Architecture & Implementation of SIP-controlled IP→PSTN Gateway

Marcin Michalak

mmich@iris.ics.agh.edu.pl
Department of Computer Science, University of Mining & Metallurgy
Al. Mickiewicza 30, Kraków, Poland

1. Introduction

IP Telephony gateways, which enable using low-cost Internet connections for long-distance or international calls gained much popularity and attention during last few years. First such gateway has been presented on Networks Expo in Boston, 26.02.1996 and it was 'Vocaltec Telephony Gateway' using Dialogic cards [1]. Since then most big network-oriented companies created their own solutions.

The aim of this article is to present design and experience gathered during building and implementing a SIP-controlled IP→PSTN gateway, called *sip-telip*. The paper contains information about hardware and software architecture of the proposed solution along with some general information about IP Telephony and gateways.

The structure of the papers is as follows. Section 2 specifies the gateway architecture along with description of software and hardware used for its implementation. Sound processing path is described in Section 3 in the form of consecutive blocks of sound processing between IP and PSTN networks. Next in Section 4 translation of signals like ringing, dialling and hanging up between two environments, which is done by SIP protocol, is described. The *sip-telip* activity example of the call processing is presented in Section 5. In Section 6 the performance issues and information about the quality of connection and problems which are encountered in IP Telephony, like latency, sound quality and bandwidth demand are briefly discussed. The paper is ended with a short summary and future development plans.

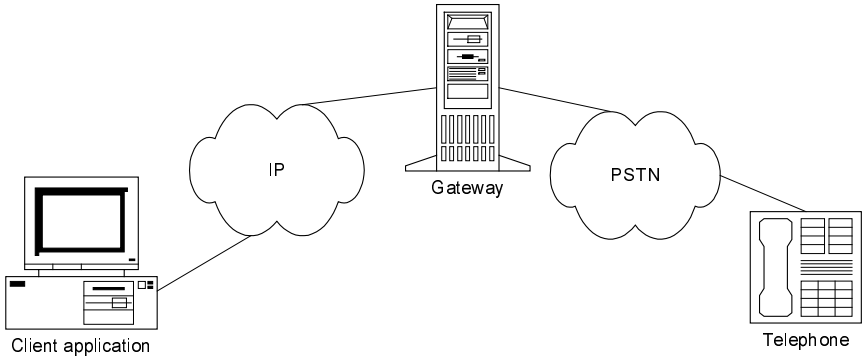

Figure 1. IP Telephony gateway

2. Architecture

The *sip-telip* application scenario is shown in Fig. 1. The gateway allows communication between people using two tools: computer and a telephone. Computer must be equipped with headphones, a microphone and a full-duplex audio card. From the software side any audioconferencing software could be used which provides interoperability with the gateway. On the other side any telephone set may be used.

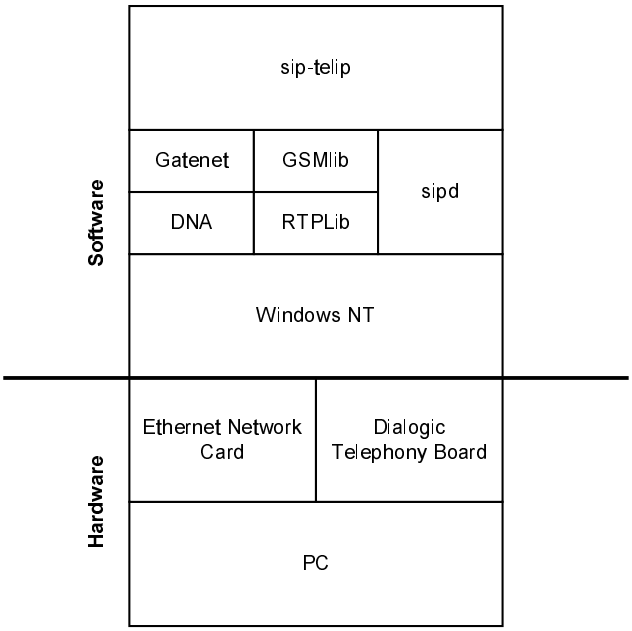

Figure 2. Hardware and software structure of the gateway

The gateway internal architecture is presented in Fig.2. It consists of two main elements: hardware and software.

2.1 Hardware architecture

In order to translate voice from Internet to either analogue or digital phone line we need a special device, such as PC, router or PBX card. This card (or a set of cards) may combine all the features needed by the gateway, i.e. all translations are done by hardware. On the other end we have simple cards which only provide access to digitised voice in computer memory and allow sending data (voice samples) from memory to the telephone network. These samples must be later processed by computer and sent to Internet via network connection (usually network card). *sip-telip* uses D/41ESC telephony card from Dialogic, which has following features:

- 4 external ports (to phone lines),
- 4 analog/digital channels, each with specialised DSP,
- SC-BUS for connecting other cards/devices,
- possibility to connect 2 channels to one port, making it full-duplex,
- A-law and μ-law sound codecs,

- echo cancellation,
- silence suppression,
- low latency.

This card comes along with SDK (called **DNA**) and a special package for IP Telephony, which is called **Gatenet.** Since the solution bases on simple telephony card, it needs a computer with Internet connection to process and transfer data to/from network. In this case it's a PC with Windows NT and standard Ethernet card with Internet connection.

2.2 Software architecture

sip-telip, as stated above, uses very simple hardware and thus all processing must be done by software. There are several components that are needed to create such gateway, which include:

- operating system – this is Windows NT 4.0.
- software creation environment – Microsoft Visual C++ 6.0.
- SDK for the telephony card – Dialogic DNA 3.1 & Gatenet 3.0 Package.
- library for signalling – sipd from University of Columbia.
- library for sending & receiving voice over Internet – RTPlib from elemedia.
- library for processing sound (optional) – GSM codec from TU Berlin.

Windows NT has been chosen because Dialogic SDK and Gatenet package is available only for this platform. MS VC++ has been chosen for the same reason, because libraries from Dialogic work best with this environment. The choice of the software components, which form the gateway, has very substantial influence on its functionality and performance. The most important features of the selected software modules are summarised below.

Dialogic SDK (DNA)

Dialogic SDK [7] is a powerful set of functions allowing a programmer to create very complicated Computer Telephony applications, like call centers, voicemail boxes and also IP telephony gateways. Most functions may be called either synchronously or asynchronously thus allowing the creation of multithreaded applications. Basic functions include:

- dx_open(), dx_close() – open/close board/channels,
- dx_sethook() – set the line on/off-hook,
- dx_play(), dx_record() – 'play' with voice samples, i.e. greetings, messages etc.,
- dx_dial() – dial the number,
- dx_getdig() – get the DTMF tones entered by the party.

The Gatenet 3.0 Package

This package [8] is a special set of functions developed to allow the creation of IP Telephony Gateways. It's a firmware for the card and a software library with special functions.
The firmware provides:

- automatic routing of channels 1-3 and 2-4 for full duplex capability,
- possibility of manually setting the buffer size to as little as 120 bytes (512 by default), thus lowering latency,
- Play/Record Direct Streaming technique (PRDS) ,
- hardware echo cancellation,
- silence suppression.

The most important features are lowering the latency and providing echo cancellation.

sipd from University of Columbia

sipd [2] is a sip proxy daemon written in 'structured C'. Its sources derive from rtsp daemon, HTTP & RFC822 parser. That's why the code is very stable and reliable. It's also very easily understandable and transparent because of lots of accurate and precise comments.

RTPlib from elemedia

RTPlib from elemedia [5] has been chosen because it's a very easy to use and complete RTP library. It's available for Win32, Linux & Solaris and is written in C. This API is very easy and is very thoroughly described in the documentation (html file) and accompanied by examples.

GSM codec from TU Berlin

GSM codec [9] is a library (with source) with functions allowing compression & decompression from linear PCM to 13.2 kbps GSM. It's also written in C and available on most system platforms.

2.3. Gateway threads functionality

sip-telip is written as a multithreaded Win32 console application and consists of several threads. These threads include:

1) Main thread and ProcessEvent()
   Used for initialisation, event collection and handling, moving between application states.
2) InThread
   Receives both RTP and RTCP packets. If this is a RTCP packet, it's checked against being a BYE packet. If it is a BYE packet, the call is terminated.
   If this is a RTP packet, included sound sample is decoded (if needed) to μ-law PCM format and sent to the Dialogic board. Currently μ-law PCM (0) and GSM(3) payloads are supported. Other codecs, such as DVI, A-law PCM and others will be implemented in the near future.
3) OutThread
   Waits for the notification (event) from the Dialogic board that the sound sample is ready, takes the sample, encodes it (if needed) and sends it inside the RTP packet.
4) RTPScheduleThread
   Required by the elemedia RTP library. It is used to schedule & execute RTP events. These events are stored in the queue each having it's own execution time. When this time comes, internal library function RTPExecute() is called with context kept in the queue element.
   Scheduling is used for sending RTCP packets holding reports at a certain time.
5) UDPListenThread
   Listens to packets coming from the net on a certain UDP port, which is usually 5060 (default SIP listen port). The body is almost directly taken from the sipd code, since the idea of operation is the same: receive the packet, create a SIP message out of it and pass it to new, dynamically created either RequestProcessThread or ResponseProcessThread.

6) RequestProcessThread

This thread is created dynamically in UDPListenThread whenever new packet with request is received. Its function is to parse the request and perform certain operations according to the contents.

3. Sound processing path

Since the gateway operation is about translating voice most of the time this process shall be described here. Figure 3 shows subsequent blocks of sound processing path. This path contains two working in parallel stream processing flows: from IP to PSTN and in opposite direction.

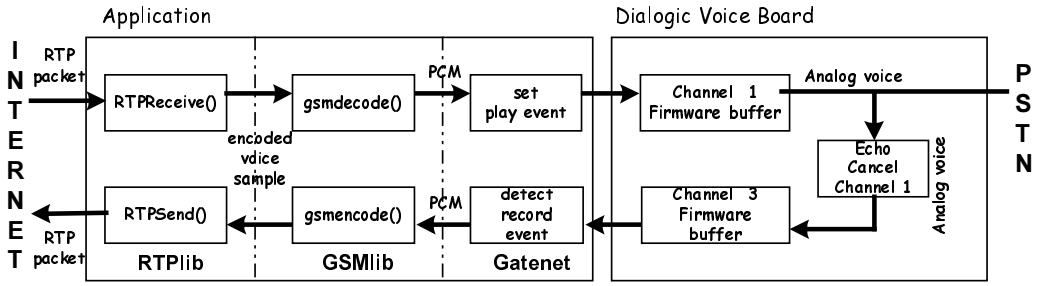

Figure 3. Sound processing path

3.1 Flow from IP to PSTN.

All operations described here are handled by InThread. They include:

- waiting for the packet to arrive (by select()),
- calling RTPReceive() – library function for receiving RTP packet on UDP port,
- calling gsmdecode(), which coverts received voice sample from GSM to μ-law PCM,
- copying received sample to firmware buffer and notifying Gatenet library that sample is ready by setting play event.

After this, sample is taken from the firmware buffer, converted to analog voice and played on the telephone line. These operations take place in the hardware (right big block in the figure).

3.2 Flow from PSTN to IP.

These operations are handled by OutThread. They include:

- waiting for the record event (set by Gatenet when the sample is ready),
- encoding sample from μ-law PCM to GSM,
- sending RTP packet,
- setting current firmware buffer as invalid.

Decoding from/to GSM is optional here and may be negotiated, since μ-law PCM is usually a basic codec for RTP transmission (it's payload number is 0).

4. Signalling: SIP and its functions

Session Initiation Protocol, proposed by IETF and described in RFC 2543 [3] according to this document is:

"an application-layer control (signalling) protocol for creating, modifying and terminating sessions with one or more participants. These sessions include Internet multimedia conferences, Internet telephone calls and multimedia distribution. Members in a session can communicate via multicast or via a mesh of unicast relations, or a combination of these."

SIP has been created as an alternative to H.323 stack in order to simplify call establishment and provide architecture for developing various types of services. It is also very important that a user may have only one, universal address in the form of 'user@host', and be contacted under this address wherever he is. This is achieved by allowing the specification of several possible contact addresses, each with a different quality. User can also specify different addresses for different time of day/night. All these contact addresses are stored in the host called 'Registrar' and are invisible to the calling user till he gets connected.

Call negotiation is done by sending requests and receiving replies in the form of text packets sent usually via UDP. There are several types of requests:

INVITE	- request new call or change existing call parameters
ACK	- confirm that response has been received and processed
CANCEL	- cancel a pending request
BYE	- terminate a call
OPTIONS	- ask about user or gateway possibilities
REGISTER	- register within this computer.

Each request is given a response, which is assigned a number indicating success or error code. Basic code types are:

1XX	- provisional response. Sent when INVITE request is processed, but not yet completed.
2XX	- indicates success.
3XX	- redirection. Specified user is not available here and moved either permanently (301) or temporarily (302).
4XX	- failures, such as bad request (400), not found (404), etc.
5XX	- Server failure.
6XX	- Global failures.

Currently, in the *sip-telip* gateway only basic SIP functionality is used. The gateway acts as a User Agent Server and SIP is used for both call establishment and termination.
Termination of a call is also present when *sip-telip* receives RTCP BYE packet, since sending BYE packets in SIP applications is not obligatory.

5. Activity example

This paragraph presents the scenario of the normal gateway operation. Following lines describe call establishment and termination paths.
As in all programs, certain operations have to be completed at the beginning (initialisation) and at the end of operation. In *sip-telip*, initialisation includes:

- opening the Dialogic board,
- setting all channels on-hook,
- setting/getting the board/channel parameters,
- creation of the notification events (for getting & putting voice samples to the board),
- preparation of voice buffers,
- creation of UDPListenThread and RTPScheduleThread,
- creation of In & Out Threads (suspended).

After these actions call may be accepted and established.

The call establishment procedure contains following steps:
1) SIP INVITE packet is sent by the caller's application
This packet must have a telephone number instead of username in the `To:` field, like:

```
To: 6173982@telip.ics.agh.edu.pl
```

2) UDPListenThread:
- receives the SIP INVITE packet,
- creates new Request out of it,
- creates new RequestProcessThread with this request as the argument and activates it.

3) RequestProcessThread:
- parses the request,
- checks if it's complete, sane and if there's a telephone number in the To: field,
- checks if call is to be handled by the gateway (currently host in To: field),
- checks if the user part in To: contains a valid phone number,
- parses SDP parameters,
- sends SIP TRYING (100) packet,
- gives control back to main thread by setting an event.

4) Main thread (function ProcessEvent()):
- sets the phone channel off-hook,
- dials the number,
- sets up the RTP connection with parsed SDP parameters (host, port, media type),
- sends OK.

5) UDPListenThread receives SIP ACK packet
- RequestProcessThread parses it,
- ResponseSend is stopped by setting an event,
- control is given back to main thread.

6) Main thread
- resumes suspended InThread & OutThread,
- starts recording from and playing to the telephony board.

In this state application starts to send RTP packets containing voice samples. These voice samples are digitised by the telephony board (PSTN->IP) and played out to the telephone line (IP→PSTN).

Call is terminated when one of the following situations occurs:
- SIP BYE packet is received,
- RTCP BYE packet is received,

- telephone user has hang up,
- critical error has occurred (for example can't read/write from network/board).

After termination the gateway is ready to handle new calls. If during the established call somebody else wants to use the gateway he's sent a `503` response: `Service temporarily unavailable`.

6. Performance issues

Apart from reliability of service, which is important for all types of applications latency is the most important issue in IP Telephony. This paragraph will describe sources of latency and solutions for minimising it.

The delay in IP Telephony applications is brought in several stages. According to the research [4] the acceptable delay for the telephone conversation is up to 150ms. This is so-called 'one-way latency', which means the time of the sound transmission between speaker and the listener.

As stated above, sources of the latency can be grouped into several stages of processing incoming/outgoing sound. These are:

1. Telephone network interface latency

   This latency comes from the 'translation' between telephone and computer system. The sound is converted from telephone to digitised PCM frames sent further to the DSP. This process usually takes very little time (below 1ms) in comparison to other stages.

2. DSP latency, coding and decoding sound.

   In this stage sound (usually voice) is coded, tones and special signals are detected or generated. This entire process is called "vocoding". If sound needs to be compressed or transcoded to another format (eg. μ-law PCM to linear PCM) it must be copied from one memory location to another and this introduces delay. This delay varies from 15 to 30ms and depends on the codec settings (i.e. processing buffer size).

3. Buffering and packetisation.

   Buffering and packetisation introduces most application latency and this area gives most opportunities to reduce it. Here latency comes from a finite size of the sample buffer and need for copying it from one place in memory to another.

   Dialogic SDK provides two methods of voice streaming. One is User I/O functions (UIO), second is Play/Record Direct Streaming (PRDS). Basic differences between these methods are shown in Figure 4.

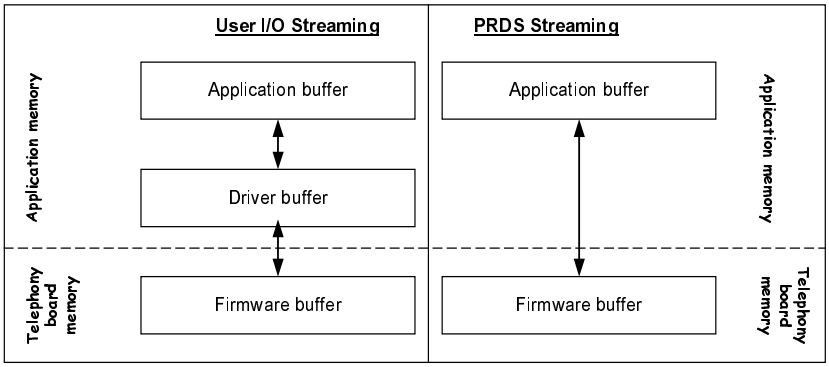

Figure 4. Differences between UIO and PRDS streaming.

The idea of UIO is to replace system read(), write() and seek() functions with new ones, which may copy buffers to/from different locations and notify the application about this. This method works very well but introduces much delay and that's why it can't be used in IP Telephony applications.

New method, Play/Record Direct Streaming (PRDS) is introduced in Gatenet package and is designed especially for IP Telephony applications. It provides direct access to firmware buffer and allows reducing it to as low as 120 bytes, which reduces latency to 82ms. This value is acceptable in this application.

	Firmware Buffer Size	Driver Buffer Size	Delay
UIO streaming	240 bytes	480	242ms
PRDS streaming	240 bytes	N/A	125ms
PRDS streaming	120 bytes	N/A	82ms

Table 1. Delay comparison for UIO and PRDS Streaming

After being prepared voice sample must be put into network packet and sent. Voice is usually sent in RTP packets over UDP and thus is preceded by a 40-byte header (20 bytes IP+ 8 bytes UDP + 12 bytes RTP). This introduces additional delay because the longer the packet, the longer the transmission. In case of a 13.2 kbps GSM codec a frame of 33 bytes is produced each 20ms. If we send each frame in a separate packet we get a 120% overhead. We can reduce the overhead by transmitting several frames in one packet but these frames must be buffered and this introduces additional delay. In order to minimise both effects a balance between buffering and packetisation must be found. In most applications packet size ranges from 150-300 bytes including header. This size also depends on network connection used, which must be entered at the beginning of program setup.

4. IP network latency.

This stage may introduce very significant latency, which can range from <1ms in local and fast networks up to more than 1000ms for slow international satellite links. It has to be considered in 'budgeting' latency but we usually have no influence on it. This depends on the end user location and may not be predicted in general. If we allow gateway access for users outside local network we can do nothing about it.

7. Summary

IP Telephony gateways are becoming very popular and are widely used, mostly because of flat rates for normally expensive, international calls. Gateways still need to be used to integrate old telephony network with new IP Telephony services. The described solution gives many application possibilities by using SIP as the signalling protocol. Apart from making regular calls the feature of user location in SIP may be used. User can specify that he's available e.g. on computer at work 9am-5pm, home phone from 5pm-10pm and busy in the remaining hours. This information will be stored in the location database and transparent to the caller, which will use one global address.

IP->PSTN gateway may also be used as a 'last resort', when user can't be located on any computer. He can be contacted then on the cellular phone.

During its operation *sip-telip* has been tested with SIPhone from Bell Labs [6] and proved to work perfectly with this application. It was possible to make a call from Sun Ultra workstation equipped with sound device and SIPhone running on Solaris to a telephone, which number has been specified in the To: field.

Tests with other SIP applications are also considered, but at the time of writing this article these were not yet publicly available.

Apart from further testing, there are also plans to add new features to the *sip-telip* gateway. These features include:

- support for new codecs, like DVI, A-law PCM and others,
- user location and address translation service, which would allow specifying address in 'normal' form, like mm@telip, which would later be translated to telephone number,
- registrar functionality.

References

[1] pulver.com & Jeff Pulver– source of all IP Telephony-related information.

[2] www.cs.columbia.edu/~hgs/sip – source of all sip-related information, home of sipd & sipc. List of sip-related documents, papers, conferences, implementations and much more.

[3] RCF 2543 – SIP.

[4] "Understanding Latency in IP Telephony" - Alan Percsy, Senior Sales Engineer, Brooktrout

[5] elemedia RTP API documentation - http://www.elemedia.com

[6] Bell Labs' SIPhone - http://www.bell-labs.com

[7] Dialogic DNA 3.1 Online Help

[8] Gatenet 3.0 package documentation

[9] GSM codec from TU Berlin - http://kbs.cs.tu-berlin.de/~jutta/toast.html