Voice over IP Telephony- phone to phone

**Ramya Dharshini Chandrasekhar, Janardhan Lavakumar,
Harsha Munikoti, Sajan Peter, David Regal**

Abstract

Voice over IP is short for Voice over Internet Protocol (VoIP). The group implemented the Telephone to Telephone model. The vision of this project involved setting up a small VoIP system successfully. In the future, this system can be the basis for a larger commercially viable system.

1. Introduction

VoIP can be defined as the ability to make telephone calls over IP based telephone networks with a suitable quality of service (QoS) and a significant cost/benefit. Voice information is sent in digital form in discrete packets, rather than the traditional circuit switched protocols of the PSTN. The system being built was tested on two phone lines. Future projects can build upon this and create a system that implements a VoIP between any two phones in a LAN.

1.1 Specifications

- 7-digit dialing service will be provided by the end product.
- Capability to detect errors, but no error correction algorithm is implemented. An error in dialing the number will prompt the user to enter the phone number again.
- The user is prompted by the service for the final number to be dialed using a dialtone.
- End-product is an easy-to-use desktop VoIP box that calls into a server for low-cost phone calls.

1.2 Challenges and problems faced

- Number of specification requirements the gateway program can fulfill without modification.
- Passing the second number to the destination PC
- Incompatibility between LineJack Drivers downloaded and the ones used by OpenH323.
- Tracking changes made by multiple programmers.
- Understanding undocumented source code of OpenH323 project.
- Scalability issues, since there is only one PSTN port for one LineJack card.

1.3 Diagram of the system and step-by-step desription of a remote call.

Figure. 1 has a detailed diagram explaining each step sequentially.

2 Programs and libraries used in the OpenH323 software suite

2.1 OpenPhone program (version 1.8.1)

OpenPhone is a GUI H.323 client program. It is only available for Windows. The user-friendly interface has many features. For instance, you can set speed dial keys. If you're using a LineJACK or PhoneJACK card, you can simply press one button and the "#" key to dial the IP of another computer that accepts H.323 calls.

A H.323 client is basically a way to have PC to PC audio or video conversations over a LAN or Internet. You have to know the IP of the PC you want to call

and be by your computer to speak into the microphone. This program is similar to MS NetMeeting or Gnomemeeting.

Debugging would be difficult since trace has little information. When running the program with the "trace" option on (trace creates a text file for debugging), OpenPhone provides little information for insight into what is happening in the program. However, since OpenPhone worked on the first try, there is no need for debugging, but it would be nice to use the trace output file for reference of a working H.323 call.

2.2 OhPhone program (version 1.3.7)

This program is available for both Windows and Linux. Also a H.323 client, but command line instead of GUI. Since it is not a GUI, knowledge of the syntax to run the program is needed. But compared to pstngw.exe, it is simple to learn and only a few arguments are needed to make a call to another computer that accepts H.323 calls.

2.3 PSTNgw program (version 1.2.2)

Main program used to for any system that will be making or accepting calls from the PSTN (Public Switched Telephone Network) and converting the PSTN call into H.323 format. Even though it has been claimed that this program should work with LineJACK, PSTNgw from OpenH323 has not been guaranteed or tested to work with these cards.

Syntax for this program is cryptic at first, and the readme.txt (with outdated commands) provides little help for the argruments that are needed by the options. If you try to view the help output ("pstngw.exe –h") from a MS-DOS shell, it flies off the top of the screen and you'll only see the last part of it. To capture the whole output, at the DOS prompt type:

```
pstngw.exe –h | MORE > pstnhelp.txt
```

Open the "pstnhelp.txt" text file that was created in the current directory and you'll have the commands (updated) for the version of PSTNgw you are using. This can also be used to redirect the screen output as the PSTNgw program is running.

Even with the up-to-date commands from the help output, it is hard to determine what the syntax is for a basic set-up. After looking through the source code and researching the archives of the OpenH323 mail list, the syntax for using a LineJACK card (syntax for Linux is different) was discovered [4]:

```
-q serialNumberOfLocalLineJACKCard IPofRemotePCacceptingH323calls –i bindToIP
```

So to make a call to 128.227.120.35 from 128.227.120.25, use the command:

```
pstngw.exe -q 355c047e 128.227.120.35 -i 128.227.120.25
```

This will allow incoming PSTN calls to come into 128.227.120.25, and be transferred to 128.227.120.35 as a H.323 call. Using Notepad or other text editor, save your frequently used commands as a .bat file and you can simply double-click the .bat file to run the PSTNgw program. Be sure to save the .bat file in the save directory as the PSTNgw program.

The serial number of the LineJACK is found on the anti-static wrapper of the bag it came in, or on a white sticker on the card. If you still can't find it, go to Quicknet.com and look for help on their website.

2.4 OpenH323 library (version 1.11.4)

Project OpenH323 starts here. Whenever a program is developed, this is the library that is used. So all of the OpenH323 programs require this dll (dynamic link library), or the equivalent static library.

Looking at the source code, this is where functions for the LineJACK card and other VoIP cards (LID cards) are found. For example, the file

openh323/src/ixjwin32.cxx contains the functions for testing if there is a dial tone, if the phone is ringing, if PSTN disconnected, etc.

If something is not working for the PSTNgw, it might be because the library functions for the VoIP cards are incorrect or there is a driver mismatch.

2.5 PWLib library (version 1.4.8)

PWLib stands for Portable Windows Library. First started just to provide conversions between Linux and Windows, it is now used for other purposes such as providing a way to create and control processes in the C++ language. For example, whenever PProcess:create or the like is seen, look to this library for how the function is working.

Also a dll that is required for the OpenH323 programs to run. Instead of being a dynamic library, it also can be compiled as a static library but the appropriate changes to OpenH323 need to be made.

3 Three Phases for testing the Phonigator system and schedule

March 21, 2003

I. Test hardware. On site PC to PC call (no programming):
phone at PC1 - - LAN - - phone at PC2

1. The IP addresses in Windows 98 and the OpenH323 software were set up.
2. From PC1, PC2 was dialed using touch pad on the phone or the software.
3. Two phones connected to the computer through the POTS port. Most of the software available to do this.

March 28, 2003

II. Test PSTNgw. On site call to PC2 :
Remote call → PSTN - - PC1 - - LAN - - PC2 - - phone at PC2

1. The computer that was called (PC1) will pick up on the PSTN port.
2. Connect to PC2 through the LAN. Using OpenPhone on PC2, the phone connected to the POTS line on PC2 will be rung. This is an on-site call similar to Phase I.

Note: No 2nd number will be entered. PC1 will not prompt for a 2nd number.

April 11, 2003

III. Test the full system. Hop-off at PC2 :

Remote call → PSTN - - PC1 - - LAN - - PC2 - - PSTN → remote phone

1. The computer that was called (PC1) will pick up on the PSTN port.
2. PC1 prompts for 2nd number with simple dial tone or beeping.
3. Number recognized by PC1 and converted from DTMF to numbers stored as an integer, an array, or a string.
4. Connect to PC2 through the LAN. Note: No look-up table for find the appropriate PC corresponding to the 2nd number dialed will be used. But for a real system and a continuation of this project, an area code to IP look-up table is needed.
5. The data representing the 2nd number is sent to PC2.
6. "Hope off" from IP. PC2 calls the 2nd number through PSTN.

4 Results (from testing)

Phase I: local call PC1 - - LAN - - local phone on PC2

Status: Works
Means: LineJACK cards are good, LineJACK drivers were installed correctly, and PCs networked together properly.

Phase II: PSTN call in - - PC1 - - LAN - - PC2 - - local phone on PC2.
Status: Successful and complete

Phase III (final phase): PSTN call in - - PC1 - - LAN - - PC2 - - PSTN call out
Status: Successful and complete.

5 Planned steps for finding a solution and fixing the problem

Quickest and easiest way to fix the problem is listed first and then more difficult ways next and so forth.

- Email the OpenH323 mailing list
- Research the internet for examples
- Determine if the correct LineJACK driver from Quicknet was installed

Problem could be a driver and OpenH323 library interaction problem. Maybe the c program was developed for an older driver and does not work with the newer LineJACK driver from Quicknet's website. See if anyone on the OpenH323 mailing list has a LineJACK driver that works with the PSTNgw program.
No drivers for LineJACK in Windows 98 are available from the OpenH323 website and searching on the internet for a current driver has been unsuccessful.

- Test PSTNgw with a incomnig H.323 call from OpenPhone or OhPhone

When the system was tested, the team never thought to invoke an H.323 from OpenPhone to the PSTNgw and thus working the system the other way around. This would be picking up where the big red 'X' left off and testing PSTNgw to see if accepts H.323 calls from another program. Of course, the PSTNgw program is still in need of programming since it cannot transfer a H.323 call. But this could help in seeing if it is failing at two points (transferring and accepting) or just one point (transferring).

- Under VC++, compile PWLib, OpenH323, and PSTNgw with no modifications

Just making the source files compile is a project in itself. Two auxiliary programs are need when compiling, one called Bison. If compiling the libraries, it may be wise to switch to "static" libraries since multiple versions will be made.

- Compile with modifications under VC++
- Test on Linux with lastest LineJACK driver

Driver for the LineJACK card under Linux is available on the OpenH323 and is most likely more current or more interoperable with the OpenH323 programs.

6 Testing phase

Testing was done in four different phases to ensure that a stable system had been VoIP system had been developed successfully. Each of the four different stages of testing are described clearly in this section.

6.1 Testing at the Communications Lab - H.323 call to PSTNgw

From the list of alternatives for fixing the problem, this was the most promising. The program OpenPhone would make a H.323 call to the other computer running PSTNgw that would dial out on the PSTN. Essentially, this method is testing the system in from the middle and working out to the PSTN [fig 2]. If it were a bridge, it would be like starting tof build the bridge from the middle above the water and working to land.

OpenPhone could transfer a H.323 call to the PSTNgw program. PSTNgw would accept the call and allow a phone number to be dialed out. This means that the PSTNgw can receive a H.323 call but cannot generate one.

6.1.2 Theory why PSTNgw call fails on incoming PSTN calls – central phone system

From looking at the trace for the failed system of Phase III, the phone system at the University of Florida (UF) appeared to be a problem. At the lab, the phone line that the computer is attached to does have a 7-digit outside number, but it is a switched phone system because when dialing out, you have to press "9" to get an outside line. PSTNgw works fine for calling out; probably because the switching function of the UF phone network has already taken place after "9" is pressed. But for incoming calls, the switching (or signaling) takes place seconds after the PSTNgw program picks up the call.

According to a professor, UF has a Centrex phone

system. All of the switching functions are done off-site at the telephone company BellSouth. Looking into Centrex phone systems, the Direct Inward Dialing (DID) feature may be the cause of the problem. DID means someone calling the university can simple enter a 7-digit number and reach a certain department, instead of dialing a main number and entering an extension. Once the call is connected, it may be trying to off-load the circuits through circuit switching or software switching. Or it is possible the Centrex phone system constantly sends signaling for control of the phone lines that is causing interference. More information was sought on UF's telephone infrastructure, but a technician in UF's Telecomm Service Department could not be reached.

Another possibility is the LineJACK cards are too sensitive. The switching of incoming telephone calls might be causing it to detect a false "wink." Basically, a "wink" is signal detected by the PSTNgw program when a remote caller hangs up. Or the cause could be from improper programming of the LineJACK's driver. The conclusion was to test the system at a place where no need to press "9" to get an outside line, for instance, at any house or ordinary apartment.

6.2 Testing the system at an apartment Theory

6.2.1 Testing PSTNgw to OpenPhone

After setting up the system of two PCs, two monitors and hub at the apartment, the theory of the UF switching problem was ready to be tested. At first, PSTNgw seem to have the same apparent problem as in the Lab, saying "failed at transferring call" but this was because of a networking problem. An Ethernet cable had accidentally been left in the "uplink" port of the hub. It was left there from when the system was connected to UF's network. Since all of the downloads and updates have been done, we won't be using the Internet anymore so no need for an uplink. Even though the system had to be moved, using the Lab was helpful because of the high-speed Internet. From the Lab, we downloaded all the necessary OpenH323 programs, and updated Windows 98 in a matter of minutes.

6.2.2 Results of this testing

Once the Ethernet cable was plugged into a valid port on the hub, the PCs where networked properly and the PSTNgw call to OpenPhone was successful. All present were very happy to see Phase II working.

6.3 Testing at a house with two phone lines

6.3.1 Testing PSTNgw to PSTNgw

For notation, the names jack 1 and jack 2 are used as names for the two phone lines. Testing for the first night, the whole group of five was present. Unfortunately, we spent most of the night watching the black MS-DOS screen full of white text telling us of failed calls. Unlike a cheering on a sports team, the computers do not play better even if fans are present

6.3.2 Results – fail to dial out 90% of the time

On the first night, only 2 successful phone calls where made in over a couple hours of testing. On a second night of testing, 3 successful calls were made. The success rate of 10% is unacceptable.

6.3.3 Testing OpenPhone to PSTNgw

Using OpenPhone to spawn a H.323 call to the PSTNgw program, it only work about 60% for dialing out on jack 2 and about 30% for jack 1.

6.3.4 Results – also fails to dial out

When the OpenPhone works, so does the PSTNgw to PSTNgw so it seems the interference fades and a window opens up so a streak of successful calls can be made.

If it is the phone line, and not the PSTNgw program, then using OpenPhone shouldn't work either. OpenPhone didn't work, so it is the phone lines. At the apartment, OpenPhone always worked.

6.3.5 Conclusion – PSTNgw works, but not with AM interface

Having shown that the phone line is causing the problem, a new place would be need for testing the VoIP system. Having the OpenPhone program fail was good news and gave hope that the system would work eventually.

Jack 2 seems to accept phone calls better, recognize digits pressed for the second number better, and make outgoing calls better than jack 1. Comparing the noise (sounds like cross-talk), jack 2 has less than jack 1. To find out if there is noise on the phone line, press a phone button and listen. Wanting to figure out more about this background noise heard on the line we turned on a radio, turned the knob slowly, listened and found that it is an AM station, 1230 kHz. This channel is ESPN Radio, guys talking about sports, and we have coined the phrase "sports-talk cross-talk." There was another AM radio talk show, female voices, on the lines but it was too faint to distinguish.

The LineJACKs probably do not have a filter for AM stations and are sensitive to this noise. This could be comparable to how the powerline networking technology, HomePlug, is sensitive to hair dryer or a vacuum cleaner. The interference wipes out the DTMF signal and also detects a false “wink” when dialing out. Both in Gainesville, the house is located only 10 miles NW from the apartment. There is no interference at the apartment. Maybe the layout of the phone lines at the house create act as an antenna tuned for these AM frequencies.

6.4 Testing at apartment with two phone lines

An additional phone line was installed at the apartment. Both of the phone lines do not have any noise detectable by the human ear.

6.4.1 Testing PSTNgw to PSTNgw

A cell phone was used to make calls into the apartment. For the second number, only 10-digits (e.g. 333 432-1111) can be entered so only local calls could be made.

6.4.2 Results – success

If the proper steps are followed for using the program, both instances of PSTNgw will remain stable on the two PCs and calls can be both ways. This is good to see it is truly bidirectional.

Even though the person called could barely hear the caller, the person called could be heard well. Maybe a PSTNgw options for volume control exists, but we do not know of one. Maybe the echo cancellation option has something to do with the volume being soft.

Even though the person called could barely hear the caller, the person called could be heard well. Maybe a PSTNgw options for volume control exists, but we do not know of one. Maybe the echo cancellation option has something to do with the volume being soft.

7. Future Plans

This project was completed successfully setting up a base where a commercially viable VoIP system could be developed in the future. Here are some guidelines which would help take this project to the next level.

7.1 Use this VoIP system to test different codecs

The LineJACK card has many different voice compression codecs that can be used. Testing different codecs the quality vs. bandwidth can be compared on a live system.

7.2 Purchase credit card and test internet switchboard

A system available from Quicknet, an account can be set up so an H.323 call can be routed to just about anywhere in the world. Using the LineJACK, PhoneJACK, or even a microphone and speakers, H.323 calls can be placed to this server program and routed to different countries. Like a phone card,

minutes are purchased but at lower rates than a phone company.

7.3 Converting to a SIP-based system

The system uses SIP as the signaling protocol for the setup and tear-down of calls (over the IP network). Calls received from and transmitted to the PSTN are subjected to translation from H.323 to SIP and SIP to H.323 signaling respectively. The actual conversation happens over a RTP (Real-time Transport Protocol) path in the network, between the two users.

Each SIP UA is also configured as a H.323 terminal to accept calls from or return calls to the PSTN via the LineJack. Each terminal, as mentioned above runs a H.323-SIP translator or gateway application, and also a H.323-based PSTN gateway application. Open Source applications like "siph323csgw" for the SIP-H.323 gateway and "PSTNgw" for the PSTN-H.323 gateway may be used.

A call is initiated when a user dials the number of the H.323 terminal. The H.323-PSTN gateway server application prompts the user to enter the phone number of the call's destination. When the user enters the same, the H.323-SIP gateway application translates the information into SIP format and the SIP UA1 sends out and SIP request (INVITE) intended for UA2, to the RS. The RS sends the directory information about UA2 to UA1 through another SIP message, and using this information, UA1 generates an SIP call to (the SIP URL of) UA2. From here on a reversal of the process described above causes the user at UA2/H.323 terminal 2 to receive the call from UA1, and when it is accepted, the call is setup.

This discussion only describes one possible way of implementing our system, which still needs to be tested. There may be other methods, depending on other needs. Regardless, the incorporation of SIP signaling in our system would make it more efficient and versatile while keeping all the features that we have on it currently

8. Conclusion

As stated originally, the goal of this project was to develop a successful telephone-to-telephone VoIP model, that could be developed in the future into a commercial venture. This project team has successfully built the desired model, and future teams can now take this once step further to make ita commercial success.

9. Acknowledgements

The team acknowledges the efforts and contribution of Dr. Haniph Latchman in providing the necessary infrastructure, lab facilities, and guidance. The group is thankful to Karthik Tripathi for making himself available to the group for assistance with programming-related issues.

References

1. http://www.tldp.org/HOWTO/VoIP-HOWTO-5.html#ss5.1
2. http://www.linuxjournal.com/article.php?sid=5941
3. http://www.voicetronix.com
4. http://ww.quicknet.net
5. http://www.vovida.org/