Royal Institute of Technology
Department of Microelectronics and Information Technology

VoIP SERVICE PROVIDER

Internet Telephony Service Provider using SIP Protocol.

Amos Muhunda Nungu
Stockholm, Sweden 2005.

Master of Science Thesis in Telecommunication
[IMIT/TSLab-2005-05]

Supervisor: **Hans Eriksson**
Examiner: **Prof. Björn Pehrson**

Abstract

Although Voice over IP (VoIP) also known as IP Telephony (IPT) has been in existence for many years, it has only recently begun to take off as a viable alternative to traditional public switched telephone networks (PSTN). Interest and acceptance has been driven by the attractive cost efficiencies that organizations can achieve by leveraging a single IP network to support both data and voice over their Internet or intranet.

The thesis explains the fundamentals of a VoIP Service Provider (based on SIP protocol), focusing on the functions and components that make a VoIP solution. Additional requirements for providing VoIP solution in organizational environment or as a business company, offering services to others. The thesis will answer the following questions: what is required for an organization or a company to deploy VoIP? What makes a VoIP system and how can one take advantage of it? Once a general understanding of VoIP is achieved, the 'Service Providers' are better prepared to tackle the more complex issues in the Internet protocol that go into deploying a secure, reliable and high performance VoIP network.

During the thesis work, I have created a package known as “ITSP-SIP” which aims at easy setup of VoIP covering the basic services. The package will be an open source, ongoing project hosted by IMIT-KTH.

Acknowledgments

I would like to thank everybody that has contributed to this project, sharing their knowledge and devoting some of their time to help me carry out this challenging task. I would like to especially thank the following people:

- Prof. Björn Pehrson: For giving me the opportunity to work on this project, and his guidance as my examiner. I would like to thank him especially on his suggestion that I should structure my report so that it can be read by different groups of people with different knowledge level and background.
- Hans Eriksson: For being around wherever I needed his help. I thank Hans for sharing his vast experience as a commercial VoIP operator. His knowledge on call detail record generation and billing in general was of great value. It was an honour to have him as my supervisor.
- Prof. Maguire: For his guidance on what the final package should contain. Even though I wasn't able to implement all what he suggested, still it was great to see things from his point of view. I wish I could implement all what he suggested to me.
- TSLab staff: For being around wherever I needed their support. Special thanks to Erik Eliasson, Johan Bilien and J-O for their readiness to assist and share their VoIP experiences with me. I held many discussions with Erik and Johan analyzing what was possible and good to be implemented. Johan guided me well with understand of Linux (Debian), to accomplish my job.
- My family: For their moral and material support, I couldn't make it to KTH in the first place without their endless support.
- Friends: For the good and bad moments we shared together, and they remained friends even when I abandoned them to concentrate on this project.

Table of Contents

Table of figures

Terminologies

- **KTH** - Kungliga Tekniska hogskolan [Royal Institute of Technology]
  This is the institute where the thesis work is carried out.
- **IMIT** – Department of Microelectronics and Information Technology
  A department at KTH, where the student is studying.
- **Tslab** - IMIT Telecommunication System Lab.
  The telecommunication laboratory at KTH where the actual work is performed.
- **SIP** - Session Initialization Protocol.
  A standard protocol for initiating an interactive user session that involves multimedia elements such as video, voice, chat, gaming, or virtual reality.
- **Internet Telephony**.
  Also known as IP Telephony, is a category of hardware and software that enables people to use the Internet as the transmission medium for telephone calls.
- **VoIP -** Voice over Internet Protocol.
  Another way of saying IP Telephony. It involves the transmission of telephone calls over a data network like the Internet and intranet.
- **SER** - SIP Express Router
  A free (open source) SIP server. It can act as SIP registrar, proxy, location and redirect server.
- **PSTN -** Public Switched Telephone Network.
  International telephone systems based on copper wires carrying analogy voice data.
- **NAT** - Network Address Translation
  The translation of an Internet Protocol address used within one network to a different IP address known within another network.
- **Firewall**
  Program or hardware device that filters the information coming through the Internet connection into your private network or computer system.
- **E.164**
  The international telephone numbering plan administered by the International Telecommunication Union (ITU), which specifies the format, structure, and administrative hierarchy of telephone numbers.
- **PBX – Private Branch eXchange.**
  A private telephone network used within an organization. Users of the PBX share a certain number of *outside lines* for making telephone calls external to the PBX.
- **Codec**
  Short for compressor/decompressor, a codec is any technology for compressing and decompressing data. Converting analogy video and audio signals into a digital format for transmission, then convert them back to analogy signals upon reaching their destination.
- **Asterisk**
  Asterisk is an Open Source telephony switching and private branch exchange running on Linux.
- **SEMS – SIP Express Media Server**
  Sems is an Open Source media server that function as a Voice mail, ISDN gateway, Conference and announcement server
- **CSD** – Communication Systems Design.
  Course offered by KTH which implements problem based learning driven by projects.
- **RTP** - **R**eal-Time **T**ransport **P**rotocol,
  Internet protocol for transmitting real-time data such as audio and video.

1 Introduction

Voice over Internet Protocol (VoIP) is a term used for voice being transported via data networks. The data network involved might be the Internet itself, or a corporate intranet, or managed networks used by local or long distance carriers and ISPs. VoIP is a technology that allows you to make telephone calls over data networks. No matter whether traditional telephony devices, PCs or dedicated terminals take part in the calls and no matter whether the calls are entirely or only partially transmitted over the Internet.

With VoIP, telephone calls are transmitted over the data network, eliminating long distance charges altogether. For users who have free, or fixed-price Internet access, they can make free telephone calls anywhere in the world to other users with VoIP services or pay relatively little amount for calls to PSTN connected telephones.

A gateway is required when we need to make a call to and from PSTN. There are many manufacturers of such gateways like [Cisco], [Avaya], [Siemens], [Fujitsu], [Mitel] and [Ericsson], there is also an open source PBX application, [Asterisk].

1.1 Why VoIP

VoIP brings many benefits to the user experience, some of which are discussed below

- Cost effective
  The use of VoIP (single infrastructure for data and voice) will remove cost, complexity and management overhead. The same technical personnel are able to operate single network for both voice and data instead of different people with different expertise.

- Convenience
  Software oriented nature makes it easy to implement innovative services, or extend existing services. Additional Services like Voice mail (VM), Conference and auto-attendant system also known as Interactive Voice Response (IVR) may be integrated easily. Further more, VM-to-IVR may be replaced by VM-to-mail. This means that you get a mail straight into your mail box instead of dialling the server to check for messages.

- Service Integration
  Additional services like instant messaging (IM), teleconference, phone book and missed call notification are integrated with ease. This will further lead to simplicity and flexibility.

- Improved mobility
  It is very easy to move an IP phone to another room/location without re-cabling.

Apart from the discussion above, VoIP has some preliminary requirements discussed in part III, which are necessary for its proper operations.

1.2 Why SIP

Session Initiation Protocol (SIP) is an application layer signalling protocol that defines initiation, modification and termination of interactive, multimedia communication sessions between users. SIP has many good features such as:

- Easy service integration
  Reusing existing well established IP protocols like SMTP and HTTP makes it very easy to integrate with new services.

- Simplicity
  Its textual based nature makes it simple to use, easy to debug, extend and process.

- High scalability and extendibility
  The end to end design where intelligence is integrated into end devices makes it highly scalable and provides an endless possibilities of extendibility.

1.3 Report Outline

The report is divided into sections, which guides a reader through increasing level of knowledge of VoIP and SIP.

Part I: is about the technologies involved, which are the building blocks for the whole report. This section describes SIP components and gives an overview of how SIP works.

Part II: talks about the requirements to be fulfilled by a VoIP service provider. The section looks at internal VoIP provider as well as commercial providers who give services to external individuals or companies. The provider can be grouped also as single-site or multi-site.

Part III: is about the inherent problems of the IP world that might affect the VoIP services offered, more discussion on NAT and Firewall will be presented.

Part IV: is an implementation of discussed technology. This is a case study: VoIP-SP at KTH-TSlab. In this section, a detailed HOWTO is given as well as explanations of why some products were chosen in favour of others.

Part V: is about further work and conclusion of the thesis work.

1.4 Background and goal

This project, "*VoIP Service Provider*" is intended to provide the VoIP services to the KTH university community. This thesis work will create a platform for VoIP researchers and other related thesis work and projects at KTH to carry out their researches. The thesis will also save as a reference for new VoIP projects in the "Communication Systems Design (CSD)" course in 2005, which starts in January to June each year.

The proposed services are:

- Telephony services (call in & out). Enabling VoIP subscribers to call in/out other VoIP subscribers as well as VoIP subscribers to call in/out non-VoIP subscribers.
- Subscriber management.
- Interfacing with an accounting system (provided by Hans Eriksson).

In the course of the work, the project will identify and analyze IP related issues (NATs and Firewalls in particular) in regard to VoIP, then provide solutions and/or recommendations on how to deal with them. Open source software will be used in this project.

Part I

2 SIP Technology

SIP is a text-based signalling protocol transported over TCP, UDP or TLS over TCP. SIP inherited some design philosophy and architecture from the Hypertext Transfer Protocol (HTTP) and Simple Mail Transfer Protocol (SMTP) to ensure its simplicity, efficiency and extensibility. SIP supports user mobility by proxying and redirecting requests to the user's current location. User authentication is provided by a challenge-based mechanism that is based on authentication in HTTP.

SIP is a signalling protocol used for establishing sessions in an IP network. A session could be a simple two-way telephone call or it could be a collaborative multi-media conference session. SIP works in concert with several other protocols and is only involved in the signalling portion of a communication session. SIP acts as a carrier for the Session Description Protocol (SDP), which describes the media content of the session, e.g. what IP ports to use, the codec being used. In typical use, SIP "sessions" are simply packet streams of the Real Time Transport Protocol (RTP). RTP is the carrier for the actual voice or video content itself. RTP Control Protocol (RTCP) is the protocol that has a function of providing feedback on the quality of the service offered by RTP.

2.1 SIP URI

SIP entities are identified using SIP URI (Uniform Resource Identifier). A SIP URI has a form of sip:username@domain, for instance, sip:amos@it.kth.se. SIP URI consists of username part and domain name part delimited by @ (at) character similar to e-mail addresses.

2.2 SIP Components

Basic SIP elements are a combination of clients and servers. SIP Servers are defined as network elements that receive SIP requests in order to service them and send back SIP responses to those requests. SIP Clients commonly known as User Agents (UAs) are any network elements that send SIP requests and receives SIP responses. Clients may or may not interact directly with a human user. Figure 1 below shows the interactions of the different components, where the media is going direct between Clients.

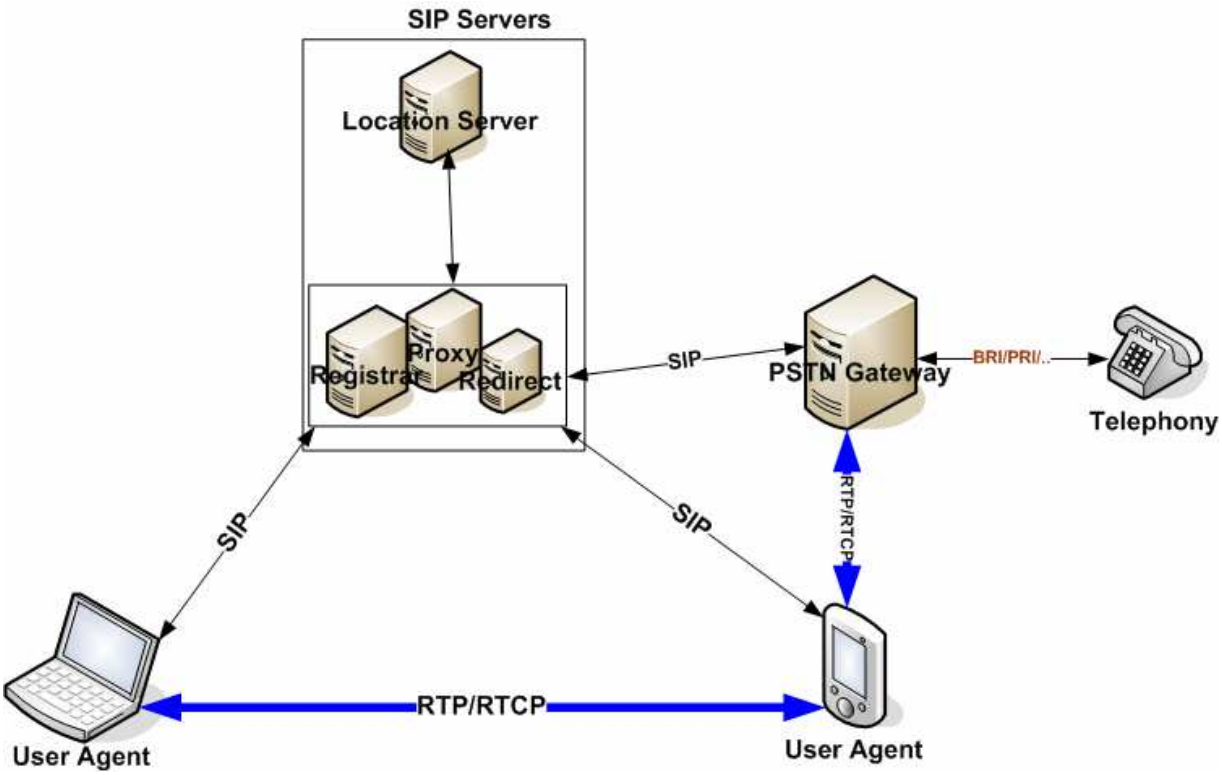

Figure 1: SIP Overview

2.2.1 SIP Servers

The SIP Server (call processing server) is the heart of a VoIP system, managing all VoIP control connections. SIP servers are usually software based and can be deployed as a single server with integrated functionality or different servers with different functions.

There are four types of SIP servers namely: proxy, redirect, location, and registrar server, the different functions they perform are normally stuffed together into a single entity as shown in figure 1.

2.2.1.1 Proxy Server

An intermediary program that acts as both a server and a client to make requests on behalf of other clients. Requests are serviced internally or by passing them on, possibly after translation, to other servers.

2.2.1.2 Redirect Server

A server that accepts a SIP request, maps the address into zero or more new addresses and returns these addresses to the client. It cannot accept calls but can generate a SIP response that instructs the UAC to contact another SIP entity.

2.2.1.3 Location Server

A location server is used by a SIP Redirect or Proxy server to obtain information about a called party's possible location(s).

2.2.1.4 Registrar Server

A server that accepts REGISTER requests. The register server may support authentication and is typically co-located with a proxy or redirect server and may offer location services.

2.2.2 SIP Clients

These can be divided into two groups: the end-points and the media gateways.

2.2.2.1 End-Devices

The user end-devices consist of IP phones and traditional phones with the help of an adaptor. IP phones may be software based ("softphones") or hardware based ("hard phones" or "handsets", like traditional phones). IP phones use the TCP/IP stack to communicate with the IP network. IP phones may also use additional protocols to support VoIP-enabled features, such as built-in IM applications or directory search functions. For convenience, IP phones use DHCP to auto-configure themselves, with the DHCP server telling the phone about the location of the configuration server, which most of the time is identical to the call processing server.

Softphones are software application running on computers, usually targeted towards mobile users. They have the same base features as hard phones. Most of the softphones have a free version. Products like minisip, kphone and linphone are open source while xlite and Sjlab are commercial software which has a free version with limited functionalities. Window Messenger (WM) is another free closed source client.

2.2.2.2 VoIP media gateway

A media gateway is an adaptor which converts signal and speech from Internet to PSTN and vice versa. In addition, media gateways have optional features, such as voice (analogy and/or digital) compression, echo cancellation, silence suppression, and statistics gathering (Call Detail Records - CDR) for administration and billing purposes.

Media gateways exist in several forms and can provide different capabilities depending on the need. For example, media gateways could be a dedicated telecommunication equipment chassis, or even generic PC running VoIP software. The simplest form of media gateway available is the ATA (Analogy Telephone Adaptor) used to connect normal telephones to the Internet.

2.2.3 IP Network

This is the IP backbone that provides the connectivity among the distributed elements (clients and servers) discussed above. The IP infrastructure must ensure smooth delivery of the voice and signalling packets to the VoIP elements.

2.3 How SIP Works

Users in a SIP network are identified by unique SIP addresses, the URI. The user ID can be either a user name or an Electronic Number (ENUM) address. Users register with a registrar server using their assigned SIP addresses. The registrar server provides this information to the location server upon request. When a user initiates a call, a SIP request is sent to a SIP server (either a proxy or a redirect server). The request includes the address of the caller (in the “From” header field) and the address of the intended callee (in the “To” header field).

2.4 SIP Messages

SIP is a text-based protocol. SIP components communicate by exchanging SIP messages. A SIP message is either a request from a client to a server, or a response from a server to a client.

2.4.1 SIP Requests

SIP requests are distinguished by having a Request-Line for a start- line. A Request-Line contains a method name, a Request-URI, and the protocol version separated by a single space character.

2.4.1.1 SIP Methods as identified in [RFC3261].

- REGISTER Registers the user agent.
- INVITE Initiates a call by inviting a user to participate in a session.
- ACK Confirms that the client has received a final response to an INVITE request.
- BYE Indicates termination of the call.
- CANCEL Cancels a pending request.
- OPTIONS Used to query the capabilities of a server or client.
- INFO Used to carry out-of-bound information, such as DTMF2 digits.

```
INVITE sip:echo@sip1.it.kth.se;user=phone SIP/2.0
From: <sip:80001@sip1.it.kth.se;user=phone>;tag=903525412
To: <sip:echo@sip1.it.kth.se;user=phone>
Call-ID: 68233794@192.16.125.146
CSeq: 101 INVITE
Contact:
<sip:80001@192.16.125.146:5062;user=phone;transport=UDP>;expires=1000
Max-Forwards: 70
User-Agent: Minisip
Content-Type: application/sdp
Via: SIP/2.0/UDP 192.16.125.146:5062;branch=z9hG4bK66732081
Content-Length: 142

v=0
o=- 3344 3344 IN IP4 192.16.125.146
s=Minisip Session
c=IN IP4 192.16.125.146
t=0 0
m=audio 32837 RTP/AVP 0
a=rtpmap:0 PCMU/8000/1
```

Figure 2: Example of Request Message (INVITE)

SIP Message detail

- Start Line is the first line of a SIP message which contains:
  - the method or Request type: INVITE
  - the Request-URI which indicates who the request is for
  - the SIP version number **SIP/2.0**
- Dialog identifier is in headers:
  - To tag, From tag, and Call-ID
- All requests and responses in this call will use this same Dialog information.
- Call-ID is unique identifier usually composed of
  - pseudo-random string "@" hostname or IP Address

- CSeq Command Sequence Number
  - Initialized at start of call (101 in this example)
  - Incremented for each subsequent request
  - Used to distinguish a retransmission from a new request
- Also contains the request type (method)

- Contact header contains a SIP URL for direct communication between User Agents
  - If Proxies do not Record-Route, they can be bypassed

- User Agent contains name of the device used

- Max-Forwards is a count decremented by each proxy that forwards the request.
- When count goes to zero, request is discarded and 483 Too Many Hops response is sent.

- Content-Type indicates the type of message body attachment (others could be text/plain, application/sdp, etc.)
- Content-Length indicates the octet (byte) count of the message body.

- Via headers show the path the request has taken in the SIP network
  - The bottom Via header is inserted by the User Agent which initiated the request
  - The top Via headers are inserted by proxies in the path
- The Via headers are used to route responses back the same way

- Content-Length represents amount of data carried by the message.

- ***SDP Message Details***

- v=Version number (ignored by SIP), marks beginning of session description
- o=Owner and session identifier
- s=Session name (Subject)
- c=Connection Data (IP Address)
- t=Time session is active
- m=Media (type, port, RTP/AVP Profile), marks beginning of media description
- a=media attribute line

Figure 3: Message Details

2.4.2 SIP Responses

SIP responses are distinguished from requests by having a Status-Line as their start-line. A Status-Line consists of the protocol version followed by a numeric Status-Code and its associated textual phrase, with each element separated by a single space character. SIP status codes are similar to HTTP status codes with little modifications.

2.4.2.1 SIP classes of Status Codes.

- 1xx: Provisional - request received, continuing to process the request;
- 2xx: Success - the action was successfully received, understood, and accepted;
- 3xx: Redirection - further action needs to be taken in order to complete the request;
- 4xx: Client Error - the request contains bad syntax or cannot be fulfilled at this server;

- 5xx: Server Error - the server failed to fulfil an apparently valid request;
- 6xx: Global Failure - the request cannot be fulfilled at any server.

```
SIP/2.0 200 OK
From: <sip:80001@sip1.it.kth.se;user=phone>;tag=903525412
To: <sip:echo@sip1.it.kth.se;user=phone>;tag=000011DC33AFE1ED
Call-ID: 68233794@192.16.125.146
CSeq: 101 INVITE
Via: SIP/2.0/UDP 192.16.125.146:5062;branch=z9hG4bK66732081
Contact: <sip:echo@130.237.203.11>
Content-Type: application/sdp
Server: Sip EXpress router (0.8.14 (i386/linux))
Content-Length: 132
Warning: 392 130.237.203.11:5060 "Noisy feedback tells:  pid=18444
req_src_ip=192.16.125.146 req_src_port=5062
in_uri=sip:echo@sip1.it.kth.se;user=phone
out_uri=sip:echo@sip1.it.kth.se;user=phone via_cnt==0"

v=0
o=username 0 0 IN IP4 130.237.203.11
s=session
c=IN IP4 130.237.203.11
t=0 0
m=audio 1322 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

Figure 4: Example of Response message (OK)

Detailed drawing of how SIP works.

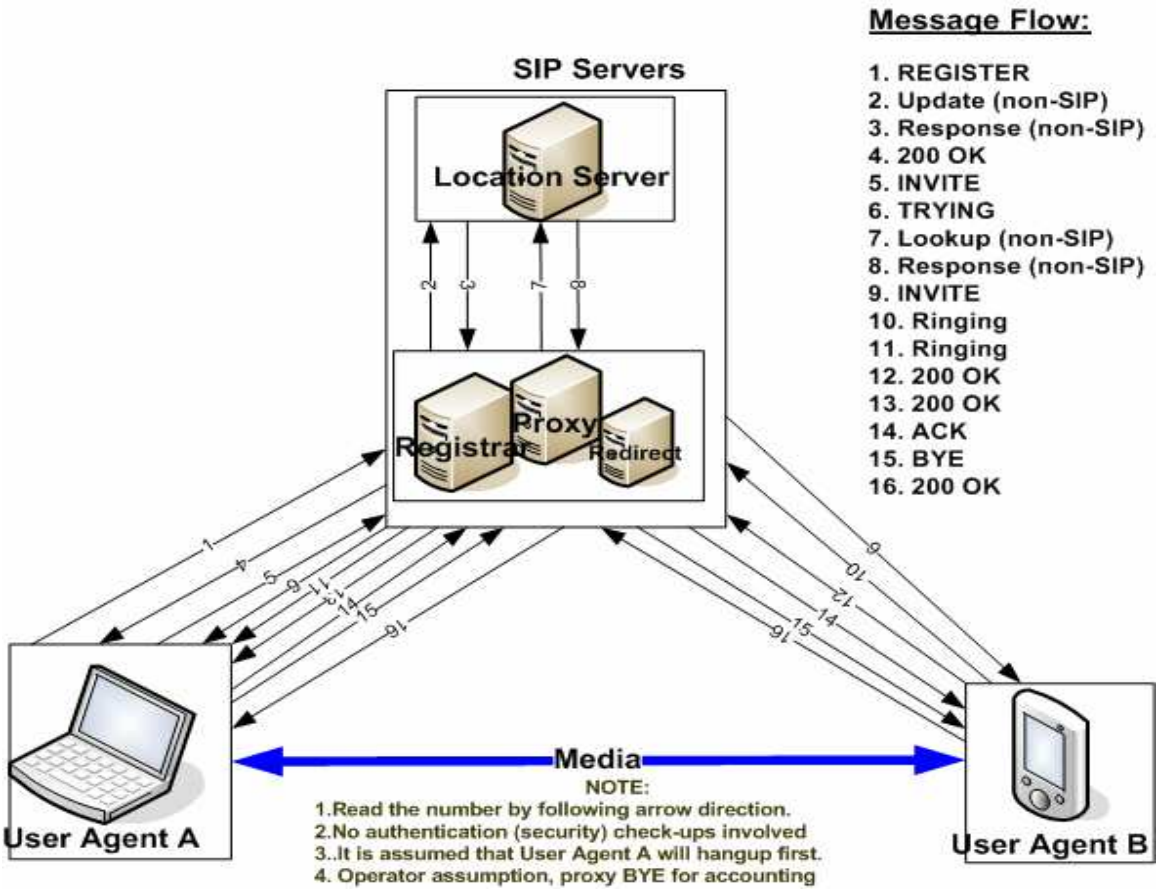

Figure 5: How SIP Works

Part II.

3 Requirements for VoIP Service Provider (VoIP-SP)

VoIP-SPs may be categorized into single site provider like local Internet Service Providers (ISP) and small companies/organizations or multi-site providers like national and international ISP and multi-national. Furthermore, they may be divided into commercial and non-commercial providers. Table below gives a summary on this grouping.

Table 1: VoIP-SPs.

	Commercial	Non-commercial
Single site	ISP	Small company
Multi-site	Public Service Provider.	Big company (multi-national)

Apart from the technological part of the system discussed in part I, VoIP-SP require a customer support system (CSS) for the services provided. The CSS will range from a simple user administration and management system, for VoIP services within an organization to complex integrated business support system (BSS) which may include billing, invoicing and order tracking for commercial operators.

3.1 The Customer Support System (CSS)

Every VoIP-SP requires a mechanism to support and control the services provided. The CSS should provide mechanism that enables technical people to administer users easily, this includes add/remove users and give/deny access rights on system usage to users.

The CSS should also be able to offer support to the users in a very easy/flexible way like user self registration (if many users are to be served), user reminder/retrieval of forgotten or lost password, user configuration/customization of personal account, ability to check/retrieve their statistics (system usage) and user subscription to optional services that don't require access right like voice mail, guest book and the like.

There system should be able to provide a mechanism auditing which is useful for keep track of activities on our VoIP system for:

- Network management
- Bill reconciliation and verification
- Monitor system usage to determine phone usage or abuse of the system.
- Aiding in future plans

It is a desired feature if the CSS can provide customer self help information like checking if they are behind NAT, checking link speed and available bandwidth.

3.2 Business Support System (BSS)

A complete BSS must be able receive and process orders and deposits from customers. Being able to create new customer accounts, send invoice and keep track/status of any pending issues. The system

should be able to notify the support department wherever things go wrong. More important, the BSS system should be able to handle billing issues depending on the policy/scenario advocated by the provider. That means the BSS should be able to tear down calls in a pre-paid scenario when user run out of balance.

3.2.1 Billing scenarios

The billing scenario will depend on the type and trust the provider has over the customers, and also the complexity of the billing system the provider uses.

3.2.1.1 Post-paid

In this scenario, the customers may sign up agreement for receiving the services and agree to pay for the service at the end of the month. From the technical point of view, the billing system is not complex in this situation; the system is only responsible for keep track of system usage for accounting purposes.

3.2.1.2 Pre-paid

The pre-paid scenario is more complex, this requires the billing system to keep track of system usage in real time so that customers may be blocked (call tear down) if they run out of credit. A complex system which is difficult to create or they cost a lot of money to buy is required.

3.2.2 Billing Policies

As a business company, VoIP providers may decide to implement different billing policies based on the market and the nature (environment) where the service is offered.

3.2.2.1 Flat Rate

Providers may decide to charge flat-rate for some service provided, this means every subscriber will get the same bill. This applies mostly if the same service is provided to everybody.

3.2.2.2 Usage based billing

This might be categorized based on the content transferred on the network (how much bandwidth) is consumed. In this case the provider might decide to separate video users from other users who just talk. Also, customers behind NAT who relay their media through our servers may pay more because they consume much of our bandwidth.

3.2.2.3 Type of calls

In places where there is high Internet penetration, the provider might decide not to charge IP to IP calls and instead charge only calls to other networks such as PSTN.

3.3 Service reliability and quality

People expect their telephone service to be very reliable, and the quality of the call to compare that of what is available (PSTN) or much better. The VoIP Service Providers need to assure customers of reliability and quality as the one they used to in normal telephone.

End users must be told that their IP Phones depends on the power in order to operate. Thus, loss of power means loss of communication unless they have power backups.

3.4 Numbering and regulation

There is regulatory uncertainty for the IPT for the moment. They are still debates if VoIP should be treated as a normal Internet application or as a telecommunication application.
Also, process on how to get the numbering (numbers that other networks can reach you) which is recognized nationally is not clear. Number portability is another issue, clients changing providers may not be guaranteed to retain their numbers if the is no such a rule in the particular country.

3.5 Interoperability

It would be better if it could function as a mail system, one can send mail to any domain. VoIP as a technology with many protocols implementations may result in many systems that do not talk to one another. So the cost benefits of all-IP calls are lost.
Operators need to negotiate more VoIP peering and interconnection agreements, and support more than one protocol if they don't have a gateway that can assist from one protocol into another.

Part III

4 VoIP Service Considerations

Here are some generic issues related to VoIP that an organization must carefully consider when deploying VoIP solutions. Most prominent are issues such as traffic parameters, security, NAT traversal and network design. An organization could be faced with service that does not function reliably or is severely degraded.

4.1 IP Traffic Parameters

Some of the IP parameters that might affect the VoIP communication are bandwidth, latency, jitter and packet loss.

4.1.1 Bandwidth

Bandwidth is the amount of data that can be transmitted in a fixed amount of time. VoIP-SPs must make sure that they don't over subscribe for the available bandwidth. As VoIP is a very sensitive real time application, the provider has to make sure there exist enough bandwidth available otherwise packets will get dropped or delayed. This results in bad quality.
Thus, the connection speed between your servers and the customer will affect the type of codec your customer uses. For example, a customer with Upload speed of 64Kbps will not likely use g.711 codec which uses about 64Kbps per channel plus IP overhead.

4.1.2 Latency

Latency (or delay) is the amount of time it takes a packet to travel from source to destination. This will be the amount of time the voice travel from the talker to the listener. Higher delay may result in lack of synchronization between speakers.
The ITU-T Recommendation G.114 [ITUG114] establishes a number of time constraints on one-way latency. The upper bound is150 ms for one-way traffic. VOIP calls must achieve the 150 ms bound to successfully emulate the Quality of Service (QoS) that today's phones provide.

One source of delay is the endpoint, if they don’t have enough processing power for encoding and decoding the voice data may result in delays. But also, each hop along the network may introduce a new queuing delay and possibly a processing delay for example a security checkpoint like firewall or encryption/decryption point.

4.1.3 Jitter

Jitter refers to non-uniform packet delays; it is a measure of time between when the packet is expected to arrive and when it actually arrives. It is often caused by low bandwidth situations in VOIP. Jitter can cause packets to arrive and be processed out of sequence.

The general mechanism to control jitter at VOIP endpoints is the use of a buffer. Buffer may be defined as a temporary storage area, usually in RAM with a purpose of holding data, which has to be processed before transferring it into next stage. But such a buffer has to release its voice packets at least every 150 ms so the variations in delay must be bounded. The buffer implementation issue is compounded by the uncertainty of whether a missing packet is simply delayed, or is actually lost.

Jitter can also be controlled throughout the VOIP network by using routers, firewalls, and other network elements that support QoS. These elements process and pass along time urgent traffic like VOIP packets sooner than less urgent data packets. However, not all network components have this functionality.

4.1.4 Packet Loss

VOIP is intolerant of packet loss. Packet loss can result from excess latency, where a group of packets arrives late and must be discarded in favour of newer ones. It can also be the result of jitter, that is, when a packet arrives after its surrounding packets have been flushed from the buffer, making the received packet useless. Packet loss may result due to the use unreliable UDP for transport, which does not guarantee packet delivery.
Use of codecs such as internet Low Bit-rate Codec (iLBC) which are tolerance to packet loss [RFC3951] may be advocated.

4.2 Power Failure

Conventional telephones continue to work even during a power failure while IP phones needs power to operate. An organization that provides uninterruptible power systems for its data network and desktop computers may have much of the power infrastructure needed to continue communication functions during power outages. Costs may include electrical power to maintain UPS battery charge, periodic maintenance costs for backup power generation systems, and cost of UPS battery replacement.

4.3 Security issues

VOIP security needs to be handled in the overall context of data security. It is important that servers are properly locked down, placed behind firewalls, patched against vulnerabilities (Operating System) and frequently monitored using intrusion-detection systems. Attacks common in the data world like denial of service, identity theft and snooping are likely to happen in VoIP implementation.

4.4 NAT + Firewall

Network Address Translation (NAT) is a technology most commonly used by firewalls and routers to allow multiple devices on a local area network (LAN) with 'private' IP addresses to share a single or few public IP address. A private IP address is an address, which can only be addressed from within the LAN, but not from the Internet outside the LAN. Firewalls are systems designed to prevent unauthorized access to or from a private networks.

The use of NAT and Firewalls in the IP world for security and/or as a supplement to the scarcity of the global IP addresses isolates the people behind those NAT/Firewalls with the rest of the world.
The firewall creates a problem that it allows only pre-configured traffic to pass through. On the other hand, NAT works in a way that users have ``private addresses'' which are not recognized outside their network, they need to change to global IP address wherever one tries to go into the Internet.
The two working mechanisms make it hard for VoIP as discussed in next section.

4.5 Design Consideration

Based on the above discussed issues, it is clear that more understanding of how to deal with those system level challenges is required. Addressing all the above discussed issues in the design, will make your VoIP service to perform in most challenging environments. The design might consider use of redundant servers for reliability, separation of services for fast processing and choosing a NAT traversal mechanism that won't consume much bandwidth.

5 NAT and Firewall

Figure 6 below elaborates how NAT operates. Several clients in the internal network share a single global IP address available to talk to the outside world. Wherever the internal computer wants to talk to outside world, the NAT will create a mapping between the internal IP:port where the request comes from to the IP:Port of the external where the request is going.
Some NAT implementations are statically configured based on services provided, for example if there exists a web server on the local network, all web server requests from outside to the global IP of the NAT machine will be mapped to the web server. Sometimes we have automatic bindings that any request from inside should be forwarded to the global address. This situation is when there is no special internal services offered, any binding request should start from inside. The dynamic binding creates a dependence that internal users can't be reached unless they start the session. This goes not only with SIP but also to any kind of session. Furthermore, unless the NAT has a static mapping table, the mapping that opens when the fi rst packet is sent out from a client through the NAT may only be valid for a certain amount of time, unless packets continue to be sent and received on that IP:port.

The firewall might be configured to allow specific traffic to pass in or out of the network; the traffic is usually identified by using their known port numbers.

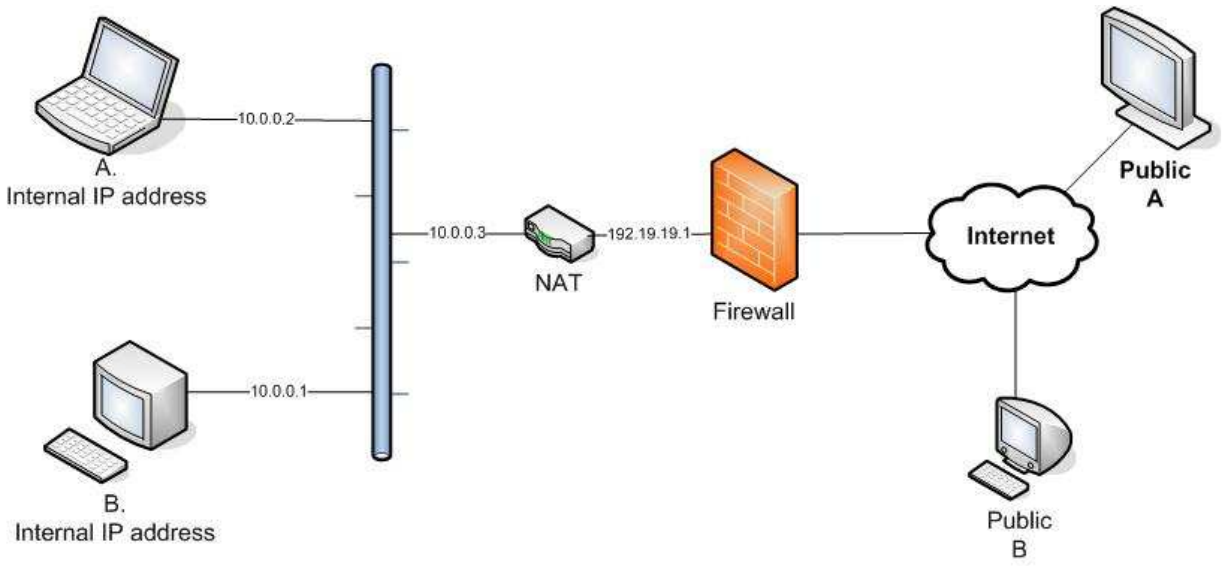

Figure 6: NAT, Firewall

5.1 NAT variations

Different types of NAT policies result in different complexity. The following terminology is adopted from [MIDCOM] working group of the IETF.

- Full Cone
- Restricted Cone
- Port Restricted Cone
- Symmetric

The first three types of NAT maintain a mapping of internal address that is independent of the destination address. The fourth type of NAT will allocate a new mapping for each independent destination address.

5.1.1.1 Full Cone

Mapping is well established and anyone from the public Internet that wants to reach a client behind a NAT needs only to know the mapping scheme in order to send packets to it.

5.1.1.2 Restricted Cone.

In the restricted cone NAT, the external IP:port pair is only opened up once the internal computer sends out data to a specific destination IP. Taking figure 6 as our example: if client A sends out a packet through port 1234 to computer public A, the NAT will maps the client's 10.0.0.2:1234 to 192.19.19.1:4567, and public A can send back packets to that destination. The NAT will block packets coming from public B, until the client sends out a packet to public B's IP address. Once that is done, both external computers A and B can send packets back to the client, and they will both have the same mapping through the NAT.

5.1.1.3 Port Restricted Cone.

A port restricted cone NAT is like a restricted cone NAT, but the restriction includes port numbers. Specifically, an external host can send a packet, with source IP address X and source port P, to the internal host only if the internal host had previously sent a packet to IP address X and port P.

5.1.1.4 Symmetric

A symmetric NAT is one where all requests from the same internal IP address and port, to a specific destination IP address and port, are mapped to the same external IP address and port. If the same host sends a packet with the same source address and port, but to a different destination, a different mapping is used. Furthermore, only the external host that receives a packet can send a UDP packet back to the internal host.

5.2 SIP, NAT and Firewall

There are two parts to a SIP call as discussed already. The first part is the signaling, which is the protocol messages that set up the call and the second part is the actual media stream, which is the RTP packets that travel directly between the end devices.

5.2.1 SIP Signalling

SIP protocol carry User's contact information (IP address and port) and its payload carry the media contact information (IP address and Port) as described in Part I, Figure 3. During Registration, the user's contact information is stored; the stored information indicates user's reach-ability. When a request arrives for this user, the proxy server will query the location server to determine how to reach, it then forward the request accordingly.

SIP signalling protocol, works on a well known port 5060. Hence, NAT and firewall traversal is slightly easy. Figure 7 below is a request - response message between a user behind NAT and the proxy server. It can be noted that the caller provides the private IP address for session creation and media also. The "*via"* line in the response, contains additional parameters, the proxy has added new parameters, the "received source port (*rport)"* and *received* to indicate where did the request come from (these are the NAT device's IP and port where the request went out from. It is a coincidence that the NAT used the same port number to deliver the request). The "rport" was empty in the request message. The proxy will communicate with the caller using these parameters. Since the binding has been created in the NAT

device, the response “Trying” will come back through to the caller without a problem. The proxy will pass on the call to the callee with the media parameters unchanged.

SIP signaling should be able to traverse any of the four types of NATs as long as the proxy returns SIP messages to the NAT from the same source port that it received the initial message on. The initial SIP message, sent to the proxy IP:port, opens up the mapping on the NAT, and the proxy returns packets to the NAT from that same IP:port. This is allowed in any NAT scenario.

Registering a client that is behind a NAT requires either a registrar that can save the IP:port in the registration information based on the port and IP that it sees as the source of the SIP message, or a client that is aware of its external mapped address and port and can insert them into the Contact information as the IP:port to receive SIP messages. Care should be taken to use a registration interval shorter than the keep alive time for the NAT mapping.

5.2.2 Media Stream (RTP)

Media streams work on dynamic port above 1023. Further more, the IP and port of the media streams are written by the clients according to what they know about themselves. A client sitting behind a NAT knows only its internal IP:port, and that is what it puts in the SDP body of the outgoing SIP message. When the destination endpoint wants to start sending packets to the originating endpoint, it will use the received SDP information containing the internal IP:port of the originating endpoint and the packets never get there.

One way audio will be experienced in this scenario (caller to callee) because the media streams from callee won’t reach caller using the provided private address.

REQUEST

```
INVITE sip:samson@voip.ssvl.kth.se SIP/2.0
Via: SIP/2.0/UDP 10.0.0.16:5060;rport;branch=z9hG4bK28F664F43BCE472C872C79AC2C394247
From: nungu <sip:nungu@voip.ssvl.kth.se>;tag=2027040426
To: <sip:samson@voip.ssvl.kth.se>
Contact: <sip:nungu@10.0.0.16:5060>
Call-ID: 4E56CEDF-583F-4B89-8922-DB72BFED3493@10.0.0.16
CSeq: 42187 INVITE
Max-Forwards: 70
Content-Type: application/sdp
User-Agent: X-Lite release 1103m
Content-Length: 259

v=0
o=nungu 1341909 1341919 IN IP4 10.0.0.16
s=X-Lite
c=IN IP4 10.0.0.16
t=0 0
m=audio 8000 RTP/AVP 0 8 3 98 101
a=rtpmap:0 pcmu/8000
a=rtpmap:8 pcma/8000
a=rtpmap:3 gsm/8000
a=rtpmap:98 iLBC/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-15
```

RESPONSE

```
SIP/2.0 100 trying -- your call is important to us
```

```
Via: SIP/2.0/UDP
10.0.0.16:5060;rport=5060;branch=z9hG4bK28F664F43BCE472C872C79AC2C394247;received=130.23
From: nungu <sip:nungu@voip.ssvl.kth.se>;tag=2027040426
To: <sip:samson@voip.ssvl.kth.se>
Call-ID: 4E56CEDF-583F-4B89-8922-DB72BFED3493@10.0.0.16
CSeq: 42187 INVITE
Server: Sip EXpress router (0.8.14 (i386/linux))
Content-Length: 0
```

Figure 7: SIP Messages - Caller behind NAT

5.3 Different NAT scenarios

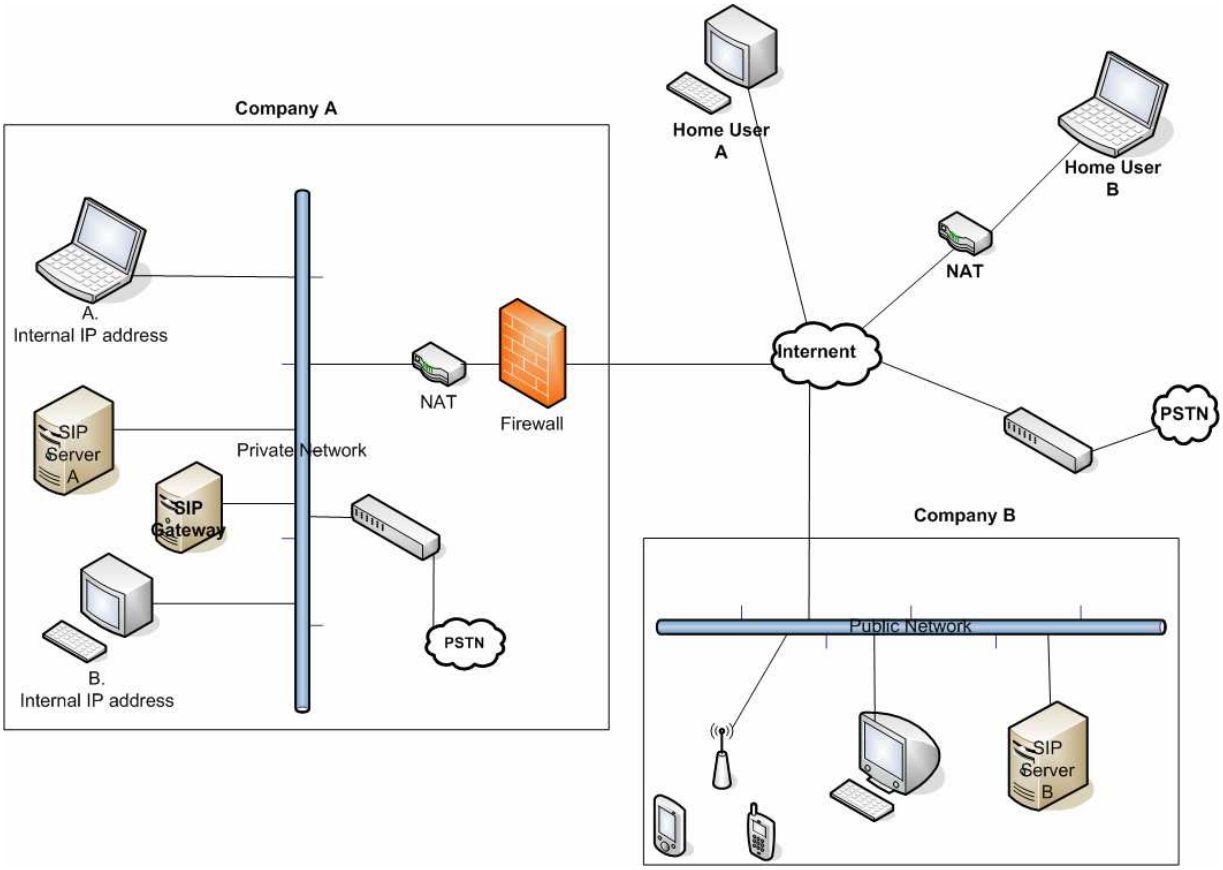

Figure 8: NAT Scenarios

The diagram above represents all possible providers discussed in Part II. Company A can be treated as a single-site non-commercial site while company B as a single-site commercial company. Combination of A and B would create a multi-site company. Now, home user B might belong into any group, any category. Worse enough is when user B belongs to company A, there are two NAT devices to take care of.

Different cases (dialogues):
Based on the diagram above, different dialogues involving NAT can be created.

- Home User A vs Home User B
  This is a general scenario that will result in one way voice as discussed in section 5.2.2.

The same will happen if client behind the NAT establishes a call with PSTN telephony.

- Home User A vs Company A
  The SIP Servers in company A must have a Full Qualified Domain Name (FQDN), and company A had to make sure that their service can be accessed from outside. It is a one NAT case scenario resulting into one way voice.
  This scenario can be further extended that company A depends on company B for VoIP services (SIP Server in public internet).
- Home User B vs Company A
  The same assumption regarding use of SIP Server in company A as above. We have two NAT in this setup; the result will be no voice in both directions.
- Within company A (A vs B)
  Two employees want to talk to each other. There shouldn't be a problem as shown in the setup because they are in the same network.
  The scenario can be further extended to assume that company A depends on company B for VoIP services. In this assumption, the router should be intelligent enough to forward the call (media) correctly by keeping local content in the local network.

5.4 Solutions available

No single solution exists to solve all the NAT scenarios discussed above easily. Below are some of the solutions implemented today and their implications.

5.4.1 SIP specific solutions

In these solutions, NAT and firewall are supposed to understand the way SIP operates and react accordingly.

5.4.1.1 Application Layer Gateways (ALGs)

This technique relies on the installation of a new, enhanced Firewall/NAT - called an Application Layer Gateway (AGL) that understands the signalling messages and their relationship with the resulting media flows. The ALG processes the signalling and media streams so it can modify the signalling to reflect the public IP addresses and ports being used by NAT. [RFC2663] states that:

> *"ALG may interact with NAT to set up state, use NAT state information, modify application specific payload and perform whatever else is necessary to get the application running across different address realms"*.

AGL drawbacks:

- Implementing AGL requires the replacement of the existing NAT/Firewall with an ALG or the upgrade of existing NAT/Firewalls to support ALG functionality.
- There is significant performance costs associated with the implementation of an ALG: the manipulation of VOIP packets may introduces latency into the system and can contribute to jitter when high call volumes are experienced.
- Depending on the firewall architecture, this can also slow down throughput in the firewall, contributing to general network congestion.
- A firewall with ALG support can be expensive, and would need to be upgraded or replaced each time the standards for VOIP change.

5.4.2 General solutions

Described below are general solutions supposed to work on any kind of protocol, not only SIP.

5.4.2.1 Static forwarding

The NAT box is configured such that all the IP addresses and port numbers of involved IP phones are known before and statically mapped. The media ports also for the involved phones should be known and configured (port forwarding). The solution is reliable but is possible only if few clients are involved.

5.4.2.2 STUN

STUN stands for Simple Traversal of UDP (User Datagram Protocol) over NAT. It is a protocol which enables your IP phone, to detect the presence and type of NAT behind which the phone is placed. An IP phone that supports STUN can intelligently modify the private IP address and port in its SIP/SDP message by using the NAT mapped public IP address and port through a series of STUN queries against a STUN server located on the public Internet. This will allow SIP signalling and RTP media to successfully traverse a NAT without requiring any configuration changes on the NAT or Proxy.

Drawbacks to STUN.

- STUN is not widely supported by most VoIP devices yet.
- STUN does not work with the symmetric NAT because the IP:port mappings is dependent on the destination.
- STUN does not support TCP protocol.
- STUN does not solve the problem that bindings expire.

5.4.2.3 UPnP

Universal Plug and Play (UPnP) is an architecture that enables discovery, event notification, and control of devices on a network, independent of operating system, programming language, or physical network connection [UPnP].

UPnP drawbacks

- UPnP relies on the NAT opening pinholes to the outside world under the dynamic control of the UPnP client - maybe a soft phone on a PC. This capability is most likely contrary to most security policies and therefore may not be accepted by communications managers of corporate customers.
- NAT Traversal is possible only if your device (IP phone) and "Internet Gateway Device" supports UPnP
- UPnP will not work in case of cascading (more than one NAT device in series) NATs.

5.4.3 Server side solutions

At this point, we assume that the solutions above are not implemented or they failed to solve the NAT problem based on the implementation.

5.4.3.1 RTP Relay

This implementation involves the use of the SIP Server to detect if the client is behind NAT, and then employ another server (RTP-Relay) for relaying the media if the NAT test succeed. Each part maintains a client-server communication with the RTP-relay server. There is one requirement that the clients (UAs) involved must be symmetric – meaning that they are sending and receiving on the same port. The

solution works over most NAT implementations. Some of the available implementations that work with the SIP Server are [rtpproxy] and [mediaproxy].

Relaying drawback:
Introduces single point of failure and consume providers (host of the media proxy) bandwidth. They might introduce latency also into the system.

5.4.3.2 B2BUA

B2BUA which stands for (back-to-back user agent) is defined in [RFC3261] as:

> "A back-to-back user agent (B2BUA) is a logical entity that receives a request and processes it as user agent server (UAS). In order to determine how the request should be answered, it acts as a user gent client (UAC) and generates requests. Unlike a proxy server, it maintains dialog state and must participate in all requests sent on the dialogs it has established. Since it is a concatenation of a UAC and UAS, no explicit definitions are needed for its behavior".

Definition above indicates that B2BUA means a system standing in the signalling AND the media path. The B2BUA was designed to provide additional features not available in a Normal SIP server like call tear down for billing purposes. The B2BUA solves NAT problem due to its nature of work by staying in the media path, but that was not its main design goal.

B2BUA drawbacks
B2BUA prevents a UA from directly contacting the destination UA. While this enables new functionality and management, it also prevents direct implementation of certain SIP based end-to-end features/services. A B2BUA's failure impacts all calls going through the box. With a SIP proxy, failure only affects new calls. Existing calls or calls to devices whose addresses have been cached are unaffected.

5.5 Summary of NAT/Firewall solutions

	ALG	**Manual**	**UPnP**	**STUN**	**RTP Relay**	***B2BUA**
Cascade NATs (ISP)	N/A	N/A	N/A	YES	YES	YES
NAT Device Configuration and/or Support	YES	Yes*	YES	NO	NO	NO
Phone (Client) Configuration	NO	Yes	NO	YES	NO	NO
Phone (Client) Support	NO	NO	YES	YES	NO	NO
Symetric NAT Support	N/A	YES	N/A	NO	YES	YES

*** - Port translation must be configured, *** - not designed to solve NAT**

5.6 Conclusion

NAT is a problem, especially for real-time communication protocol like SIP because they include their IP addresses in their messages. The problem is big in a sense that all providers discussed in part II are susceptible; you can't predict the behavior of the caller as mobility is the key.

Based on NAT variations discussed in section 5.1, they can further be classified into two groups: symmetric and asymmetric. If the client is behind the asymmetric type, then the solution for NAT traversal depends on the client's capability. The client must find out how its internal IP:port looks to the world and then it must put that information into the SDP message instead of the information reflecting its internal IP:port. Two methods for a client to determine the NAT mapped public IP:port has been discussed. The first is to ask the NAT: UPnP. The second method is to ask someone outside the NAT on the public Internet: STUN.
For symmetric NAT, the RTP Relay which acts as the second endpoint to each of the actual endpoints that are attempting to communicate with each other should be used to relay the media. Another solution here is the use of ALG.

As discussed above, NAT is a problem to VoIP. I have indicated also some of the possible solutions which have some implications. I would suggest the use of STUN and UPnP wherever possible to let users (phones) handle NATs themselves. But, RTP Relay should be employed on the SIP server so that those NAT cases that the client can't will be handled at the server side.

Part IV

6 Case Study - An Implementation at KTH – TSLab.

To fulfil the project goal of implementing a VoIP Services at KTH, I have implemented and tested the described technology at KTH in the TSLab.

6.1 Choices implemented

I have tried many open source implementations to come up with products to use in this case study. Below are the products considered and choices made in the implementation.

6.2 Monitoring Tools

We need monitoring tools for troubleshooting purposes, I have used some tools to be able to diagnose problems or learn packet flows.

6.2.1 Sip Scenario generator

I have installed and used the sip scenario generator [SIPSC] for SIP Call Flow debugging purposes. The programe generates actual call processing trace, captured using ethereal/tcpdump and display them in html or text format. The tool is written in perl, no configuration required, just download and run it with the captured file as an argument. Below is part of the diagram drawn from ethereal captured file when calling the announcement server.

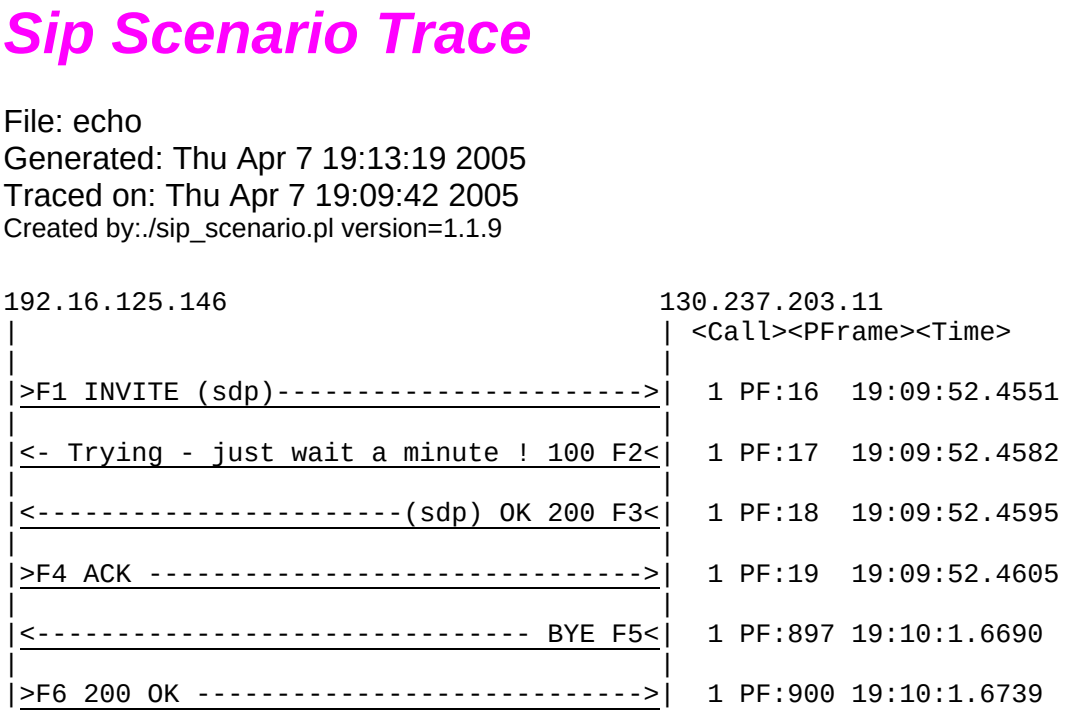

Figure 9: Call flow for announcement server.

6.2.2 Network grep (ngrep)

ngrep is a text based tool for watching network traffic. It is based on the libpcap library, which provides packet capturing functionality. ngrep allows regular expression style filters to be used to select traffic to be displayed. Running ngrep with -t option enables packet capturing with time stamp. Captured files are in text format, hence can be read and analyzed easily. Below is a single message captured using ngrep with this command "***ngrep -d any -t port 5060***".

```
2005/04/09 11:13:07.456836 81.231.254.130:34710 -> 130.237.203.11:5060
INVITE sip:echo@sip1.it.kth.se;user=phone SIP/2.0
From: <sip:80001@sip1.it.kth.se;user=phone>;tag=903525412
To: <sip:echo@sip1.it.kth.se;user=phone>
Call-ID: 68233794@192.16.125.146
CSeq: 101 INVITE
Contact:
<sip:80001@192.16.125.146:5062;user=phone;transport=UDP>;expires=1000
User-Agent: Minisip
Content-Type: application/sdp
Via: SIP/2.0/UDP 192.16.125.146:5062;branch=z9hG4bK66732081
Content-Length: 142

v=0
o=- 3344 3344 IN IP4 192.16.125.146
s=Minisip Session
c=IN IP4 192.16.125.146
t=0 0
m=audio 32837 RTP/AVP 0
a=rtpmap:0 PCMU/8000/1
```

Figure 10: ngrep capture.

6.3 SIP SERVER

This is the central point in VoIP service provider.

6.3.1 Open Source SIP Servers

There are many open source SIP server implementations today. Some of them deploy all the four functionalities described above while others implement only part of the functions. Some of these open source implementations are: - SER, Partysip, Yxa, sipXpbx and Vovida (VOCAL).

6.3.1.1 SER

SIP Express Router (SER) developed by [IPTEL] is a high-performance, configurable, free SIP server which can act as SIP registrar, location, proxy and redirect server. SER is written in C and is licensed under the GPL.

6.3.1.2 Partysip

Partysip is a SIP registrar, a SIP redirect server and a SIP stateful proxy server, built upon [oSIP] GNU SIP stack. Partysip is written in C, licensed under GPL.

6.3.1.3 Yxa

[Yxa] is an open source SIP library and server written in Erlang and is developed by KTH.

6.3.1.4 sipXpbx

The sipXpbx as described on [sipfoundry] page is a "SIP PBX combining: Call routing, registry/redirect server, and Media Server with auto-attendant and voice mail applications".
sipXpbx is distributed under the Lesser General Public License (LGPL).

6.3.1.5 Vocal.

Vocal which stands for (Vovida Open Communication Library) is a SIP server with H.323 and MGCP translators for non-SIP endpoints developed by [VOVIDA].

6.3.2 Alternative selected

SIP Express Router (SER) was implemented in this thesis due to its maturity, high-performance, high configurability and due to the fact that it has been deployed in a large extent compared to the others. SER can act as SIP registrar, location, proxy or redirect server.
SER is built around a processing core [IPTBOOK] that receives SIP messages and enables the basic functionality of handling SIP messages through modules. By using its configuration file (ser.cfg), SER controls which modules and how they should be loaded and then, utilizes the functionalities offered by those modules. Technical information of SER as written on [IPTEL]:

> "C-Written. Ported to Linux (PC, IPAQ), BSD (PC) and Solaris (Sun). Throughput thousands of calls per second (CPS) on a dual-CPU PC (capacity needed to cover Bay Area) and hundreds of CPS on Compaq IPAQ. Support for both IPv4 and IPv6. Small footprint size: 300k core, all common modules (optional) up to 630k."

6.3.2.1 Security in SER

SER provides two kinds of security: authentication (digest) and policy based access control list (ACL). Digest authentication is used to check authorized users of the system. ACL is used to safe guide additional services/resources like PSTN gateway. The [RFC 3261] specify the use of two function www_authorize() for REGISTER messages and proxy_authorize() for other messages. The two functions are used to check user's credential against the values stored in the database subscriber table. If the supplied credentials are correct then these functions will return TRUE, otherwise FALSE is returned. SER has other 2 functions: check_to() and check_from() whose task is to validate usernames against the digest credentials to avoid identity theft.

In REGISTER messages you have to check the *"To header"* because this is the header field that contain the SIP URI being registered. The correct way is to first call www_authorize and then check_to, which would verify if the username in To and digest credentials are the same.

Figure 11: SER Authentication Example

```
if (!www_authorize("voip.kth","subscriber")) {
                  www_challenge("voip.kth","0");
                  break;
};

if (!check_to()) {
                sl_send_reply("401", "Unauthorized");
               break;
};
```

For INVITE messages, call proxy_authorize and then check_from to verify if the usernames in “From” header field and digest credentials are the same. That would prevent people from hijacking identity of someone else.

Figure 12: SER Proxy Authorize Example

```
if (!proxy_authorize("voip.kth","subscriber")) {
            proxy_challenge("voip.kth","0");
           break;
}

if (!check_from()) {
          sl_send_reply("403", "Use From=ID");
          break;
};
```

6.4 Database

The chosen SIP Server supports MySQL, Postgres and text databases. I installed MySQL database because it has commercial support, much more widely used and has many books for reference.

6.5 Customer Support System

My implementation falls into the single-site non-commercial provider as discussed in part II. I found Serweb which is a web interface user provisioning system developed by [IPTEL]. The choice was easy to make because I couldn't find anything else to use in its place. Serweb is easy to use and provides most of the features required by the CSS. Serweb can also provide phone book and IM. Parallel with this thesis, Hans Eriksson (thesis supervisor) is developing an open source BSS system.

6.6 Media gateway

Two products are available, Asterisk and SEMS.
Asterisk was implemented as our PSTN gateway due to its maturity and the ability to provide PBX functionalities.

One BRI (basic rate interface) card was installed in a PC running Linux (Debian) and asterisk was configured to enable VoIP calls to be routed through to PSTN destinations.

6.7 NAT Traversal

RTP proxying was chosen as the way to go when dealing with NAT. As a provider, you have control of the NAT detection and decision of what action to take. This approach also makes client configuration much simpler. The approach is not be the optimal in all cases; the use of STUN should be encouraged wherever possible as it will offload the network.

Further, the choice between [rtpproxy] and [mediaproxy] was hard to make. The two products almost work on the same principle, mediaproxy written in Perl while rtpproxy in C. I tested both of them with my symmetric NAT implementation (Linux computer) and got the same NAT traversal result. I implemented [rtpproxy].

6.8 Additional Services – Media Server

VM, conference and announcements can be achieved by using both SEMS and Asterisk.
SEMS was implemented for voice mail, announcement and conference functions. SEMS is also developed by [IPTEL], easy to implement and works fine with SER.

7 Step by step HowTo

In this section, I'll explain the steps I have undertaken to setup the VoIP-SP at TSLab.

7.1 Overview

This installation guideline was a progressive manual based on my thesis time plan. Some of the installation instructions are just pointed out to the installation manuals in the respective packages. Every problem encountered during the installation process is noted here.

It is possible to install the VoIP components in many different sequences. But I recommend starting with the basic components, testing them, and going on with the others afterwards.

Basic software components: MySQL and SIP server (SER).

Once done with the basic components, you may think of installing additional components like a Web server (with PHP support), a mail server for sending out mails (voice mail and customer support), serweb as a customer support system, and the database for back end support.

I started my work by creating my own working environment whereby I installed and configured DNS and DHCP servers on Linux redhat.
In this document, commands starting with $ can be executed as a normal user, those starting with # should be executed as the super user (root).

7.2 Assumptions

I assume the following in this document:-

- You have a network in place.
- You are setting up a new system from zero, otherwise you only need to add SIP entries to the existing DNS server and go ahead to SER.
- All downloads are saved in "/usr/local/src" directory
- Your database (mysql) admin password is *secret*.
- Use of Linux(Redhat), choose package accordingly.
- Using SER 0.8.14 on Debian or Redhat.

7.3 DNS entries.

The DNS must be configured to enable users to locate SIP services using the "SRV Record" feature. Include the line in the domain's zone file.
"_Service._Proto.Name TTL Class SRV Priority Weight Port Target"

Where: -

Service = sip
Protocol = udp or tcp.

Name = Domain name providing the VoIP-SIP services.
TTL = Time To Live (cache life time).
Class = Type of record (IN for Internet)
SRV = Type of record (SRV).
Priority = determine proxy query order given multiple SRV records. The lower the better.
Weight = used to determine relative capacity between SRV fields with same priority.
Port = Port of service offered. (SIP = 5060).
Target = the proxy (SER) server machine.

7.4 MySQL Client and Server.

7.4.1 Download and Install

Download the following from (www.mysql.com) into our download folder, rpm or Debian equivalent.

- MySQL server (MySQL-server-4.0.18-0.i386.rpm)
- MYSQL client (MySQL-client-4.0.18-0.i386.rpm)
- Libraries and header files (MySQL-devel-4.0.18-0.i386.rpm)
- Dynamic client libraries (MySQL-shared-compat-4.0.18-0.i386.rpm)

Go to the download folder and run the command.

```
$ cd /usr/local/usr/src
# rpm -ivh MySQL-server-4.0.18-0.i386.rpm
# rpm -ivh MySQL-client-4.0.18-0.i386.rpm
# rpm -ivh MySQL-devel-4.0.18-0.i386.rpm
# rpm -ivh MySQL-shared-compat-4.0.18-0.i386.rpm
```

Change root password for MySQL database

```
# mysqladmin -u root password 'secret'.
```

Noted problems:

- Linux (Debian) using 'apt-get installation' will disable the remote login by default. You need to edit the 'my.cnf' under /etc/mysql/my.cnf. Comment the [skip networking (mysql 4.0) or bind address (mysql 4.1)] to enable remote login, which is useful if the database is on a different machine.

7.5 SIP Server – SER

I downloaded the source code, edited the "Makefile" of the main distribution and the "Makefile" of accounting module to enable mysql database support which is disabled by default. For more details, please refer to Admin's guide and SER-HOWTO, found at [IPTEL].

7.5.1 Download and install

Download source code '*ser-0.8.14_src.tar.gz*' from ftp://ftp.berlios.de/pub/ser/0.8.14/src/
into our download directory. unpack the package, enter the new created folder and make some changes to enable mysql database support.

```
$ tar -zxvf ser-0.8.14_src.tar.gz
$ cd ser-0.8.14
$ vi Makefile
```

find section '*exclude_modules?*= ' , remove mysql from the excluded list.

```
$ vi modules/acc/Makefile
```

uncomment the line DEFS+=-DSQL_ACC (remove the #sign).

After the changes above, installation is accomplished by running commands below

```
$ make all
# make install
```

This will put files required directories:-

- Utilities into '/usr/local/sbin/'
- Configuration file into '/usr/local/etc/ser'
- Modules into '/usr/local/lib/ser/modules'

7.5.2 Creating the SER database

Run the command below to create the database

```
$ cd /usr/local/sbin/
# ser_mysql.sh create
```

password '*secret*'

Database created is *ser* with [*ser]* as default user and [*heslo]* as password.

Noted problem:

- Directory *'/usr/local/sbin/*' must be in your $PATH to be able to run the above command without error. It can be achieved by running the command:

```
# export PATH=/usr/local/sbin:$PATH
```

7.5.3 add new users.

SER contains a utility named '*serctl*', which can be used to manage SER including adding new users into the database.

First you need to set the environment variable SIP_DOMAIN to your local SIP realm, e.g

```
# export SIP_DOMAIN=mydomain.com
```

Now you can add some new users (parameters are name, password and email)

```
#serctl add nungu nungu nungu@mydomain.com
#password: heslo
```

NB: enter admin-password for ser database (default is heslo)
More functions of the utility c an be found by running/typing serctl on the command line

7.5.4 The SER configuration file: (ser.cfg)

The SER server behavior (SIP message handling, call setup and policies) is specified in the *ser.cfg* file, which is the main SER configuration file. A default *ser.cfg* file is located at '*/usr/local/etc/ser/*'. That is sufficient to allow you start making calls between clients, but it does not provide any kind of policies, security and any additional services like voice mail. A complete

configuration file which covers authentication, NAT support (rtpproxy), accounting, voice mail, announcement and conference capabilities is provided in Appendix 1.

7.5.4.1 ser.cfg structure.

All configuration scripts in ser.cfg follow the SER language syntax also known as SER request routing language [IPTBOOK]. Six logical sections are dictated in the following ordering:

Global configuration section*:*

these values affect the behaviour of the server such as IP address and port number on which it will be listening, the debug level, the number of spawned children processes, and log level used for the syslog.

Module loading section*:*

This section contains a list of external modules (libraries) that are needed to expose functionality not provided by the core. These modules are shared object ".so" files and are loaded with the "loadmodule" command;

Module-specific parameters section*:*

This section determines the behaviour of the loaded modules. Settings for parameters required by some modules to function properly are done here.

Route blocks section*:*

The route blocks contain the request processing logic, which includes built-in actions as well as actions exported by modules. There is usually the main route block which is the entry point of processing a SIP message and controls how each received message is handled. Additional route blocks can also implement and be called from the main route block or from other secondary route blocks.
Route is usually defined as route[n] and called out by route (n) where n=0 is for the main block. For example, in the configuration file provided in Appendix 1, I use route [1] for message relaying (outgoing route), route [2] for PSTN outgoing calls and route [4] for offline users. Note that route (1) is called in the main block as well as these secondary route blocks.

Reply route block section.

This is an optional route block that may be utilized to handle replies to SIP messages. The reply route block must be called before sending the message out for the reply to follow the reply route. In my implementation, I have one reply route because I want to test for callee also if is behind NAT.

Failure_route block section:

Optionally, if modules supporting reply processing are loaded, one or more *failure_route blocks* containing logic triggered by received replies may be implemented.

7.6 User Provisioning - Serweb

Serweb is a web interface for SER user self-provisioning and administration. This represents the CSS discussed in part II.

Requirements:
- Apache web server
- PHP 4.3 and later with support for mysql (php-mysql package)

- MySQL 4.0
- A running SER with MySQL support
- Mail Transfer Agent (MTA) for user registration confirmation and password reminder (optional). You need to disable the online registration and password reminder if MTA is not installed.

7.6.1 Apache Installation

Download from http://httpd.apache.org/, (December 2004), preferably new version - Apache 2.0.
Follow the installation guideline.

7.6.2 PHP Installation

Download version 4 from (http://www.php.net/downloads.php), (Accessed December 2004). Serweb doesn't work well with php version 5.
Follow the installation guideline.

7.6.3 SERWEB Installation

Download the serweb source file from (ftp://ftp.berlios.de/pub/ser/latest/contrib/) (Accessed December 2004).

Installation procedure:

- Unpack (two folders of interest (html and phplib))
- Move the phplib folder into root directory
- Move contents of html directory 'serweb' directory, created inside the root directory

Commands:

```
$ tar –zxvf serweb_2004-07-27.tar.gz
$ cd  serweb_2004-07-27
# mv phplib /var/www/html
# mv html /var/www/html/serweb
```

Open the file config.php located in serweb and change the values for host, database name, user and password and other values to fit your environment.
Go to the browser and point to the directory where you will have a choice of user_interface (normal user) or admin to get 'index.php'.

Noted Problems:

- Line 178 of config.php must be changed to reflect the folder containing serweb, otherwise nothing will happen when trying to login. Below is the line in my configuration file.
  ->$this->root_path="/html/serweb/";
- Line 216 of config.php to reflect your realm (the realm users get registered into in the subscriber table). Otherwise, "*bad username*" error wherever trying to login. Below is the line in my configuration file (note: the used domain doesn't exist).
  -> $this->realm=$this->domainname=$this->default_domain="voip.kth";

7.7 NAT Support

RTP Relay was implemented. As discussed in section 5.5, client side solution is preferred solution, but server side solution should exist in case client solution is not available or fails to resolve the NAT at hand.

7.7.1 How RTP Relay works

The module for NAT detection is loaded in SER together with other modules. This module will test clients for NAT address existence. The modules have abilities to tell registrar server to mark the client as behind NAT during registration using the *setflag* function, which can be can be checked later when client try to contact another client or be contacted. The module has some more set of actions to perform like updating IP address in the payload or invoking the RTP Proxy (server) to handle the media session.

I implemented [rtpproxy] in my final setup.

7.7.2 RTPPROXY

RTPproxy is a proxy for RTP streams that can help SER handle NAT situations.

7.7.2.1 How it works

As defined in [rtpproxy] README file:

- "When SER receives INVITE reqiest, it extracts call-id from it and communicates it to the proxy via Unix domain socket. Proxy looks for an existing sessions with such id, if the session exists it returns UDP port for that session, if not, then it creates a new session, binds to a first empty UDP port from the range specified at the compile time and returns number of that port to a SER. After receiving reply from the proxy, SER replaces media ip:port in the SDP to point to the proxy and forwards reqiest as usually;

- when SER receives non-negative SIP reply with SDP it again extracts call-id from it and communicates it to the proxy. In this case the proxy does not allocate a new session if it doesn't exist, but simply performs a lookup among existing sessions and returns either a port number if the session is found, or error code indicating that there is no session with such id. After receiving positive reply from the proxy, SER replaces media ip:port in the SIP reply to point to the proxy and forwards reply as usually;

- after the session has been created, the proxy listens on the port it has allocated for that session and waits for receiving at least one UDP packet from each of two parties participating in the call. Once such packet is received, the proxy fills one of two ip:port structures associated with each call with source ip:port of that packet. When both structures are filled in, the proxy starts relaying UDP packets between parties;

- the proxy tracks idle time for each of existing sessions (i.e. the time within which there were no packets relayed), and automatically cleans up a sessions whose idle times exceed the value specified at compile time (60 seconds by default)".

7.7.2.2 Download and installation

Get a copy from CVS, and install by running the commands below:

```
# cvs -d:pserver:anonymous@cvs.berlios.de:/cvsroot/ser co rtpproxy
# cd /rtpproxy
# ./configure
# make
# make install
```

Start the application by running the command below, where it will run in the background.

```
# rtpproxy
```

7.8 SEMS (SER Express Media Server)

Sems is a free media server which helps you adding voice services to your VoIP system. The current version provides voice mail, conference and announcement. Sems can be installed on a separate machine or on the same machine where the proxy server is running. It requires SER instance for its functioning. The source code 'sems_2004-07-27.tar.gz' can found at ftp://ftp.berlios.de/pub/ser/latest/contrib/, (December 2004), fetch it into our download directory. Unpack the package, enter the new created folder and run make and make install as shown below.

```
$ tar -zxvf sems_2004-07-27.tar.gz
$ cd serms_2004-07-27
$ make all
# make install
```

Configuration file 'sems.conf' is placed into '/usr/local/etc/sems/'. No changes required for voicemail/conference support unless you have changed the path for the media files or ser's fifo file which is normally located in "*/tmp*" folder.

7.8.1 How to use it features

- Voice Mail: Every individual has to enable/activate his/her VM service to be able to receive as they will be sent direct into the registered mail. This may be archived from the client side using the webpage (serweb) or an admin can do it from the back end.
- Conference: Any registered user can call conference@yourdomain and be connected.
- Announcement: Any registered user can call announcement@yourdomain and be connected. Mainly used for IVR.

7.9 Clients used in the Tests.

I used free software based SIP phones through out my work. Some of the clients I used are minisip, Linphone, SJphone, Kphone, Xlite and Window Messenger.
The motivation was to enable customers choose any client to use without dependence on the server part. All clients worked together, was able to place and receive calls with all the combination except for minisip vs Linphone where Linphone will crush always in the process.

7.9.1.1 Minisip

Developed at KTH, written in C++, is an open source running on Linux. It supports security (MIKEY/SRTP) and IPSEC.

7.9.1.2 Linphone

Linphone is an open source SIP client running on Linux. Linphone includes a sip test server called "sipomatic" that automatically answers to calls by playing a pre-recorded message [linphone].

7.9.1.3 SJphone

Developed by [SJLabs], not an open source but they have a free version, running on Linux, Mac and Window Operating systems.

7.9.1.4 Kphone

Developed by [kphone], written in C++, is an open source running on Linux and has support for IPv6.

7.9.1.5 Xlite

Developed by [xten], is not an open source but there is a free version running on window and macOS. The Linux better version is now available.

7.9.1.6 Window Messenger

Developed by Microsoft, is free and works on windows OS only.

7.10 Ongoing Work

Creating a VoIP SIP package that will enable people to make easy VoIP setup for any purpose. The package is based on my thesis work and will be an ongoing open source project.
Currently, offers SIP Server – SER (registrar, location and proxy) with mysql support and accounting, SEMS for voice mail and conference, rtpproxy for NAT traversal and it uses serweb as a customer support system.

In the near future, it will include a complete business support system with billing capabilities. I hope that the project will involve many KTH members and possibly non KTH members also as it get big.
The initial package can be accessed at: http://www.web.it.kth.se/~iw03_amn/ITSP (This link might change in the future).

Part V

8 Conclusion

The industry is facing a major shift from circuit-based voice networks to packet-based converged networks that will enable reduced costs, convenience and new revenue-generating services. But, the circuit based communication has a long history; it has been implemented in a large base and almost covers everyone today. On the contrary, IPT depends on data packet networks which are still being deployed, haven't yet covered all the areas as PSTN does. Hence, transition will be continuous, no big bang.

While in transition, providers should workout to find the better solution to overcome the technical and non-technical issues surrounding the IPT phenomena, some of which have been discussed in this thesis work.

8.1 Thesis goal

All the technical work discussed in this thesis was carried successfully in TSLab. The project goal is full-field, the CSD teams managed to take off using my project as a reference and they are still using it. User management is fine as described in part II. I have managed to interface with the billing system offered by Hans Eriksson. However, the billing system is not discussed in this report as it being tested for final release.
My implementation uses source codes only, this makes it easy to add and/or remove functionalities as desired. This also has great advantage that when I tested the package I compiled based on my work, it worked without any modification on Debian, RedHat and Fedora Linux distributions. I happily declare that the project goal is fulfilled.

8.2 Public SIP Server.

As KTH-TSlab has their own SIP client (minisip), it will be good if KTH can host a public SIP Server and configure minisip with default “SIP-KTH” configuration. The configuration can be changed at will. This will help to market minisip as a product. But, the public server will give KTH an image in the VoIP-SIP world which might fuel to start a commercial operation if they wish.

9 Future work

9.1 Extend the implementation to KTH.

I have implemented my services in the Tslab. I hope that the work will soon be implemented at the whole university.

9.2 Phone2PC Calling.

The current implementation offers communication in computer to computer (PC2PC) using softphones and also computer to PSTN (PC2Phone) with the aid of the PSTN gateway. The implementation lacks the PSTN to computer (Phone2PC) calling due to the fact that we don't have internal numbers that can be reached from outside. Moreover, the PSTN gateway implemented (BRI) offers only 2 channels, this also need upgraded to enable more channels for supporting more than two calls at a time.

9.3 Improving the ongoing work.

9.3.1 Billing

Include the BSS, developed by Hans Eriksson into the package once is ready.

9.3.2 moving to SER 0.9.

SER 0.9 has more features than 0.8.14, but it is not yet stable. It would be good to benefit the features provided by the SER by migrating to the new version once it is stable.

9.3.3 NAT – STUN

Far end (RTP Relaying) is a great way of solving the NAT problem since the client doesn't require doing anything. Moreover it is a transparent NAT traversal solution where clients with different NAT capabilities and scenarios are treated the same. But also more bandwidth will be consumed if we allow every client behind the NAT to use our network for media sessions given that we have a huge client base behind NAT. Solution around this is to allow STUN capable clients to use STUN servers to detect and solve the NAT problems.

9.3.4 More setup options

Current configuration has bundled all additional services together. I would like to provide options so that people can choose from (SIP Server only, with SEMS, with rtpproxy and the default – all together).

10 References

[RFC3261] Rosenberg et al., “SIP: Session Initiation Protocol”. RFC 3261, June 2002.

[MAGUIRE] Maguire, J., “Practical Voice Over IP (VoIP): SIP and related protocols”. http://www.imit.kth.se/courses/2G1325/. April 2004.

[SIP] “Session Initiation Protocol (sip) Charter”. http://www.ietf.org/html.charters/sip-charter.html [Accessed on January 2005]

[IPTBOOK] Niccolini et al. “IP Telephony Cookbook” http://www.informatik.uni-bremen.de/~prelle/terena/cookbook/Cookbook_D2/index.html [Accessed on December 2004]

[RFC2916] P. Faltstrom., “ E.164 number and DNS”. RFC 2916, September 2000.

[RFC2663] Srisuresh, P. Holdrege, M. “IP Network Address Translator (NAT) Terminology and Considerations”. RFC 2663, August 1999.

[RFC3235] Senie, D. “Network Address Translator (NAT)-Friendly Application Design Guidelines”. RFC 3235, January 2002

[RFC3489bis] Simple Traversal of UDP Through Network Address Translations (NAT) (STUN). Internet Draft, July 2004.

[RFC2543] Roach A. B., Session Initiation Protocol (SIP) – Specific Event Notification , RFC 2543, June 2002

[RFC1918] Rekhter et al. “Address Allocation for Private Internets”. RFC 1918, February 1996

[IPTEL] SIP Express Router http://iptel.org/ser [Accessed on November 2004]

[SERUSERS] “Mail Support for users of the SIP Express Router”. http://www.archivum.info/serusers@iptel.org/ [Accessed on Dec. 2004 – April 2005]

[ASTERISK] The Open Source Linux PBX http://www.asterisk.org/ [Accessed on January 2005]

[VOIPWIKI] A reference guide to all things VOIP. http://www.voip-info.org/wiki+SIP. [Accessed on Nov. 2004.]

[SIPINTRO] “An Introduction to the SIP Protocol for the Impatient Technologist” http://www.toyz.org/cgi-bin/sipwiki.cgi/SipIntro [Accessed on January 2005]

[mediaproxy] MediaProxy a far-end NAT traversal solution http://www.ag-projects.com/SER_Media_Proxy.html [Accessed on January 2005]

[rtpproxy] PortaOne nathelper RTP proxy: http://www.portaone.com/resources/downloads/index.html [Accessed on January 2005]

[FCP] Firewall Communication Protocol http://www.iptel.org/fcp [Accessed Nov. 2004]

[ITUG114] International Telecommunication Union. ITU-T Recommendations G.114 (2003). “One-way Transmission Time”. Revised Draft.

[MIDCOM] Rosenberg, Weinberger, Huitema, Mahy, STUN - Simple Traversal of UDP Through NATs Internet Draft: Internets draft-rosenberg-midcom-stun-00.txt. October 2001.

[STUN]	Rosenberg et al. “STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)”. RFC 3489, March 2003
[UPnP]	Universal Plug and Play Forum http://www.upnp.org [Accessed on December 2004]
[DUNDi]	Distributed Universal Number Discovery http://www.dundi.com/ [Accessed on April 2005]
[SEMS]	SIP Express Media Server http://sems.berlios.de/ [Accessed on December 2004]
[VOVIDA]	communications community site. http://www.vovida.org/ [Accessed on December 2004]
[SIPFOUNDRY]	A SIP Enterprise PBX. http://www.sipfoundry.org/sipXpbx/index.html. [Accessed on December 2004]
[PARTYSIP]	The partysip SIP proxy server. http://www.nongnu.org/partysip/partysip.html. [Accessed on December 2004]
[oSIP]	The GNU oSIP library. http://www.gnu.org/software/osip/ [Accessed on December 2004]
[YXA]	SIP software written in Erlang. http://www.stacken.kth.se/project/yxa/ [Accessed on December 2004]
[SIPSC]	The SIP Scenario Generator. http://www.iptel.org/~sipsc/ [Accessed on January 2005].
[minisip]	Internet telephone on linux. http://minisip.org/. [Accessed on December 2004.]
[linphone]	Internet telephony on linux. http://www.linphone.org/ [Accessed on January 2005.]
[kphone]	Internet telephony on linux. http://www.wirlab.net/kphone, [Accessed on December 2004.]
[xlite]	Internet telephony on window. http://www.xten.com, [Accessed on December 2004.]
[sjlab]	Internet telephony on linux and window. http://www.sjlabs.com/, [Accessed on February 2005].
[messenger]	Instant Messaging service http://www.microsoft.com/windows/messenger, [Accessed on December 2004]
[pandora]	Installing Asterisk. http://users.pandora.be/Asterisk-PBX/InstallAsterisk.htm. [Accessed on February 2005].
[Cisco]	Cisco Systems, Inc (2005). http://cisco.com/. [Accessed on January 2005.]
[Siemens]	Siemens AG (2005).http://siemens.com. [Accessed on January 2005].
[Avaya]	Avaya (2005). http://www.avaya.com/. [Accessed on January 2005].
[Fujitsu]	Fujtisu (2005). http://www.fujitsu.com/global/. [Accessed on Januari 2005]
[Mitel]	IP Telephony Systems and Solutions (2005). http://www.mitel.com/. [Accessed on Januari 2005]
[Ericsson]	Ericsson – the world leading supplier in telecommunication (2005). http://www.ericsson.com/. [Accessed on January 2005]
[Alcatel]	Alcatel (2005).http://www.alcatel.com/. [Accessed on January 2005]

11 Appendix

11.1 Appendix 1: SER Configuration file

This is a complete configuration file of the implementation at KTH-TSLab.

11.1.1 Structure

The configuration has all 5 out of 6 logical sections discussed in part IV section 11.5.4
Global section
Module loading section
Module configuration section
Route Block section
- Main Route.
- Route [1]: Message Relaying Route and NAT traversal handling.
- Route [2]: PSTN calls ACL verification
- Route [4]: Offline users and media server (VM, conference, announcement)

Replay Route Block section
- Onreply_route[1]: Reply messages are checked for testing if callee is behind NAT.

----------Configuration file start here----------

```
#SER 0.8.14 Configuration file by Amos Nungu.
#....
# Proxy server + Nathelper + SEMS (VM, conference, announcement) + DB-ACC + PSTN
#....
#20050315, Tslab-KTH

# ----------- global configuration parameters ------------------------

debug=9
fork=yes
log_stderror=yes
check_via=no
dns=no
rev_dns=no
port=5060
children=4
fifo_mode=0666
fifo="/tmp/ser_fifo"
#listen=192.16.125.xxx
#alias=192.16.125.xxx
#alias=voip.kth
# ------------------ module loading ----------------------------------

loadmodule "/usr/local/lib/ser/modules/mysql.so"
loadmodule "/usr/local/lib/ser/modules/sl.so"
loadmodule "/usr/local/lib/ser/modules/tm.so"
loadmodule "/usr/local/lib/ser/modules/rr.so"
loadmodule "/usr/local/lib/ser/modules/maxfwd.so"
loadmodule "/usr/local/lib/ser/modules/usrloc.so"
loadmodule "/usr/local/lib/ser/modules/registrar.so"
loadmodule "/usr/local/lib/ser/modules/vm.so"
loadmodule "/usr/local/lib/ser/modules/group.so"
loadmodule "/usr/local/lib/ser/modules/uri.so"
loadmodule "/usr/local/lib/ser/modules/acc.so"
loadmodule "/usr/local/lib/ser/modules/nathelper.so"
loadmodule "/usr/local/lib/ser/modules/auth.so"
```

```
loadmodule "/usr/local/lib/ser/modules/auth_db.so"
loadmodule "/usr/local/lib/ser/modules/textops.so"

# ------- setting module-specific configuration parameters ------------

# ------db_url settings-----------
modparam("usrloc|acc|auth_db|voicemail|group", "db_url",
"mysql://ser:heslo@localhost/ser")

# -- usrloc params --
modparam("usrloc", "db_mode", 2)

# -- auth params --
modparam("auth_db", "calculate_ha1", yes)
modparam("auth_db", "password_column", "password")

# -- rr params --
modparam("rr", "enable_full_lr", 1)

#----accounting parameters----
modparam("acc","log_missed_flag",3)
modparam("acc","log_level",1)
modparam("acc","log_flag",1)
modparam("acc", "db_flag", 1)
modparam("acc", "db_missed_flag", 3 )

#-----Nathelper parameters-----
modparam("registrar", "nat_flag", 6) # indicates client behind NAT
modparam("nathelper", "natping_interval", 30)
modparam("nathelper", "ping_nated_only", 1)
modparam("nathelper","rtpproxy_sock", "/var/run/rtpproxy.sock") #rtpproxy
conect to ser

# -------------------------  request routing logic -------------------

# main routing logic

route{
      log(1, "..............\n");
      log(1, "entering main loop\n");

      # ------------initial sanity checks --------------------------
      if (!mf_process_maxfwd_header("10")) {
            sl_send_reply("483","Too Many Hops");
            break;
      };
      if ( msg:len > max_len ) {
            sl_send_reply("513", "Message too big");
            break;
        };

      #-------------NAT Test-----------------------------------------
      if (nat_uac_test("3")) {
            log(1, "src address different than via header->NAT detected\n");
            log(1, "force_rport and fix_nated_contact and setflag(6)\n");
            if ((method=="REGISTER") || (!search("^Record-Route:"))) {
                  # Rewrite contact with source IP of   signalling
                  fix_nated_contact();
                  if (method=="INVITE") {
                        fix_nated_sdp("1");
                  };
                  force_rport();# Add rport parameter to topmost Via
                  setflag(6);
            };
```

```
append_hf("P-hint: fixed NAT contact for request\r\n");
};

#----------------Record Route Processing-------------------------
if (method!="REGISTER") {
      record_route();
};

#----------------Media tear down---------------------------------
if (method=="BYE" || method=="CANCEL") {
      unforce_rtp_proxy();
};

setflag(1); #account all calls, may be moved later to PSTN Section only.

# ------------ loose-route processing--------------------------------
if (loose_route()) {
      #how about NATed re-invite?
      route(1);
      break;
};

#---------------PSTN Call Processing-----------------------------

#IF PSTN Destination through our gw: (internal dialling plan is 5 digits)
#Assumption: 6 to 20 digits must be pstn.
if (uri=~"^sip:[0-9]{6,20}@.*") {
            log(1, "PSTN call...\n");
            route(2);
            break;
};

#-----------Call type processing-----------------------------------
if (!uri==myself) {
      route(1);
      break;
};

if (uri==myself) {

      if (method=="REGISTER") {

            #digest authentication
      /*    if (!www_authorize("voip.kth", "subscriber")) {
                        www_challenge("voip.kth", "0");
                  break;
             };
      */
            #checking To username against URI
            #table or digested credentials
            if (!check_to()){
                  sl_send_reply("401", "Unauthorized");
                  break;
            };

            #Remove any credentials before relaying
            consume_credentials();

             save("location");
             break;
      };#End Method Register

      #Find the canonical username
      lookup("aliases");
```

```
            #verify you are stiil resiponsible after lookup
            if (!uri==myself) {
                  route(1);
                  break;
            };

            # does the user wish redirection on no availability? (i.e., is s/he
            # in the voicemail group?)
            if (is_user_in("Request-URI", "voicemail")) {
                  setflag(4);
            };

            # native SIP destinations are handled using our USRLOC DB
            ##offline or non-existence users in route(4), the route handles
SEMS..
            if (!lookup("location")) {
                  route(4);
                  break;
            };

            # forward to current uri now;

            route(1);

      };


} #end -main logic


#-------Default message handler logic (Relaying
Messages)-----------------------------
route[1]{
      # !! Nathelper
      if (uri=~"[@:](192\.168\.|10\.|172\.(1[6-9]|2[0-9]|3[0-1])\.)" &&
!search("^Route:")){
            sl_send_reply("479", "We don't forward to private IP addresses");
            break;
      };

      # if client or server know to be behind a NAT, enable relay
      if (isflagset(6)) {
            force_rtp_proxy();
      };

      # NAT processing of replies; apply to all transactions (for example,
      # re-INVITEs from public to private UA are hard to identify as
      # NATed at the moment of request processing); look at replies

      t_on_reply("1");

      # send it out now;

      if (!t_relay()) {
            if (isflagset(6)) {
                  unforce_rtp_proxy();
            };
            sl_reply_error();
      };
}   #End of Route 1
```

```
#-------PSTN Routing logic-----------
route[2] {

      # now check if it really is a PSTN destination which should be handled
      # by our gateway; if not, and the request is an invitation, drop it --
      # we cannot terminate it in PSTN; relay non-INVITE requests -- it may
      # be for example BYEs sent by gateway to call originator
      if (!uri=~"sip:\+?[0-9]+@.*") {
            if (method=="INVITE") {
                  sl_send_reply("403", "Call cannot be served here");
            } else {
                  forward(uri:host, uri:port);
            };
            break;
      };

      # in all other cases, we need to check the request against access control
lists;
      # first of all, verify request originator's identity

      /* uncomment this if realy have pstn connection
      if (!proxy_authorize(   "voip.kth", "subscriber" ))  {
            proxy_challenge( "voip.kth" , "0");
            break;
      };
      */
      #checking From username against URI table or digested credentials
      if (!check_from()){
            sl_send_reply("401", "Unauthorized");
            break;
      };

      #Remove any credentials before relaying
      consume_credentials();

      # authorize only for INVITEs -- which causes PSTN costs
      if (method=="INVITE") {
            # does the authenticated user have a permission for local
            # calls (destinations beginning with a single zero)?
            # (i.e., is s/he in the "local" group?)
            if (uri=~"sip:[1-9][0-9]+@.*") {
                  if (!is_user_in("credentials", "local")) {
                        sl_send_reply("403", "No permission for local calls");
                        break;
                  };
            #same for long-distance(domestic) (destinations begin with one
zeros")
            } else if (uri=~"sip:0[1-9][0-9]+@.*") {
                  if (!is_user_in("credentials", "ld")) {
                        sl_send_reply("403", " no permission for Domestic calls
");
                        break;
                  };
            # the same for international calls (two zeros)
            } else if (uri=~"sip:00[1-9][0-9]+@.*") {
                  if (!is_user_in("credentials", "int")) {
                        sl_send_reply("403", "International permissions
needed");
                        break;
                  };
            # everything else is denied
            } else {
                  sl_send_reply("403", "Forbidden");
                  break;
```

```
            };
      };

      # if you have passed through all the checks, let the call go to GW!

      rewritehostport("192.16.125.xxx:5060");
      route(1);
} # End route 2

#-----------------------------VM, Conference, announcement  and offline
users------------------------------
route[4]{
      log(1,"-------------------------------------------\n");
      log(1, "entering route[4] = requested user not online\n");
      # non-Voip -- just send "off-line"
      if (!(method == "INVITE" || method == "ACK" || method== "CANCEL" || method
== "REFER"
            || method == "BYE"))  {
             log(1, "no invite,ack,cancel,refer->return 404\n");
             sl_send_reply("404", "Unknown Request");
             break;
       };

      # not voicemail subscriber and no echo/conference call; logs for
troubleshooting purposes.
      if ( isflagset(4)) {
            log(1, "flag(4) active\n");
      };
      if (uri =~ "conference") {
            log(1, "conference call\n");
      };
      if (uri =~ "echo") {
            log(1, "echo call\n");
      };

      if ( !( isflagset(4) || (uri =~ "conference") || (uri=~ "echo") ) ) {
            log(1, "no voicemail subscriber->return 404");
            # call invitations to off-line users are reported using the
            # acc_request action; to avoid duplicate reports on request
            # retransmissions, request is processed statefuly (t_newtran,
            # t_reply)
            if ((method=="INVITE" || method=="ACK") && t_newtran() ) {
                  t_reply("404", "Not Found");
                  #acc_request("404", "Not Found"); #produce error sometimes
                  break;
            };
            # all other requests to off-line users are simply replied
            # statelessly and no reports are issued
            sl_send_reply("404", "Not Found and no voicemail turned on");
            break;
       };

      # forward to voicemail now
      ####### t_relay_to_udp("IP of SEMS", "port");
      #The above line, if code below, is running on a separate instance

      # switch to statefull mode:
      if (!t_newtran()){
            sl_send_reply("500","could not create transaction");
                  break;
      };

      # prevent timeout on the other side:
       t_reply("100","Trying - just wait a minute !");
```

```
	if (method=="INVITE"){

		if (uri=~"conference") {
			# assumes that Sems configuration parameter 'fifo_name='
			# has been set to /tmp/am_fifo
			log(1, "forward to conference\n");
			if(!vm("/tmp/am_fifo","conference")) {
					t_reply("500","error contacting sems");
			};
				break;
		};

			if (uri=~"echo") {
			log(1, "forward to announcement\n");
			if(!vm("/tmp/am_fifo","announcement")) {
					t_reply("500","error contacting sems");
			};
				break;
			};

			# Otherwise, redirect to voicemail
		log(1, "forward to vm\n");
			if(!vm("/tmp/am_fifo","voicemail")) {
			 t_reply("500","error contacting sems");
			};
			#break;


		if (method=="BYE" || method=="CANCEL") {
			#BYE is a reserved name which tells Sems t call.
			if(!vm("/tmp/am_fifo","bye")) {
				t_reply("500","error contacting sems");
			};
			break;
		};
	};

}


onreply_route[1] {

	# Not all 2xx messages have a content body so here we
	# make sure our Content-Length > 0 to avoide a parse error
		# 183 = session in progress

	if (isflagset(6) && status =~ "(183)|2[0-9][0-9]") {
		if (!search("^Content-Length:\ 0")) {
			force_rtp_proxy();
		};
	};

	  # otherwise, is it a transaction behind a NAT and we did not
	# know at time of request processing ? (RFC1918 contacts)
	if (nat_uac_test("1")) {
		fix_nated_contact();
	 };
}
```

11.2 Appendix 2: Asterisk installation and configuration files

For the PSTN gateway, we used a BRI (Basic Rate Interface) card. Thanks to Johan Bilien for setting up the the Asterisk server. The information below is based on Johan's manual for the “PSTN gateway laboratory of march 2005. The PCs used has Debian operating system, which comes with Asterisk package.

11.2.1 Installation

Step by step installation process.

11.2.2 Install Zaptel

Zaptel is an interface and a set of drivers to various PSTN cards, mostly manufactured by Digium. We use the Debian package for it:

```
# apt-get install zaptel
```

11.2.3 Install the card driver (kernel module)

To compile the driver, it is necessary to have a copy of the headers corresponding to the current kernel. This way we can compile modules that will happily run with the current kernel. On Debian this can be done with:

```
# export KERNEL_VERSION=`uname -r`
# apt-get install "kernel-headers-${KERNEL_VERSION}"
```

Now it's time to get the driver itself. For BRI cards with a HFC chipset, we use a modified version of the zaptel framework. junghanns.net provides modifications to zaptel to handle BRI cards with an HFC chipset. Get the latest version of bristuff from http://www.junghanns.net/asterisk/downloads/

The Debian packages for Asterisk and Zaptel includes the bristuff patches, so we only need to compile the kernel module (driver for the BRI card).

```
$ tar xvzf bristuff-0.2.0-RC7j.tar.gz
$  cd bristuff-0.2.0-RC7j
$  cd zaphfc
$  make
#  cp ./zaphfc.ko /lib/modules/${KERNEL_VERSION}/misc
   # depmod -a
```

One problem can happen when Linux tries to use his own drivers for the card, instead of using the one we have just installed. To prevent that, I recommend removing those built-in drivers:

```
      # mv /lib/modules/${KERNEL_VERSION/kernel/drivers/isdn
/root/isdn-backup
      # depmod -a
```

After that, a reboot of the computer could be necessary to be sure that the card is initialized properly.

```
# reboot
```

11.2.4 Install Asterisk

We are ready to install Asterisk. On Debian this is rather simple:

```
# apt-get install asterisk
```

After that on Debian you need to modify the file /etc/default/asterisk with RUNASTERISK=yes.
You can then start Asterisk with:

```
#/etc/init.d/asterisk start
```

11.2.5 Configuring zaptel

The card driver requires some configuration to tell the card how it should communicate with the other side. This configuration is done in the file /etc/zaptel.conf. First you must configure the spans. A span is a bunch of channels that share a common configuration. Usually, one span corresponds to one port on the card.
In this setup, these parameters were used:

span=1,1,2,ccs,ami

The span 1 is configured to be the primary synchronization source (1), to use a Common Channel Signaling (ccs), to use ami coding.

Then you need to configure the channels. For BRI, you have 3 channels, which are numbered from 1 to 3. 3 is for signalling (D channel), and the other two for traffic (B channels). So a typical configuration, and the one we will use for this lab, will be:

bchan=1-2
dchan=3

Finally you need to configure the tone zone, which is the kind of tone your PSTN operator uses in your country.

Asterisk will be running as the user asterisk. Therefore you should be sure that this user has read/write access to the devices. The simplest is to make asterisk the owner of the /dev/zap/ directory and the files in it:

```
# chown -R asterisk.asterisk /dev/zap
```

Each time you change something in the /etc/zaptel.conf file, you can apply the new settings with

```
# ztcfg
```

Complete zaptel configuration file.

```
span=1,1,2,ccs,ami
bchan=1-2
dchan=3
defaultzone=se
loadzone=se
```

11.2.6 Configuring Asterisk

The next step is to configure asterisk itself. First we must configure the input and output (channels) of our gateway. After that we are ready to configure a dialing plan. All the configuration files for Asterisk are located in /etc/asterisk/.

sip.conf

This file holds the configuration of Asterisk's SIP interface. In this file you will configure the SIP users that are likely to interact with asterisk. In our case, the handling of SIP users is done by SER, so we had a very simple SIP configuration in Asterisk. The file is divided in a general section, with options that apply to all the SIP communications (starting with [default]), followed by series of per SIP account options.

Complete sip configuration file.

```
; SIP Configuration for Asterisk
[general]
context=fromsip                 ; Default context for incoming calls
port=5060                     ; UDP Port to bind to (SIP standard port is 5060)
bindaddr=0.0.0.0                ; IP address to bind to (0.0.0.0 binds to all)
srvlookup=yes                   ; Enable DNS SRV lookups on outbound calls
disallow=all                    ; First disallow all codecs
allow=ulaw                      ; Allow codecs in order of preference
rtpholdtimeout=300            ; Terminate call if 300 seconds of no RTP activity
```

zapata.conf

Next, we must configure how Asterisk will use zaptel and our BRI card. This is done in zapata.conf. In most cases you can use the default options. The [channels] section is used to assign options to physical channels on the BRI interface. The group option is used to assign a group number to a bunch of channels. This is used for outgoing calls (see extensions.conf).

Complete zapata configuration file.

```
;Zapata telephony interface
;
[channels]
switchtype=euroisdn
signalling=bri_cpe_ptmp
echocancel=yes
echocancelwhenbridged=yes

group=1
callerid=asreceived
context=frompstn
prilocaldialplan=unknown
pridialplan = unknown
channel => 1-2
```

Dialing plan: extensions.conf

Now the need to setup the dialing plan. which is done in extensions.conf. This file is divided into contexts and extensions. A context is a sub-section in the dialing plan. For example, in the sip.conf and zapata.conf we refer to 2 contexts, fromsip and frompstn. We need to define those two contexts in the dialing plan. Calls coming from SIP will go directly to the fromsip context, calls coming from the PSTN to the frompstn context.

An extension corresponds to a phone number. For each phone number or bunch of phone numbers, we define a list of actions to do one after the other. To describe an extension, you can either give the entire phone number, or use some kind of filter to match a group of numbers. In this case, you must precede the extension with a . After that, an X matches any digit, an N matches any digit larger or equal to 1 and a dot '.' matches one or more characters.

Some extensions have a special meaning: the s extensions is always the first one executed in a context, the t extensions is executed when the timeout of an action was reached.

The most common command is Dial, which answers the call, then dial another call and patch the traffic through. This is the one we use to build gateways.

Complete extension configuration file.

```
; Static extension configuration file, used by the pbx_config module. This is
where you configure
;all your inbound and outbound calls in Asterisk.
;The "General" category is for certain variables.
;
[general]
;
static=yes
writeprotect=no
;
[globals]

[fromsip]


; For now block international calls
exten => _00.,1,Hangup

; as well as mobiles
exten => _070.,1,Hangup
exten => _073.,1,Hangup
exten => _076.,1,Hangup

; as well as special tariffs
exten => _0900.,1,Hangup
exten => _0939.,1,Hangup
exten => _0942.,1,Hangup

exten => _X.,1,Dial(ZAP/g1/${EXTEN}|20,tr) ;
exten => _X.,2,Hangup

;exten => _X.,1,Dial(SIP/ssvl/${EXTEN:1}|20,tr) ;

[frompstn]
exten => _X.,1,Dial(SIP/tslab@ssvl.kth.se,20,tr)   ; Dial ${EXTEN}@ssvl.kth.se

[frompstn2]
exten => _X.,1,Dial(SIP/ssvl/${EXTEN},20,tr)    ; Dial ${EXTEN}@ssvl.kth.se
```

11.3 Appendix 3: Implementation Reference Diagram

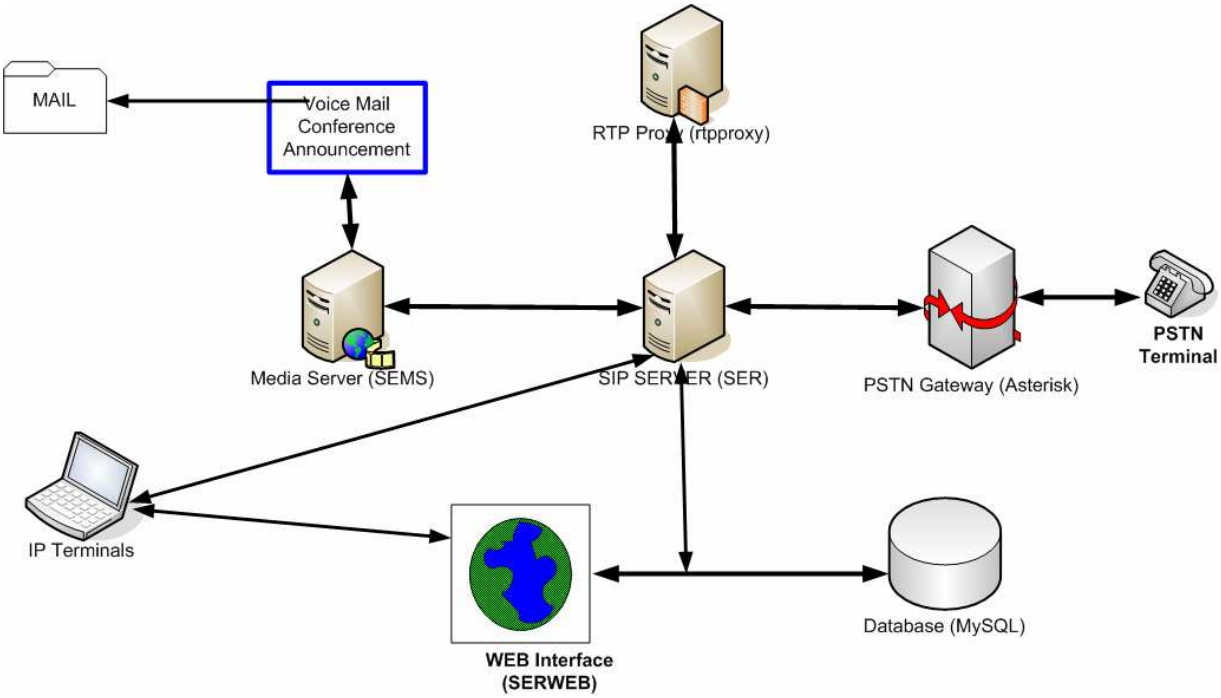

Figure 13: KTH-TSLab Implementation Overview