOS Score
Excellent	5
Good	4
Fair	3
Poor	2
Bad	1

**Table 2.3 MOS Quality Opinion Scale**

The main problem with subjective testing is the difficulty in its implementation. These tests are slow and heavily dependant on the test subjects and environment. Because these tests rely on the subjects' opinions, many subjects are needed so the average MOS reflects an accurate measure of quality. The environment the tests are performed in will also influence results; therefore it is desired to have an acoustically identical environment for each test iteration.

### **2.5.2 Objective Evaluation of Voice Quality**

Objective testing of voice quality does not involve listening tests or human subjects. Objective tests attempt to measure the differences between received and transmitted signals to determine the quality of the received signal [13]. Testing methods which model the physiology of human hearing and sound perception have been developed and show a high correlation with MOS scores. Some objective testing methods are:

- Bark Spectral Distortion (BSD)
- Perceptual Evaluation of Speech Quality (PESQ)
- Perceptual Speech Quality Measure (PSQM)
- Measuring Normalized Blocks (MNB)

These tests are all similar in that they attempt to mimic the human perception of sounds to assess quality measurements, but vary in levels of correlation with MOS. The tradeoff with objective tests is that while they may not deliver identical results as subjective tests, they are much quicker and easily repeatable.

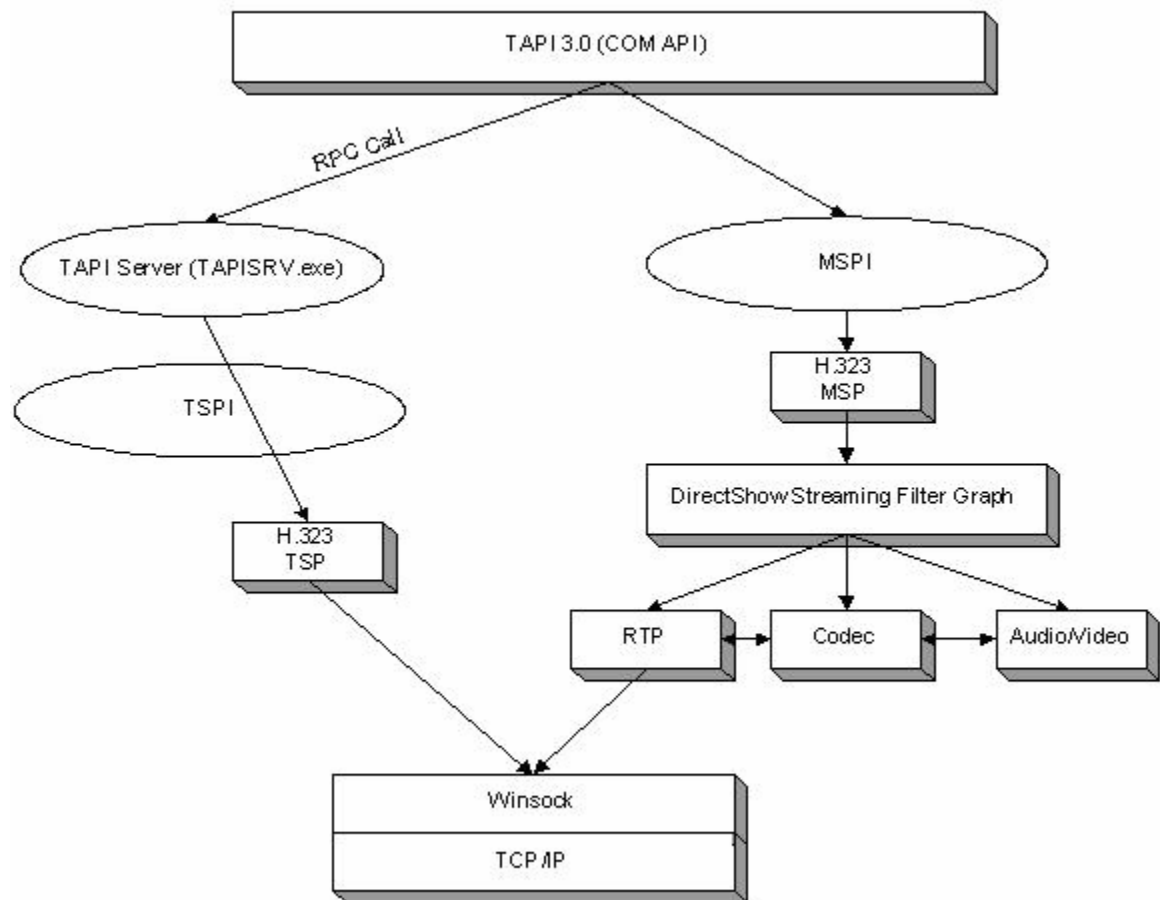


## **2.6 Microsoft Telephony Application Programming Interface**

TAPI 3.0 provides simple and generic methods for making connections between two computers and accessing any media streams involved in that connection [3]. It is implemented as a suite of Component Object Model (COM) [14] objects which allow future component upgrades and also allows developers to create TAPI 3.0 enabled applications in any programming language which supports pointers. There are four major components to TAPI 3.0:

- TAPI 3.0 COM API
- Telephony Service Providers
- Media Stream Providers
- TAPI Server

A diagram of the TAPI architecture is shown in Figure 2.3.

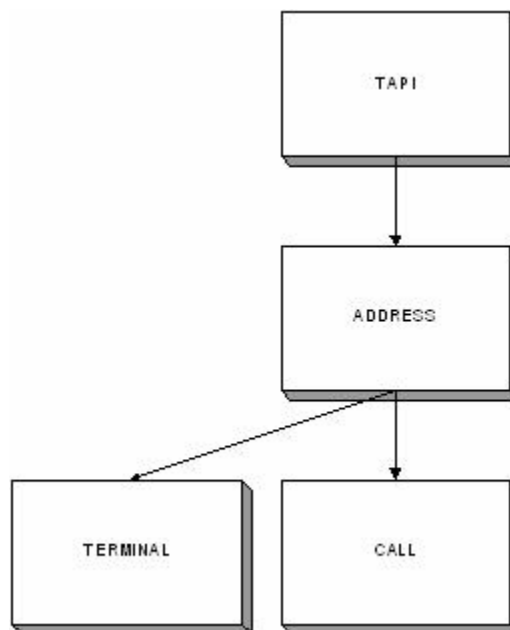


**Figure 2.3 Diagram of TAPI Architecture**

There are four objects contained in the TAPI 3.0 API:

- TAPI
- ADDRESS
- TERMINAL
- CALL

The relationship between TAPI 3.0 objects is shown in Figure 2.4.



**Figure 2.4 Diagram of TAPI Object Interaction**

The TAPI object is the applications entry point to TAPI 3.0. This object represents all telephony resources to which the local computer has access, allowing an application to enumerate all local addresses which may include network cards or modems. An ADDRESS object represents the origination or destination point for a call. Address capabilities such as media and terminal support can be retrieved from this object.

An application can wait for a call on an ADDRESS object or use the ADDRESS object to create an outgoing CALL object. The CALL object represents the connection between the local address and a remote address. All call control, such as connecting and disconnecting, is done through the CALL object. The TERMINAL object represents the sink, or renderer, at the termination or origination point of the connection. In the Project54 VoIP system the TERMINAL object is mapped to hardware, such as speakers or a microphone, but can also be mapped to a file.

The process to place a TAPI call is:

1. Create and initialize a TAPI object.
2. Use the TAPI object to enumerate the available ADDRESS objects (for example network card or modem) on the computer.
3. Enumerate the supported address types (ex. IP addresses or phone numbers) for each ADDRESS object.
4. Choose an ADDRESS object based on desired address type and media type (such as audio or video) supported.
5. Use the CreateCall method of the ADDRESS object to create a CALL object associated with a particular destination address and media types.
6. Select appropriate terminals (ex. microphones, speakers, files) to associate with the CALL object. These terminals are selected according to user preferences and direction of media streams of a call. For example, it may be desired to render a media stream using a speaker or a file, but it is not possible to render a stream using a microphone.
7. Call the Connect method of the CALL object to connect the call.

A Telephony Service Provider (TSP) is a dynamic link library (DLLs) that is responsible for providing a layer of abstraction between the protocol independent TAPI interfaces with protocol specific call-control mechanisms. A TAPI application uses standardized commands, TAPI then passes information to a TSP and the TSP then handles the specific commands that must be exchanged with a device. TSPs are analogous to device drivers and communicate directly with hardware devices such as modems. In the case of H.323 communications, the associated TSP implements the all of the necessary H.323 signaling.

Just as a TSP abstracts call control, a Media Stream Provider (MSP) abstracts media control for an application. An MSP provides a uniform method of accessing all streams involved in a call and supports the DirectShow API as the primary stream handler. Communication between TAPI function calls and the MSP are abstracted by the Media Stream Provider Interface (MSPI). MSPs must be paired with a TSP and implement DirectShow interfaces for the associated TSP.

The TAPI Server process (TAPISRV.exe) abstracts TAPI Service Provider Interface (TSPI) from TAPI 3.0 and earlier versions to allow TSPs developed for earlier versions of TAPI to be compatible with TAPI 3.0. TSPI is the interface TAPI uses to communicate with the TSPs. TAPISRV.exe is implemented as a service process within SVCHOST [15] and communicates with TAPI applications through a Remote Procedure Call (RPC) interface.

## **CHAPTER 3**

### **SYSTEM DESCRIPTION**

#### **3.1 Design Requirements**

One of the major goals of this system is that it should be “open” or based on existing standards. While there are commercial systems which utilize VoIP to route radio communication over the Internet, these systems use proprietary products and technologies. These systems do not conform to established standards and are not interoperable with other systems from other companies. If a company’s system is used which utilizes proprietary technologies, all other components in the system must also be from the same company. This would make the system very rigid in terms of design requirements and it could not be tailored to an individual agency’s needs. The system should be based on industry wide standards so that standards-based commercial products can be used interchangeably throughout the system. The system should also be based on industry wide standards to allow the development of products to suit specific users’ needs.

The system should be able to operate over low bandwidth Internet connections to make the system available to as many departments as possible. In many rural areas of states such as New Hampshire, only low bandwidth Internet connections (accessible through telephone lines) are available. Even if a high bandwidth connection is available, in the event of a power outage this connection may become unavailable. It is likely that backup power and a telephone line will be available during a power outage therefore the

system will continue to be useable if only a low bandwidth connection is required for acceptable system performance.

The system must be cost effective if it is to go into wide spread implementation. If the system cost approaches that of purchasing new mobile radios it will not be practical.

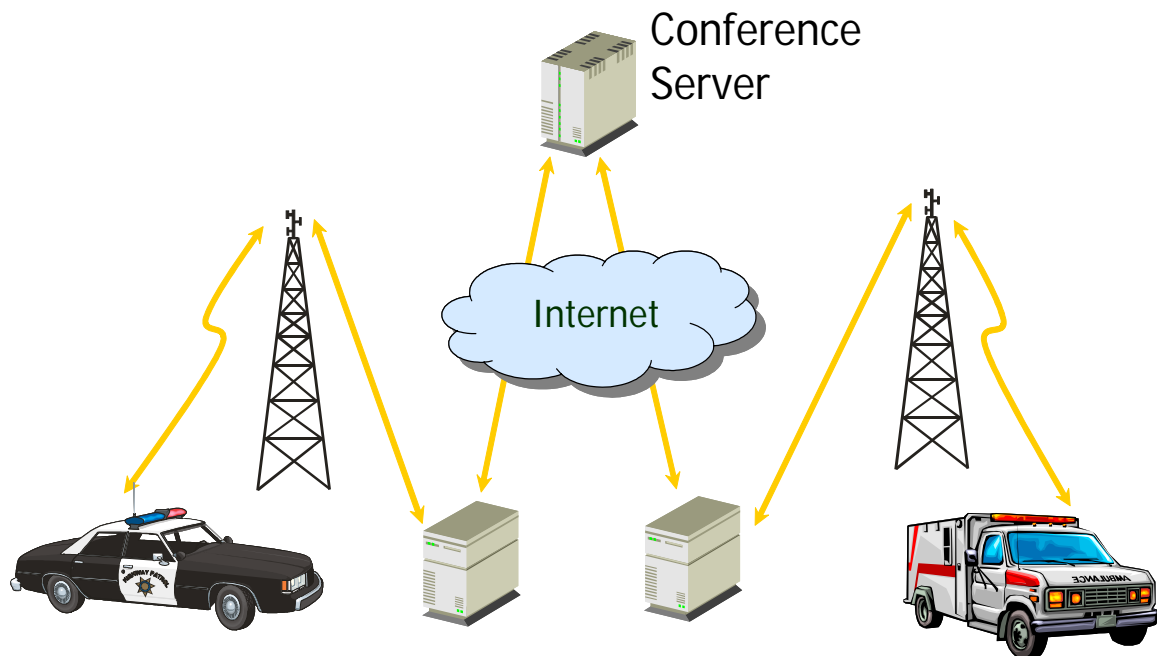
Voice quality through the system must be considered. Because voice is packetized and sent across an IP network, with no guaranteed QoS, there is no way to know if, when, or in what order packets will arrive. Lost, delayed, or reordered packets will have a negative impact on voice quality. The voice quality through the system must be similar to that of current mobile radios or the system will not be desirable.

### **3.2 System Level Description**

When a mobile unit wishes to communicate with an incompatible mobile unit it will first transmit back to its headquarters. At the originating unit's headquarters is a desktop computer connected through an interface to a radio compatible with the mobile units. The interface between the computer and radio allows audio to be exchanged between the two. The interface then passes the mobile unit's transmission from the radio to the computer. This interface also activates the radio's Push-To-Talk (PTT) button to transmit communications from other conference members that is received by the computer. Transmission then travels from the desktop computer to the conference server where it is then redistributed to other members of the same conference. When the transmission is received at the other conference member's headquarters, the PTT button of the radio is activated. Once the PTT button is activated, audio from the computer will

be transmitted over the radio and to the final destination, the previously inaccessible mobile unit.

The desktop computer in the system is running a H.323 compliant software application that is used to connect to a conference server and join an established VoIP audio conference. The conference server is used to host multiple H.323 compliant conferences and mixes all media streams received from conference members and redistributes them to other members of the same conference. A system level diagram is shown in Figure 3.1.



**Figure 3.1 System Level Diagram**

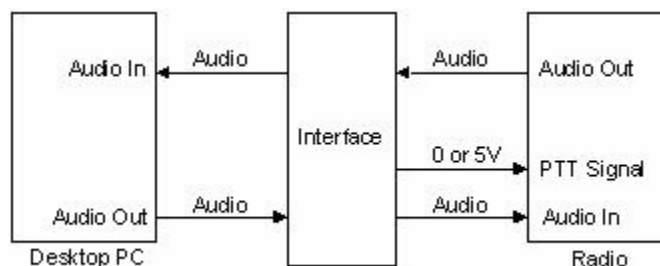


## CHAPTER 4

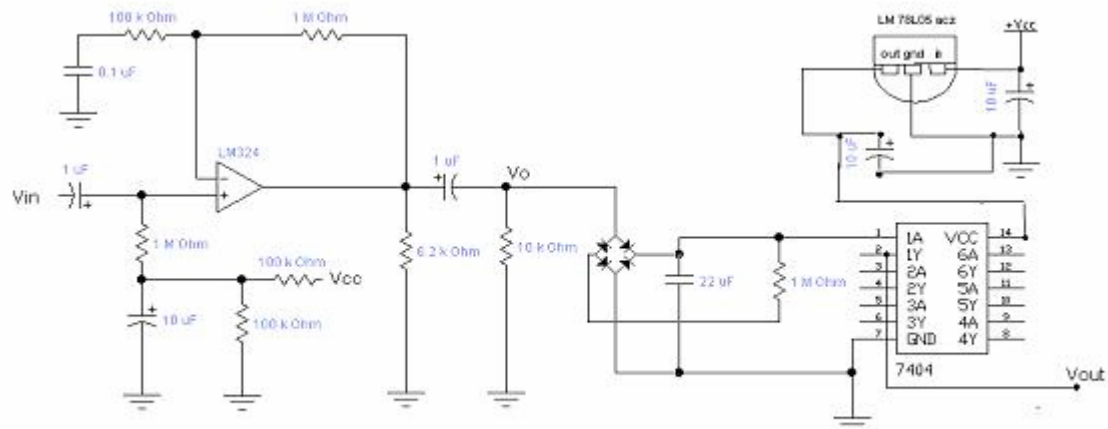
### SYSTEM HARDWARE

#### 4.1 Prototype Mobile Radio/PC Interface

In order to exchange audio between the desktop PC client and a mobile radio in this system, an interface is needed. A block diagram showing the interface in the system is shown in Figure 4.1. The interface's main function is to activate the radio's Push-To-Talk (PTT) button whenever audio is sent from the PC to the radio. Sending audio from the radio to the PC is trivial and is accomplished with a cable connection between the radio's speaker output and the audio input on the PC's soundcard. A prototype hardware solution for the interface to send audio from the PC to the radio was developed and is shown in Figure 4.2.



**Figure 4.1 Block Diagram of PC/Radio Interface in System**



**Figure 4.2 Circuit Diagram of Mobile Radio/PC Interface**

The interface outputs a logic low (0 volts) signal, which indicates to the radio that the PTT signal is active, whenever a voice signal is input. The choice of voltage levels is due to the requirements of the Astro brand radio by Motorola, which this interface was designed for. The interface consists of three stages. The first stage is an op-amp based amplifier. This stage is used to amplify the audio signal from the PC which is necessary for stage 2 to operate properly. The second stage consists of a full-wave rectifier followed by a low-pass filter. The output of stage 2 is a signal which is logic high whenever a voice signal is present and logic low otherwise. Stage 3 is a digital inverter which inverts the output of stage 2. The final result is a signal that is 0 volts whenever voice is being transmitted and 5 volts at all other times.

## **CHAPTER 5**

### **SYSTEM SOFTWARE**

#### **5.1 First Virtual Communications Conference Server**

First Virtual Communications' (FVC) Conference Server is used to host the VoIP conferences. Conference Server is a standards based (including H.323 and SIP) software multipoint control unit (MCU) that runs on the Windows 2000 Advanced Server platform [16]. A maximum of 25 users may be connected to the server at any one time, but H.323 simple cascading is supported to extend the number of users. H.323 simple cascading allows Conference Server to connect to other Conference Server MCUs to distribute users of a single conference across multiple servers. Once connected to the remote MCU, the local Conference Server is viewed as a single node, likewise, the remote server and clients are also viewed as a single node. When multiple servers are cascaded, audio is only delivered to those who require it for delivery to local endpoints. Conference Server supports the following audio codecs: G.711, G.723, and G.729A. Details of each codec are shown in Table 5.1.

Audio Codec	Coding Technique	Bandwidth Required (kbit/sec)
G.711	Pulse Code Modulation (PCM)	64
G.723	Multi-pulse Max Likelihood Quantization Algebraic Code-Excited Linear Prediction (MP-MLQ ACELP)	5.6
		6.4
G.729A	Algebraic Code-Excited Linear Prediction (ACELP)	8

**Table 5.1 Audio Codecs Supported by FVC Conference Server**

### **5.1.1 Audio Mixing**

The mixing of multiple audio streams into a single stream is supported. Conference Server has the ability to mix up to eight individual streams into a single stream which is then redistributed to all clients with the exception of the sender. The streams to include in the mix are switched according to the dialog flow in a manner to minimize audio clipping. Audio clipping occurs at the beginning or end of a stream because voice activity was not detected properly.

### **5.1.2 Silence Detection / Background Noise Suppression**

Conference Server analyzes each audio stream it receives to determine which should be mixed and redistributed. If an audio stream is active but much lower in magnitude than others, it is assumed the user is silent and the active stream is background noise. This active stream is not mixed or redistributed to others to conserve bandwidth.

### **5.1.3 Audio Requiring H.323 Endpoints**

Some H.323 clients need to receive a stream for the entire time they are connected to a conference. If the stream to these clients ends, they view this as a disconnection from the server. To support these clients, Conference Server generates a stream of silence when no audio is present.

### **5.1.5 Server Configuration**

Conference Server is configured by a set of Telnet commands. Some of the more common commands are shown in Table 5.2. These commands can be issued directly during a Telnet session or through the web-based Graphical User Interface (GUI) included with the server, shown in Figure 5.1.

Command	Switches	Parameters	Function
Conf	N/A	N/A	Displays list of conferences and conference attributes
Conf	-a	Conference Name, Conference Number	Adds a conference
Conf	-d	Conference Number	Deletes a Conference
Pref-aud-codecs	-c	Conference Number, Audio Codec	Specifies required codec for a conference
User-conf	N/A	User IP address, Conference Number	Informs server in which conference to place user upon connection

**Table 5.2 Common FVC Server Telnet Commands**

NETWORK

CONFERENCES

USERS

MONITORING

HELP

Add H.323

Add CUSM

Edit

Copy

Remove

ID	Conference Name	E	V	A	C	H	W	O	B	P	R	S	M	T	N	?	@	&	Users
1	Portsmouth Dover Durham	*	*	*	*	*													0
2	Nashua Hudson Londonderry	*	*	*	*	*													0
3	Concord Barnstead	*	*	*	*	*													0

☒ Keep List Updated

Conference Attributes:

[E] Enabled:	n/a	[B] Broadcast:	n/a
[V] Video:	n/a	[P] Private:	n/a
[A] Audio:	n/a	[R] Root:	n/a
[C] Chat:	n/a	[S] Self-reflect:	n/a
[H] H.323:	n/a	[M] Multicast:	n/a
[W] CU-SeeMe Data Collaboration:	n/a	[T] Template:	n/a
[O] Broadcast w/Participation:	n/a	[N] Continuous Presence:	n/a
[?] SIP Enabled:	n/a	[@] T.120 Encryption:	n/a
[&] People&Content:	n/a		

Figure 5.1 Screenshot of Web-Based GUI

## 5.2 Project54 Conference Client Application

The Project54 Conference Client Application (CCA) is an H.323 compliant application for connecting to and participating in H.323 audio conferences. It is a Win32 application built using Microsoft's Telephony Application Programming Interface (TAPI) version 3.0 which is based on Microsoft's Component Object Model (COM). The use of TAPI 3.0 in the application requires that Microsoft Windows 2000 or Windows XP be used as an operating system. Earlier versions of Windows do not support TAPI 3.0.

### **5.2.2 P54 Conference Client Application Software Description**

There are two main parts of the CCA. The first part handles communication with the server to obtain information from the server including conferences currently available to join. The second handles the joining of conferences and the flow of audio between the client and server. A flowchart for the CCA is shown in Figure 5.2.

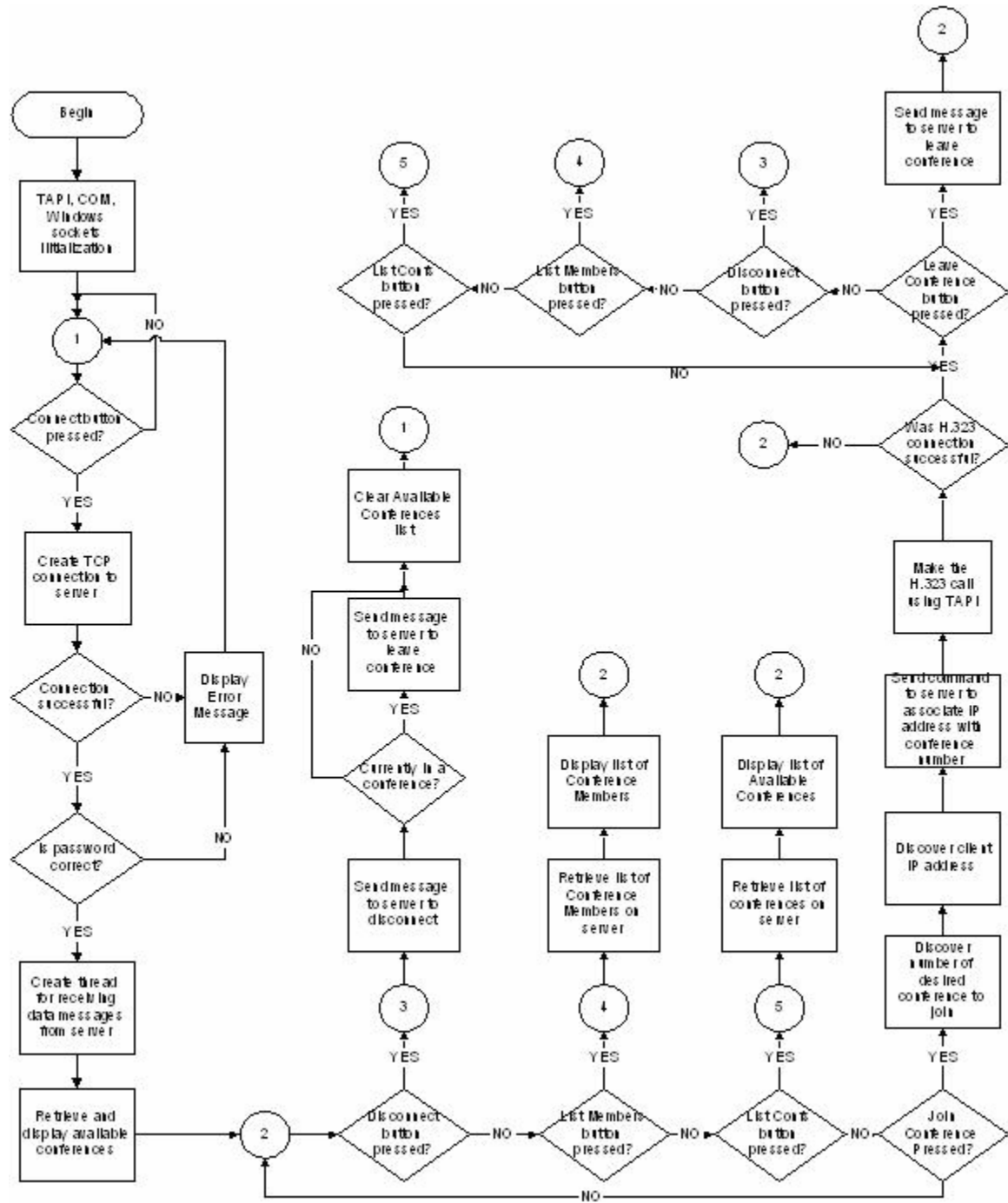


Figure 5.2 Flowchart of Conference Client Application (CCA)



When the CCA is started, a configuration file is checked to determine if certain administrative features should be displayed in the GUI. These features offer additional user information and control of the application. The CCA then waits for the user to enter an IP address and password and choose to connect to the conference server. If the IP address and password are valid, a TCP connection is made to the conference server. If the connection is successful, a separate thread is spawned to receive messages from the server. The server is then queried for a list of conferences currently available to join. Once this list is received it is displayed in the Available Conferences area of the CCA. The CCA then waits for one of four events to take place. The user can choose to update the list of conferences, list the members of a particular conference, join a conference, or disconnect from the conference server.

When the user chooses to join a conference, the desired conference name is retrieved. The client's IP address is then discovered. The TCP connection with the server is used to send this information along with a message signaling the server to associate the client's IP address with the specified conference name. This association is the method by which the client informs the server which conference it wishes to join after an H.323 connection is made.

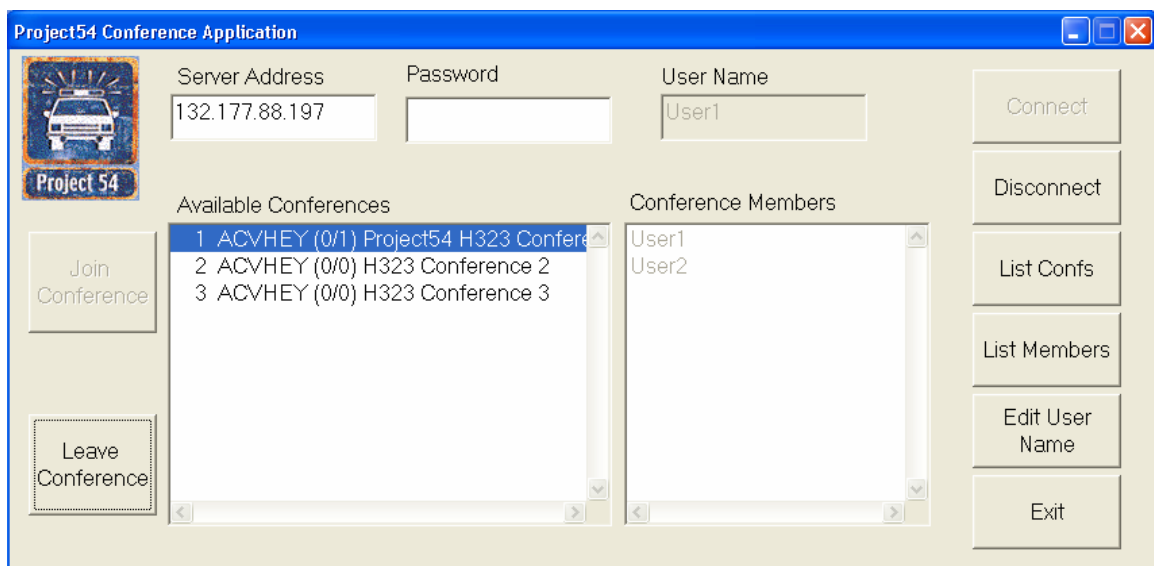
The available TAPI Address objects on the system are then discovered and the object that supports audio is used. This address object is then used to create a TAPI Call object which is used to communicate with the server. The primary audio capture (microphone) and playback devices (speakers) are then discovered. These devices will be used during conferencing and will be unavailable to other applications until the CCA is shut down. The Call object is then used to make an H.323 connection to the server. Any

audio input to the microphone will be sent to the server to be distributed to other members of the conference and audio received from the server will be played over the speakers.

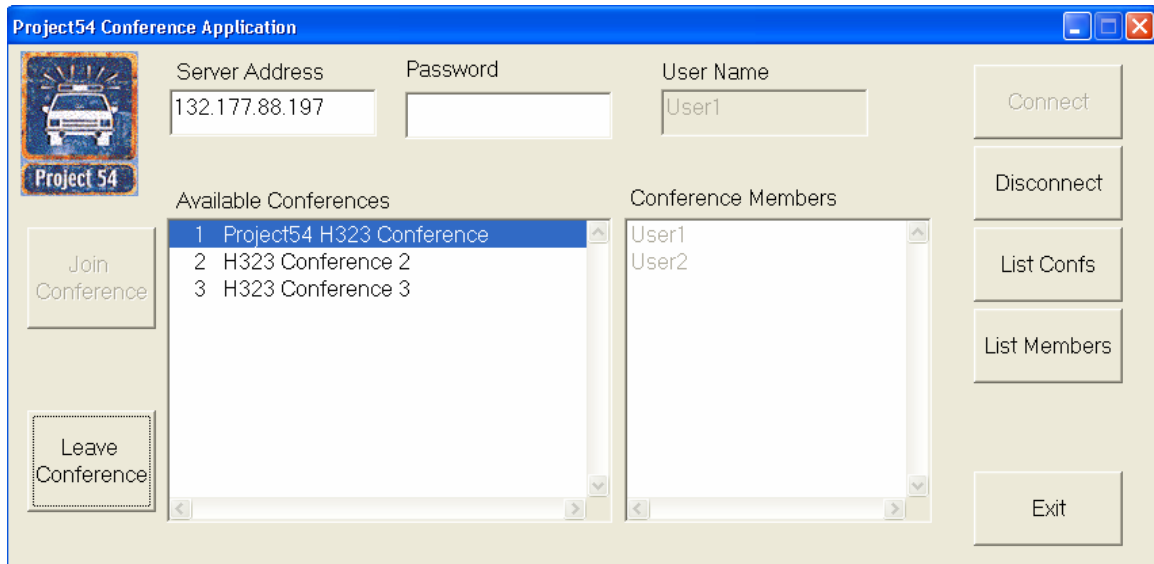
The audio codec used by the client is determined by settings within the operating system and must be the same as the one the conference is configured to use. If a connection attempt with the wrong audio codec is made, the connection will be refused by the server.

### **5.2.3 Graphical User Interface**

A screen shot of the full featured version of the CCA graphical user interface (GUI) is shown in Figure 5.3 and the limited featured version is shown in Figure 5.4.



**Figure 5.3 Screenshot of Full Featured CCA Graphical User Interface**



**Figure 5.4 Screenshot of CCA Graphical User Interface with Limited Features**

There are two versions of the GUI because most users will not require all of the information and features of the full-featured version. The full-featured version will be used by system administrators for debugging or special configuration situations. This version provides more information about the configuration of each conference in the “Available Conferences” window and the ability to change the current user name via the “Edit User Name” button. The letters preceding the conference name indicate conference attributes that are enabled. Table 5.3 shows the possible conference attributes. After the conference attributes is an indication of the number of video senders and lurkers (users who have no camera or choose not to send a video stream) in the following form: (senders/lurkers). Currently, the video features of the conference server are not used in this application. This causes the number of video senders to always be displayed as zero and the number of video lurkers is the number of conference members. The “Edit User Name” button is used to change the current user name. Pressing this button will allow the

“User Name” field to become accessible and the button text will change to “Set Name”.

When the “Set Name” button is pressed the new user name is set to be whatever has been entered in the “User Name” field and the button text returns to “Edit User Name”.

Code	Attribute
A	Audio
B	Broadcast
C	Chat
E	Enable / Disable
H	H.323 Enabled
I	Invitation Only
K	Loopback
L	Local
M	Multicast
N	Continuous Presence
O	Broadcast with Participation
P	Private
R	Root
S	Self-reflect
T	Template
U	Authentication
V	Video
W	White Board
X	Multicast / Broadcast
Y	Audio Mixing
\$	SIP
@	T.120 encryption
&	People + Content

**Table 5.3 Conference Attributes**

The process of connecting to a conference server and participating in conferences is identical for both versions of the GUI. Before a connection is made, a conference server’s IP address and password must be entered into their respective fields.

After these entries are made, the button labeled “Connect” is used to make a TCP connection to the conference server. If a connection could not be made, a message is displayed indicating the error which prevented the connection. Once connected, a list of conferences hosted on the server is displayed in the “Available Conferences” field. This list can be updated at any time while connected to the server by pressing the “List Confs” button. The list of members currently participating in a conference is displayed by selecting the desired conference and pressing the “List Members” button. A list of conference members is then displayed in the “Conference Members” field. A conference is joined by selecting it in the “Available Conferences” field and pressing the “Join Conference” button. If the conference was joined successfully, the “Join Conference” button is disabled and the “Leave Conference” button is enabled. If it is desired to join another conference, the “Leave Conference” button must be used before joining another conference. If the user wishes to leave a conference and disconnect from the server, the “Disconnect” button may be pressed. The “Disconnect” button handles leaving a conference and disconnecting from the server.

### **5.3 Project54 Conference Configuration Application**

The Project54 Conference Configuration Application (Configuration App) is used for remote configuration of a FVC Conference Server. It is a Win32 application designed to provide a user-friendly interface for exchanging information and conference configuration.

### 5.3.1 Configuration Application Software Description

The main purpose of the Configuration App is exchanging information with and issuing commands to an FVC Conference Server. A flowchart of the Configuration App is shown in Figure 5.5.

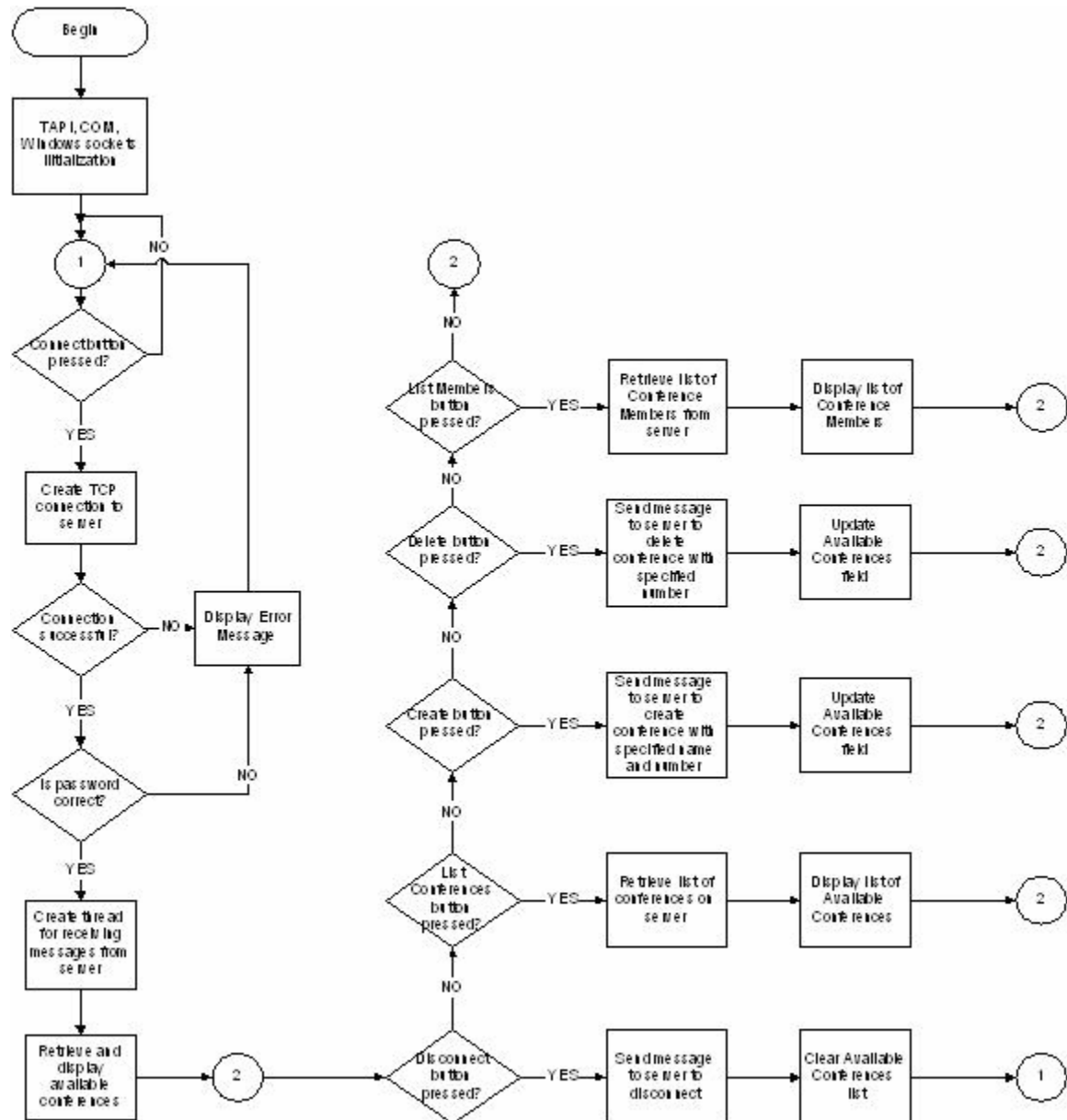


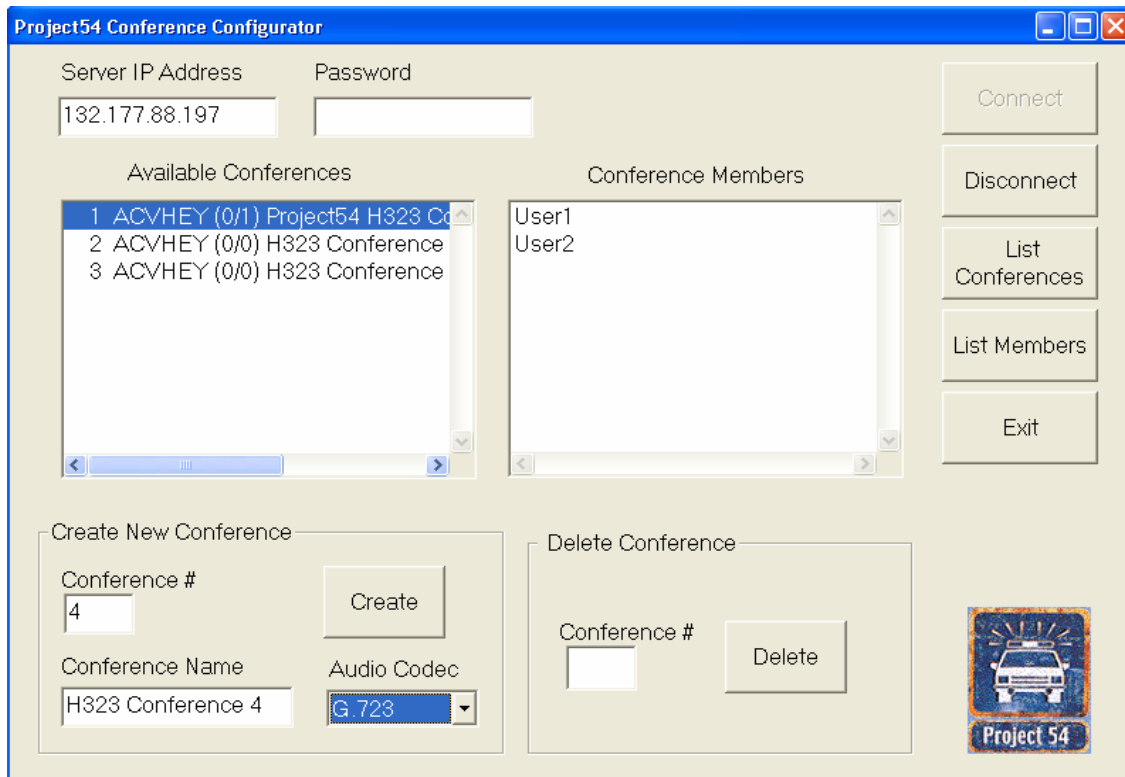
Figure 5.5 Flowchart of Conference Configuration Application

When the Configuration App is started, it waits for the user to enter an IP address and password and choose to connect to a conference server. If the IP address is that of a conference server, a TCP connection is established. If the IP address is not valid, the user is informed that the connection failed. Once connected the server verifies the password entered by the user. If the password is invalid, Configuration App is notified by the server. The Configuration App then displays a message to the user indicating the password was incorrect, and disconnects from the server. If the password is valid, the user is granted access to the server.

Once access has been granted, the Configuration App queries the server for a list of currently available conferences. Once received, the available conferences are then displayed. The user can then choose to create a new conference or delete one of the current conferences. At any time the user can also choose to refresh the list of current conferences located on the server.

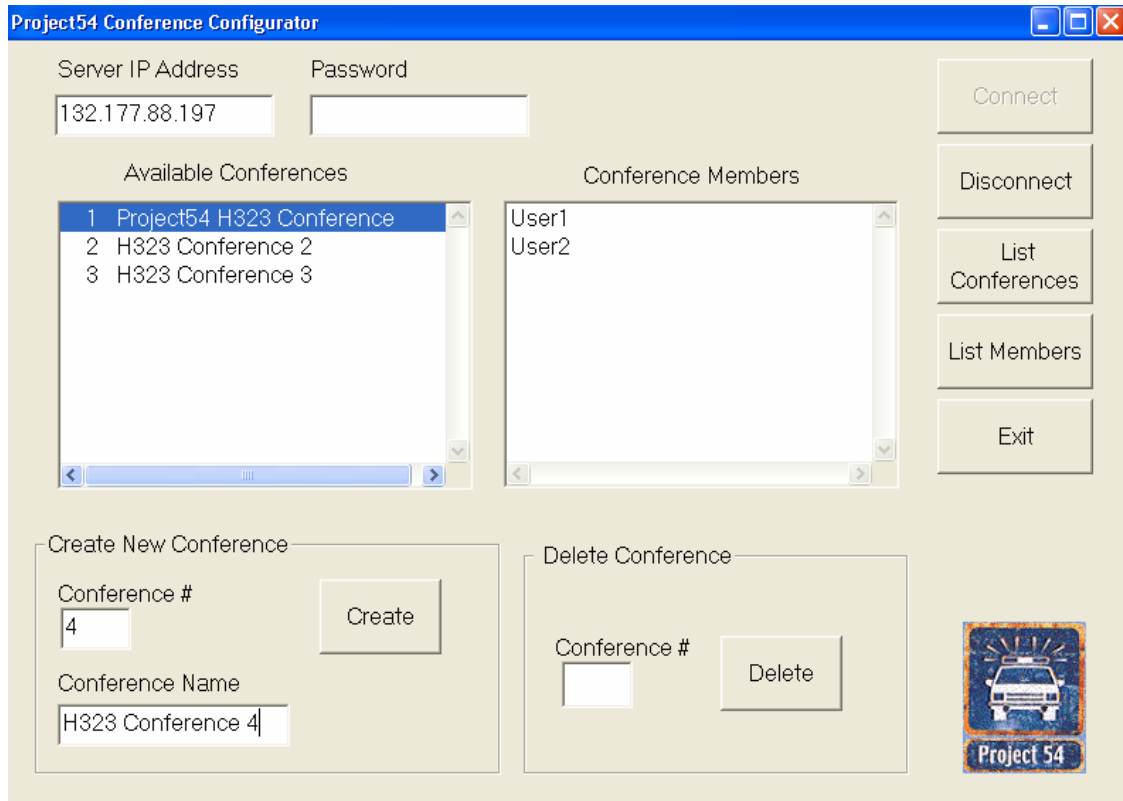
### **5.3.2 Graphical User Interface**

A screen shot of the full featured version of the CCA graphical user interface (GUI) is shown in Figure 5.6 and the limited featured version is shown in Figure 5.7.



**Figure 5.6 Screenshot of Full Featured Configuration App Graphical User Interface**





**Figure 5.7 Screenshot of Configuration App Graphical User Interface with Limited Features**

There are two versions of the GUI because most users will not require all of the information and features of the full-featured version. The full-featured version will be used by system administrators when additional information or special conference configurations are desired. This version provides more information about the configuration of each conference in the “Available Conferences” window and the ability to change the audio codec a created conference will use. The letters preceding the conference name indicate conference attributes that are enabled. Table 5.3 shows the possible conference attributes. Following the conference attributes is an indication of the number of video senders and lurkers (user who has no camera or is choosing not to send a video stream) in the following form: (senders/lurkers). Currently, the video features of

the conference server are not used in this system. This causes the number of video senders to always be displayed as zero and the number of video lurkers is the number of conference members. The option is also available to set the audio codec a conference will use once created. The default codec is G.723 which is a low bit rate codec and should be suitable for most applications.

To make a connection with a conference server the user must first enter a valid IP address and password for a particular server. The connection is then made by pressing the “Connect” button. If the connection could not be made, a message is displayed indicating the error which prevented the connection. When a successful connection is made, a list of conferences currently on the server is displayed in the “Available Conferences on Server” field. The user then may perform one of five actions: create a new conference, delete a conference currently hosted by the server, refresh the list of available conferences, refresh the list of members for a particular conference or disconnect from the server. To create a conference, a name must be entered into the “Conference Name” field and a unique conference number between 0 and 65,525 must be entered into the “Conference #” field. The previous “Conference Name” and “Conference #” fields are contained within the “Create New Conference” area of the application. The conference is then created by pressing the “Create” button. If creation was successful, the list of available conferences will be updated to include the newly created conference. To delete a conference, enter its number in the “Conference #” field within the “Delete Conference” area of the application. When the “Delete” button is pressed, the conference will be removed from the server and the list of available conferences will be updated. At any time the list of available conferences may be refreshed by the user by pressing the

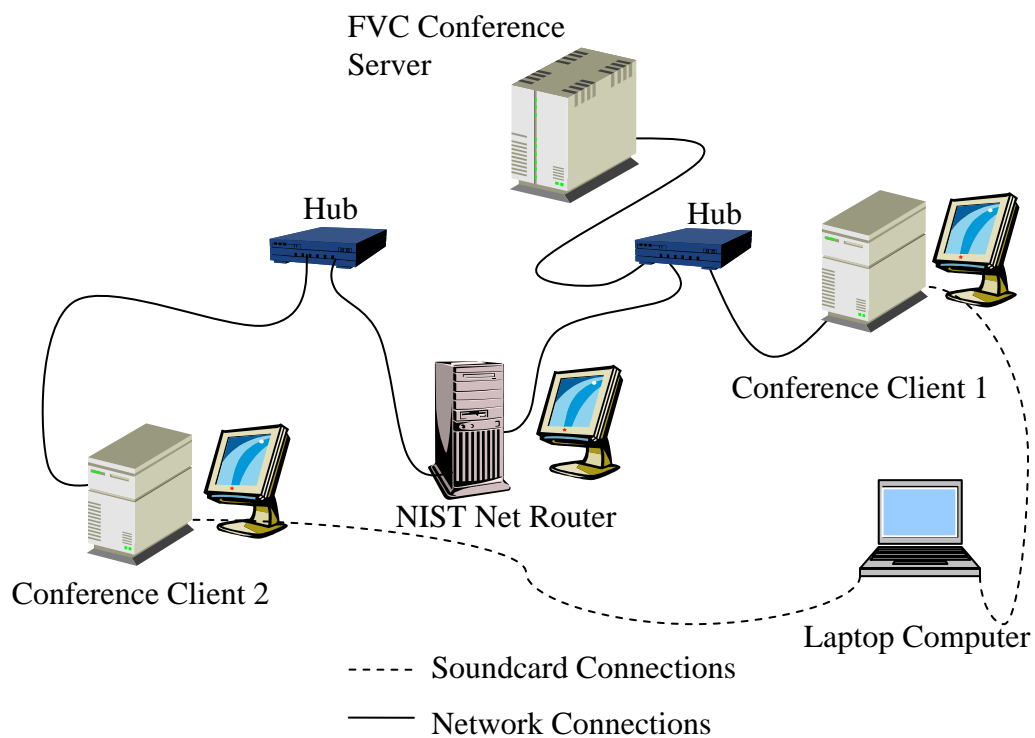
“List Conferences” button. To list the members for a particular conference, the desired conference must first be selected in the “Available Conferences” field. The “List Members” button is then pressed to list current members of the selected conference. To disconnect from the server, the “Disconnect” button is pressed.

## CHAPTER 6

### SYSTEM TESTING

#### 6.1 System Level Description

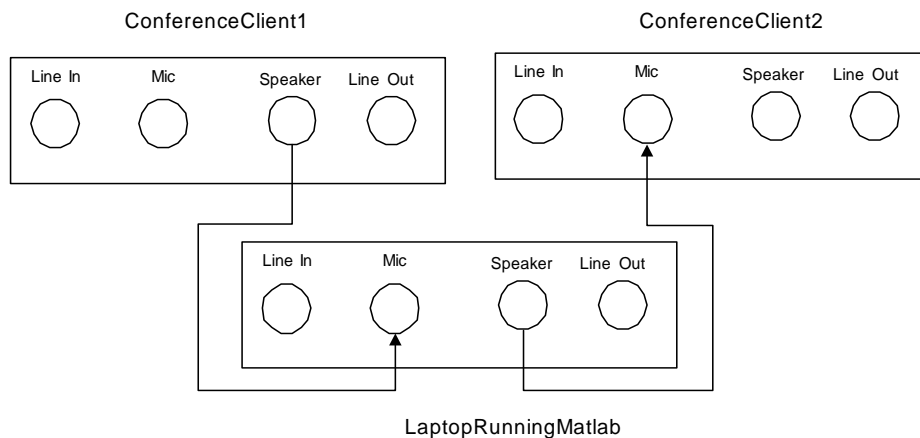
To determine the performance of the system under varying levels of QoS an automated testing system was created. This system provides a controlled environment where network impairments, similar to those encountered on the Internet, could be emulated. A diagram of the test setup is shown in Figure 6.1.



**Figure 6.1 Diagram of Test System**

In the test setup, two conference clients were connected across two different network subnets. A laptop computer was used to output speech samples to Client 2. Client 2 then transmitted this speech to the conference server where it was redistributed to Client 1. The laptop computer was then used to record degraded speech samples received by Client 1.

The clients, server, and router were implemented on desktop computers. Clients 1 and 2 used Microsoft's Windows 2000 for an operating system and the CCA for participating in conferences. The server used Windows 2000 Advanced Server for an operating system and the FVC Conference Server to host voice conferences. To route between the two different subnets of the clients, the National Institute of Standards and Technology Network Emulation Tool (NIST Net) [11] router was used. The laptop computer used MATLAB to output voice to Client 2 and record the voice from Client 1. The connections between the laptop soundcard and client soundcards are shown in Figure 6.2.

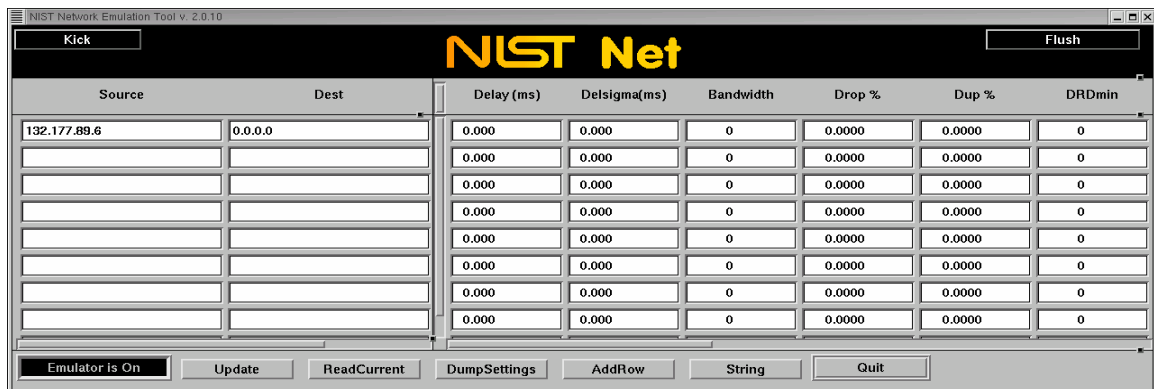


**Figure 6.2 Connections between Test System Soundcards**

## **6.2 NIST Net Router**

The NIST Net router uses Linux as an operating system with the IP forwarding (to enable routing) feature activated. The “Routed” daemon is not active because there are no other routers in this system and Routing Information Protocol (RIP) is used for routing. The router also has the NIST Net extension added to the Linux kernel.

NIST Net is a general-purpose tool for emulating impairments in IP networks [17]. It is designed to allow controlled, reproducible experiments with network performance sensitive applications. NIST Net is implemented as a kernel module extension to the Linux operating system and is configured through a set of commands or an X Window based GUI. A screen shot of the NIST Net GUI is shown in Figure 6.3.



**Figure 6.3 Screenshot of NIST GUI**

The tool can be used to emulate numerous network impairments, including: variable packet delay distributions, congestion and packet loss, and bandwidth limitation. The X Windows interface allows the user to select and monitor specific traffic streams passing through the router and apply selected impairments to those streams.

To correctly emulate the way packets are typically dropped by routers, NIST Net utilizes the Derivative Random Drop (DRD) algorithm which is a variation of the Random Early Detection (RED) algorithm. Both RED and DRD are proactive packet dropping techniques for congestion avoidance. The RED algorithm drops packets randomly from its buffers if the queue length exceeds a certain threshold. In the case of DRD, queue utilization (ratio of service rate to arrival rate) is used to determine when packet drops will occur [18]. DRD is not typically used in routers because an instantaneous burst in traffic could lead to a large number of near simultaneous drops and hence the correlation of restarts from multiple TCP connections. These correlated drops are avoided in NIST Net because each source/destination pair has its own queue which is independent of the others. The problem of simultaneous drops is not present with the RED algorithm, but it is more difficult to implement, which is the reason it is not used in the NIST Net emulator.

### **6.3 Test Setup**

Two scripts are used to automate the testing process. To automate the playing and recording of audio, a MATLAB script is used on the laptop computer in the test setup. To automate the changing of network impairment settings on the NIST Net router, a Linux shell script is used. The MATLAB script shown in APPENDIX B is used on a laptop computer to automate the process of playing and recording voice. The script plays a pre-recorded voice file into Client 2 then immediately begins recording the audio received and output by Client 1. The audio output by Client 1 is saved into a wav file with a filename indicating the network impairments that were present at the time of recording. The file name format is as follows: "DegradedVoice\_Bandwidth\_Drop

Percentage\_Level of Jitter”. For example, if the voice was recorded across a network where the bandwidth was 3,600 bytes/sec, 5 percent of sent packets were dropped, and there was 20 milliseconds of jitter, the filename would be “DegradedVoice\_3600\_5\_20.wav”. To determine the values of bandwidth, drops, and jitter to use for the filenames, three text files are used and must exist before execution of the MATLAB script begins. The three text files are: “Bandwidth.txt”, “Drops.txt”, and “Jitter.txt”. There should be one column of numbers representing the respective network impairments settings in each file. The values in these three text files should have the units shown in Table 6.1. After the recording is saved to a wav file, the script pauses execution while network impairment settings on the NIST Net emulator are updated for the next iteration of testing. After a predetermined timeout, the script resumes execution, and loops back to repeat the playing and recording of voice. The names to be used for the recorded files are updated to reflect the current network impairment settings. Execution of the script continues until the values from all three text files have been used.

Filename	Units of Values in File
Bandwidth.txt	Bytes/Sec
Drops.txt	Percentage (%)
Jitter.txt	Milliseconds (ms)

**Table 6.1 Units for Values in Text Files**

During testing, text commands for the NIST Net emulator are used instead of the NIST Net GUI. The Linux shell script titled “VQ\_Testing”, shown in APPENDIX C,



is used to automate the process of issuing commands to the emulator. The start of execution of the shell script is synchronized with the MATLAB script from section 5.3, which ensures network impairment settings do not change while recording is in progress. The values of network impairments to be used are specified within the shell script.

#### **6.4 Source and Recorded Files**

One voice recording from each of three different male speakers was obtained using the Windows Sound Recorder program. Each voice file contains a phrase which is a mix of speech and silence similar to a brief portion of normal conversation. The recordings were originally various lengths but each was reduced to 8.192 seconds which is approximately the length suggested in section 7 of ITU-T recommendation 8.30 [19] and sampled at 8 kHz for a total of 65,536 samples. The length of the voice file was chosen such that the number of samples was a power of 2. Processing at a later point utilizes the Fast Fourier Transform (FFT), which operates more efficiently on data whose length is a power of 2.

The length of the recorded voice file is 20 seconds to compensate for delays in the playing/recording process and in network transmission. Recording for 20 seconds ensures the entire 8.192 seconds of desired voice is captured.

#### **6.5 Test Scenario**

The recorded phrase from one speaker is played at Client 2, travels over the network, and is recorded at Client 1. This process was repeated once for each value of each network impairment listed in APPENDIX A. This experiment is repeated five times for the recorded phrase from each speaker.

#### **6.6 Evaluating Voice Quality using PESQ**

The Perceptual Evaluation of Speech Quality (PESQ) method is used to evaluate the voice quality of the recorded samples. PESQ is an objective testing method which attempts to determine the relative quality of a recorded voice file when compared

with a source voice file by modeling the way humans perceive sounds [20]. PESQ is suitable for measuring quality when the source waveform has been distorted by the use of non-linear audio codecs, packet drops, and variations in delay (jitter). Other objective testing methods such as BSD, PSQM, and MNB have been found to produce inaccurate results and therefore are not useful in measuring quality when those distortions are present. PESQ has been found to produce inaccurate results and is not suitable for determining the effects of: talker echo, listening levels, or conversational delay. It is not desired to evaluate the system under these conditions; therefore PESQ should yield useful results.

A program written in C was developed to automate the process of applying the PESQ algorithm to each degraded voice file. The PESQ algorithm is actually implemented as a separate executable file and is accessed using the C function “\_popen”. The same three text files as mentioned in section 5.3 are used to determine the voice file names to analyze. The PESQ algorithm takes the original voice file and each of the recorded files as arguments and outputs a voice quality score between -0.5 and 4.5, with -0.5 being the worst and 4.5 being the best. Typically, PESQ scores fall in the MOS range of 1 to 4.5 except in extremely noisy environments where the score will fall below 1. The scores are then saved to three Excel spreadsheets. Each spreadsheet contains the scores obtained while a different network impairment was varied, with the filename of the spreadsheet indicating the varied parameter (ex. bandwidth.xls, drops.xls, and jitter.xls).

### **6.7 Testing for Delay**

The test system was not used to evaluate the effects delay will have on this system. Delay does not have a direct effect on voice quality, but affects conversation

flow; therefore testing voice quality using the test system and previously described testing procedure would not yield useful results. To determine if excessive one-way delay will negatively affect this system when implemented across the Internet, round-trip delay times to various towns surrounding Durham, NH were measured. The towns selected for this test were: Nashua, Keene, New London, and Rindge. All four towns are located in the state of New Hampshire. Within each town, the IP address of a local college was used for testing. Table 6.2 shows the locations, colleges, and corresponding IP addresses used for this test. Each college is physically located in the town indicated in Table 6.2, with the exception of New Hampshire Community Technical College which is located in Claremont, New Hampshire, but has its website hosted in Nashua, New Hampshire. To measure delay times, Internet Control Message Protocol (ICMP) messages were sent using the “ping” command. The ICMP messages carried 100 data bytes, which is similar to the amount of data bytes in packets carrying voice. Fifty of these ICMP messages were sent to each destination and the maximum round-trip time was noted. The fifty ICMP messages were sent every twenty minutes over a 24-hour period. Three different 24-hour periods were used for each site. Delay was measured over a 24-hour period because traffic and delay over the Internet will vary throughout the day.

Site Name	Location	IP Address
New Hampshire Community Technical College	Nashua	192.233.239.56
Keene State College	Keene, NH	158.65.8.164
Colby-Sawyer College	New London, NH	206.34.181.149
Franklin Pierce College	Rindge, NH	64.78.52.243

**Table 6.2 Locations and IP Addresses Used in Delay Tests**

### **6.8 PC/Radio Interface**

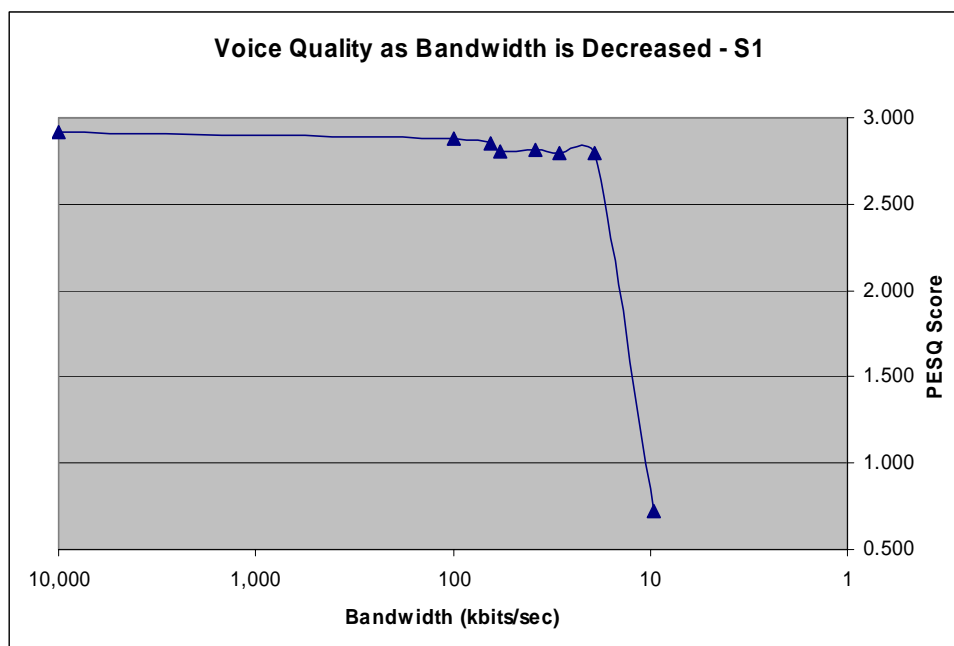
To test the functionality of the PC/Radio interface it was connected between a PC and a mobile radio. A second radio was used to listen to transmissions sent from the radio that was connected to the interface. Audio was then played from the PC into the interface and listening tests were performed to determine if the interface was functioning properly.

## CHAPTER 7

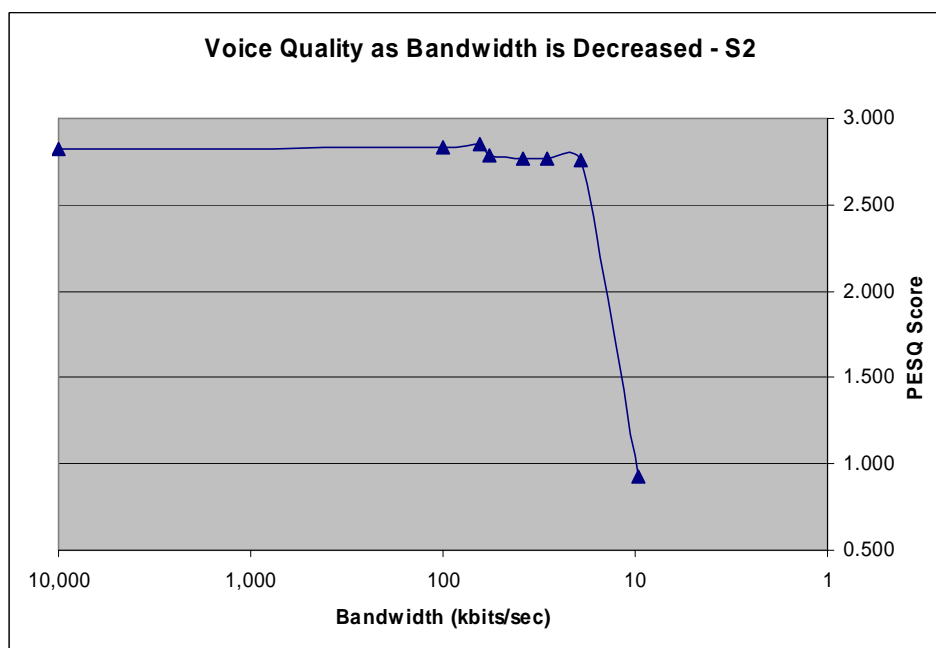
### RESULTS AND DISCUSSION

#### 7.1 Results of Bandwidth Limitation

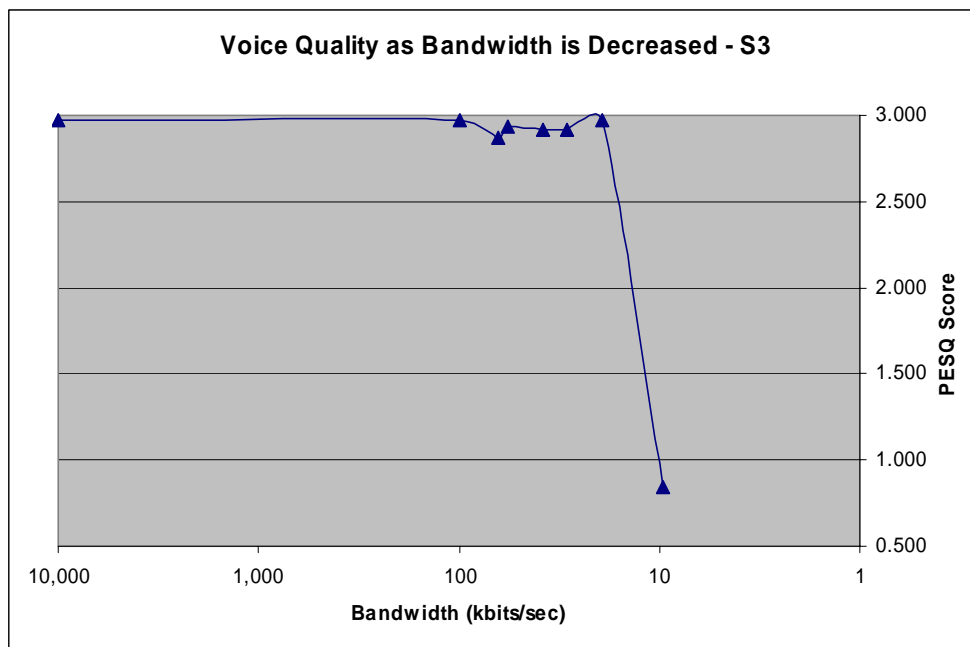
Bandwidth limitation was the first network impairment used to evaluate voice quality against. The bandwidths used in this test were: 10,000, 100, 64, 57.6, 38.4, 28.8, 19.2, and 9.6 kbit/sec. The audio codec used for this and all remaining tests is G.723, which has a nominal bitrate of 6.4 kbit/sec. The notation of S1, S2, and S3 used in this thesis is used to indicate speakers 1 through 3 respectively. The PESQ scores resulting from each of the five testing iterations for S1, S2, and S3 were averaged together. A PESQ score of 3.0 is the best possible score and indicates excellent voice quality. A score between 2.0 and 3.0 indicates good voice quality with some noticeable degradation. A score between 2.0 and 1.0 indicates the voice quality is significantly degraded and a listener may encounter difficulty in recognizing some words. A score below 1.0 indicates voice quality which is degraded to a point where a listener will have difficulty recognizing all words. The results for S1, S2, and S3 are shown in Figure 7.1, Figure 7.2, and Figure 7.3 respectively.



**Figure 7.1 Average PESQ Scores for Varied Bandwidth - S1**



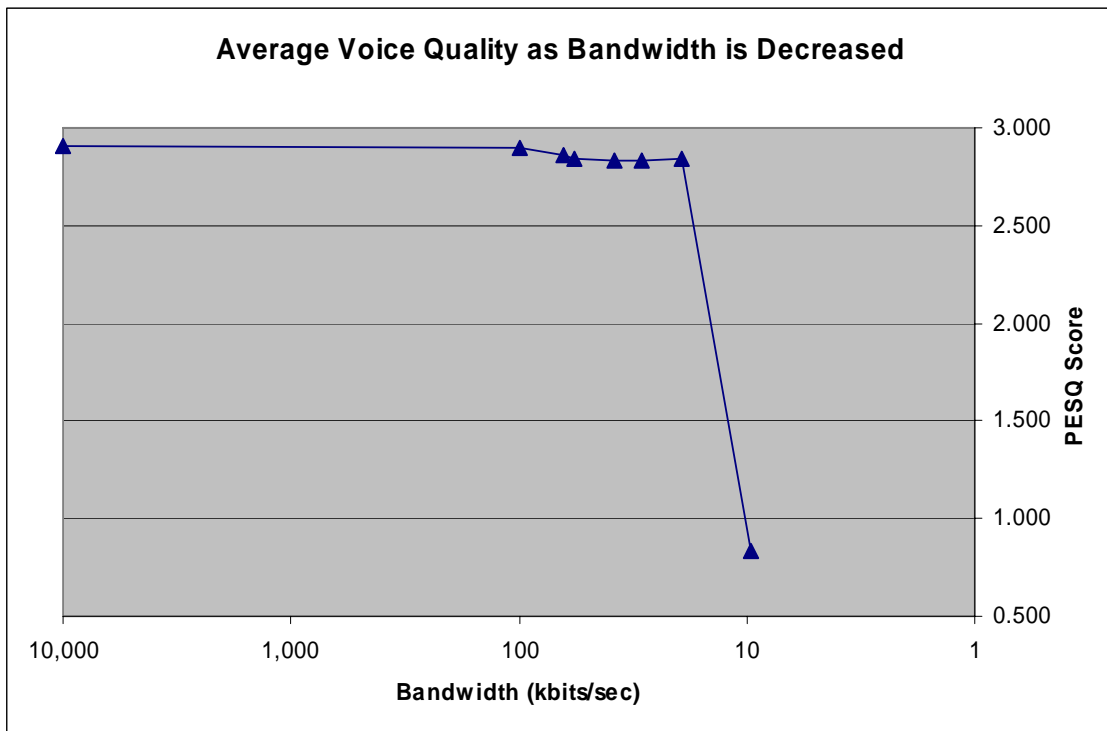
**Figure 7.2 Average PESQ Scores for Varied Bandwidth - S2**



**Figure 7.3 Average PESQ Scores for Varied Bandwidth - S3**



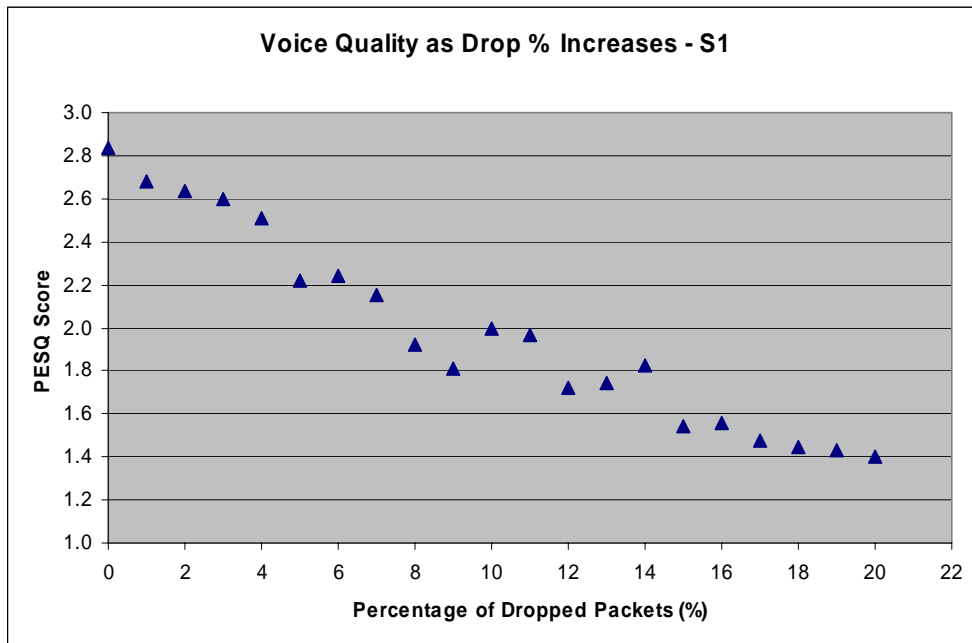
The PESQ scores from S1, S2, and S3 were averaged together to show the general trend in voice quality as bandwidth is decreased. The average of the PESQ scores is shown in Figure 7.4.



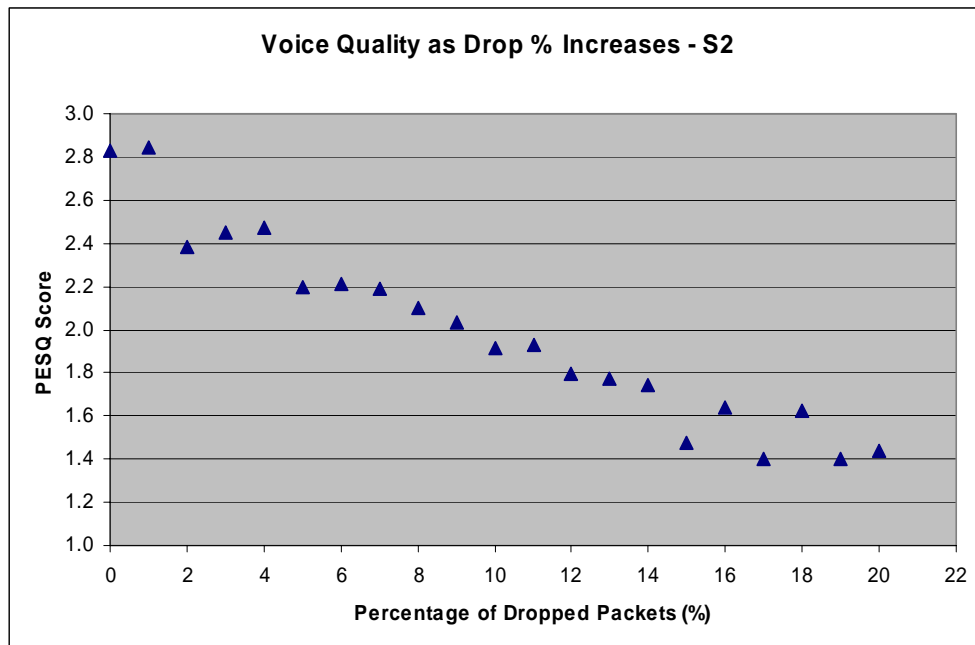
**Figure 7.4 Average PESQ Scores as Bandwidth is Decreased – S1, S2, S3**

## **7.2 Results of Increasing Percentage of Dropped Packets**

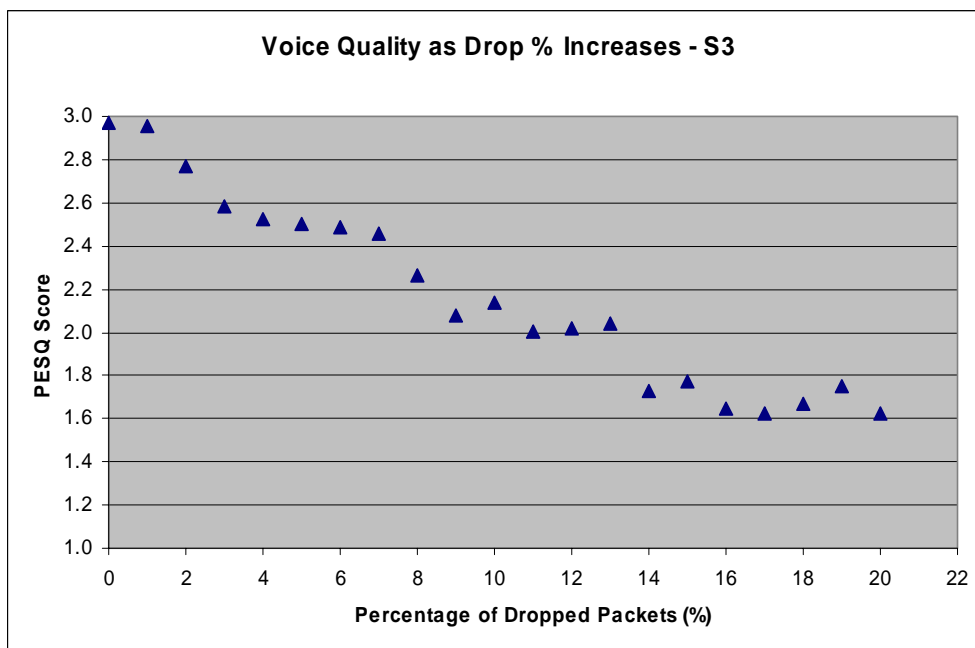
The second network impairment to evaluate voice quality against was the percentage of dropped packets. The percentage of dropped packets was varied between 0 and 20 percent. The PESQ scores resulting from each of the five testing iterations for S1, S2, and S3 were averaged together. The results for S1, S2, and S3 are shown in Figure 7.5, Figure 7.6, and Figure 7.7 respectively.



**Figure 7.5 Average PESQ Scores for Increased Drop Percentage - S1**

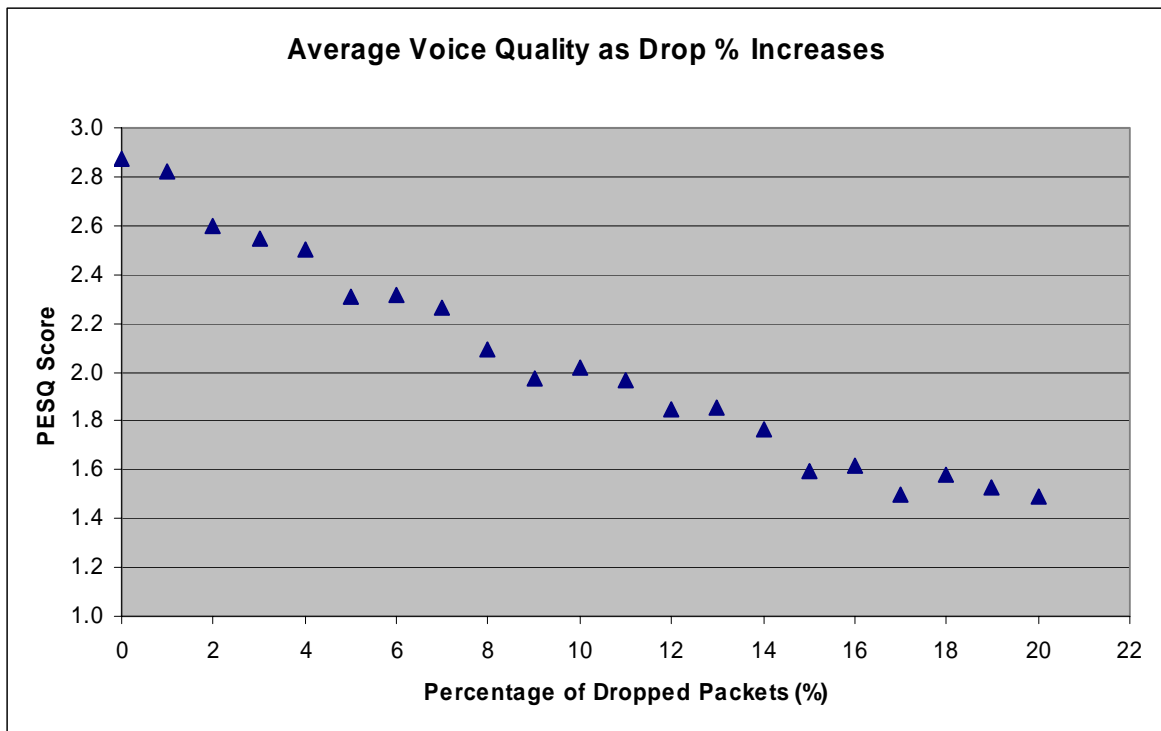


**Figure 7.6 Average PESQ Scores for Increased Drop Percentage - S2**



**Figure 7.7 Average PESQ Scores for Increased Drop Percentage - S3**

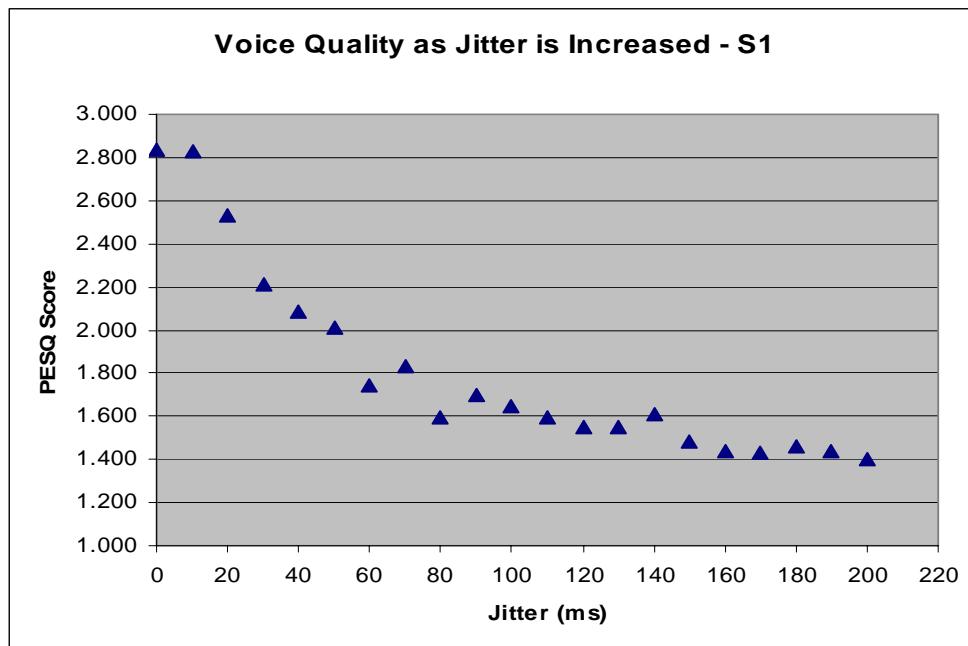
The PESQ scores from S1, S2, and S3 were averaged together to show the general trend in voice quality as the percentage of dropped packets is increased. The average of the PESQ scores is shown in Figure 7.8.



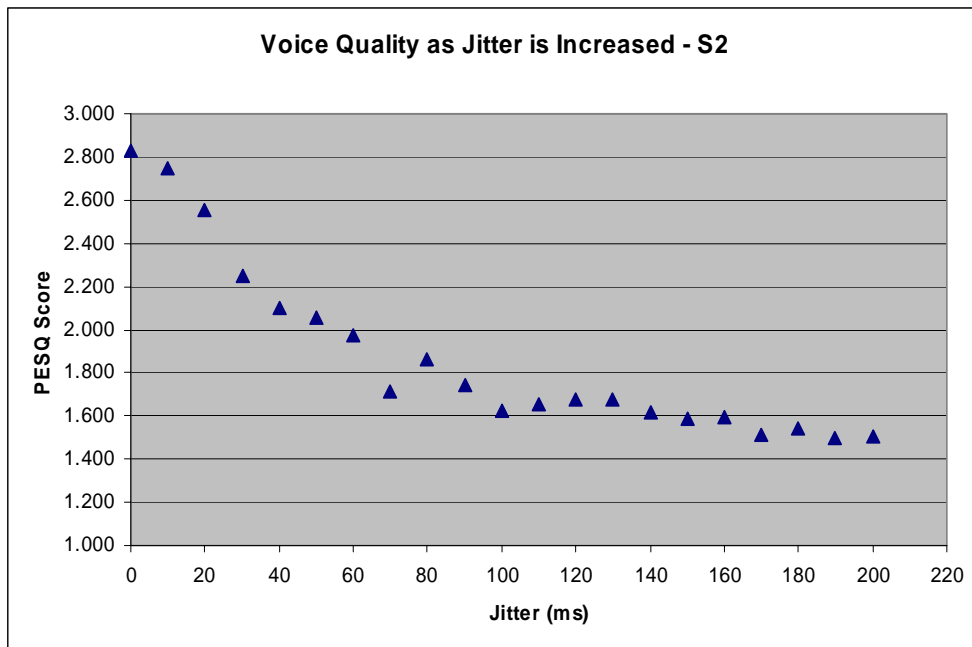
**Figure 7.8 Average PESQ Scores as Drop Percentage is Increased – S1, S2, S3**

### **7.3 Results of Increasing Level of Jitter**

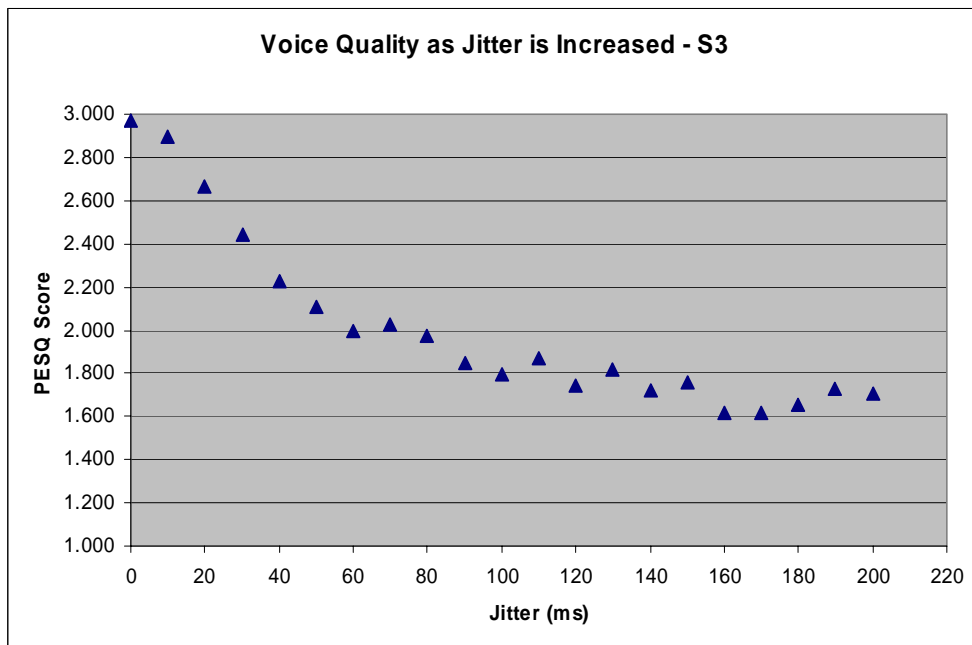
The third network impairment to evaluate voice quality against was the level of variable delay through the network or jitter. A jitter buffer is used at each client to remove a certain amount of jitter by holding packets for a limited period. This buffer can only remove a limited amount jitter because the longer packets are held, the more delay is introduced, which is undesirable. As the level of jitter increases beyond the buffer size, the effects of the jitter are noticed and voice quality begins to degrade. The level of jitter was varied between 0 and 200 ms in 10 ms increments. The PESQ scores resulting from each of the five testing iterations for S1, S2, and S3 were averaged together. The results for S1, S2, and S3 are shown in Figure 7.9, Figure 7.10, and Figure 7.11 respectively.



**Figure 7.9 Average PESQ Scores for Increasing Jitter - S1**

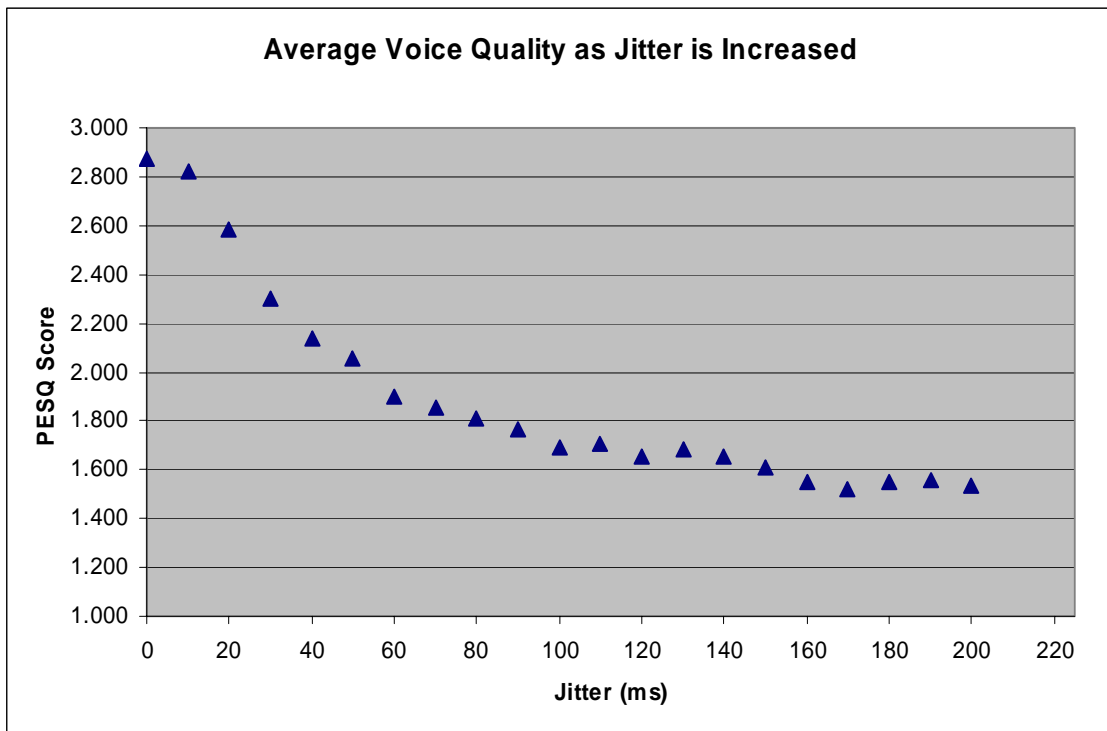


**Figure 7.10 Average PESQ Scores for Increasing Jitter - S2**



**Figure 7.11 Average PESQ Scores for Increasing Jitter - S3**

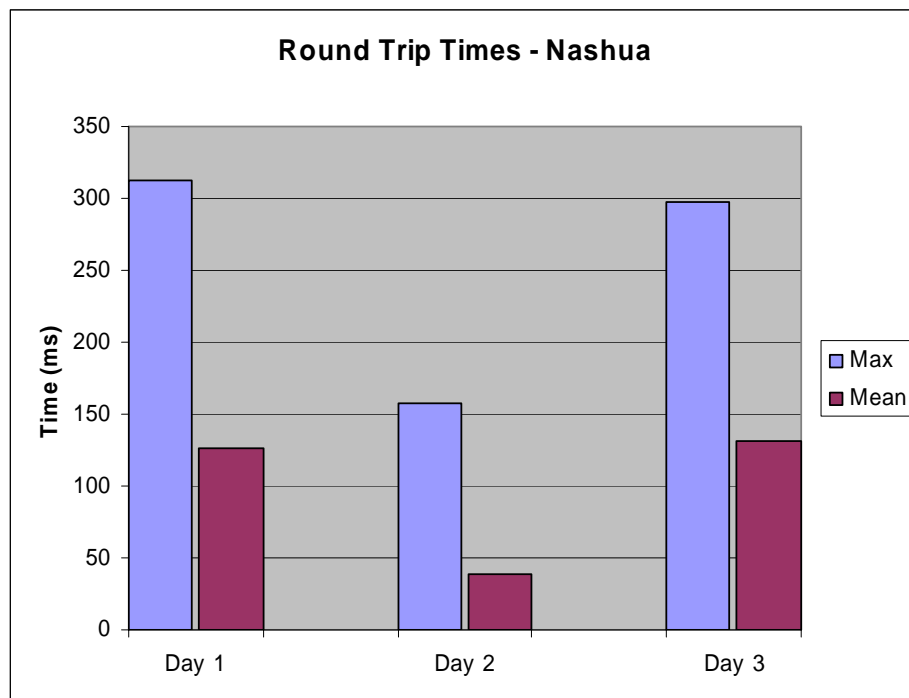
The PESQ scores from S1, S2, and S3 were averaged together to show the general trend in voice quality as the level of jitter is increased. The average of the PESQ scores is shown in Figure 7.12.



**Figure 7.12 Average PESQ Scores as Level of Jitter is Increased – S1, S2, S3**

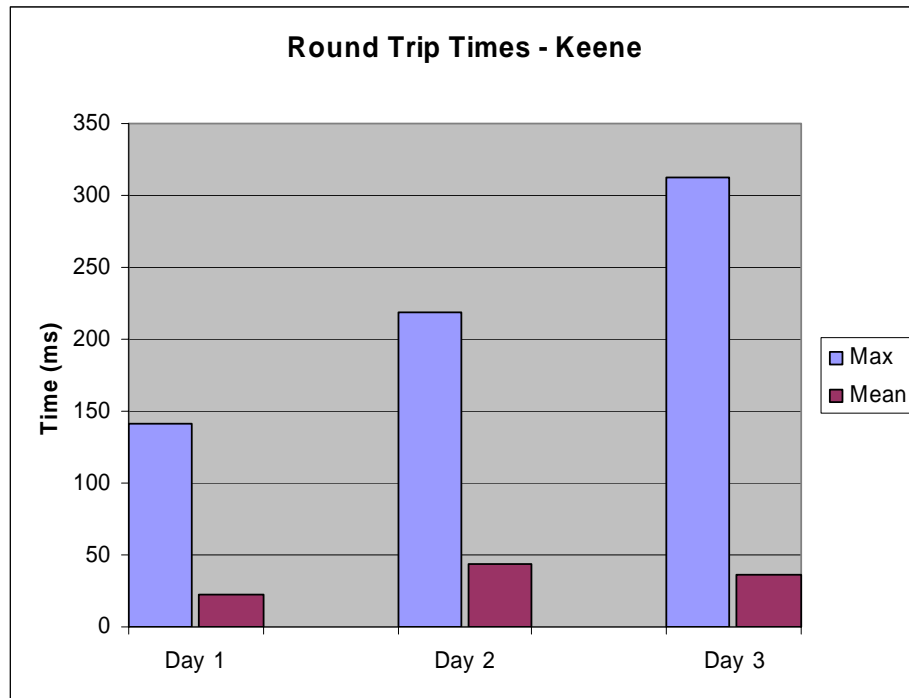
### **7.3 Results of Delay Testing**

The results of delay testing for Nashua, Keene, New London, and Rindge are shown in Figure 7.13, Figure 7.14, Figure 7.15, and Figure 7.16 respectively. The delay tests yielded 50 round-trip times every twenty minutes each day. The maximum of these 50 round-trip times was recorded which produced 72 values each day. Figures 6.13 through 6.16 show the maximum and mean of these 72 values for each day.

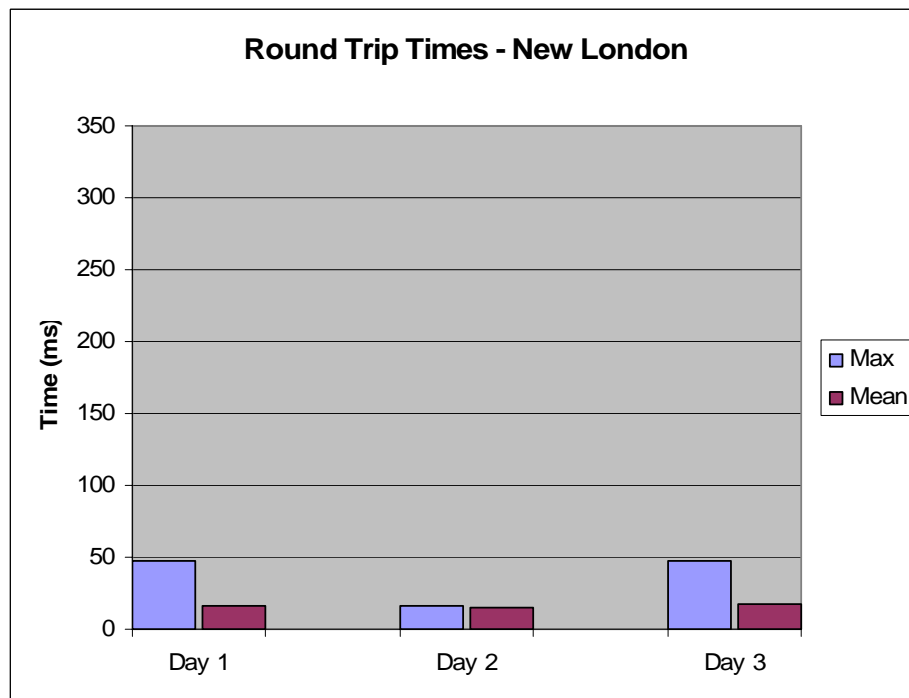


**Figure 7.13 Round Trip Times - Nashua**





**Figure 7.14 Round Trip Times – Keene**



**Figure 7.15 Round Trip Times - New London**

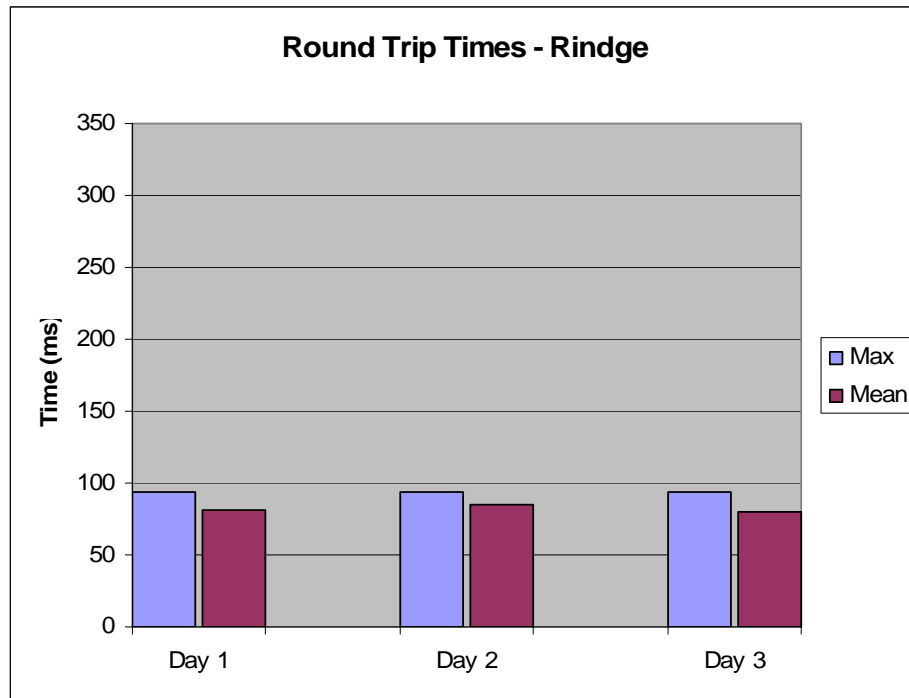


Figure 7.16 Round Trip Times - Rindge

#### **7.4 Results of PC/Radio Interface Testing**

Listening tests were performed to determine if the PC/Radio interface was functioning properly. These listening tests determined that the interface introduced a noticeable amount of clipping on the first word of an utterance. Multiple tests using different phrases yielded the same results of the first part of the first word of an utterance being lost. Measurements were also taken to determine the exact delay through the interface. The delay was measured as the time it took the output to transition from logic high ( $\sim 3.8\text{V}$ ) to logic low (below  $200\text{mV}$ ) after the threshold voltage of  $150\text{ mV}_{rms}$  had been reached on the input. Table 7.1 shows the response time of the interface versus input RMS voltages.

Input Voltage <sub>rms</sub> (mV)	Response Time (ms)
900	18.82
800	18.72
700	18.18
600	20.61
500	19.81
400	23.64
300	47.64
200	84.55

**Table 7.1 Response Time Versus Input Voltage for PC/Radio Interface**

### **7.5 Discussion of Results**

Judging by the PESQ scores obtained, it may appear as though this system will not be able to deliver adequate voice quality when implemented across the Internet. The highest PESQ score the system was able to achieve was 3.0 out of a maximum score of 4.5 with no network impairments present and the score decreased as network impairments were introduced. This was unexpected because before the actual testing system was used, basic conversation was attempted across a conference hosted by the server. During this conversation, voice quality seemed acceptable and both users stated the voice sounded clear. The low levels of voice quality must therefore be attributed to other factors besides poor system performance. One such factor is the cabling used to connect the soundcards in the test system. Electrical noise, including noise from the laptop power supply, from the surrounding environment was picked up by these cables. The laptop power supply actually introduced so much noise that it was not connected during testing. Because the effects of the cables could not be removed, in the evaluation of this system, 3.0 will be considered the best possible score and will represent an

undistorted signal. This is acceptable because the goal of this testing was to determine the degradation of voice quality due to network errors, not overall voice quality. The mapping between the results of this testing, PESQ scores, and actual subjective MOS is shown in Table 7.2.

Quality of Speech	Test Result	PESQ Score	MOS Score
Excellent	3	4.5	5
Good	2	3.5	4
Fair	2	2.5	3
Poor	0	1.5	2
Bad	-0.5	0.5	1

**Table 7.2 Mapping Between Test Scores, PESQ Scores and MOS**

When observing Figures Figure 7.1 through Figure 7.12 it appears at certain points that as the particular network impairment is increased, the PESQ score improves. At these points it seems as though voice quality improves slightly as network performance is degraded, which is counter-intuitive. This can be explained by again noting that correlation between PESQ scores and subjective MOS has an average value of 0.935 and in some cases has been found to be as low as 0.81. What this means is that the actual voice quality and the PESQ score reported may differ by an average of 6.5 % and a maximum of 19%. Since PESQ results can be inaccurate by as much as 19 %, they should not be taken literally but used more as a rough guide for determining if voice quality will be acceptable.

Due to the system architecture, the round-trip times obtained from delay testing can be directly evaluated against acceptable one-way trip times to maintain conversation previously discussed in this thesis (section 2.4.1). Table 2.1 shows the maximum

allowable delay to maintain normal conversation to be 400 ms. The results shown in Figures 6.13 through 6.16 indicate the system will perform robustly with regard to delay when implemented across the Internet because all delay times are significantly less than 400 ms.

The 400 ms limit for delay is accurate when attempting to have a conversation but many radio communications between emergency service personnel is not of the form of a regular conversation. Their communication is usually of the form of short messages to relay information often ending in a specific indication that the current speaker is finished. Therefore, having the ability to have a smooth flow of conversation is of little importance and it is allowable for delays to be longer than 400 ms. This lower importance of conversation flow is prevalent in the APCO Project 25 standard for digital radios. Project 25 compliant radios transmit at 9600 baud or 1200 bytes/second and use 512 byte data packets for carrying voice information. Before a transmission occurs, a full data packet must be collected which equates to a 430 ms delay in transmission. The receiver must also receive an entire packet before decoding can take place, which adds an additional 430 ms delay for a total of 860 ms of delay. This delay cannot be avoided and does not take into account transmission time. An actual transmission between radios will always be delayed longer than 860 ms due to propagation delays.

While testing the PC/Radio interface circuit, noticeable clipping of the first word of an utterance was observed. With the current hardware solution for the interface, there is little that can be done to eliminate this clipping because delays introduced by the analog electronics are unavoidable. In the future it must be determined whether or not this clipping will cause a great disturbance to users of the system.

## **CHAPTER 8**

### **CONCLUSION**

A system utilizing VoIP network communication was desired to solve the problem of mobile radio interoperability. The main goal of this thesis was to design, implement, and test a system to route mobile radio communications between non-interoperable mobile radios over IP based networks. A system that connects non-interoperable radio networks was designed using both off-the-shelf and custom hardware and software. Conferences are hosted using First Virtual Communication's software based MCU and follow the H.323 standard for packet-based network audio, video, and data communications. The FVC software based MCU may run on either a desktop PC or a server class computer, which would be desirable for optimum performance. Custom software developed for the system includes an H.323 compliant application to participate in conferences and an application to perform conference management across the Internet. Since the conferences follow the H.323 standard it is possible to use any applications that also follow the standard, but it is suggested that the custom applications be utilized because of their ease of use. The applications to join the conference will run on desktop PCs located at the headquarters of each department. To interface the desktop PCs with the radio system, a hardware interface has been designed and prototyped.

Extensive testing of this system has been performed to ensure the level of performance will be satisfactory when implemented over the Internet. IP networks such as the Internet do not offer QoS guarantees, therefore the system's performance must

remain consistent when network errors are encountered. A test network was set up which consisted of a FVC Conference Server, H.323 endpoints, and a NIST Net router. This test network allowed the introduction of network impairments (jitter, packet loss, and bandwidth limitation) in a controlled setting. When the network impairments were introduced, the Project54 VoIP system's performance remained at an acceptable level. Other testing was performed to determine if transmission delays across the Internet would be so large that they would negatively affect the system's performance. It was found that transmission delays would be small enough so that the system's performance would remain acceptable.

The Project54 VoIP system has been shown to exhibit robust performance when faced with typical network impairments encountered on the Internet. It offers a standards-based cost-effective solution that will make communication between incompatible mobile radios possible.

## **CHAPTER 9**

### **SUGGESTIONS FOR FUTURE ENHANCEMENTS**

This section lists several ideas for improving the system developed for this thesis.

First, the PC/Radio interface circuit must be investigated further. The major issue with the current design is the clipping of the first word of an utterance. It must be determined if this clipping causes the loss of too much information and disrupts the users of the system. If the clipping caused by the current design is found to be too disruptive, a new solution must be found. If the current design's performance is acceptable, it should be redesigned with manufacturability in mind, as the current design is only a prototype. One alternative solution would be to implement a Push-To-Talk system in software. To accomplish this, the audio stream that is about to be sent to the system's speakers could be captured and buffered. The audio stream could be captured and buffered using a virtual sound card driver which calls the real driver when the audio stream is to be output. While the audio is being buffered, the PTT signal could be activated. Once the PTT signal has been activated, the buffer would then release the audio stream to be output.

Second, voice codecs are continually being improved to allow the transmission of voice using lower bit-rates and better performance under error conditions. Currently, there are codecs that use lower bit-rates than those used by the Project54 VoIP system, but these codecs do not offer the level of voice quality desired. In the future lower bit-rate



codecs will improve in quality and when they do, should be used in the system as a lower bit-rate will make the system more robust.

Finally, before the Project54 VoIP system is relied upon, further testing should be performed. It would be desirable to perform the tests described in this thesis with more conference clients, as this would better emulate real-world conditions. Further testing should also be performed after the system is implemented across the Internet. Because of the unpredictable nature of the Internet it is possible that network errors may occur that were not present in the simulated environment. The same testing procedures should be performed, without introducing network impairments, using more iterations to be sure the system is tested against a variety of network conditions.

## REFERENCES

- [1] A. L. Kun, W. T. Miller, W. H. Lenharth, "Modular System Architecture for Electronic Device Integration in Police Cruisers," *Proceedings of the 2002 IEEE Intelligent Vehicle Symposium*, Versailles, France, June 18-20, 2002
- [2] "Packet Based Multimedia Communications Systems," ITU-T Recommendation H.323, November 2000
- [3] "IP Telephony with TAPI 3.0," MSDN Library, April 2000
- [4] "SIP : Session Initiated Protocol," IETF Request For Comments 3261, June 2002
- [5] "RTP : A Transport Protocol for Real-Time Applications," IETF Request For Comments 1889, January 1996
- [6] JPS Communications, JPS Communications Homepage, <http://www.jps.com/index.html> [Accessed: 3 February 2003].
- [7] Catalyst Communications, Catalyst Communications Homepage, <http://www.catcomtec.com> [Accessed: 2 April 2003].
- [8] S. Pracht, D. Hardman, "Voice Quality in Converging Telephony and IP Networks," Agilent Technologies Whitepaper, August 2002, pp. 11
- [9] "One-way Transmission Time," ITU-T Recommendation G.114, May 2000
- [10] R.J.B. Reynolds, A.W. Rix, "Quality VoIP – an engineering challenge," *BT Technology Journal*, April 2001, pp. 24
- [11] S. Yamarchy, "Subjective and Objective Evaluation of Voice Quality Over Internet Telephony," (Master's Project), U. of New Hampshire, Durham, 2000
- [12] "Methods for subjective determination of transmission quality," ITU-T Recommendation P.800, August 1996
- [13] "Objective Quality Measurement of Telephone Band (300 – 3400 Hz) Speech Codecs," ITU-T Recommendation P.861, February 1998
- [14] Dale Rogerson, "Inside COM", Microsoft Press, Redmond, WA, 1997
- [15] "Description of Svchost.exe in Windows 2000 – Microsoft Knowledge Base Article 250320", <http://support.microsoft.com/default.aspx?scid=kb;EN-US;250320> [Accessed: 10 February 2003].

- [16] First Virtual Communications Conference Server V6.0 Online Help
- [17] NIST Net, NIST Net Homepage, <http://www-x.antd.nist.gov/nistnet/> [Accessed: 3 February 2003].
- [18] Jonathan Davidson, "Voice over IP," Cisco Systems, June 1998, [http://www.dtr.com.br/cdrom/cc/sol/mkt/ent/ndsgn/voice\\_dg.htm](http://www.dtr.com.br/cdrom/cc/sol/mkt/ent/ndsgn/voice_dg.htm) [Accessed: 2 April 2003].
- [19] "Subjective Assessment of Telephone-Band and Wideband Digital Codecs," ITU-T Recommendation P.830, February 1996
- [20] "Perceptual evaluation of speech quality (PESQ), an objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs," ITU-T Recommendation P.862, February 2001

## APPENDIX A

### NETWORK IMPAIRMENT VALUES USED FOR SYSTEM TESTING

Bandwidth (kbit/sec)	Drop Percentage (%)	Jitter (ms)
10,000	0	0
100	1	10
64	2	20
57.6	3	30
38.4	4	40
28.8	5	50
19.2	6	60
9.6	7	70
	8	80
	9	90
	10	100
	11	110
	12	120
	13	130
	14	140
	15	150
	16	160
	17	170
	18	180
	19	190
	20	200

## APPENDIX B

### MATLAB SCRIPT FOR VOICE QUALITY TESTING

```
clear all;

%Fill the bandwidth, drops, and jitter arrays with the values from their respective text
files
FID_bw = fopen('bandwidth.txt','rt');
FID_drop = fopen('drops.txt','rt');
FID_jitter = fopen('jitter.txt','rt');

bw_index = 0;
%fill bandwidth array with vals from txt file
while 1

    temp = fgetl(FID_bw);
    if ~ischar(temp), break, end
    bw_index = bw_index + 1;
    bandwidth(bw_index,1:length(temp)) = temp;

end

drop_index = 0;
%fill drops array with vals from txt file
while 1

    temp = fgetl(FID_drop);
    if ~ischar(temp), break, end
    drop_index = drop_index + 1;
    drops(drop_index,1:length(temp)) = temp;

end

jitter_index = 0;
%fill drops array with vals from txt file
while 1

    temp = fgetl(FID_jitter);
    if ~ischar(temp), break, end
    jitter_index = jitter_index + 1;
    jitter(jitter_index,1:length(temp)) = temp;
```

end

```
fclose(FID_bw);
fclose(FID_drop);
fclose(FID_jitter);
i=1;
j=1;
k=1;
pause(10);
for i = 1:bw_index %this is the loop that changes bandwidth values (bytes/sec)
```

```
    [signal,FS,NBits] = wavread('SourceFile.wav');
```

```
    wavplay(signal,FS,'async');
```

```
    Deg = wavrecord(160000,8000,1); %Record 10 secs of voice
```

```
    filename = ['DegradedVoice_',];
    filename = strcat(filename,num2str(bandwidth(i,:)));
    filename = strcat(filename,'_');
    filename = strcat(filename,num2str(drops(j,:)));
    filename = strcat(filename,'_');
    filename = strcat(filename,num2str(jitter(k,:)));
    filename =strcat(filename,'.wav');
```

```
    wavwrite(Deg,FS,filename);
```

```
    pause(25);
```

end

```
i=1;
```

```
j=1;
```

```
k=1;
```

```
for j = 1:drop_index %this loop changes drop (%) values
```

```
    [signal,FS,NBits] = wavread('SourceFile.wav');
```

```
    wavplay(signal,FS,'async');
```

```
    Deg = wavrecord(160000,8000,1); %Record 20 secs of voice
```

```
    filename = ['DegradedVoice_',];
    filename = strcat(filename,num2str(bandwidth(i,:)));
    filename = strcat(filename,'_');
    filename = strcat(filename,num2str(drops(j,:)));
```

```

filename = strcat(filename, '_');
filename = strcat(filename, num2str(jitter(k,:)));
filename = strcat(filename, '.wav');

wavwrite(Deg, FS, filename);

pause(25);
end
i=1;
j=1;
k=1;

for k = 1:jitter_index %this loop changes Jitter values(in ms)
    [signal,FS,NBits] = wavread('SourceFile.wav');
    wavplay(signal,FS,'async');
    Deg = wavrecord(160000,8000,1); %Record 10 secs of voice
    filename = ['DegradedVoice_',];
    filename = strcat(filename, num2str(bandwidth(i,:)));
    filename = strcat(filename, '_');
    filename = strcat(filename, num2str(drops(j,:)));
    filename = strcat(filename, '_');
    filename = strcat(filename, num2str(jitter(k,:)));
    filename = strcat(filename, '.wav');

    wavwrite(Deg,FS,filename);

    pause(25);
end

```

## APPENDIX C

### LINUX SHELL SCRIPTS

1. *RunAfterBoot*: This script performs the following functions

- Sets the IP address of each Ethernet card in the PC
- Configures Linux PC to act as a router by enabling IP forwarding
- Loads the NIST Net module and starts the GUI.

```
#fix up routes
```

```
route del -net 132.177.88.0 netmask 255.255.252.0 dev eth0
```

```
route del -net 132.177.89.0 netmask 255.255.252.0 dev eth1
```

```
route add -net 132.177.88.0 netmask 255.255.255.0 dev eth0
```

```
route add -net 132.177.89.0 netmask 255.255.255.0 dev eth1
```

```
#turn on ip forwarding
```

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

```
#start Nistnet
```

```
/root/nistnet/Load.Nistnet
```

```
xnistnet
```



2. *VQ\_Testing*: This script automatically changes the NIST Net router settings during

testing

list=0

source\_IP\_addr="132.177.89.6" #this is the source of traffic

dest\_IP\_addr="0.0.0.0" #all addresses

#bw is bandwidth and the values are in bytes/sec

bw[1]=1250000

bw[2]=12500

bw[3]=8000

bw[4]=7200

bw[5]=4800

bw[6]=3600

bw[7]=2400

bw[8]=1200

#the values for drops are in percentages (%)

drop[1]=0

drop[2]=0.5

drop[3]=1

drop[4]=2

drop[5]=3

drop[6]=4

drop[7]=5

drop[8]=6

drop[9]=7

drop[10]=8

drop[11]=9

drop[12]=10

drop[13]=11

drop[14]=12

drop[15]=13

drop[16]=14

drop[17]=15

drop[18]=16

drop[19]=17

drop[20]=18

drop[21]=19

drop[22]=20

```

#the values of jitter are in milliseconds
jitter[1]=0
jitter[2]=10
jitter[3]=20
jitter[4]=30
jitter[5]=40
jitter[6]=50
jitter[7]=60
jitter[8]=70
jitter[9]=80
jitter[10]=90
jitter[11]=100
jitter[12]=110
jitter[13]=120
jitter[14]=130
jitter[15]=140
jitter[16]=150
jitter[17]=160
jitter[18]=170
jitter[19]=180
jitter[20]=190
jitter[21]=200

i=1
j=1
k=1
#echo ${temp[1]}
#first thing, turn on the NistNet emulator
cnistnet -u

#This while loop varies bandwidth
while [ $i -lt 9 ]
do
j=1
k=1

cnistnet -a $source_IP_addr $dest_IP_addr --delay 0 ${jitter[k]} --bandwidth ${bw[i]} -
-drop ${drop[j]}

i=$((i+1))
#echo Press any key to continue testing
#read keypress
sleep 30s
done

```

```

i=1
k=1
#This while loop varies drop %
while [ $j -lt 23 ]
do
cnistnet -a $source_IP_addr $dest_IP_addr --delay 0 ${jitter[k]} --bandwidth ${bw[i]} -
-drop ${drop[j]}

j=$((j+1))
#echo Press any key to continue testing
#read keypress
sleep 30s
done

j=1
i=1
#This while loop varies jitter
while [ $k -lt 22 ]
do

cnistnet -a $source_IP_addr $dest_IP_addr --delay 0 ${jitter[k]} --bandwidth ${bw[i]} -
-drop ${drop[j]}

k=$((k+1))

#echo Press any key to continue testing
#read keypress
sleep 30s
done

#echo $list
#list=$((list+1))

```

## **APPENDIX D**

### **CONFERENCE SERVER INSTALLATION AND ADMINISTRATION**

The instructions found in this section are used to install and administrate the First Virtual Communications Conference Server software on a server which is running Windows 2000 Advanced Server. In the instructions to follow, specific names of items such as buttons or screens are indicated by double quotation marks. For example, if a button named Next is to be pressed, it will be shown as “Next”.

#### **D.1 Installation Instructions**

1. Open the Conference Server “Setup.exe” file
2. Press the “Next” button of the “Welcome” screen
3. In the next screen to appear, “Install Type”, choose the “Custom Install” option and press the “Next” button
4. In the next screen, “Component Setup Types”, choose the “Full Install” option and press the “Next” button.
5. Press the “Begin” button to begin installation
6. Press the “Next” button on the “Welcome” screen
7. The next screen to appear is “Software License Agreement”. Choose to accept the agreement and press “Next”
8. The next screen to appear is “Customer Information”. Enter the user’s name and organization in the “Name” and “Organization” fields respectively. Enter the license key in the “CUseeMe Conference Server” field and press “Next”

9. Enter the server's IP address in the "IP Address" field and press "Next"
10. Choose the installation location on the server's hard drive and press "Next"
11. Press the "Install" button
12. When the "Installation Success" screen is displayed, the installation of the Conference Server is complete. Press the "Finish" button to continue
13. The "Welcome" screen for the Tomcat application appears next. Press "Next" to continue with Tomcat installation
14. The next screen to appear is "License". Choose to accept the agreement and press "Next"
15. Choose the installation location on the server's hard drive and press "Next"
16. Choose the desired port to use when connecting to the server to access the administrative web pages. Press the "Next" button to continue
17. Press "Install" to begin installation of the Tomcat application
18. When the "Installation Completed" screen appears, Tomcat has been installed successfully. Press the "Finish" button to continue
19. The "Welcome" screen for the Conference Server Administrator appears next. Press "Next" to continue with installation
20. The "Database" screen appears next. Enter the IP address and port number of the server where user records will be hosted. In a domain with one Conference Server, this is the IP address of the Conference Server. Press "Next" to continue
21. The "Administrative Server" screen appears next. Enter the IP address and host name of the server. In a domain with one Conference Server, select the option labeled "This will be the Administrative Server". Press "Next" to continue

22. In the “Ready To Install” screen that appears next, press the “Install” button to begin installation
23. When the “Installation Success” screen appears, the installation was successful. Press the “Finish” button to continue.
24. Choose the installation location on the server’s hard drive for the “Web Admission Center”. Press “Next” to continue
25. Press “Install” to install the “Web Admission Center”
26. When the “Installation Completed” screen appears, the “Web Admission Center” was installed successfully. Press “Finish” to continue
27. Installation of all Conference Server components is now complete. Choose the desired options on the “Installation Completed” screen and press “Finish”

## **D.2 Initial Configuration Instructions**

The instructions found in the following sections are for preparing the Conference Server for use in the Project54 VoIP system.

### **D.2.1 Allowing Telnet Access to Server**

After the initial installation, Telnet access to the server is not available. To allow Telnet access, first locate the “cucs.cfg” file on the server. This file holds the initialization parameters for the server. Add the following command to the “cucs.cfg” file: allow-wpconfig 255.255.255.255. This command allows any IP address to make a Telnet connection to the server, but password protection restricts access.

### **D.2.2 Setting Telnet Password**

The following instructions are used to add or edit a Telnet password for the Conference Server.

1. Run the “Regedit” Windows application
2. Open the “HKEY\_LOCAL\_MACHINE” folder in the left frame of the “Registry Editor” dialog box.
3. Open the “SOFTWARE” folder.
4. Open the “CUseeMe Networks” folder.
5. Open the “Conference Server” folder.
6. Open the “6.0” folder.
7. Double-click the “Management Password” icon in the right frame of the “Registry Editor” dialog box. The “Edit String” dialog box appears.
8. Enter the password to restrict Telnet access in the “Value data” text field.
9. Click “OK”
10. Close the “Registry Editor” dialog box.
11. Restart Conference Server so that the password will take effect.

### **D.2.3 Editing Default Conferences**

After installation, the Conference Server hosts a group of default conferences that are started whenever the server is reset. These conferences are not needed by the Project54 VoIP system and should be disabled. To permanently disable these conferences first locate the “cucs.cfg” file on the server. Remove the all of the conference descriptions located in this file. Save the “cucs.cfg” file and restart the server.











### **D.3 Administration Instructions**

The following information is used for managing conferences for use in the Project54 VoIP system.

#### **D.3.1 Accessing Web Interface**

To access the administration web pages for a Conference Server, open a web browser and point it to the following URL: `http://SERVER IP ADDRESS:SERVER PORT #/cucs/cucs.html`, where `SERVER IP ADDRESS` is the IP address of the Conference Server and `Port #` is the port number established during installation (default is 8080). This brings up the Conference Server Login page. If this is the first time logging in, the Username and Password will be “Administrator” and “cuseeme” respectively. Once logged in successfully, the username and password should be changed to provide necessary security measures. After logging in, the main entry page is displayed with buttons which are links to the five main areas of the interface. Once past the main entry page, navigation icons for the major areas will appear in a column on the left side of the browser window. These buttons and navigation icons are shown in Table D.1.



Area	Button on Entry Page	Navigation Icon
Network		
Conferences		
Users		
Monitoring		
Help		

**Table D.1 Links to Five Main Areas of User Interface**

The “Network” link in Table D.1 leads to the network configuration area of the user interface. If using more than one Conference Server, this is where server-wide settings are defined. Also from the “Network” area, a variety of tasks related to setup and maintenance of the Conference Servers can be performed.

The “Conferences” link leads to the “Conference” area of the user interface. The “Conference” area is where all conference configuration is performed.

The “Users” link leads to the “Users” area of the user interface. The “Users” area is where user accounts are created and managed. The server does not keep a record of every user to connect to the server, but keeps records of users with special access status.

The “Monitoring” link leads to the “Monitoring” area of the user interface. The “Monitoring” area is used to access real-time information concerning activity of the Conference Server and monitor and control client participation in an active conference.

The “Help” link displays the Conference Server Online Help which contains extensive information on using and configuring the Conference Server.

### **D.3.1 Conference Administration Using Web Interface**

#### **Creating a Conference**

1. Choose the “Conferences” link on the bottom of the entry page or left column of any other page
2. Press the “Add H.323” button near the top of the page
3. The conference settings web page appears next. Enter the desired “Conference ID” and “Conference Name”
4. Remove the check from the “Video” checkbox
5. Enter the maximum number of participants for the conference
6. Scroll down to the “Required Codecs” section. Enter the audio codec to be used for this conference. For the Project54 VoIP system, the codec used is G.723.
7. Scroll back to the upper section of the page and press the “Next” button
8. The next page to appear is used to define the server layout within the domain. If only one server is being used, no changes need to be made. Press the “Save the Conference” button indicated by Figure D.9.1 to continue.



**Figure D.9.1 Save the Conference Button**

9. Choose the “OK” button of the pop-up window labeled “Save the conference”. The conference has now been created successfully.

### **Deleting a Conference**

1. Choose the “Conferences” link on the bottom of the entry page or left column of any other page
2. Select the conference to delete in the conference information window
3. Press the “Remove” button from above the information window
4. Choose “OK” in the “Confirm Delete” pop-up window that appears
5. The conference is now deleted

### **Adding a User Record**

1. Choose the “Users” link on the bottom of the entry page or left column of any other page
2. Press the “ADD” button
3. Enter the “Username” and “Password” for the user
4. Select the permission this user will have
5. All other information on this form is optional and may be entered if desired
6. Choose “Add User”
7. A new user record has been created

### **Deleting a User Record**

1. Choose the “Users” link on the bottom of the entry page or left column of any other page
2. Press the “Search” button on the “User Menu Page” page that appears
3. Enter appropriate search criteria in the “Search User Database” page and click “Search”
4. Choose the selection check box next to the name of the user record you want to remove.
5. Choose “Remove Selected”. The “Remove User Record” page is displayed. Choose “Yes” to remove the user record
6. The selected user record has now been removed

### **Editing a User Record**

1. Choose the “Users” link on the bottom of the entry page or left column of any other page
2. Press the “Search” button on the “User Menu Page” page that appears
3. Enter appropriate search criteria in the “Search User Database” page and click “Search”
4. Choose the selection check box next to the name of the user record to edit
5. Press the “Edit” button. The “Edit User Record” page will be displayed
6. Make any desired edits and press “Submit Edits”
7. The changes to the selected user record has been made

## **APPENDIX E**

### **CONFERENCE CONFIGURATION APPLICATION INSTALLATION AND USAGE**

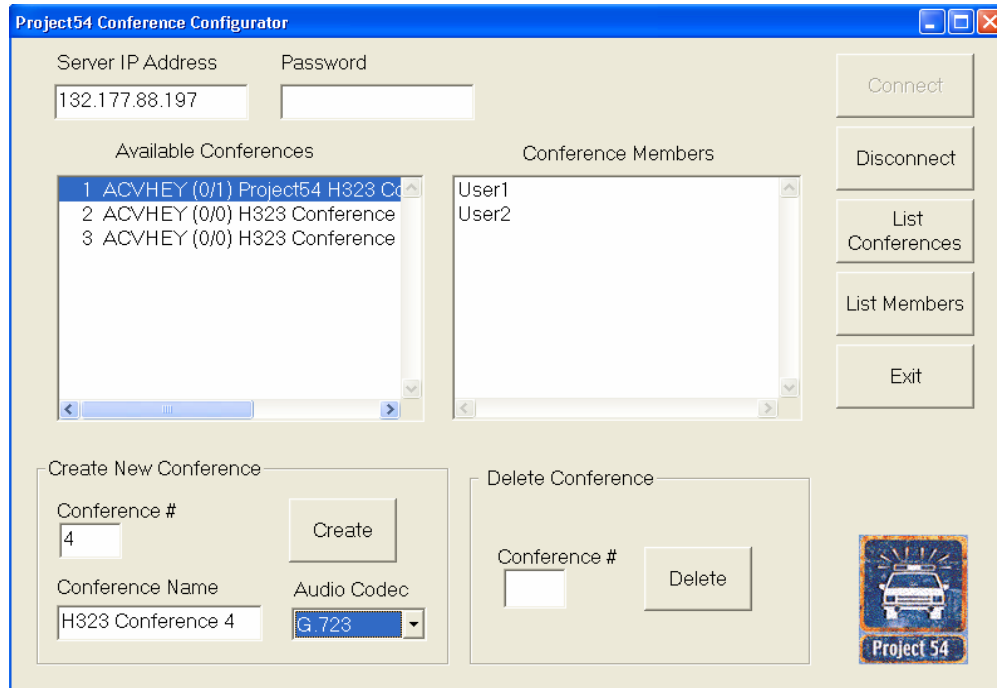
The instructions found in this section are for installing and using the Conference Configuration Application developed for the Project54 VoIP system.

#### **E.1 Installation Instructions**

To use the Conference Configuration Application, place the “Configure\_Conference.exe” executable on the PC being used for conference administration. The “Admin.txt” file, used by the application to determine user interface settings, should be located in the same directory path as the “Configure\_Conference.exe” executable. If this file is not present, one which doesn’t offer the administrator settings will be automatically created by the Conference Configuration Application.

## **E.2 Usage Instructions**

The fields and buttons referenced in the following instructions are shown in the GUI in Figure E.1.



**Figure E.1 Conference Configuration Application GUI**

### **Connecting to Conference Server**

1. Enter the server's IP address and password into the "Server IP Address" and "Password" fields respectively.
2. Press "Connect" button

### **Disconnecting from Conference Server**

1. Press the "Disconnect" button

### **Creating a Conference**

1. Enter a unique number to identify the conference in the “Conference #” field located in the “Create New Conference” area. This number cannot be a number that has already been assigned to a current conference.
2. Enter a name to identify the conference in the “Conference Name” field.
3. (Optional) Select an audio codec in the “Audio Codec” pull-down box. This option is only available in the administrator version of the Conference Configuration Application.
4. Press “Create” button

### **Deleting a Conference**

1. Enter the number to identify the conference to be deleted in the “Conference #” field located in the “Delete Conference” area.
2. Press “Delete” button

### **List Members of a Conference**

1. Select a conference in “Available Conferences” field
2. Press “List Members” button
3. A list of members of the selected conference will appear in the “Conference Members” field

**Refresh Available Conferences List**

1. Press the “List Conferences” button
2. A current list of the conferences currently being hosted by the conference server will be displayed in the “Available Conferences” field



## **APPENDIX F**

### **PROJECT54 VOIP CLIENT APPLICATION INSTALLATION AND USAGE**

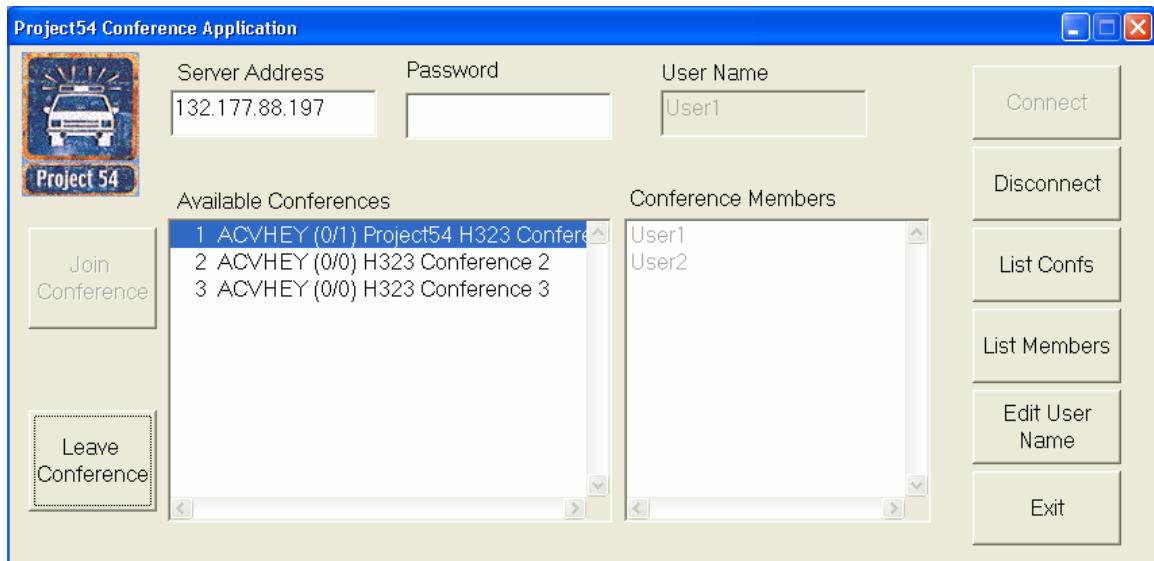
The instructions found in this section are for installing and using the Client Application developed for the Project54 VoIP system.

#### **F.1 Installation Instructions**

To use the Client Application, place the “P54 Conference.exe” executable on the PC being used for conference participation. The “Admin.txt” file, used by the application to determine user interface settings, should be located in the same directory path as the “P54 Conference.exe” executable. If this file is not present, one which doesn’t offer the administrator settings will be automatically created by the Client Application. The “username.txt” file, used by the application to determine the user’s username for conference participation, should also be located in the same directory path as the “P54 Conference.exe” executable. If this file is not present, the user will be given the opportunity to set a username when the Client Application is run for the first time. When the username is set, the “username.txt” file will be automatically created by the Client Application.

## **F.2 Usage Instructions**

The fields and buttons referenced in the following instructions are shown in the GUI in Figure F.1.



**Figure F.1 Project54 Conference Client Application**

### **Connecting to Conference Server**

1. Enter the server's IP address and password into the "Server IP Address" and "Password" fields respectively.
2. Press "Connect" button

### **Disconnecting from Conference Server**

1. Press the "Disconnect" button

### **List Members of a Conference**

1. Select a conference in “Available Conferences” field
2. Press “List Members” button
3. A list of members of the selected conference will appear in the “Conference Members” field

### **Refresh Available Conferences List**

1. Press the “List Conferences” button
2. A current list of the conferences currently being hosted by the conference server will be displayed in the “Available Conferences” field

### **Join a Conference**

1. Select desired conference in “Available Conferences” field
2. Press “Join Conference” button
3. All members of the conference joined will be displayed in the “Conference Members” field

### **List Members of a Conference**

1. Select desired conference in “Available Conferences” field
2. Press “List Members” button
3. A current list of members of the selected conference is displayed in the “Conference Members” field

### **Modify User Name**

1. Press the “Edit User Name” button. The name of the button will become “Set User Name”
2. The “User Name” field becomes accessible
3. Edit the user name
4. Press the “Set User Name” button. The client’s username has been changed and the “User Name” field becomes inaccessible
5. The “Set User Name” button reverts back to “Edit User Name”

## **APPENDIX G**

### **NIST NET INSTALLATION AND USAGE GUIDE**

The instructions in this section assume NIST Net version 2.0.10 being installed on a desktop PC running Red Hat Linux 7.2.

#### **G.1 Installation Instructions**

1. The NIST Net package is distributed in a gzipped Tar file. At the command prompt navigate to the directory to install the NIST Net directory. Enter the following command: “tar -xvzf nistnet.2.0.10.tar.gz”. This produces a “nistnet” directory.

2. Enter the following commands to set up the Linux configuration:

- cd /usr/src/linux
- make menuconfig
- make dep

3. Build and install the NIST Net module, API library, and user interface by entering the following commands:

- cd nistnet
- make
- make install

4. The following commands must be entered after installing NIST Net for the first time:

- `mknod /dev/hitbox c 62 0`
- `mknod /dev/nistnet c 62 1`
- `mknod /dev/mungebox c 63 0`
- `mknod /dev/spybox c 64 0`

5. The NIST Net module must be loaded after every reboot of the computer. Enter the following command to load the module: `"/nistnet/Load.Nistnet"`

## **G.2 Usage**

There are two interfaces to the NIST Net module. The first is the GUI interface shown in Figure 6.3. The second is the “cnistnet” command line interface. The options to use with the “cnistnet” command are shown below:

1. “-u”

- Turn on the NIST Net module

2. “-d”

- Turn off the NIST Net module

3. “-a” src dest --Network impairment(s)

- Adds one or more of the following network impairments between src and dest:

`--delay delsigma` Delay in milliseconds followed by milliseconds of jitter

`--drop` Percentage of packets to drop

`--dup` Percentage of duplicate packets

`--bandwidth` Bandwidth in bytes/second

4. “-r” src dest --Network impairment(s)

- Removes one or more of the following network impairments between src and dest:

--delay  $\mu$   $\sigma$  Delay in milliseconds followed by milliseconds of jitter

--drop                      Percentage of packets to drop

--dup                        Percentage of duplicate packets

--bandwidth                Bandwidth in bytes/second

5. “-h”

- Displays a list of all NIST Net commands