Mouth-To-Ear Latency in Popular VoIP Clients

Chitra Agastya, Dan Mechanic, and Neha Kothari
Department of Computer Science
Columbia University, New York, NY 10027
{csa2111, mechanic, nk2338}@columbia.edu

July 9, 2009

ABSTRACT

Most popular instant messaging clients are now offering Voice-over-IP (VoIP) technology. The many options running on similar platforms, implementing common audio codecs and encryption algorithms offers the opportunity to identify what factors affect call quality. We measure call quality objectively based on mouth-to-ear latency. Based on our analysis we determine that the mouth-to-ear latency can be influenced by operating system (process priority and interrupt handling), the VoIP client implementation and network quality.

1. INTRODUCTION

Many IP telephony clients and instant messaging clients are now offering VoIP technology. These are distributed for popular platforms such as Windows and Linux. The combination of many clients running on similar platforms, implementing common audio codecs and encryption algorithms offers the opportunity to perform a comparative study to identify what factors affect call quality. We measure call quality objectively using a mouth-to-ear delay metric.

Mouth-to-ear delay measures the time delay between when the speaker utters a word and when the listener actually hears it. Mouth-to-ear delay can be influenced by: A/D-D/A audio conversion, VoIP client implementation (including codecs and encryption solutions) , operating system and network quality.

2. EXPERIMENTS

The measurements for mouth-to-ear latency are performed using the adelay utility [2] against audio files where a source audio signal is recorded on the left channel and the right channel contains the source audio delayed by the cost of a VoIP call including the A/D-D/A conversion of the audio.

2.1 Experiment Setup

Source audio from an MP3 player is split via its headphone jack using a 1/8" Stereo TRS (tip, ring, sleeve- Figure 1) - 2x Mono TS (tip, sleeve) audio splitter (Figure 2). One channel is then patched to the mic-in on the caller machine. The callee machine's headphone jack and the second channel from the mp3 player are connected to the line-in jack of a MacBook laptop using a 2x Mono - 1x Stereo converter. The resulting stereo audio is recorded and converted to Sun AU files using Audacity 1.2.5 [12] on a MacBook laptop via the line-in interface. Fig 3. sows the setup.

Over the course of six weeks (Nov-Dec 2008) we captured 326 samples of various combinations of platforms, clients, codecs and encryption options. Additionally, metrics regarding memory usage and OS priority were taken from the caller and callee machines during recording. The audio files were then analyzed for latency using the adelay utility developed in the Internet Real-Time Laboratory at Columbia University [2]. Figure 3 shows the setup for testing mouth-to-ear latency on soft VoIP clients.

2.2 Hardware

The following are the details of the hardware used in our experiments:

The caller machine, which receives our sound source via its microphone input, is a Dell Optiplex GX260 (Intel Pentium 4 2.8Ghz machine with 1 Gig RAM, Intel 82540EM Gigabit Ethernet Controller and an Intel 82801DB/DBL/DBM AC'97 Audio Controller. The callee machine, which receives audio from the network, is a Dell Optiplex GX620 with a Dual Core Intel Pentium D 3.4Ghz, 1 Gig RAM, Broadcom NetXtreme BCM5751 Gigabit Ethernet and an Intel 82801G AC'97 Controller.

2.3 Operating Systems

The experiments were run to test VoIP clients on two very popular platforms: Linux and Windows XP

2.3.1 Linux

We used two flavors of the Linux kernel to perform our tests. One version was Fedora Core 9's default kernel 2.6.27.5-37.fc9 with Alsa sound library version alsa-lib-1.0.17-2-fc9. The Alsa audio libraries [13] have become the default for Linux as of kernel version 2.6 [11].

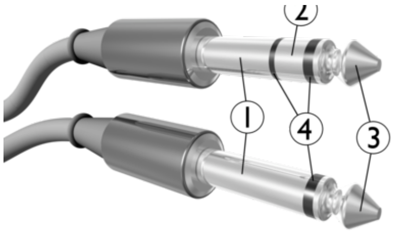

Figure 1: Image showing sleeve (1) ring (2) and tip (3) and insulating ring (4) sections of TRS/TS connectors [16]

Figure 2: Image showing TRS to 2xTS splitter

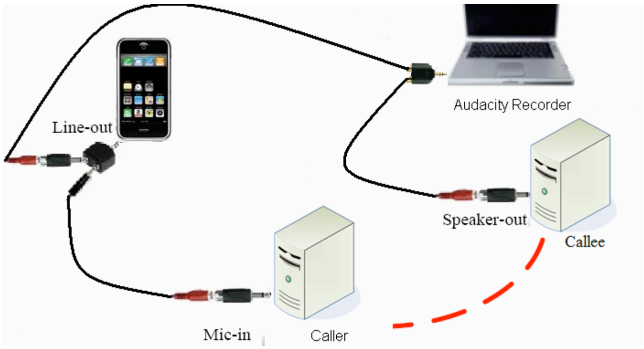

Figure 3: Setup for measuring mouth-to-ear latency

Users of audio recording and production software have demanding requirements for their systems to handle high quality audio whilst maintain extremely low latencies. The Stanford University Center for Computer Research in Music and Acoustics (CCRMA), distribute rpm'd kernels for Fedora which include real-time preemption patches by Ingo Molnar. According to CCRMA: "While the stock Fedora kernels will also work for non-critical audio work the real-time preemption patches are pretty much a requirement for reliable behavior at low latencies" [7]. In addition to the standard 2.6.27.5-37.fc9 kernel provided by the Fedora repositories, we tested clients on the real-time kernel, kernel-rt-2.6.24.7-1.rt3.2.fc9 kernel provided by CCRMA (referred to as RT in the graphs in this report)

The PulseAudio networked sound server is enabled by default in Fedora, however we removed these packages. With PulseAudio we experienced difficulty getting sound to work consistently. PulseAudio [14] is currently still under development and a major stable release was not available at the time of our experiments.

2.3.2 Windows

We used Windows XP Professional version 2002 Service Pack 2 for testing clients.

3. MOUTH TO EAR DELAY TESTS

Mouth-to-ear delay is the time delay incurred in speech by the IP telephony system. This is usually measured in milliseconds and is the time taken from when a user begins to speak until when the listener actually hears the speech. This one-way latency is known as mouth-to-ear delay. The ITU-T recommendation is that “up to 150 ms mouth-to-ear delay can be tolerated by the human ear with virtually no quality loss”.

Table 2 in the Appendix A shows how one study [15] has suggested the relationship between the perceived link quality vs. mouth-to-ear delay for IP telephony [4].

3.1 Basic Loopback Tests

We performed simple loopback tests in an attempt to remove the A/D-D/A conversion variable from the equation for latency. These experiments were staged using audio cables to perform loopback tests by routing the audio via the mic-in and out via a headphone jack on the same machine running a simple sound recording program. Figure 4 shows the setup for the basic Loopback tests we performed. These tests were performed on both the Linux kernels as well as on the Windows platform.

This experiment failed, apparently due the audio hardware in our test machines. No delay could be detected even when a second machine was piggybacked into the audio chain. The AC’97 audio controller has a front end analog audio mixer which allows analog routing (no A/D-D/A conversion) from microphone/line-in to line-out [3]. All tests in this fashion showed no delay. A loopback client to force conversion would be necessary to properly test A/D-D/A conversion delay, however such a client would introduce it’s own delay. These tests were abandoned.

3.2 Hard Phones

We performed the mouth to ear latency tests on two Grandstream GXP2000 phones. Figure 5 illustrates the set up for the latency test with the Grandstream phones. The Grandstream GXP2000 phone supports a 2.5 mm audio jack. To perform these experiments we first configure the Grandstream phones as SIP user agents with user names and IP addresses. Then, we connect one mono line from the audio source directly into the audacity recorder. The second line from the audio source is patched into the audio in of one of the phones using a standard 1/8” audio jack to 2.5mm audio jack converter. Finally we connect the audio out of the other phone into the Audacity recorder. Then, calls from one Grandstream phone to the other are established using their configured IP addresses.

We performed the mouth-to-ear latency tests on these phones for the following codecs: G.711 µ-law, G.711 a-law for both RTP as well as encrypted RTP (SRTP) media. We attempted using iLBC codec. However, the phones experienced difficulty with this codec

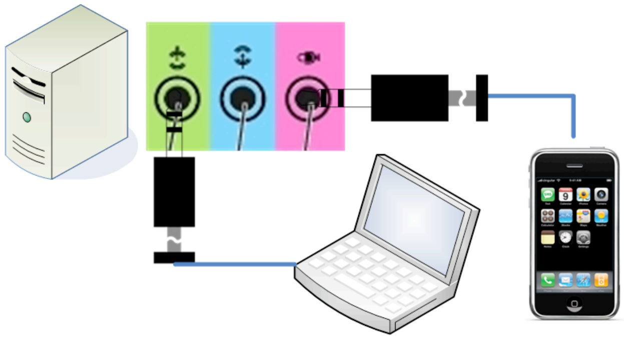

Figure 4: Setup for loopback test

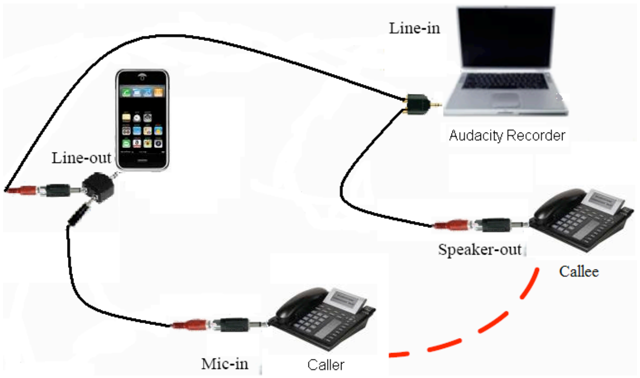

Figure 5: Setup for measuring hard phone latency

and produced audio distortion rendering the recordings useless. We did not take packetization intervals into account. The usable readings collected from these tests are tabulated in Table 3 in the Appendix A.

In our tests, we observe that the latency is larger for packets with SRTP encryption than for packets without encryption.

3.3 Soft Phones

We performed tests using several VoIP clients on Linux as well as on the Windows operating systems. While taking measurements, we performed the tests on different codecs supported by the clients.

3.3.1 Tests on Linux

In order to see how real-time kernels can impact latency, we performed tests on two different versions of the kernel: the Linux kernel as distributed by Fedora and a real-time kernel distributed by the planet CCRMA project at Stanford University.

Since none of the popular VoIP clients (gtalk, Windows messenger, and yahoo) are distributed for Linux with the exception of Skype, we performed the tests on some VoIP clients we used for the VoIP lab exercises. Clients we performed our experiments on include Ekiga, Linphone, Skype and Twinkle.

Ekiga: We used Ekiga version 2.0.12-2.fc9 [10] for our experiments. Ekiga supports many audio codecs. We performed our tests for some of the popular codecs: Speex 16kHz, Speex 8kHz, iLBC, G.711 μ-law, G.711 a-law and G721. We ran these tests on both the real-time kernel as well as the regular kernel. Ekiga does not support encryption so no latency tests were performed with encryption.

Linphone: We used Linphone version 2.1.1-1.fc9 for our experiments and performed our tests on the client for the following codecs: Speex 16kHz, Speex 8kHz, G.711 μ-law, G.711 a-law. We ran these tests on both the real-time kernel as well as the regular kernel. Linphone also does not support encryption so no latency tests were performed with encryption.

Skype: We used Skype version 2.0.0.72-fc5.i586 for our tests. Skype is perhaps the most popular peer to peer VoIP client and is distributed for both Windows as well as Linux. We found Skype to be an interesting candidate for this project as we could perform the latency tests for our three operating systems: FC9 kernel, CCRMA real-time kernel and Windows XP.

Twinkle: Twinkle is a SIP client that we used extensively for our VoIP lab exercises. We consider Twinkle an interesting candidate for these tests because it supports many codecs and also supports encryption using zRTP, a key agreement protocol used to improve upon the security of SRTP. We used Twinkle version 1.2-3.fc9.i386 for our experiments. We observe the latency to be larger when packets are encrypted. We also observe that the latency is larger for the FC9 kernel vs. real-time kernel.

Table 4 through Table 8 shows the different results for our tests on Linux.

3.3.2 Tests on Windows

To make a comparative analysis of clients on different operating systems we performed our experiments on Windows as well. The clients we tested include Skype, AIM, yahoo, Ekiga, Windows messenger and gtalk. Ekiga and Skype are the only clients in this set available for both Windows and Linux. We noted the bit rate and also the memory footprint for each client when the call is in process. To identify the codecs that these clients support, we trace the signaling packets using Wireshark. The clients that use SIP signaling mention the media attributes with the codecs in the SDP message.

AIM: We used AIM v6.8.14.6. AIM uses the GIPS iSAC codec. One interesting observation we made during our tests was that this client sometimes has flash advertisements during a call. The latency values were seen to go as high as 786 ms when such advertisements were being downloaded. In the absence of these advertisements, the call latency was seen to be around 120ms.

Ekiga: We used Ekiga v3.0.0 for our tests on Windows. We consider Ekiga an interesting candidate because it is distributed

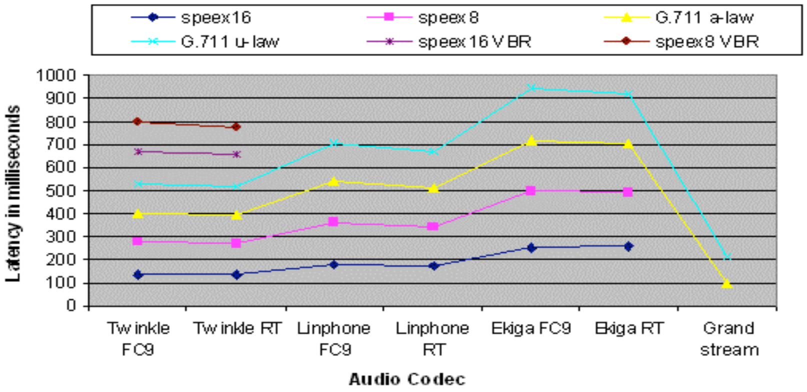

Figure 6: Graph showing latency trends by codec for different VoIP clients

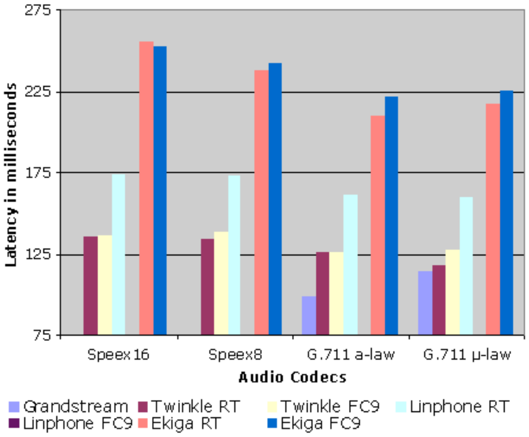

Figure 7: Chart showing latency values by codec for different VoIP clients

for both Windows and Linux platforms. This client also supports a variety of codecs, which the user can choose from.

Gtalk: Google Talk supports the following standard voice codecs: G.711 a-law, G.711 µ-law, G.723, iLBC, and Speex. It also supports iSAC, IPCMWB, EG711U, EG711A. Gtalk does not allow the user to specify codec preference [5].

Windows Messenger: We used Windows Messenger v4.7.3000 for our experiments. Wireshark captured SDP packets showing that this client supports a variety of codecs: red, SIREN, G7221, DVI4, G.711 a-law, G.711 µ-law, G723 and GSM.

Figure 1: Chart showing latency for different Windows clients

Skype: We used Skype v3.8.0.180 for our experiments. The readings for Skype provide a good comparison between Windows and Linux. Skype reports the codec it is using as Sinusoidal Voice Over Packet Coder (SVOPC).

Yahoo: We used Yahoo messenger v9.0.0.2034 for our tests. This client reports using the TrueSpeech codec.

Table 9 shows our results for Windows.

4. ANALYSIS

The experiments for this project were performed over a period of 6 weeks (Nov-Dec'08). The data that we have collected has been used to analyze some of the factors that affect the mouth-to-ear latency.

4.1 Codecs

The analysis of the different codecs used by the clients we tested on Linux is given below for both the kernels.

Figure 7 shows the latency of different clients using different codecs across the real-time(RT) as well as Fedora Core9 (FC9) kernel. The Grandstream phones seem to have least latency which could possibly be explained by the dedicated nature of their hardware.

As can be seen from the figure 6, different clients seem to show the same trend for similar codecs. For example, latency for Speex16 and Speex8 follows a similar trend for different clients. Similarly G.711 µ-law and G.711 a-law follow similar trends for different clients. Codecs performing well on one client seem to perform well across all clients (e.g. G.711 A-law) and codecs performing badly on one client seem to perform badly on all clients (e.g. iLBC). The effect of the codec seems to be a constant across all the clients. Another observation is that Twinkle appears to show similar latency for different codecs. This latency is under 150ms for all codecs used in the experiments.

4.2 Kernels

As predicted, the real-time CCRMA kernel improves upon the performance of processing audio over FC9 kernel in most cases.

Figure 10 shows the mouth-to-ear latency for the two flavors of linux kernels used in our experiments. As seen from the graphs, the latency values on CCRMA are lower than that on the FC9 kernel. The same observations are made for Ekiga and Linphone.

4.3 Encryption

Among the clients we tested, only two support encryption: Twinkle and Grandstream. While Grandstream uses sRTP to encrypt packets, Twinkle uses zRTP. Encryption appears to increase latency. The graph for latency of encrypted vs unencrypted packets for Twinkle and Grandstream are shown in the Figures 8 and 9.

The graphs clearly illustrate that the latency incurred on encrypted media packets is greater. The percentage overhead for encryption in the case of Twinkle is 6.67% for the real-time kernel and 6.83% for the FC9 kernel. The percentage overhead for encryption in the case of Grandstream is 14.15%.

4.4 Windows Clients

Most of the clients we tested on Windows did not support selecting a codec explicitly. Figure 11 shows the latency for different VoIP clients tested on Win XP.

Our analysis shows that Windows Messenger has the smallest latency. Windows Messenger is the only client test which uses Siren, a codec licensed from Polycom, which may contribute to it's fine performance. Other popular clients like AIM, gtalk, Skype and Yahoo have higher latency values. Of these, AIM, gtalk and Skype all have similar latency values, likely due to using the same GIPS iSAC audio codec.

The readings observed here vary from the ones observed in a similar tests conducted in 2005 at Columbia University[6]. In

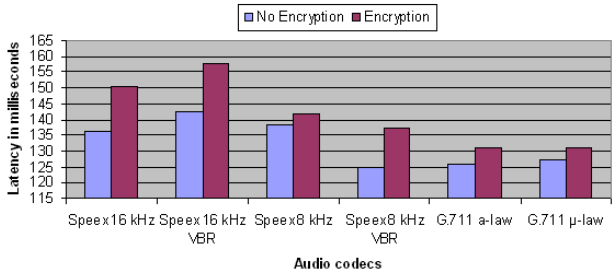

Figure 8: Chart showing effect of encryption on Twinkle latency

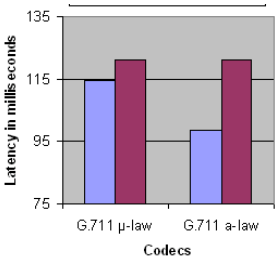

Figure 9: Chart showing effect of encryption on Grandstream latency

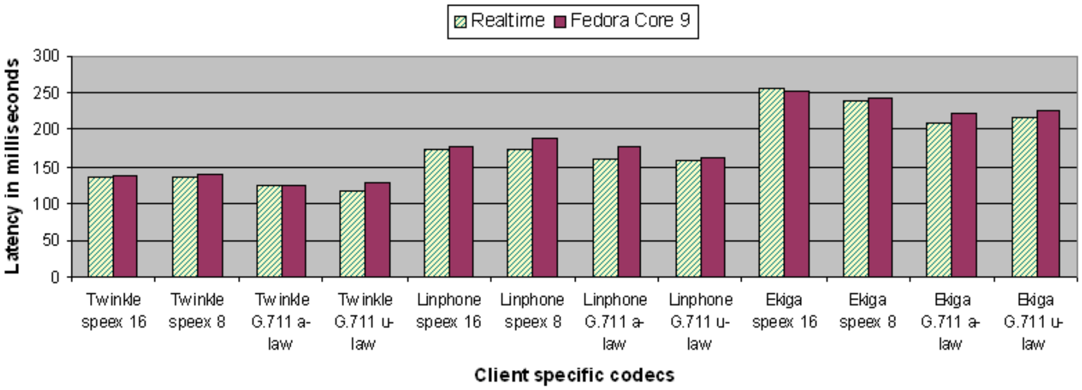

Figure 10: Chart comparing latency on Realtime vs Fedora Core 9 kernels on Linux

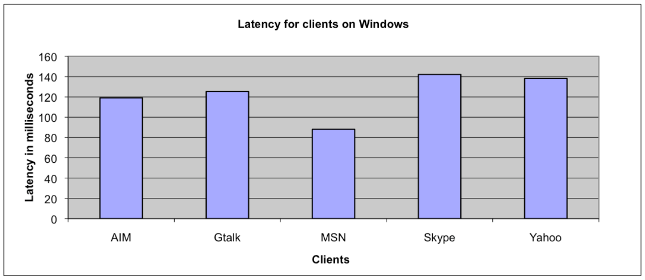

Figure 11: Chart showing latency for different Windows clients

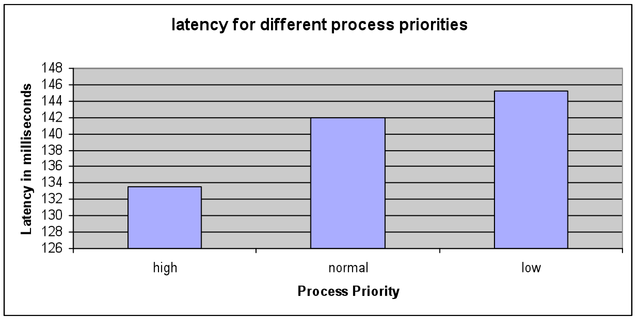

Figure 12: Chart showing Skype latency for different process priorities

those tests Skype had a much lower latency than the other IM clients. The older version (v1.4.0.84) ran at a high process priority while the current version (v3.8.0.180) runs at normal.

To complete our analysis, we tested Skype with different process priorities by setting the priority to high, normal and low and calculated the mouth-to-ear latency. The results are plotted in Figure 12. There was a difference in latency of roughly 7 milliseconds between high and normal priorities and 4 milliseconds between normal and low priorities.

4.5 Comparisons across all platforms

Table 1 in the Appendix A shows the latency for the different clients in increasing order of mouth-to-ear latency. Our observation showed majority of the clients have latencies between 0-300ms range. We did not observe any latency in the 300-450ms range. Ekiga on windows shows latency values greater than 450ms.

4.6 Conclusions

The results we have gathered suggest that operating system features such as real-time kernels and process priority and VoIP client implementations such as encryption do affect call quality. Codecs seem to show a similar performance trend across clients. However, more investigation needs to be done with respect to packetization interval of the codecs to understand their behavior better.

4.7 Further Work

From our observations we see several things worthy of investigation: Windows Messenger has the best performance among all clients we tested, including hard phones. What are the underlying reasons? Is it possible that Microsoft took advantage of owning operating system to improve performance, or is it the codec? Could extensions be made to Alsa to improve its performance for VoIP? And of course, are these improvements at the expense of some subjective measure of quality, such as MOS.

5. ACKNOWLEDGMENTS

We are thankful to Alan Crosswell of CUIT and Andrea Forte for providing us with the necessary hardware for the experiment setup. Our sincere thanks to Salman Abdul Baset for guiding us through the project and providing us with the NIC cards for the MOS experiment setup. We thank Supreeth Subramanya for checking our progress on the project. Finally we like to thank Prof. Henning Schulzrinne for his suggestions on future work and for reviewing this report.

6. REFERENCES

[1] http://en.wikipedia.org/wiki/Mean_Opinion_Score

[2] http://www.cs.columbia.edu/irt/software/adelay/

[3] http://download.intel.com/support/motherboards/desktop/sb/ac97_r23.pdf

[4] http://www.telephonyworld.com/training/brooktrout/iptel_latency_wp.html

[5] http://code.google.com/apis/talk/open_communications.html

[6] www1.cs.columbia.edu/~salman/presentations/skype-infocom06.ppt

[7] http://ccrma.stanford.edu/planetccrma/software/installplaneteight.html

[8] http://www.alsa-project.org

[9] http://ekiga.org/

[10] http://en.wikipedia.org/wiki/ZRTP

[11] William von Hagen "Migrating to Linux kernel 2.6 -- Part 3: Using the 2.6 Kernel with your current system" whitepaper TimeSys Corporation

[12] http://audacity.sourceforge.net

[13] http://www.alsa-project.org/

[14] http://www.pulseaudio.org

[15] Brooktrout Whitepaper “Understanding Latency in IP Telephony”

[16] http://en.wikipedia.org/wiki/TRS_connector

Appendix A

Table 1: Table showing the latency of all clients tested

Client	OS	Kernel	Codec	Latency
MSN	Windows	XP	Siren	88.05
Grandstream	NA	NA	G.711 A-law	98.86
Grandstream	NA	NA	G.711 µ-law	114.61
Twinkle	Linux	Fedora Core 9	G.711 µ-law	116.17
Twinkle	Linux	CCRMA	Speex 8 kHz VBR	118.22
AIM	Windows	XP	GIPS iSAC	119.10
Twinkle	Linux	Fedora Core 9	Speex 8 kHz VBR	124.76
Gtalk	Windows	XP	GIPS iSAC	125.28
Twinkle	Linux	CCRMA	G.711 A-law	125.80
Twinkle	Linux	Fedora Core 9	G.711 A-law	125.84
Twinkle	Linux	CCRMA	Speex 8 kHz	134.22
Twinkle	Linux	CCRMA	Speex 16kHz	135.63
Twinkle	Linux	Fedora Core 9	Speex 16kHz	136.11
Yahoo	Windows	XP	TrueSpeech	138.20
Twinkle	Linux	Fedora Core 9	Speex 8 kHz	138.24
Twinkle	Linux	CCRMA	Speex 16 kHz VBR	141.12
Skype	Windows	XP	SVOPC	142.18
Twinkle	Linux	Fedora Core 9	Speex 16 kHz VBR	142.54
Grandstream	NA	NA	iLBC	156.04
Linphone	Linux	CCRMA	G.711 µ-law	159.21
Linphone	Linux	CCRMA	G.711 A-law	160.80
Linphone	Linux	Fedora Core 9	G.711 µ-law	163.88
Linphone	Linux	CCRMA	Speex 8 kHz	172.96
Linphone	Linux	CCRMA	Speex 16 kHz	173.65
Linphone	Linux	Fedora Core 9	Speex 16 kHz	176.98
Linphone	Linux	Fedora Core 9	G.711 A-law	177.02
Linphone	Linux	Fedora Core 9	Speex 8 kHz	187.95
Ekiga	Linux	CCRMA	G.721	204.45
Ekiga	Linux	CCRMA	G.711 A-law	209.54
Ekiga	Linux	Fedora Core 9	G.721	213.07
Ekiga	Linux	CCRMA	G.711 µ-law	216.88
Skype	Linux	Fedora Core 9	SVOPC	218.72
Ekiga	Linux	Fedora Core 9	G.711 A-law	221.88
Ekiga	Linux	Fedora Core 9	G.711 µ-law	225.69
Skype	Linux	CCRMA	SVOPC	234.87
Ekiga	Linux	CCRMA	Speex 8 kHz	237.91
Ekiga	Linux	Fedora Core 9	Speex 8 kHz	242.59
Twinkle	Linux	CCRMA	G.711 µ-law	247.67
Ekiga	Linux	Fedora Core 9	Speex 16 kHz	252.75
Ekiga	Linux	CCRMA	Speex 16 kHz	255.31
Ekiga	Linux	CCRMA	iLBC	262.95
Ekiga	Linux	Fedora Core 9	iLBC	295.30
Ekiga	Windows	XP	G.711 µ-law	467.21
Ekiga	Windows	XP	Speex 16kHz	483.03
Ekiga	Windows	XP	Speex 8kHz	624.73
Ekiga	Windows	XP	iLBC	635.88
Ekiga	Windows	XP	G.711 A-law	677.26

Table 2: Perceived quality vs. mouth-to-ear latency

Latency in ms	Perceived Quality
0-150	Excellent
150-300	Good
300-450	Poor
>450	Unacceptable

Table 3: Latency for Grandstream, in ms

Codec	RTP	SRTP
G.711 µ-law	114.61	121.18
G.711 a-law	98.86	121.17

Table 4: Mouth-to-ear latency in for Ekiga, in ms

Codec	CCRMA kernel	FC9 kernel
Speex 16kHz	255.31	252.75
Speex 8 kHz	237.91	242.59
iLBC	262.95	295.30
G.711 µ-law	216.88	225.69
G.711 a-law	209.54	221.85
G721	204.45	213.07

Table 5: Mouth-to-ear latency reading for Linphone, in ms

Codec	CCRMA kernel	FC9 kernel
Speex 16kHz	173.65	176.98
Speex 8 kHz	172.96	187.95
G.711 µ-law	159.21	163.88
G.711 a-law	160.78	177.02

Table 6: Mouth-to-ear latency for Skype, in ms

CCRMA Kernel	FC9 Kernel
234.87	226.04

Table 7: Mouth-to-ear latency for Twinkle, in ms

Codec	CCRMA Kernel		Linux Kernel	
	RTP	ZRTP	RTP	ZRTP
Speex 16	135.63	156.04	136.11	150.43
Speex16 VBR	141.12	142.85	142.54	157.86
Speex8	134.22	134.76	138.23	141.77
Speex8 VBR	118.22	133.06	124.76	137.32
G.711 a-law	125.79	128.25	125.84	131.02
G.711 µ-law	117.96	128.36	127.41	131.23

Table 8: Mouth-to-ear latency reading for Ekiga, in ms

Codec	Latency in ms
Speex16	483.03
Speex8	624.73
iLBC	635.88
G.711 µ-law	467.21
G.711 a-law	677.26

Table :9 Latency of clients on Windows XP SP2, in ms

Clients	Latency in ms
AIM	119.10
Gtalk	125.28
Windows Messenger	88.05
Skype	142.18
Yahoo	138.20