N° d'ordre : .....

N° de série : .....

Democratic And Republic of Algeria

Ministry Of Higher Education And Scientific

Final Year Project Report

Presented at

Echahid Hamma Lakhdar University of 'El Oued

Faculty of Technology

Department of Electrical Engineering

In Partial Fulfillment for the Requirement Of The Degree of

MASTER ACADEMIC

Telecommunications

Presented by

Dadi Rabah and Slimani Abdennacer

TITLE

Implementation of a VoIP PBX Using the Asterisk open source

Section may 2016. Jury composed of :

Mr. Hetteri messoud	M.A.A	President
Mr. Hima abdelkader	M.A.A	Supervisor
Mr. Chemsa ali	M.C.B	Examiner
Mr. Dogaa laid	invite	Examiner

Scholar year 2015/2016

ملخص

في هذا المشروع المتواضع سنقوم باستخدام برنامج أستريسك بمصدر مفتوح كمنصة هاتفية PBX وكان هدف المشروع تطوير نظام الهاتف عن طريق الأنترنت معتمدا على البرمجة وليس على العتاد { الهاردوير} وقد بدأ في عام 1999 من طرف مهندس البرمجة " مارك سبنسر" مؤسس شركة ديجيوم و المحفز الأساسي لهذا المشروع هو عدم وجود أي مشروع مفتوح المصدر في مجال الاتصالات وكل المنتوجات المتوفرة آنذاك كانت خاصة.

ومنذ انطلاق هذا المشروع كانت وراءه شركة ديجيوم وقد تطور هذا المشروع منذ انطلاقته فآخر اصدار لهذا البرنامج يتضمن عدة مزايا متطورة في ادارة المكالامات و التوافق مع بعض الأجهزة التي كانت في السابق حصرية

وقد قسمنا هذه المذ كرة الى قسمين:

في القسم الأول سنتناول فيه الجانب النظري من مفاهيم وأساسيات المتعلقة يهذا الموضوع

أما الجزء الثاني سنقوم بتقديم أستريسك وتصميم المنصة الهاتفية PBX

Abstrait

Avec ce mémoire, nous allons utiliser le projet open source Asterisk comme un PBX. Asterisk est un projet open source qui a commencé avec l'objectif principal de développer une plate-forme de téléphonie IP, entièrement basé sur le logiciel (donc ne dépend pas du matériel) et sous une licence open.

Ce projet a été lancé en 1999 par l'ingénieur Mark Spencer à Digium. La principale motivation de ce projet est que dans le secteur des télécommunications, il n'y a pas de solutions ouvertes, et la plupart des solutions disponibles sont basées sur des standards propriétaires, qui sont non compatibles. Derrière le projet Asterisk il y a une entreprise, Digium, qui le sponsor depuis que le projet a été née dans ses laboratoires.

Le projet Asterisk a beaucoup grandi depuis sa naissance, offrant dans ses dernières versions des fonctionnalités avancées pour la gestion des appels et la compatibilité avec certains matériels qui étaient auparavant des solutions exclusives. En raison de cela, Asterisk est en train de devenir une alternative sérieuse à toutes ces solutions, car il a atteint un niveau de maturité qui le rend très stable.

Cette thèse sera divisée principalement en deux blocs totalement complémentaires. Dans le premier bloc, nous allons faire un peu de théorie sur le PBX et les différents protocoles utilisés. Sur le deuxième bloc, nous allons introduire le Asterisk et le développement d'un PBX VoIP basé sur le projet Asterisk.

Abstract

With this master thesis we are going to use the Asterisk open source project as a PBX.

Asterisk is an open source software that was developed with the main objective of developing an IP telephony platform. The main motivation behind the creation of this software was that in the telecommunications sector there is no open solutions, and most of the available solutions are based on proprietary standards, which are close and not compatible.

The Asterisk software has grown up a lot since its birth, offering in its latest versions advanced functionalities for managing calls and compatibility with some hardware that previously was exclusive solutions. Due to that, Asterisk is becoming a serious alternative to all these solutions because it has reached a level of maturity that makes it very stable.

Table of contents

Table of contents

Table of Tables

Table of Figures

Abbreviation

ACD: Automatic Call Distributor is a feature used to route calls in a call center environment to the appropriate person based on factors such as availability, call usage, time, etc.

Agent: Member of a queue.

AGI: Asterisk Gateway Interface.

ATA: Analog Telephone Adapter, a device used to connect an analog phone to a digital line.

CDR: Call Detail Record. This is the log of a call.

Codec: A Codec is a piece of code that encodes or decodes audio using a given type of algorithm.

CRM: Customer Relationship Management.

DID: Direct Inward Dialing simply refers to the phone number dialled by a caller to reach our telephone system.

DISA: Direct Inward System Access.

Firewall: A device that exists at the border of two or more networks or network segments, and applies policies to the traffic that traverses those borders based on the security requirements of the network.

Follow-Me: This feature of TrixBox uses ring groups to allow a user to float between multiple extensions.

FXO: The Foreign eXchange Office is the end point of a connection. It is the FXO device that receives a call.

FXS: A Foreign eXchange Station is the sender of the call to an end-point device.

IAX: Inter-Asterisk eXchange protocol. The protocol is developed by Digium as a simpler and easier-to-manage alternative to using SIP for VoIP.

ISDN: Integrated Services Digital Network. This gained some popularity within small to medium-sized businesses as a cost-effective way of connecting to the PSTN and getting some advanced services, like many lines to one office or voice and data lines on one service. ISDN is a digital service and offers a few more features over POTS.

ITSP: An Internet Telephone Service Provider can deliver telephone network connectivity to our Asterisk PBX over Internet rather than over analog phone lines that need to be physically installed at our location.

IVR: Interactive Voice Response is known in the TrixBox system as the Digital Receptionist. This is the system that creates voice-prompt menus to help callers locate the appropriate person to speak to.

Hard phone: This is a hardware-based telephone.

Overhead Paging: Public Announcement System.

PBX: PBX (Private Branch eXchange) refers to the telephone switching system installed in a private location such as our office.

Abbreviation

POTS: Plain Old Telephone Service. This is commonly used for residential purposes. POTS is an analog system and is controlled by electrical loops. It is provided by copper wires run to residences and places of business and is therefore the cheapest and easiest telephone service to roll out.

Predictive Dialer: Predictive Dialer is a software that dials ahead of a user in order to determine if the dialled number is answered by a human rather than by a fax machine or is ringing out. It is used in call centers to increase productivity.

PSTN: Public Switched Telephone Network refers to the public phone network that carries all traditional phone calls.

Queues: A call queue is a function that places callers into a waiting room while they wait for the next available agent.

Ring Groups: A ring group is a collection of extensions that will all ring at the same time when a call is transferred to the group's extension number.

SIP: Session Initiation Protocol. This is a commonly used VoIP protocol.

SoftPhone: This is a software-based telephone.

Trunk: A trunk is a channel that operates between two distinct points. This can be either between PBXs within an organization, or between the organization's PBX and its provider.

T1/E1: This is common in larger companies, although in recent years it has become more affordable. T1/E1 is a digital service and offers yet more features than ISDN, the most important feature being increased bandwidth that translates, in telephony, to more telephone lines.

VoIP: The term VoIP simply means the ability to send voice communication over existing network wires using the same methods that are used for other internet services such as email, web surfing, or instant messaging.

Introduction

A **PBX** is an acronym for a Private Branch eXchange, which provides the internal telephone system. Telephone exchanges were initially under the control of the telephone providers, such as AT&T in the US or PTT in Algeria. These companies handled all line provisioning and call routing between the businesses and the public. Initially, the routing of calls was done by a team of operators sitting in the offices of the telephone companies and routing calls by plugging and unplugging cables to connect one caller to another. Eventually, as the reliance and the demands of this service grew, technology evolved to the point where we had automatic systems managing these calls.

As the modern telephone networks began to take shape, private companies saw a greater reliance on telephone communication. Many decided to implement their own services so that they could handle calls internal to the organization. Usually, the equipment was leased or bought from the telephone companies mentioned previously, so they were quite happy to help with these services. These companies also got to charge for the lines and calls connecting the company externally, and so they could profit from this too. So the expensive digital lines were now being used only as a means of communicating outside the building, rather than using externally provided lines for all communication.

At this point, it became obvious that there was a need for these companies to install their own telephone equipment to route internal calls and, in some cases, to make sure calls going out or coming into the company went via the correct routes. For example, you don't want Alice in accounting calling Bob in HR through a line that leaves the company and crosses continents if they sit within the same building. Therefore, there is a requirement for a PBX to effectively manage calls and ensure that they go via the most cost effective and reliable routes in order to keep the company communicating internally between departments and employees, and externally with customers and suppliers.

In its basic form, a PBX is the interface between the public telephone network and the private network within the company. Since most companies need fewer phones lines than the number of employees they have, they can get away by having a few outgoing lines but many internal extensions so that employees can converse internally. This costs little more than the maintenance of the PBX and internal cabling, and there are no line rentals or other call charges being paid to the telecommunications provider. The PBX then handles all of the routing in and out of the company using the lines effectively. The PBX also handles calls within the company so that a call from one internal phone to another does not have to go out onto the phone circuits and back in.

As PBXs became more common, businesses and their employees required more features and functionality such as voicemail, call parking, call transfers, music on-hold, IVR menus, least-cost routing, and an Automatic Call Distributor (ACD) in order to provide for calling groups. With the increase in demand for communications in all aspects of a business, the features required in a phone system become more complex and more expensive. If modern companies had to rely on the telecommunications provider for all these features, the cost of communication could become prohibitively high.

In this thesis we will design and implement a PBX system based on Asterisk software, the first chapter will present some theoretical background that will help the reader understand the technology used in this PBX like VoIP and the different protocols used for transporting the voice data. In the second chapter will deal with practical aspect of the PBX like installation and configuration of ASTERISK and its different features, and we will end this thesis by a general conclusion that we will try to make some recommendations for future students that wish to continuo this modest work.

Chapter One Voice over IP

1.1 Introduction

before we go through the technical aspect of the VoIP let us first check some definitions and introduce the PBX from a historical aspect and a technical one. so we will go in this chapter through the protocols used for call control like (SIP, IAX) and others for voice data transport like RTP.

1.2 The Traditional PBX System

It is not hard to spot a traditional PBX system. It is usually a large box full of mechanical switches and relays mounted on a wall in 'the phone room'. When a company's requirement changes, they generally contact their PBX provider who will charge varying rates to make hardware and configuration changes to fit the new requirements. With PBXs being very complicated and each differing from the others greatly, it can take a considerable level of training and experience to provide the support for a busy PBX system. This leads to most PBX customers relying on their PBX suppliers for, often expensive, support. So while by bringing the communications internally businesses could benefit from savings on line rentals, they still often had a reliance on their providers for support. Often, the companies selling and supporting the PBXs were the same telecommunication companies providing the external lines.

With a traditional PBX system we would also almost always purchase our phone system from the same manufacturer as the PBX system, usually with very few options to choose from when it comes to contract options and hardware such as telephone handsets or headsets. Adding features like voicemail can usually be an expensive add-on to the base system, sometimes requiring an entirely new piece of equipment! A traditional PBX system has the following structure:

Figure 0-1 Traditional PBX

Although some legacy PBX systems now have options for network access and VoIP functionality, these options are often very expensive upgrades and they generally lack the features and configuration options in the newer VoIP systems.

1.3 Hybrid PBX System

A hybrid PBX system combines the features of a traditional PBX system with VoIP functionality. In some cases, the VoIP functionality may just be the way the PBX communicates with the phones. Some other VoIP functionalities may include the ability to have remote extensions or **Soft Phones**, and the ability to use Internet Telephone Service Providers (ITSPs) and not just the traditional public telephone network. The main added benefit is the combined functionality, as we can keep all our existing lines and numbers and add in VoIP for substantial savings where possible.

The Asterisk PBX system is a full hybrid system combining numerous types of connections to the public telephone network as well as VoIP functionality including:

- Use of industry-standard SIP-compliant phones
- Remote extensions using either SIP-compliant phones, or Soft Phones
- Support for IAX (Inter-Asterisk eXchange)
- Bridging remote Asterisk systems together to act as a single system

Following is an example of a hybrid PBX system:

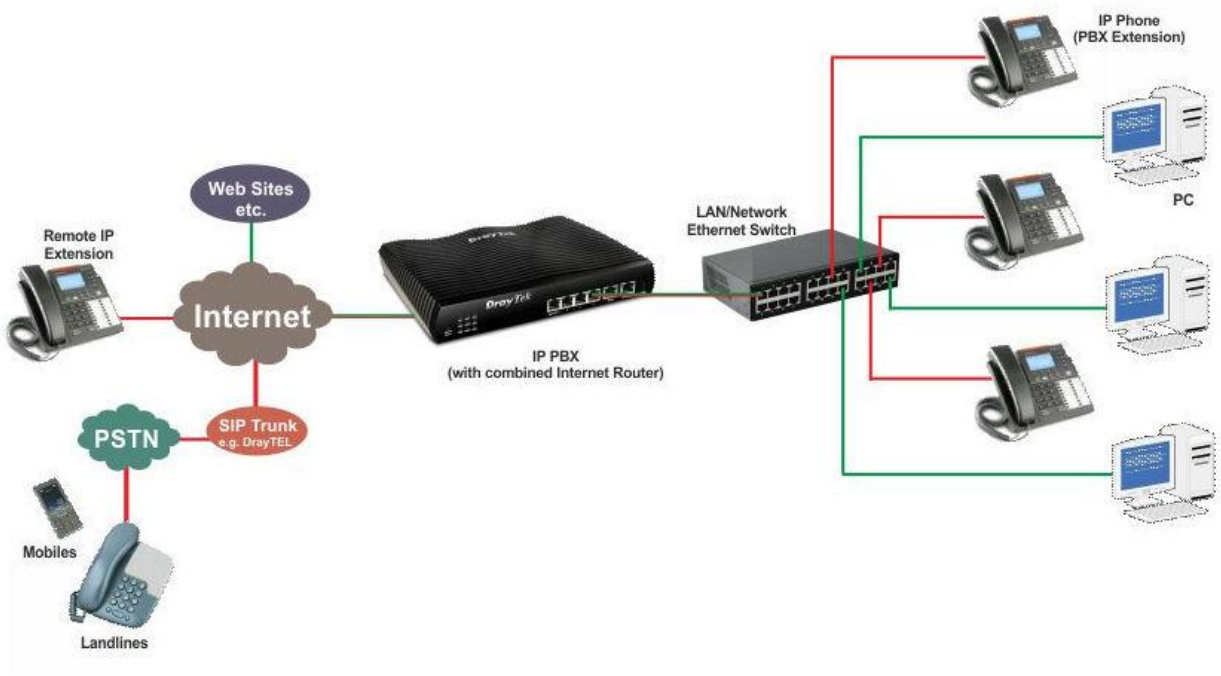

Figure 0-2 Hybrid PBX

1.4 Voice over Internet protocol (VoIP)

We have covered, in brief, how a traditional PBX system could lack some of the features of a Voice over Internet Protocol system. We can now take a look at VoIP in a little bit more detail to get an idea of what the benefits are.

Firstly, it's important to realize that VoIP doesn't entirely replace the PSTN (although it could). VoIP is yet another, cheaper, and easier way to connect to the PSTN. You can make and receive calls that are initiated and terminated entirely across VoIP and you can call a standard PSTN number from VoIP and vice-versa, as long as your ITSP (Internet Telephony Service Provider) supports it or if you link your VoIP system to the PSTN yourself. Both of these are options to consider with the PBX.

A VoIP system can use a variety of protocols and we will cover some of those protocols relevant to our situation as we come across them. VoIP is a catch-all term for these protocols and refers to transferring voice data over the Internet.

As the Internet grew and became a more flexible system than the PSTN, it became apparent that it was possible and, in many cases, preferable to use the Internet for carrying voice as

well as data. There were a few limitations that had to be overcome before this could be feasible. For example, data connections can tolerate some latency in communication but latency in voice can be very annoying as it leads to gaps in conversation and constant repetition. As Internet connection latency decreased and speeds increased, voice communication has become more viable.

There is a tendency to think of VoIP as a new technology. However, it is almost two decades old and has only recently become so popular because there are now a few good pieces of software that use this technology. There are also many companies investing in VoIP, since the data lines that provide Internet services are now at a level where they are usually reliable enough to be used for voice communication. Customers and employees expect these data lines to be low-latency, clear, and always available. While many Internet services still have problems, the situation is certainly much better than it was in the late 80s and early 90s when VoIP was first touted as the killer technology. It wasn't quite there then, but is certainly getting there now.

The most important facet of VoIP is that it is "over Internet Protocol". This means that it benefits from the layered design of Internet communication and can be a very flexible communication mechanism. A VoIP implementation can generally be shifted from one service provider to another with little or no effect on the systems in use. Anyone that has gone through the nightmare of moving just a single telephone number between providers will realize the benefit VoIP brings in this area. Flexibility in communication is an important aspect for businesses as it helps to control the business process. VoIP is also many times cheaper than traditional telephone services as it can be routed over a variety of cheap lines. The most important aspect here is usually the long distance rates. Calls can traverse the Internet until they get to the same country, state, or city as the recipient before touching the PSTN and in some cases bypass the PSTN entirely, meaning that we are no longer shackled to our telecommunications provider. We can pick and choose from the many Internet Providers and/or Internet Telephone Service Providers. The one downside to VoIP is that Internet connections are often less stable than the PSTN and therefore we can have occasional downtime in our telephony service. This can be mitigated by having multiple providers with failover, something which is near to impossible or prohibitively expensive with a PSTN service.

1.5 Signaling Transport

as we know that in order a call to be established we need some message interchange inorder that the call will be setup. in this chapter we will go through the different signaling protocols theory that are used in the PBX. and in the next chapter we will see the media transport and the different type of mechanisms that are used to establish a call.

1.5.1 H.323

A related Internet communications protocol is the ITU recommendation H.323, entitled "Packet-Based Multimedia Communication." H.323 is introduced as a related protocol to SIP for signaling VoIP and multimedia communication.

1.5.2 Introduction to H.323

H.323 is an umbrella recommendation that covers all aspects of multimedia communication over packet networks. It is part of the H.32x series of protocols that describes multimedia communication over ISDN, broadband (ATM), telephone (PSTN), and packet (IP) networks. Originally developed for video conferencing over a single LAN segment, the protocol has been extended to cover the general problem of telephony over the Internet. The first version was approved by the ITU in 1996 and was adopted by early IP telephony networks. Version 2 was adopted in 1998 to fix some of the problems and limitations in version 1. Version 3 was adopted in 1999 and includes modifications and extensions to enable communications over a larger network. Version 4 was adopted in 2000 with some major changes to the protocol. Versions 5 and 6 made very small changes to the protocol.

1.5.3 H.323 Entities

Let's start by explaining the names that H.323 uses for various entities that appear in the VoIP network.

- **A Terminal** is typically a software or hardware VoIP phone. Certain programs (for example, a voice mail software) could also introduce themselves as terminals in the protocol exchange.

- **A Gateway** is a device that allows a bidirectional communication with devices in another telecommunication network. The other network is usually PSTN but you can also have a H.323-to-SIP gateway or even a H.323-to-H.323 gateway. Formally, a gateway consists of two sub-components:

(1) Media Gateway Controller (MGC) handles call signaling and

(2) Media Gateway (MG) routes the audio (and possibly video) streams.

You will usually find the two components implemented within a single box but they can also be separate if you want the gateway to scale to a higher number of concurrent calls (in that situation, you typically have a single MGC and several MGs).

- **A Multipoint Conference Unit (MCU)** is a device that is used for multiparty conferencing. Again, it formally consists of two function blocks, a Multipoint Controller (MC) and Multipoint Processor (MP) where the latter is responsible for mixing the audio/video channels for the conference.

Terminals, gateways, and MCUs are collectively referred to as Endpoints. In addition to endpoints, the H.323 network can optionally have a fourth component, a Gatekeeper. Gatekeepers play the role of central controllers in the network. The most important tasks of a gatekeeper are registration of endpoints and call admission. The set of endpoints managed by the same gatekeeper is called a Zone.

1.5.4 Communication Between Entities

Let's now look how the individual entities that in the H.323 network use the various sub-protocols of H.323. First, for the endpoint-gatekeeper and gatekeeper-gatekeeper communication, a subset of the H.225.0 protocol is used. This subset of H.225.0 is known as RAS (Registration, Admission, Status). H.225.0-RAS contains messages for endpoint registration and unregistration at the gatekeeper, messages for call admission, call end, gatekeeper discovery, etc. The H.225.0-RAS messages are sent over the UDP protocol and the gatekeeper listens at port 1719/udp (unicast) and 1718/udp (multicast). The multicast address reserved for gatekeeper communication is 224.0.1.41. For call signaling between endpoints, H.323 uses the protocol Q.931. Q.931 has been borrowed from ISDN and it's messages contain the typical telephony data (like calling and called number).

However, Q.931 does not have certain data fields that are needed for Voice over IP communication (for example IP addresses and listening ports). To solve this, Q.931 messages embed H.225.0 messages that carry the complete information. The H.225.0 message is encoded using ASN.1 PER to a binary form and then inserted into the corresponding Q.931 message to a field that can carry a custom string of bytes (known as the User-User Information Element). Generally speaking, the embedding of messages of one protocol inside the messages of another protocol is used quite frequently throughout H.323.

Third, the H.245 protocol is used to negotiate audio (and video) parameters between endpoints. The negotiation covers codecs, IP addresses and ports, i.e. the parameters needed for RTP streams.

Last but not least, the Real Time Protocol (RTP) is used to carry the audio/video streams between the communicating endpoints.

1.6 IAX (The “Inter-Asterisk eXchange” Protocol)

IAX PROTOCOLIAX is an application layer protocol used for controlling and managing multimedia sessions over an IP network. It was created by open source community of branch exchange and its primary target was to efficiently control only voice calls over internet, but it is also capable of holding video streams with it. Mark Spencer Asterisk developed this protocol with the vision to decrease the complexity and to reduce the deficiencies of VOIP communication. IAX is an "all in one" protocol because of its ability to transmit media and control sessions together within same protocol. Unlike SIP or H.323 it does not require the support of RTP protocol. It uses single UDP stream to transmit and receive both signaling and media over static internal port number '4569'. Using single UDP static port IAX can bypass easily through firewalls and no other protocols are required to enable NAT with it. There is no requirement of extra configurations in the core network using IAX protocol to pass Nat Firewalls. It uses less overheads than RTP protocol and requires less bandwidth. IAX uses only 20% overhead with 4 bytes over a packet while RTP uses 60% of overheads with 12 bytes on each voice packet. IAX also has an ability of multiplexing and tunneling multiple channels over a single link. Data from multiple multimedia sessions are merged into single packet, thus reducing the IP overheads and reduce latency. By using G.729 compression codec multiple calls can be sent over single MB bandwidth. This protocol supports native encryption using different scenarios like AES-128 method. Unlike text commands in SIP, IAX uses binary data which

is easily understandable by most of machines. All the signaling information is transferred only over data link layer. DTMF(duel tone mode frequency) tones are also send along with signaling information. Like SIP, IAX also has a mechanism of call flow. In IAX enabled communication, call creation and termination messages are exchanged directly among users, there is no involvement of server between them. When a new user enters in a network, it gets registered with the server and the users are connected with each other in the form of Peer-to-Peer connectivity. Let us take the example of call flow between user A and user B. When user A wants to communicate with user B, it dials its number and a 'New' message is send to user B along with DTMF tone. User B receives this message and respond with 'Accept' message. User A reply with an 'ACK'. When the device of user B starts ringing it sends 'Ring' message to user A and gets the 'ACK' in response from user A. After receiving the call user B tells the user A by 'Answer' message and in reply gets an 'ACK'. During conversation multimedia frames are exchanged among users. To terminate the call one user sends 'Hang-up' message to other user and after receiving 'ACK' session the line gets cleared.

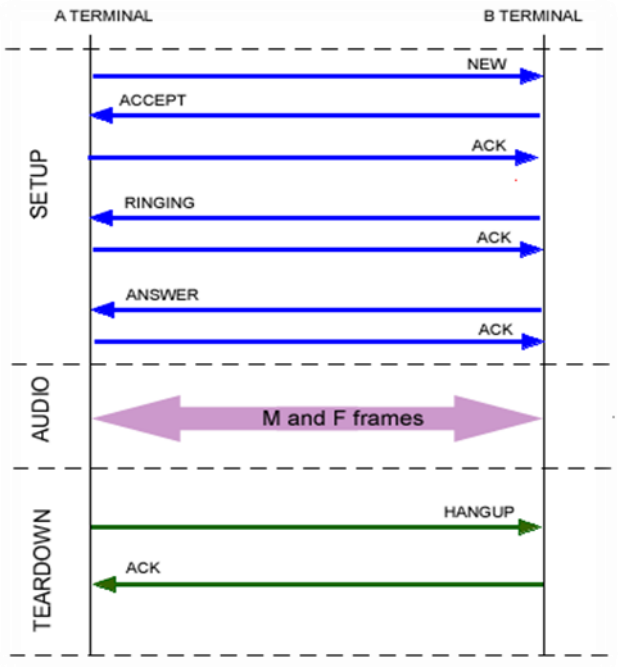

Figure 1-3 IAX complete call flow

To check the peers during calls and after the call, IAX has the mechanism to exchange keep alive messages among the peers. During the call session one user sends a 'Ping' message to other user in order to check its availability and user which receives the ping message respond with 'Pong' message to ensure that, I am alive. When there is no call session running, peers send 'Poke' messages to their neighbors and respond with 'Pong' message. These messages also maintains the quality of service, by comparing the values of 'Jitter' and 'Dropped frames' enclosed in initial Ping/Poke frames with the Poke frames which are send in response. A flow of call monitoring messages is shown in Fig. 1-3

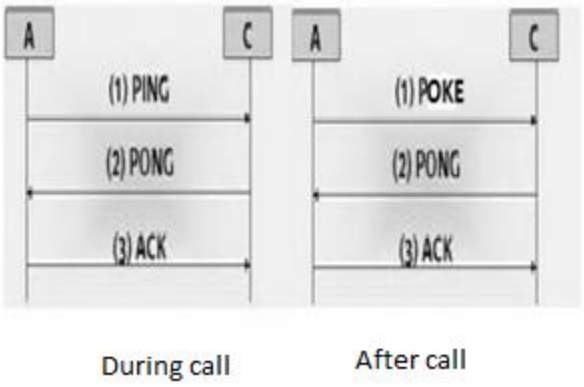

Figure 1-4 IAX Call monitoring

1.7 SIP – Overview

SIP is a signaling protocol used to create, modify, and terminate a multimedia session over the Internet Protocol. A session is nothing but a simple call between two endpoints. An endpoint can be a smart phone, a laptop, or any device that can receive and transmit multimedia content over the Internet. SIP is incorporated with two widely used internet protocols: *HTTP* and *SMTP*. From HTTP, SIP borrowed the client-server architecture and the use of URL and URI. From SMTP, it borrowed a text encoding scheme and a header style. also SIP takes the help of SDP (Session Description Protocol) which describes a session and RTP (Real Time Transport Protocol) used for delivering voice and video over IP network. we will introduce the different elements related to the SIP protocol.

1.7.1 SIP URI

SIP entities are identified using SIP URI (Uniform Resource Identifier). A SIP URI has form of sip:username@domain, for instance, sip:ali@company.com. As we can see, SIP URI consists of username part and domain name part delimited by @ (at) character. SIP URIs are similar to e-mail addresses, it is, for instance, possible to use the same URI for e-mail and SIP communication, such URIs are easy to remember.

1.7.2 SIP Network Elements

Although in the simplest configuration it is possible to use just two user agents that send SIP messages directly to each other, a typical SIP network will contain more than one type of SIP elements. Basic SIP elements are user agents, proxies, registrars, and redirect servers. We will briefly describe them in this section.

Note that the elements, as presented in this section, are often only logical entities. It is often profitable to co-locate them together, for instance, to increase the speed of processing, but that depends on a particular implementation and configuration.

1. Client (user Agent)

Internet end points that use SIP to find each other and to negotiate a session characteristics are called user agents. User agents usually, but not necessarily, reside on a user's computer in form of an application--this is currently the most widely used approach, but user agents can be also cellular phones, PSTN gateways, PDAs, automated IVR systems and so on.

User agents are often referred to as User Agent Server (UAS) and User Agent Client (UAC). UAS and UAC are logical entities only, each user agent contains a UAC and UAS. UAC is the part of the user agent that sends requests and receives responses. UAS is the part of the user agent that receives requests and sends responses.

Because a user agent contains both UAC and UAS, we often say that a user agent behaves like a UAC or UAS. For instance, caller's user agent behaves like UAC when it sends an INVITE requests and receives responses to the request. Callee's user agent behaves like a UAS when it receives the INVITE and sends responses.

2. Servers

Servers are in general part of the network. They possess a predefined set of rules to handle the requests sent by clients. Servers can be of several types :

- Proxy Server: These are the most common type of server in a SIP environment. When a request is generated, the exact address of the recipient is not known in advance. So the

client sends the request to a proxy server. The server on behalf of the client (as if giving a proxy for it) forwards the request to another proxy server or the recipient itself.

- Redirect Server: A redirect server redirects the request back to the client indicating that the client needs to try a different route to get to the recipient. It generally happens when a recipient has moved from its original position either temporarily or permanently.
- Registrar Server: The registrar is a special SIP entity that receives registrations from users, extracts information about their current location (IP address, port and username in this case) and stores the information into location database.
- Location Server: The addresses registered to a Registrar are stored in a Location Server.

1.7.3 SIP - Messaging

Communication using SIP (signaling) implicate the interchange of series of *messages*. Messages are transported independently by the network. Usually they are transported in a separate UDP datagram each. Each message consist of "first line", message header, and message body. The first line identifies the type of the message. There are two type of the messages: *requests(*often called method) and *responses*. Requests are usually used to initiate some action or inform the recipient of requesting something. Replies are used to confirm that a request was received and processed and contain the status of the processing.

1. **-SIP Requests (methods):**

SIP requests or methods are considered “verbs” in the protocol, since they request a specific action to be taken by another user agent or server. in SIP protocol there exist six original methods: The INVITE, REGISTER, BYE, ACK, CANCEL, and OPTIONS. And the NOTIFY, PUBLISH, REFER, MESSAGE, INFO, PRACK, UPDATE, SUBSCRIBE methods are described in separate RFCs.

- INVITE :Invites a user to a call
- ACK : Acknowledgement is used to facilitate reliable message exchange for INVITEs.
- BYE :Terminates a connection between users
- CANCEL :Terminates a request, or search, for a user. It is used if a client sends an INVITE and then changes its decision to call the recipient.

- OPTIONS :Solicits information about a server's capabilities.
- REGISTER :Registers a user's current location
- NOTIFY is used by a user agent to convey information about the occurrence of a particular event.
- PUBLISH: is used by a user agent to send (or publish) event state information to a server known as an event state compositor
- REFER : is used by a user agent to request another user agent to access a URI or URL resource.
- MESSAGE : is used to transport instant messages (IM) using SIP.IMs usually consists of short messages exchanged in near-real time by participants engaged in a text conversation
- INFO: is used by a UA to send call signaling information to another UA with which it has an established media session. The request is end-to-end and is never initiated by proxies
- PRACK: is used to acknowledge receipt of reliably transported provisional responses.
- UPDATE: is used to modify the state of a session without changing the state of the dialog.
- SUBSCRIBE: is used by a UA to establish a subscription for the purpose of receiving notifications (via the NOTIFY method) about a particular event. A successful subscription establishes a dialog between the UAC and the UAS.

2. SIP Responses:

A SIP response is a message generated by a UAS or a SIP server to reply a request generated by a UAC. There are six classes of SIP responses. The first five classes were borrowed from HTTP; the sixth was created for SIP. as shown in the next table

Class	Description	Action
1xx	Informational	This indicates the status of the call prior to completion—also known as a provisional response.
2xx	Success	The request has succeeded. If it was for an INVITE, ACK should be sent; otherwise, stop the retransmissions of the request.
3xx	Redirection	The server has returned possible locations. The client should retry the request at another server.
4xx	Client error	The request has failed due to an error by the client. The client may retry the request if it is reformulated according to the response.
5xx	Server failure	The request has failed due to an error by the server. The request may be retried at another server.
6xx	Global failure	The request has failed. The request should not be tried again at this or other servers.

Table 1-0-1 ***Table response messages***

1.8 Session Description Protocol (SDP)

1.8.1 Introduction

One of the most important uses of SIP is to negotiate the setup of sessions, as the name suggests. To do this, SIP uses another protocol, Session Description Protocol, to describe the actual parameters of the media session. This includes information such as media type, codec, bit rate, and the IP address and port

numbers for the media session. In short, negotiating media sessions is all about exchanging the data necessary to begin the RTP media sessions or SRTP media sessions that will be described later. So let us have some description of this protocol.

1.8.2 Session Description Protocol :

SDP stands for Session Description Protocol. It is used to describe multimedia sessions in a format understood by the participants over a network. Depending on this description, a party decides whether to join a conference or when or how to join a conference.

- The owner of a conference advertises it over the network by sending multicast messages which contain description of the session e.g. the name of the owner, the name of the session, the coding, the timing etc. Depending on these information the recipients of the advertisement take a decision about participation in the session.
- SDP is generally contained in the body part of Session Initiation Protocol (SIP).
- SDP is defined in RFC 2327. An SDP message is composed of a series of lines, called fields, whose names are abbreviated by a single lower-case letter, and are in a required order to simplify parsing

1.8.3 Purpose of SDP

The purpose of SDP is to convey information about media streams in multimedia sessions to help participants join or gather info of a particular session.

- SDP is a short structured textual description.
- It conveys the name and purpose of the session, the media, protocols, codec formats, timing and transport information.
- A tentative participant checks these information and decides whether to join a session and how and when to join a session if it decides to do so.
- The format has entries in the form of <type> = <value>, where the <type> defines a unique session parameter and the <value> provides a specific value for that parameter.
- The general form of a SDP message is: x = parameter1 parameter2 ... parameterN
- The line begins with a single lower-case letter, for example, x. There are never any spaces between the letter and the =, and there is exactly one space between each parameter. Each field has a defined number of parameters.

1.8.4 Session Description Parameters

Session description (* denotes optional)

- v = (protocol version)
- o = (owner/creator and session identifier)
- s = (session name)
- i =* (session information)

- u =* (URI of description)
- e =* (email address)
- p =* (phone number)
- c =* (connection information - not required if included in all media)
- b =* (bandwidth information)
- z =* (time zone adjustments)
- k =* (encryption key)
- a =* (zero or more session attribute lines)

Protocol Version

The v = field contains the SDP version number. Because the current version of SDP is 0, a valid SDP message will always begin with v = 0

1. Origin

The o = field contains information about the originator of the session and session identifiers. This field is used to uniquely identify the session.

- The field contains: o=<username><session-id><version><network-type><address-type>
- The **username parameter** contains the originator's login or host.
- The **session-id** parameter is a Network Time Protocol (NTP) timestamp or a random number used to ensure uniqueness.
- The **version** is a numeric field that is increased for each change to the session, also recommended to be a NTP timestamp.
- The **network-type** is always IN for Internet. The address-type parameter is either IP4 or IP6 for IPv4 or IPv6 address either in dotted decimal form or a fully qualified host name.

2. Session Name and Information

The s= field contains a name for the session. It can contain any nonzero number of characters. The optional i= field contains information about the session. It can contain any number of characters.

3. URI

The optional u= field contains a uniform resource indicator (URI) with more information about the session.

4. E-Mail Address and Phone Number

The optional e= field contains an e-mail address of the host of the session. The optional p= field contains a phone number.

5. Connection Data

The c= field contains information about the media connection.

- The field contains: c = <network-type><address-type><connection-address>
- The **network-type** parameter is defined as IN for the Internet.
- The **address-type** is defined as IP4 for IPv4 addresses and IP6 for IPv6 addresses.
- The **connection-address** is the IP address or host that will be sending the media packets, which could be either multicast or unicast.
- If multicast, the connection-address field contains: connection-address = base-multicast-address/ttl/number-of-addresses

where ttl is the time-to-live value, and number-of-addresses indicates how many contiguous multicast addresses are included starting with the base-multicast address.

6. Bandwidth

The optional b= field contains information about the bandwidth required. It is of the form: b = modifier:bandwidth − value

7. Time, Repeat Times, and Time Zones

The t= field contains the start time and stop time of the session. t = start − time stop − time

The optional r= field contains information about the repeat times that can be specified in either in NTP or in days (d), hours (h), or minutes (m).

The optional z= field contains information about the time zone offsets. This field is used if are occurring session spans a change from daylight savings to standard time, or vice versa.

8. Media Announcements

The optional m= field contains information about the type of media session. The field contains: m = media port transport format − list

- The media parameter is either audio, video, text, application, message, image, or control. The port parameter contains the port number.
- The transport parameter contains the transport protocol or the RTP profile used.
- The format-list contains more information about the media. Usually, it contains media payload types defined in RTP audio video profiles.

9. Attributes

The optional a= field contains attributes of the preceding media session. This field can be **used to extend SDP to provide more information about the media**. If not fully understood by a SDP user, the attribute field can be ignored. There can be one or more attribute fields for each media payload type listed in the media field.

Attributes in SDP can be either

- session level, or
- media level.

Session level means that the attribute is listed before the first media line in the SDP. If this is the case, the attribute applies to all the media lines below it.

Media level means it is listed after a media line. In this case, the attribute only applies to this particular media stream.

SDP can include both session level and media level attributes. If the same attribute appears as both, the media level attribute overrides the session level attribute for that particular media stream. Note that the connection data field can also be either session level or media level.

10. SDP Extensions

There are a number of SDP extensions that have been defined. Common ones are summarized in the following table.

Attribute	Name
a=rtcp	Port and IP address for RTCP [6]
a=mid	Media session identifier and grouping of media streams
a=group	
a=setup	Connection-oriented media using as TCP transport
a=connection	
a=key-mgt	Key management for MIKEY
a=crypto	Key management for SRTP
a=floorctrl	Binary Floor Control Protocol (BFCP) information
a=confid	
a=userid	
a=floorid	
a=fingerprint	Connection-oriented media using TLS [
a=label	Media label
a=accept-types	Message Session Relay Protocol (MSRP) information
a=accept-wrapped-types	
a=max-size	
a=path	
a=ice-pwd	Interactive connectivity establishment (ICE)
a=ice-ufrag	
a=ice-lite	
a=ice-mismatch	
a=ice-options	
a=chatroom	Chat room name for MSRP

Table 1-0-2 Common SDP Extensions

- The a = setup and a = connection attributes are used for connection oriented media, such as TCP.
- The first m= media line is for a BFCP stream running over TLS over TCP.

- The a=connection:new indicates that a new TCP connection needs to be opened and that this endpoint will do a passive open (the other endpoint will do the active open).
- The a = fingerprint contains a fingerprint of the certificate to be exchanged during the TLS handshake.
- The a = confide and a = userid attributes contain the conference ID and user ID of the user.
- The a = floorid attributes indicate that floor 1 is associated with a = label:1, which is associated with the m = audio stream while floor 2 is associated with a = label:2, which is associated with the m = video stream.

1.9 A Simple Session Establishment Example

After we have seen the Signaling protocol. now we will show the basic message exchange between two SIP devices to establish and tear down a session. This example will be introduced using call flow diagrams between a called and calling party, along with the details of each message. Each arrow in the figures represents a SIP message, with the arrowhead indicating the direction of transmission. The red line in the figures indicates the media stream. In these examples, the media will be assumed to be RTP packets containing audio, but it could be another protocol. Details Media are covered in the next Chapter.

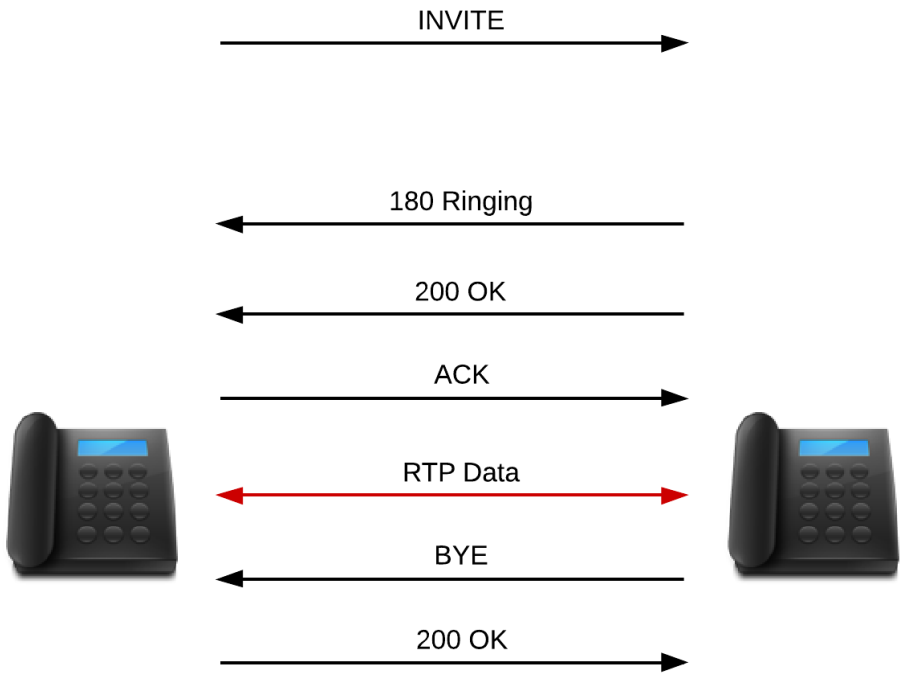

Figure 1-5 the SIP message exchange between two SIP-enabled devices.

The two devices could be SIP phones, phone clients running on a laptop or PC (known as *softclients*), PDAs, or mobile phones. It is assumed that both devices are connected to an IP network such as the Internet and know each other's IP address.

The calling party, Tesla, begins the message exchange by sending a SIP I NVITE message to the called party, Marconi. The INVITE contains the details of the type of session or call that is requested. It could be a simple voice (audio) session, a multimedia session such as a videoconference, or a gaming session.

The INVITE message is shown in FIG 6.

```
INVITE sip:Marconi@radio.org SIP/2.0
Via: SIP/2.0/UDP lab.high-voltage.org:5060;branch=z9hG4bKfw19b
Max-Forwards: 70
To: G. Marconi <sip:Marconi@radio.org>
From: Nikola Tesla <sip:n.tesla@high-voltage.org>;tag=76341
Call-ID: j2qu348ek2328ws
CSeq: 1 INVITE
Subject: About That Power Outage...
Contact: <sip:n.tesla@lab.high-voltage.org>
Content-Type: application/sdp
Content-Length: 158
v=0
o=Tesla 2890844526 2890844526 IN IP4 lab.high-voltage.org
s=Phone Call
c=IN IP4 100.101.102.103
t=0 0
m=audio 49170 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

Figure 1-6 INVITE SIP message

Since SIP is a text-encoded protocol, this is actually what the SIP message would look like "on the wire" as a UDP datagram being transported over, for example, Ethernet. The fields listed in the INVITE message are called header fields.

They have the form Header: Value CRLF. The first line of the request message, called the start line, lists the method, which is INVITE, the Request-URI, then the SIP version number (2.0), all separated by spaces. Each line of a SIP message is terminated by a CRLF (Carriage Return Line Feed). The Request-URI is a special form of SIP URI and indicates the resource to which the request is being sent, also known as the request target.

The first header field following the start line shown is a Via header field. Each SIP device that originates or forwards a SIP message stamps its own address in a Via header field, usually written as a host name that can be resolved into an IP address using a DNS query. The Via header field contains the SIP version number (2.0), a "/", then UDP for UDP transport, a space, the hostname or address, a colon, then a port number (in this example the "well-known" SIP port number 5060). The branch parameter is a transaction identifier. Responses relating to this request can be correlated because they will contain this same transaction identifier.

the Max-Forwards field is initialized to some large integer and decremented by each SIP server, which receives and forwards the request, providing simple loop detection.

The next header fields are the To and From header fields, which show the originator and destination of the SIP request. SIP requests are routed based on the Request-URI instead of the To URI. This is because the Request-URI can be changed and rewritten as a request is forwarded, while the To URI generally stays the same. When a name label is used, as in this example, the SIP URI is enclosed in brackets <>. The name label could be displayed during alerting, for example, but is not used by the protocol.

The Call-ID header field is an identifier used to keep track of a particular SIP session. The originator of the request creates a locally unique string. Some older implementations also add an "@" and its host name to the string. In addition to the Call-ID, each party in the session also contributes a random identifier, unique for each call. The initiator of the session that generates the establishing INVITE generates the unique Call-ID and From tag. In the response to the INVITE, the user agent answering the request will generate the To tag. The combination of the local tag (contained in the From header field), remote tag (contained in the To header field), and the Call-ID uniquely identifies the established session, known as a *dialog*. This dialog identifier is used by both parties to identify this call because there could be multiple calls set up between them. Subsequent requests within the established session will use this dialog identifier.

the CSeq header field, or command sequence. It contains a number, followed by the method name, INVITE in this case. This number is incremented for each new request sent. In this example, the command sequence number is initialized to 1, but it could start at another integer value.

The Via header fields plus the Max-Forwards, To, From, Call-ID, and CSeq header fields represent the minimum required header field set in any SIP request message. Other header fields can be included as optional additional information, or information needed for a specific request type.

A Contact header field is also required in this INVITE message, which contains the SIP URI of Tesla communication device, known as a user agent (UA); this URI can be used to route messages directly to Tesla.

The optional Subject header field is present in this example. It is not used by the protocol, but could be displayed during alerting to aid the called party in deciding whether to accept the call. The same sort of useful prioritization and screening commonly performed using the

Subject and From header fields in an e-mail message is also possible with a SIP INVITE request.

The Content-Type and Content-Length header fields indicate that the message body is Session Description Protocol or SDP and contains 158 octets of data. A blank line separates the message body

from the header field list, which ends with the Content-Length header field. In this case, there are seven lines of SDP data describing the media attributes that the caller Tesla desires for the call. This media information is needed because SIP makes no assumptions about the type of media session to be established the caller must specify exactly what type of session (audio, video, gaming) that he wishes to establish. The SDP field names are listed in Table 3, but a quick review of the lines shows the basic information necessary to establish a session.

SDP Parameter	Parameter Name
v=0	Version number
o=Tesla 2890844526 2890844526 IN IP4 lab.high-voltage.org	Origin
s=Phone Call	Call subject
c=IN IP4 100.101.102.103	Connection
t=0 0	Time
m=audio 49170 RTP/AVP 0	Media
a=rtpmap:0 PCMU/8000	Attributes

Table 1-0-3 SDP field names

Table 3 includes the:

• Connection IP address (100.101.102.103);

• Media format (audio);

• Port number (49170);

• Media transport protocol (RTP);

• Media encoding (PCM μ Law);

• Sampling rate (8,000 Hz).

The next message in Figure X.X is a 180 Ringing message sent in response to the INVITE. This message indicates that the called party, Marconi, has received the INVITE and that alerting is taking place. The alerting could be ringing a phone, a flashing message on a screen, or any other method of attracting the attention of the called party, Marconi.

The 180 Ringing is an example of a SIP response message. Responses are numerical and are classified by the first digit of the number. A 180 response is an *informational class* response, identified by the first digit being a 1. Informational responses are used to convey noncritical information about the progress of the

call. The response code number in SIP alone determines the way the response is interpreted by the server or the user. The reason phrase, Ringing in this case, is suggested in the standard, but any text can be used to convey more information.

The 180 Ringing response has the following structure:

```
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP lab.high-voltage.org:5060;branch=z9hG4bKfw19b
;received=100.101.102.103
To: G. Marconi <sip:marconi@radio.org>;tag=a53e42
From: Nikola Tesla <sip:n.tesla@high-voltage.org>;tag=76341
Call-ID: j2qu348ek2328ws
CSeq: 1 INVITE
Contact: <sip:marconi@tower.radio.org>
Content-length: 0
```

Figure 1-7 180RINGING SIP message

The message was created by copying many of the header fields from the INVITE message, including the Via, To, From, Call-ID, and CSeq, then adding a response start line containing the

SIP version number, the response code, and the reason phrase. This approach simplifies the message processing for responses.

The Via header field contains the original branch parameter but also has an additional received parameter. This parameter contains the literal IP address that the request was received from (100.101.102.103), which typically is the same address that the URI in the Via resolves using DNS (lab.high-voltage.org). Note that the To and From header fields are not reversed in the response message as one might expect them to be. Even though this message is sent to Marconi from Tesla, the header fields read the opposite. This is because the To and From header fields in SIP are defined to indicate the direction of the request, not the direction of the message. Since Tesla initiated this request, all responses to this INVITE will read To: Marconi From: Tesla. The To header field now contains a tag that was generated by Marconi. All future requests and responses in this session or dialog will contain both the tag generated by Tesla and the tag generated by Marconi. The response also contains a Contact header field, which contains an address at which Marconi can be contacted directly once the session is established. When the called party, Marconi, decides to accept the call (i.e., the phone is answered), a 200 OK response is sent. This response also indicates that the type of media session proposed by the caller is acceptable. The 200 OK is an example of a *success class* response. The 200 OK message body contains Marconi's media information:

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP lab.high-voltage.org:5060;branch=z9hG4bKfw19b
;received=100.101.102.103
To: G. Marconi <sip:marconi@radio.org>;tag=a53e42
From: Nikola Tesla <sip:n.tesla@high-voltage.org>;tag=76341
Call-ID: j2qu348ek2328ws
CSeq: 1 INVITE
Contact: <sip:marconi@tower.radio.org>
Content-Type: application/sdp
Content-Length: 155
v=0
o=Marconi 2890844528 2890844528 IN IP4 tower.radio.org
s=Phone Call
c=IN IP4 200.201.202.203
t=0 0
m=audio 60000 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

Figure 1-8 200 OK SIP message

This response is constructed the same way as the 180 Ringing response and contains the same To tag and Contact URI. The media capabilities, however, must be communicated in a SDP message body added to the response. the SDP contains:

• End-point IP address (200.201.202.203);

• Media format (audio);

• Port number (60000);

• Media transport protocol (RTP);

• Media encoding (PCM μ-Law);

• Sampling rate (8,000 Hz).

The final step is to confirm the media session with an *acknowledgment* request. The confirmation means that Tesla has successfully received Marconi's response. This exchange of media information allows the media session to be established using another protocol: RTP in this example.

```
ACK sip:marconi@tower.radio.org SIP/2.0
Via: SIP/2.0/UDP lab.high-voltage.rg:5060;branch=z9hG4bK321g
Max-Forwards: 70
To: G. Marconi <sip:marconi@radio.org>;tag=a53e42
From: Nikola Tesla <sip:n.tesla@high-voltage.org>;tag=76341
Call-ID: j2qu348ek2328ws
CSeq: 1 ACK
Content-Length: 0
```

Figure 1-9 ACK SIP message

The command sequence, CSeq, has the same number as the INVITE, but the method is set to ACK. At this point, the media session begins using the media information carried in the SIP messages. The media session takes place using another protocol, typically RTP. The branch parameter in the Via header field contains a newer transaction identifier than the INVITE, since an ACK sent to acknowledge a 200 OK is considered a separate transaction. This message exchange shows that SIP is an end-to-end signaling protocol. A SIP network or SIP server is not required for the protocol to be used. Two end

points running a SIP protocol stack and knowing each other's IP addresses can use SIP to set up a media session between them. Although less obvious, this example also shows the client-server nature of the SIP protocol. When Tesla originates the INVITE request, he is acting as a SIP client. When Marconi responds

to the request, he is acting as a SIP server. After the media session is established, Marconi originates the BYE request and acts as the SIP client, while Tesla acts as the SIP server when he responds. This is why a SIP-enabled device must contain both SIP user agent server and SIP user agent client software during a typical session, both are needed. This is quite different from other client-server Internet protocols such as HTTP or FTP. The Web browser is always an HTTP client, and the Web server is always an HTTP server, and similarly for FTP. In SIP, an end point will switch back and forth during a session between being a client and a server.

In Figure 2.1, a BYE request is sent by Marconi to terminate the media session:

The Via header field in this example is populated with Marconi's host address and contains a new transaction identifier since the BYE is considered a separate transaction from the INVITE or ACK transactions shown previously. The To and From header fields reflect that this request is originated by Marconi, as they

are reversed from the messages in the previous transaction. Tesla, however, is able to identify the dialog using the presence of the same local and remote tags and Call-ID as the INVITE, and tear down the correct media session. Notice that all of the branch IDs shown in the example so far begin with the string z9hG4bK. This is a special string that indicates that the branch ID has been calculated using strict rules defined in RFC 3261 and is as a result usable as a transaction identifier.

The confirmation response to the BYE is a 200 OK:

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP tower.radio.org:5060;branch=z9hG4bK392kf
;received=200.201.202.203
To: Nikola Tesla <sip:n.tesla@high-voltage.org>;tag=76341
From: G. Marconi <sip:marconi@radio.org>;tag=a53e42
Call-ID: j2qu348ek2328ws
CSeq: 1392 BYE
Content-Length: 0
```

Figure 1-10 200 OK SIP message

The response echoes the CSeq of the original request: 1392 BYE. No ACK is sent since ACK is only sent in response to INVITE requests.

1.10 Media Transport

Establishing media sessions is one of the most important applications of SIP in Internet communications. An understanding of the issues relating to media transport of voice, video, DTMF, and text helps motivate the media negotiation capabilities of SIP. In this chapter, the Real-Time Transport Protocol (RTP) will

be introduced as the protocol that transports actual media samples. The basic steps in audio media encoding and decoding are discussed. The RTP header format is covered along with common RTP topologies. The RTP Control Protocol (RTCP) is introduced as a way to monitor call quality. RTP profiles and common audio codecs are discussed.

1.10.1 Real-Time Transport Protocol (RTP)

Real-Time Transport Protocol was developed to enable the transport of real time datagrams containing voice, video, or other information over IP. RTP was not the first VoIP protocol used on the Internet. Network Voice Protocol (NVP) was implemented in 1973 to carry real-time voice communications over the Internet. Early versions of RTP, first implemented in 1992, were used to transport voice over the Internet's multicast backbone (MBONE). Both H.323 and SIP use RTP for media transport, making it the most common standard for Internet communications.

RTP is defined by the IETF proposed standard RFC 3550 (which updates the original RFC 1889). RTP does not provide any quality of service over the IP network, RTP packets are handled the same as all other packets in an IP network. However, RTP allows for the detection of some of the impairments introduced by an IP network, such as:

• Packet loss;
• Variable transport delay;
• Out of sequence packet arrival;
• Asymmetric routing.

Here is how RTP fits into the common media processing steps.

1.10.2 Coding.

The coding step involves analog to digital conversion (A/D), which is implemented by low pass filtering, followed by sampling. The determination of how many bits per sample is specific to a particular codec (coder/decoder) algorithm. The particular codec used is transported by RTP in the payload type field. The sampling rate is carried in the offer/answer exchange in SDP, which negotiates the media session.

1. Packetization.

The packetization step involves breaking the codec sample data into individual datagrams for transport. The determination of packet size is based on a tradeoff between packetization delay (how many sampling intervals must pass before enough data is ready for the datagram) and transport efficiency (each datagram has the fixed overhead of the RTP header and lower layer headers). Typically, packet sizes are chosen to be small so that packetization time is around 20 ms to 30 ms. Packetization involves adding the RTP header to the codec payload.

2. Transport

RTP, as the name suggests, has a real-time nature, which requires a minimum latency (delay) across the Internet. There is no time to detect a missing packet, signal loss, and wait for a retransmission. This might be possible for nonreal-time streaming media, but not real-time media. As a result, RTP does not usually use TCP transport but instead uses UDP transport. So, datagrams may be lost

or may arrive out of sequence. Various fields in the RTP header field allow the detection of this.

3. Depacketization.

The depacketization step involves removing the RTP header from the codec payload.

4. Buffering.

The buffering step involves storing or buffering the codec samples before beginning playback. The choice of the buffer size for this step is critical for media quality. Too short a buffer will result in the buffer emptying and gaps in the media playback, while too long a buffer will introduce unpleasant latency. Adapting the size of the play back buffer when jitter or delay variation is occurring is best for media quality.

5. Decoding.

The decoding step involves sending the encoded packets to the decoder algorithm. The right codec is chosen based on the received payload type in the RTP header.

6. Playback

The playback step involves rendering the media to the user as audio, video, or perhaps text (as we shall see in real-time text or text over IP, ToIP).

In terms of media quality, the two most important factors are the packet loss rate and the end-to-end latency. Lost packets mean gaps in the playback stream that the codec algorithm must try to compensate for. Different codecs use different techniques for packet loss concealment (PLC). For example, interpolation can be used to try to predict the missing samples based on received samples either side of the lost ones. A simple replay algorithm can be useful for some media types. Silence or comfort noise insertion can be used to prevent users noticing the dead air of lost samples. Some codecs employ forward error correction (FEC), which allows partial reconstruction of missing packets under low loss conditions. Note that packets aren't really "lost" on the Internet, instead they are discarded by routers in the Internet due to congestion, or discarded by the RTP stack due to out of order arrival or late arrival resulting in missing their playback interval.

The end-to-end latency of real-time communications in general must be kept less than 150 ms. Longer latency than this affects the perceived quality of the call, resulting in users interrupting each other and starting and stopping when both parties speak at the same time.

There are many sources of delay in the media path. The codec itself introduces delay as it gathers at least one sample or frame before beginning coding and decoding. The packetization step introduces a delay of around 20 ms, the time it takes to gather a full packet's worth of data before sending it over the Internet. Transport delays are added by the routers and switches that forward and process the IP packets across the Internet. And finally, the buffering delay of the receiver to deal with jitter or delay variation also introduces latency.

Real-Time Transport Protocol (RTP) is an application layer protocol that uses UDP for transport over IP. RTP is not text encoded, but uses a bit-oriented header similar to UDP and IP. RTP version 0 is only used by the vat audio tool for MBONE broadcasts. Version 1 was a pre-RFC implementation and is not in use. The current RTP version 2 packet header has 12 octets. RTP was

designed to be very general; most of the headers are only loosely defined in the standard; the details are left to profile documents.

The header contains:

• Version (V): This 2-bit field is set to 2, the current version of RTP.

• Padding (P): If this bit is set, there are padding octets added to the end of the packet to make the packet a fixed length. This is most commonly used if the media stream is encrypted.

• Extension (X): If this bit is set, there is one additional extension following the header (giving a total header length of 14 octets). Extensions are defined by certain payload types.

• CSRC count (CC): This 4-bit field contains the number of content source identifiers (CSRC) that are present following the header. This field is only used by mixers that take multiple RTP streams and output a single RTP stream.

• Marker (M): This single bit is used to indicate the end of a complete frame in video, or the start of a talk-spurt in silence-suppressed speech.

• Payload type (PT): This 7-bit field defines the codec in use. The value of this field matches the profile number listed in the SDP.

• Sequence Number: This 16-bit field is incremented for each RTP packet sent and is used to detect missing/out of sequence packets.

• Timestamp: This 32-bit field indicates in relative terms the time when the payload was sampled. This field allows the receiver to remove jitter and to play back the packets at the right interval assuming sufficient buffering.

• Synchronization source identifier (SSRCI): This 32-bit field identifies the sender of the RTP packet. At the start of a session, each participant chooses an SSRC number randomly. if the two participants choose the same number, they have to change the value of the SSRCI until each party is unique.

• Contributing source identifier (CSRC): There can be none or up to 15 instances of this 32-bit field in the header. The number is set by the CSRC count (CC) header field. This field is only present if the RTP packet is being sent by a mixer, which has received RTP packets from a number of sources and sends out combined packets. A non multicast conference bridge would utilize this header.

RTP allows detection of a lost packet by a gap in the Sequence Number. Packets received out of sequence can be detected by out-of-sequence Sequence Numbers. Note that RTP allows detection of these transport-related problems but leaves it up to the codec to deal with the problem. For example, a video codec may compensate for the loss of a packet by repeating the last video frame, while an audio codec may play background noise for the interval. Variable delay or jitter can be detected by the Timestamp field. A continuous bit rate codec such as PCM will have a linearly increasing Timestamp. A variable bit rate codec, however, which sends packets at irregular intervals, will have an irregularly increasing Timestamp, which can be used to play back the packets at the correct interval.

RTP media sessions are unidirectional they define how media is sent from the media source to the media sink. As such, a normal bidirectional media session is actually two RTP sessions, one in each direction. In a multimedia session established with SIP, the information needed to select codecs and send the RTP packets to the right location is carried in the SDP message body. Under some scenarios, it can be desirable to change codecs during an RTP session. An example of this relates to the transport of dual tone multiple frequency (DTMF) digits. A low bit rate codec that is optimized for transmitting vocal sounds will not transport the superimposed sine waves of a DTMF signal without introducing significant noise, which may cause the DTMF digit receiver to fail to detect the digit. As a result, it is useful to switch to another codec when the sender detects a DTMF tone. Because an RTP packet contains the payload type, it is possible to change codecs on the fly without any signaling information being exchanged between the UAs. On the other hand, switching codecs in general should probably not be done without a SIP signaling exchange (re-INVITE) because the call could fail if one side switches to a codec that the other does not support. The SIP re-INVITE message exchange allows this change in media session parameters to fail without causing the established session to fail. The use of random numbers for SSRC provides a minimal amount of security against “media spamming” where a literally *uninvited* third party tries to break into a media session by sending RTP packets during an established call. Unless the third party can guess the SSRC of the intended sender, the receiver will detect a change in SSRC number and either ignore the packets or inform the user that something is going on. This behavior for RTP clients, however, is not universally accepted, because in some scenarios (wireless hand-

off, announcement server, call center, and so forth) it might be desirable to send media from multiple sources during the progress of a call.

1.11 RTP Control Protocol (RTCP)

The RTP Control Protocol (RTCP) is a related protocol also defined in RFC3550 that allows participants in an RTP session to send each other quality reports and statistics, and exchange some basic identity information. The five types of RTCP packets are shown in Table 12.2. RTCP has been designed to scale for very large conferences. Because RTCP traffic is all overhead, the bandwidth allocated to these messages remains fixed regardless of the number of participants. That is, the more participants in a conference, the less frequently RTCP packets are sent.

The use of reports allows feedback on the quality of the connection including information such as:

• Number of packets sent and received;

• Number of packets lost;

• Packet jitter.

By default, RTCP uses the next highest port from the RTP port, although this can be changed in the SDP offer/answer exchange.

1.12 Audio Codec

Once we have the audio signal represented as a sequence of samples, the next step is to compress it to reduce the consumption of network bandwidth required to transmit the speech to the receiving party.

The compression and decompression is handled by special algorithms we call codecs (COder-DECoder). Let's have a look at some popular codecs that are being used in Voice over IP. All the codecs we list here expect the input to be audio sampled at 8 kHz with 16-bit samples.

1. G.711

G.711 is a codec that was introduced by ITU in 1972 for use in digital telephony, i.e. in ISDN, T.1 and E.1 links. The codec has two variants: A-Law is being used in Europe and in international telephone links, u-Law is used in the U.S.A. and Japan.

G.711 uses a logarithmic compression. It squeezes each 16-bit sample to 8 bits, thus it achieves a compression ratio of 1:2. The resulting bit rate is 64 kbit/s for one direction, so a call consumes 128 kbit/s (plus some overhead for packet headers). This is quite a lot when compared with other codecs. This codec can be used freely in VoIP applications as there are no licensing fees. It works best in local area networks where we have a lot of bandwidth available. It's benefits include simple implementation which does not need much CPU power (can be implemented using a relatively simple table lookup) and a very good perceived audio quality.

2. G.729

G.729 is a codec that has low bandwidth requirements but provides good audio quality (MOS = 4.0). The codec encodes audio in frames, each frame is 10 milliseconds long. Given the sampling frequency of 8 kHz, the 10 ms frame contains 80 audio samples. The codec algorithm encodes each frame to 10 bytes, so the resulting bit rate is 8 kbit/s for one direction. When used in VoIP, we usually send 3-6 G.729 frames in each packet. We do this because the overhead of packet headers (IP, UDP, and RTP together) is 40 bytes and we want to improve the ratio of "useful" information.

G.729 is a licensed codec. As far as end users are concerned, the easiest path to using it is to buy a hardware that implements it (be it a VoIP phone or gateway). In such case, the licensing fee has already been paid by the producer of the chip used in the device.

A frequently used variant of G.729 is G.729a. It is wire-compatible with the original codec but has lower CPU requirements.

3. G.723.1

G.723.1 is a result of a competition that ITU announced with the aim to design a codec that would allow calls over 28.8 and 33 kbit/s modem links. There were two very good solutions and ITU decided to use them both. Because of that, we have two variants of G.723.1. They both operate on audio frames of 30 milliseconds (i.e. 240 samples), but the algorithms differ. The bit rate of the first variant is 6.4 kbit/s and the MOS is 3.9. The bit rate of the second variant is 5.3 kbit/s. The encoded frames for the two variants are 24 and 20 bytes long, respectively.

4. GSM 06.10

GSM 06.10 (also known as GSM Full Rate) is a codec designed by the European Telecommunications Standards Institute for use in the GSM mobile networks. This variant of the GSM codec can be freely used so you will often find it in open source VoIP applications. The codec operates on audio frames 20 milliseconds long (i.e. 160 samples) and it compresses each frame to 33 bytes, so the resulting bitrate is 13 kbit/s (to be precise, the encoded frame is exactly 32 and 1/2 byte, so 4 bits are unused in each frame).

5. Speex

Speex is an open source patent-free codec designed by the Xiph.org Foundation. It is designed to work with sampling rates of 8 kHz, 16 kHz, and 32 kHz and can compress audio signal to bitrates between 2 and 44 kbit/s. For use in VoIP telephony, the most usual choice is the 8 kHz (narrow band) variant.

6. iLBC

iLBC (internet Low Bit Rate Codec) is a free codec developed by Global IP Solutions (later acquired by Google). The codec is defined in RFC3951. With iLBC, you can choose to use either 20 ms or 30 ms frames and the resulting bitrate is 15.2 kbit/s and 13.33 kbit/s, respectively. Much like Speex and GSM 06.10, you will find iLBC in many open source VoIP applications.

Design & Implementation of the PBX

2.1 introduction

This chapter we will divided into two main sections the first will deal with all the aspect of the PBX from hardware selection and subscriber planning. where the second section will deal with the realization of the first section by installing the asterisk and making the needed configuration

2.2 Introduction to ASTERISK:

Asterisk was originally created by Mark Spencer, the CEO and founder of Digium Ltd. While Asterisk is a fully open-source product, Digium manufactures hardware components for connecting to the public telephone network. Digium has a complete range of cards from analog cards to digital interface cards to connect. There are also other cards available from other suppliers and you can use standard modems such as those used in dial-up Internet access.

On September 23, 2004, Mark Spencer released the 1.0.0 version of Asterisk during his keynote speech at the first **AstriCon**, which is the official Asterisk user and developer's conference. Just over a year later, the stable version 1.0.9 was released and is now quite locked down as far as its features go. What started off simply as a side project changed almost overnight when Mark Spencer hooked up with Jim Dixon who was working on the Zapata drivers. All of a sudden Asterisk had a means of connecting to the telephone network. The telephony revolution had begun.

Asterisk is open source. It implements communications in software instead of hardware. This allows new features to be rapidly added with minimal effort. You can easily make your own changes or additions. With its included support, rich set of configuration files, and open source code, every aspect of Asterisk can be customized to meet your needs.

New interfaces and technologies are easily added to Asterisk. With Asterisk you can take control of your communications. Once a call is in your Linux sever with Asterisk, anything can be done with it. Asterisk gives you fine-grained control over every aspect of your communications.

Asterisk is incredibly efficient. A small PC will serve many telephone users. With Asterisk you can easily build a telephone system for the smallest or the largest enterprise, There are Asterisk server running thousands of phones right now. You can easily scale or combine Asterisk systems to serve a number of users in any number of locations. When combined with low-

cost Linux telephony hardware, Asterisk creates a PBX at a fraction of the price of traditional PBX systems. Asterisk provides better functionality than the most expensive proprietary systems. Asterisk includes feature such as voicemail, interactive voice response IVR,) and conferencing which are very expensive in proprietary systems.

With Asterisk, you can make calls through the telephone company, or make calls over the Internet. With the appropriate hardware, Asterisk supports telephony over the PSTN without any Internet connection. It is much cheaper to send telephone calls over the Internet than through the telephone companies. Asterisk can pay for itself with the money you save on your phone bill.

With Asterisk PBX's and Interactive Voice Response (IVR) applications are rapidly created and deployed. The powerful command line interface and feature rich text configuration files support rapid configuration and real-time diagnostics. Asterisk's unusually flexible dial plan allows seamless integration of IVR and PBX functionality. Asterisks Features are easily implemented using nothing more than extension logic. it supports a wide range of protocols for handling and transmitting voice over traditional telephony interfaces. Asterisk supports US and European standard signaling types used in standard business phone systems. This allows Asterisk to bridge between next generation voice-data integrated networks and existing network infrastructure.

Asterisk not only supports traditional phone equipment it provides this equipment with additional capabilities.

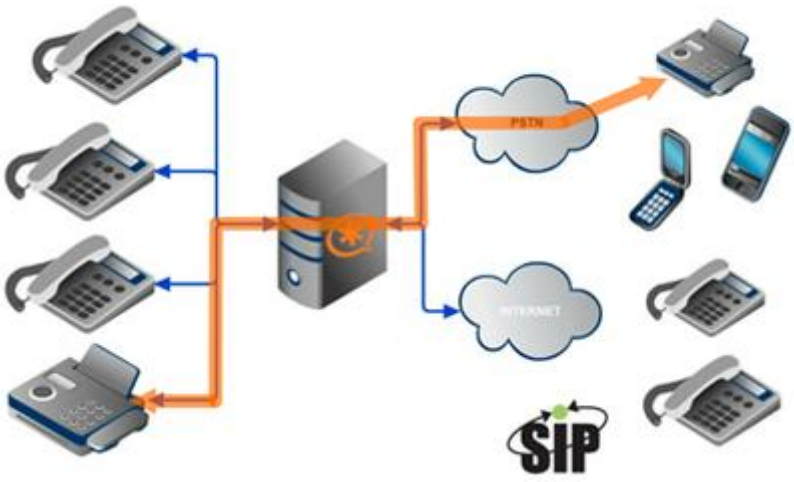

Figure 0-1 Asterisk PBX system

2.3 Asterisk Architecture

Before we dive too far into the various types of modules, let's first take a look at the overall architecture of Asterisk.

Asterisk a big program with many components, with complex relationships. To be able to use it, you don't have to know how everything relates in extreme detail. Below is a simplified diagram intended to illustrate some of the major components of Asterisk. It is useful to understand how a component may relate to things outside Asterisk as Asterisk is not typically operating without some connectivity or interaction with other network devices or files on the local system.

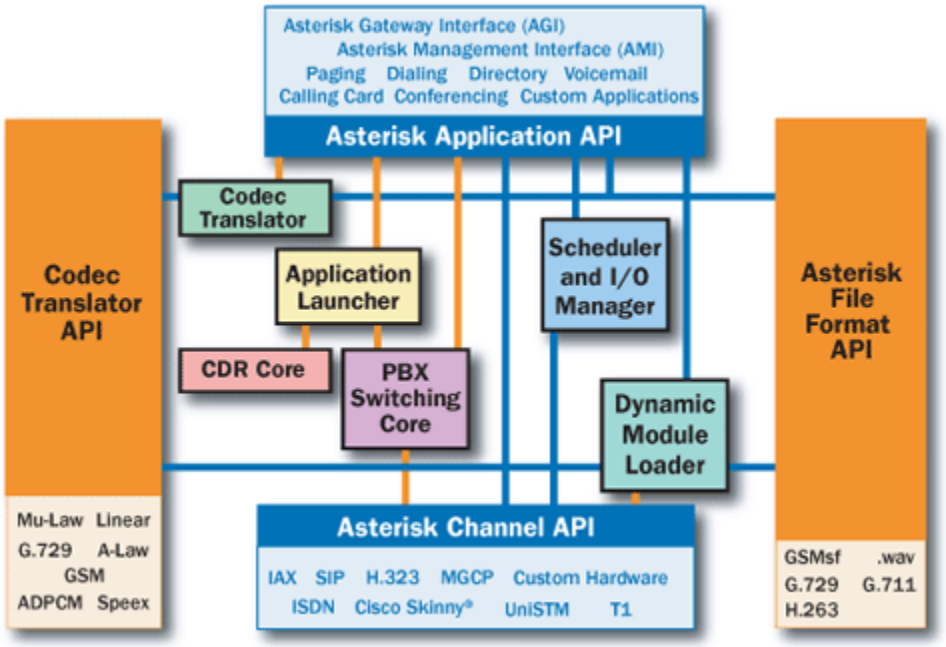

Figure 0-2 Asterisk Architecture

Asterisk has a **core** that can interact with many **modules**. Modules called channel drivers provide **channels** that follow Asterisk **dial plan** to execute programmed behavior and facilitate communication between devices or programs outside Asterisk. Channels often use **bridging** infrastructure to interact with other channels. We'll describe some of these concepts in brief below.

2.3.1 Channels

A channel is the equivalent of a telephone line, but in digital format. It usually consists of an analog or digital (TDM) signaling system or a combination of codec and signaling protocol (e.g. SIP-GSM, IAX-uLaw). In the beginning, all telephony connections were analog and susceptible to echo and noise. Later, most systems were converted to digital systems, with the analog sound converted into digital format by PCM (Pulse Code Modulation) in most cases. This format allows voice transmission in 64 kilobits/second without compression.

Asterisk channels supported:

1. chan_console: supports a sound card (OSS or ALSA) - dial string: console/dsp)
2. chan_sip: supports voice over IP using SIP protocol. – dial string: sip/channel
3. chan_iax: supports voice over IP using IAX2 protocol, - dial string: iax2/channel
4. chan_h323: H.323 is one of the oldest and most implemented voice over IP protocols. It's useful when one attempts to connect to existing H.323 networks. There are different flavors of H.323 in Asterisk: chan_h323 and chan_oh323. A third implementation is being developed by Digium: the first version in asterisk-add-ons subversion. Chan_h323 can be used in Asterisk as a gateway. Asterisk can point to a gatekeeper, but can't work as one. Dial string h323/hostname if using a gatekeeper, h323/extension@hostname if going directly to the gateway.
5. chan_mgcp: supports the voice over IP protocol using MGCP. Currently Asterisk supports MGCP phones, but it cannot connect to a VoIP provider using MGCP. Dial string: MGCP/aaln/1@hostname
6. chan_sccp: Supports Cisco voice over IP skinny protocol. There are two versions: chan_skinny and chan_sccp2.
7. chan_unicall: Implements MFC/R2 as a signaling protocol for E1s used in China, Latin America and several other countries. It is supported by a third party channel driver called Unicall. It uses the chan_unicall channel driver.
8. chan_agent: Used for ACD (Automatic Call Distribution). It is not related to a specific hardware or protocol. It can also be used for mobility, allowing any person to use any phone just by logging in to the agent.
9. chan_local: Is a pseudo channel, it simply loops back into the dialplan in a different context. Useful for recursive routing.

2.3.2 Codecs and codec translation

We usually try to put as many voice connections as possible in a data network. Codecs enable new features in digital voice. Compression is one of the most important ones, since it allows compression rates larger than 8 to 1. Other features include voice activity detection, packet loss concealment and comfort noise generation. There are several codecs available for Asterisk and these codecs can be transparently translated from one to another. Internally, Asterisk uses slinear as the stream format when it needs to convert from one codec to another. Some codecs in Asterisk are supported only in pass-through mode, and these codecs can't be translated.

The following codecs are supported:

• G.711 ulaw (USA) – (64 Kbps).
• G.711 alaw (Europe) – (64 Kbps).
• G.723.1 – Only pass-through mode
• G.726 – (16/24/32/40kbps)
• G.729 – Needs licensing (8Kbps)
• GSM – (12-13 Kbps)
• iLBC – (15 Kbps)
• LPC10 - (2.5 Kbps)
• Speex - (2.15-44.2 Kbps)

2.3.3 Protocols

Sending data from one phone to another should be easy provided that the data find a path to the other phone by themselves. Unfortunately, it doesn't happen this way, and a signaling protocol is necessary in order to establish connections between phones, discover end devices and implement telephony signaling. Recently, it has become very common to use SIP as a signaling protocol. H.323 is largely implemented in voice over IP networks and most legacy implementations use this protocol. IAX is another option that is becoming popular because it works well with NAT traversal and some bandwidth can be saved as well.

Asterisk supports:

• SIP
• H323
• IAXv1 e v2

• MGCP

• SCCP (Cisco Skinny)

• Nortel unistim

2.3.4 Applications

To bridge calls from one phone to another an application named Dial() is used. Most Asterisk features like voicemail and conferencing are implemented as applications. You can see available Asterisk applications by using the “core show applications” console command. You can add

applications from asterisk-add-ons, from third-party providers or even develop some applications yourself.

2.4 Asterisk Features

Amazingly enough, Asterisk has more features than most traditional PBX systems, which are composed of a large box full of hardware. Hence, the Asterisk mantra of it's just software. The following is only a partial list of the many features included with Asterisk:

- **Automated Attendant**: An automated system for answering incoming calls and routing them based on the caller's responses to voice prompts.
- **Blacklists**: Blacklisting is the ability to easily add numbers to a central database that will prevent calls from the blacklisted phone numbers being processed by the system.
- **Call Detail Records**: The detailed call reports and usage statistics to show an administrator the activity of the phone system.
- **Call Forward on Busy**: This feature automatically forwards a call to another extension if the called extension is busy.
- **Call Forward on No Answer**: This feature automatically forwards a call to another extension if the called extension does not answer.
- **Call Parking**: This feature refers to placing a call into a holding state so that it can be picked up at another extension.
- **Call Queuing**: A system that allows inbound callers to sit in a holding room listening to music on-hold until the next available agent is available to speak to them.

- **Call Recording**: The ability to record inbound or outbound calls to .wav files.
- **Call Routing**: Based on the phone number that was dialed (DID) or the number that was called from (ANI), a call can be routed to a specified extension, group, queue, etc.
- **Call Transfer**: This refers to the ability to transfer an existing call to another extension.
- **Caller-ID**: Caller-ID is used to display the phone number and other available information of the user that is calling into the system.
- **Conference Bridging**: Asterisk has the ability to create conference rooms that multiple people can attend at one time for group meetings.
- **Interactive Directory Listing**: A Company directory system that can look up users by first or last name.
- **Interactive Voice Response** (**IVR**): This system uses pre-recorded voice menus to prompt callers to make selections via their phone such as "*press 1 for sales, 2 for support*".
- **Music On-Hold:** Asterisk can play MP3 files to callers who are on-hold or waiting in a queue.
- **Remote Office Support:** Asterisk uses Internet Protocols for communication. Hence, users can be at remote locations and have access to the system via broadband Internet connection.
- **VoIP Gateways**: Using the new Internet Telephone Service Providers (ITSPs), an Asterisk system can have telephone network connectivity without having to use a normal analog service provider.
- **Voicemail:** Each user in an Asterisk system can have their extension and voicemail account. the voicemail can be retrieved via their phone, from a remote location, sent via email.

2.5 PBX hardware

after we have introduced Asterisk let us now have a brief idea about the hardware requirement for our PBX.

2.6 Choosing the Server hardware:

Picking the right server is a key decision when running Asterisk. The last thing a company wants to hear is that their phone system is down. Asterisk *can* run on obsolete hardware, but you will get what you pay for. Reliable, capable equipment is the foundation for any reliable, capable PBX system. So there are many major points that we should take inconsideration the CPU speed, RAM and the size of the hard drive.

1. Processor speed is the most important feature when looking at a server to run Asterisk. The more processing power, the more responsive the system will be when it is placed under heavy call loads. Asterisk runs well on any modern processor, handling moderate call loads without any issue. However, this does depend on how the system is configured to handle calls. Transcoding is when the server is handling a conversation that is coming in with one codec and coverts it on-the-fly to another. This happens a lot more than thought, as most VoIP telephones transmit in μ-Law, which is the standard codec for telephone conversations. If the server is using the GSM codec for outbound calls, it needs to "transcode" the conversation and convert it from μ-Law to GSM. This, by itself, is pretty simple; however, when the server starts having to transcode multiple conversations simultaneously, more processing time is required. Protocol translation is the same problem as transcoding, except instead of converting the audio codec, it needs to translate the protocol used.
2. RAM usage on Asterisk is pretty low. Asterisk can easily fit within a 64MB footprint even on a fairly large install. Since Asterisk is modular, trimming RAM consumption is as easy as removing modules from the startup sequence. A bare bones Asterisk startup can fit within a memory footprint of fewer than 30MB.
3. Storage space is probably one of the least important choices when choosing a server for Asterisk. Hard drives keep getting larger and cheaper with each passing month, allowing even a low-end computer to have massive amounts of space. Asterisk, by itself, hardly takes up any room; however, when voice prompts for Interactive Voice Response (IVR) menus and voice mail start being added to the system, Asterisk's footprint starts growing. Hard drive size needs to be determined by the amount of users on the system and the amount of voice mail expected.

2.7 Terminal Equipments:

Phones are the most important part of a PBX setup. This is how most users interface with the PBX system. Choosing the proper phone is key to a successful PBX deployment. so in this section we will deal with the different phones that can be used for our PBX.

2.7.1 Soft Phones:

The easiest phone to set up with Asterisk is a soft phone. A soft phone is a computer program that emulates a phone on your PC. Soft phones are easy to set up and can be configured in a matter of minutes. They're usually very easy to use, often displaying a telephone-like interface on the screen. We have used in our case X-LITE, Zoiper soft phones.

here an example of how to configure your SIP account in the application for X-LITE soft phone:

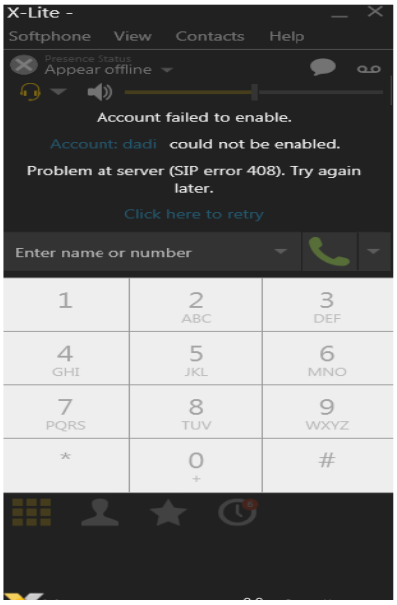

Figure 0-3 Soft phone

2.7.2 Hard Phones

The alternatives to soft phones are hard phones the phones we've used the past years: a physical device that sends and receives telephone calls. Hard phones are on the opposite side of the spectrum from soft phones: they're expensive and often harder to set up than their software

counterparts. However, most users prefer a hard phone; it's what they're accustomed to. The most common hard phones include IP phones: analog phones connected to an Analog Terminal Adapter (ATA) and analog phones connected via interface cards. Each of these has their advantages and disadvantages, which we'll discuss in the following sections.

2.7.3 IP Phones

IP phones are one of the most common solutions you'll see for VoIP in a business environment. They plug in to an Ethernet connection and emulate a regular analog phone. They're made by numerous companies, including Cisco Systems, Polycom, Aastra, and Siemens, just to name a few. The price and quality of these phones run the gamut, but the general rule of "you get what you pay for" applies here. some examples of IP phone are shown the following Fig

Figure 0-4 IP Phone

2.7.4 Analog Telephone Adapters

ATAs are the bridge between the world of analog telephones and the world of VoIP. They are small devices, usually in the form of a small plastic cube, with a power port, one or more telephone jacks, and an Ethernet port. An analog phone connected through an ATA can participate in phone calls on a VoIP network.

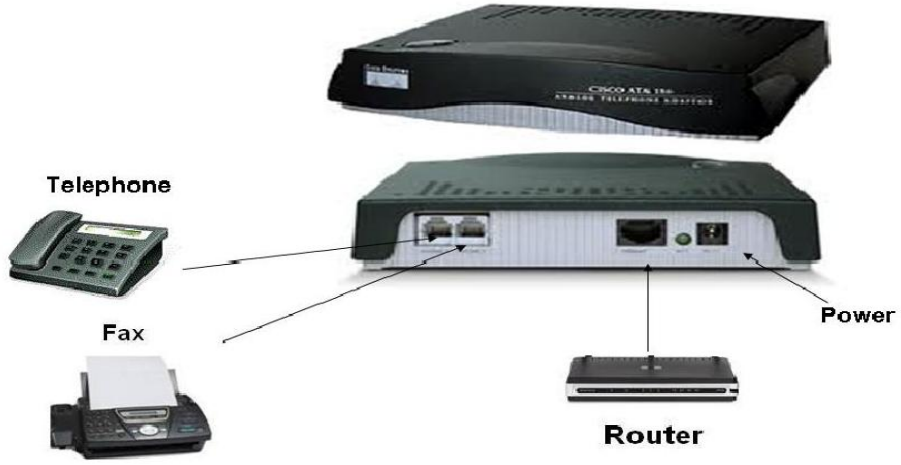

Figure 0-5 Example of ATA

2.7.5 Interface Cards

Analog phones do not always need an ATA. Asterisk supports multiple interface cards that allow analog phones to connect directly to an internal port on the server. Digium sells numerous cards supported by its Zaptel drivers. These cards support anywhere from 1 to 96 phones depending on how they are configured. There are also other cards that support anywhere from a single phone line to an entire T1/E1.

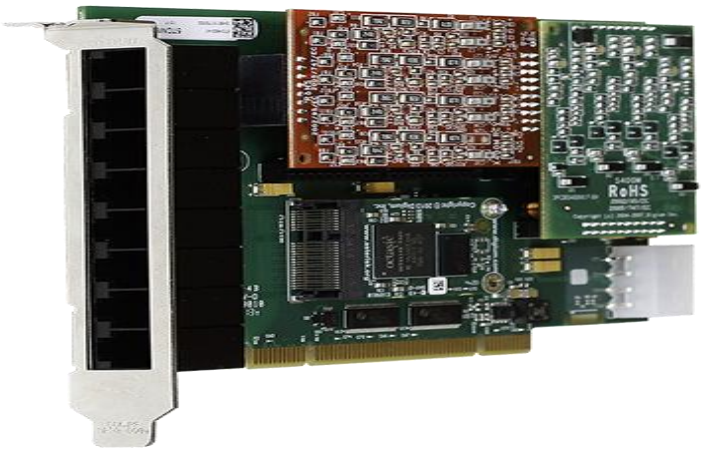

Figure 0-6 ATA 08 Analogue ports interface

we call :

- FXS Foreign eXchange Subscriber interface is the port that actually delivers the analog line to the subscriber. In other words it is the "plug on the wall" that delivers a dial tone, battery current and ring voltage.
- FXO – Foreign eXchange Office interface is the port that receives the analog line. It is the plug on the phone or fax machine, or the plug(s) on your analog phone system. It delivers an on-hook/off-hook indication (loop closure). Since the FXO port is attached to a device, such as a fax or phone, the device is often called the "FXO device".

2.8 Connecting to the PSTN

Asterisk allows you to seamlessly bridge circuit-switched telecommunications networks with packet-switched data networks. Because of Asterisk's open architecture (and open source code), it is ultimately possible to connect any standards-compliant interface hardware. The selection of open source telephony interface boards is currently limited, but as interest in Asterisk grows, that will rapidly change. At the moment, one of the most popular and cost-effective ways to connect to the PSTN is to use the interface cards described earlier.

2.9 Configuring the IP Network

When looking at the IP network from a network management standpoint, VoIP conversations

using the μ-Law codec are 8KB/s data transfers that run for the duration of the calls. While this amount of traffic is negligible if designing a network for an office of ten people, it starts to add up quickly when designing the network for a voicemail server serving 10,000 people. For example, if there are 2500 simultaneous phone calls connecting to and from the server, that would be a constant stream of 20 MB/s being transferred across the network.

When designing networks for VoIP, virtual local area networks (VLANs) are a big help. VLANs are a software feature in networking switches that allow managers to set up virtual partitions inside the network. For example, you can set up a switch to have even-numbered ports on VLAN A and odd-numbered ports on VLAN B. When plugging networking equipment into the switch, equipment on VLAN A won't be able to connect to equipment on VLAN B, and vice versa, allowing the two VLANs to be independent of one another. VLANs help immensely for a

VoIP network since they keep voice and data traffic separate from each other. The last thing you want is a giant multicast session. By keeping the computers on separate VLANs, computer traffic will not interfere with voice traffic, allowing a user to make a large file transfer and not see any degradation of the voice quality on their phone system.

2.10 PBX Implementation

When planning for a production PBX system, we hope to spend more time in the planning stage than in actually building the system. A poorly designed system will make for numerous changes after the system is up and running. Most businesses will not tolerate too many changes after their telephone system is supposed to be fully functional. Proper planning also reduces the administrative burden. We should ensure that we plan adequately in order to create a system that fits the needs of our business, since reconfiguration of a live system can often lead to downtime.

To help plan and deploy our system, we will go through each of the primary functions and see what role these functions play in a deployment plan. We will achieve this through a case studies of our office at work that will give us a realistic view of the planning process as far as possible.

2.11 The Plan

There are a number of areas we need to consider when building our telephone system, such as the physical infrastructure for the stability and security of the system, the need to lock the PBX, the need to provide adequate heating control, and so on. Most of this is very specific to our environment and should be taken in consideration . Besides these, the most important areas, on which this chapter focuses, are those relating to the configuration of the PBX system itself. We will need to consider the following:

- Extensions
- Ring groups
- Call queues
- Connectivity
- DID Lines (Direct Inward Dial)
- Telephones

- Hard phones
- Soft phones

- IVR (Interactive Voice Response)
- Fax requirements

These are the main, or most common, areas of concern when planning our deployment. We will cover each of these in detail and then look at case studies implementing these.

2.11.1 Extensions

If we ask ten telephone engineers about the probable length of our extensions, we will get ten different answers. Some will say to use the shortest length possible to support our actual extensions, others have lookup charts to help figure it out, and still others use some unknown mysticism to conjure up the optimal extension length.

Of course, there are simple methods and a few key points that we have to take into consideration.

2.11.2 Number of Employees

Even in an office of less than ten people, we should never recommend using single-digit extensions. This is extremely limiting not only from a headcount perspective, but also very limiting in terms of voice menus that may play to our advantage. With an Asterisk-based PBX system, a small business with just a few employees could sound like a big company when its clients, vendors, or even competitors call in. Hence, for a business of any size, having an introductory announcement saying "*Thank you for calling the ATM Company; press 1 for Ali, 2 for Khalid, or 3 for Ibrahim*" does not portray much in the way of professionalism. Even if there are only three employees in the company, a far better approach would be to have an announcement that says "*Thank you for calling the ATM Company; if you know the extension of the party you are calling, you may enter it at any time; to speak directly to an agent, press the # key. For Sales press 1, for Billing press 2, for Marketing press 3; please make your selection now*".

This gives a more professional appearance to the company. If this is the first time a person is trying to contact our company, we want them to feel that they are in the hands of professionals of whatever trade or business we are in. It also gives us room for expansion and staff changes. If

our company suddenly grows or an employee is replaced we wouldn't have to reconfigure the phone system to reflect this.

We should keep in mind the actual numbering scheme. For the sake of smooth functionality, we should avoid using extensions that begin with any of the numbers we may use in our IVR menus. If we use menus like "*Press 1 for Sales, 2 for Billing, and 3 for Support*", then we should try to avoid using extensions that begin with 1, 2, or 3. If we cannot avoid using these numbers, it will not seriously impact our system, but it can introduce unwelcome delays while the system is trying to figure out if it needs to go to a menu or an extension. It's also a good idea to group extensions where possible, so we have specific ranges for specific functions of the system.

2.11.3 Departmental Considerations

When planning our extensions, we need to remember that there are more areas of the telephone system, such as ring groups and queues, than the actual phones using extension numbers. From an organizational point of view, most companies will group extensions based on department such as 2xx for Sales, 3xx for Marketing, and 4xx for Support. The same works well for organizing ring groups and queues. Using the last example, we might use 200 for the Sales queue, 300 for the Marketing queue, and so on. While using a fixed method like this may not be the most efficient number scheme in terms of using all the numbers in a range, it does allow for a large amount of growth, changes, and flexibility.

So let us create our extension list, first start by grouping our company into groups such as departments, create a list of queues if we are going to use them, and create a list of ring groups.
for our case we have the following groups:

- Regional manger Office:
    - the regional manger 101
    - the secretary 102
    - Reception 103

and 5 sub direction as follow:

- Technical sub direction: with the following department
    - SDT Manager 201
    - Energy department 211/212

- Transmission Department 221/222
- Acceptance department 231/232
- IT Department 241/242

- Deployment sub direction: with the following department:
  - Sub direction Manger301
  - Civil engineering 311/312
  - Acquisition department 321/322
  - Implementation department 331/332
  - Radio planning department 341/342
- Commercial and marketing sub direction: with the following department:
  - Sub direction Manger 401
  - marketing Department 411/412
  - Direct sell Department 421/422
  - Indirect sell Department 431/432
- Logistics sub direction: with the following department:
  - Sub direction Manger 501
  - Logistic Department 511/512/513
  - Human Resources Department 521/522/523

Each department will have a Ring group and one call queue that contain secretary and reception. For extensions that will also have voicemail, we will also need the following information:

- Voicemail password
- Email address (optional, notifications, and attachments)
- Pager email address (optional, for short notifications)
- Email attachment yes/no (if set to yes, a .wav file will be emailed to the recipient)
- Play caller-id yes/no (if set to yes, caller-id will be read prior to the message)

- Play envelope yes/no (if set to yes, date and time will be read prior to the message)
- Delete voicemail yes/no (if set to yes, voicemail will be deleted after it is emailed)
- Voicemail context (this is used to group people into separate isolated groups such as having multiple companies hosted on the same PBX system)

2.11.4 Ring Groups

A **ring group** is a group of extensions that can all be made to ring at the same time when a single extension is called. This can be a useful feature within an organization as it allows the nearest available user to answer the phone.

Ring groups can be configured as 'ring all', or 'hunt'. When configured as ring all the incoming call will ring at each extension simultaneously, whereas a hunt group will try ringing each extension individually. The important information relating to the ring groups are following:

- The name of the group
- The number assigned to this group
- The ring strategy of the group (a ring all or a hunt group)
- The audio announcement to be played
- The prefix for the caller-id
- The destination to route a call if no one is available (to voicemail or to an operator for example)
- The extensions that are members of this group

for example

Group Name	Group #	Ring Strategy	Announcement	CID Prefix	N/A Destination	Members
Human Resource	2000	Ring All	HR_greet	HR	VM300	521,522,523
Logistic Department	2001	Hunt	Log_greet	Logistics	VM301	511,512,513

Table 2-0-1 Design Example of Ring Group

2.11.5 Call Queues

A call queue is different from a ring group in that the caller is not sent immediately to all the available agents. When a caller is sent to a call queue, they are sent to a virtual holding area to wait for the next available agent. During the wait, they can be listening to music on-hold and be told their position in the queue and the estimated hold time.

When planning call queues we should consider what we want the callers to hear, how much wait callers can tolerate, and how many agents we need online at a particular time. When planning our queues, we should record:

- A unique name for the queue
- A unique number to identify the queue
- A password for access to the queue
- The announcement to be played to the caller periodically
- Category of hold music to be played
- Ring strategy
- Static members of the queue

Queue Name	Queue #	Password	Announcement	On-Hold Music Category	Ring Strategy	Static Agents
Secretary	2000		Secretary_queue	Default	ringall	102,103

Table 0-2 Design Example of Call Queue

Asterisk Installation and configuration

3.1 Introduction

In this chapter we will illustrate the installation and the configuration so that we can setup our PBX and make calls, test the features .

3.2 Installing asterisk

We will install Asterisk on Debian Lunix or any variant. All required packages will be installed via apt, but Asterisk will be installed from source. Let's see how to install Asterisk on Debian.

There are many versions of Asterisk available on their website, but the latest is 13 with LTS (Long Term Support). So we will download it from source and installation steps in the following sections.

1. **Update Your System:**

Open a terminal window and we Switch to the Root User by typing the Command *sudo -i*

then we download and install the updates:

apt-get update && apt-get upgrade -y

2. **Install Required Dependencies**

apt-get install -y build-essential linux-headers-`uname -r` openssh-server apache2 mysql-server\ mysql-client bison flex php5 php5-curl php5-cli php5-mysql php-pear php5-gd curl sox\libncurses5-dev libssl-dev libmysqlclient-dev mpg123 libxml2-dev libnewt-dev sqlite3\libsqlite3-dev pkg-config automake libtool autoconf git unixodbc-dev uuid uuid-dev\libasound2-dev libogg-dev libvorbis-dev libcurl4-openssl-dev libical-dev libneon27-dev libsrtp0-dev\ libspandsp-dev libmyodbc

3. **Reboot server** *reboot*

4. **Install Legacy pear requirements:**

we insure that we are logged as root and we type in the terminal :

pear install Console_Getopt

5. **Install and Configure Asterisk:**

- *cd /usr/src*
- *wget http://downloads.asterisk.org/pub/telephony/dahdi-linux-complete/dahdi-linux-complete-current.tar.gz*
- *wget http://downloads.asterisk.org/pub/telephony/libpri/libpri-1.4-current.tar.gz*
- *wget http://downloads.asterisk.org/pub/telephony/asterisk/asterisk-13-current.tar.gz*
- *wget -O jansson.tar.gz https://github.com/akheron/jansson/archive/v2.7.tar.gz*
- *wget http://www.pjsip.org/release/2.4/pjproject-2.4.tar.bz2*

6. **Compile and install DAHDI** (optional):

- *cd /usr/src*
- *tar xvfz dahdi-linux-complete-current.tar.gz*
- *rm -f dahdi-linux-complete-current.tar.gz*
- *cd dahdi-linux-complete-**
- *make all*
- *make install*
- *make config*
- *cd /usr/src*
- *tar xvfz libpri-1.4-current.tar.gz*
- *rm -f libpri-1.4-current.tar.gz*
- *cd libpri-**
- *make*
- *make install*

7. **Compile and install pjproject**

- *cd /usr/src*
- *tar -xjvf pjproject-2.4.tar.bz2*
- *rm -f pjproject-2.4.tar.bz2*
- *cd pjproject-2.4*

- *CFLAGS='-DPJ_HAS_IPV6=1' ./configure --enable-shared --disable-sound --disable-resample --disable-video --disable-opencore-amr*
- *make dep*
- *make*
- *make install*

8. Compile and Install jansson

- *cd /usr/src*
- *tar vxfz jansson.tar.gz*
- *rm -f jansson.tar.gz*
- *cd jansson-**
- *autoreconf -i*
- *./configure*
- *make*
- *make install*

9. Compile and install Asterisk

- *cd /usr/src*
- *tar xvfz asterisk-13-current.tar.gz*
- *rm -f asterisk-13-current.tar.gz*
- *cd asterisk-**
- *contrib/scripts/install_prereq install*
- *./configure*
- *contrib/scripts/get_mp3_source.sh*
- *make menuselect*

You will be prompted at the point to pick which modules to build. Most of them will already be enabled, but if you want to have MP3 support (eg, for Music on Hold), you need to manually turn on 'format_mp3' on the first page

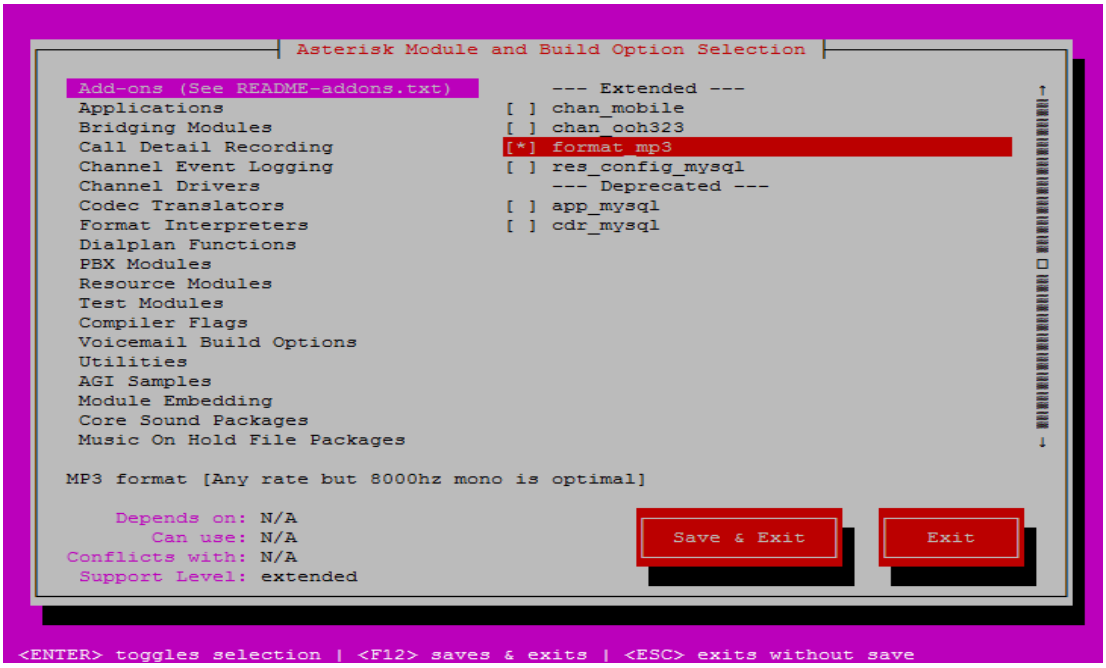

Figure 3-1 Asterisk Menu to choose the needed Modules

After selecting 'Save & Exit' you can then continue

- *make*
- *make install*
- *make config*
- *ldconfig*
- *update-rc.d -f asterisk remove*

10. **Install Asterisk Soundfiles**

- *cd /var/lib/asterisk/sounds*
- *wget http://downloads.asterisk.org/pub/telephony/sounds/asterisk-core-sounds-en-wav-current.tar.gz*
- *wget http://downloads.asterisk.org/pub/telephony/sounds/asterisk-extra-sounds-en-wav-current.tar.gz*
- *tar xvf asterisk-core-sounds-en-wav-current.tar.gz*
- *rm -f asterisk-core-sounds-en-wav-current.tar.gz*
- *tar xfz asterisk-extra-sounds-en-wav-current.tar.gz*
- *rm -f asterisk-extra-sounds-en-wav-current.tar.gz*

- *# Wideband Audio download*
- *wget http://downloads.asterisk.org/pub/telephony/sounds/asterisk-core-sounds-en-g722-current.tar.gz*
- *wget http://downloads.asterisk.org/pub/telephony/sounds/asterisk-extra-sounds-en-g722-current.tar.gz*
- *tar xfz asterisk-extra-sounds-en-g722-current.tar.gz*
- *rm -f asterisk-extra-sounds-en-g722-current.tar.gz*
- *tar xfz asterisk-core-sounds-en-g722-current.tar.gz*
- *rm -f asterisk-core-sounds-en-g722-current.tar.gz*

Asterisk is now installed and ready to use. You can login to asterisks console by the following command: *asterisk -cvvvvvvvv*

3.3 Asterisk configuration

after we have installed the PBX software we need to make some configuration so that we can make calls, we will use SIP as our communication protocol.

there exist three major steps in Asterisk configuration:

Channel configuration: in our case we will configure SIP channels and this is done by editing the file SIP.conf .

Users : we will define the user Data in the file Users.conf

Dial plan configuration : we need to edit Extension.conf

and other optional configuration that need to be customized depending on the needs : voicemail, music on hold...

all the configuration files are contained in the directory /etc/asterisk

3.3.1 Asterisk config sip.conf

The SIP# channel module is arguably the most mature and feature-rich of all the channel modules in Asterisk. This is due to the enormous popularity of the SIP protocol, which has taken over the VoIP/telecom industry and been implemented in thousands of devices and PBXs. If you look through the *sip.conf* file in the *./configs* subdirectory of your Asterisk source you will notice a wealth of options available. Fortunately, the default options are normally all you need, and

therefore you can create a very simple configuration file that will allow most standard SIP telephones to connect with Asterisk. The first thing you need to do is edit the configuration file in your */etc/asterisk* directory called *sip.conf.*

```
[general]
bindport = 5060
bindaddr = 192.168.1.9
context = default
disallow = all
allow = ulaw
maxexpirey = 120
defaultexpirey = 80

[101]
type=friend
secret= 101
host= dynamic
context= Gm

[102]
type=friend
secret=102
host=dynamic
context= Gm
```

the *sip.conf* file is divided to many parts .

1. General section

The SIP file is read from top down. The first section contains the global parameters [general]. The main options are:

• allow/disallow: Defines which codecs can be used.

• bindaddr: Address to be bond to Asterisk SIP listener. If you set it up as 0.0.0.0 (default) it will bind to all interfaces.

• context: Sets the default context for all clients unless changed in the client section.

• bindport: SIP UDP port to listen.

• maxexpirey: Maximum time to register (seconds)

• defaultexpirey: Default time to register (seconds)

• register: Registers Asterisk to another Host.

2. Clients section

After finishing the general sections, it is time to set up the SIP clients.

• [name]: When a SIP device connects to Asterisk, it uses the username part of the SIP URI to find the peer/user.

- type: Configures the connection class. Options are peer, user, and friend:
    - peer: Asterisk sends calls to a peer.
    - user: Asterisk receives calls from a user.
    - friend: Both at same time.
- host: IP address or host name. The most common option is “dynamic”, used when the host registers to Asterisk.
- secret: Password to authenticate peers and users

3.3.2 User configuration

users.conf is a configuration file aimed at defining a "user". It can define a user with an optional SIP phone, IAX2 phone, Zaptel phone and/or just about any other type of phone. Some dial plan can be generated for the user. here is an example

```
[general]
; Full name of a user
;
fullname = Secretary
;
; Starting point of allocation of extensions
;
```

```
userbase = 102
;
; Create voicemail mailbox and use use macro-stdexten
;
hasvoicemail = yes
;
; Set voicemail mailbox 102 password to 1234
;
vmsecret = 1234
;
; Create SIP Peer
;
hassip = yes
;
; Create IAX friend
;
hasiax = no
;
; Create H.323 friend
;
;hash323 = no
; Remaining options are not specific to users.conf entries but are general.
;
callwaiting = yes
threewaycalling = yes
callwaitingcallerid = yes
transfer = yes
canpark = yes
cancallforward = yes
callreturn = yes
callgroup = 1
```

```
pickupgroup = 1

[101]
fullname = Regional Manger
email =
secret = 1234
zapchan = 1
hasvoicemail = yes
vmsecret = 1234
hassip = yes
hasiax = no
hash323 = no
hasmanager = no
callwaiting = yes
context = Manger
```

3.3.3 DIAL PLAN

Dial plan is Asterisk's heart. It defines how Asterisk handles each and every call to the PBX. It consists of extensions that make an instruction list for Asterisk to follow. Instructions are fired by digits received from the channel or application. In order to configure Asterisk successfully, it is crucial to understand the dial plan. Most of the dial plan is contained in the extensions.conf file at the /etc/asterisk directory. This file uses the simple group grammar and has four major concepts:

- Extensions
- Priorities
- Applications
- Contexts

Let's now create a basic dial plan. If you installed the sample files (make samples), the extensions.conf already exists. Save it with another name and start with a blank file.

1. Extensions

The dial plan is a set of defined extensions. An extension is a string that will trigger an event when a call is made. Extensions can be either literal or pattern.

example :

exten=>100,1,Dial(SIP/100,20)

exten=>100,2,hangup()

The instruction “exten” describes the next step for the call. 100 is the set of n digits received (called number). The numbers “1” and “2” are priorities that define the execution order. Dialing “100” will ring the SIP IP phone registered as “100”. If the call is unanswered in 20 seconds it will hang-up.

Extension Syntax:

exten=> number (name), {priority|label{+|-}offset}[(alias)],application

The extension command “exten=” is followed by an extension number or name, a comma, a priority, another comma and, finally, the application.

Extension is the matching address of the call (the phone number). Priorities are used to execute the steps in order of priority. Application is the action to be taken (dial, playback, hangup). Each action has a different application.

2. Priorities

Priorities are numbered steps for execution in each dialed extension. Each priority calls a specific application. Usually, the numbers start with “1” and increment by 1 to each line in the extension definition. In version 1.2, it is common to use the “n” priority as next, instead of manually assigning the numbers. If numbers are not sequential, execution is aborted.

3. Applications

Applications play an important role in Asterisk. They handle voice channels, playing tones, the acceptance of digits called to the PBX, as well as call hang-up. Applications can be called with options that affect their behavior. You can use ‘core show applications’ in the Asterisk’s command line interface to show available applications. When you build your first dial plan you will start to understand what applications are appropriated.

CLI>core show applications

4. Contexts

Contexts play an important role in Asterisk's dial plan configuration and security. Contexts define a scope, which allows separation of the dial plan into different parts. It is important to understand that contexts are intimately bond to channels. When a Asterisk receives a phone call, the call is processed within its incoming context section. The incoming context is always defined by the channel configuration file (iax.conf, sip.conf, zap.conf…).

In our case, for example, we have two classes of users: "managers" and "subordinate". Suppose now that want to play different messages for "subordinate" and "managers" when they dial "900". You can accomplish this by defining the incoming context in the channel files (sip.conf, iax.conf, zap.conf).

In the example below, when 211 dials 900, it receives the message "you are a guest". When 100 dials the same number, it receives a different message: "you are a manager".

sip.conf

```
[101]
context=managers
host=dynamic
…
[211]
context=guests
host=dynamic
…
```

extensions.conf

```
[managers]
exten=>900,1,Playback(youareamanager)
[guests]
exten=>900,1,Playback(youareaguest)
```

Contexts receive a name inside brackets ([]). All instructions after the definition are part of the context. To start a new context, simply insert the new context. A context ends when another

starts. There are two special contexts in the extensions.conf file. The context [globals] is used to define variables whereas the context [general] is used to define some general options.

Conclusion

According to our abstract , our work deals with the implementation of a PBX . We have divided our work into two main section, the first one deals with the theoretical part and the second part is the design one.

the purpose of the first part is to provide a theoretical back ground principle of what is a PBX and Voip, we have introduced protocols widely used in telecommunication field like H332,SIP and other also we explained how the calls are processed we have well explained the SIP protocol because its widely used in commercial life .

the second part we have introduced the Asterisk software that we have used as the engine of our PBX and highlighted the different hardware needed so that we can make a functional PBX

we have take as a case study our regional office because it is a good case so that we can see the performance of our system, so we have make our design on a real situation . we have in section dealed with the different aspect of the implementation starting from the installation of Asterisk to the configuration of the PBX.

the world of asterisk is very big and we hope that we have contributed by this modest work in attracting the attention of our colleagues and Professors to this world.

we can recommend as future work the following:

- the implementation of a graphical interface using Python or any other programming language
- the design and implementation of an IP phone using a DSpic or any other Microcontroller.
- the design and implementation of an FXO FXS interfaces using microcontroller

Bibliography

- **Alan B. Johnston** "SIP Understanding the Session Initiation Protocol, Second Edition"
- Rosenberg, J., et al., "SIP: Session Initiation Protocol," RFC 3261, 2002.
- Rosenberg, J., "A Presence Event Package for the Session Initiation Protocol (SIP)," IETF Internet-Draft, Work in Progress, January 2003.
- Roach, A., "Session Initiation Protocol (SIP)-Specific Event Notification," RFC 3265, 2002.
- Benjamin Jackson, Champ Clark III and Larry Chaffin and Johnny Long "Asterisk Hacking" 2007
- David Gomillion and Barrie Dempster "Building Telephony Systems with Asterisk" 2005
- Flavio E. Gonçalves and Revision: Luis F. Gonçalves "Asterisk PBX Configuration Guide" 2006
- Jim Van Meggelen, Jared Smith, and Leif Madsen "AsteriskTMThe Future of Telephony" 2006
- https://wiki.asterisk.org
- www.asterisk.org