Design of PSTN - VoIP Gateway for rural environments

Dissertation

submitted in partial fulfillment of the requirements
for the degree of
Master of Technology

by

Sravana Kumar K
(Roll no. 05329034)

under the guidance of
Prof. Sridhar Iyer

Dept. of Computer Science and Engineering (KReSIT)
Indian Institute of Technology Bombay

2007

Dissertation Approval Sheet

This is to certify that the dissertation entitled

Design of PSTN - VoIP Gateway for rural environments

by

Sravana Kumar K

(Roll no. 05329034)

is approved for the degree of **Master of Technology**.

____

Prof. Sridhar Iyer

(Supervisor)

____

Prof. Anirudha Sahoo

(Internal Examiner)

____

Dr. Vijay T. Raisinghani

(External Examiner)

____

Prof V. M. Gadre

(Chairperson)

Date: ____________

Place: ____________

INDIAN INSTITUTE OF TECHNOLOGY BOMBAY

CERTIFICATE OF COURSE WORK

This is to certify that **Mr. Sravana Kumar K** was admitted to the candidacy of the M.Tech. Degree and has successfully completed all the courses required for the M.Tech. Programme. The details of the course work done are given below.

Sr.No.	Course No.	Course Name	Credits
		Semester 1 (Jul – Nov 2005)	
1.	HS699	Communication and Presentation Skills (P/NP)	4
2.	IT601	Mobile Computing	6
3.	IT605	Computer Networks	6
4.	IT608	Data Warehousing and Data Mining	6
5.	IT619	IT Foundation Laboratory	8
6.	IT623	Foundation course of IT - Part II	6
		Semester 2 (Jan – Apr 2006)	
7.	HS700	Applied Economics (Institute Elective)	6
8.	IT628	Information Technology Project Management	6
9.	IT680	Systems Lab	6
10.	IT694	Seminar	4
		Semester 3 (Jul – Nov 2006)	
11.	IT620	New Trends in Information Technology	6
12.	IT625	Information and Communication Technologies	6
		M.Tech. Project	
13.	IT696	M.Tech. Project Stage - I (Jul 2006)	18
14.	IT697	M.Tech. Project Stage - II (Jan 2007)	30
15.	IT698	M.Tech. Project Stage - III (Jul 2007)	42

I.I.T. Bombay   Dy. Registrar(Academic)

Dated:

Acknowledgements

I take this opportunity to express my sincere gratitude to **Prof. Sridhar Iyer** for his constant support and encouragement. His excellent guidance has been instrumental in making this project work a success. And special thanks to **Prof. Anirudha Sahoo** for his co-guidance.

I would like to thank **Janak** and **Priyesh** for their constant help throughout the project. I would also like to thank my colleagues in our Lab for being supportive to my project. Special thanks to **Kushal** for being a supportive friend and the KReSIT department for providing me world class computing infrastructure.

I would also like to thank my **family** and **friends** especially the entire **M.Tech Batch**, who have been a source of encouragement and inspiration throughout the duration of the project.

Last but not the least, I would like to thank the entire KReSIT family for making my stay at IIT Bombay a memorable one.

Sravana Kumar K
I. I. T. Bombay
July 9th, 2007

Abstract

A Voice over IP(VoIP) gateway is one of the various components used in realizing the convergence of data and telephone networks. It enables voice communication between callers on the IP network and the Public Switched Telephone Network(PSTN). There are many vendors that provide VoIP-PSTN gateways although most of them are very expensive. Soft PBX is a software-based PBX solution using VoIP technology. Asterisk is an example of a Soft PBX. If we want to maintain a VoIP-based telephone exchange, the cost of the entire system proves to be very expensive and it is not suitable for rural deployment. Also most of the devices require AC power which is not easily available in rural areas. There is currently no device in the market which provides the functionalities of both gateway and PBX and yet operates on DC power. Our aim is to design such a box which performs as a VoIP gateway with inbuilt Asterisk PBX and at the same time consume less power and be relatively inexpensive.

Asterisk has its own authentication for SIP users. Generally, large organizations and universities maintain some external authentication for their employees/students such as LDAP. Many of these organizations provide VoIP telephony facilities to their people. So, for unique authentication of users, there is a need to provide an external authentication mechanism in Asterisk, which can interface with the authentication system in use at the organization. So, one more aim of this project is to integrate Asterisk with LDAP.

Contents

List of Figures

List of Tables

Abbreviations

PoP	: Point of Presence
ARPU	: Average Revenue Per User
DoT	: Department of Telecommunications
VPT	: Village Public Telephone
PSTN	: Public Switched Telephone Network
VoIP	: Voice over Internet Protocol
LDAP	: Lightweight Directory Access Protocol
PBX	: Public Branch eXchange
FXO	: Foreign Exchange Office
FXS	: Foreign Exchange Station
ATA	: Analog Telephone Adapter
PCI	: Peripheral Component Interconnect
PC	: Personal Computer
DSL	: Digital Subscriber Line
IVR	: Interactive Voice Responce
MODEM	: MOdulator and DEModulator
SIP	: Session Initiation Protocol
SDP	: Session Description Protocol
API	: Application Programming Interface
IDE	: Integrated Development Environment
POTS	: Pain Old Telephone Service
PAM	: Pluggable Authentication Module

Chapter 1

Introduction and Motivation

Around 70% of India's population lives in villages. These villages typically consist of around 250 households and are located between two to three kilometers from each other. There are large towns located every 30 to 40 kilometers, which are served by optical fiber backbone. These act as a Point of Presence (PoP) for telecommunication for surrounding villages, with each PoP serving around 250 to 300 villages. The problem of connecting these PoPs to the villages can be regarded as the problem of *last mile* rural connectivity [1].

1.1 Last mile rural connectivity

Cellular coverage around the towns typically extends only for about five kilometers. This is because traditional cellular coverage proves too expensive to connect all villages to their nearest PoP. Income levels for rural India are lower than the national average. Average monthly income in rural India is around INR 2500, and it is estimated that the average household can spend less than INR 100 per month on telecom services. Since Average Revenue Per User (ARPU) from villages is too low to recover infrastructure and service costs, cellular coverage in rural India is negligible at present [2]. The key to Internet and telephony access for rural India is a public kiosk which provides a basket of services. A kiosk can expect one or two voice calls to be made on a continuous basis, and most of a kiosk's income is expected to be from voice traffic. Assuming two kiosks per village, the revenue of each can only be of the order of INR 5000 per month. Most existing initiatives for the last mile focus on extending reach from the PoP to the village kiosk. The Department of Telecommunications (DoT), through its Village Public Telephone (VPT) scheme, aims to have at least one telephone installed in each of approximately six lakh villages identified in the 2001 census [3]. As of August 2005, VPTs were deployed in 83.3% of the targeted villages [4]. The next phase involves installing a second telephone in villages with a population over 2000. Besides DoT and TRAI (Telecom Regulatory Authority of India) initiatives, WLL (Wireless in Local Loop) solutions using corDECT and WiFiRe, designed to operate in the unlicensed band, and the Digital Gangetic Plain project using multi-hop 802.11 links with directional antennas are recent attempts to address the last mile issue.

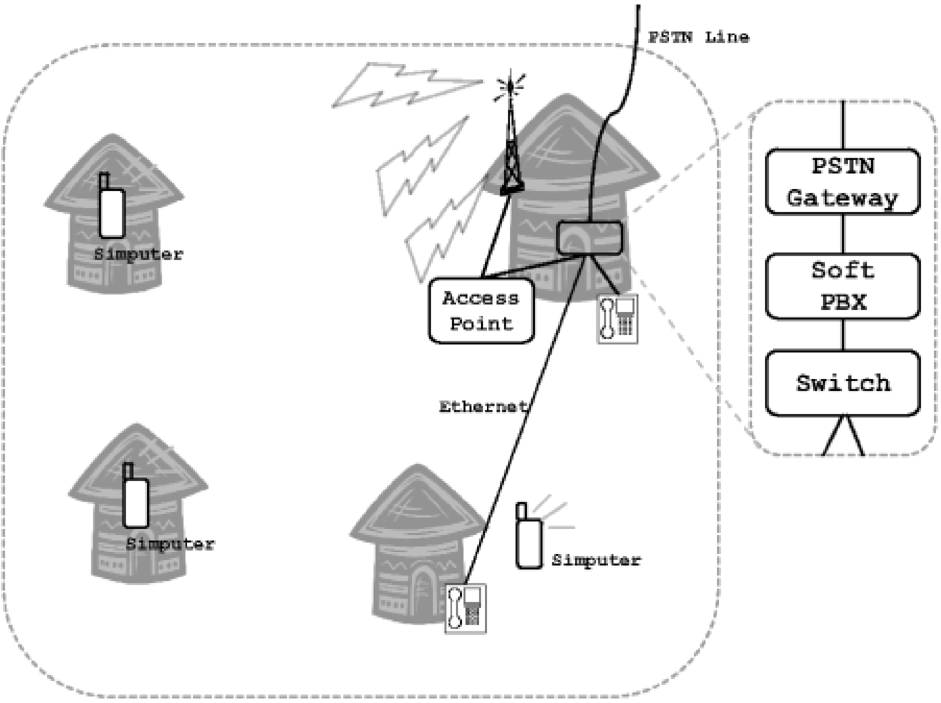


Figure 1.1: Intra-village connectivity: architecture

1.2 Intra-village connectivity

Subsequent to our experiments for intra-village connectivity, we address what we term the *intra-village connectivity* issue for rural connectivity: given that one telephone line reaches a village by either wired or wireless means, how can we extend the reach further within the village? An acceptable solution would ideally be:

- easy to deploy and maintain,
- using off-the-shelf components,

- low on operational expenditure, and
- low on power consumption.

We proposed an architecture based on Wi-Fi and Ethernet for intra-village connectivity, which we will explain in the Chapter 3. We also believe it is easy to deploy, maintain, and replicate in other rural scenarios.

1.3 Problem Statement

We found that Ethernet is suitable for connecting the gateway to places that are nearby, with distances in the tens of meters. Beyond this, repeaters are required and the physical security of the cable is difficult to ensure. The system cost in general is determined by the cost of the hardware running the exchange, the gateway, the network infrastructure, and the number of client devices deployed. We ran the software exchange (Asterisk) on a fully configured system. This has two constraints: i) the location of the software exchange is constrained by the lack of AC power, and ii) such a setup is not cost effective.

We have focused on solving the following problems:

- we have to design a single integrated DC-powered device that combines the Gateway and the Soft PBX
- setup Asterisk to authenticate users using OpenLDAP

1.4 Thesis Outline

Chapter 2 discusses various kinds of hardware and software used for VoIP-PSTN integration. In Chapter 3, we present a case study of Timbaktu, which includes an architecture based on Wi-Fi and Ethernet for intra-village connectivity and the performance of Asterisk PBX in wired and wireless media. Chapter 4 discusses different experiments which we have done to achieve feasible and affordable solution for rural connectivity, detailed explanation of affordable solution. Chapter 5 discusses our integration of LDAP authentication in Asterisk. It includes an overview of the registration process, and discusses how we have used PAM to authenticate with LDAP. Finally, chapter 6 concludes this thesis with a pointer to areas where further work is required.

Chapter 2

Survey of Hardware and Software components

In this chapter, we discuss various hardware devices and software components we have used in our work.

2.1 Hardware

First, we introduce some basic terminology required to understand VoIP hardware. FXO(Foreign eXchange Office) and FXS(Foreign eXchange Station) are basic kinds of interfaces provided by VoIP gateways. An FXO interface lets a gateway appear like a calling device to a PSTN telephone exchange, while an FXS interface lets a gateway appear like a telephone exchange to calling devices.

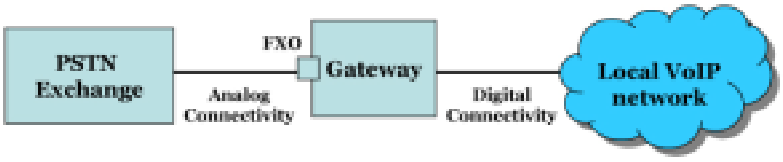


Figure 2.1: FXO interface

We have performed a number of experiments with different hardware and software in our search for an affordable VoIP gateway solution. The hardware devices which we have investigated are Sipura SPA-3000, Linksys SPA-3102, Linksys PAP2 ATA, Digium X100P PCI card, and Intel V.92 MODEM. Other hardware we needed for our experiments includes a fully configured PC, a VIA motherboard, and a 40-pin IDE flash memory.

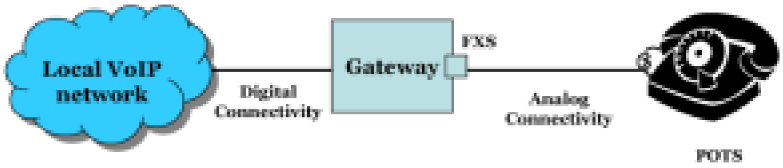


Figure 2.2: FXS interface

2.1.1 Sipura SPA-3000 VoIP gateway

The Sipura SPA-3000 [5] is a small light plastic unit designed to provide VoIP capabilities. In technical terms, this box has both an FXS and an FXO interface — the FXS interface allows a normal telephone to be turned into an IP phone and the FXO interface provides connectivity to a PSTN line. These interfaces can be configured independently using the on board web interface. When we log in as an admin user and switch to advanced mode, we can create SIP users and extensions, etc.

Figure 2.3: Sipura SPA-3000 Gateway

Once setup and working, apart from allowing us to make VoIP calls, the SPA-3000 provides the following functions:

- *A PSTN to VoIP gateway*: This allows us to call in over our PSTN phone line and make a VoIP call.
- *A VoIP to PSTN gateway*: We can call in over VoIP and make a call over our PSTN phone line.
- *Power Cut Protection*: If the SPA-3000 looses either the power or it's network connection, it can be configured to directly connect the two interfaces together - so

the phone will be effectively directly connected to the phone line so we can still make calls over our phone line during a power cut as if the device were not connected.

- *VoIP Calls*: We can receive calls to our VoIP address (or number) on the same phone as we can make and receive phone calls on our PSTN line.
- *Remote PSTN Line*: We can configure multiple Sipura units to allow us to access our PSTN line in a certain location while we are in a different building or even country with just an internet connection.
- *Asterisk*: If we require more flexibility or features, we can use the SPA-3000 as an FXS and FXO interface to the popular Asterisk open source PBX system. We will explain this in detail in chapter 4.

2.1.2 Linksys SPA-3102 VoIP gateway

Figure 2.4: Linksys SPA-3102 gateway

The Linksys SPA-3102 [6] is another VoIP gateway. It provides the same functionality as the Sipura SPA-3000. It has one FXO port and one FXS port.

If power and/or IP network connectivity is lost to the unit or the VoIP service is down, calls can be sent to a traditional carrier via the FXO interface. This device is similar in features to the Sipura SPA-3000, but is less expensive.

2.1.3 Linksys PAP2 ATA

The Linksys PAP2 is an Analog Telephone Adapter which enables high-quality feature-rich telephone service through ethernet cable or DSL Internet connection. It has two FXS interfaces where we can connect existing phones. Each phone jack operates independently,

Figure 2.5: Linksys PAP2 Analog Telephone Adapter

with separate phone service and phone numbers — like having two phone lines. It can be configured either through the analog telephone via voice prompts or through a web-based configuration (the adapter has a built-in web server). It supports SIP and supports multiple voice compression techniques. It supports all standard enhanced call features such as Caller ID, Call Waiting, Voicemail, Call Forwarding, Distinctive Ring, etc.

2.1.4 Digium X100P PCI card

The X100P is the standard single port FXO interface for Asterisk. It provides a single, full featured FXO interface for connecting the open source Asterisk PBX server to the PSTN.

Figure 2.6: X100P PCI card from Digium

Digium X100P allows Asterisk to make calls to or receive calls from a traditional analog phone line. The X100P is affordable and ideal component for building Interactive

Voice Response (IVR) and voicemail applications.

It supports all standard enhanced call features including Caller ID, Call Conferencing, and Call Waiting. By combining the X100P and the Asterisk PBX, one can easily and economically implement a sophisticated call center.

2.1.5 V.92 Data MODEM

MODEM includes Modulator and Demodulator. A MODEM is a device that enables a computer to transmit data over, for example, telephone or cable lines. Computer information is stored digitally, whereas information transmitted over telephone lines is transmitted in the form of analog waves. A MODEM converts between these two forms. V.92 internal data MODEM is PCI card. It has FXO port to connect PSTN network. It looks identical to Digium X100P card, but it doesn't have any VoIP gateway functionality. The Zaptel drivers for X100P can be modified to work with this card, so that it can work as a VoIP gateway. We will explain in detail in chapter 4.

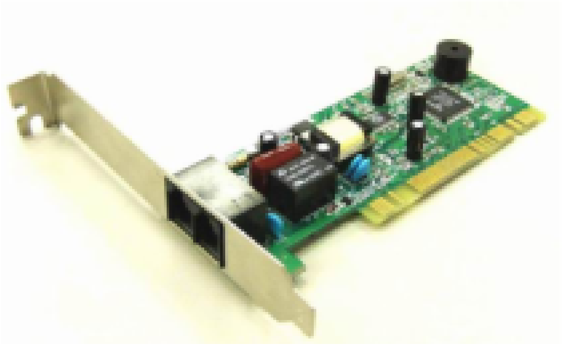


Figure 2.7: V.92 Internal Data MODEM

2.1.6 VIA motherboard

We used PC1500 VIA motherboard[7], because of its compact size and low power consumption. It has one PCI slot where we can fix either Digium X100P card or V.92 internal data MODEM. Instead of using an SMPS as used in normal PC, it uses 12V DC adapter which further reduces power consumption. It is an effective motherboard to port AstLinux on it.

Figure 2.8: Effective motherboard from VIA: VIA PC1500

2.1.7 IDE Flash Memory

We used IDE flash memory instead of a hard disk because of its much lower power consumption and higher reliability. Flash memory is many times more reliable than hard drives due to the lack of moving parts, etc. However, flash memory has a limited number of write cycles, so extra care has to be taken when running software from compact flash cards.

Figure 2.9: 40-pin IDE Flash Memory

2.2 Software

In this section, we summarize the various software we have used as part of this work, such as Asterisk PBX, AstLinux and Softphones.

2.2.1 Soft PBX

As our Soft PBX, we have used Asterisk [8]. With Asterisk, we can easily build our own multiprotocol VoIP PBX on Linux. Asterisk supports many VoIP protocols, such as SIP, SDP, and MGCP, and can interoperate with almost all standards-based telephony equipment using relatively inexpensive hardware.

Asterisk needs no additional hardware for VoIP. For interconnection with digital and analog telephony equipment, Asterisk supports a number of hardware devices, most notably all of the hardware manufactured by Asterisk's sponsors, Digium [8].

2.2.1.1 Asterisk base components

Asterisk is composed of the following major components [9]:

- *Dynamic Module Loader*: When Asterisk is first started, the Dynamic Module Loader loads and initializes each of the drivers which provide channel drivers, file formats, call detail record backends, codecs, applications, etc., linking then with the appropriate internal APIs.

- *PBX Switching*: The essence of Asterisk is a Private Branch Exchange Switching system, connecting calls together between various users and automated tasks. The Switching Core transparently connects callers arriving on various hardware and software interfaces.

- *Application Launcher*: Launches applications which perform services for user, such as voicemail, file playback and directory listing.

- *Codec Translator*: Codec Translator uses codec modules for the encoding and decoding of various audio compression formats used in the telephony industry. A number of codecs are available to suit diverse needs and arrive at the best balance between audio quality and bandwidth usage.

- *Scheduler and I/O Manager*: It handles low-level task scheduling and system management for optimal performance under all load conditions.

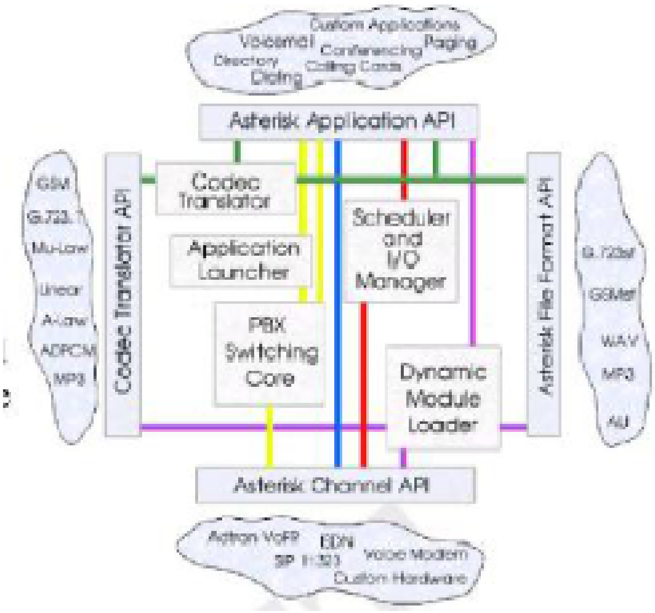


Figure 2.10: Asterisk API architecture

2.2.1.2 Loadable Module APIs

Four APIs [9] are defined for loadable modules, facilitating hardware and protocol abstraction. Using this loadable module system, the Asterisk core is independent of the details of how a caller is connecting, what codecs are in use, etc.

- *Channel API*: The channel API handles the type of connection a caller is arriving on, be it a VoIP connection, ISDN, PRI, Robbed bit signaling, or some other technology. Dynamic modules are loaded to handle the lower layer details of these connections.

- *Application API*: The application API allows for various task modules to be run to perform various functions. Conferencing, Paging, Directory Listing, Voicemail, In-line data transmission, and any other task which a PBX system might perform now or in the future are handled by these separate modules.

- *Codec Translator API*: It loads codec modules to support various audio encoding and decoding formats such as GSM, Mu-Law, A-law, and even MP3.

- *File Format API*: It handles the reading and writing of various file formats for the storage of data in the file system.

The following section is a short summary of the protocols that Asterisk supports.

2.2.1.3 Protocols

Session Initiation Protocol: SIP (Session Initiation Protocol) is an application-layer control protocol that can establish, modify, and terminate multimedia sessions such as Internet telephony calls (VoIP). SIP can also invite participants to already existing sessions, such as multicast conferences. Media can be added to (and removed from) an existing session. SIP transparently supports name mapping and redirection services, which supports personal mobility - users can maintain a single externally visible identifier regardless of their network location.

SIP supports five facets of establishing and terminating multimedia communications:

- *User location*: determination of the end system to be used for communication;
- *User availability*: determination of the willingness of the called party to engage in communications;
- *User capabilities*: determination of the media and media parameters to be used;
- *Session setup*: "ringing", establishment of session parameters at both called and calling party;
- *Session management*: including transfer and termination of sessions, modifying session parameters, and invoking services;

SIP is a component that can be used with other IETF protocols to build a complete multimedia architecture, such as the Real-time Transport Protocol (RTP) for transporting real-time data and providing QoS feedback, the Real-Time streaming protocol (RTSP) for controlling delivery of streaming media, the Media Gateway Control Protocol (MEGACO) for controlling gateways to the Public Switched Telephone Network (PSTN), and the Session Description Protocol (SDP) for describing multimedia sessions. Therefore, SIP should be used in conjunction with other protocols in order to provide complete services to the users. However, the basic functionality and operation of SIP does not depend on any of these protocols.

Session Description Protocol: The Session Description Protocol (SDP) describes multimedia sessions for the purpose of session announcement, session invitation and other forms of multimedia session initiation. Session directories assist the advertisement of conference sessions and communicate the relevant conference setup information to prospective participants. SDP is designed to convey such information to recipients. SDP is purely a format for session description - it does not incorporate a transport protocol, and is intended to use different transport protocols as appropriate including the Session Announcement Protocol (SAP), Session Initiation Protocol (SIP), Real-Time Streaming Protocol (RTSP), electronic mail using the MIME extensions, and the Hypertext Transport Protocol (HTTP).

The SDP text messages include:

- session name and purpose,
- time the session is active,
- media comprising the session, and
- information to receive the media (address etc.).

2.2.2 AstLinux

AstLinux [10] is a custom Linux distribution centered around Asterisk. The main issue with normal Asterisk is that without a kernel and other base software (a distribution) installed, it doesn't do anything. It is an application and we need much more than an application for an actual working telephony solution. The existing approach to this was to install a generic, general-purpose Linux distribution and to install Asterisk. This approach has several problems. The biggest of these problems is bloat. To have a complete Asterisk solution using a general purpose distro, we will need to know a great deal about Linux, Asterisk, and network protocols.

AstLinux has been designed with running from Compact Flash memory in mind. However, it can also be installed to a regular hard drive like other distributions. AstLinux currently occupies around 40mb of disk space, so any Compact Flash card (or hard drive) of 64mb or larger will work perfectly.

2.2.3 Softphones

A softphone is a software for making telephone calls over the Internet using a general purpose computer, rather than using dedicated hardware. Often a softphone is designed to behave like a traditional telephone, sometimes appearing as an image of a phone, with a display panel and buttons with which the user can interact. A softphone is usually used with a headset connected to the sound card of the PC. A softphone connects to a PBX through a LAN. Some examples of softphones are SJPhone, x-lite, etc.

Chapter 3

Timbaktu Case-study

3.1 Timbaktu Collective

The Timbaktu Collective [11] is a voluntary organization working for sustainable development in the drought prone Anantapur district of Andhra Pradesh. There are about a dozen homes, a school, a community kitchen, and a few administrative buildings within a one km radius as shown in Figure 3.1.

3.2 Problem in Communication

Because of its remote location and mountainous area, Timbaktu lacks cellular coverage and its connectivity is through a single PSTN line. This phone is currently located in the community kitchen. The phone sees frequent use(mostly incoming), and each time a user needs to walk to the kitchen to make or receive a call. The remote location of Timbaktu prevents deployment of additional lines from the nearest switch. It is located in a place surrounded by hills, and this eliminates cellular coverage as an alternative. So, single telephone line is used to provide telephone connectivity to all users at their residence. Timbaktu thus becomes a good case study for our architecture. However, the area provides a challenging environment for wireless deployment. The village has dense foliage and lacks an external electricity supply. Solar panels (DC power) are installed to meet power requirements.

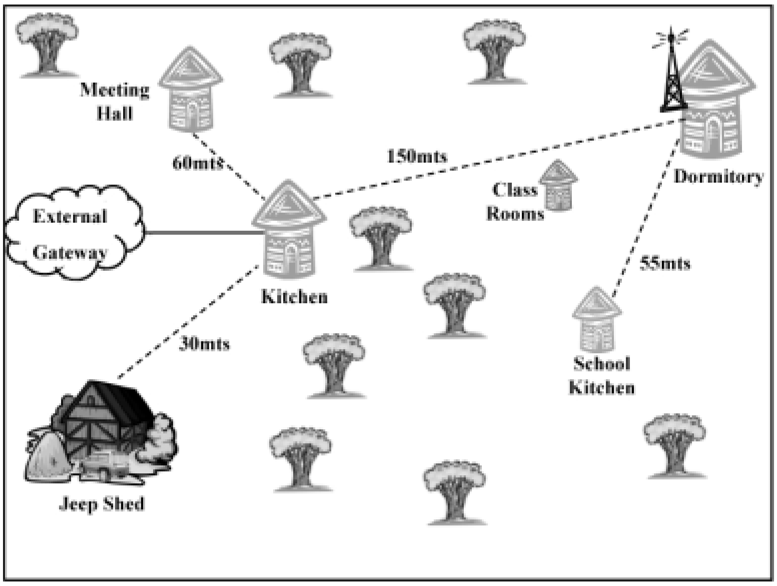


Figure 3.1: Timbaktu layout

3.3 Proposed Architecture

We proposed an architecture which solves the Timbaktu communication problem [12]. We will start by discussing the software exchange, VoIP gateway and communication devices that we used in our village architecture.

- ***Software exchange***: We chose the Asterisk Soft PBX [8] for reasons which we have summarized in section 2.2. The Asterisk server was setup on a Intel laptop with a wireless card running Fedora core 3. Asterisk installation and configuration details are given in [13].

- ***VoIP-PSTN gateway***: The PSTN line requires an interface to communicate with the software exchange. This PSTN gateway allows calls to be made or received from a landline to any user on the network. We used the Sipura SPA-3000 VoIP gateway for this purpose.

- ***Communication Devices***: An analog phone connection was provided using a Linksys Analog Phone Adapter (ATA PAP2). The adapter allows the use of the phone in a VoIP-enabled network. The adapter was connected directly to the AP through the Ethernet port. We also set up the Asterisk server to translate numbers dialed from the phone into SIP usernames. As our Wi-Fi handheld we chose the

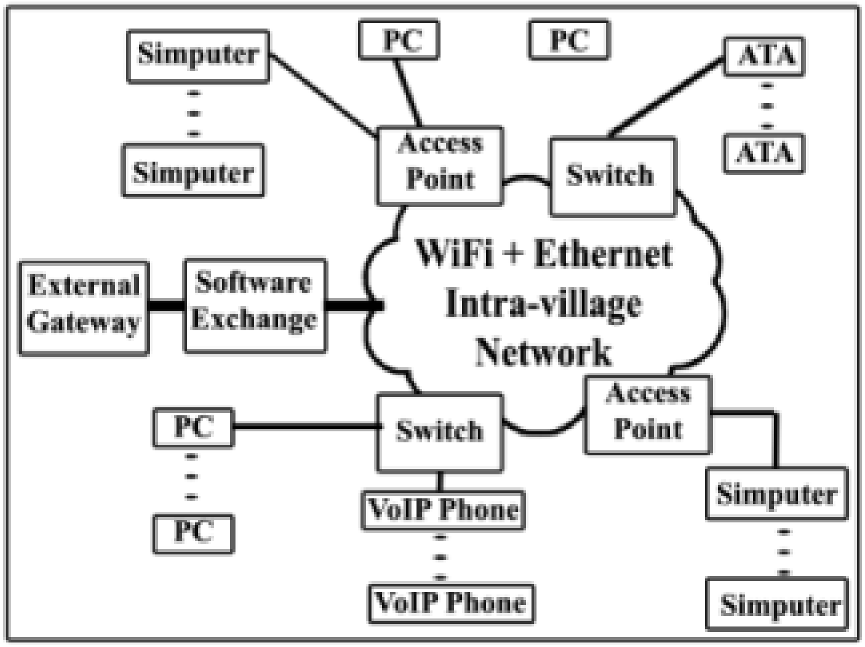


Figure 3.2: Intra-village connectivity: Architecture

Simputer. Simputer is a low-cost handheld running on open source softwares. It has a Strong Arm SA-1100 RISC CPU with 32 MB of DRAM. It runs Linux kernel 2.4 and has a SIP-based VoIP client. We used a 3Com USB wireless adapter to enable wireless connectivity on the Simputer.

3.3.1 Architectural details

Our aim in our case-study was to allow people of Timbaktu to make and receive calls from multiple locations using the single PSTN line. We do this by using existing computers at the school and residences as communication devices. To connect the PSTN line to locations close to the kitchen, we established an Ethernet network to carry VoIP connections. We also used hand-held PDAs (Simputers) as mobile clients. As proof of concept for the mobile devices, we installed an Access Point (AP) at the dormitory (where the solar panels are located) and established VoIP calls to Simputers located at various distances up to the kitchen.

We set up the wireless network in Figure 3.3 as follows:

1. Established location and installed AP and antenna.

2. Perform RF measurements.

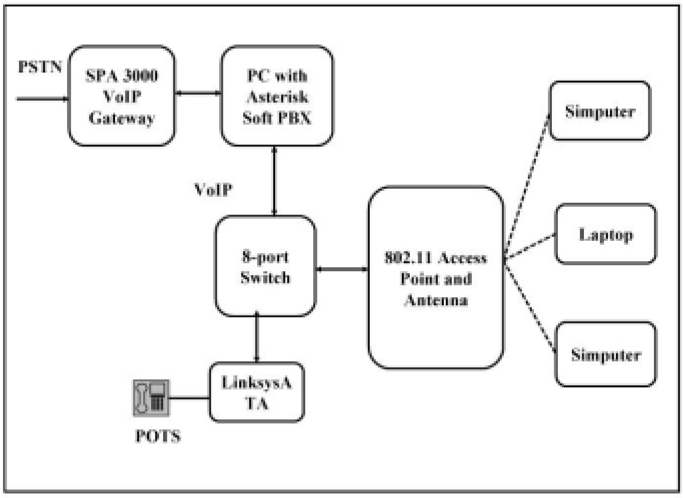


Figure 3.3: Case-study setup

3. Setup the software PBX and interface it with the AP.
4. Setup the analog phone and wireless handheld.
5. Perform VoIP tests using different client devices.

3.4 VoIP Tests

The system deployment architecture is shown in Figure 3.2. We used multiple laptops, Simputers and analog phones to perform our VoIP tests. We evaluated the voice quality over different combinations of clients and at different locations. Calls between external landlines and the local devices (Simputers, analog phones and laptops) were tested. Intra-village calls between devices in the network were also evaluated. The clarity of a voice call was used to define the quality of the call. Table 3.4 summarizes the voice quality for various configurations. Landline calls were made and received from various devices.

3.5 Case-study results

Quality of the calls were very good except with the Simputer. Calls between landline and Simputer were of average quality. For intra-village connectivity, we found that we were able to do laptop to laptop calls quite easily over VoIP plus Wi-Fi. The quality was good and we had also used this mode of operation to do various tests about usable

Device (From/To)	PC	Simputer	Phone	Landline
PC	Very good	Poor	Good	Very good
Simputer	Average	Poor	Poor	Average
Phone	Good	Average	Good	Very good
Landline	Very good	Average	Very good	-

Table 3.1: VoIP quality test results.

wireless ranges. We were able to do laptop to Simputer and vice versa too. The audio from Simputer to laptop was clear, but laptop to Simputer audio was quite poor.

We suspect this is because of processing delays and codec problems at the Simputer end. Simputer to Simputer voice calls were very difficult to do over Wi-Fi. This was the case when two Simputers were connected through an AP, and also when both were connected directly in adhoc mode. We also managed to connect from the phone to the Simputer and get a clear call through over VoIP.

3.6 Measuring Asterisk performance in wired and wireless media

We have done some experiments to measuring Asterisk performance in wired and wireless media. We used `callgen` module [14] to get the Asterisk performance. `callgen` is one of the component in `yate`(Yet Another Telephony Engine) Software. (Figure 3.4) shows the basic setup which we have used for our experiment. The results that we have got in our experimenting with wired and wireless media are presented here. Table 3.6 shows the Response Time and Answer Time of Asterisk. Response Time means time taken by Asterisk to respond to client request. Answer Time means time taken by Asterisk to fulfil the client request. Figure 3.5 shows the Asterisk Response Times in wired and wireless media. From the graph, we can see that as the number of parallel calls in wireless medium increases, the packet loss and error rate increase enormously. But, in the wired medium there isn't much packet loss. So, it is necessary to improve the SIP protocol to get better performance from Asterisk in wireless medium.

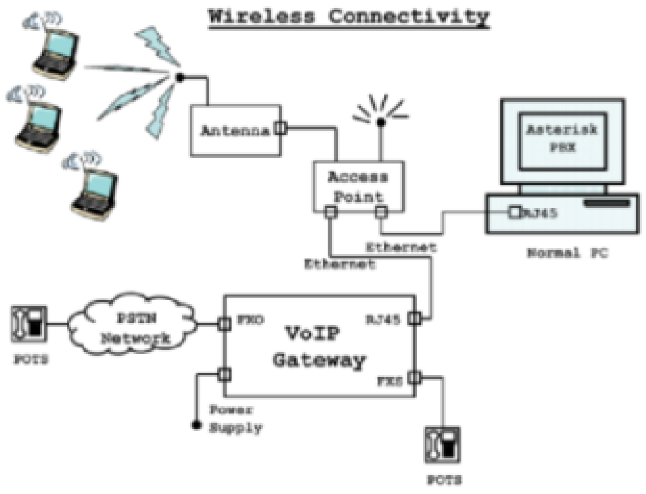


Figure 3.4: Wireless architecture for Asterisk testing

	Response Time				Answer Time			
No. of Calls	100	1000	10000	20000	100	1000	10000	20000
Wired(in ms)	12231	12257	15225	17820	12415	12366	15406	31646
Wireless(in ms)	94785	167219	184237	195117	95072	167361	184359	195874

Table 3.2: Asterisk Performance in Wired and Wireless media

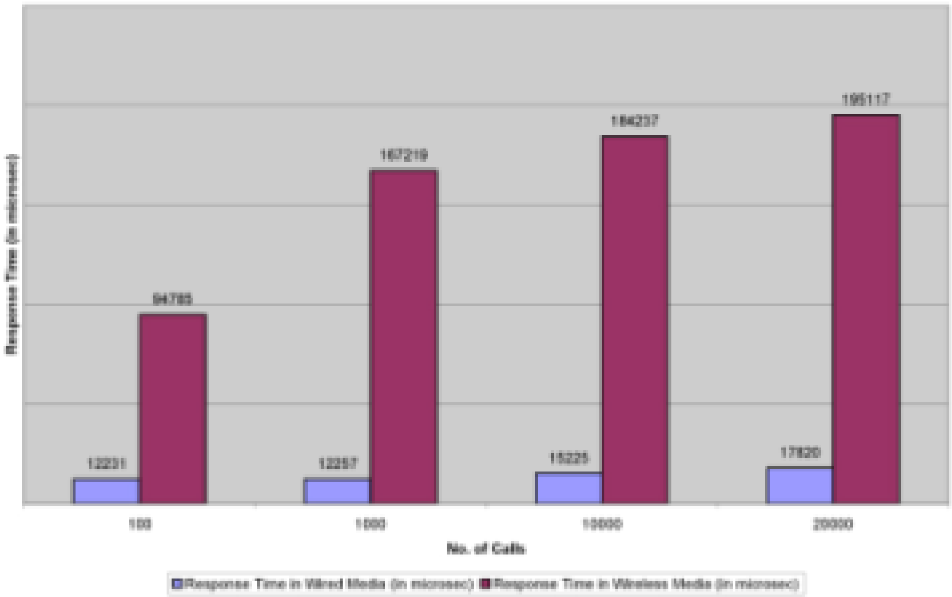


Figure 3.5: Asterisk Response Times in wired and wireless media

Chapter 4

Affordable solution for PSTN-VoIP integration

We took a look at various devices that can be used to build VoIP gateways. To design an affordable VoIP-PSTN gateway, we have performed various experiments with these devices.

4.1 Experiments

Here we have described the setup of the experiments and presented a comparison of their cost, and advantages & disadvantages.

1. ***Sipura SPA-3000 with Normal PC***: We performed the first experiment with a normal PC and Sipura SPA-3000. The Asterisk server is installed and configured on the computer system. SPA-3000 is the gateway that enables PSTN-VoIP integration. The SPA-3000 needs to be configured to work along with asterisk on the network.

   Advantages:

   - Sipura SPA-3000 has nice web interface, so it is easy to install and configure the system.

   Disadvantages:

   - It is most expensive setup of all the experiments that we have performed.
   - High power consumption because Asterisk is installed on a fully configured system. It is wastage of computing power.

2. ***Sipura SPA-3000 with VIA motherboard***: As in the previous solution we were using a costly motherboard and thus also were wasting computation resources, we replaced the processing unit with a inexpensive motherboard. We used VIA motherboard along with the SPA-3000 to build the system.

   Advantages:

   - VIA motherboard is in compact size and low power consuming device, so there is efficient usage of computational power.
   - VIA motherboard is inexpensive, so it reduces the overall system cost.

   Disadvantages:

   - The power consumption is still high because of the hard disk.

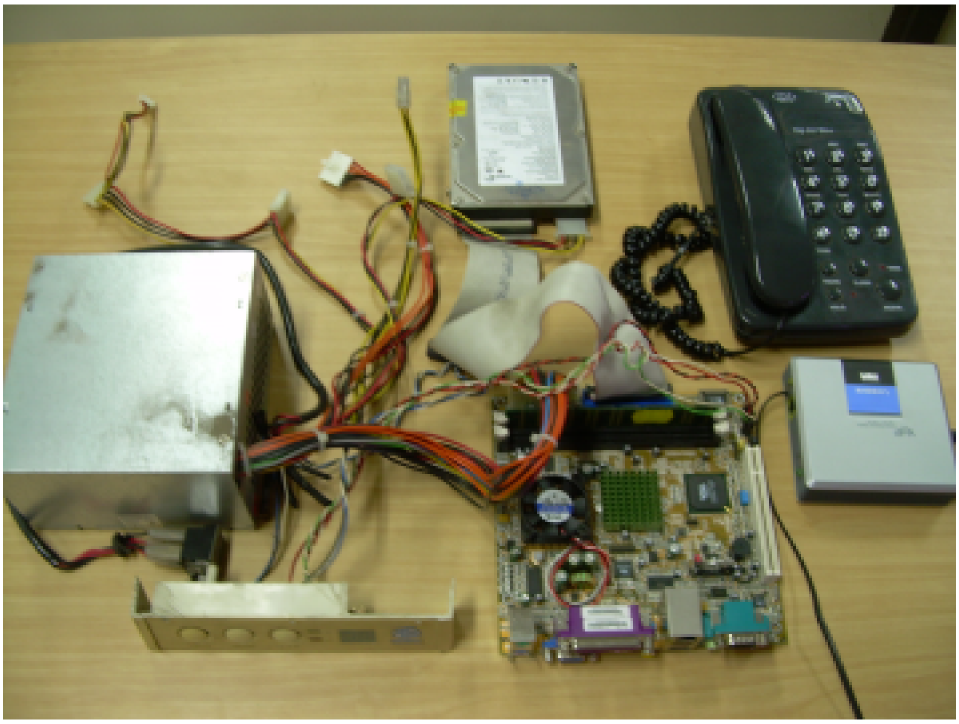


Figure 4.1: Experiment of SPA-3000 with VIA motherboard

3. ***Digium X100P with Normal PC***: Next we focused on reducing the cost of the gateway. We replaced the SPA-3000 with the Digium PCI card. The Digium card provides the functionality of the gateway, however we can not fine tune it like the SPA-3000.

   Advantages:

   - Asterisk provides us the Zaptel drivers to communicate with Digium card. We need to plug the card into PCI slot and install the Zaptel drivers, so no extra setup effort needed.
   - X100P cards are cheaper than Sipura SPA-3000, so the cost of gateway is reduced.

   Disadvantages:

   - No fine tuning of the system is possible, like in SPA-3000
   - High power consumption because of the use of normal PC.

4. ***Digium X100P with VIA motherboard***: In this, we have done the same experiment with VIA motherboard. We found that Digium X100P [15] works in VIA motherboard. This setup takes less power and is inexpensive.
   Advantages:

   - VIA motherboard consumes 12V DC power.
   - No extra setup effort for gateway.

   Disadvantages:

   - No fine tuning of the system is possible, because X100P doesn't have GUI capability to tune the system.

5. ***Normal Data MODEM with VIA motherboard***: Going further on the cost reduction issue we used the normal data MODEM in place of the Digium card. The Intel V.92 MODEM has very slightly different hardware from the X100P from Digium. This requires some code modification in the Asterisk's Zaptel driver's code.

   If we are compiling Asterisk from source, edit the zaptel/wcfxo.c file as follows [16]:
   Existing code:

   ```
   static struct pci_device_id wcfxo_pci_tbl[] __devinitdata = {
    { 0xe159, 0x0001, 0x8085, PCI_ANY_ID, 0, 0, (unsigned long)
       &wcx101p },
    { 0x1057, 0x5608, PCI_ANY_ID, PCI_ANY_ID, 0, 0, (unsigned long)
       &wcx100$ };
   ```

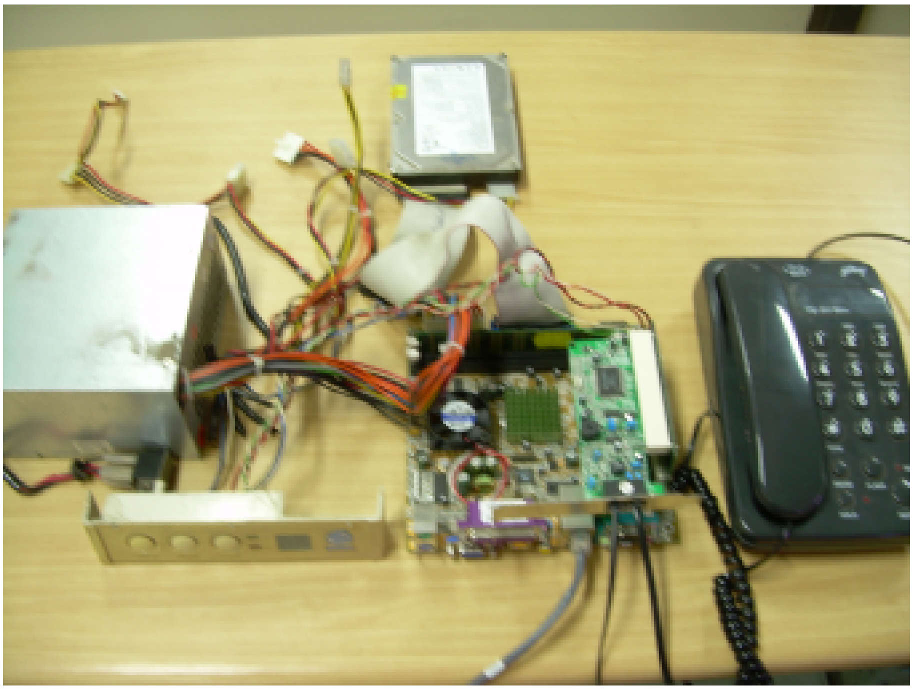


Figure 4.2: Experiment of X100P with VIA motherboard

Now change the wcfxo_pci_tbl[] in zaptel/wsfxo.c to:

```
static struct pci_device_id wcfxo_pci_tbl[] __devinitdata = {
 { 0xe159, 0x0001, 0x8085, PCI_ANY_ID, 0, 0, (unsigned long)
    &wcx101p },
 { 0xe159, 0x0001, 0x8086, PCI_ANY_ID, 0, 0, (unsigned long)
    &wcx101p },
 { 0x1057, 0x5608, PCI_ANY_ID, PCI_ANY_ID, 0, 0, (unsigned long)
    &wcx100$ };
```

The added line in the middle will allow the wcfxo driver to work with off the shelf Intel V.92 Winmodem cards as well as with the Digium X100P cards.

We also tried to reduce the power consumption of the system by replacing the hard disk with a 40-pin IDE flash. IDE flash is connected to the motherboard on its 40-pin slot used to connect hard-disk data bus.

Advantages:

- The cost of normal MODEM is less than X100P card, so the cost of system is reduced.

- We are making efficient resource utilization by using VIA motherboard.
- We replaced hard disk with IDE flash memory, so the system become compact in size.

Disadvantages:

- Code modification in Asterisk is required to make Asterisk work with data MODEM.
- Flash memory has less write cycles, so the life of the system is reduced.
- Flash memory provides slow access to data.

4.2 Conclusion: Affordable Solution to Gateway PBX

After we did the above list of experiments, we concluded that a better and feasible solution in order to reduce cost and power consumption for gateway PBX is V.92 Internal Data MODEM with VIA motherboard using 40-pin IDE flash memory. Table 4.2 shows the cost measurement of various experiments. The system cost includes Access Point, directional antenna, VoIP gateway, motherboard, normal POTS, ethernet cables. The approximate costs of the various components as of Dec 2006 are as follows: the cost of Access Point was Rs. 5,100, directional antenna cost was Rs. 8,250, Sipura SPA-3000 cost was Rs. 7,000, normal PC cost was Rs. 20,000, normal POTS cost was Rs. 500, X100P card cost was Rs. 2,500 , V.92 data MODEM cost was Rs. 500, VIA motherboard cost was Rs. 5,000, and IDE flash memory(2GB) cost was Rs. 1,000. Since Asterisk PBX is open source, there won't be any software cost in our proposed system.

Figure 4.3 shows our V.92 internal Data MODEM with VIA motherboard using flash memory experiment. From Table 4.2, our setup costs approximately Rs. 20,000. If number of clients to that system is 10, then each user has to pay approximately Rs. 2,000. If clients are 50, then each user has to pay approximately Rs. 400. So, server side cost depends on the number of users connected to our system. Suitable client side options are POTS with ATA, Simputer, Normal Desktop PC with softphone, Laptop with softphone or any other VoIP hard phone.

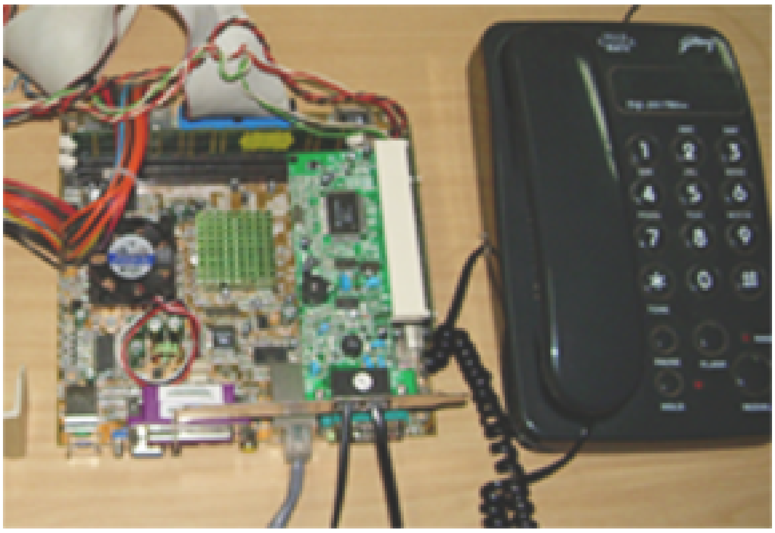


Figure 4.3: Feasible and affordable solution to VoIP Gateway PBX

S.No	*Server side solutions*	*System Cost*	*10 Users*	*50 Users*
1	Sipura SPA3000 with Normal PC	Rs. 41,350	Rs. 4,135	Rs. 827
2	Sipura SPA3000 with VIA motherboard	Rs. 26,350	Rs. 2,635	Rs. 527
3	Digium X100P with Normal PC	Rs. 36,850	Rs. 3,685	Rs. 737
4	Digium X100P with VIA motherboard	Rs. 21,850	Rs. 2,185	Rs. 437
5	V.92 Data MODEM with VIA motherboard	Rs. 19,850	Rs. 1,985	Rs. 397

Table 4.1: Cost comparison of various experiments

Chapter 5

LDAP Authentication in Asterisk

The SIP authentication process in Asterisk, as described in RFC 3310, relies on a challenge/response procedure similar to HTTP digest authentication. On a REGISTER request issued by a UAC(User Agent Client), the Registrar challenges the UAC, which registers again providing the challenge response.

We have a real interest in integrating an existing external authentication source with a SIP Registrar such as Asterisk, because organizations providing VoIP telephony facilities to their people require authentication of their users. By integrating an external authentication mechanism in Asterisk, it can interface with the authentication system in use at the organization.

In this chapter, we describe authentication of users using LDAP(Lightweight Directory Access Protocol). LDAP can store the users' passwords either in clear text or in hashed form.

5.1 Registration without an LDAP server

We describe in this section the general case, where the user provided credentials are checked against a local database by Asterisk, e.g. the configuration file `sip.conf`, in which each user has a matching entry. The user password can be stored either in clear text or in an hashed form, which is formed by calculating the MD5 hash of the username:realm:password chain. Here is an example of an entry for a user in the `sip.conf` file:

```
[username]
type=friend
context=from-sip
```

```
secret=secret
host=dynamic
```

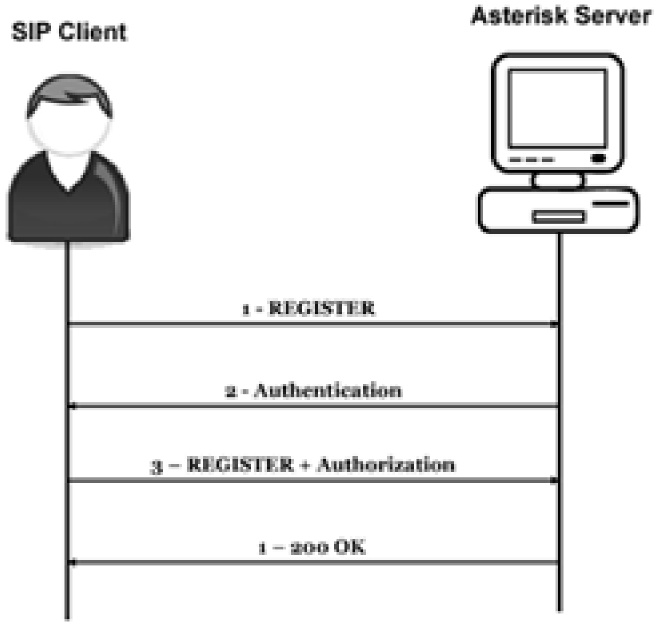


Figure 5.1: Registration in Asterisk without an LDAP server

Figure 5.1 shows a simple registration procedure where the authentication source resides in the SIP registrar itself. Whenever the client connects with the SIP registrar, the registrar will check the client credentials against the entries of `sip.conf`. This is how registration usually works in Asterisk. User credentials will be stored in a `sip.conf` file which is maintained by the Asterisk administrator.

5.2 Registration with LDAP authentication

In this section, we describe the LDAP server and the integration of LDAP with Asterisk using PAM module.

5.2.1 LDAP server

We used OpenLDAP as the LDAP server. OpenLDAP is a free, open source implementation of LDAP. LDAP is a platform-independent application protocol for querying and modifying directory services running over TCP/IP.

A directory is a set of information with similar attributes organized in a logical and hierarchical manner. The most common example is the telephone directory, which consists of a series of names (either of a person or an organization) organized alphabetically, with an address and phone number attached.

OpenLDAP has three main components:

- slapd: stand-alone LDAP daemon and associated overlays and tools,
- libraries implementing the LDAP protocol, and
- sample clients.

We installed and configured an OpenLDAP server. The installation process and configuration details are described in [17].

5.2.2 LDAP client

We chose to use a PAM module as the LDAP client. PAM stands for Pluggable Authentication Modules [18]. It is a library used to control the functions of various applications that have the capability to use the PAM libraries. It is based on a series of library modules, some of which depend on configuration files [19]. Locations of PAM configuration files and library modules are:

- All PAM applications are configured in the directory `‘‘/etc/pam.d’’` or in a file `‘‘/etc/pam.conf’’`.
- The library modules are normally stored in the directory `‘‘/lib/security’’`.
- The configuration files are located in the directory `‘‘/etc/security’’`.

Applications need explicit support for PAM in order to be able to use it for authentication. We patched Asterisk with `pam_ldap` module in order to get this functionality [20].

5.3 Integrating LDAP in Asterisk

Our goal here is to retrieve the user password stored in the LDAP server during the Authentication process.

During the *Authentication* process, Asterisk will connect to the configured LDAP server as an administrator and pulls the `userPassword` LDAP value for the specified user. The `userPassword` LDAP value can be either a clear text password, or a hashed value of username:realm:password.

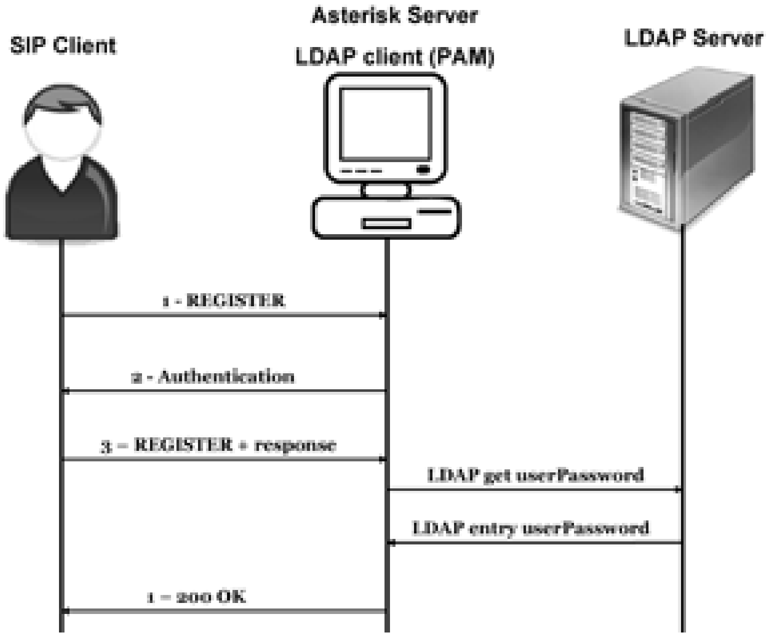


Figure 5.2: Registration in Asterisk with an LDAP server

Figure 5.2 shows the SIP registration procedure where the authentication source resides in an external LDAP server.

5.3.1 Implementation Details

In order to let Asterisk connect to the LDAP server, we added the LDAP module into the Asterisk source code. The main files comprising the LDAP module are `chan_sip.c` and `res_auth.c`. The functions added by this module are:

- *static int load_module(void)*:

  This function loads the LDAP configuration file and executes the *parse_config* function.

- *static int parse_config(void)*:

  This function parses the entire configuration file such as `auth.conf` and connects to the LDAP server.

- *static int ldap_connect(void)* :
  Connects to the configured LDAP server. It returns 1 if already connected, 0 on success, -1 on error.

- *int ast_parse_secret(const char *username, const char *secret, char *pwd)*:
  This function parses the secret configuration string for Asterisk resources. The user credentials format should be in the form :

  ```
  <auth_proxy>:[auth_db:[password]]
  ```

  'auth_proxy' indicates the actual location of the authentication process. Asterisk may retrieve user passwords from an external database depending on the value of 'auth_db', and then authenticate the users. We give pam as the value for this parameter.
  'auth_db' refers to the database/protocol where user information can be pulled from. We give file as the value for this parameter, and the relevant information is stored in sip.conf
  'pwd' where the function will store the parsed password string.

- *static int get_ldap_password(const char *username, char* password)*:
  This function finds the user password from an LDAP database based on the LDAP filter set in the auth.conf file. The arguments are username and password. Username is the username for which the password is searched and password is the string filled with the retrieved LDAP password attribute value. This function returns -1 if no entry was found.

- *static enum check_auth_result check_auth(...)*:
  Check user authorization from peer definition. Some actions, like REGISTER and INVITEs from peers require authorization (if the peer has a secret set). This function returns 0 on success, non-zero on error.

- *static int ldap_disconnect(void)*:
  Disconnects from the LDAP server.

To connect our Asterisk server to the LDAP server, we need to configure some files, which are sip.conf and extentions.conf.

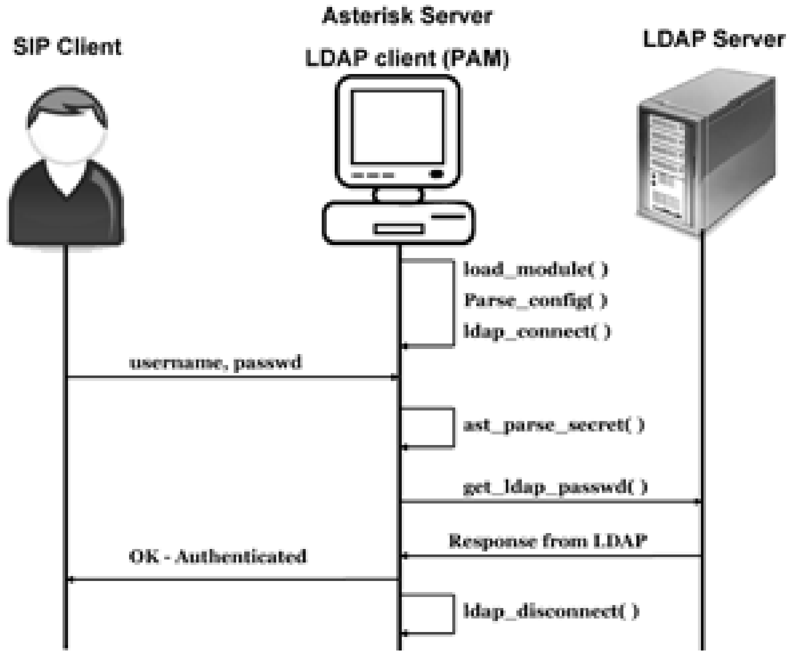


Figure 5.3: Sequence diagram of LDAP authentication in Asterisk

sip.conf: In our setup, the `sip.conf` file does not contain any secret (clear or hashed), and we add an attribute '`auth_type`' that specifies the type of authentication, set to `pam` as shown in the following fragment:

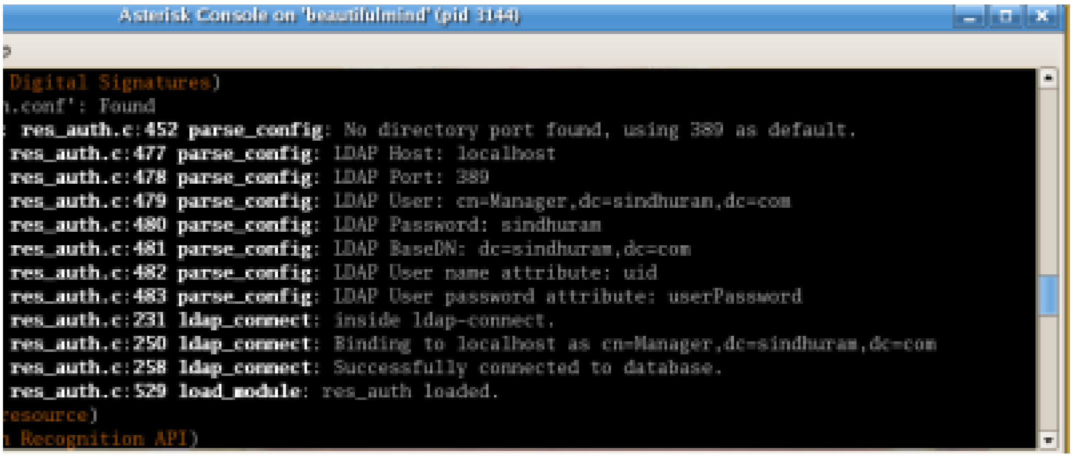
```
[username]
type=friend
context=from-sip
auth_type=pam
host=dynamic
```

extensions.conf: In this file, we give the phone extensions to the SIP users. So, this file contains dial plan of the PBX system. The example of `extensions.conf` is shown below:

```
[from-sip]
exten => 100,1,dial(SIP/sravana)
exten => 200,1,dial(SIP/janak)
```

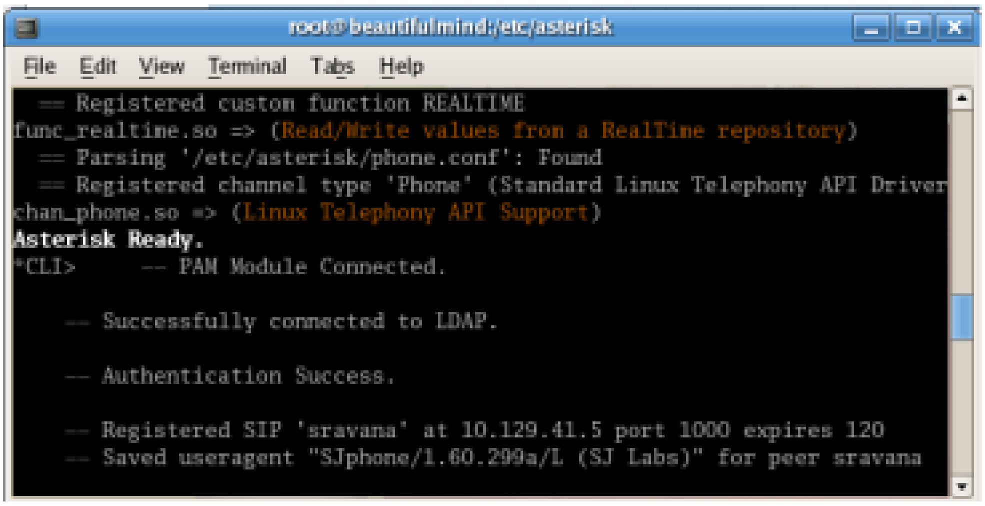
5.4 Results

The LDAP server must be started before running Asterisk. Asterisk could load all modules, including our LDAP module. Figure 5.4 shows how Asterisk PBX connects to the LDAP server.



```
 Digital Signatures)
n.conf': Found
: res_auth.c:452 parse_config: No directory port found, using 389 as default.
 res_auth.c:477 parse_config: LDAP Host: localhost
 res_auth.c:478 parse_config: LDAP Port: 389
 res_auth.c:479 parse_config: LDAP User: cn=Manager,dc=sindhuran,dc=com
 res_auth.c:480 parse_config: LDAP Password: sindhuran
 res_auth.c:481 parse_config: LDAP BaseDN: dc=sindhuran,dc=com
 res_auth.c:482 parse_config: LDAP User name attribute: uid
 res_auth.c:483 parse_config: LDAP User password attribute: userPassword
 res_auth.c:231 ldap_connect: inside ldap-connect.
 res_auth.c:250 ldap_connect: Binding to localhost as cn=Manager,dc=sindhuram,dc=com
 res_auth.c:258 ldap_connect: Successfully connected to database.
 res_auth.c:529 load_module: res_auth loaded.
esource)
n Recognition API)
```

Figure 5.4: Asterisk connects to LDAP server module



```
  == Registered custom function REALTIME
func_realtime.so => (Read/Write values from a RealTime repository)
  == Parsing '/etc/asterisk/phone.conf': Found
  == Registered channel type 'Phone' (Standard Linux Telephony API Driver
chan_phone.so => (Linux Telephony API Support)
Asterisk Ready.
*CLI>      -- PAM Module Connected.

    -- Successfully connected to LDAP.

    -- Authentication Success.

    -- Registered SIP 'sravana' at 10.129.41.5 port 1000 expires 120
    -- Saved useragent "SJphone/1.60.299a/L (SJ Labs)" for peer sravana
```

Figure 5.5: Softphone connects to Asterisk PBX successfully

When a SIP user connects to Asterisk, the user has to provide a username and a password. Then Asterisk uses the pam module to get the user information from LDAP server. Asterisk verifies the user's credentials and authenticates the user. Now the SIP user can make and receive calls. The successful SIP user authentication is shown in figure 5.5.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

We have designed a feasible and affordable solution for Gateway PBX. Data MODEM and VIA motherboard are inexpensive devices. If we install AstLinux on VIA motherboard and modify the zaptel drivers to work with data MODEM, it works as a VoIP gateway and also provides PBX functionality. This is our proposed solution to Gateway PBX.

We have also configured an Asterisk server to authenticate with an LDAP server.

6.2 Future Work

In future, we can design special purpose hardware to run Asterisk. An effort has already been made in this direction by the Free Telephony Project [21]. They plan to build an embedded Asterisk box.

We have performed an experiment for measuring registration time in Asterisk. It is quite high for wireless medium. But, registration has to be done only once, before the user conversation. After that, the latency of SIP packets should be considered. To reduce that, we can do SIP message compression which can improve packet loss in the wireless medium.

Bibliography

[1] CEWiT. Center of excellence in wireless and information technology. http://www.cewit.org/.

[2] Consultation paper by TRAI. Growth of telecom services in rural india. Downloadable from http://www.trai.gov.in/consultation.htm, October 2004.

[3] Department of Telecommunications, India. Guidelines for Universal Service Support. Downloadable from http://www.dotindia.com/uso/usoindex.htm, 2005.

[4] Department of Telecommunications, India. Village Public Telephone—Database and Monitoring System. http://www.moc.gov.in/report.asp, 2005.

[5] Sipura, Inc. Telephone Gateway - SPA3000. http://www.sipura.com/products/spa3000.htm, 2003.

[6] Linksys, Inc. Telephone Gateway - SPA3102. http://www.voiptalk.org/products/Linksys+SPA+3102, 1992.

[7] VIA Technologies, Inc. VIA pc1500 mainboard.

[8] Digium, Inc. Asterisk: Complete IP PBX in software. http://www.asterisk.org/about, 1999.

[9] Digium, Inc. Asterisk Architecture Overview. http://www.asteriskvoipnews.com/asterisk_development/asterisk_architect%ure_overview.html, 1999.

[10] Miro International Pty Ltd. Astlinux: Custom Linux distro centered around asterisk. http://www.astlinux.org/.

[11] The Timbaktu Collective. Timbaktu: Voluntary organisation. http://www.timbaktu.org/.

[12] K. Sravana Kumar, Janak Chandarana, Srinath Perur, Raghuraman, Sridhar Iyer and Sameer Sahasrabuddhe. Voip-based Intra-Village Teleconnectivity: An Architecture and case-study. WISARD, COMSWARE, Bangalore, 2007.

[13] Andy Powell. Getting Started With Asterisk. `http://www.automated.it/guidetoasterisk.htm`, 1999.

[14] Yate — Yet another telephony engine. Call Generator. `http://yate.null.ro/pmwiki/index.php?n=Main.Callgen`, 2005.

[15] Digium, Inc. Digium X100P FXO PCI card.

[16] The forum for Asterisk and VoIP. Intel V.92 Modem use as FXO card. `http://www.asteriskguru.com/board/image-vp3269.html`, 2006.

[17] Fedoranews.org. How to setup and maintain openldap server for your network. `http://fedoranews.org/mediawiki/index.php/How_to_setup_and_maintain_Ope%nLDAP_server_for_your_network`, 2007.

[18] Padl. Pam: Pluggable Authentication Module. `http://www.padl.com/OSS/pam_ldap.html`, 2006.

[19] Philippe Sultan, INRIA. Sip peers external authentication. `http://www-rocq.inria.fr/who/Philippe.Sultan/Asterisk/asterisk_sip_exte%rnal_authentication.html`, 2005.

[20] PADL Software Pty Ltd. The pam_ldap module. `http://www.padl.com/OSS/pam_ldap.html`, 2006.

[21] Free Telephony Project. Embedded Asterisk box. `http://www.rowetel.com/ucasterisk/index.html`, 2006.