Session Initiation Protocol

KK Tan and HL Goh

Department of Electrical and Computer Engineering,
National University of Singapore
4, Engineering Drive 3, Singapore 117576

Abstract

Session Initiation Protocol (SIP) is a new and emerging protocol that is used to establish and release the connection between two end systems. It is used in preference to the older H323 protocol. Both protocols provide a similar set of services but SIP is much simpler because it has less logical components. This paper describes the implementation of a VoIP application-using SIP as the handshaking protocol. The implementation of the SIP is performed with the aid of

Libsip++ library from Columbia University. The audio signal is transmitted via the Microsoft's windows API. The Vo IP system was created using Delphi 5.0. As the Lipsip++ was created using Microsoft's C++ 6.0, some incompatibility problem arises. There will be a section discussing these problems and it's solution. The VoIP system consists of a SIP server and two user agent. The SIP server is linked to a HTTP server to facilitate the update of user configuration via Internet. This paper will also introduce a new function called chatboard that will facilitate users with speech handicap.

Keywords: SIP, H.323, VoIP, IP, HTTP, Tcl, TCP/IP

1. INTRODUCTION

Internet Protocol or IP is rapidly gaining ground as an alternative to the traditional audio and video transport methods used for telephony today. Voice over IP or VoIP has emerged as the tag line for transmission of voice or video over IP-based data networks. In addition, the world of VoIP promises to take users beyond the telephone call of today by adding multimedia conference calling, personal mobility, World Wide Web based click to call and other such advanced applications. The ITU-T H.323 protocol suite is the dominant VoIP protocol suite as measured by the number of commercially available products. The H.323 protocol suite is also the dominant VoIP protocol suite as measured by the size and complexity of the specifications. The use of H.323 results in a steep learning curve, high cost of implementation, high connection setup latency, and difficulty in achieving interoperability in heterogeneous networks. Although the ITU-T H.323 protocol suite currently dominates the VoIP world, there exists a lightweight contender for call signaling that avoids all the complexity, high connection setup latency, and implementation difficulties of H.323. The Session Initiation Protocol or SIP brings simplicity, familiarity, and clarity of purpose to IP telephony that Internet savvy network professionals will appreciate. SIP-based products are on the market now and more are under development.Currently, Microsoft's new operating system, XP, will ship with a SIP User Agent, and the company's next generation of instant messaging clients will also be based on SIP. Been the most widely used operating system; Microsoft decision to use SIP will ensure its proliferation in VoIP.

This paper will serve to provide a foundation on how a SIP based VoIP system can be created.

2. SIP ARCHITECTURE

SIP provides the basic elements of telephony: call setup and termination, call configuration, and data transfer. This is accomplished using SIP for call setup and termination portion, SDP to describe call configuration, and RTP for data transfer. RTCP is also used for data stream management. SIP can run over any datagram or stream protocol such as UDP, TCP, ATM, and frame relay. SIP is commonly run over TCP/IP because of inexpensive widespread connectivity, directory services, naming services, and a widely known development environment. The audio and video data streams are transported using the Real-time Transport Protocol (RTP) over UDP. SIP call signaling messages can be carried over UDP or TCP, with UDP being the preferred method because of its better performance and scalability. One important consideration when using SIP over UDP is that the entire message should fit within a single packet. If a SIP message is fragmented into multiple datagrams, the probability of losing the entire message increases with the number of fragments. When SIP messages are being transmitted over a WAN, the retransmissions that result due to lost fragments can seriously degrade call-signaling performance. The default port for SIP is 5060 although any available user port may be used. The port to be used for RTP/RTCP is specified in SIP call signaling messages. Figure 1 shows SIP over TCP/IP.

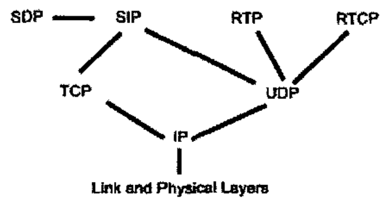

Fig.1 Communication Layer

3. LIBSIP++ INTEGRATION

There are several interaction problems linked to the re use of component. Major problems arise from the fact that the main program for the VoIP application was developed in Borland complier, while the Libsip++ was developed in Microsoft visual c++. The object and module format generated in visual c++ is different from the format used by Borland complier. Below will elaborate these problems.

3.1..Coff and OMF

Until the first Microsoft Win32 programming tools appeared, almost all compilers produced OBJ files in the Intel OMF format. The original Windows worked with an object module format known as Common Object File Format (COFF), which is the official machine-code format for UNIX System V. COFF is relatively easy to work with. COFF format OBJs also have the advantage of being much closer in format to Portable Executable files, the native executable format for Win32. A COFF-format linker should have much less work to create an EXE or DLL from a COFF file than from an Intel OMF-format file.

Additional information in special records of the LIB file lets the linker quickly. Borland use OMF-style OBJs and LIBs, while Microsoft's 32-bit compilers produce COFF-format OBJs and LIBs.

Borland's TLINK refuses to work with COFF-format OBJ or LIB files. This result in problems when try to load the Libsip++ library into Delphi 5.0.

3.2 _St dcall and _Cdecl

Borland and Microsoft also disagree on linker naming conventions. Every function in a DLL or OBJ has a linker name. The linker uses the linker name to resolve functions that were prototyped during compile time. The linker will generate an unresolved external error if it can't find a function with a linker name that it thinks is needed by the program.

With regard to linker function names, Borland and Microsoft disagree on these points:

1. Visual C++ sometimes decorates exported __stdcall functions.
2. Borland expects imported __cdecl functions to be decorated.

For example a DLL created with Visual C++ that contains a __stdcall function called MyFunction(), Visual C++ will give the function a linker name that looks like _MyFunction@4. When the Borland linker tries to resolve calls made to this function, it expects to find a function with the name MyFunction. Since the import library for the Visual C++ DLL doesn't contain a function called MyFunction, the Borland linker reports an unre solved external, which means it couldn't find the function.

3.3 Wrapper

To solve the format incompatible problem, 2 utility functions IMPLIB and TDUMP in the Borland C++ builder 6 to create was used to create a wrapper that can provide a common interface between the Libsip++ component and the rest of the program.

The IMPLIB utility was used to create the import lib of the libsip++ objects. After that TDUMP.exe is then employed to obtained the structure and the convention used in the Libsip++. Using the IMPORT and EXPORT command, a wrapper interface that can translate the conventions used in the libsip++ to be Borland compliance was created.

4. TRANSMITTING AUDIO SIGNAL

After the SIP user agents complete the handshaking negotiation using SIP protocol, the user agent will then be ready to transmit the audio signal. The recording and play back of audio signal is performed using windows audio DLL. The user agent is able to transmit the audio signal in a duplex manner this is made possible by multi-threading.

To record the audio signal from the microphone and play back the audio signal received some DLLs that are inherent in the Microsoft operating system was used. In the windows 98, multimedia-related services can be found from the WINMM.DLL. In the Delphi, the RTL unit named MMSYSTEM is the key. For example, to open an output device, the WaveOutopen function will be called; to open an input device, Wavelnopen function will be called.

To play audio, the first thing is to open a handle to the output device desired (just as a file must be opened before it can be accessed). For output, it is really not important which device is the best (the user's computer can have several sound device installed).

Playing a buffer of the audio is done using the waveoutwrite function. The format of the buffer used in the program is a plain and simple PCM. Before getting a buffer playing with waveoutwrite a few things need to be done. First, a header structure needs to be set up. The header describes the location of the buffer storing the actual audio data, as well as how many times it should be looped. After the header has been set up, the audio driver is instructed to stand by for the data. This is done with the waveoutprepareheader function.

After Header preparation, the buffer is ready for playing. After it has finished playing, the header must be unprepared. With the header preparation and un-prepare requirements, audio playing is actually a repetition of the three calls: prepare, write, and unprepared.

The exchange of audio data between the 2user agents is through the TCP/IP layer. This can be performed easily in Delphi. Using the TTCP control found on the Internet page of the component palette. The code calls the Listen method and waits for the Connection Request event. In the event handler for the Data Arrival event is set. Calling the listen method should put the component in the listening state for new connections. However, if the state property is not queried before doing this, connections are simply refused.

The Data Arrival event is the most important -and the most difficult event to handle. This event handler reads the data from the TCP connection into an OleVariant, and then stores the raw binary data in either of the two output buffers created using Tthread class. The threaded buffer will then play one at a time, using the default audio output device set up using Windows control Panel

5. SIP SERVER

A SIP proxy server that receives a SIP request from a user agent acts on behalf of the user agent in forwarding or responding to the request. A proxy server typically has access to a database or a location service to aid it in processing the request (determining the next hop). The interface between the proxy and the location service is not defined by the SIP protocol. A proxy can use any number of types of databases to aid in processing a request. Databases could contain SIP registrations, or any other type of information about where a user is located. A proxy server is different from a user agent or gateway in two key ways:

1. A proxy server does not issue a request; it only responds to requests from a user agent. (A Cancel request is the only exception to this rule.)

2. A proxy server has no media capabilities.

The figure 2 below shows the role of a SIP proxy server in a transaction.

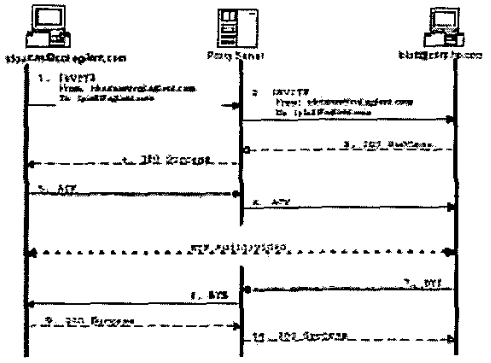

Fig 2 SIP operations with Proxy

5.1 SIP Server Implementation

A SIP server with the capability to serve as a proxy server for an incoming request call was created. Upon the received of an incoming request it will look up in the database for the identification of the callee. Depending on the policy set by the callee, the server will react accordingly. If the callee had set the policy to reject all call or did not leave a contactable address, the caller will received a message that the callee is not available. On the other hand if the callee set the policy of accepting all call the caller is automatically connected to the callee terminal. The server that was created is a stateful one, at anytime the administrators are able to check the transaction that had been made via the server. In order, to allow the user to change their configuration easily, a HTTP server was implemented to handle all the changing of the configuration through FORM.

5.2 HTTP FORM

The FORM is created using Tool command Language (Tcl) rather than the conventional scripting language like ASP and JavaScript. The main advantage for doing so is the portability of TCL, it can run on platforms as varied as VAXen and Macs, as well as almost all flavors of Unix. It also allows for its library to be extended by the programmer, either in C or in Tcl. This will allow porting of the HTTP system to a Linux operating system (OS) if necessary (presently Windows NT was used as the OS for the HTTP server). Tcl's simple structure of "command argument" and strong string and list processing capabilities make it easy to perform complex parsing operations and string manipulation that is often encountered in HTTP FORM.

6. CHATBOARD

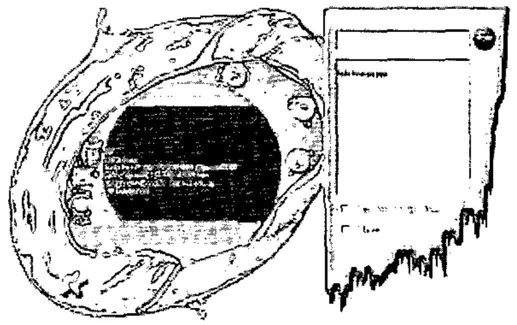

Fig 3 Chatboard

The figure 3 above show the VoIP application with chatboard function activated. There are two main reasons for this additional feature. One of the reasons is to provide a form of communication for people who had speech handicap. For user who is unable to speak, he can activate the chatboard, which function like an Internet Relay Chat (IRC). Upon the activation of the chatboard the audio communication will be suspended, the user will then key in the message they wish to convey on the chatboard. The message will be transmitted to the other end in the form of text string. At the receiving end, a text to speech procedure will be perform on the received text string to convert the text to a computerize speech. To relief the user from the chore of typing, the chatboard contain speech recognition ability. When the speech recognition is on, the user just need to speak into the microphone and text will appear on the chatboard.

The second reason for having the chatboard is that it can be used when the Internet traffic is heavy and the Quality of sound of the transmitted audio is affected. As the audio packet is large, in the event of heavy Internet traffic the audio stream maybe delayed or jitter may appear in the transmission. The using of chatboard on the other hand require much less bandwidth and the transmission of text will remain instantaneous, this is due to the small packet size of the text string. With the text -To-speech function there will not be much difference in term of user experience as compare to audio communication.

Figure 4 show the actions carrie d out by the Callee when there is an incoming call. Appropriate SIP responses will be sent via a DLL created using Libsip++ and once the call is established, audio transmission will be carried out. The user will be given a choice of whether to carry the communication using the default audio mode or change to chatboard.

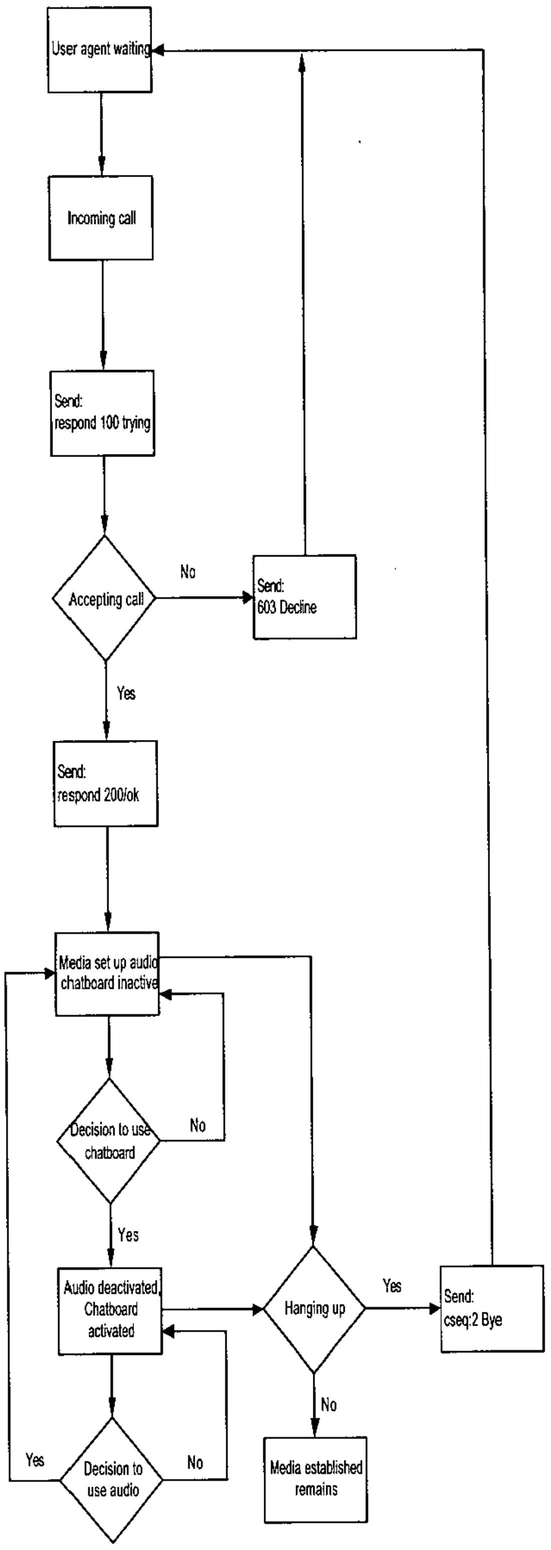

Fig 4 Callee flowchart

7. CONCLUSION

IP telephony is becoming the new leader in the telephony world. Day by day the number of companies interested in IP telephony is continuously increasing and new attractive services are in progress. Users demand new facilities and services that must be provided by the companies using the best tools to achieve their objectives. One of the ways to improve the ease of development of IP telephony application is to use a simpler handshaking protocol.

With the beginning of the new millennium, the digital industry becomes more and more integrated; it is common for two seemingly unrelated technologies to be combining into a new innovation. This serves to be a motivation for future work.

Integration of VoIP, Speech recognition with Bluetooth will provide a new way for people to remotely control their home appliances. In the future, air-conditioner can be switched, oven can be turned on simply by calling home. And this future is not far, with the SIP architecture ready; we are now in the position to create a VoIP system that is connected to the Public Switched Network (PSTN). And with a SIP/PSTN gateway, analog audio voice can be digitized and transmit the audio signal to the Bluetooth's controller at home. The Bluetooth's controller will in turn perform a speech recognition procedure on the received audio and then execute the command to turn in the appliances accordingly.

Hopefully, this paper is able to provide an idea on how to implements a SIP based VoIP application and also some idea on future SIP work.

8. ACKNOWLEDGMENT

I would like to take this opportunity to thank Prof Tan Kok Kiong and Dr Loh Wai Lung for giving me the chance to work on this project. I would also like to thank the people of Mechatronic and Automation laboratory who have lent help and support to this project.

I am grateful to Columbia University, NY, for granting the permission to use the SIP library in this project.

REFERENCES

[1] Danji Yakimovich,James M.Bieman,Victor R.Basili "Software architecture classification for estimating the cost of COTS integration" Page297-Page299 (1999).

[2] Ian Sommervile, "Software Engineering", Addison-Wesley, 2001

[3] Alan B Johnston "Understanding the Session Initiation Protocol", Artech House, Boston, (2000).

[4] M. Handley, H. Schulzrinne, E. Schooler and J. Rosenberg, "SIP: Session Initiation Protocol," Request for Comments 2543, Internet Engineering TaskForce, Mar. 1999.

[5] J. Rosenberg and H. Schulzrinne, "The Session Initiation Protocol: Providing Advanced Telephony Services across the Internet," *Bell Labs Technical Journal*, Vol. 3, No. 4, Oct/Dec. 1998, pp. 144-160.

[6] Don Libes "Writing CGI scripts in Tcl" (1987)

[7] J. Lennox, J. Rosenberg and H. Schulzrinne, "Common Gateway Interface
for SIP," Internet Draft, Internet Engineering Task Force, May 1999.