CENTRO UNIVERSITÁRIO DE BRASÍLIA - UNICEUB

ADRIANO LUIS NISHIYAMA

SOLUÇÕES DE IPBX BASEADO EM SOFTWARE LIVRE

Brasília - DF
2006

ADRIANO LUIS NISHIYAMA

SOLUÇÕES DE IPBX BASEADO EM SOFTWARE LIVRE

Trabalho apresentado à banca examinadora da Faculdade de Ciências Exatas e Tecnológicas, para conclusão do curso de Engenharia da Computação. Prof. Orientador: M.Sc. Antonio José Gonçalves Pinto.

Brasília - DF
2006

ADRIANO LUIS NSIHIYAMA

SOLUÇÕES DE IPBX BASEADO EM SOFTWARE LIVRE

COMISSÃO EXAMINADORA

_____

Prof. Orientador Antonio José Gonçalves Pinto

_____

Aderlon Marcelino Queiroz

_____

Cláudio Penedo

Brasília, 03 de julho de 2006.

Aos meus pais, irmãos e meus amigos.

AGRADECIMENTOS

Ao meu pai, Katsugi Nishiyama, à minha mãe, Helena Kimie Nishiyama, e aos meus irmãos, Alexandre Luis Nishiyama e Rodrigo Luis Nishiyama.

Aos meus amigos que moram na Área Octogonal Sul, que sempre estiveram comigo nessa longa caminhada.

Às minhas amigas, Larissa Pretti e Heloisa Viera Curvelo, por estarem sempre ao meu lado e me incentivarem e apoiarem em todos os momentos.

Ao meu professor orientador, M.Sc. Antonio José Gonçalves Pinto, pelo apoio e incentivo.

E a todo corpo docente da instituição UniCeub, meu muito obrigado.

“Se você quer ter sucesso na vida, tem que ter dedicação total, buscar seu último limite e dar o melhor de si.” (Airton Senna)

RESUMO

A tecnologia de voz sobre IP (VoIP) possibilita a transmissão de voz e de dados em uma única rede de serviços. Essa tecnologia utiliza a técnica de comutação de pacotes, ao contrário das redes de telefonia convencional que utilizam a comutação de circuitos.

O VoIP tem como um dos atrativos principais, proporcionar novos tipos de serviços além dos encontrados na telefonia convencional. Tendo em vista essa possibilidade, a proposta deste projeto tem como finalidade implementar uma solução de IPBX (*Internet Protocol Branch Exchange*) de alta disponibilidade baseado no software livre Asterisk.

Nesta implementação serão utilizados além do software livre Asterisk, o protocolo SIP (*Session Initiation Protocol*), onde serão definidos os planos de discagem e o protocolo URCAP (Userland Common Address Redundancy Protocol), que possibilitará a implementação de um serviço de alta disponibilidade.

ABSTRACT

The voice over IP technology (VoIP) enables the transmission of voice and data in the same net of services. This technology uses the commutation technical of packages, unlike the telecommunication net that uses the commutation of circuits.

One of the main attractives of VoIP is that it proportionates new kinds of services, besides the ones which are found in the conventional telephony. In view of this possibility, the proposal of this project is implement a high IPBX availability solution based on the free software Asterisk.

In this implementation, besides the free software Asterisk, it will be used the SIP protocol, where the planes of dialing will be defined, and the URCAP protocol which will enable the implementation of a high availability service.

SUMÁRIO

AGRADECIMENTOS **5**

RESUMO **7**

ABSTRACT **8**

LISTA DE TABELAS **11**

LISTA DE FIGURAS **12**

LISTA DE ABREVIAÇÕES **13**

CAPÍTULO 1. INTRODUÇÃO **14**
1.1 MOTIVAÇÃO 14
1.2 OBJETIVOS 15
1.3 METODOLOGIA 15

CAPÍTULO 2. TELEFONIA **17**
2.1 HISTÓRIA DA TELEFONIA 17
2.2 CENTRAIS TELEFÔNICAS 19
2.3 MULTIPLEXAÇÃO DOS TELEFONES 21
2.4 PRINCÍPIOS DE SINALIZAÇÃO 24
2.5 CODIFICAÇÃO DA VOZ 25
2.6 PBX 27

CAPÍTULO 3. VOZ SOBRE O PROTOCOLO INTERNET **30**
3.1 INTRODUÇÃO 30
3.2 CODIFICAÇÃO DO SINAL 31
3.3 PROTOCOLO TCP/IP 32
3.3.1 PROTOCOLO IP 33
3.3.2 PROTOCOLO TCP 34
3.3.3 PROTOCOLO UDP 34
3.4 QUALIDADE DE SERVIÇO 36
3.4.1 PARÂMETROS DE QUALIDADE DE SERVIÇO 36
3.4.1.1 TAXA DE TRANSMISSÃO 37
3.4.1.2 ATRASO (DELAY) 37
3.4.1.3 JITTER 38
3.4.1.4 PERDAS DE PACOTE 39
3.4.2 ARQUITETURAS UTILIZADAS PARA PROVER QOS 40
3.4.2.1 ARQUITETURA INTSERV 40
3.4.2.2 ARQUITETURA DIFFSERV 43
3.4.3 RTP/RTCP 44
3.5 PROTOCOLOS DE SINALIZAÇÃO VOIP 47
3.5.1 IAX 48
3.5.2 H.323 48
3.6 SIP 50

3.6.1 MENSAGEM SIP 51
3.6.2 SDP 53
3.6.3 ENDEREÇAMENTO SIP 54
3.6.4 SINALIZAÇÃO SIP 55
3.6.4.1 SINALIZAÇÃO DIRETA 55
3.6.4.2 SINALIZAÇÃO INDIRETA 56
3.6.5 SIP X H.323 57

CAPÍTULO 4. IMPLEMENTAÇÃO DO PABX **60**
4.1 ASTERISK 61
4.1.1 ARQUITETURA DO ASTERISK 62
4.1.1.1 CANAIS 64
4.1.1.2 CODECS 65
4.1.1.3 PROTOCOLOS 66
4.1.1.4 APLICAÇÕES 66
4.1.2 APLICAÇÕES IMPLEMENTADAS 66
4.1.3 INSTALAÇÃO E CONFIGURAÇÃO BÁSICA DO ASTERISK 68
4.1.3.1 PROTOCOLO SIP 70
4.1.3.2 PLANO DE DISCAGEM 73
4.1.4 SERVIÇOS AVANÇADOS 75
4.1.4.1 CAPTURA DE CHAMADA 75
4.1.4.2 SALAS DE CONFERÊNCIA 76
4.1.4.3 MÚSICA EM ESPERA 77
4.1.4.4 ESTACIONAMENTO DE CHAMADA 77
4.1.4.5 TRANSFERÊNCIA DE CHAMADAS 78
4.2 CLUSTER DE ALTA DISPONIBILIDADE 80
4.2.1 IMPLEMENTAÇÃO DO UCARP 83

CAPÍTULO 5. TESTES E RESULTADOS **87**
5.1 ESTABELECIMENTO DE CHAMADA ENTRE DOIS TERMINAIS 88
5.2 MUSICA EM ESPERA 88
5.3 TRANSFERÊNCIA DE CHAMADA 89
5.4 CONFERÊNCIA 89
5.5 CAPTURA DE CHAMADA 89
5.6 ESTACIONAMENTO DE CHAMADA 90
5.7 DISTRIBUIÇÃO AUTOMÁTICA DE CHAMADA 90
5.8 CLUSTER 91

CAPÍTULO 6. CONCLUSÃO **92**

REFERÊNCIAS BIBLIOGRÁFICAS **93**

CAPÍTULO 7. ANEXOS **95**
7.1 ANEXO 1 – SIP.CONF 95
7.2 ANEXO 2 – EXTENSIONS.CONF 106
7.3 ANEXO 3 – FEATURES.CONF 114

LISTA DE TABELAS

Tabela 2.1 - Código Morse 17
Tabela 3.1- Padrões de codificação ITU-T 29
Tabela 3.2 - Protocolos de Aplicação disponível na Arquitetura TCP/IP 36
Tabela 3.3 - Sensibilidade dos serviços de acordo com os parâmetros QoS .. 40
Tabela 3.4 - Padrões referenciados na Recomendação H.323 48
Tabela 3.5 - Classes de resposta e respectivas funcionalidades no SIP 53

LISTA DE FIGURAS

Figura 2.1 - Penetração da tecnologia digital nas redes telefônicas 22
Figura 2.2 - Multiplexação FDM 23
Figura 2.3 - Multiplexação TDM. 23
Figura 2.4 - Digitalização da Voz. 25
Figura 2.5 - Amostra de um sinal analógico. 26
Figura 3.1 - Arquitetura TCP/IP 33
Figura 3.2 - Atraso (Delay) de pacote. 38
Figura 3.3 – Jitter. 39
Figura 3.4 - Modelo de referência para implementação da arquitetura IntServ 41
Figura 3.5 - Exemplo de um modelo RSVP 43
Figura 3.6 - Redefinição do campo ToS do datagrama IPv4 44
Figura 3.7 - Cabeçalho RTP 46
Figura 3.8 - Modelo de sinalização direta 55
Figura 3.9 - Modelo de sinalização indireta 57
Figura 3.10 - Comparativo de sinalização SIP X H.323 58
Figura 4.1 - Cenário de implementação 60
Figura 4.2 - Arquitetura Asterisk 63
Figura 4.3 - Cenário do cluster 83

LISTA DE ABREVIAÇÕES

CRLF - Carriage Return Line Feed
CU – Atualmente não Usado
DSCP – Valor Especifico do Serviço Diferenciado
FDM - Multiplexação por Divisão de Freqüência
HTTP - Protocolo de Transferência de Hipertexto
IANA – Autoridade de Numeros Atribuidos da Internet
IAX – Permutação entre Asterisk
IETF - Força Tarefa de Engenharia da Internet
IP - Protocolo de Internet
PABX - Central Telefônica Automática Privada
PAX – Central de Ramais Automático
PBX - Central Telefonica Privada
PCM - Modulação por Código Associado a Pulsos
QoS - Qualidade de Serviço
RDSI - Rede Digital de Serviços Integrados
RFC - Requerimento para comentários
RSVP – Protocolo de Reserva de Recurso
RTCP - Protocolo de Controle de Transporte em Tempo Real
RTP - Protocolo de Transporte em Tempo Real
RTPC - Rede Telefonia Pública Comutada
SDP - Protocolo de Descrição de Sessão
SIP - Protocolo de Inicialização de Sessão
SMTP - Protocolo de Transferência de Simples Mensagens
TCP - Protocolo de Transmissão de Sessão
TDM - Multiplexação por Divisão de Tempo
ToS - Tipo de Serviço
UDP - Protocolo de Datagramas de Usuário
VoIP - Voz sobre o Protocolo Internet

CAPÍTULO 1. INTRODUÇÃO

1.1 MOTIVAÇÃO

A motivação desse projeto está relacionada às transformações que estão ocorrendo no ramo das Telecomunicações. A idéia de convergência e integração de serviços vem ganhando força atualmente, principalmente com o sucesso da Internet. Vários serviços estão aparecendo e alguns ganhando destaque, como por exemplo, o VoIP.

A tecnologia VoIP vem ganhando reconhecimento, principalmente pelo sucesso das empresas que a adotaram nos seus portifólios de serviço. O Skype e a Vonage são alguns casos de empresas que obtiveram sucesso na utilização dessa tecnologia, principalmente por serem empresas que não possuem nenhuma infra-estrutura de rede, garantindo o fornecimento de seus serviços utilizando a plataforma pública da Internet e a disponibilidade de largura de banda de outras operadoras.

Além de possibilitar o surgimento desse novo tipo de prestação de serviço, a tecnologia VoIP está possibilitando o ingresso de empresas que atuam em outros segmentos. As operadoras de TV a cabo são um exemplo das oportunidades criadas pelo surgimento dessa tecnologia. Essas empresas que antes eram apenas provedoras de serviço de TV por assinatura, estão hoje oferecendo acesso à Internet e concentrando seus esforços para conquistar uma parcela do mercado de telefonia, através da tecnologia VoIP (Voz over IP).

Outra motivação encontrada na escolha do projeto, está relacionada as possibilidades que a implementação da tecnologia VoIP pode trazer ao

mercado. Nesse sentido, procurou-se, ao longo desse projeto, constatar a possibilidade de agregar novas funcionalidades aos serviços de telefonia, sendo essa, uma justificativa para a adoção por parte das empresas, principalmente de Telecomunicações, em investirem nessa tecnologia.

1.2 OBJETIVOS

Esse projeto tem como objetivo dar continuidade ao projeto do ex-aluno da instituição UniCeub, Flávio de Castro Carneiro, cujo o projeto visava aumentar a confiabilidade de um sistema de telefonia IP (Internet Protocol). O projeto consistia na implementação de PABX (Private Automatic Branch Exchange) utilizando um ambiente de voz sobre protocolo de rede Internet (VoIP) com dois *callmanage*rs, gerenciadores de chamada, em cluster, ativo-passivo, para aumentar a disponibilidade do ambiente.

Visando contribuir positivamente ao projeto citado, tanto em termos qualitativos quanto funcionais, o foco agora é agregar valores para que a sua implementação se torne ainda mais atrativa. **Para isso, o objetivo desse trabalho é incorporar serviços que usualmente são implementados na telefonia corporativa, mais especificamente, no PABX**. Esses serviços serão citados e explicados mais à frente.

1.3 METODOLOGIA

Esse trabalho está organizado em seis capítulos. Os primeiros três capítulos fazem a apresentação do tema do projeto, fornecendo o embasamento teórico e a tecnologia implementada. Nos capítulos seguintes são analisadas as abordagens e as técnicas discutidas durante os capítulos iniciais. O trabalho encerra com as conclusões e as sugestões de estudos futuros no último capítulo.

Sendo assim, a organização desse trabalho pode ser detalhada da seguinte forma:

- Capítulo 1: O capítulo inicial traz a introdução do assunto que será abordado ao longo do trabalho, especificando a motivação, os objetivos e a metodologia do estudo.
- Capítulo 2: Nesse capítulo serão abordados os assuntos referentes à telefonia convencional, discutindo-se de forma sucinta, alguns conceitos relevantes sobre a telefonia sem entrar em muitos detalhes;
- Capítulo 3: Depois de introduzir alguns conceitos sobre a telefonia no capítulo anterior, o objetivo nesse capitulo é se aprofundar nos conceitos sobre a tecnologia VoIP. Ainda nesse capítulo, serão abordados alguns conceitos fundamentais relacionados a essa tecnologia;
- Capítulo 4: Após os dois capítulos de revisão bibliográfica, nesse capítulo será apresentado o cenário de implementação do projeto, bem como a implementação e a configuração do sistema proposto;
- Capítulo 5: Nesse capítulo serão apresentados os resultados obtidos durante a realização dos testes;
- Capítulo 6: Esse é o capitulo de conclusão da monografia. Aqui serão apresentadas as considerações finais do trabalho. Também serão apresentadas algumas propostas para trabalhos futuros.

CAPÍTULO 2. TELEFONIA

2.1 HISTÓRIA DA TELEFONIA

A necessidade de criar mecanismos através dos quais o homem pudesse interagir a longas distâncias fez com que a expansão dos horizontes da comunicação se tornasse uma contingência. Essa idéia pode ser constatada já na da Idade Média, onde grandes imperadores, devido às suas conquistas, sentiram a necessidade de utilizar um mecanismo através do qual eles pudessem ter uma atuação mais presente em seus vastos territórios. Como solução para esse problema foram utilizados mensageiros que transportavam as ordens escritas pelo imperador e as entregavam ao seu destinatário. Apesar de ser um mecanismo muito simples, lento e sem confiabilidade, teve um papel importante para aquela época por ser um meio de comunicação pioneiro a longas distâncias. (FERRARI, 2005)

Em 24 de maio de 1844, Samuel Morse revolucionou os meios de comunicação transmitindo entre Washington e Baltimore, EUA, a primeira mensagem utilizando como meio de transmissão as linhas metálicas. Esse sistema de comunicação foi batizado com o nome de TELEGRAFIA e utilizava a codificação de letras em sinais elétricos. (FERRARI, 2005)

Com o invento do telégrafo e do código de sinais, mostrado na Tabela 2.1, Samuel Morse deu inicio a um novo conceito de comunicação chamado de Telecomunicações. Esse novo conceito que surge tem como objetivo transmitir “palavras” a longas distâncias entre um emissor e um receptor, através de sinais, em um meio de transmissão.

Passados mais 30 anos, Alexander Graham Bell e seu ajudante

Thomas A. Watson desenvolveram um aparelho de telecomunicações com fins de transmitir sons por meio de sinais elétricos. Esse dispositivo foi patenteado no dia 7 de Março de 1876 e recebeu o nome de Telefone.(FERRARI, 2005)

Tabela 2.1 - Código Morse

a	• —	l	• — • •	x	— • • —	1	• — — — —
b	— • • •	m	— —	y	— • — —	2	• • — — —
c	— • — •	n	— •	z	— — • •	3	• • • — —
d	— • •	o	— — —	ch	— — — —	4	• • • • —
e	•	p	• — — •	w	• — —	5	• • • • •
f	• • — •	q	— — • —	ä	• — • —	6	— • • • •
g	— — •	r	• — •	é / ë	• • — • •	7	— — • • •
h	• • • •	s	• • •	ï	— • • — —	8	— — — • •
i	• •	t	—	ñ	— — • — —	9	— — — — •
j	• — — —	u	• • —	ö	— — — •	0	— — — — —
k	— • —	v	• • • —	ü	• • — —		

O telefone foi descoberto devido às pesquisas desenvolvidas para a criação de um telegrafo harmônico, que tinha como objetivo transmitir notas musicais através da eletricidade. Watson, durante um teste mal-sucedido, conseguiu transmitir um som inesperado que despertou a curiosidade de Graham Bell. Ao analisar o que havia acontecido, Bell descobriu que o equipamento conseguia reproduzir uma corrente elétrica cuja variação acontecia na mesma intensidade que as correntes de ar variavam de densidade. Surge então, o conceito básico para a criação do dispositivo que seria o maior representante das Telecomunicações. (MARTINS, 2003)

Há uma grande discussão em torno da patente do telefone, a grande maioria considera que foi Graham Bell o dono dessa patente, porém existe quem considere Antonio Meucci o verdadeiro inventor. Os defensores de Meucci alegam que por falta de recursos não foi possível patetear o invento,

Bell inclusive teria usado o laboratório onde se encontrava o material da pesquisa de Meucci. Em 15 de Junho de 2002 o Congresso dos Estados Unidos na resolução 269, reconheceu o Teletrophone (Telefone), nome dado ao invento, como sendo de propriedade intelectual de Antonio Meucci. (MARTINS, 2003)

2.2 CENTRAIS TELEFÔNICAS

A invenção do telefone inicia uma nova fase no mundo das telecomunicações, e o primeiro passo para esse novo mundo será a criação de centrais telefônicas.

As centrais telefônicas surgem para suprir a necessidade de interligar os vários usuários, pois havia uma adoção crescente do uso do telefone. As linhas diretas e dedicadas já não atendiam mais a demanda que apelava por um equipamento que fornecesse a possibilidade de se comunicar não apenas com um terminal, mas que fosse possível escolher um dentre os vários usuários que possuísse um telefone. (MARTINS, 2003)

O principio básico utilizado pelas centrais telefônicas, e que é utilizado atualmente, é o chaveamento/comutação de circuitos. Essa técnica também foi responsável pelo uso do termo Rede Telefônica Pública Comutada (RTPC). A comunicação utilizando essa técnica é feita em três etapas, que são: COLCHER (2005)

- Estabelecimento de circuito: Nessa primeira etapa, os terminais devem alocar um circuito ou conexão dedicada para que a comunicação seja estabelecida;
- Transmissão: Depois de estabelecido o circuito os terminais agora podem enviar e receber informações entre eles; e

- Desconexão do circuito: Geralmente nessa etapa a conexão é desfeita depois de terminada a comunicação entre os terminais. O circuito é totalmente desfeito e poderá ser utilizado para outras transmissões.

Nas primeiras centrais telefônicas a conexão era estabelecida manualmente, ou seja, quando se desejava realizar uma chamada, o terminal origem requisitava uma conexão acionando um operador, que fechava fisicamente o circuito entre os terminais, para a ligação ocorrer, e liberava o canal quando a conversa terminava. Essa técnica ficou conhecida como chaveamento físico manual.

Passados alguns anos, Almom B. Strowger desenvolveu a primeira central telefônica automática. Strowger era dono de uma funerária e via seus negócios serem prejudicados por uma suposta operadora que trabalhava em uma Central Telefônica em Laporte (Indiana, EUA). Essa operadora era casada com o dono de uma outra funerária e o beneficiava desviando as ligações dos clientes de Strowger para a sua funerária. Em decorrência dessa fraude, Strowger via seu negócio sendo ameaçado e sentia a necessidade de criar um mecanismo mais eficiente e que fosse capaz de evitar esse tipo de problema. Sendo assim, no ano de 1891, ele desenvolveu e patenteou a primeira chave seletora automática inspirada no movimento dos braços das telefonistas na mesa telefônica. (FERRARI, 2005).

O mundo das Telecomunicações crescia rapidamente e vários países implementavam o serviço telefônico em seus territórios. Nesse contexto, surgiu um mercado cada vez mais interessante e que incentivava a criação de novas ferramentas que tornassem o serviço mais robusto, rápido e barato.

Com a invenção do transistor, os sistemas computacionais evoluíram e se tornaram mais rápidos e confiáveis. Porém, foi com o desenvolvimento dos circuitos integrados que os computadores tornaram-se figuras importantes no aprovisionamento dos serviços de telecomunicação, permitindo a criação e implementação de novas centrais telefônicas digitais. (COLCHER, 2005).

As novas centrais telefônicas surgem juntamente com a implementação de circuitos que possibilitaram a transmissão de sinais digitais entre as centrais. Passados anos, essas transformações se tornaram predominantes, apenas as linhas dos assinantes, conhecidas como última milha, é que ainda continuam analógicas.

A digitalização dos sinais juntamente com a utilização dos sistemas computacionais para o aprovisionamento de serviços de telecomunicações foi o primeiro passo para convergência dessas tecnologias.

2.3 MULTIPLEXAÇÃO DOS TELEFONES

Os sistemas telefônicos inicialmente eram completamente analógicos, ou seja, o sinal gerado pelos terminais telefônicos, as centrais, a transmissão e a chegada eram todos analógicos. O tipo de multiplexação utilizado era o FDM (Multiplexação por Divisão de Freqüência). Com o passar dos anos houve mudanças em relação ao sinal que não era mais totalmente analógico, e sim apenas o envio, a recepção e as centrais, passando apenas a transmissão a ser digital. A digitalização do sinal possibilitou a utilização da TDM (Multiplexação por Divisão de Tempo) e tornou-se obrigatório a utilização de um tipo de codificação para transformar o sinal de analógico para digital (A/D) e de digital para analógico (D/A). (COLCHER, 2005).

Predomina-se atualmente, no sistema telefônico, o sinal analógico proveniente dos terminais. Existem alguns países que utilizam o RDSI – Rede Digital de Serviços Integrados - que usam o sinal completamente digital. Nesse serviço o objetivo é a transmissão de dados, voz e imagens numa única rede. A Figura 2.1 mostra a penetração dos sinais digitais na rede telefônica, iniciando com todos os enlaces analógicos e chegando até a configuração das redes RDSI. (COLCHER, 2005).

A multiplexação é um processo muito importante para o sucesso dos sistemas telefônicos, pois através dela é possível trafegar vários sinais combinados em um mesmo meio de transmissão. Os dois tipos mais utilizados de multiplexação são: Multiplexação por Divisão de Freqüência (FDM) e Multiplexação por Divisão de Tempo (TDM). (COLCHER, 2005)

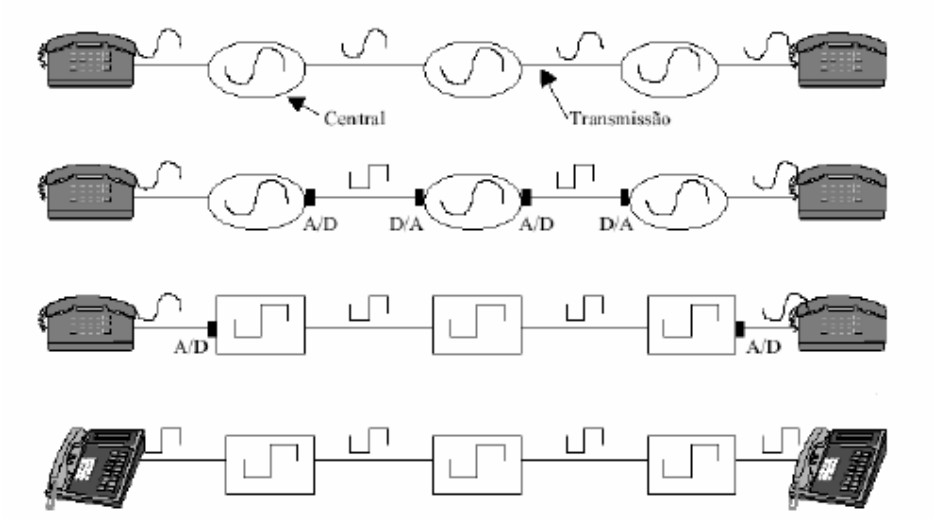

Figura 2.1 - Penetração da tecnologia digital nas redes telefônicas

Na multiplexação por divisão de freqüência, cada usuário possui sua própria largura de banda que é adquirida pela divisão do espectro de freqüência em canais lógicos. A interferência nesse processo é evitada porque a modulação de cada canal analógico é feita em freqüências diferentes entre si. A Figura 2.2 mostra a multiplexação de vários canais

telefônicos, onde pode se perceber que cada sinal possui sua própria largura de banda deslocada para que possa ser transmitida em um mesmo meio físico. (COLCHER, 2005)

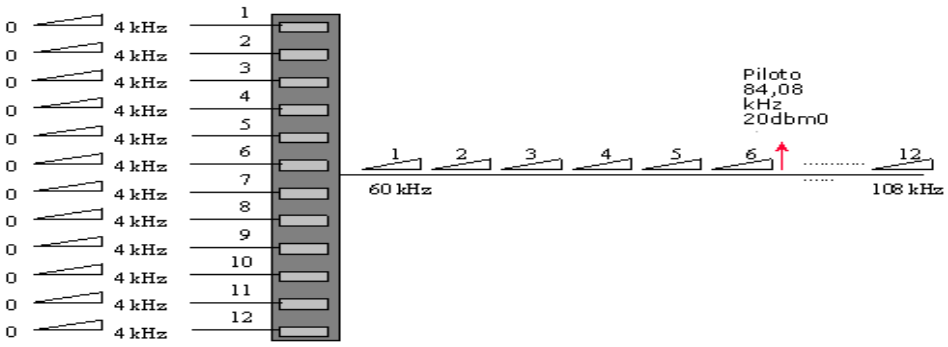

Figura 2.2 - Multiplexação por Divisão de Freqüência.

Na multiplexação por divisão tempo cada canal utiliza um intervalo de tempo específico para transmissão (time slot), enquanto os demais canais aguardam a sua vez para poderem transmitir. Nesse tipo de multiplexação trabalha-se com circuitos eletrônicos e sinais digitais. Na figura 2.3 ilustra o modo como funciona a multiplexação por Divisão de Tempo. Vários canais chegam a um determinado multiplexador que aloca o meio de transmissão por um determinado intervalo de tempo e esse transmite até que chegue a vez de outro canal. (COLCHER, 2005)

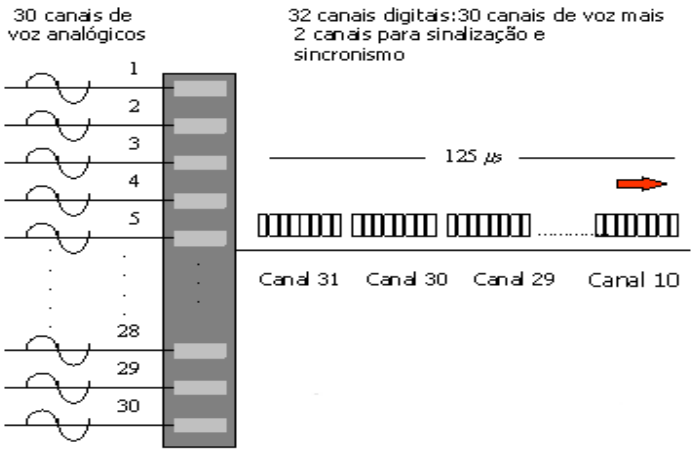

Figura 2.3 - Multiplexação por Divisão de Tempo

2.4 PRINCÍPIOS DE SINALIZAÇÃO

A sinalização telefônica é um mecanismo criado para prover as centrais telefônicas às informações necessárias para que seja possível estabelecer uma ligação. Essas informações têm como conteúdo os dados dos terminais que desejam estabelecer uma comunicação, o estado que se encontra o sistema e boletins e alertas referentes aos procedimentos que estão sendo realizados. A importância dos mecanismos de sinalização acompanha a evolução dos meios de comunicação e ganha força com as melhorias feitas para proporcionar aos sistemas telefônicos certa independência no funcionamento e manutenção dos serviços prestados. (FERRARI, 2005).

De forma a ficar mais clara o que seria o mecanismo de sinalização, um dos autores de minhas bibliografias compara as funções realizadas pelos mecanismos de sinalização com o sistema nervoso central de um ser humano. Nesse contexto, o sistema nervoso apenas coordena as atividades do organismo sem assumi-las ou executá-las, ou seja, faz o monitoramento, e o envio de informações para que as atividades sejam executadas da melhor forma possível. COLCHER (2005)

Há três áreas funcionais em que podem ser classificados os mecanismos de sinalização. Essas áreas são: COLCHER (2005)

- Sinalização de Supervisão: É o conjunto de sinais destinados a informar as condições e o estado das linhas para que uma chamada possa ser realizada. A sinalização de supervisão é responsável por informar a disponibilidade dos terminais e o início e término de uma chamada para tarifação.
- Sinalização de indicação de usuário: É o conjunto de sinais

responsáveis por informar o estado em que se encontra o sistema ao assinante que está tentando realizar uma chamada. Esse tipo de sinalização informa ao usuário se o sistema está preparado para realizar uma chamada, se o terminal do número discado está ocupado e se o número discado é inexistente.

- Sinalização de numeração: É responsável por interpretar os números discados por um terminal e encaminhar esse pedido de estabelecimento de conexão até o terminal solicitado.

2.5 CODIFICAÇÃO DA VOZ

A digitalização dos meios de comunicação teve um papel muito importante para a nova tendência de mercado que busca a convergência dos serviços de voz e de dados. A voz, que é um sinal analógico, pode agora ser transmitida através de uma rede de computadores, cujo meio de transmissão é digital. Para isso é necessário que a voz passe por um processo de digitalização que é realizado por equipamentos conhecidos como codificadores de voz. (COLCHER, 2005)

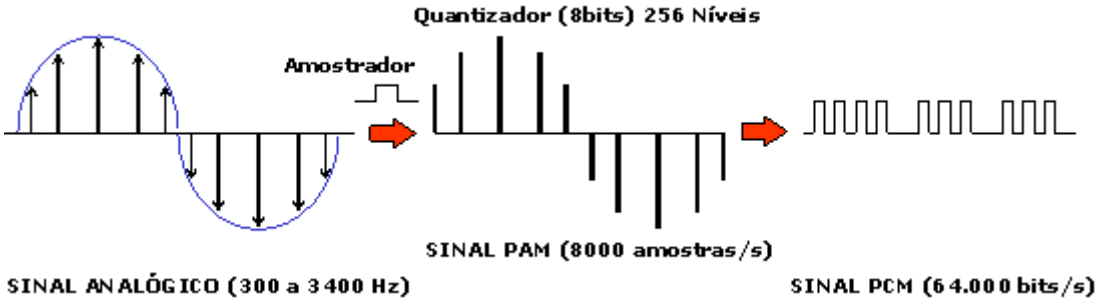

Figura 2.4 - Digitalização da Voz.

A digitalização só foi possível com a introdução dos sistemas de transmissão PCM (Pulse Code Modulation), que transformam um sinal analógico em uma série de pulsos binários que podem ser manipulados.

Esse sistema é composto por de várias etapas (Figura 2.4) nas quais o sinal é tratado para que seja possível sua transmissão.

A primeira etapa é conhecida como Teoria de Amostragem, que consiste em substituir o sinal analógico por uma sucessão de amostra de curta duração em intervalos de tempos regulares. Esse Teorema de Amostragem também é conhecido como Teorema de Nyquist. (FERRARI, 2005).

De acordo com o teorema de Nyquist, se um sinal tem uma freqüência máxima $f_{max,}$ a taxa de amostragem deverá ter no mínimo duas vezes esse valor para que seja possível reconstruir o sinal sem perdas. (FERRARI, 2005).

A voz humana utiliza uma faixa de freqüência entre 0 a 4000 Hz, segundo Nyquist a taxa de amostragem deverá então ser de 8000 amostras por segundo, ou seja, o intervalo de tempo de uma amostra e outra do mesmo sinal são de 125 μs. Na Figura 2.5 é apresentado um sinal que passou por um processo de amostragem. (FERRARI, 2005)

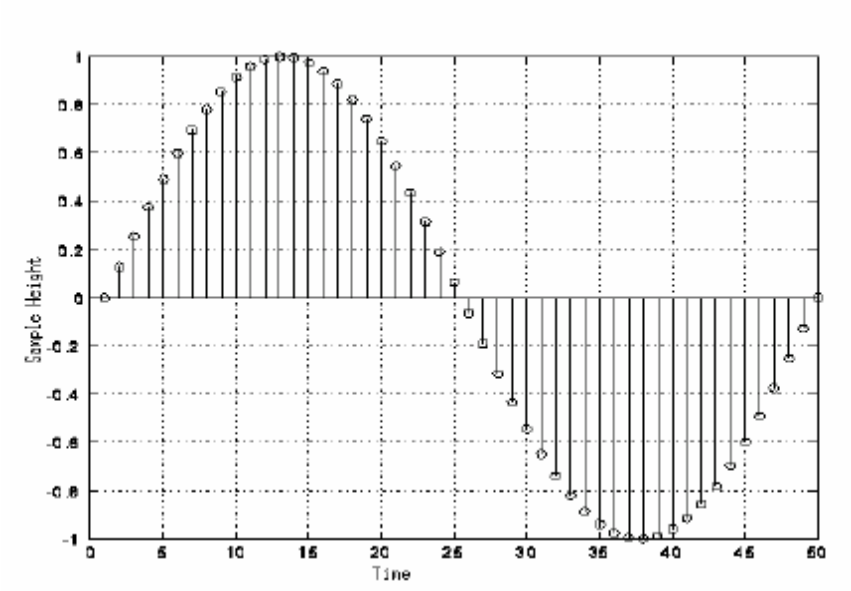

Figura 2.5 - Amostra de um sinal analógico.

Depois de feitas as amostras a próxima etapa a ser realizada é a Quantização, que consiste em aproximar as amplitudes das amostras para que assumam valores discretos. Cada amostra é representada por 1 byte ou 8 bits, o que torna possível representar a amplitude do sinal analógico em 256 níveis de tensão elétrica. Para que seja possível representar essas 8000 amostras, com cada uma possuindo 8 bits, é necessário ter uma velocidade de transmissão de 64 kbps. (FERRARI, 2005).

Após os sinais serem amostrados, quantizados, a próxima etapa é codificar o sinal. Nessa etapa as amplitudes adquiridas na quantização são convertidas em uma combinação de bits zero e um. E finalmente, depois codificados é necessário comprimir os dados e para isso é utilizado um algoritmo de compressão. Essa compressão é feita em tempo real e é importante para que se tenha uma economia na largura de banda.

2.6 PBX

A sigla PBX significa *Private Branch Exchange* que no português significa Central Particular Tributária da Central Pública. O PBX foi criado principalmente para atender ao mercado empresarial. Os primeiros PBX's surgiram na década de 80 e eram manuais. Nessa versão manual, recebeu o nome de PMBX (*Private Manual Branch Exchange*) ou simplesmente PBX (*Private Branch Exchange*) como é mais conhecido. (FERRARI, 2005).

Os PMBX's funcionam utilizando uma telefonista que tem a função de completar as chamadas entre os ramais. Nesse sistema a telefonista recebe a chamada com o número do ramal ou o nome da pessoa e a seção de quem se deseja comunicar e então é completada a ligação. Quando se deseja realizar uma chamada externa e o ramal possui um discador é

necessário fazer um pedido à telefonista e essa disponibiliza uma linha através da qual será possível realizar a chamada. (FERRARI, 2005).

Para tornar o PMBX mais eficiente foi implementado um sistema que realiza as chamadas entre os ramais. Esse sistema fazia a comutação das chamadas automaticamente e foi chamado de PAX (*Private Automatic Exchange*). Com o PAX e o PMBX os usuários eram obrigados a ter dois aparelhos em sua mesa, um para ligações internas e outro para ligações externas. (FERRARI, 2005)

Apesar de tornar o sistema mais eficiente, o número de ramais poderia crescer de tal forma que as telefonistas não seriam capazes de atender a demanda de ligações e ainda existia a necessidade dos usuários possuírem dois aparelhos. Esses problemas foram superados mais tarde com o surgimento do PABX (*Private Automatic Branch Exchange*) que é a fusão do PMBX com o PAX.

O PABX elimina a necessidade de utilização de telefonistas, caso a empresa deseje. Embora, algumas empresas como, por exemplo, os hotéis, que não dispõem de equipamentos de bilhetagem das ligações efetuadas pelos hóspedes, utilizam-se desse artifício para fazer as cobranças. Outras facilidades podem ser adquiridas com a utilização de um PABX como, por exemplo, transferência, captura de ligações, chamada em espera e conferência. (FERRARI, 2005).

Como desvantagem o PABX é em geral uma tecnologia proprietária que fica limitada ao fabricante. Qualquer adição de funcionalidade, modificação e manutenção são de exclusividade dos fabricantes, não podendo ser contratada outra empresa para fazer esse tipo de serviço.

Antes de começar a falar sobre a tecnologia VoIP – Voz sobre o Protocolo IP, vale lembrar que a primeira tentativa de se convergir os serviços em uma única rede foi a implementação das Redes Digitais de Serviços Integrados – RDSI. Esse serviço tem como principio a digitalização de toda a rede de serviços de telefonia convencional, que como já foi dito anteriormente, possui a linha do assinante ainda sendo analógica.

A RDSI tem como principal objetivo fornecer serviços de dados, vídeo e voz em uma única infra-estrutura, porém essa rede não obteve o seu sucesso esperado, principalmente devido ao enorme custo em digitalizar as linhas dos assinantes. Outro, não menos importante fator, seria o surgimento do ADSL (Assymmetric Digital Subscriber Line ou Linha Digital Assimétrica para Assinantes), que fornecia os serviços de dados a custos mais baixos.

Após essa tentativa, surge mais uma vez a vontade de se convergir os vários serviços em uma única rede, porm dessa vez, essa vontade chega de forma mais encorpada principalmente com o sucesso da Internet. No próximo capitulo serão abordados com maiores detalhes essa tecnologia que se pretende convergir os serviços de voz e dados em uma única rede chamado VoIP.

CAPÍTULO 3. VOZ SOBRE O PROTOCOLO INTERNET

3.1 INTRODUÇÃO

Atualmente, uma grande revolução vem acontecendo nas telecomunicações, resultante do incrível crescimento das redes baseadas em pacotes, especialmente da Internet. Essa revolução está unificando os mundos de dados e telecomunicações em uma só rede convergente.

Essa proposta de unificar as redes tornou-se tão interessante e importante dentro do mercado, que até mesmo as operadoras telefônicas tradicionais estão adaptando-se à nova realidade tecnológica. Essas adaptações estão ocorrendo principalmente através do desenvolvimento de soluções que visam otimizar o uso de suas infra-estruturas baseadas na comutação de circuitos atualizando-as para a comutação de pacotes.

A busca dessa convergência para as empresas telefônicas não está apenas relacionada à redução de custo das chamadas, afinal não seria necessário investir tanto na adaptação de uma rede que possui os mesmo fins. A convergência traz, entre outros benefícios, o aumento da receita proporcionada pela oferta de novas funcionalidades que tornam a aquisição dos serviços oferecidos mais atrativa para o usuário.

O VoIP insere-se nesse cenário como uma das formas de manifestação de convergência de serviços de voz com o serviço de dados que está ganhando cada vez mais força no mercado mundial. O VoIP significa Voz sobre Protocolo Internet e permite a digitalização, codificação, empacotamento e envio de sinais de voz em tempo real utilizando o conjunto de protocolos das redes TCP/IP

A tecnologia VoIP está possibilitando o ingresso de novos competidores no mercado de voz que não possuem qualquer facilidade de rede e são bem diferentes das operadoras de telecomunicações tradicionais, como por exemplo, Vonage no Estados Unidos e Skype não somente na Europa mas também em diversos países pelo mundo.

Estas empresas estão abocanhando uma pequena parcela do mercado de telecomunicação, conseqüências da prática de preços baixos, pois os mesmos não possuem nenhuma infra-estrutura de rede ou plataformas operacionais como as operadoras tradicionais de telecomunicações. Essa prática é viável desde que seus serviços de voz sejam oferecidos sobre a plataforma pública da Internet.

3.2 CODIFICAÇÃO DO SINAL

Nos sistemas telefônicos tradicionais o sinal analógico utiliza uma banda de 4 kHz, e é digitalizado com uma taxa de amostragem de 8 kHz para que a sua recuperação possa ser feita com qualidade. Como cada amostra é representada por um byte, cada canal de voz necessita de uma banda de 64 kbit/s.

Tabela 3.1 - Padrões de codificação ITU-T

Padrão ITU	Descrição	Taxa (Kbps)	Retardo (ms)	Complexidade
G. 711	PCM	64	0,75	----------
G. 726	ADPCM	32	1	Baixa
G. 728	LD-CELP	16	3-5	Baixa
G. 729	CS-ACELP	8	10	Média
G. 723.1	ML-MLQ	6.3	30	Alta
G. 723.1	ACELP	5.3	30	Alta

Já nos sistema de transmissão de voz sobre o protocolo IP, a demanda por banda é crítica, tornando-se necessário utilizar também

algoritmos de compressão do sinal de voz. Esses algoritmos têm um papel muito importante na economia de banda e por isso incentivam o desenvolvimento de tecnologias mais complexas para a digitalização e compressão de voz. Essas evoluções foram registradas através de recomendações do ITU-T.

3.3 PROTOCOLO TCP/IP

As redes IP constituem o grande universo de milhões de computadores interligados em todo o mundo com uma expectativa constante de crescimento. Estas redes se desenvolveram com facilidade devido ao crescimento da Internet e à homologação do TCP/IP como protocolo de suporte às aplicações em redes.

Como os computadores de uma rede devem adotar as mesmas regras para o envio e o recebimento de informações, para que seja possível se comunicarem, o TCP/IP foi se tornando o protocolo padrão devido ao enorme crescimento da Internet. Pode-se até implementar redes utilizando outros protocolos de comunicação, mas caso haja interesse de se utilizar a Internet, deve-se adaptar os equipamentos de rede a fim de operar em TCP/IP.

O sucesso do protocolo TCP/IP teve como principal fator a flexibilidade para poder se conectar a diversas redes heterogêneas comutadas por pacote. As redes TCP/IP foram desenvolvidas com o discurso de que poderiam ser usadas sobre qualquer tipo de meio físico, sendo estes de qualquer tecnologia, apresentando ou não confiabilidade, com alto ou baixo desempenho.

Parece estranho e confuso o TCP/IP ter uma arquitetura como aparece em vários documentos, pois o TCP e o IP são protocolos que são

vistos na própria Arquitetura TCP/IP, respectivamente na camada transporte e na camada de Rede, como pode ser visto na figura 3.2, mas por sua grande aceitação e por serem os dois protocolos mais importantes utilizados na Internet, podem ser utilizados para denominar a Arquitetura da Internet.

Figura 3.1 - Arquitetura TCP/IP

3.3.1 PROTOCOLO IP

O IP significa "*Internet Protocol*" que traduzindo para o português significa Protocolo da Internet. O IP é um protocolo de camada de inter-rede descrito no RFC (*Request for Comments*) 791 da IETF (*Internet Engineering Task Force*), que foi publicado em Setembro de 1981.

O IP é um protocolo não orientado à conexão, cuja função é transferir blocos de dados denominados datagramas ou pacotes da origem até o destino. O IP oferece um serviço de datagramas não confiável também chamado de melhor esforço, não garantindo a entrega dos datagramas. Se a aplicação precisar de confiabilidade, esta é adicionada na camada de transporte.

3.3.2 PROTOCOLO TCP

O TCP (*Transmission Control Protocol*) é um padrão definido na RFC 793, que fornece um serviço de entrega de pacotes confiável e orientado à conexão, o que significa que neste nível vão ser solucionados todos os problemas de erros que não forem solucionados no nível IP, dado que esse último é um protocolo sem conexão. Ser orientado à conexão significa que todos os aplicativos baseados em TCP, como protocolo de transporte, antes de iniciar a troca de dados, precisam estabelecer uma conexão.

O TCP é responsável pela recuperação de dados corrompidos, perdidos, duplicados ou entregues fora de ordem, solucionando o problema do protocolo IP. O mecanismo que é empregado conceitualmente funciona utilizando um conjunto de dados (ACK - *Acknowledgement*) que informa qual pacote que o receptor espera receber. De posse dessa informação sabe-se que, até o pacote que o receptor deseja receber, todos os outros foram recebidos com sucesso. Com esse mecanismo, apenas pacotes com problemas terão que ser enviados, o que reduz o tráfego na rede e agiliza o envio dos pacotes.

Além disso, o protocolo TCP tem a capacidade de transmitir dados nas duas direções entre seus usuários, e pode ainda decidir quando começar e quando parar de transmitir esses dados. Contudo, caso o usuário utilize a função *push*, todos os dados que estão no *buffer* aguardando transmissão serão transmitidos.

3.3.3 PROTOCOLO UDP

O UDP (*User Datagram Protocol*) é um padrão TCP/IP e está definido pela RFC 768. O UDP é usado por alguns programas em vez do TCP para o

transporte rápido de dados. Porém o UDP é pouco confiável, sendo um protocolo não orientado a conexão e que não fornece garantia de entrega e nem verificação de dados.

Para programas que precisam de uma maior velocidade de processamento, o UDP é a escolha acertada como protocolo da camada de transporte. O UDP não faz as verificações feitas pelo protocolo TCP, e por isso não perde tempo no envio dos pacotes.

Para poder trabalhar com mais de um serviço, tanto o protocolo UDP como o protocolo TCP trabalham com o conceito de portas, embora tecnicamente existam diferenças na maneira como as portas são utilizadas em cada protocolo, a idéia é a mesma. A utilização das portas permite que vários programas estejam em funcionamento, ao mesmo tempo, no mesmo computador, trocando informações com um ou mais serviços.

Algumas aplicações possuem portas definidas, como mostra a tabela 3.2, enquanto outras aplicações, o TCP e o UDP atribuem uma porta. Todos os números de porta de servidor UDP e TCP menores que 1.024 e alguns números mais altos são reservados e registrados pela IANA (*Internet Assigned Numbers Authority)*.

Tabela 3.2 - Protocolos de Aplicação disponível na Arquitetura TCP/IP

APLICAÇÂO	PORTA	DESCRIÇÃO
FTP – File Transport Protocol	**21**	**File Transfer [Control]**
SSH – Secure Shell	**22**	**SSH Remote Login Protocol**
TELNET	**23**	**Telnet**
SMTP – Simple Mail Transport Protocol	**25**	**Simple Mail Transfer**
DOMAIN	**53**	**Domain Name Server**
HTTP – Hiper-Text Transfer Protocol	**80**	**World Wide Web HTTP**

3.4 QUALIDADE DE SERVIÇO

Para poder prover um serviço utilizando a tecnologia VoIP com qualidade, deve-se ter uma atenção especial no transporte da voz, pois nesse tipo de serviço utiliza-se a comutação de pacotes ao contrário dos sistemas telefônicos convencionais onde é utilizada a comutação de circuito. Na comutação de pacotes não há o estabelecimento de um caminho dedicado entre as estações e as mensagens, caso sejam maiores que um valor determinado, devem ser quebrado em unidades menores chamado de pacotes.

Mesmo com essas mudanças, o maior desafio é provar que o serviço de voz sobre IP é capaz de fornecer um nível de qualidade parecido com os sistemas tradicionais de telefonia para que convença os usuários, que já estão acostumados com os telefones, que a tecnologia VoIP pode ser implementada sem maiores perdas.

Dessa forma, surge o conceito de QoS (*Quality of Service*) que não é uma tecnologia e nem mesmo um protocolo. A QoS funciona utilizando parâmetros de eficiência que são pré-determinados com o objetivo de garantir que esses valores acordados sejam cumpridos para que o serviço prestado seja provido de forma satisfatória.

3.4.1 PARÂMETROS DE QUALIDADE DE SERVIÇO

A obtenção de QoS pode ser constata em um serviço se todos os parâmetros que forem pré-definidos, de forma que o mínimo seja alcançado.

3.4.1.1 TAXA DE TRANSMISSÃO

A taxa de transmissão, ou vazão, corresponde à quantidade de informação que pode ser transferido de um ponto a outro de uma rede em um determinado período de tempo. A transmissão dessas informações esta limitada pelo tamanho da largura de banda. Largura de banda é quantidade de informações que podem ser enviadas através de uma conexão.

No caso da tecnologia VoIP, a taxa de transmissão, dependendo do número de usuários que estão utilizando a banda, não é tão exigida em comparação com outros tipos de serviço, como por exemplo, serviços de vídeo.

3.4.1.2 ATRASO (DELAY)

É o tempo necessário que um pacote leva para sair da origem e ser transmitido até o seu destino. Na tecnologia VoIP, a demora na chegada dos pacotes pode ocasionar um certo desconforto, semelhante ao encontrado nas comunicações utilizando satélites.

Os principais fatores responsáveis pelos atrasos na transmissão de um pacote são:

- Atraso de propagação: Corresponde ao tempo necessário para propagação dos bits de um pacote através de um meio físico até o seu destino. Esse tempo depende das características do meio de propagação dos bits e da distância percorrida;
- Atraso de codificação e decodificação: Em aplicações de voz, normalmente os sinais são codificados para que seja possível sua transmissão através das redes de comutação de pacotes. Esse

processo de codificação e decodificação leva certo tempo para que seja processado, ocasionando em atraso;

- Atraso de empacotamento e desempacotamento: É o tempo necessário para que os dados sejam empacotados a fim de serem transmitidos na rede TCP/IP

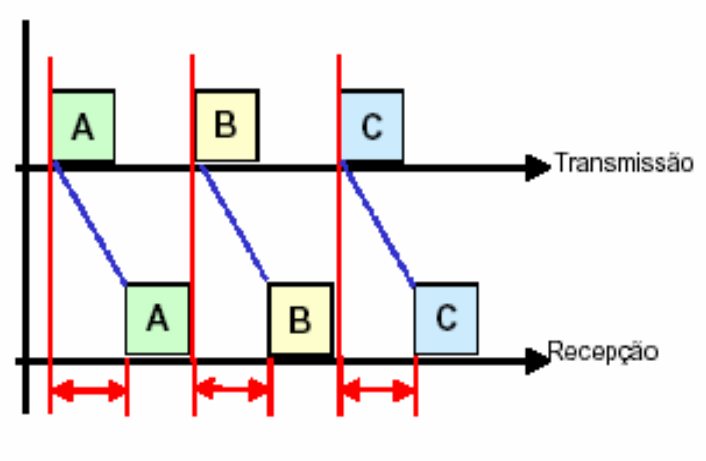

Figura 3.2 - Atraso (Delay) de pacote.

3.4.1.3 JITTER

Jitter é a variação de atrasos em uma transmissão fim-a-fim. Pacotes podem sair da origem com um atraso constante entre os mesmos e podem chegar ao destino com atrasos variáveis. Embora os vários pacotes de um arquivo ou de uma aplicação em tempo real (telefonia, videoconferência) possuam a mesma origem e os mesmo destinos eles podem sofrer atrasos diferentes durante o percurso.

O que causa o *jitter* são os atrasos de processamento e os atrasos decorrentes das filas enfrentadas pelos pacotes nos vários roteadores do percurso. Esses atrasos trazem como conseqüência a criação de um buffer, cujo seu tamanho dependerá do *jitter* gerado, ocasionando mais e mais atraso.

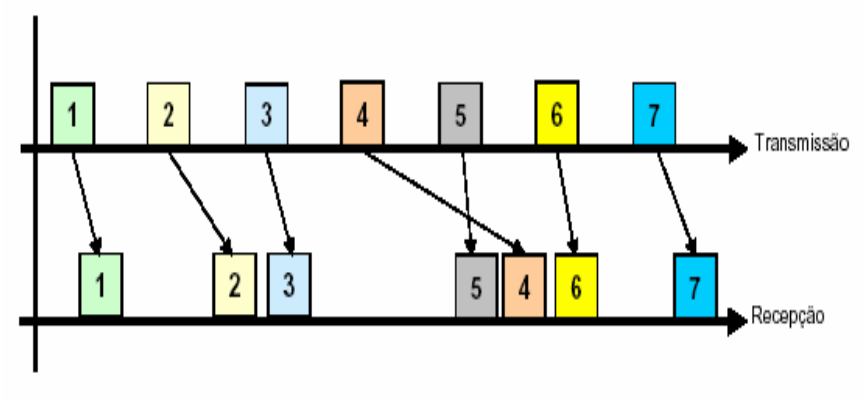

Figura 3.3 – Jitter.

3.4.1.4 PERDAS DE PACOTE

A perda de pacotes é um parâmetro que também merece atenção, pois traz como conseqüência a perda de qualidade na conversação entre os usuários. A conversa se torna “quebrada”, pois os pacotes que se perderam traziam alguns trechos da conversa. Abaixo serão citadas algumas causas da perda de pacotes.

- Erros: Os pacotes podem ser enviados ou chegarem ao seu destino com algum problema, ou podem ser corrompidos durante a sua transmissão;
- Problemas no enlace: Pode ocorrer algum problema nos equipamentos físicos ou nos meios de transmissões entre os terminais;
- Atraso excessivo: Os roteadores podem descartar pacotes caso o tempo de transmissão seja superior ao tolerável;
- Congestionamento: Os roteadores podem descartar pacotes caso o buffer atinja o máximo permitido;

Tabela 3.3 - Sensibilidade dos serviços de acordo com os parâmetros QoS

Aplicação	Largura de Banda	Atraso	Jitter	Perda de Pacote
VoIP	**Baixa**	**Alta**	**Alta**	**Média**
Videoconferência	**Alta**	**Alta**	**Alta**	**Média**
E-mail	**Baixa**	**Baixa**	**Baixa**	**Alta**
Transferência de Arquivos	**Baixa**	**Baixa**	**Baixa**	**Alta**

3.4.2 ARQUITETURAS UTILIZADAS PARA PROVER QOS

Com a finalidade de fornecer garantias de qualidade de serviço a IETF, por volta dos anos 90, introduziu duas arquiteturas: IntServ (*Integrated Services*) e a arquitetura DiffServ (*Differentiated Services*) que serão abordadas nos próximos tópicos.

3.4.2.1 ARQUITETURA INTSERV

Essa arquitetura, também conhecida pelo termo serviços integrados, foi desenvolvida com o objetivo de alocar recursos para que seja possível a transmissão de vídeo e som sem interrupção. Utiliza-se nessa arquitetura a classificação dos serviços em categorias e de acordo com as mesmas podem-se fornecer diferentes graus de comprometimento de recursos. O IntServ é composto basicamente por quatro componentes que serão relatados abaixo.

- Controle de admissão: Determina a possibilidade de um fluxo ser aceito ou não. Para que seja aceito a rede deve poder oferecer, pelo menos, o mínimo de recursos e que esses não interfiram nos fluxos já aceitos anteriormente;

- Escalonador de pacotes: Tem a função de gerenciar os buffers das filas de saída dos roteadores e estações com o objetivo de transmitir os pacotes de modo que atenda os parâmetros de QoS especificados;
- Classificador: Possui a responsabilidade de encaminhar os pacotes aos buffers de saída de acordo com a categoria que o fluxo pertence;
- Policiamento: Verifica se os pacotes foram classificados de acordo com os parâmetros de tráfego e QoS negociados para o fluxo;

Além dos componentes citados acima, a arquitetura IntServ trabalha utilizando o protocolo RSVP (*Resource Reservation Protocol*). Sua utilização não é obrigatória, porém constitui-se na opção mais conveniente.

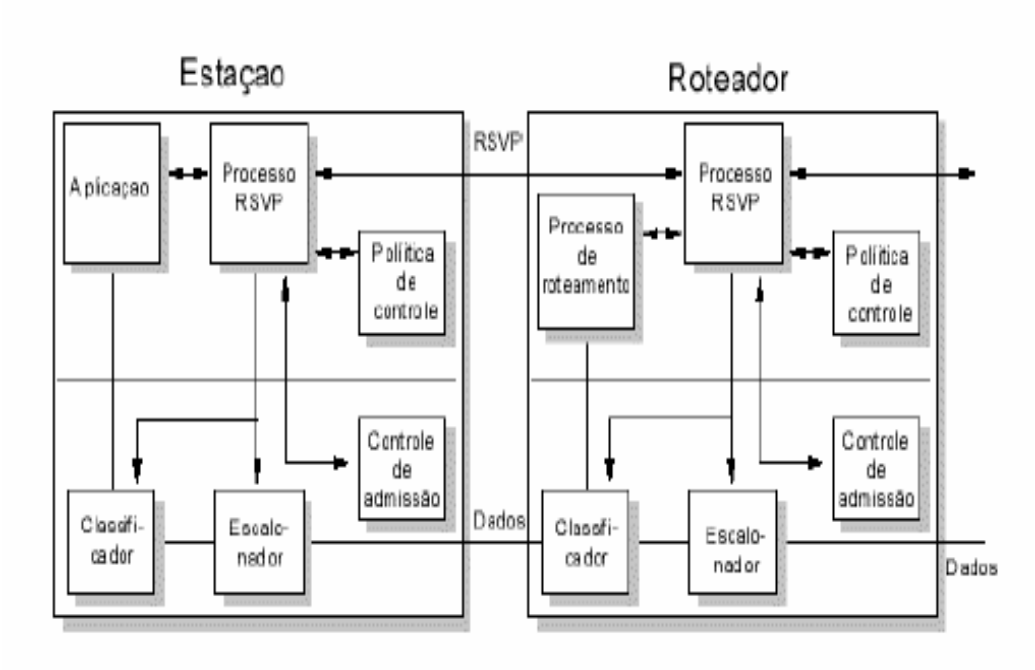

Figura 3.4 - Modelo de referência para implementação da arquitetura IntServ.

O protocolo RSVP é especificado na RFC 2205 e tem a finalidade de reservar e manter, durante uma sessão, parâmetros que satisfaçam uma

determinada qualidade de serviço. Entende-se como sessão um fluxo de dados com uma origem e um destino, onde o protocolo RSVP cria um caminho entre eles, perguntando a todos os elementos que intermedeiam esse percurso a possibilidade de suportar a qualidade desejada e, em caso de uma resposta positiva, reservam-se às necessidades solicitadas pelo serviço. Para tanto, esses elementos intermediários devem suportar o protocolo RSVP.

O protocolo RSVP não é um protocolo de roteamento e nem realiza o transporte de dados. Ele é um protocolo de controle que faz as consultas à base de dados dos roteadores para obter um caminho entre a origem e o destino. Para realizar essas consultas são utilizados métodos de trocas de mensagens, sendo a RESV e PATH as principais no protocolo RSVP.

A mensagem PATH é enviada pela origem e se propaga através da rede, seguindo uma rota informada pelos mecanismos de encaminhamento até o destino. Os equipamentos intermediários, quando recebem a mensagem PATH, criam um estado conhecido como PATH *state*, onde serão armazenados os parâmetros requisitados.

Depois que a mensagem chega ao destino, uma nova mensagem é enviada seguindo o caminho inverso. Essa mensagem, chamada RESV, conterá o QoS requisitado e tramitará até a origem de forma que todos os equipamentos intermediários possam efetuar a reserva de recursos especificada na mensagem PATH. A Figura 3.5 ilustra um exemplo desse processo de negociação.

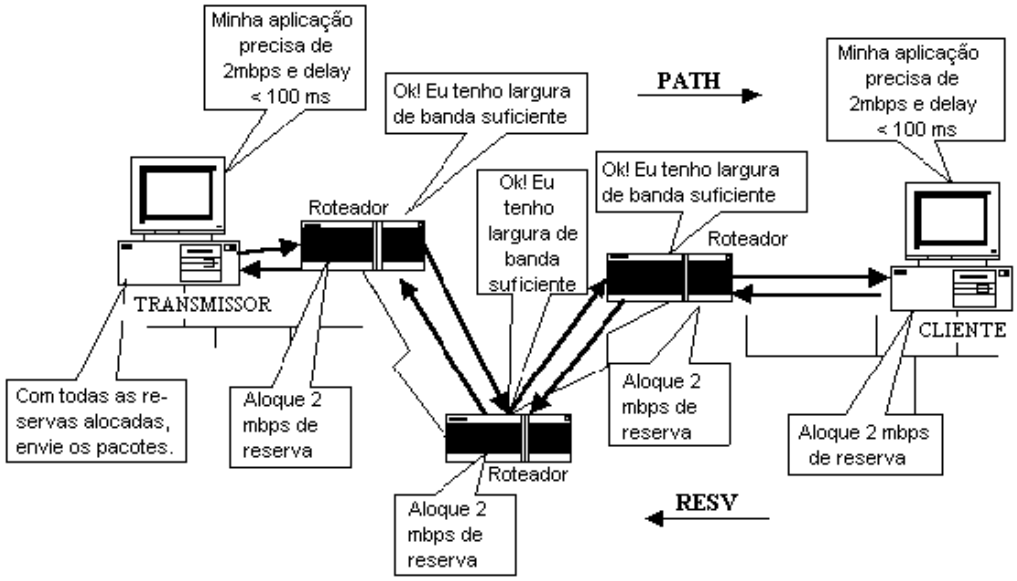

Figura 3.5 - Exemplo de um modelo RSVP

3.4.2.2 ARQUITETURA DIFFSERV

A Arquitetura Diffserv, também conhecida como serviço diferenciado, trabalha com a idéia de diferenciar os diversos serviços e classificá-los em classes de acordo com o QoS desejado. Nessa arquitetura o tratamento passa a ser baseado em agrupamento de fluxos e não como no serviço integrado, onde o tratamento dos fluxos é individual.

A classificação dos pacotes é feita preenchendo o campo DS (Differentiated Services) que se encontra no cabeçalho do protocolo IP. Esse campo na verdade é o antigo ToS (*Type of service*) que até então não tinha uma funcionalidade e, por isso, foi aproveitado pelo serviço diferenciado para informar o grau de prioridade com que um pacote será encaminhado através dos nós da rede.

Porém o campo ToS teve que sofrer algumas adaptações, mesmo porque a sua criação não teve como objetivo as finalidades aqui descritas. A

figura 3.6 mostra essas modificações, onde os seis primeiros bits do DS são chamados de DSCP (*Differentiated Services Code Point*), sendo utilizados para determinar a qual classe o pacote pertence. Já os dois últimos bits são chamados de CU (*Correntlly Unused*) e ainda não possuem uma funcionalidade.

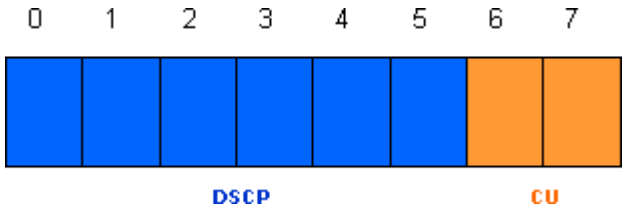

Figura 3.6 - Redefinição do campo ToS do datagrama IPv4

3.4.3 RTP/RTCP

O RTP (*Real-time Transport Protocol*) é um protocolo que oferece funções de transporte de rede fim-a-fim utilizado para serviços que exigem o transporte de pacotes em tempo real. Esse protocolo não tem a finalidade de reservar recursos e também não garante qualidade de serviço (QoS), mas auxilia para que os protocolos das camadas inferiores possam oferecer esse tipo de transporte.

O protocolo RTP é responsável pela transmissão dos dados, e como esse não possui controle de fluxo ou capacidade de verificação de erros, essa responsabilidade fica por conta do protocolo RTCP (*Real-time Control Protocol*).

O RTP e RTCP podem ser usados acima do protocolo UDP, uma vez que os dados multimídia precisam ser transportados com uma latência muito baixa. O RTP costuma ser associado a uma porta do UDP de número par e o RTCP é associado à próxima porta ímpar do UDP.

O RTP é protocolo que tem a responsabilidade de definir como deve ser feita a fragmentação do fluxo de dados, adicionando o cada fragmento informações de seqüência e de tempo de entrega. Essas informações são encontradas nos 12 bytes que constitui o pacote RTP.

No pacote RTP encontra-se o campo “V” (*Version*) que indica a versão do protocolo utilizado, atualmente estamos na versão 2. Encontramos também o campo “P” (*Padding*) que indica a presença ou não de preenchimento das posições finais do pacote com a finalidade de alinhamento, o campo “X ” (*Extension*) indicando ou não a presença de extensão do cabeçalho, o campo “CC” (*CSRC Counter*) que indica o numero de CSRC após o cabeçalho fixo e o campo “M” (*Market Bit*) que delimita um conjunto de dados.

Além dos campos citados acima, existe o campo “PT” (*Payload Type*) que possui sete bits e que tem a função de definir que tipo de dados há no pacote e como deve ser a interpretação por parte da aplicação. Existe também o campo Numeração Seqüenciada que possui 16 bits e serve para ordenar os pacotes de uma comunicação, sendo que o número do primeiro pacote é escolhido aleatoriamente e os outros pacotes recebem o número na seqüência.

E para finalizar os campos com 32 bits, temos o *Time Stamping* que ilustra o momento em que o primeiro pacote de dados foi gerado, esse dado é utilizado pelo destino para sincronismo e cálculo do *jitter*, o campo SSRC (*Synchronization Source*), que tem a finalidade de trazer informações necessárias para possibilitar o destino agrupar os pacotes com o mesmo SSRC, e o campo CSRC (*Contributing Source*) que identifica as fonte que contribuíram para a formação dos dados contidos no pacote. Abaixo a Figura 3.7 ilustra o pacote RTP.

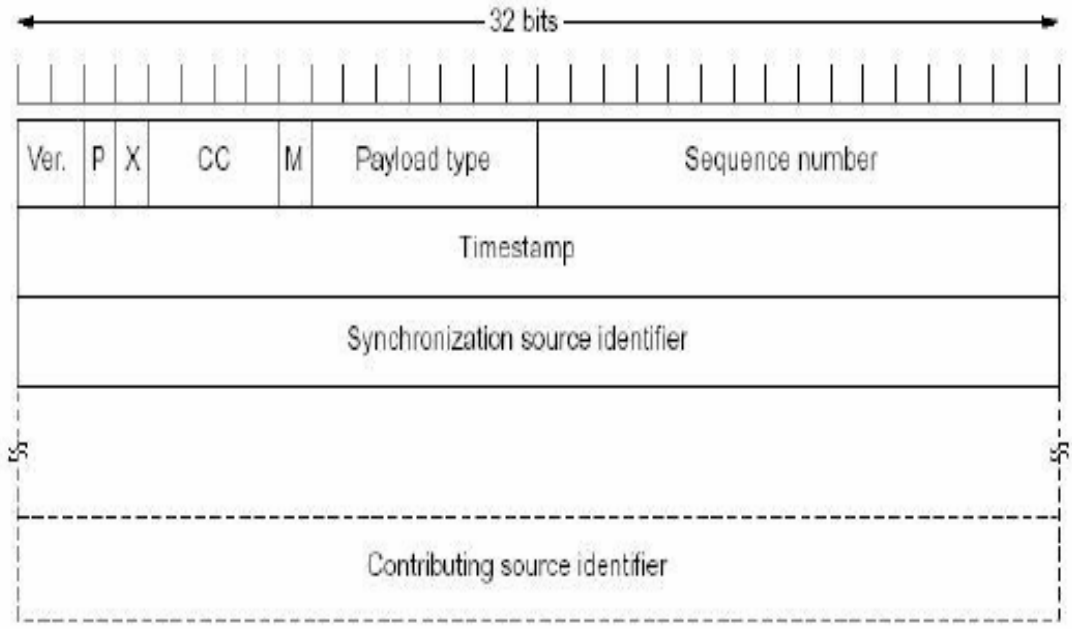

Figura 3.7 - Cabeçalho RTP

Já o protocolo RTCP funciona em paralelo com o protocolo RTP e tem como objetivo possibilitar certo controle e identificação dos participantes da comunicação. Sua ação se baseia na transmissão de pacotes de controle de uma forma periódica para todos os participantes, podendo assim ter informações e controle de uma sessão.

O RTCP possui como função informar a qualidade da distribuição dos dados de um fluxo, podendo assim, ter um controle e diagnosticar falhas nas transmissões. Outro beneficio de se ter noção desses dados é a possibilidade de adaptar a taxa de transmissão de modo que funcione da forma mais eficiente.

Uma outra funcionalidade que o RTCP proporciona é a possibilidade de transportar um identificador de nível de transporte persistente mais conhecido como CNAME. Essa função é extremamente útil, devido ao

campo SSRC poder mudar, caso exista uma interrupção na transmissão. Nessa situação, caso a transmissão seja estabelecida, o CNAME poderá ser utilizado para que se mantenha as mesmas informações sobre cada participante.

Os diversos tipos de informações gerados pelo protocolo RTCP são citados abaixo:

- SR (*Sender Report*): Contêm informações sobre o envio e recebimento de pacotes RTP por participantes ativos em uma sessão;
- RR (*Receiver Report*) Contem informações sobre o recebimento de pacotes RTP por participantes que não são ativos em uma sessão;
- SDES (*Source Description*): Possui informações descritivas dos participantes de uma sessão, possui informações do seu CNOME;
- BYE: Indica o fim da participação de uma aplicação em uma sessão; e
- APP: Contêm funções especificas de cada aplicação.

3.5 PROTOCOLOS DE SINALIZAÇÃO VOIP

Assim como na telefonia convencional a sinalização é um importante mecanismo para proporcionar telefonia IP utilizando a tecnologia VoIP. Os principais protocolos utilizados na tecnologia VoIP são o H.323, SIP e mais recentemente o IAX (*Inter-Asterisk Exchange*).

Nesse projeto foi escolhido o protocolo SIP, mas antes de se abordar esse protocolo serão abordados, de forma resumida, os protocolos H.323 e o IAX.

3.5.1 IAX

IAX significa *Inter-Asterisk Exchange* e, é um protocolo desenvolvido pela Digium, sendo criado com o objetivo de estabelecer comunicação entre servidores Asterisk. O IAX é um protocolo que pode realizar o controle e a transmissão de dados de mídia.

Por ser capaz de transportar dados de mídia, o IAX dispensa a utilização dos protocolos RTP e RTCP. Além disso, o controle e a sinalização são passados utilizando-se uma única porta, 4569; trazendo como beneficio a unificação de duas atividade distintas em um único protocolo, minimizando problemas com o NAT.

Porém, o protocolo IAX traz como desvantagem a pouca disponibilidade de aparelhos IP que suportam esse protocolo, sendo esse e a pouca aceitação do mercado fatores que culminarão na escolha de um outro protocolo.

3.5.2 H.323

O H.323 é um protocolo proposto pela ITU-T com o objetivo de especificar sistemas de comunicação multimídia em redes baseadas em pacote. Além de proporcionar esse tipo de comunicação, o protocolo estabelece padrões para codificação e decodificação de fluxos de dados, garantindo assim que os produtos baseados no padrão H323 possam interoperar independente do fabricante.

O protocolo H.323 suporta tanto o protocolo TCP quanto o UDP. Para o transporte de sinais de controle é recomendável a utilização de protocolos confiáveis, que garantem a entrega dos pacotes e a ordem como foram

enviados.

Na arquitetura do protocolo H.323 existem quatro elementos que juntos possibilitam a comunicação multimídia , são eles:

- Terminais: São dispositivos específicos de recepção e envio de dados multimídia, não possuindo nenhuma restrição quanto ao tipo e o fabricante, desde que os mesmo suportem os padrões de codificação estabelecidos pelo protocolo;
- Gateways: São necessários para que diferentes tipos de rede possam estabelecer uma comunicação;
- Gatekeepers: São componentes opcionais em um sistema H.323. Funcionam como *gateway* de gerência, permitindo um controle centralizado do sistema, podendo gerenciar a largura de banda e controle de chamada para registrar os participantes;
- Multipoint Control Units (MCU's): Possibilitam a conferência entre três ou mais participantes, intermediando as negociações de parâmetros de comunicação entre eles.

A ITU-T desenvolveu recomendações que possibilitam que o H.323 ofereça outros tipos de serviços e funcionalidades. O quadro 3.4. os descreve.

Tabela 3.4 - Padrões referenciados na Recomendação H.323

PADRÔES	DESCRIÇÃO
H.245 **H.225.0**	**Protocolos de controle e sinalização**
H.246 **H.248**	**Interoperabilidade com redes de comutação de circuitos**
H.235	**Segurança**
H.450.X	**Serviços Suplementares**
H.460.X	**Extenção aos sistemas de sinalização**
H.501 **H.510** **H.530**	**Gerência e Segurança**

3.6 SIP

SIP (*Session Initiation Protocol*) é um protocolo proposto pela IETF (*Internet Engineering Task Force*), que apesar do curto tempo do processo de padronização, tem sido adotado por muitos fabricantes da área da telefonia e dados por causa da sua flexibilidade e capacidade de interoperar com outras aplicações de Internet e de arquitetura aberta.

O protocolo SIP tem a função de iniciar, configurar, gerenciar e encerrar sessões multimídia, como chamadas telefônicas ou conferências multimídia. O SIP é um protocolo de sinalização de nível de aplicação e pode funcionar sobre os protocolos UDP ou TCP.

Na arquitetura do SIP são definidas 05 entidades:

- Agentes de usuário: formado por uma parte cliente capaz de iniciar requisições SIP e por uma parte servidor, capaz de receber

e responder requisições SIP;

- Servidores de *proxy*: funciona como um intermediário, fazendo requisições para outros clientes que não podem fazer requisições diretamente;
- Servidores de redirecionamento: A função do servidor de redirecionamento SIP é fornecer a resolução de nome e locação do usuário. O servidor de redirecionamento SIP reponde ao pedido do Agente do Usuário fornecendo informações sobre o endereço do servidor para que o cliente possa contatar o endereço diretamente.
- Servidores de registro: fornece um serviço de informação de localidades; ele recebe informações do Agente do Usuário e armazena essa informação de registro.
- Gateways: funcionam como uma interface de uma rede que implementa o protocolo SIP com outra que implementa outro protocolo.

3.6.1 MENSAGEM SIP

O sistema de mensagem no protocolo SIP é bem simples, podendo ser requisições ou respostas. O SIP é um protocolo baseado em outros protocolos como é o caso do SMTP (*Simple Mail Transfer Control*) e o famoso HTTP *(Hiper Text Transport Protocol*), e assim como eles o SIP é um protocolo textual.

Uma mensagem SIP é constituída por várias linhas com a terminação dessas, por meio de caracteres CRLF (*Carriage Return Line Feed*). Além desse campo, as mensagens desse protocolo possuem uma linha inicial, um cabeçalho e o corpo da mensagem.

Nas mensagens de requisição, o formato da mensagem é caracterizado por utilizar uma linha de requisição como linha de inicio. Essa linha é formada por um método, um endereço e a versão do protocolo utilizado.

São especificados seis métodos principais utilizados para compor a linha de requisição. São eles:

- INVITE: Utilizado para convidar um novo participante para participar de uma sessão;
- ACK: Utilizada para confirmar o recebimento de uma requisição INVITE;
- BYE: Utilizado para finalizar uma sessão;
- CANCEL: Cancela uma requisição prévia;
- REGISTER: Registra informações de localização do participante;
- OPTIONS: Utilizado para consultar a disponibilidade de um servidor.

Já nas mensagens de resposta SIP utiliza-se uma linha de status como linha de início. Essa linha de status é composta pela versão do protocolo, um código numérico informando o status e sua frase textual correspondente.

O código informando o status é composto por três dígitos, sendo o primeiro digito utilizado para informar o tipo de resposta que esta sendo enviada. Já a frase textual serve como um complemento para justificar a mensagem de resposta. A tabela 3.5 mostra as classes e suas respectivas funcionalidades.

Tabela 3.5 - Classes de resposta e respectivas funcionalidades no SIP

CLASSE	FUNCIONALIDADE
1XX	**Resposta Informativa**
2XX	**Resposta de Sucesso**
3XX	**Resposta de Redirecionamento**
4XX	**Resposta de Falha de Requisição**
5XX	**Resposta de Falha em Servidor**
6XX	**Resposta de Falha Geral**

O cabeçalho da mensagem também é definido nas especificações do SIP, sendo uma seqüência estruturada de campos. Esses campos são parecidos, sintaxe e semanticamente, com os campos do cabeçalho do protocolo HTTP. Sua utilização nem sempre é obrigatória, podendo algumas vezes ser até opcional.

O corpo da mensagem é responsável por transportar informações relevantes para o contexto de algumas operações. Esse campo pode ser utilizado tanto para requisições quanto para respostas.

Quando a mensagem transporta algum tipo de informação, essa é indicada no valor do campo *Content –Type* que se encontra no cabeçalho da mensagem. A principal informação que pode ser transportada pelo SIP é o SDP e essa é indicada no campo *Content – Type* pelo valor *application/sdp*. Essa mensagem é utilizada para possibilitar a negociação de parâmetros necessários para estabelecer a sessão

3.6.2 SDP

SDP (*Session Description Protocol*) é um protocolo muito usado em conjunto com o SIP e tem a finalidade de transmitir informações sobre as

sessões. A descrição de uma sessão envolve a especificação de parâmetros relacionados aos pacotes de mídia envolvidos em uma sessão. Esses parâmetros podem ser divididos em dois níveis, o nível de parâmetros de sessão e o nível de parâmetros de mídia.

O nível de parâmetros de sessão contêm informações como o nome da sessão, o seu criador e o tempo que a sessão deve estar ativa. Já os níveis de parâmetros de mídia possuem informações do tipo de mídia, número da porta, protocolo de transporte e formato da mídia.

Assim como o SIP, o SDP também é um protocolo textual. Esse tipo de representação de informação foi adotado para facilitar a portabilidade, permitir uma variedade de formas de transporte e possibilitar que ferramentas que se baseiam em texto pudessem gerar e processar as descrições das sessões.

Para minimizar os problemas de largura de banda, causados pela utilização de textos ao invés de representações binárias, as descrições são especificadas de uma forma mais compacta, substituindo o nome dos campos por caracteres únicos.

3.6.3 ENDEREÇAMENTO SIP

Os endereços SIP possuem seu formato semelhante com os endereços de E-mail. Uma das diferenças encontradas é que os endereços de E-mail utilizam-se de um URL mailto (mailto. exemplo@endereço.com), enquanto que nos endereços SIP possui a seguinte sintaxe sip: joão@endereço.com.

O formato do endereço SIP é definido como um URI (*Universal*

Resouce Identifier) e possibilita adicionar informações referentes à porta, para qual a requisição dever ser enviada, e parâmetros que modificam o comportamento de uma requisição.

3.6.4 SINALIZAÇÃO SIP

O protocolo SIP possibilita a sinalização em dois diferentes modos: a sinalização direta ou per to per, que é realizada entre agentes SIP de forma que a o estabelecimento de uma conexão não necessite um servidor *proxy* e a sinalização indireta, que necessita de um servidor *proxy* e normalmente, de um ou outro servidor de apoio para estabelecer uma conexão.

3.6.4.1 SINALIZAÇÃO DIRETA

Na sinalização direta, o estabelecimento da conexão é realizado da forma mais simples. O agente cliente envia uma requisição diretamente para o agente servidor e esse responde também de forma direta. Ou seja, nesse tipo de sinalização não requer nenhum tipo dispositivo que faça essa a intermediação entre os agentes. A figura 3.8 ilustra a forma direta de sinalização.

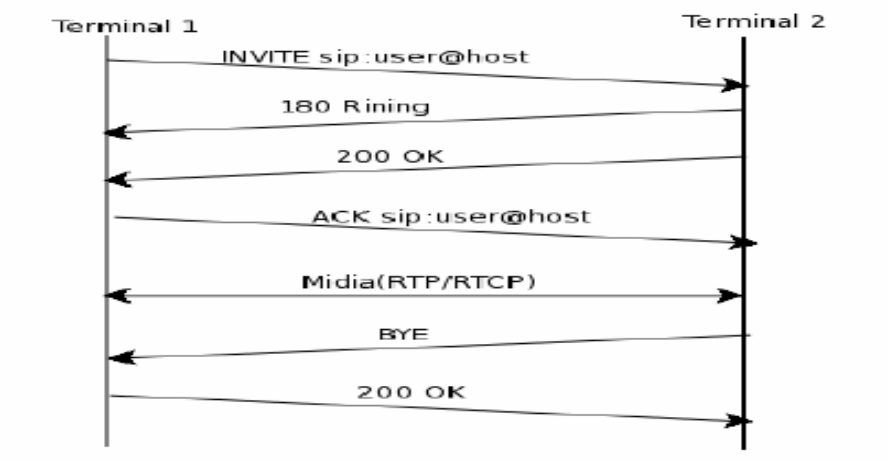

Figura 3.8 - Modelo de sinalização direta

De forma mais detalhada, a sinalização ocorre a partir de uma requisição INVITE, enviada diretamente pelo agente um para o agente dois. Após o agente dois receber essa requisição INVITE, ele a responde informando o tipo de CODEC que foi selecionado e a porta pelo qual recebera o RTP. Além disso, o agente dois envia também uma mensagem de OK aceitando a chamada requisitada.

Assim que o agente um recebe as duas mensagens enviadas pelo agente dois, é enviada uma requisição ACK confirmando o recebimento das mensagens. Com isso, é estabelecida uma conexão, possibilitando a comunicação entre os terminais.

Quando um dos terminais deseja encerrar a conversação uma mensagem BYE é enviada pelo agente que deseja desfazer a conexão, no caso da figura 3.9 o agente dois, para o agente da outra ponta. Após receber a mensagem de BYE o agente envia uma mensagem OK finalizando a chamada.

3.6.4.2 SINALIZAÇÃO INDIRETA

Na sinalização indireta o método para se estabelecer uma conexão se difere da sinalização direta pela utilização de um servidor de *proxy*. O servidor *proxy* é responsável por encaminhar as mensagens recebidas pelos agentes que não possuem permissão, ou seja, ele funciona como um intermediário.

O servidor de *proxy* também possui a função de proteger os servidores de registros e de redirecionar, pois sua utilização evita que esses servidores sejam acessados de forma direta.

A sinalização indireta começa de forma parecida com a sinalização direta, diferenciando-se apenas pelo destino que as requisições tomarão e pelo preenchimento do campo VIA presente no cabeçalho da mensagem.

O campo VIA é importante porque o endereço do agente e o endereço de cada servidor por onde a mensagem passa estão incorporados neste campo, tornando mais fácil o retorno da mensagem de resposta. Esse mecanismo funciona colocando, em ordem, o endereço em uma linha do campo VIA, cada vez que a mensagem passar por um servidor e após chegar ao destino, o campo VIA é copiado para a mensagem de resposta e toda vez que passar pelo servidor o endereço do mesmo é retirado. Caso o endereço não corresponda ao endereço do servidor a mensagem é automaticamente descartada

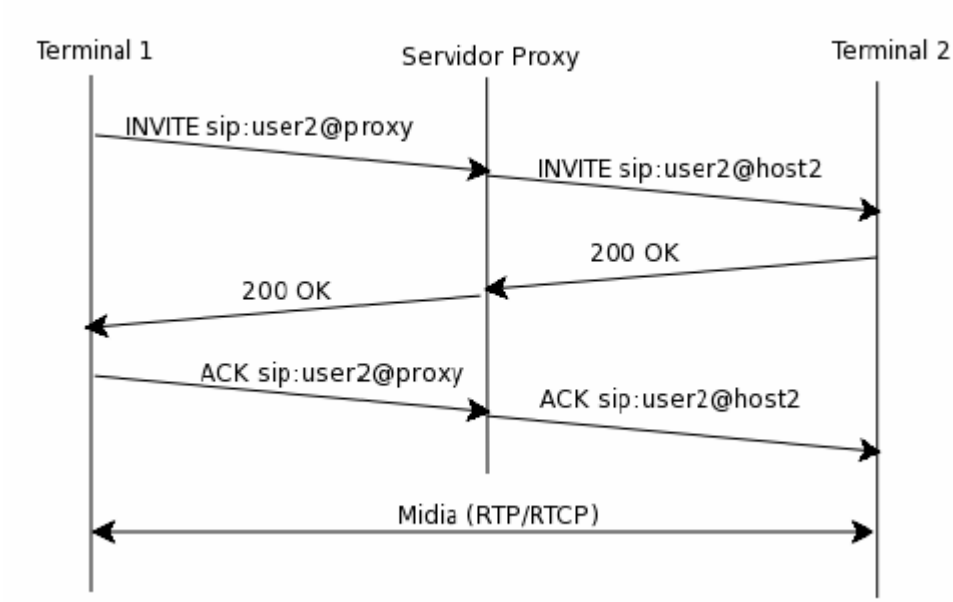

Figura 3.9 - Modelo de sinalização indireta

3.6.5 SIP X H.323

Os protocolos SIP e H.323 são atualmente os dois principais

protocolos responsáveis pela sinalização utilizados na tecnologia VoIP. O protocolo H.323 foi definido pala ITU-T no ano de 1996 e tem uma grande aceitação no mercado. Já o protocolo SIP foi definido três anos depois e vem ganhando cada vez mais espaço no mercado.

As vantagens de se utilizar o protocolo SIP começam pela sua agilidade na hora de se estabelecer uma conexão. Por utilizar menos procedimentos, como mostra a figura 3.10, o protocolo SIP estabelece conexões mais rapidamente que o protocolo H.323.

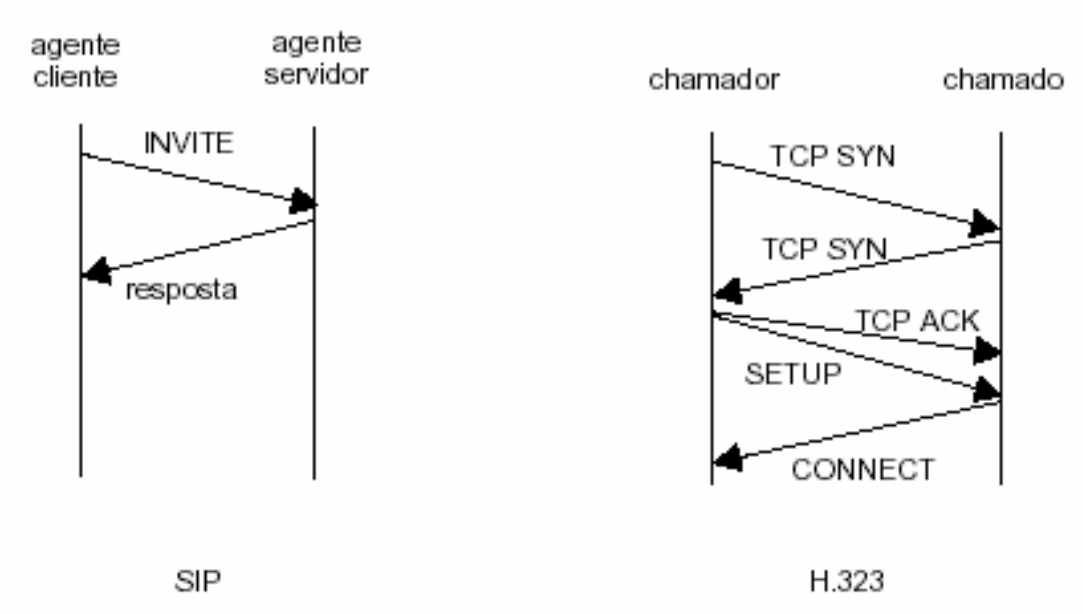

Figura 3.10 - Comparativo de sinalização SIP X H.323

Outra vantagem de se utilizar o protocolo SIP é por suas mensagens serem baseadas em texto, o que torna mais fácil o seu entendimento e permiti que novas características sejam inclusas de forma mais fácil e de maneira aleatória no cabeçalho de suas mensagens. Já no protocolo H.323 essas características devem ser inclusas em locais pré-definidos.

Para finalizar, a sinalização no protocolo H.323 funciona de forma

centralizada, independente do número de usuários, diferentemente do protocolo SIP que funciona de forma distribuída pelos usuários participantes, graças ao seu cabeçalho que permite incorporar algumas informações básicas para o tratamento das requisições e respostas.

CAPÍTULO 4. IMPLEMENTAÇÃO DO PABX

A arquitetura proposta nesse projeto será baseada na utilização de um software livre chamado *Asterisk*. Esse software será implementado em dois servidores, que farão a comunicação entre si e com os terminais utilizando um *switch*, como é mostrado na Figura 4.1.

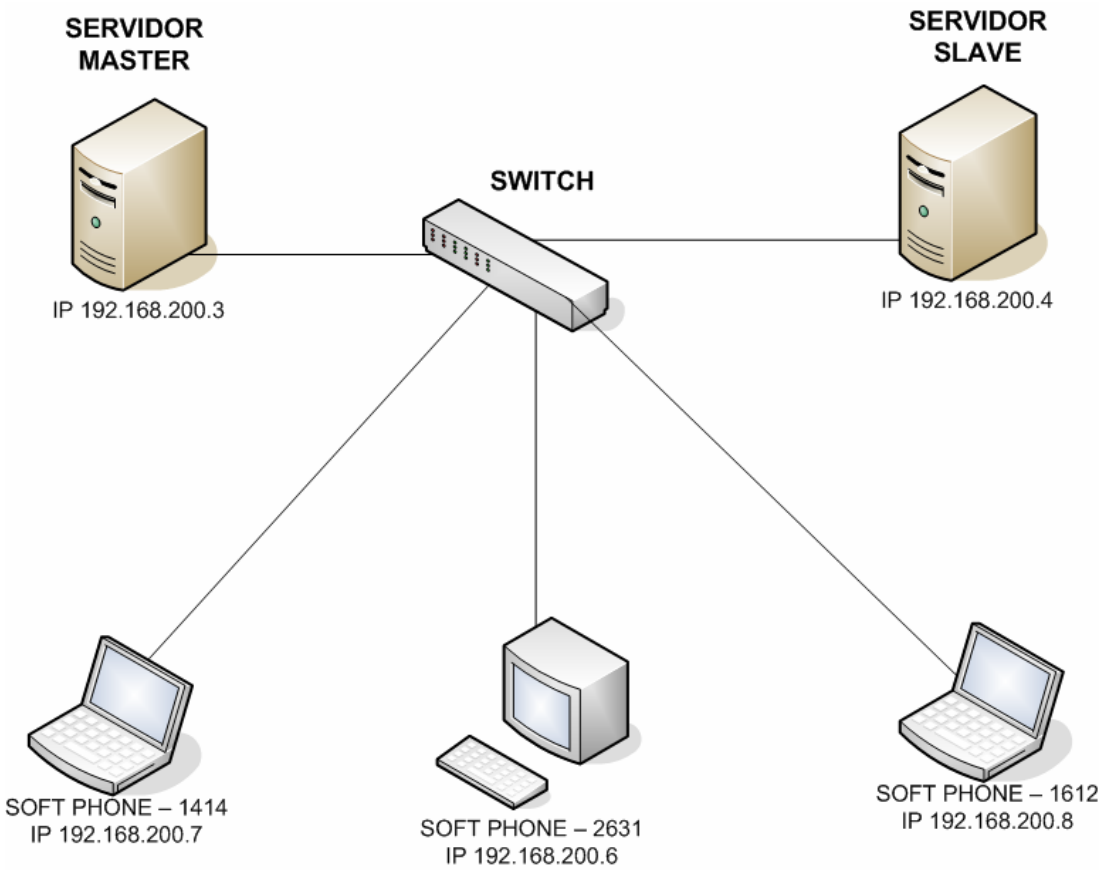

Figura 4.1 - Cenário de implementação

Os servidores da arquitetura serão o PABX do projeto possuem as seguintes configurações:

- Sistema Operacional Debian Sarge GNU/ Linux kernel 2.6.8-386;

- Software PABX IP Asterisk 1.0.9;
- Music player mpg123;
- Já o ambiente físico utilizado na implementação e testes do projeto possuem as seguintes características:
- Servidor Máster: microcomputador AMD Athlon XP 2000, 512 MB de memória RAM;
- Servidor SLAVE: microcomputador AMD Athlon XP 2000, 256 MB de memória RAM;
- Switch D-Link 8 portas 10/100 Mbps;
- Cliente 1414: microcomputador AMD Athlon XP 2000, 256 MB de memória RAM;
- Cliente 1612: Notebook Intel Pentium Centrino 1.7, 1 GB de memória RAM;
- Cliente 2631: microcomputador Intel Pentium 4 2.4GH e 512 MB de memória RAM;
- Após a apresentação do cenário os próximos capítulos apresentarão com mais detalhes o software *Asterisk* e o URCAP, que foi escolhido para prover alta disponibilidade ao sistema.

4.1 ASTERISK

O *Asterisk* é um software de código aberto criado pela Digium Inc. Esse software possibilita a implementação de um PABX utilizando a tecnologia de voz sobre IP. O software *Asterisk* foi criado para ser utilizado em plataforma Linux ou qualquer outra plataforma Unix.

A Digium possui sua sede em *Huntsville*, Alabama(EUA) e é a criadora e desenvolvedora pioneira do *Asterisk*, sendo que esse é considerado o primeiro software livre utilizado na implementação de um PABX.

A escolha pelo software *Asterisk* tem como razões principais a redução de custos devido à sua distribuição e possibilidade de rodar em sistemas operacionais gratuitos. O uso do *software Asterisk* permite ter um sistema de telefonia independente dos fornecedores, ou seja, as empresas que adotarem o PABX utilizando o *software Asterisk* não serão mais dependentes dos fornecedores para manutenção e modificação dos seus sistemas. E finalmente, o PABX utilizando esse software poderá ter maior flexibilidade e escalabilidade por ser um sistema de simples configuração como será apresentado mais a frente.

4.1.1 ARQUITETURA DO ASTERISK

O Asterisk foi desenvolvido para possibilitar o máximo de flexibilidade ao sistema. Para isso foram definidos alguns API`s (*Application Prograrmming Inteface*) especiais em torno do núcleo, deixando-o transparente a protocolos, CODECS e *hardware.* Com isso, o software *Asterisk* se torna uma ferramenta poderosa por ser compatível com qualquer tipo de tecnologia existente, sem a necessidade de mudanças no núcleo do programa.

A possibilidade de carregar os módulos de forma separada é outra característica do software Asterisk, o que possibilita ao administrador maior flexibilidade na configuração do sistema ao permitir a escolha dos módulos de acordo com a proposta do projeto que foi estabelecido.

De forma a ficar esclarecedor, API é um conjunto de rotinas e padrões estabelecidos por um software para utilização de suas funcionalidades por programas aplicativos, isto é, programas que não querem envolver-se em detalhes da implementação do software, mas apenas usar seus serviços.

A Figura 4.1 mostra a organização da Arquitetura do *software Asterisk,* mais a frente serão detalhados cada um desses elementos.

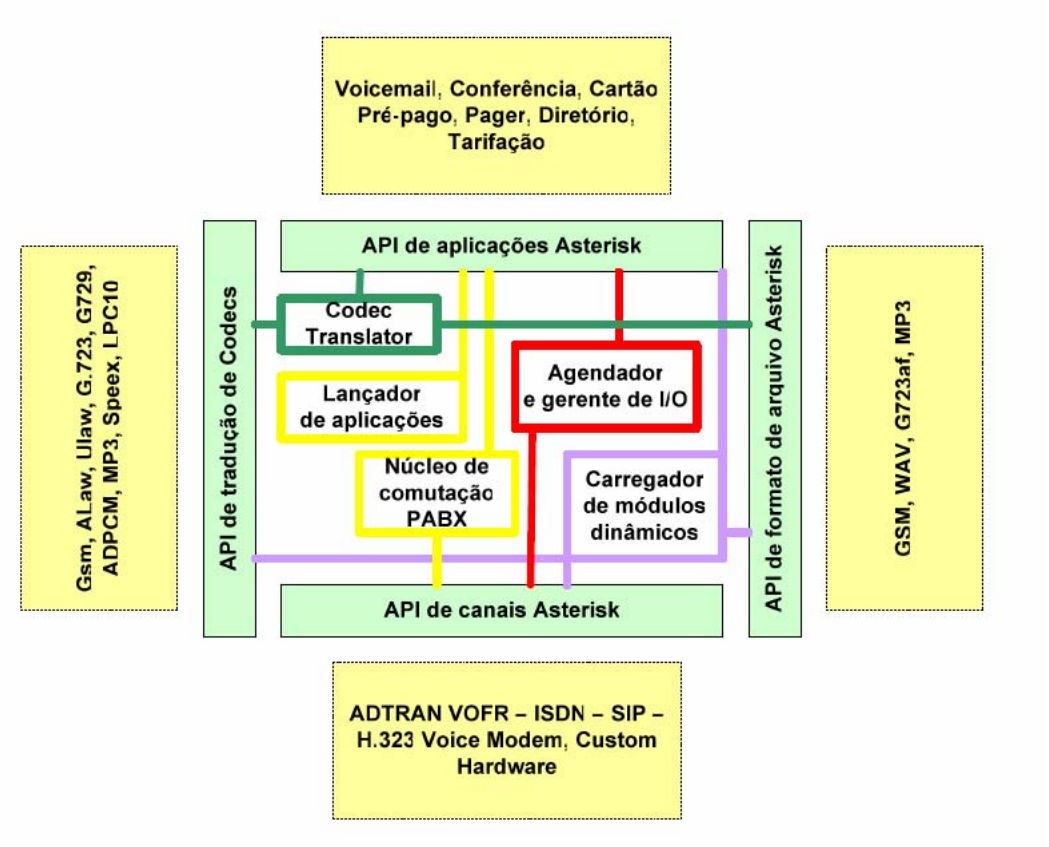

Figura 4.2 - Arquitetura Asterisk

- O núcleo da Arquitetura do software Asterisk é dividido nos seguintes módulos:
- Módulo PABX: É o principal módulo, responsável pela conexão das chamadas entre os vários usuários e por realizar as tarefas ditas automáticas;
- Lançador de Aplicativos: É o módulo responsável por executar os serviços estipulados no PABX, como por exemplo nesse trabalho será implementada a captura de chamada;
- Tradutor de CODECs: É responsável por traduzir os CODECs

suportados pelo software Asterisk;

- Módulo de Agendamento e Gerenciamento de E/S: É responsável pelo agendamento de tarefas de baixo nível e gerencia o sistema para torná-lo mais eficiente.

Já em relação às APIs, a arquitetura do software Asterisk pode ser dividida em quarto partes que serão vistas nos próximos itens.

4.1.1.1 CANAIS

Esse tipo de API é responsável por fazer a compatibilidade do núcleo com os diversos tipos de conexões onde as chamadas podem ser originadas. Para ficar mais clara a função dos canais pode-se dizer que um canal é o equivalente a uma linha telefônica na forma de um circuito de voz digital. Nessa linha podem ser transmitidos um sinal analógico ou alguma combinação de CODEC e o protocolo de sinalização.

O Asterisk suporta os seguintes canais:

- Agent: Um canal de agente DAC;
- Console: Cliente de console Linux, driver para placas de som (OSS ou ALSA);
- H323: Um dos protocolos mais antigos de VoIP;
- IAX e IAX2: Inter-Asterisk Exchange protocol, o protocolo do Asterisk;
- MGCP: Media Gateway Control Protocol, outro protocolo de VOIP.
- Modem: Usado para linhas ISDN e não modems.
- NBS: Usado para broadcast de som.
- Phone: Canal de telefonia do Linux.

- SIP: Session Initiation Protocol, o protocolo de VoIP mais comum.
- Skinny: Um driver para o protocolo dos telephones IP da Cisco.
- VOFR: voz sobre frame-relay.
- VPB: Linhas telefônicas para placas da Voicetronix.
- ZAP: Para conectar telefones e linhas com placas da Digium. Também usado para TDMoE (TDM sobre Ethernet) e para o Asterisk zphfc (ISDN em modo NT).

4.1.1.2 CODECS

É a API responsável por permitir que várias chamadas telefônicas sejam transmitidas em uma única rede de dados. Isso é possível utilizando os CODECS que têm a função de codificar o sinal em uma forma que use menos banda passante. O software Asterisk possui também tradutores de CODECs, que permitem que canais que utilizem diferentes CODECs com diferentes taxas de compressão de dados possam se comunicar.

O Asterisk suporta os seguintes CODECs:

- G.711 ulaw - (usado nos EUA) – (64 kbps);
- G.711 alaw - (usado na Europa e no Brasil) – (64 kbps);
- G.723.1 – Precisa de licenciamento (5.3-6 kbps);
- G.726 – 32 kbps no Asterisk 1.0.3, 16/24/32/40 kbps no CVS HEAD;
- G.729 – Precisa de licença, a menos que esteja usando o modo passthru.Versão gratuita disponível para uso em países sem patentes ou para uso educacional. (8kbps);
- GSM – (12-13 kbps);
- iLBC – (15 kbps);

- LPC10 - (2.5 kbps);
- Speex - (2.15-44.2 kbps).

4.1.1.3 PROTOCOLOS

Esse é a API responsável por possibilitar que os terminais possam se comunicar. É nessa API que se encontram os protocolos de sinalização que são responsáveis por estabelecer as conexões, determinar o ponto de destino e também desfazer as conexões. Os protocolos suportados pelo software Asterisk são:

- SIP
- H323
- IAXv1 e v2
- MGCP
- SCCP (Cisco Skinny).

4.1.1.4 APLICAÇÕES

Essa API é responsável por permitir que várias funcionalidades possam ser executadas no PABX implementado. Entre as funcionalidades destaca-se a captura de chamada, conferência e DAC que serão implementados nesse trabalho e que serão detalhados mais a frente.

4.1.2 APLICAÇÕES IMPLEMENTADAS

Abaixo serão listadas as aplicações que serão incorporadas ao PABX nesse projeto e mais adiante serão apresentados o modo como foram implementados.

Distribuidor automático de chamadas: É uma aplicação muito utilizada em *Call Centers*. Esse tipo de aplicação atende a chamada efetuada pelo usuário, caso não tenha nenhum operador disponível e enquanto aguarda a disponibilidade de algum atendente, uma música é executada e assim que um dos atendentes é liberado a chamada é automaticamente encaminhada para ele.

Servidor de música em espera: Esse é o sistema responsável pela musica em espera. A maioria das centrais necessita de um CD ou rádio acoplado para executar a música, já no *Asterisk* essas músicas rodam no formato MP3.

Sala de conferência: Permite que vários usuários possam falar em conjunto. Nesse sistema é disponibilizada uma sala onde, para entrar, basta apenas discar o ramal que é definido na configuração do sistema.

Captura de chamada: Nessa aplicação o usuário possui a capacidade de capturar uma chamada de um outro terminal que esta tocando sem se deslocar do seu local. Para isso basta discar, do seu telefone, um conjunto de números definidos na configuração do sistema, para que possa atender a ligação do ramal que está tocando.

Transferência de chamadas: Esse tipo de serviço possibilita que o usuário possa, a partir do seu terminal, transferir uma chamada utilizando um conjunto de comandos definidos na configuração do sistema.

Estacionamento de chamadas: Utilizado para colocar uma chamada em modo de espera. Nesse tipo de serviço o usuário digita um conjunto de comandos definidos na configuração do sistema quando desejar que o

usuário que esta na ligação aguarde um tempo.

4.1.3 INSTALAÇÃO E CONFIGURAÇÃO BÁSICA DO ASTERISK

Para começar a instalar o software Asterisk é preciso, primeiramente fazer a opção por um sistema operacional. Nesse projeto foi escolhido a distribuição do Linux - Suse 9.2. É recomendável, para não ter problemas de rendimento, escolher a instalação do sistema operacional com o mínimo de recursos possíveis e optar por uma rede com endereços de IP fixo. Lembrando que são apenas recomendações para tornar o sistema mais eficaz e de fácil configuração, não quer dizer que outras formas de configuração tornarão impossível a implementação do PABX.

Depois de instalado o sistema operacional deve-se instalar alguns pacotes adicionais necessários para a instalação e implementação do PABX, abaixo segue a tabela 4.1. contendo esses pacotes:

Tabela 4.1 - Pacotes Adicionais

Pacotes Adicionais
Gcc – GNU C Compiler and support.
Cvs – Concurrent Versions System.
Ncurses – New curses libraries.
Curses-devel – Biblioteca para desenvolvimento com ncurses.
Bison – The GNU parser generator.
Termcap – Termcap library.
Openssl – Secure Sockets and TLS Layer Security.
Openssl-developer – Bibliotecas do openssl.
Zlib-devel

Após a instalação dos pacotes adicionais, o sistema operacional esta pronto para que seja instalado e configurado o *software Asterisk*. Para tornar

mais simples sua implementação, o *firewall* foi desabilitado para que os protocolos de voz sobre IP possam operar. Caso se deseje configurar o *firewall*, as portas TCP e UDP são ambas configuradas na porta 5060.

Com o Linux e os pacotes adicionais devidamente instalados, pode-se começar a instalação do software *Asterisk*. Como primeiro passo, deve-se obter os arquivos de instalação, mas para isso, primeiramente, será apresentado o conceito de CVS, pois esse será utilizado na obtenção dos arquivos de instalação.

O CVS é um repositório central que as empresas desenvolvedoras utilizam para controle do código fonte. Esse controle é realizado através de atualizações, à medida que novas mudanças acontecem. A utilização do repositório CVS traz também como beneficio a possibilidade de armazenar todas as versões em um único arquivo, de forma a permitir que sejam gravadas apenas as diferenças entre uma e outra versão. Assim, caso ocorra algum problema com alguma das versões, esse pode ser levantado e corrigido sem grandes perdas.

Os arquivos de instalação do *Asterisk* serão obtidos do servidor CVS da Digium seguindo os seguintes comandos:

```
cd /usr/src/
export CVSROOT=:pserver:anoncvs@cvs.digium.com:/usr/cvsroot
cvs login
password (anoncvs)
cvs checkout –r v1-0 asterisk libpri asterisk-sounds asterisk-addons
```

Depois de executar os comandos acima citados, deve-se compilar e instalar o *Asterisk*, pois os comandos utilizados eram apenas para baixar os arquivos. Abaixo segue os comandos para executar essas ações.

```
cd /usr/src/asterisk/
make clean
make
make install
make samples
```

Após esses comandos, o *Asterisk* está pronto para ser executado com sucesso. A seguir serão apresentados os comandos para iniciar e parar o sistema.

```
Para iniciar:
/usr/sbin/asterisk –vvvgc

Para parar:
CLI>stop now
```

4.1.3.1 PROTOCOLO SIP

O protocolo SIP, como já foi mencionado anteriormente, é um protocolo baseado em texto, similar ao HTTP e SMTP, desenhado para iniciar, manter e terminar sessões de comunicação interativa entre usuários. Esse protocolo possui, entre outros elementos, um servidor de *proxy*, um servidor de redirecionamento e um servidor de localização, onde todos esses servidores podem ser representados em um único software *Asterisk*.

Neste capítulo serão apresentadas as configurações que irão possibilitar a comunicação entre os ramais. Para isso, deve-se cadastrar os números dos terminais no arquivo sip.conf, que pode ser encontrado o seguinte endereço: /etc/asterisk/sip.conf.

Na primeira parte do arquivo sip.conf devem-se encontrar as configurações globais, [general], onde são descritas as configurações gerais

do servidor. Posteriormente, são definidos os parâmetros de cada um dos clientes, onde serão cadastrados os respectivos ramais. Abaixo serão listados os principais parâmetros utilizados na configuração seguida da configuração adotada no trabalho.

- Allow: Permite que um determinado codec seja usado;
- Bindaddr: Endereço IP onde o *Asterisk* irá esperar pelas conexões SIP. O comportamento padrão é esperar em todas as interfaces endereços secundários;
- Port: Porta que o *Asterisk* deve esperar por conexões de entrada SIP. O padrão é 5060;
- Disallow: Proíbe determinado codec;
- Type: Configura a classe de conexão, as opções são: peer, user e friend;
    1. Peer: Entidade que o *Asterisk* envia chamadas;
    2. User: Entidade que faz chamadas através do Asterisk;
    3. Friend: Funciona tanto como Peer quanto como User, o que faz sentido na utilização de telefones;
- Host: Configura o endereço IP ou o nome do host. Pode se usar também a opção dynamic, onde se espera que o telefone se registre, é a opção mais comum;
- Username: Esta opção configura o nome do usuário que o Asterisk tenta conectar quando uma chamada é recebida.

Esses são apenas alguns parâmetros utilizados na configuração do arquivo sip.conf, existem outras que podem ser utilizadas e que podem ser vista no Anexo 1. Os quadros a seguir mostram as configurações feitas no projeto. O primeiro quadro mostra as configurações gerais do arquivo e o segundo as configurações dos ramais utilizados no projeto.

Configuração geral do Arquivo

```
 [general]
context=default
port=5060
bindaddr=10.61.217.90
srvlookup=yes
```

Entradas dos ramais

```
[4102]
type=friend
username=4102
canreinvite=no
disallow=all
callerid=daniel
host=dynamic
nat=yes
allow=gsm
allow=ulaw

[4103]
type=friend
username=4103
can reinvite=no
disallow=all
callerid=rodrigo
host=dynamic
nat=yes
allow=gsm
allow=ulaw

[4104]
type=friend
username=4104
can reinvite=no
disallow=all
callerid=pedro
host=dynamic
nat=yes
allow=gsm
allow=ulaw
```

4.1.3.2 PLANO DE DISCAGEM

O plano de discagem é a parte principal na configuração do PABX, é nele que se define a maneira como o s*oftware Asterisk* irá conduzir as chamadas. Sua estrutura é montada utilizando instruções que o *Asterisk* irá seguir passo a passo a partir dos dígitos recebidos de um ramal ou aplicação.

A maior parte das configurações do plano de discagem são feitas no extensions.conf, que se encontra no diretório /etc/asterisk. Esse arquivo pode ser dividido em 4 partes:

- Contexto;
- Extensões;
- Prioridades;
- Aplicações;

O contexto tem um papel importante na organização e segurança do plano de discagem. Ele possui também a responsabilidade de organizar o plano de discagem em diversas partes de acordo com as ações definidas para cada ramal.

Para um melhor entendimento do que foi dito até aqui, pode-se, por exemplo, configurar os ramais em duas classes, os que podem fazer ligações de longa distância e os que não podem. Essa divisão pode ser obtida criando-se dois contextos: [autorizados] e [não autorizados] e dentro desses, o *Asterisk* possibilita configurá-los de forma que só os ditos autorizados conseguiriam completar as ligações de longa distância.

Essa possibilidade só pode ser alcançada porque os contextos estão

diretamente relacionados aos canais, ou seja, cada canal esta presente dentro de pelo menos um contexto e esse é responsável por processar as ligações. Os contextos são declarados dentro de chaves ([EXEMPLO]) e todas as instruções, representadas por extensões, colocadas após a declaração fazem parte desse contexto.

As extensões também fazem parte do plano de discagem e são os responsáveis por disparar os eventos. É através deles que se pode saber qual será o próximo passo que a chamada seguirá. Sua sintaxe é representada da seguinte forma:

```
exten=> numero (nome), prioridade, aplicação
```

Prioridade são os passos numerados na execução de cada extensões. Cada prioridade chama uma aplicação específica. Normalmente as prioridades começam com 1 e aumentam de uma a uma em cada extensão, sendo rodadas na ordem numérica.

As aplicações, são partes fundamentais no plano de discagem, pois a sua utilização é que dita o que cada extensão irá fazer. Dentre as aplicações, a *Dial*() se destaca nesse projeto, pois é devido à sua utilização é que se pode iniciar uma sessão. A configuração básica do arquivo extensions.conf utilizada nesse projeto é mostrada no quadro abaixo. Mais a frente outras extensões serão incluídas com o objetivo de incluir novas funcionalidades ao PABX.

```
[principal]

exten=>4102,1,Dial(SIP/4102,20)
exten=>4103,1,Dial(SIP/4103,20)
exten=>4104,1,Dial(SIP/4104,20)
```

As configurações básicas de um PABX já estão concluídas. O próximo passo nesse projeto será a implementação dos serviços avançados. No próximo tópico serão implementadas essas novas funcionalidades.

4.1.4 SERVIÇOS AVANÇADOS

Nessa parte do projeto serão apresentados os diversos serviços implementados no PABX e a maneira como foi feita a configuração de cada um deles. O arquivo features.conf controla a maioria dos serviços que são citados abaixo.

- Captura de chamadas;
- Sala de conferência;
- Música em espera;
- Estacionamento de chamadas;
- Transferência de chamada;
- Distribuição automática de chamada.

4.1.4.1 CAPTURA DE CHAMADA

A captura de chamada é um serviço que permite que o usuário consiga atender uma chamada destinada a outro usuário sem que haja necessidade de se deslocar até o telefone que está tocando.

Para habilitar esse serviço é necessário configurar um grupo de telefones que podem atender aos telefones desse grupo. No caso desse projeto o arquivo que deve ser configurado é o sip.conf e todos os telefones estão cadastrados no mesmo grupo podendo então, atender as ligações destinadas aos outro telefones.

Depois de configurado o grupo deve-se inserir no arquivo features.conf o seguinte comando:

```
Pickupexten=77;
```

Esse comando configura a extensão que será usada para a captura das chamadas. No caso do projeto, ao discar o numero 77, a chamada em execução será capturada pelo telefone.

4.1.4.2 SALAS DE CONFERÊNCIA

As salas de conferência possibilitam a comunicação entre mais de dois usuários em uma mesma sessão. Para que seja possível esse serviço, é necessário configurar primeiramente o arquivo extensions.conf, criando um novo contexto como foi implementado no projeto e é mostrado abaixo:

```
[conferencia]

exten=>1010,1,MeetMe(1010)
exten=>1011,1,MeetMe(1011)
```

Os comandos acima mostram a criação de duas salas de conferência, a primeira sala no ramal 1010 e a segunda no ramal 1011, sendo que na segunda sala existe uma senha de acesso (1234) que será criada no arquivo meetme.conf mostrado abaixo.

```
[rooms]

conf=>1010
conf=>1011,1234
```

4.1.4.3 MÚSICA EM ESPERA

Há diversas formas de se utilizar a música em espera. Nesse projeto foi escolhido o mpg 123. Para isso deve-se baixar e compilar o pacote mpg 123, que se encontra na pagina http://www.mpg123.de/cgi-bin/siteexplorer.cgi?/mpg123/.

Depois de compilar o mpg 123 deve-se adicionar no arquivo /etc/asterisk/zapata.conf a seguinte linha:

```
[channels]
musiconhold=default
```

Em seguida retire no arquivo /etc/asterisk/musiconhold.conf o comentário da seguinte linha:

```
default=>mp3;/var/lib/asterisk/mohmp3
```

Como forma de testar a música de espera foi criado um contexto que ao discar o numero 6666 escuta-se uma música durante 10 segundos.

```
[testemusica]
exten=>6666,1,WaitMusicOnHold(10)
```

4.1.4.4 ESTACIONAMENTO DE CHAMADA

O estacionamento de chamada tem como princípio colocar uma chamada em um modo de espera após discar uma extensão e depois recuperá-la discando uma outra extensão que é anunciada pelo sistema.

Para habilitar esse serviço é preciso incluir no arquivo extensions.conf a seguinte linha:

```
include=>parkedcalls
```

Depois, deve-se configurar o arquivo features.conf como mostrado a seguir. Nessa configuração serão definidas as extensões as quais deve-se discar para colocar a chamada em espera, as extensões serão utilizadas para estacionar a chamada e o tempo que elas ficaram retidas.

```
parkext=>700
parkpos=>701-720;
context=>parkedcalls
parkingtime=>60
```

4.1.4.5 TRANSFERÊNCIA DE CHAMADAS

A transferência de chamadas é utilizada para encaminhar uma chamada para um outro terminal. Existem duas formas de transferir uma chamada: a transferência às cegas e a transferência assistida.

Na transferência às cegas o usuário encaminha a chamada e desliga o telefone. Já na transferência assistida, antes de encaminhar a chamada o recurso possibilita que se fale com o usuário para quem se pretende encaminhar a chamada, dizendo dessa forma, quem é o usuário que será encaminhado.

A configuração desse serviço é feita no arquivo features.conf. Será apresentada a seguir a forma como ele foi configurado nesse projeto.

```
Transferdigittimeout=>3
Xfersound=beep
Xferfailsound=bepperr

[featuremap]
Blindxfer=>#1
Disconnect=>#0
Automon=>*1
Atxfer=>#2
```

4.1.4.6 DISTRIBUIÇÃO AUTOMÁTICA DE CHAMADAS

O serviço de distribuição de chamadas tem sua importância principalmente em empresas de *Call Center*. O princípio desse serviço é enfileirar as chamadas e encaminhá-las de acordo com a disponibilidade dos atendentes.

A utilização de filas evita perder chamadas quando os atendentes não estão disponíveis, o que melhora a produtividade e a qualidade do atendimento.

Essas filas são definidas no arquivo queues.conf e os atendentes no arquivo agents.conf. Abaixo serão apresentadas as configurações utilizadas nesse projeto. Primeiramente será apresentado o arquivo queues.conf.

```
[exemplo]
Music=default
Timeout=2
Retry=2
Maxlen=0
Member=>Agent/300
Member=>Agent/301
```

Depois de configurado o arquivo queues.conf deve-se configurar o arquivo agents.conf, onde serão habilitado os atendentes.

```
[agents]
autologoff=15
ackcall=yes
wrapuptime=5000
musiconhold=>default
custom_beep=beep
group=1
agent=>300,300,Atendente1
agent=>301,301,Atendente2
```

E para finalizar a configuração desse serviço é necessário criar a fila no plano de discagem e o login do agente.

```
exten=>1234,1,Answer
exten=>1234,2,Queue(exemplo)

exten=>0000,1,Wait,1
exten=>0000,2,AgentCallBackLogin()
```

4.2 CLUSTER DE ALTA DISPONIBILIDADE

O conceito de disponibilidade tem relação com a probabilidade de um sistema estar disponível em um dado momento. Sendo assim, quando determinado sistema possui disponibilidade de 98%, significa que em um dado momento em que se deseja utilizar esse sistema, tem-se 98% de probabilidade de que o sistema esteja em funcionamento.

Todo sistema, de uma forma geral, possui uma disponibilidade que o caracteriza. Essa disponibilidade em sistemas que não possuem mecanismos especiais que visem de alguma forma melhorar essa característica, possuem uma disponibilidade conhecida como disponibilidade básica.

Nos sistemas que possuem algum mecanismo especializado em

detectar, recuperar e mascarar as falhas pode-se aumentar a disponibilidade, de forma que possua as características que o considerem um sistema de alta disponibilidade. Esses sistemas possuem uma porcentagem bem próxima dos 100% de disponibilidade, sendo que as chances do sistema estar inoperante são desprezíveis ou até mesmo inexistentes.

Neste projeto uma das características que se deseja alcançar é a alta disponibilidade e para que seja possível será utilizado um protocolo que possibilite que os servidores estejam em cluster. Mais a frente será detalhado esse protocolo, mas antes será abordado o conceito de cluster.

Cluster é o nome que se dá ao processo de interconexão de múltiplas máquinas com o objetivo de se obter um aumento de disponibilidade e ou desempenho.

Há vários tipos de cluster, entre eles, pode-se mencionar os de processamento distribuído ou processamento paralelo e o cluster de alta disponibilidade que foi utilizado nesse projeto.

O cluster de processamento distribuído tem como objetivo aumentar a disponibilidade e performance do sistema. Para obter esse aumento de desempenho, o cluster de processamento distribuído utiliza como método a distribuição, entre as estações, de pequenas tarefas para que sejam processadas ao mesmo tempo, diminuindo assim, o tempo de execução de uma grande tarefa.

Já cluster de alta disponibilidade é desenvolvido para prover ao sistema uma disponibilidade de serviço e recurso de forma continua através do uso de redundância. Essa redundância é obtida através do

compartilhamento de uma interface virtual entre um grupo de equipamentos cooperativos.

Os clusters de alta disponibilidade são comumente conhecidos como *Failover Cluster. Failover* é o processo no qual uma máquina assume os serviços que a outra máquina estava executando quando essa apresentar alguma falha.

É importante que nesse tipo de cluster as máquinas envolvidas possuam recursos equivalentes. Por recursos deve-se entender ser uma placa de rede, um disco rígido, os dados neste disco ou qualquer elemento necessário à prestação de um determinado serviço.

O PABX desse projeto, como já foi dito anteriormente, tem como característica a alta disponibilidade. Para isso será utilizado o protocolo UCARP como ferramenta para possibilitar que os terminais tenham menor probabilidade de desconexão durante uma conversação.

A sigla UCARP tem como significado Userland CARP (*Common Address Redundancy Protocol*). Este protocolo é uma alternativa melhorada do VRRP (*Virtual Router Redundancy Protocol*), protocolo utilizado e desenvolvido pela empresa CISCO.

A palavra *userland* é utilizada para significar que o protocolo CARP funciona em modo de usuário, o que facilita o processo de implementação por não precisar aplicar nenhum patch ao Kernel.

O CARP é um protocolo *multicast* que possibilita agrupar vários equipamentos físicos funcionando de modo singular sob a representação de endereços virtuais. Nesse contexto, uma das máquinas é escolhida para ser

o principal e esse tem a responsabilidade de responder a todos os pacotes endereçados ao conjunto de equipamentos. Já as outras máquinas são conhecidas como *slaves* e ficam apenas na escuta esperando para assumir a função da máquina principal.

De tempos em tempos a máquina principal envia a rede, por *multicast*, informações sobre o seu estado, preparando assim, as maquinas *slaves* para que essas tenham condições de assumir o papel de principal caso aconteça algum problema.

Como condição para correto funcionamento, o CARP exige que cada interface de rede tenha o seu próprio IP real, pois só assim a maquina principal poderá se comunicar com as maquinas salves através de *multicast*.

4.2.1 IMPLEMENTAÇÃO DO UCARP

De acordo com o cenário apresentado na Figura 4.2, os servidores terão cada um o seu respectivo IP e devido à utilização do protocolo UCARP esses dois servidores serão representados por um IP virtual.

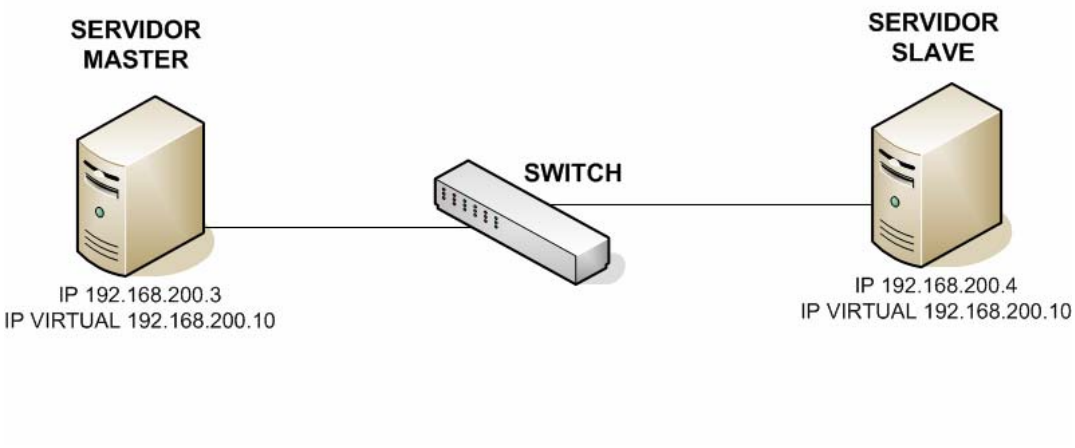

Figura 4.3 - Cenário do cluster

A necessidade de cada servidor possuir um IP estático, como foi abordada anteriormente, se deve à comunicação entre os servidores ser realizada da forma *multcast*. Essa comunicação é importante para que os dois servidores estejam alinhados, e caso haja algum problema como o servidor *máster* o servidor *slave* assuma as funções dele.

Para começar a implementação do cluster será necessário adquirir os pacotes de instalação do URCAP. Os pacotes utilizados nesse projeto foram adquiridos no site oficial do UCARP – http://www.ucarp.org.

Depois de ter feito o download dos pacotes deve se compilar o arquivo e instalar o UCARP. Nota-se que é importante que se instale os arquivos nos dois servidores

Agora, para finalizar a implementação do projeto, serão apresentados os métodos utilizados para configurar o UCARP e, conseqüentemente, o cluster de alta disponibilidade.

Para facilitar o entendimento da implementação, primeiramente será configurado o servidor *máster* e depois o servidor *slave.*

```
Utilizar o seguinte comando
ucarp -i eth0 -P -k 1 -s 192.168.0.1 -v 15 \-p password -a 192.168.0.10 -u
/etc/ucarp/vip-up.sh \-d /etc/ucarp/vip-down.sh -B
```

Depois insere-se os seguintes comandos no arquivo vip-up.sh que se encontra em: vi /usr/local/sbin/vip-eth0-up.sh

```
#!/bin/bash
/sbin/ip addr add 192.168.1.10/24 dev eth0
```

Faça o mesmo no arquivo vip-down.sh que se encontra em: vi /usr/local/sbin/vip-eth0-down.sh

```
#!/bin/bash
/sbin/ip addr del 192.168.1.10/24 dev eth0
```

Já no servidor *slave* o procedimento é parecido como pode ser notado a seguir.

```
Utilizar o seguinte comando
ucarp -i eth0 -P -k 1 -s 192.168.0.2 -v 15 \password -a 192.168.0.10 -u
/etc/ucarp/vip-up.sh \-d /etc/ucarp/vip-down.sh -B
```

Depois, seguindo o mesmo caminho do servidor máster, insira os seguintes comandos no arquivo vip-up.sh que se encontra em: vi /usr/local/sbin/vip-eth0-up.sh

```
#!/bin/bash
/sbin/ip del add 192.168.1.10/24 dev eth0
```

O mesmo deve ser feito no arquivo vip-down.sh que se encontra em: vi /usr/local/sbin/vip-eth0-down.sh

```
#!/bin/bash
/sbin/ip addr add 192.168.1..1/24 dev eth0
```

A título de curiosidade serão apresentadas, a seguir, algumas vantagens que culminarão com a escolha do protocolo UCARP ao invés de se utilizar o protocolo *heartbeat.*

- Facilidade de implementação;

- Velocidade de *Take Over* / *Take Back;*
- Baixo uso da rede, pois os pacotes utilizados para verificação são pequenos;
- Não necessita de interfaces de redes adicionais para a utilização do cluster.

CAPÍTULO 5. TESTES E RESULTADOS

Neste capitulo será apresentada a maneira como os testes foram realizados e quais os resultados obtidos a partir desses testes. Os clientes SIP foram todos softphone X-LITE, pois os telefones IP do fabricante SOYO estavam trazendo problemas de configuração durante a implementação do PABX.

Os testes foram realizados em etapas. Algumas funcionalidades foram testadas durante um tempo satisfatório e pré-determinado, já outras, por não serem possíveis utilizar o método citado, foram realizadas utilizando como método repetição.

Para o estabelecimento das chamadas foram utilizados 3 terminais com os seguintes ramais: 1414, 1610 e 2631. Esses números foram escolhidos de forma aleatória, evitando-se que fossem consecutivos, com o objetivo de tornar mais real a aplicação do PABX.

Abaixo estão apresentadas as etapas dos testes das funcionalidades do PABX:

- Estabelecimento de chamada entre dois terminais;
- Música em espera;
- Transferência de chamada;
- Conferência;
- Captura de chamada;
- Estacionamento de chamada;
- Distribuição automática de chamada;
- Cluster

5.1 ESTABELECIMENTO DE CHAMADA ENTRE DOIS TERMINAIS

Foram realizados testes entre os 03 terminais com a duração de um minuto para cada chamada, como mostra a Tabela 5.1.

Tabela 5.1 - Estabelecimento de chamada

ORIGEM	DESTINO	DURAÇÃO
1414	1610	1 mim
1414	2631	1 mim
2631	1610	1 mim

Os testes obtiveram um resultado satisfatório, sendo que todas as ligações transcorreram com uma qualidade semelhante às ligações efetuadas no sistema telefônico tradicional.

5.2 MUSICA EM ESPERA

Para a realização dos testes da funcionalidade música em espera, foi criada uma extensão no arquivo extensions.conf para acionar a música. O numero criado foi 6666 e todos os terminais realizaram a chamada para este ramal. O tempo de teste para cada ligação foi de 30 segundos, pois no momento da criação da extensão, esse foi o tempo estipulado para a duração da ligação.

Durante os testes, os ramais 1414, 1610 e 2631 conseguiram acessar ao ramal 6666 e durante essas chamadas a musica pode ser ouvida perfeitamente.

5.3 TRANSFERÊNCIA DE CHAMADA

Os testes realizados para se testar a funcionalidade transferência de chamada foram feitos utilizando os 3 terminais 1414, 1610 e 2631 de acordo com a tabela 5.2.

Tabela 5.2 - Transferência de chamadas

ORIGEM	DESTINO	TRANSFÊRENCIA
1414	1610	2631
1414	2631	1610
2631	1610	1414

As ligações, entre a origem e o destino, ocorreram de forma satisfatória e a transferência foi realizada com sucesso.

5.4 CONFERÊNCIA

Nesse tipo de funcionalidade pode-se utilizar mais de 3 terminais, porém, nesse projeto, foram utilizados apenas os três terminais que foram implementados. Os resultados obtidos foram bastante satisfatórios. Foram realizados dois testes com todos os terminais, durante dois minutos cada, e obteve-se um som perfeitamente inteligivel entre os terminais.

5.5 CAPTURA DE CHAMADA

A captura de chamada foi realizada com a participação de todos os terminais desse projeto e foram realizadas de acordo com a Tabela 5.3.

Tabela 5.3 - Captura de chamadas

ORIGEM	DESTINO	CAPTURA
1414	1610	2631
1414	2631	1610
2631	1610	1414

Como mostrado na tabela, a origem disca o ramal destino e a chamada é atendida no ramal captura. Os testes foram realizados com sucesso.

5.6 ESTACIONAMENTO DE CHAMADA

Os testes realizados com a funcionalidade estacionamento de chamada tiveram como fator limitante o tempo que foi programado para a música em espera. A Tabela 5.4 mostra como foi o procedimento dos testes.

Tabela 5.4 - - Estacionamento de chamada

ORIGEM	DESTINO	CAPTURA	TEMPO
1414	1610	2631	10 seg
1414	2631	1610	15 seg
2631	1610	1414	20 seg

O número origem disca o ramal destino e após 10 segundos de ligação a chamada é estacionada sendo capturada depois de certo tempo, como especificado na Tabela 5.4. Os testes foram bem sucedidos e a funcionalidade bem executada.

5.7 DISTRIBUIÇÃO AUTOMÁTICA DE CHAMADA

Para a realização dos testes desta funcionalidade foram cadastrados dois atendentes, o ramal 1414 e o ramal 1610. Os testes foram divididos em duas partes: uma com os dois telefones no gancho e a outra com os dois telefones fora do gancho e após a ligação um dos telefones é posto no gancho.

No primeiro caso, o ramal 2631 efetuou a chamada para o atendente

que se encontrava no ramal 1414, e depois o ramal 1631 fez a mesma ligação com o atendente do ramal 1610 atendendo. Nos dois casos os resultados foram satisfatórios e ocorreram sem problemas.

Já no segundo caso, o ramal efetuou a chamada e como nenhum dos atendentes estava disponível, a chamada foi posta no modo de espera. Após 15 segundos o ramal 1610 colocou o telefone no gancho e a chamada foi transferida para o seu ramal sem nenhum problema, demonstrando que o serviço foi realizado com sucesso.

5.8 CLUSTER

Para uma melhor análise, os testes do cluster foram divididos em duas partes. Na primeira parte, foram realizados os testes durante uma chamada simples entre dois terminais e na segunda parte foram realizados os testes durante uma conferência entre os três terminais.

Na primeira parte foi utilizada a mesma metodologia aplicada nos testes do tópico 5.1 - estabelecimento de chamada entre dois terminais. Durante a chamada entre os terminai o servidor Máster, servidor onde estava sendo executado o software Asterisk, foi desligado e como esperado, o servidor Slave assumiu as funções do servidor Máster e os *softphone* foram recadastrados nesse servidor podendo assim realizar as ligações e utilizar as funções normalmente.

Na segunda parte utilizou-se a metodologia aplicada na conferência de chamada do tópico 5.4. Nesse teste os terminais discavam para a sala de conferência e durante a chamada o servidor Máster foi desativado. Os resultados foram semelhantes aos encontrados no primeiro teste.

CAPÍTULO 6. CONCLUSÃO

De acordo com as pesquisas realizadas e os testes executados pode-se concluir que a tecnologia de voz sobre IP tem um futuro bastante promissor. A implementação do PABX utilizando o software Asterisk mostrou-se bastante flexível, possibilitando a implementação de diversas funcionalidades.

A utilização do software Asterisk também possibilitou a arquitetura do projeto ser bastante flexível, podendo adicionar novos ramais ou mudá-los de forma rápida e sem modificar fisicamente o projeto.

Outro ponto que merece ser destacado na implementação desse projeto foi o cluster de alta disponibilidade que possibilitou, dentro do esperado, um serviço com maior disponibilidade, enriquecendo ainda mais as qualidades do PABX implementado.

A tecnologia de voz sobre IP deverá ter grande aceitação, principalmente no mercado corporativo, devido a suas funcionalidades e a escalabilidade que é proporcionada pela sua implementação, como visto no projeto. Já o mercado residencial, deverá demorar um pouco mais, devido aos custos de implementação e a grande maioria da população ainda não esta preparado para esse tipo de tecnologia, preferindo ainda a telefonia convencional.

Propõem-se como projetos futuros:

- A implementação de IPBX simulando duas filiais situadas em pontos distantes, utilizando o protocolo IAX para a comunicação entre os servidores;
- A implementação de um IPBX com possibilidade de comunicação com a rede telecomunicação convencional.

REFERÊNCIAS BIBLIOGRÁFICAS

COLCHER, Sergio; GOMES, Antônio Tadeu A.; SILVA, Anderson Oliveira da, FILHO, Guido L. de Souza; Soares, Luiz Fernando G. **VoIP: Voz sobre IP**. - 1 ed. Rio de Janeiro: Elsevier, 2005

FERNANDES, Nelson Luiz Leal. **Voz sobre Ip:Uma visão geral** - Rio de Janeiro, fevereiro, [2003]. Acesso em: 01 jun. 2003. Disponível em: **http://www.ravel.ufrj.br/publicacoes/tipos.php?IdTipo=4**.

FERRARI, Antonio Martins. **Telecomunicações: Evolução & Revolução**. 9. ed. São Paulo: Erica, 2005

FILHO, Huber Bernal. **Soluções corporativas usando VoIP**. 2005. Acesso em: 01 jun. 2006. Disponível em: **http://www.teleco.com.br/**

MARTINS, Roberto. **A Fundamentação da Telefonia através da História - Parte 1: Da Invenção ao Início do Século XX** [2003] pesquisa realizada para a Fundação Telefônica, em 2002)**.** Acesso em: 01 jun. 2006. Disponível em: **http://www.museudotelefone.org.br/descoberta.htm.**

MELCHIOR, Silvia Regina Barbuyl. **Soluções corporativas usando VoIP**. 2004. Acesso em: 01 jun. 2006. Disponível em: **http://www.teleco.com.br/**

SILVA, Juliana Daniela da; COSTA, Karla Medeiros; CARVALHO, Suelma de Oliveira Carvalho, MARQUES, Viviane Mendes. Qualidade de Serviço em ambiente VoIP. 2005. 84 f. Trabalho de Conclusão CURSO SUPERIOR DE TECNOLOGIA EM REDES DE COMUNICAÇÃO, CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE GOIÁS.

SOARES, Luiz Fernando G; LEMOS, Guido; COLHER, Sérgio. **Redes de computadores:** das LANs, MANs e WANs às redes ATM. 2. ed. Rio de Janeiro: Campus, 1995.

SOUZA, José Marcio de. **Protótipo de um sitema de VoIP(Voz sobre IP)**. 2001. 62 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau

CAPÍTULO 7. ANEXOS

7.1 ANEXO 1 – SIP.CONF

```
; SIP Configuration example for Asterisk
;
; Syntax for specifying a SIP device in extensions.conf is
; SIP/devicename where devicename is defined in a section below.
;
; You may also use
; SIP/username@domain to call any SIP user on the Internet
; (Don't forget to enable DNS SRV records if you want to use this)
;
; If you define a SIP proxy as a peer below, you may call
; SIP/proxyhostname/user or SIP/user@proxyhostname
; where the proxyhostname is defined in a section below
;
; Useful CLI commands to check peers/users:
;   sip show peers          Show all SIP peers (including friends)
;   sip show users          Show all SIP users (including friends)
;   sip show registry       Show status of hosts we register with
;
;   sip debug               Show all SIP messages
;
;   reload chan_sip.so              Reload configuration file
;                           Active SIP peers will not be reconfigured
;

[general]
context=default                 ; Default context for incoming calls
;allowguest=no                  ; Allow or reject guest calls (default is yes,
this can also be set to 'osp'
                                ; if asterisk was compiled with OSP support.
;realm=mydomain.tld             ; Realm for digest authentication
                                ; defaults to "asterisk"
                                ; Realms MUST be globally unique according to
RFC 3261
                                ; Set this to your host name or domain name
bindport=5060                   ; UDP Port to bind to (SIP standard port is
5060)
bindaddr=0.0.0.0                ; IP address to bind to (0.0.0.0 binds to all)
srvlookup=yes                   ; Enable DNS SRV lookups on outbound
calls
                                ; Note: Asterisk only uses the first host
```

```
                        ; in SRV records
                        ; Disabling DNS SRV lookups disables the
                        ; ability to place SIP calls based on domain
                        ; names to some other SIP users on the Internet

;domain=mydomain.tld            ; Set default domain for this host
                        ; If configured, Asterisk will only allow
                        ; INVITE and REFER to non-local domains
                        ; Use "sip show domains" to list local domains
;domain=mydomain.tld,mydomain-incoming
                        ; Add domain and configure incoming context
                        ; for external calls to this domain
;domain=1.2.3.4                 ; Add IP address as local domain
                        ; You can have several "domain" settings
;allowexternalinvites=no ; Disable INVITE and REFER to non-local
domains
                        ; Default is yes
;autodomain=yes                 ; Turn this on to have Asterisk add local
host
                        ; name and local IP to domain list.
;pedantic=yes                   ; Enable slow, pedantic checking for
Pingtel
                        ; and multiline formatted headers for strict
                        ; SIP compatibility (defaults to "no")
;tos=184                 ; Set IP QoS to either a keyword or numeric val
;tos=lowdelay                   ;
lowdelay,throughput,reliability,mincost,none
;maxexpiry=3600                 ; Max length of incoming registration we
allow
;defaultexpiry=120      ; Default length of incoming/outoing registration
;notifymimetype=text/plain ; Allow overriding of mime type in MWI NOTIFY
;checkmwi=10                    ; Default time between mailbox checks for
peers
;vmexten=voicemail      ; dialplan extension to reach mailbox sets the
                                ; Message-Account in the MWI
notify message
                                ; defaults to "asterisk"
;videosupport=yes       ; Turn on support for SIP video
;recordhistory=yes      ; Record SIP history by default
                        ; (see sip history / sip no history)

;disallow=all           ; First disallow all codecs
;allow=ulaw             ; Allow codecs in order of preference
;allow=ilbc             ;
;musicclass=default     ; Sets the default music on hold class for all SIP
calls
                        ; This may also be set for individual users/peers
```

```
;language=en                    ; Default language setting for all
users/peers
                                ; This may also be set for individual users/peers
;relaxdtmf=yes                  ; Relax dtmf handling
;rtptimeout=60                  ; Terminate call if 60 seconds of no RTP
activity
                                ; when we're not on hold
;rtpholdtimeout=300             ; Terminate call if 300 seconds of no RTP activity
                                ; when we're on hold (must be > rtptimeout)
;trustrpid = no                 ; If Remote-Party-ID should be trusted
;sendrpid = yes                 ; If Remote-Party-ID should be sent
;progressinband=never           ; If we should generate in-band ringing
always
                                ; use 'never' to never use in-band signalling, even
in cases
                                ; where some buggy devices might not render it
                                ; Valid values: yes, no, never Default: never
;useragent=Asterisk PBX         ; Allows you to change the user agent
string
;promiscredir = no              ; If yes, allows 302 or REDIR to non-local SIP
address
                                ; Note that promiscredir when redirects are made
to the
                        ; local system will cause loops since SIP is incapable
                        ; of performing a "hairpin" call.
;usereqphone = no               ; If yes, ";user=phone" is added to uri that
contains
                                ; a valid phone number
;dtmfmode = rfc2833             ; Set default dtmfmode for sending DTMF.
Default: rfc2833
                                ; Other options:
                                ; info : SIP INFO messages
                                ; inband : Inband audio (requires 64 kbit codec -
alaw, ulaw)
                                ; auto : Use rfc2833 if offered, inband otherwise

;compactheaders = yes           ; send compact sip headers.
;sipdebug = yes                 ; Turn on SIP debugging by default, from
                                ; the moment the channel loads this configuration
;subscribecontext = default     ; Set a specific context for SUBSCRIBE requests
                                ; Useful to limit subscriptions to local extensions
                                ; Settable per peer/user also
;notifyringing = yes            ; Notify subscriptions on RINGING state

;
; If regcontext is specified, Asterisk will dynamically create and destroy a
; NoOp priority 1 extension for a given peer who registers or unregisters with
; us.  The actual extension is the 'regexten' parameter of the registering
```

```
; peer or its name if 'regexten' is not provided.  More than one regexten may
; be supplied if they are separated by '&'.  Patterns may be used in regexten.
;
;regcontext=sipregistrations
;
; Asterisk can register as a SIP user agent to a SIP proxy (provider)
; Format for the register statement is:
;       register => user[:secret[:authuser]]@host[:port][/extension]
;
; If no extension is given, the 's' extension is used. The extension needs to
; be defined in extensions.conf to be able to accept calls from this SIP proxy
; (provider).
;
; host is either a host name defined in DNS or the name of a section defined
; below.
;
; Examples:
;
;register => 1234:password@mysipprovider.com
;
;     This will pass incoming calls to the 's' extension
;
;
;register => 2345:password@sip_proxy/1234
;
;    Register 2345 at sip provider 'sip_proxy'.  Calls from this provider
;    connect to local extension 1234 in extensions.conf, default context,
;    unless you configure a [sip_proxy] section below, and configure a
;    context.
;    Tip 1: Avoid assigning hostname to a sip.conf section like [provider.com]
;    Tip 2: Use separate type=peer and type=user sections for SIP providers
;           (instead of type=friend) if you have calls in both directions

;registertimeout=20          ; retry registration calls every 20 seconds (default)
;registerattempts=10                ; Number of registration attempts before we
give up
                             ; 0 = continue forever, hammering the other
server until it
                             ; accepts the registration
                             ; Default is 0 tries, continue forever
;callevents=no                      ; generate manager events when sip ua
performs events (e.g. hold)

;----------------------------------------- NAT SUPPORT ------------------------
; The externip, externhost and localnet settings are used if you use Asterisk
; behind a NAT device to communicate with services on the outside.
```

```
;externip = 200.201.202.203        ; Address that we're going to put in
outbound SIP messages
                                   ; if we're behind a NAT

                                   ; The externip and localnet is used
                                   ; when registering and communicating with other
proxies
                                   ; that we're registered with
;externhost=foo.dyndns.net         ; Alternatively you can specify an
                                   ; external host, and Asterisk will
                                   ; perform DNS queries periodically.  Not
                                   ; recommended for production
                                   ; environments!  Use externip instead
;externrefresh=10                  ; How often to refresh externhost if
                                   ; used
                                   ; You may add multiple local networks.  A
reasonable set of defaults
                                   ; are:
;localnet=192.168.0.0/255.255.0.0; All RFC 1918 addresses are local
networks
;localnet=10.0.0.0/255.0.0.0       ; Also RFC1918
;localnet=172.16.0.0/12            ; Another RFC1918 with CIDR notation
;localnet=169.254.0.0/255.255.0.0 ;Zero conf local network

; The nat= setting is used when Asterisk is on a public IP, communicating
with
; devices hidden behind a NAT device (broadband router).  If you have one-
way
; audio problems, you usually have problems with your NAT configuration or
your
; firewall's support of SIP+RTP ports.  You configure Asterisk choice of RTP
; ports for incoming audio in rtp.conf
;
;nat=no                            ; Global NAT settings  (Affects all peers
and users)
                        ; yes = Always ignore info and assume NAT
                        ; no = Use NAT mode only according to RFC3581
                        ; never = Never attempt NAT mode or RFC3581 support
                                   ; route = Assume NAT, don't send rport
                                   ; (work around more UNIDEN bugs)

;rtcachefriends=yes                ; Cache realtime friends by adding them to the
internal list
                                   ; just like friends added from the config file only on
a
                                   ; as-needed basis? (yes|no)
```

```
;rtupdate=yes                    ; Send registry updates to database using
realtime? (yes|no)
                                 ; If set to yes, when a SIP UA registers
successfully, the ip address,
                                 ; the origination port, the registration period, and
the username of
                                 ; the UA will be set to database via realtime. If not
present, defaults to 'yes'.

;rtautoclear=yes                 ; Auto-Expire friends created on the fly on the
same schedule
                                 ; as if it had just registered? (yes|no|<seconds>)
                                 ; If set to yes, when the registration expires, the
friend will vanish from
                                 ; the configuration until requested again. If set to
an integer,
                                 ; friends expire within this number of seconds
instead of the
                                 ; registration interval.

;ignoreregexpire=yes             ; Enabling this setting has two functions:
                                 ;
                                 ; For non-realtime peers, when their registration
expires, the information
                                 ; will _not_ be removed from memory or the
Asterisk database; if you attempt
                                 ; to place a call to the peer, the existing
information will be used in spite
                                 ; of it having expired
                                 ;
                                 ; For realtime peers, when the peer is retrieved
from realtime storage,
                                 ; the registration information will be used
regardless of whether
                                 ; it has expired or not; if it expires while the
realtime peer is still in
                                 ; memory (due to caching or other reasons), the
information will not be
                                 ; removed from realtime storage

; Incoming INVITE and REFER messages can be matched against a list of
'allowed'
; domains, each of which can direct the call to a specific context if desired.
; By default, all domains are accepted and sent to the default context or the
; context associated with the user/peer placing the call.
; Domains can be specified using:
; domain=<domain>[,<context>]
; Examples:
```

```
; domain=myasterisk.dom
; domain=customer.com,customer-context
;
; In addition, all the 'default' domains associated with a server should be
; added if incoming request filtering is desired.
; autodomain=yes
;
; To disallow requests for domains not serviced by this server:
; allowexternaldomains=no

; fromdomain=mydomain.tld ; When making outbound SIP INVITEs to
                          ; non-peers, use your primary domain "identity"
                          ; for From: headers instead of just your IP
                          ; address. This is to be polite and
                          ; it may be a mandatory requirement for some
                          ; destinations which do not have a prior
                          ; account relationship with your server.

[authentication]
; Global credentials for outbound calls, i.e. when a proxy challenges your
; Asterisk server for authentication. These credentials override
; any credentials in peer/register definition if realm is matched.
;
; This way, Asterisk can authenticate for outbound calls to other
; realms. We match realm on the proxy challenge and pick an set of
; credentials from this list
; Syntax:
;         auth = <user>:<secret>@<realm>
;         auth = <user>#<md5secret>@<realm>
; Example:
;auth=mark:topsecret@digium.com
;
; You may also add auth= statements to [peer] definitions
; Peer auth= override all other authentication settings if we match on realm

;------------------------------------------------------------------------------
; Users and peers have different settings available. Friends have all settings,
; since a friend is both a peer and a user
;
; User config options:       Peer configuration:
; --------------------       -------------------
; context                     context
; permit                     permit
; deny                       deny
; secret                     secret
; md5secret                     md5secret
; dtmfmode                      dtmfmode
; canreinvite                  canreinvite
```

```
; nat                    nat
; callgroup               callgroup
; pickupgroup             pickupgroup
; language                language
; allow                   allow
; disallow                disallow
; insecure                insecure
; trustrpid               trustrpid
; progressinband          progressinband
; promiscredir            promiscredir
; useclientcode           useclientcode
; accountcode             accountcode
; setvar                  setvar
; callerid                callerid
; amaflags                amaflags
; call-limit              call-limit
; restrictcid             restrictcid
; subscribecontext        subscribecontext
;                         mailbox
;                         username
;                         template
;                         fromdomain
;                         regexten
;                         fromuser
;                         host
;                         port
;                         qualify
;                         defaultip
;                         rtptimeout
;                         rtpholdtimeout
;                         sendrpid

;[sip_proxy]
; For incoming calls only. Example: FWD (Free World Dialup)
; We match on IP address of the proxy for incoming calls
; since we can not match on username (caller id)
;type=peer
;context=from-fwd
;host=fwd.pulver.com

;[sip_proxy-out]
;type=peer                  ; we only want to call out, not be called
;secret=guessit
;username=yourusername      ; Authentication user for outbound proxies
;fromuser=yourusername      ; Many SIP providers require this!
;fromdomain=provider.sip.domain
;host=box.provider.com
;usereqphone=yes            ; This provider requires ";user=phone" on URI
```

```
;call-limit=5                   ; permit only 5 simultaneous outgoing calls to this
peer

;---------------------------------------------------------------------------
; Definitions of locally connected SIP phones
;
; type = user  a device that authenticates to us by "from" field to place calls
; type = peer  a device we place calls to or that calls us and we match by host
; type = friend two configurations (peer+user) in one
;
; For local phones, type=friend works most of the time
;
; If you have one-way audio, you propably have NAT problems.
; If Asterisk is on a public IP, and the phone is inside of a NAT device
; you will need to configure nat option for those phones.
; Also, turn on qualify=yes to keep the nat session open

;[grandstream1]
;type=friend
;context=from-sip           ; Where to start in the dialplan when this phone
calls
;callerid=John Doe <1234>; Full caller ID, to override the phones config
;host=192.168.0.23          ; we have a static but private IP address
                            ; No registration allowed
;nat=no                             ; there is not NAT between phone and
Asterisk
;canreinvite=yes            ; allow RTP voice traffic to bypass Asterisk
;dtmfmode=info                      ; either RFC2833 or INFO for the
BudgeTone
;call-limit=1               ; permit only 1 outgoing call and 1 incoming call at
a time
                            ; from the phone to asterisk
                            ; (1 for the explicit peer, 1 for the explicit user,
                            ; remember that a friend equals 1 peer and 1 user
in
                            ; memory)
;mailbox=1234@default               ; mailbox 1234 in voicemail context
"default"
;disallow=all               ; need to disallow=all before we can use allow=
;allow=ulaw                 ; Note: In user sections the order of codecs
                            ; listed with allow= does NOT matter!
;allow=alaw
;allow=g723.1                       ; Asterisk only supports g723.1 pass-thru!
;allow=g729                 ; Pass-thru only unless g729 license obtained
;astdb=chan2ext/SIP/grandstream1=1234        ; ensures an astDB entry
exists
```

```
;[xlite1]
; Turn off silence suppression in X-Lite ("Transmit Silence"=YES)!
; Note that Xlite sends NAT keep-alive packets, so qualify=yes is not needed
;type=friend
;regexten=1234                  ; When they register, create extension
1234
;callerid="Jane Smith" <5678>
;host=dynamic                   ; This device needs to register
;nat=yes                ; X-Lite is behind a NAT router
;canreinvite=no                 ; Typically set to NO if behind NAT
;disallow=all
;allow=gsm              ; GSM consumes far less bandwidth than ulaw
;allow=ulaw
;allow=alaw
;mailbox=1234@default,1233@default   ; Subscribe to status of multiple
mailboxes


;[snom]
;type=friend            ; Friends place calls and receive calls
;context=from-sip       ; Context for incoming calls from this user
;secret=blah
;subscribecontext=localextensions       ; Only allow SUBSCRIBE for local
extensions
;language=de                    ; Use German prompts for this user
;host=dynamic                   ; This peer register with us
;dtmfmode=inband        ; Choices are inband, rfc2833, or info
;defaultip=192.168.0.59         ; IP used until peer registers
;mailbox=1234@context,2345      ; Mailbox(-es) for message waiting indicator
;vmexten=voicemail      ; dialplan extension to reach mailbox
                ; sets the Message-Account in the MWI notify message
                ; defaults to global vmexten which defaults to "asterisk"
;restrictcid=yes        ; To have the callerid restriced -> sent as ANI
;disallow=all
;allow=ulaw             ; dtmfmode=inband only works with ulaw or alaw!


;[polycom]
;type=friend            ; Friends place calls and receive calls
;context=from-sip       ; Context for incoming calls from this user
;secret=blahpoly
;host=dynamic                   ; This peer register with us
;dtmfmode=rfc2833       ; Choices are inband, rfc2833, or info
;username=polly                 ; Username to use in INVITE until peer
registers
                        ; Normally you do NOT need to set this parameter
;disallow=all
;allow=ulaw             ; dtmfmode=inband only works with ulaw or alaw!
```

```
;progressinband=no         ; Polycom phones don't work properly with "never"


;[pingtel]
;type=friend
;secret=blah
;host=dynamic
;insecure=port                    ; Allow matching of peer by IP address
without matching port number
;insecure=invite          ; Do not require authentication of incoming
INVITEs
;insecure=port,invite     ; (both)
;qualify=1000             ; Consider it down if it's 1 second to reply
                          ; Helps with NAT session
                          ; qualify=yes uses default value
;callgroup=1,3-4          ; We are in caller groups 1,3,4
;pickupgroup=1,3-5        ; We can do call pick-p for call group 1,3,4,5
;defaultip=192.168.0.60          ; IP address to use if peer has not registred

;[cisco1]
;type=friend
;secret=blah
;qualify=200              ; Qualify peer is no more than 200ms away
;nat=yes                  ; This phone may be natted
                          ; Send SIP and RTP to the IP address that packet
is
                          ; received from instead of trusting SIP headers
;host=dynamic                    ; This device registers with us
;canreinvite=no                  ; Asterisk by default tries to redirect the
                          ; RTP media stream (audio) to go directly from
                          ; the caller to the callee.  Some devices do not
                          ; support this (especially if one of them is
                          ; behind a NAT).
;defaultip=192.168.0.4           ; IP address to use until registration
;username=goran                  ; Username to use when calling this device
before registration
                          ; Normally you do NOT need to set this parameter
;setvar=CUSTID=5678              ; Channel variable to be set for all calls
from this device
```

7.2 ANEXO 2 – EXTENSIONS.CONF

```
; Static extension configuration file, used by
; the pbx_config module. This is where you configure all your
; inbound and outbound calls in Asterisk.
;
; This configuration file is reloaded
; - With the "extensions reload" command in the CLI
; - With the "reload" command (that reloads everything) in the CLI

;
; The "General" category is for certain variables.
;
[general]
;
; If static is set to no, or omitted, then the pbx_config will rewrite
; this file when extensions are modified.  Remember that all comments
; made in the file will be lost when that happens.
;
; XXX Not yet implemented XXX
;
static=yes
;
; if static=yes and writeprotect=no, you can save dialplan by
; CLI command 'save dialplan' too
;
writeprotect=no

; You can include other config files, use the #include command (without the
';')
; Note that this is different from the "include" command that includes contexts
within
; other contexts. The #include command works in all Asterisk configuration
files.
;#include "filename.conf"

; The "Globals" category contains global variables that can be referenced
; in the dialplan with ${VARIABLE} or ${ENV(VARIABLE)} for Environmental
variable
; ${${VARIABLE}} or ${text${VARIABLE}} or any hybrid
;
[globals]
CONSOLE=Console/dsp                     ; Console interface for demo
;CONSOLE=Zap/1
;CONSOLE=Phone/phone0
IAXINFO=guest                           ; IAXtel username/password
```

```
;IAXINFO=myuser:mypass
TRUNK=Zap/g2                    ; Trunk interface
;
; Note the 'g2' in the TRUNK variable above. It specifies which group (defined
; in zapata.conf) to dial, i.e. group 2, and how to choose a channel to use in
; the specified group. The four possible options are:
;
; g: select the lowest-numbered non-busy Zap channel (aka. ascending
sequential hunt group).
; G: select the highest-numbered non-busy Zap channel (aka. descending
sequential hunt group).
; r: use a round-robin search, starting at the next highest channel than last
time (aka. ascending rotary hunt group).
; R: use a round-robin search, starting at the next lowest channel than last
time (aka. descending rotary hunt group).
;
TRUNKMSD=1                      ; MSD digits to strip (usually 1 or 0)
;TRUNK=IAX2/user:pass@provider

;
; Any category other than "General" and "Globals" represent
; extension contexts, which are collections of extensions.
;
; Extension names may be numbers, letters, or combinations
; thereof. If an extension name is prefixed by a '_'
; character, it is interpreted as a pattern rather than a
; literal.  In patterns, some characters have special meanings:
;
;   X - any digit from 0-9
;   Z - any digit from 1-9
;   N - any digit from 2-9
;   [1235-9] - any digit in the brackets (in this example, 1,2,3,5,6,7,8,9)
;   . - wildcard, matches anything remaining (e.g. _9011. matches
;       anything starting with 9011 excluding 9011 itself)
;
; For example the extension _NXXXXXX would match normal 7 digit dialings,
; while _1NXXNXXXXXX would represent an area code plus phone number
; preceeded by a one.
;
; Each step of an extension is ordered by priority, which must
; always start with 1 to be considered a valid extension.
;
; Contexts contain several lines, one for each step of each
; extension, which can take one of two forms as listed below,
; with the first form being preferred.  One may include another
; context in the current one as well, optionally with a
; date and time.  Included contexts are included in the order
; they are listed.
```

```
;
;[context]
;exten => someexten,priority,application(arg1,arg2,...)
;exten => someexten,priority,application,arg1|arg2...
;
; Timing list for includes is
;
;   <time range>|<days of week>|<days of month>|<months>
;
;include => daytime|9:00-17:00|mon-fri|*|*
;
; ignorepat can be used to instruct drivers to not cancel dialtone upon
; receipt of a particular pattern.  The most commonly used example is
; of course '9' like this:
;
;ignorepat => 9
;
; so that dialtone remains even after dialing a 9.
;

;
; Here are the entries you need to participate in the IAXTEL
; call routing system.  Most IAXTEL numbers begin with 1-700, but
; there are exceptions.  For more information, and to sign
; up, please go to www.gnophone.com or www.iaxtel.com
;
[iaxtel700]
exten                                                                    =>
_91700XXXXXXX,1,Dial(IAX2/${IAXINFO}@iaxtel.com/${EXTEN:1}@iaxtel)

;
; The SWITCH statement permits a server to share the dialplain with
; another server. Use with care: Reciprocal switch statements are not
; allowed (e.g. both A -> B and B -> A), and the switched server needs
; to be on-line or else dialing can be severly delayed.
;
[iaxprovider]
;switch => IAX2/user:[key]@myserver/mycontext

[trunkint]
;
; International long distance through trunk
;
exten => _9011.,1,Dial(${TRUNK}/${EXTEN:${TRUNKMSD}})
exten => _9011.,2,Congestion
[trunkld]
;
; Long distance context accessed through trunk
```

```
;
exten => _91NXXNXXXXXX,1,Dial(${TRUNK}/${EXTEN:${TRUNKMSD}})
exten => _91NXXNXXXXXX,2,Congestion

[trunklocal]
;
; Local seven-digit dialing accessed through trunk interface
;
exten => _9NXXXXXX,1,Dial(${TRUNK}/${EXTEN:${TRUNKMSD}})
exten => _9NXXXXXX,2,Congestion

[trunktollfree]
;
; Long distance context accessed through trunk interface
;
exten => _91800NXXXXXX,1,Dial(${TRUNK}/${EXTEN:${TRUNKMSD}})
exten => _91800NXXXXXX,2,Congestion
exten => _91888NXXXXXX,1,Dial(${TRUNK}/${EXTEN:${TRUNKMSD}})
exten => _91888NXXXXXX,2,Congestion
exten => _91877NXXXXXX,1,Dial(${TRUNK}/${EXTEN:${TRUNKMSD}})
exten => _91877NXXXXXX,2,Congestion
exten => _91866NXXXXXX,1,Dial(${TRUNK}/${EXTEN:${TRUNKMSD}})
exten => _91866NXXXXXX,2,Congestion

[international]
;
; Master context for international long distance
;
ignorepat => 9
include => longdistance
include => trunkint

[longdistance]
;
; Master context for long distance
;
ignorepat => 9
include => local
include => trunkld

[local]
;
; Master context for local, toll-free, and iaxtel calls only
;
ignorepat => 9
include => default
include => parkedcalls
include => trunklocal
```

```
include => iaxtel700
include => trunktollfree
include => iaxprovider
;
; You can use an alternative switch type as well, to resolve
; extensions that are not known here, for example with remote
; IAX switching you transparently get access to the remote
; Asterisk PBX
;
; switch => IAX2/user:password@bigserver/local

[macro-stdexten];
;
; Standard extension macro:
;   ${ARG1} - Extension  (we could have used ${MACRO_EXTEN} here as
well
;   ${ARG2} - Device(s) to ring
;
exten => s,1,Dial(${ARG2},20)                          ; Ring the interface, 20
seconds maximum
exten => s,2,Goto(s-${DIALSTATUS},1)                   ; Jump based on
status (NOANSWER,BUSY,CHANUNAVAIL,CONGESTION,ANSWER)

exten => s-NOANSWER,1,Voicemail(u${ARG1})              ; If unavailable,
send to voicemail w/ unavail announce
exten => s-NOANSWER,2,Goto(default,s,1)                ; If they press #, return
to start

exten => s-BUSY,1,Voicemail(b${ARG1})                  ; If busy, send to
voicemail w/ busy announce
exten => s-BUSY,2,Goto(default,s,1)                    ; If they press #,
return to start

exten => _s-.,1,Goto(s-NOANSWER,1)                     ; Treat anything
else as no answer

exten => a,1,VoicemailMain(${ARG1})                    ; If they press *,
send the user into VoicemailMain

[demo]
;
; We start with what to do when a call first comes in.
;
exten => s,1,Wait,1                ; Wait a second, just for fun
exten => s,2,Answer                ; Answer the line
exten => s,3,DigitTimeout,5        ; Set Digit Timeout to 5 seconds
exten => s,4,ResponseTimeout,10    ; Set Response Timeout to 10
seconds
```

```
exten => s,5,BackGround(demo-congrats)  ; Play a congratulatory message
exten => s,6,BackGround(demo-instruct)  ; Play some instructions

exten => 2,1,BackGround(demo-moreinfo)  ; Give some more information.
exten => 2,2,Goto(s,6)

exten => 3,1,SetLanguage(fr)            ; Set language to french
exten => 3,2,Goto(s,5)                  ; Start with the congratulations

exten => 1000,1,Goto(default,s,1)
;
; We also create an example user, 1234, who is on the console and has
; voicemail, etc.
;
exten => 1234,1,Playback(transfer,skip)         ; "Please hold while..."
                                ; (but skip if channel is not up)
exten => 1234,2,Macro(stdexten,1234,${CONSOLE})

exten => 1235,1,Voicemail(u1234)                ; Right to voicemail

exten => 1236,1,Dial(Console/dsp)               ; Ring forever
exten => 1236,2,Voicemail(u1234)                ; Unless busy

;
; # for when they're done with the demo
;
exten => #,1,Playback(demo-thanks)              ; "Thanks for trying the demo"
exten => #,2,Hangup                     ; Hang them up.

;
; A timeout and "invalid extension rule"
;
exten => t,1,Goto(#,1)                  ; If they take too long, give up
exten => i,1,Playback(invalid)          ; "That's not valid, try again"

;
; Create an extension, 500, for dialing the
; Asterisk demo.
;
exten => 500,1,Playback(demo-abouttotry); Let them know what's going on
exten => 500,2,Dial(IAX2/guest@misery.digium.com/s@default)      ; Call the
Asterisk demo
exten => 500,3,Playback(demo-nogo)      ; Couldn't connect to the demo site
exten => 500,4,Goto(s,6)                ; Return to the start over message.

;
; Create an extension, 600, for evaulating echo latency.
;
```

```
exten => 600,1,Playback(demo-echotest)  ; Let them know what's going on
exten => 600,2,Echo                     ; Do the echo test
exten => 600,3,Playback(demo-echodone)  ; Let them know it's over
exten => 600,4,Goto(s,6)                ; Start over

;
; Give voicemail at extension 8500
;
exten => 8500,1,VoicemailMain
exten => 8500,2,Goto(s,6)
;
; Here's what a phone entry would look like (IXJ for example)
;
;exten => 1265,1,Dial(Phone/phone0,15)
;exten => 1265,2,Goto(s,5)

;[mainmenu]
;
; Example "main menu" context with submenu
;
;exten => s,1,Answer
;exten => s,2,Background(thanks)                 ; "Thanks for calling press 1 for
sales, 2 for support, ..."
;exten => 1,1,Goto(submenu,s,1)
;exten => 2,1,Hangup
;include => default
;
;[submenu]
;exten => s,1,Ringing                            ; Make them comfortable with 2
seconds of ringback
;exten => s,2,Wait,2
;exten => s,3,Background(submenuopts)     ; "Thanks for calling the sales
department.  Press 1 for steve, 2 for..."
;exten => 1,1,Goto(default,steve,1)
;exten => 2,1,Goto(default,mark,2)

[default]
;
; By default we include the demo.  In a production system, you
; probably don't want to have the demo there.
;
include => demo
;
; Extensions like the two below can be used for FWD, Nikotel, sipgate etc.
; Note that you must have a [sipprovider] section in sip.conf whereas
; the otherprovider.net example does not require such a peer definition
;
;exten => _41X.,1,Dial(SIP/${EXTEN:2}@sipprovider,,r)
```

```
;exten                                                                    =>
_42X.,1,Dial(SIP/user:passwd@${EXTEN:2}@otherprovider.net,30,rT)

; Real extensions would go here. Generally you want real extensions to be 4
or 5
; digits long (although there is no such requirement) and start with a single
; digit that is fairly large (like 6 or 7) so that you have plenty of room to
; overlap extensions and menu options without conflict.  You can alias them
with
; names, too and use global variables

;exten => 6245,hint,SIP/Grandstream1&SIP/Xlite1 ; Channel hints for
presence
;exten => 6245,1,Dial(SIP/Grandstream1,20,rt)   ; permit transfer
;exten => 6245,1,Dial(${HINT},20,rtT)           ; Use hint as listed
;exten => 6361,1,Dial(IAX2/JaneDoe,,rm)         ; ring without time limit
;exten => 6389,1,Dial(MGCP/aaln/1@192.168.0.14)
;exten => 6394,1,Dial(Local/6275/n)             ; this will dial ${MARK}
exten=>4102,1,Dial(SIP/4102,20)
exten=>4103,1,Dial(SIP/4103,20)


;exten => 6275,1,Macro(stdexten,6275,${MARK})    ; assuming ${MARK} is
something like Zap/2
;exten => mark,1,Goto(6275|1)                   ; alias mark to 6275
;exten => 6536,1,Macro(stdexten,6236,${WIL})    ; Ditto for wil
;exten => wil,1,Goto(6236|1)
;
; Some other handy things are an extension for checking voicemail via
; voicemailmain
;
;exten => 8500,1,VoicemailMain
;exten => 8500,2,Hangup
;
; Or a conference room (you'll need to edit meetme.conf to enable this room)
;
;exten => 8600,1,Meetme(1234)
;
; Or playing an announcement to the called party, as soon it answers
;
;exten = 8700,1,Dial(${MARK},30,A(/path/to/my/announcemsg))
;
;
; For more information on applications, just type "show applications" at your
; friendly Asterisk CLI prompt.
;
; 'show application <command>' will show details of how you
; use that particular application in this file, the dial plan.
```

7.3 ANEXO 3 – FEATURES.CONF

```
; Sample Parking configuration
;

[general]
parkext => 700                  ; What extension to dial to park
parkpos => 701-720          ; What extensions to park calls on. These needs
to be
                                ; numeric, as Asterisk starts from the start position
                                ; and increments with one for the next parked call.
context => parkedcalls          ; Which context parked calls are in
;parkingtime => 45          ; Number of seconds a call can be parked for
                                ; (default is 45 seconds)
;transferdigittimeout => 3  ; Number of seconds to wait between digits when
transfering a call
;courtesytone = beep            ; Sound file to play to the parked caller
                                ; when someone dials a parked call
;xfersound = beep           ; to indicate an attended transfer is complete
;xferfailsound = beeperr    ; to indicate a failed transfer
;adsipark = yes                 ; if you want ADSI parking announcements
;findslot => next           ; Continue to the 'next' free parking space.
                                ; Defaults to 'first' available
;pickupexten = *8           ; Configure the pickup extension.  Default is *8
;featuredigittimeout = 500  ; Max time (ms) between digits for
                                ; feature activation.  Default is 500


[featuremap]
;blindxfer => #1            ; Blind transfer
;disconnect => *0           ; Disconnect
;automon => *1                  ; One Touch Record
;atxfer => *2               ; Attended transfer

[applicationmap]
; Note that the DYNAMIC_FEATURES channel variable must be set to use
the features
; defined here.  The value of DYNAMIC_FEATURES should be the names of
the features
; to allow the channel to use separated by '#'.  For example:
;    Set(DYNAMIC_FEATURES=myfeature1#myfeature2#myfeature3)
;
;testfeature => #9,callee,Playback,tt-monkeys   ;Play tt-monkeys to
                                                ;callee if #9 was pressed
```