UNICEUB – CENTRO UNIVERSITÁRIO DE BRASÍLIA
FAET – FACULDADE DE CIÊNCIAS EXATAS E TECNOLOGIA
CURSO DE ENGENHARIA DA COMPUTAÇÃO

DANILO AUGUSTO VICENTE LARA

IPBX UTILIZANDO SOFTWARE LIVRE ASTERISK

Brasília
2007

DANILO AUGUSTO VICENTE LARA

IPBX UTILIZANDO SOFTWARE LIVRE ASTERISK

Trabalho apresentado à banca examinadora da Faculdade de Ciências Exatas e Tecnologia, para conclusão do curso de Engenharia da Computação. Professor orientador: M.Sc. Antônio José Gonçalves Pinto.

Brasília
2007

DANILO AUGUSTO VICENTE LARA

IPBX UTILIZANDO SOFTWARE LIVRE ASTERISK

Trabalho apresentado à banca examinadora da Faculdade de Ciências Exatas e Tecnologia, para conclusão do curso de Engenharia da Computação. Professor orientador: M.Sc. Antônio José Gonçalves Pinto.

Brasília, 05 de Dezembro de 2007

Banca Examinadora

_____

Professor orientador M.Sc. Antônio José Gonçalves Pinto

_____

Professor M.Sc. Roberto Schaefer de Azeredo

_____

Professor M.Sc. Miguel Archanjo Júnior

Aos meus pais, familiares, namorada e amigos. Em especial a minha amada mamãe, a quem devo tudo que sou.

AGRADECIMENTOS

Aos meus pais, Iolanda e Augustinho, por todo apoio e incentivo recebidos. Em especial a minha mamãe, por toda ajuda e amor incondicionais dedicados, a quem devo mais esta conquista.

A minha namorada, Bruna, por todo apoio, compreensão e dedicação durante todo o curso.

Ao meu amigo Helder, por toda ajuda, apoio com materiais de pesquisa e colaboração recebidos.

Aos meus amigos do curso que me acompanharam durante a jornada acadêmica, sempre presentes e ajudando em todos os momentos.

Aos meus colegas de trabalho, por todo apoio, críticas construtivas e incentivo recebidos.

Ao meu professor orientador, M.Sc. Antônio José Gonçalves Pinto, por todo conhecimento transmitido, incentivo, paciência e por sua orientação sempre precisa me direcionando sempre para o caminho correto.

A todo o corpo docente do UniCEUB, pelo apoio e conhecimento transmitidos.

"Tell me, and I forget, Teach
me, and I may remember,
Involve me, and I learn."
(Benjamim Franklin).

RESUMO

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

A tecnologia VoIP vêm crescendo em um ritmo avançado, e com isto o crescimento de produtos e serviços oferecidos nesta área também aumentaram bastante. Tendo em vista este crescimento, este projeto tem como finalidade implementar um IPBX (*Internet Protocol Branch Exchange*) utilizando o software livre *Asterisk* que permita a comunicação com a rede pública de telefonia comutada e com telefones analógicos.

Neste projeto serão utilizados além do software livre *Asterisk*, o protocolo SIP e um adaptador de telefone analógico (ATA – *Analog Telephone Adapter*) para transformar um telefone analógico comum em um telefone IP. Além disso, serão utilizadas técnicas de programação para criar uma interface WEB de administração do *Asterisk*, permitindo facilitar o seu gerenciamento.

Palavras Chave: Asterisk, VoIP, SIP, ATA, IPBX, PSTN, Ruby on Rails, Flex, Soft phone.

ABSTRACT

The VoIP technology provides voice and data transmission on the same services network. This technology works with packet switching, different from the conventional telephony which works with circuit switching.

The VoIP technology grews up and continue growing quickly, so the offering of products and services based on this technology increased a lot. In despite of this increase, this project aims to implement an IPBX (Internet Protocol Branch Exchange) using the free software Asterisk to provide communication with the public switched telephone network.

On this project, beyond the free software Asterisk, will be used the SIP protocol and an analog telephone adapter (ATA) to convert an analog telephone in an IP telephone. More than this will be used programming techniques to create a WEB interface to manage the Asterisk, so making its management easy.

Keywords: Asterisk, VoIP, SIP, ATA, IPBX, PSTN, Ruby on Rails, Flex, Soft phone.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

VoIP – Voice over Internet Protocol

PABX – Private Automatic Branch Exchange

PBX – Private Branch Exchange

IPBX – Internet Protocol Private Branch Exchange

PSTN – Public Switched Telephone Network

CDR – Call Detail Records

ATA – Analog Telephone Adapter

WEB – World Wide Web

IP – Internet Protocol

ITU – International Telecommunications Union

ATT – American Telephone and Telegraph Company

EUA – Estados Unidos da América

CTB – Companhia Telefônica Brasileira

RTPC – Rede de Telefonia Pública Comutada

PMBX – Private Manual Branch Exchange

PAX – Private Automatic Exchange

TCP/IP – Transmission Control Protocol/Internet Protocol

TCP - Transmission Control Protocol

UDP – User Datagram Protocol

NAT – Network Address Translation

PAT – Port Address Translation

IPV4 – Internet Protocol Version 4

RFC – Request for Comments

RTP – Realtime Protocol

QoS – Quality of Service

SIP – Session Initiation Protocol

GUI – Graphical User Interface

DG – Distribuidor Geral

HTTP – Hypertext Transfer Protocol

SMTP – Simple Mail Transfer Protocol

IETF – Internet Engineering Task Force

DoS – Denial of Service

TLS – Transport Layer Security

GNU – Gnu´s Not Unix

DSP – Digital Signal Processor

ISDN – Integrated Service Digital Network

I/O – Input/Output

BSD – Berkeley Software Distribution

IVR – Interactive Voice Response

CERT – Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil

FXO – Foreign Exchange Office

FXS – Foreign Exchange Station

R2 – Sinalização R2

CSV – Comma-Separated Value

SQL – Structured Query Language

DDD- Discagem Direta a Distância

DDI – Discagem Direta Internacional

URA – Unidade de Resposta Audível

RIA – Rich Internet Applications

IAX – Inter-Asterisk Exchange

SUMÁRIO

1 INTRODUÇÃO

1.1 Motivação

"Eu acredito que nos próximos três anos, VoIP usando soluções *Open Source*, como o *Asterisk*, irão gerar mais negócios que todo o mercado Linux de hoje. Hoje as soluções PABX são incrivelmente caras, fechadas e proprietárias. O *Asterisk* é aproximadamente um décimo do preço de um PABX proprietário."[John Hall,2004]

A frase acima de John Hall mostra o futuro promissor do *Asterisk* e do VoIP. Essa foi a principal motivação para a realização deste trabalho. Além disso, há também a possibilidade de utilização de software livre, o que permite um conhecimento e estudo mais aprofundado do sistema, assim como uma possibilidade maior de customização da solução. A questão do custo da solução foi levada em conta, o que colaborou para a utilização do software livre *Asterisk*.

Outro item muito importante que serviu como motivação para desenvolver este trabalho foi a não dependência de empresas de manutenção de PABXs proprietários. Utilizando o *Asterisk* como um PABX, ou IPBX, toda a administração e manutenção do sistema podem ser feitas por um profissional diretamente no servidor, desde que este profissional conheça como funciona o software *Asterisk*, que é aberto e qualquer pessoa pode estudá-lo. A utilização de uma solução não-proprietária permite um maior controle sobre o sistema, já que qualquer intervenção pode ser feita a qualquer momento sem ter que esperar um técnico específico, como ocorre quando é utilizada uma solução proprietária.

1.2 Objetivos

O principal objetivo deste trabalho é facilitar a administração do *Asterisk*. A administração do *Asterisk* é feita através de arquivos texto onde cada parâmetro deve ser configurado. Essa administração é feita, por padrão, no terminal do servidor *Asterisk*, o que causa certa resistência por parte de muitos usuários que acham difícil essa administração. Pensando em uma solução para isto surgiu a idéia e motivação para este projeto, onde as tarefas de adição de ramais e consulta às informações das

ligações efetuadas e recebidas serão simplificadas e não serão mais efetuadas através do terminal do servidor, mas sim através de um navegador de internet.

Outro objetivo é mostrar a comunicação com a rede de telefonia pública através do *Asterisk* usando um soft phone (telefone via software), ou seja, efetuar e receber ligações através do servidor *Asterisk*.

Outro objetivo é mostrar a comunicação com um telefone analógico através de um *soft phone*. Para essa tarefa será utilizado um dispositivo conhecido como ATA (*Analog Telephone Adapter*) para prover a comunicação entre o telefone analógico e o servidor *Asterisk*.

Um objetivo secundário é estudar o software livre *Asterisk* e a tecnologia VoIP. Aprofundar os conhecimentos dos protocolos utilizados na tecnologia VoIP, assim como suas aplicações e suporte pelo *Asterisk*.

1.3 Estrutura do Trabalho

Esse trabalho está organizado em seis capítulos. Os primeiros dois capítulos fazem a apresentação do tema do projeto, fornecendo o embasamento teórico e a tecnologia utilizada para a implementação. 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 deste trabalho pode ser detalhada da seguinte forma:

O primeiro capítulo explica a motivação para a realização deste trabalho, assim como seus objetivos.

O segundo capítulo explica o sistema de telefonia convencional, mostrando seu funcionamento e histórico, e explica o que é a transmissão de voz sobre IP e as vantagens da mesma sobre a telefonia convencional.

O terceiro capítulo trata da implementação do projeto, quais tecnologias foram utilizadas e o porquê da utilização de cada uma, assim como a instalação e configuração das partes do projeto.

O quarto capítulo mostra o funcionamento do projeto, onde a adição dos ramais será efetuada via interface WEB e também a consulta de informações das ligações efetuadas e recebidas. Este capítulo mostra quais equipamentos foram utilizados para efetuar e receber ligações.

O quinto capítulo apresenta a conclusão do projeto, comentando sobre sua viabilidade e utilidade na área da telefonia IP e da telefonia convencional.

2 TELEFONIA E VOZ SOBRE IP

2.1 Histórico da Telefonia

Em 1844, Samuel Morse enviou sua primeira mensagem usando seu sistema de telegrafia entre Washington e Baltimore. Aproximadamente dez anos depois, a telegrafia já era disponível em vários países como um serviço para o público geral. Contudo, passaram-se cerca de vinte anos até que se tornasse possível, supostamente por acaso, a conversão de sinais de voz em sinais elétricos para transmissão. (COLCHER, 2005)

Em 1875, enquanto o cientista Alexander Graham Bell e seu jovem ajudante Thomas A. Watson se dedicavam a um projeto relacionado ao sistema de telegrafia e sem, a princípio, qualquer relação com o telefone, estranhamente o aparato experimental no qual trabalhavam transmitiu um som totalmente diferente do esperado. Analisando o que havia ocorrido, Bell percebeu que, devido à forma com que uma parte do equipamento de recepção havia sido montada naquela ocasião, ele conseguia produzir uma corrente elétrica cuja variação acontecia na mesma intensidade que o ar variava de densidade junto ao transmissor. A descoberta permitiu que, a partir de vários refinamentos, em 14 de fevereiro de 1876, Bell submetesse sua patente do telefone, descrevendo seu aparato como "... o aparelho para transmitir voz e outros sons (...) pelas variações da corrente elétrica, similares às variações do ar, acompanhando cada palavra pronunciada...". Em 1877, Graham Bell fundaria a primeira companhia Bell de telefonia. (COLCHER, 2005)

Nesse ínterim, surgia a primeira organização regulatória internacional de telecomunicações, criada inicialmente para tratar questões de interoperabilidade entre os sistemas de telegrafia adotados em diferentes países. Essa organização veio posteriormente a se tornar o ITU (*International Telecommunications Union*). Atualmente, o setor T do ITU (*Telecom standardization*) é responsável pela padronização técnica e de operação de sistemas de telecomunicações, e os padrões por ele definidos são conhecidos, no jargão ITU, como Recomendações ITU-T. (COLCHER, 2005).

Bell e Watson aperfeiçoaram o sistema e já em 1878 aparecia a primeira rede pública comercial de Telefonia, em *New Haven*, EUA. Em 1885 foi formada a ATT –

American Telephone and Telegraph Company – que até 1984 exerceu o monopólio da telefonia nos Estados Unidos (sendo desmembrada por ordem judicial anti-truste), servindo de modelo administrativo e fonte de padrões técnicos para o resto do mundo.

2.2 Um pouco de história (do Brasil)

Dom Pedro II teve um papel significativo para a promoção definitiva do Invento de Bell. Em 1877 foi realizada a Exposição do Centenário, na Filadélfia, que comemorava os 100 anos da independência dos Estados Unidos. Na data prevista para a apresentação dos trabalhos expostos, 25 de junho de 1877, uma comissão da qual Dom Pedro II fazia parte passou a tarde assistindo a demonstrações. Quando, ao final, chegaram ao local onde Bell expunha seu invento, Dom Pedro, apesar da resistência dos demais membros da comissão, insistiu em experimentar o aparelho. Enquanto Bell permanecia na ponta de transmissão do fio, a 150 metros de distância, Dom Pedro começava a escutar nitidamente sua voz declamando Shakespeare: "*To be or not to be*...". Ele teria então pronunciado as palavras que entrariam para a história: "Meu Deus, isto fala!". A repercussão do invento ganharia então as manchetes dos jornais. (COLCHER, 2005)

O Brasil acompanhou de perto o desenvolvimento da Telefonia (mas não conseguiu sua ampla divulgação, com média de poucos aparelhos por pessoa), instalando seu primeiro aparelho telefônico (interno) no Palácio São Cristóvão, Rio de Janeiro, em 1878. Logo em seguida, 15 de Novembro de 1879, o decreto imperial 7.539 criava a CTB – Companhia Telefônica Brasileira. Com o tempo outras concessionárias foram autorizadas, e finalmente criou-se a Telebrás – monopólio estatal – para controlar todo o sistema de telecomunicações brasileiro.

Em 1998 o monopólio estatal foi privatizado, transformando-se em monopólios privados regionais, dominados principalmente por grupos multinacionais portugueses, espanhóis e norte-americanos.

O aparelho telefônico proposto por Bell é basicamente o telefone convencional que encontramos hoje em dia, baseado em componentes eletro-mecânicos. Mais

recentemente surgiram os telefones eletrônicos, substituindo os antigos componentes eletro-mecânicos por dispositivos e circuitos eletrônicos. (REIS, 2003).

2.2.1 Centrais Telefônicas

Com o crescimento da demanda por serviços de telefonia, não era mais possível ter um sistema como a invenção de Bell, com linhas diretas e dedicadas entre os usuários. A solução foi a utilização de recursos compartilhados *chaveados* (ou *comutados*) entre as diversas conversações – por isso o uso do termo Rede Telefônica Pública Comutada (RTPC), em inglês, PSTN (*Public Switched Telephone Network*), usado até hoje para se referir ao sistema telefônico público em geral. Em particular, a forma de chaveamento tradicionalmente utilizada em sistemas telefônicos é conhecida como *chaveamento/comutação de circuitos*. Para haver comunicação telefônica, é necessário que se estabeleça um circuito (caminho) entre a origem e o destino durante todo o tempo de conversação. (COLCHER, 2005).

Nos primeiros sistemas telefônicos, o circuito estabelecido entre os interlocutores era feito por uma técnica conhecida como *chaveamento físico manual*, na qual operadores humanos, nas centrais telefônicas, recebiam pedidos de ligação (conexão) e eram encarregados de fechar fisicamente (através de cabos e conectores) os circuitos entre o chamador e o chamado, bem como liberar esse circuito após o término da conversação. A figura 1 mostra uma telefonista efetuando o chaveamento manualmente. (COLCHER, 2005).

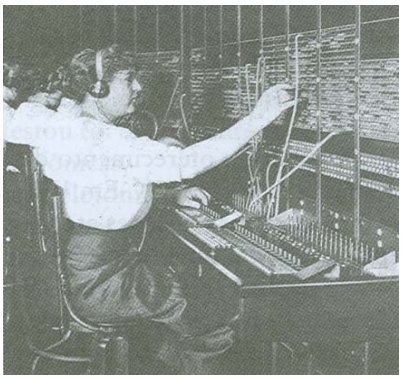

Figura 1: Telefonista efetuando um chaveamento manualmente. (COLCHER, 2005).

Nessa época, existia uma manivela junto ao equipamento do usuário que fazia parte de um conjunto chamado *magneto*. Para realizar uma chamada, a pessoa girava a manivela de seu telefone, gerando uma corrente elétrica que fazia acionar um alarme na mesa operadora da central. A telefonista atendia e, ao ser informada pelo chamador sobre o destino da ligação desejada, fazia tocar a campainha no telefone desejado (usando uma manivela similar na própria mesa). Caso o telefone chamado fosse atendido, a telefonista poderia então completar a ligação usando um cordão condutor unindo os terminais do chamador e do destino solicitado. (COLCHER, 2005).

A primeira central automática eletromecânica de chaveamento foi inventada em 1891 por Almon Strowger, dispensando os operadores humanos. Essa central possuía capacidade apenas para 56 terminais telefônicos. Com a invenção de Strowger, os telefones passaram a não utilizar mais a antiga manivela; usuários podiam indicar diretamente o número do destinatário através de um novo tipo de *dispositivo de discagem*. (COLCHER, 2005).

A figura 2 mostra o chaveador de Strowger e um dos primeiros aparelhos telefônicos com discador.

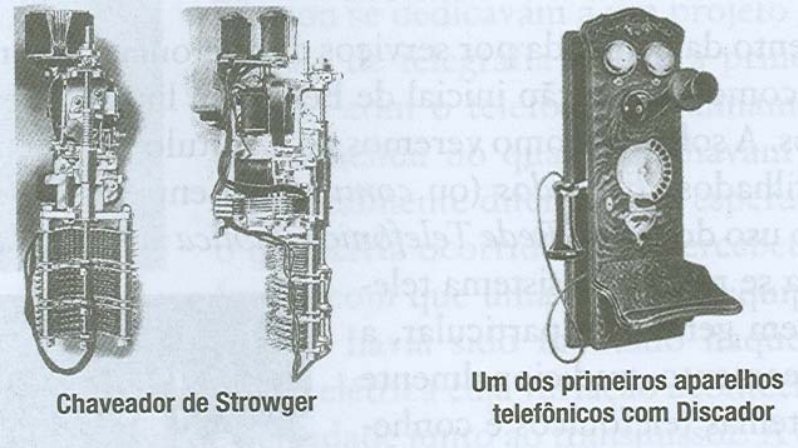

Figura 2: Chaveador de Strowger e um dos primeiros aparelhos telefônicos com discador. (COLCHER, 2005).

A escala de oferecimento do serviço telefônico também começou a crescer no início do século XX. Em 1913, Paris já contava com cerca de 93 mil telefones manuais, com as ligações atendidas por telefonistas. Em Nova York, na mesma época, já havia uma rede com cerca de 500 mil telefones, sendo que a automação do sistema

se iniciaria em 1919. Em 1922 e 1925, antes mesmo de Paris e Estocolmo, foram inauguradas no Brasil (mais precisamente em Porto Alegre), as duas primeiras centrais automáticas do país (sendo que a primeira delas foi a terceira central automática das Américas, depois apenas das de Chicago e Nova York). (COLCHER, 2005).

2.3 Rede Pública de Telefonia

Considerando um esquema básico de Rede Pública de Telefonia existe uma Central Telefônica ligada a vários Centros de Comutação, cada um deles por sua vez ligado a vários aparelhos telefônicos de assinantes (Figura 3). Os Centros de Comutação são distribuidores locais (por exemplo, um bairro) enquanto a Central Telefônica abrange uma área maior (uma pequena cidade ou uma região de uma grande metrópole). (REIS, 2003). Assinante é o consumidor final, residência ou empresa, onde está instalado o aparelho telefônico. (REIS, 2003).

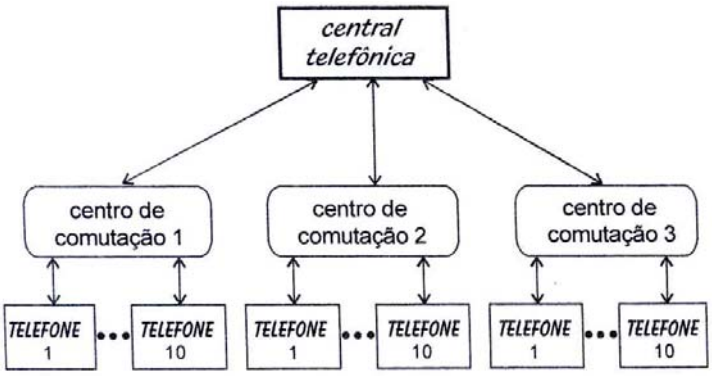

Figura 3: Esquema Básico de Rede Pública de Telefonia. (REIS, 2003).

Em uma operação típica, ilustrada na figura 4, o assinante (no caso Telefone 3) indica ao seu centro de Comutação (nº 2) que deseja se comunicar com o Telefone 8 do Centro de Comutação 3. O Centro repassa a informação para a Central Telefônica, que faz a ligação para o Centro de Comutação 3 e este para o telefone 8.

Assim se completa a ligação entre Telefone 3 do Centro 2 com o telefone 8 do Centro 3. (REIS, 2003).

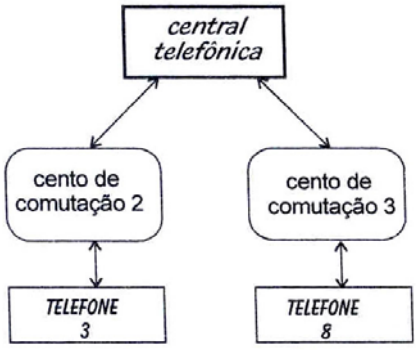

Figura 4: Ligação entre telefones de diferente Centro de Comutação. (REIS, 2003).

Um caso mais simples é a ligação entre aparelhos telefônicos de um mesmo Centro de Comutação, demonstrado na figura 5. Basta que ele complete a ligação, sem intervenção da Central Telefônica. Inversamente, o caso mais complexo será de aparelhos telefônicos instalados em áreas de diferentes Centrais Telefônicas, como demonstrado na figura abaixo. O Centro de Comutação repassa a informação para sua Central Telefônica, completando-se a ligação com o centro de comutação e o assinante desejado (REIS, 2003).

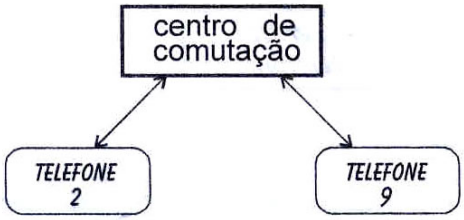

Figura 5: Ligação entre telefones de um mesmo Centro de Comutação. (REIS, 2003).

2.3.1 Linha do Assinante

O aparelho telefônico convencional não dispõe de energia própria. A central telefônica possui uma bateria central, ligada por dois fios ao assinante, que fornece a ele a tensão necessária, cerca de 48 volts dc. (REIS, 2003).

2.4 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.

2.5 O que é um PABX (*Private Automatic Branch Exchange*)

Um PABX é uma central telefônica em que são conectadas as linhas telefônicas e os telefones (aparelhos) de cada usuário chamados de ramais. Com o PABX não há mais a necessidade de uma linha para cada aparelho, pois cada usuário necessita apenas de um ramal (aparelho) em sua mesa e será capaz de atender as

ligações de varias linhas e efetuar ligações de varias linhas. A figura 6 abaixo mostra o exemplo de um PBX.

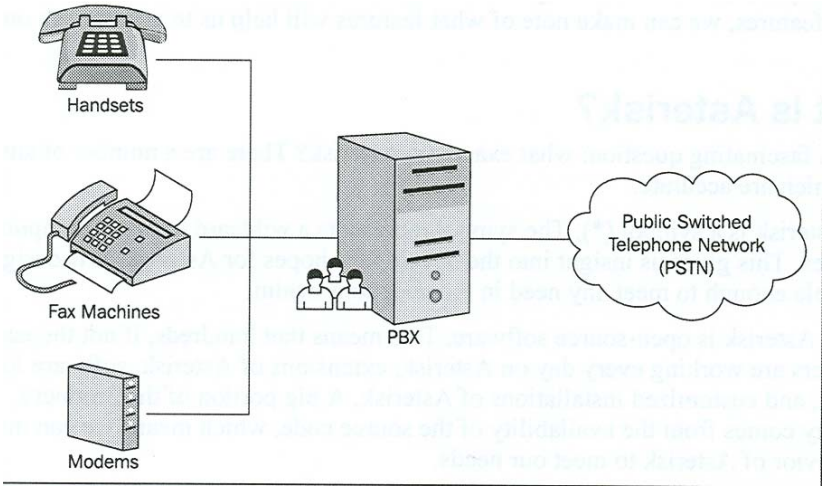

Figura 6: PBX conectando três dispositivos. (GOMILLION, 2005).

Com a utilização de um PABX passamos ter comunicação interna ente os ramais (aparelhos), sigilo entre as ligações, podendo transferir as ligações de uma linha recebidas por um ramal e transferi-la para outro ramal.

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.

2.6 Protocolos de transporte da arquitetura Internet TCP/IP

A Arquitetura Internet permite que sejam utilizados dois tipos de protocolo em sua camada de transporte. O principal deles é o TCP (*Transmission Control Protocol*), que opera no modo orientado à conexão fornecendo um serviço de transferência de dados confiável. A outra opção é o UDP (*User Datagram Protocol*). O UDP opera no modo sem conexão e fornece um serviço datagrama não-confiável, sendo uma simples extensão do protocolo IP, que é o responsável pelo serviço de inter-rede na arquitetura Internet TCP/IP. (SOARES, 1995).

A figura 7 mostra as camadas da arquitetura Internet TCP/IP.

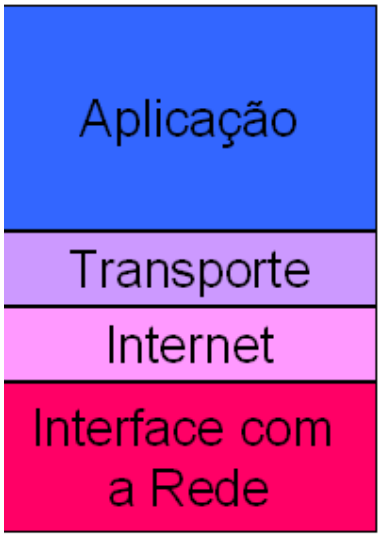

Figura 7: Camadas da arquitetura Internet TCP/IP. (TORRES, 2007)

2.7 O protocolo TCP (*Transmission Control Protocol*)

Em um nível mais baixo, as redes de computadores provêem a entrega de pacotes sem confiança. Os pacotes podem ser perdidos ou destruídos quando erros de transmissão interferem nos dados, quando o hardware de rede falhar, ou quando as redes estiverem com uma carga de dados muito alta para acomodar o tráfego presente. Sistemas de comutação de pacotes mudam as rotas dinamicamente, entregam pacotes fora de ordem, com *delay* ou duplicados. Mais ainda, as tecnologias da camada física das redes podem ditar o tamanho de pacote ideal ou colocar outros impecilhos que devem ser resolvidos para alcançar taxas de transferência eficientes. (COMER, 2005).

Em um nível mais alto, programas geralmente precisam enviar grandes volumes de dados de um computador para outro. Usando um sistema de entrega de pacotes sem confiança para transferência de grandes volumes de dados se torna uma tarefa complicada e requer que os programadores criem mecanismos de detecção e recuperação de erros em cada aplicação. (COMER, 2005).

O TCP é um protocolo orientado à conexão que fornece um serviço confiável de transferência de dados fim a fim. O TCP foi projetado para funcionar com base em um serviço de rede sem conexão e sem confirmação. A figura 8 mostra o posicionamento do TCP na arquitetura Internet TCP/IP. (SOARES, 1995).

Figura 8: Posicionamento do TCP na arquitetura TCP/IP.

O TCP interage de um lado com processos das aplicações e do outro lado com o protocolo da camada inter-rede da arquitetura Internet. A interface entre os processos de aplicação e o TCP consiste em um conjunto de chamadas semelhantes às que os sistemas operacionais fornecem aos processos de aplicação para manipulação de arquivos. Por exemplo, existem chamadas para abrir e fechar conexões e para enviar e receber dados em conexões previamente estabelecidas. A interface entre o TCP e a camada inferior define um mecanismo através do qual as duas camadas trocam informações assincronamente. (SOARES, 1995).

O TCP é capaz de transferir uma cadeia (*stream*) contínua de octetos, nas duas direções, entre seus usuários. Normalmente o TCP decide o momento de parar de agrupar os octetos e de, conseqüentemente, transmitir o segmento formado por esse agrupamento. Porém, caso deseje, o usuário do TCP pode fazer uso da função *push* que faz com que o TCP transmita imediatamente os octetos que estão nos seus *buffers* aguardando transmissão. (SOARES, 1995).

Quando dois processos desejam comunicar-se, as instâncias do TCP às quais eles estão associados devem estabelecer uma conexão. Quando a comunicação

termina, a conexão é encerrada liberando os recursos por ela utilizados. O algoritmo de *three-way handshake* é utilizado na abertura de conexões. Todas as mensagens trocadas identificam a conexão com números de seqüência baseados em relógios, que são utilizados para evitar que a duplicação de mensagens com pedido de abertura de conexão provoque o estabelecimento de conexões inválidas. (SOARES, 1995).

Os processos de aplicação transmitem seus dados fazendo chamadas ao TCP, passando como parâmetros os *buffers* onde estão os dados. O TCP empacota os dados armazenados nos *buffers* em segmentos e chama o módulo IP para transmitir os segmentos para o TCP destino. O TCP receptor coloca os dados recebidos em segmentos nos *buffers* do usuário destinatário e notifica-o da entrega. (SOARES, 1995).

2.8 O protocolo IP (*Internet Protocol*)

O IP foi projetado para permitir a interconexão de redes de computadores que utilizam a tecnologia de comutação de pacotes. O ambiente inter-rede consiste em hosts conectados a redes que por sua vez são interligadas através de *gateways*, como mostra a figura 9. (SOARES, 1995).

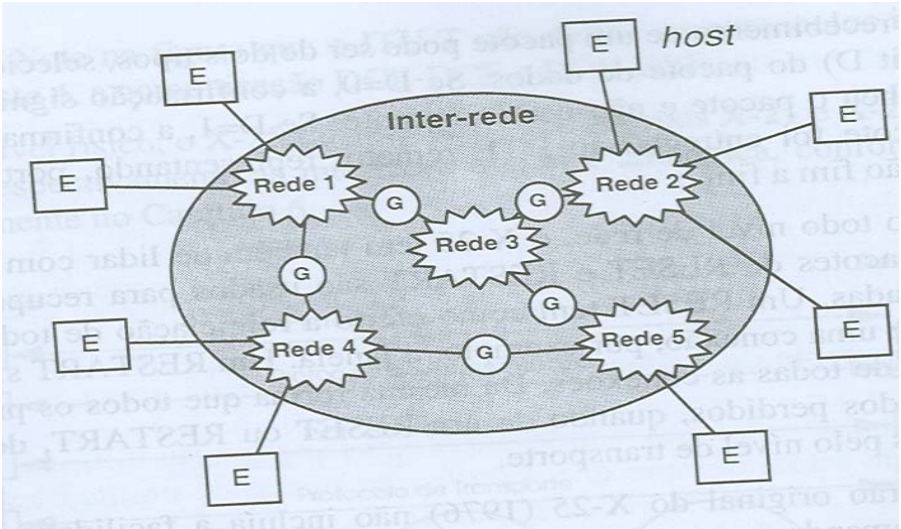

Figura 9: *Hosts* conectados a redes e as mesmas interligas através de *gateways*. (SOARES, 1995).

As redes que fazem parte da inter-rede variam de redes locais (por exemplo, redes *Ethernet*) até redes de grande porte (por exemplo, a *Arpanet*). (SOARES, 1995).

Ao contrário do protocolo X.25, o protocolo IP é um protocolo sem conexões. Sua função é transferir blocos de dados denominados *datagramas* da origem para o destino, onde a origem e o destino são *hosts* identificados por endereços IP. O protocolo IP também fornece o serviço de fragmentação e remontagem de datagramas longos, quando necessário, para que eles possam ser transmitidos através de redes onde o tamanho máximo permitido para os pacotes é pequeno. (SOARES, 1995).

O serviço oferecido pelo IP é sem conexão. Portanto, cada datagrama IP é tratado como uma unidade independente que não possui nenhuma relação com qualquer outro datagrama. A comunicação é não-confiável, não sendo usados reconhecimentos fim a fim ou entre nós intermediários. Nenhum mecanismo de controle de erros nos dados transmitidos é utilizado, exceto um *checksum* do cabeçalho que garante que as informações nele contidas, que são usadas pelos *gateways* para encaminhar os datagramas, estão corretas. Nenhum mecanismo de controle de fluxo é empregado. (SOARES, 1995).

2.9 NAT (*Network Address Translation*)

2.9.1 Introdução ao NAT

Com o crescimento exponencial da utilização da Internet, começa a haver a possibilidade de uma escassez de endereços IP válidos, ou seja, endereços que sejam roteáveis na Internet. Nesse contexto, torna-se útil um protocolo que permita que o crescimento da utilização da rede global não seja freado e que, ao mesmo tempo, não esgote os endereços que são previstos pelo IPV4 (*Internet Protocol Version* 4).

O NAT vem sendo largamente utilizado por muitos administradores de rede para atender a essa demanda. (DUARTE, 2002).

A razão pela qual o NAT é tão importante é, como já foi dito, que o IPV4 fornece um número limitado de endereços (existem 4 octetos, totalizando 32 bits de endereçamento). Com o advento da RFC 1918, foram criadas regras que permitiam a utilização de endereços não-roteáveis nas redes locais, evitando que toda e qualquer

máquina que quisesse se conectar a uma rede tivesse que ser reconhecida por um único e exclusivo endereço em toda a Internet. Com isso, criam-se redes isoladas. Ainda há a necessidade de essas redes se comunicarem. É aí que entra a tradução de endereços. (DUARTE, 2002).

O NAT opera normalmente em um roteador ou em um *firewall*, que são dispositivos que recebem conexões de diferentes redes em seus terminais, como podemos ver na figura 10 abaixo: (DUARTE, 2002).

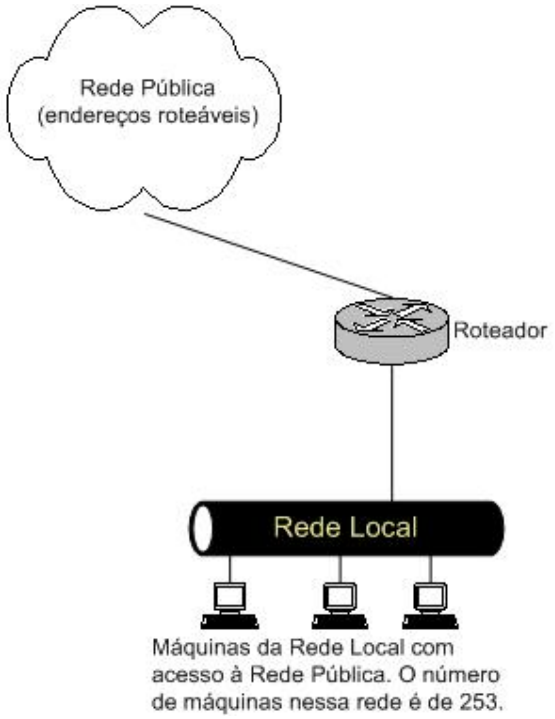

Figura 10: Roteador, com NAT, interligando a rede pública a rede local. (DUARTE, 2002).

Nota-se na figura que existem 253 máquinas na rede local querendo acessa a rede pública ou válida. Em uma situação normal, sem NAT, haveria a necessidade de 253 endereços válidos para prover esse acesso. (DUARTE, 2002).

2.9.2 Tipos de NAT

Existem três tipos de NAT, são eles:

- NAT estático

- NAT Dinâmico
- PAT – *Port Address Translation*

2.9.2.1 NAT Estático

Como o próprio nome indica, o NAT estático define um endereço fixo de tradução de uma máquina da Rede Local para a Rede Pública. Esse tipo de NAT é muito utilizado quando se quer ocultar o endereçamento interno de uma máquina para a rede pública e também torná-la visível para a mesma. (DUARTE, 2002).

2.9.2.2 NAT Dinâmico

Nesta técnica, trabalha-se com uma faixa de endereços que ficam à disposição do dispositivo tradutor (*firewall* ou roteador) para realizar a conversão de endereços. A cada requisição feita, ele consulta essa faixa e utiliza-se do primeiro endereço livre que encontrar. Este modelo é considerado o mais flexível, pois permite uma série de diferentes soluções para fazer a conversão. (DUARTE, 2002).

2.9.2.3 PAT – Port Address Translation

O *Port Address Translation* éo tipo de NAT que mais economiza endereços válidos (roteáveis), pois a tradução é feita no modelo N para 1, ou seja, todos os endereços da rede local são traduzidos para um único endereço válido. Esse tipo de NAT é, na verdade, um caso especial do NAT dinâmico, onde as traduções são feitas sob demanda, ou seja, só existe tradução quando houver uma requisição realizada. (DUARTE, 2002).

2.10 VoIP

2.10.1 O que é VoIP?

VoIP é um acrônimo para "Voz sobre IP" (*Voice over IP*). É um termo utilizado para definir o serviço que consiste em transmitir informação de voz através do Protocolo IP - TCP ou UDP.

De uma forma geral, significa enviar informação de voz em formato digital dentro de pacotes de dados, ao invés do tradicional protocolo de comutação de circuitos utilizado há décadas pelas companhias telefônicas.

A maior vantagem da tecnologia VoIP é a redução dos custos de utilização dos serviços de telefonia comum, principalmente em ambientes corporativos. As redes de dados já instaladas passam também a transmitir voz e, dessa forma, os custos podem ser reduzidos independente do dia da semana, da hora e duração da chamada.

Como decorrência do apelo em torno do nome Internet, uma gama de serviços tradicionalmente vinculados a redes específicas passou a ser também oferecida sobre redes IP. Fluxos para distribuição de áudio e vídeo (*streaming*) de emissoras de rádio e TV pela Internet são lugar-comum hoje, por exemplo. A provisão de serviços de comunicação vocálica sobre redes IP (VoIP), em especial, tem recebido grande atenção das concessionárias de telefonia regionais e de longa distância, dos provedores de serviços Internet, e de outros provedores de serviços de comunicação (como TV a cabo, por exemplo). (COLCHER, 2005).

O conceito de VoIP tomou forma em meados da década de 1990, quando surgiu o primeiro software comercial – o Internet Phone (da VocalTec Communications) – a permitir a troca de pacotes IP transportando amostras de voz entre computadores pessoais. Contudo, naquela época a qualidade da comunicação não chegava nem próximo da qualidade padrão dos sistemas telefônicos convencionais. Mas a tecnologia VoIP evoluiu rapidamente, e, por volta de 1998, algumas pequenas companhias já eram capazes de oferecer serviço de VoIP, com certa qualidade, interligado ao serviço de telefonia convencional. (COLCHER, 2005).

2.10.2 A tecnologia

VoIP é a terceira geração do sistema de telefonia. Os sistemas originais de telefonia eram todos analógicos. Eles convertiam o sinal de voz em ondas elétricas analógicas e essas ondas eram convertidas novamente para sinal de voz na outra ponta. Nos anos 50 as redes digitais substituíram as redes analógicas. Numa rede digital, os sinais de voz são convertidos para uns e zeros e enviados digitalmente através da rede de telefonia e depois convertidos de volta na outra ponta. Isto era feito com uma série de repetidores e como todos os sinais eram ou um ou zero, eles podiam ser recriados na outra ponta de maneira bastante idêntica de como eles foram transmitidos. Isto eliminou o maior problema dos sistemas de telefonia analógicos, onde o ruído era amplificado conforme o sinal analógico era amplificado conforme ele se tornava fraco com a distância. Nas redes analógicas, quanto maior era a distância, menor era a qualidade. Esta perca em qualidade foi praticamente eliminada com as redes digitais.

Chamadas VoIP consistem de dois principais componentes, uma porção portadora que transporta a comunicação de voz e outra porção de sinalização que contêm as informações para criar e finalizar as chamadas. Como chamadas VoIP são sensíveis a atrasos (*delays*), VoIP emprega duas técnicas básicas para ajudar a eliminar esses atrasos. Primeiro as chamadas VoIP são transportadas em pacotes pelo *Realtime Protocol* (RTP), que foi projetado para transmissão de dados sensíveis ao tempo. Pacotes RTP são transportados pelo protocolo de transporte UDP, mas o RTP adiciona informações de seqüência para que os pacotes possam ser ordenados na ordem correta e adiciona também *time stamping* para ajudar as pontas a gerenciar o *Jitter*. A figura 11 exemplifica esse processo.

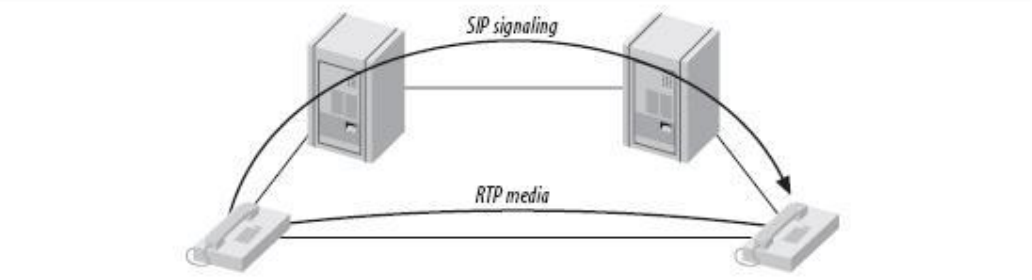

Figura 11: Principais componentes de uma chamada VoIP. (MEGGELEN et al,2005)

A segunda técnica é o uso de algoritmos de codificação de voz chamados *codecs*, que comprimem os dados e gerencia a qualidade da chamada. Comprimindo os dados, menos informação tem que ser transmitida, ajudando assim a manter a qualidade de serviço (QoS).

2.10.3 *Jitter* (Retardo Variável)

O *Jitter* é a variação do tempo entre a chegada de pacotes consecutivos.

No caso da utilização de redes de comunicação que apresentam variação estatística do retardo (como as redes comutadas por pacotes), tal variação deve ser compensada. (COLCHER, 2005)

Na figura 12, a linha horizontal superior apresenta os surtos de voz e de silêncio sendo gerados na fonte a uma taxa constante. Os surtos de voz são divididos em pacotes, que são as unidades que transitarão na rede de comunicação (os surtos de silêncio não são transmitidos). Uma vez que o pacote é gerado, ele é imediatamente entregue para transmissão. Se os pacotes sofrerem retardos variáveis, chegarão ao destino não mais preservando a continuidade, conforme mostra a linha horizontal inferior da figura, podendo gerar intervalos de silêncio dentro de um surto de voz, ou diminuir, e até mesmo eliminar, intervalos de silêncio, o que pode causar a perda da inteligibilidade da informação no destino. Alguma forma de compensação dessa variação estatística do retardo deve ser realizada.

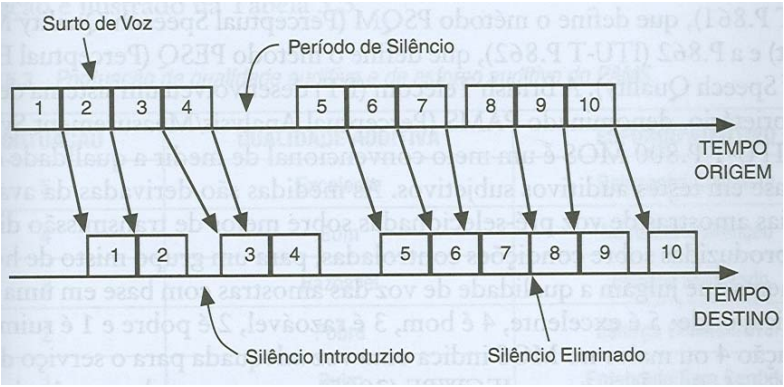

Figura 12: Efeito da variação estatística do retardo na comunicação de voz.
(COLCHER, 2005)

A estratégia utilizada pelos algoritmos de compensação baseia-se fundamentalmente em assegurar uma reserva de pacotes antes de dar início ao processo de reprodução, introduzindo um retardo inicial a cada surto de voz. Aparentemente, o problema estaria resolvido se escolhêssemos o retardo inicial bem grande; entretanto, o valor desse retardo está limitado pelo máximo retardo de transferência (desde a geração até a reprodução) permitido para o sinal de voz, sem que haja perda da interatividade da comunicação. (COLCHER, 2005)

2.11 IPBX

2.11.1 O que é um IPBX

Um IPBX, ou PABX IP, é um sistema completo de telefonia que fornece chamadas telefônicas em cima das redes de dados IP. Todas as conversações são enviadas como pacotes de dados sobre a rede. (GALEA, 2007).

A tecnologia inclui funcionalidades avançadas de comunicação, e, também, fornece o remédio certo para a dor de cabeça da escalabilidade e robustez que todas as corporações perseguem. O PABX IP é capaz de se conectar as linhas tradicionais da Rede Pública tradicional via *gateways* opcionais — de forma que as atualizações constantes no sistema de comunicação da empresa para essa rede avançada de voz e de dados sejam executadas quase que instantaneamente. (GALEA, 2007).

As corporações não precisam fazer uma ruptura nas suas operações e na infra-estrutura corrente de comunicação externa. Com o PABX IP implementado, uma corporação pode até mesmo manter seus números telefônicos regulares. Dessa forma, o PABX IP comuta ligações locais sobre a rede de dados interna da corporação e permite que todos os usuários compartilhem as mesmas linhas telefônicas externas. (GALEA, 2007).

Um sistema PABX IP consiste de um ou mais telefones SIP, um servidor PABX IP e um *gateway* VoIP opcional para conectar-se as linhas da rede pública existentes. O servidor PABX IP funciona de forma similar a um servidor *proxy*: os clientes SIP, sendo tanto *soft phones* quanto aparelhos IP´s, registram-se com o servidor PABX IP, e quando eles desejam fazer uma chamada eles solicitam ao

PABX IP estabelecer a conexão. O PABX IP tem um diretório de todos os telefones/usuários e seus endereços SIP correspondentes e assim é capaz de conectar uma chamada interna ou rotear uma chamada externa tanto por um *gateway* VoIP como por uma operadora de serviço VoIP. (GALEA,2007).

2.11.2 Benefícios na utilização de um IPBX

Fácil de instalar e configurar do que um sistema de telefonia proprietário. Um PABX IP roda como software sobre um computador e pode fazer uso de todo o poder de processamento avançado do computador e de interface de usuário bem como características de janelas. Qualquer pessoa proficiente em redes e computadores pode instalar e manter um PABX IP. Em contraste, um sistema de telefonia proprietário frequentemente requer um instalador treinado naquele sistema proprietário particular. Fácil de gerenciar decorrente da interface de configuração baseado *web*/GUI. Um PABX IP pode ser gerenciado via uma interface de configuração baseado em *web* ou uma interface gráfica de usuário (GUI), permitindo você facilmente fazer manutenção e ajustes fino no seu sistema de telefonia. Sistemas Proprietários de telefonia possuem interfaces de difícil uso que frequentemente são projetadas para serem usadas somente por técnicos de telefonia. (GALEA, 2007).

Economia significativa de recursos usando operadoras VoIP.Com um PABX IP você pode facilmente usar um fornecedor de serviço VoIP para chamadas de longa distância e internacional. A economia mensal é significativa. Você pode facilmente conectar sistemas telefônicos entre matrizes e filiais e fazer ligações telefônicas gratuitas.

Eliminação do emaranhado de fios da rede telefonia. Um PABX IP permite você conectar aparelhos telefônicos IP diretamente a uma porta padrão da rede de computadores (que pode ser compartilhada com o computador adjacente). *Soft phones* podem ser instalados diretamente no PC. Você pode agora eliminar o emaranhado de fios dos telefones e fazer acréscimo ou movimentação dos ramais muito facilmente. Em novos escritórios você pode eliminar completamente as portas extras a serem usadas por telefones. (GALEA, 2007).

Não ficar preso a um fornecedor específico. Os PABXs IP são baseados no padrão aberto do SIP. Você pode agora fazer um mixto e combinar quaisquer aparelhos telefônicos IP´s SIP ou *soft phone* com qualquer PABX IP baseado no SIP, *gateways* PSTN ou com operadora VoIP. Em contraste, um sistema de telefonia proprietário frequentemente requer telefones proprietários para usar funcionalidades avançadas, e módulos de expansão proprietários para acrescentar funcionalidades. (GALEA, 2007).

Escalabilidade. Os Sistemas proprietários ficam facilmente incipientes para atender ao crescimento da demanda e necessidades dos clientes: quando da adição de linhas telefônicas ou de ramais frequentemente exigem módulos caros de hardware. Em alguns casos você necessita de um novo sistema de telefonia inteiramente novo. Isso não acontece com um PABX IP: um computador padrão pode facilmente manipular um grande número de linhas telefônicas e ramais — apenas acrescentando-se mais telefones em sincronia justamente com a expansão da rede. (GALEA, 2007).

Duplicação das funcionalidades do sistema telefônico pela metade do preço. Já que um PABX IP é baseado em software, é muito fácil para os desenvolvedores acrescentar e melhorar o conjunto de funcionalidades. Muitos sistemas de telefonia VoIP, como o *Asterisk*, já vem com um conjunto rico de funcionalidades, incluindo auto atendimento, *voicemail*, conferências, geração de relatórios avançados e mais além. Estas opções são freqüentemente muito caras em sistemas proprietários. (GALEA, 2007).

Permite a movimentação constante de mesas e o *roaming* (movimentação do usuário). (GALEA, 2007).

Mudanças constantes de layout das mesas — o processo de ser capaz de se movimentar facilmente o escritório/mesa em função da tarefa em tela tem se tornado muito popular. Infelizmente PABXs tradicionais requerem que os ramais sejam re-jampeados no DG para o novo lugar. Com um PABX IP o usuário simplesmente leva seu telefone para a sua nova mesa — sem necessidade de fazer o re-jampeamento, só conectando à tomada de rede mais próxima. (GALEA, 2007).

Os usuários podem fazer *roaming* também — se um empregado precisar trabalhar de casa, ele pode simplesmente roda seu *soft phone* SIP e ser capaz de atender chamadas no seu ramal, exatamente como eles fariam no escritório. As chamadas podem ser roteadas para qualquer lugar do mundo por causa das características do protocolo SIP. (GALEA, 2007).

Melhor usabilidade do telefone: telefones SIP são muito fáceis de usar. Os empregados frequentemente fazem uso constante de funcionalidades avançadas de telefonia: Estabelecimento de conferência, transferência de chamada — sobre um PABX velho tudo requer instruções. Já isso não acontece com um PABX IP — todas as funcionalidades são facilmente executadas a partir de uma interface amigável de usuário baseado em janela. Além disso, os usuários conseguem uma melhor visão do status de outros ramais e de linhas entrantes e filas de chamadas via a janela da aplicação cliente do PABX IP. Os sistemas proprietários frequentemente requerem "sistemas" de telefonia caros para que você possa ter uma idéia do que está acontecendo com o seu sistema de telefonia. Assim mesmo, a informação de status é algo muito confuso na melhor das hipóteses. (GALEA, 2007).

2.12 Protocolo SIP (*Session Initiation Protocol*)

O protocolo de iniciação de sessão entrou no mundo do VoIP como uma tempestade. Originalmente considerado nada mais do que uma idéia interessante, o SIP hoje promete destronar o conhecido protocolo H.323 como escolha de protocolo para VoIP. A premissa do SIP é que cada conexão final é um *peer*, e o protocolo negocia as características entre eles. O que torna o SIP uma escolha bastante interessante é que ele é um protocolo relativamente simples, com uma sintaxe similar aos protocolos de outras famílias, como o HTTP e o SMTP. (MEGGELEN et al,2005)

O SIP é suportado no *Asterisk* pelo módulo *chan_sip.so*. (MEGGELEN et al,2005)

2.12.1 História do protocolo SIP

O SIP foi originalmente submetido ao IETF (*Internet Engineering Task Force*) em Fevereiro de 1996. O resumo inicial parecia nada mais do que o SIP como é conhecido hoje e continha somente um único tipo de requisição: uma chamada de requisição de instalação. Em Março de 1999, após 11 revisões, nascia a RFC 2543 do SIP.

De primeiro, o SIP foi totalmente ignorado, já que o H.323 era o protocolo escolhido para negociação de transporte VoIP. No entanto o SIP começou a ganhar

popularidade e como havia vários fatores diferentes que aceleraram o seu crescimento, somos levados a pensar que grande parte do seu sucesso foi devido a sua especificação que é disponibilizada livremente. (MEGGELEN et al,2005)

2.12.2 Futuro do protocolo SIP

O SIP ganhou seu lugar como um protocolo justificado para VoIP. Todo novo usuário e produtos coorporativos suportam SIP e todos os outros produtos já existentes serão dificilmente vendidos a não ser que possuem uma maneira de migração para o SIP. É esperado do SIP mais do que somente suporte a VoIP, mas sim incluindo habilidades para transmitir vídeo, música e qualquer outro tipo de multimídia em tempo real. A expectativa é que isso ocorra nos próximos anos. (MEGGELEN et al, 2005).

2.12.3 Considerações de segurança sobre o protocolo SIP

O SIP utiliza um sistema de *challenge/response* para autenticar os usuários. Um *INVITE* inicial é enviado para o *proxy* onde o dispositivo que se deseja comunicar esteja conectado. O *proxy* então envia de volta uma mensagem "407 *Proxy Authorization Request*", que contem um conjunto aleatório de caracteres conhecidos como *nonce*. Este *nonce* é usado juntamente com a senha para gerar um *hash* MD5, o qual é enviado de volta *INVITE* subseqüente. Se o *hash* MD5 for igual ao gerado pelo *proxy*, o cliente é então autenticado. (MEGGELEN et al, 2005).

Ataques de negação de serviço (DoS – *Denial of Service*) são, provavelmente, o tipo de ataque mais comum em comunicações VoIP. Um ataque de negação de serviço pode ocorrer quando um grande número de requisições *INVITE* inválidas são enviadas ao *proxy* tentando assim sobrecarregar o sistema. Esse tipo de ataque é relativamente simples de implementar e os efeitos para os usuários são imediatos. O SIP possui vários métodos para minimizar os efeitos de ataques de negação de serviços, mas eles são impossíveis de prevenir totalmente. (MEGGELEN et al, 2005).

O SIP implementa um esquema para garantir que um mecanismo de transporte seguro e criptografado (chamado *Transport Layer Security* – TLS) seja utilizado para estabelecer uma comunicação segura entre o emissor e o domínio do receptor da chamada. Além disso, a requisição é enviada de um modo seguro para o dispositivo final, baseado nas políticas locais de segurança da rede. Note que a criptografia da media (que é, um fluxo RTP) está além do escopo do SIP e deve ser tratada separadamente. (MEGGELEN et al, 2005).

2.12.4 SIP e NAT

Provavelmente o maior obstáculo que o SIP tem que superar é o desafio de fazer suas transações através de NAT. Como o SIP encapsula as informações de endereçamento nos seus quadros de dados, e o NAT ocorre numa camada de rede mais baixa, as informações de endereçamento não são modificadas e, portanto os fluxos de media não vão ter as informações de endereçamento corretas necessárias para completar a conexão quando está sendo utilizado NAT. Além disso, os *firewalls* normalmente integrados com NAT não irão considerar o fluxo de media entrante como sendo parte de uma transação SIP e irá bloquear a conexão.

3 IMPLEMENTAÇÃO DO PROJETO

3.1 *Asterisk*

A necessidade e algumas vezes o custo, são os principais motivos para as invenções e com o *Asterisk* não foi diferente. Em 1999, Mark Spencer com quase quatro mil dólares de capital, abriu um empresa prestadora de suporte técnico comercial e livre ao Linux, chamada Linux Support Services. (GONZALEZ, 2007).

Com a grande demanda de chamados técnicos, Mark sentiu a necessidade de um sistema telefônico que pudesse auxiliar no suporte técnico 24h realizando algumas tarefas, como: atender automaticamente as ligações, coletar a identificação do cliente, gravar a mensagem de suporte ou dúvida, localizar um técnico disponível e enviar a mensagem. Dessa forma, conseguiriam, supostamente, atender maior quantidade de chamados, de forma ágil e prática. Naquela época, com o baixo capital, Mark não tinha condições de adquirir um sistema telefônico com as funcionalidades necessárias para esta solução.

Mark possuía uma experiência de cinco anos com o Linux, havia participado no desenvolvimento de diversos programas de código aberto e na completa ausência de alguma pessoa que pudesse lhe auxiliar e explicar a complexidade de tal tarefa, decidiu que iria projetar e desenvolver o seu próprio sistema telefônico utilizando equipamentos emprestado. Em alguns meses de desenvolvimento, Mark possuía uma plataforma livre de telefonia que supria suas necessidades na Linux Support Services e o disponibilizou na Internet com o nome de *Asterisk*, Mark batizou seu projeto de *Asterisk* (asterisco) por ser tanto uma tecla do telefone comum, como também um símbolo curinga no GNU/Linux, por exemplo, rm –rf *. (GONZALEZ, 2007).

Nessa mesma época um homem chamado Jim Dixon do projeto Zapata Telephony, sabia que o motivo que elevava o preços das placas de telefonia, eram os Processadores de Sinal Digital (DSP), insatisfeito, realizou algumas experiências em seu computador pessoal com a placa Mitel39000C "ISDN Express Development Card", escrevendo um *driver* para o FreeBSD. (GONZALEZ, 2007).

Comprovando seus estudos, Dixon concluiu que a placa utilizou pouco processamento em um Pentium III 600Mhz, e com aquela máquina conseguiria

gerenciar de 50 a 75 canais e o que limitava era a forma ineficiente de gerenciar o Entrada/Saída (I/O). Com esses resultados Dixon comprou o necessário para desenhar uma nova placa que usasse o I/O de forma eficiente. (GONZALEZ, 2007).

Mais uma vez Dixon obteve sucesso em seus experimentos e sabia que o conceito utilizado era revolucionário, acabou disponibilizando o desenho completo e os arquivos da placa na Internet, assim começou a organização Emiliano Zapata. Algumas pessoas entraram em contato com Dixon e sempre com a mesma pergunta: ”- Legal, mas você têm para Linux?”. Ele nunca tinha visto um Linux em funcionamento, então comprou o Linux Red Hat 6.0 e começou a estudá-lo para poder portar seu driver do BSD para um módulo do kernel Linux. Começou com algumas dificuldades e mesmo assim disponibilizou na Internet, pois sabia que em algum lugar do mundo algum guru4 do Linux iria ajudá-lo. (GONZALEZ, 2007).

Em menos de 48 horas recebeu um e-Mail do Mark Spencer, o qual se ofereceu para fazer exatamente isto, e falando que tinha algo que seria perfeito para a coisa toda,o *Asterisk* . (GONZALEZ, 2007).

Até o momento o *Asterisk* era um conceito funcional, porém não tinha uma forma real de funcionar de forma prática e útil. A junção do *Asterisk* de Mark, a placa de Dixon e o módulo do kernel Linux desenvolvido por ambos era perfeita e possibilitou o crescimento até se tornar um PABX real que poderia se comunicar com telefones e linhas reais. (GONÇALVES, 2006).

Em 2001, com o crescimento da economia, Mark percebeu que seria mais vantajoso dar continuidade no desenvolvimento do *Asterisk* do que continuar com o suporte técnico na Linux Support Services, então não era o melhor nome para uma empresa de telefonia, assim resolveram alterara o nome da empresa para Digium. (GONZALEZ, 2007).

3.1.1 O que é o *Asterisk*?

Essa é uma pergunta interessante: O que exatamente é o *Asterisk*? Existem várias respostas para essa pergunta.

Primeiro, *Asterisk* é um símbolo (*). Este símbolo representa um curinga (*wildcard*) em muitas linguagens de computador. Isto mostra a intenção dos desenvolvedores do *Asterisk*. Ele foi projetado para ser flexível o bastante para atender a qualquer necessidade no reino da telefonia. (GOMILLION, 2005).

Segundo, *Asterisk* é um programa de código fonte aberto (*open source*). Isto quer dizer que centenas, se não milhares, de desenvolvedores estão trabalhando todo dia no *Asterisk*, extensões do *Asterisk*, programas para o *Asterisk* e instalações customizadas do *Asterisk*. Grande parte da flexibilidade do *Asterisk* vem da disponibilidade do código fonte, o que quer dizer que qualquer um pode modificar o comportamento do *Asterisk* para atender a suas necessidades. (GOMILLION, 2005). Ele foi criado pela Digium Inc. O programa *Asterisk* foi criado para ser usado em plataforma Linux ou qualquer outra plataforma Unix.

Finalmente, e mais importante, *Asterisk* é um *framework* que possibilita a seleção e remoção de módulos particulares, possibilitando assim a criação de um sistema de telefonia customizado. A arquitetura bem projetada do *Asterisk* provê flexibilidade permitindo assim a criação de módulos que estendam seu sistema de telefonia, ou até mesmo servem como substitutos para os módulos padrões do *Asterisk*. (GOMILLION, 2005).

3.1.2 *Asterisk* é um PBX (*Private Branch Exchange*)

Asterisk é *Private Branch Exchange* (PBX). Um PBX pode ser entendido como um quadro de distribuição, conectando a um ou mais telefones de um lado e normalmente conectando a uma ou mais linhas telefônicas do outro lado. Isto é normalmente mais econômico do que possuir uma linha telefônica para cada telefone. (GOMILLION, 2005).

3.1.3 *Asterisk* é um sistema IVR (*Interactive Voice Response*)

Resposta de voz interativa, ou do inglês IVR (*Interactive Voice Response*), revolucionou a área de telefonia em geral. A capacidade e flexibilidade de sistemas de telefonia programáveis provê a habilidade de responder aos clientes de várias maneiras.

O *Asterisk* pode ser usado para provê serviços 24 horas, enquanto reduz a carga de trabalho dos empregados a qualquer hora. *Asterisk* provê a possibilidade de reproduzir arquivos de áudio, ler textos e ainda, buscar informações em um banco de dados. Esta é a tecnologia usada em sistemas de telefonia de bancos e sistemas de pagamentos de contas. Quando se liga para um banco ouve-se uma variedade de gravações e comandos que podem ser passados normalmente usando o teclado do telefone. Por exemplo, após ouvir a mensagem de saudação é solicitado o número da conta e senha para transações bancárias. É possível ainda ouvir informações personalizadas, que são buscadas em um banco de dados, como por exemplo, as últimas transações realizadas para aquela conta. Sistemas como estes podem ser, e muitos têm sido implementados usando o *Asterisk*. (GOMILLION, 2005).

3.1.4 *Asterisk* é um sistema de correio de voz

Asterisk possui um sistema de correio de voz (*voicemail*) totalmente funcional. Este sistema de correio de voz é surpreendentemente poderoso. Ele suporta diferentes contextos de correio de voz, o que permite que várias organizações possam ser hospedadas em um mesmo servidor, cada uma com seu correio de voz separado e independente. Ele suporta diferentes configurações de fuso horário, o que permite que os usuários possam acompanhar quando as chamadas foram recebidas. Ele ainda possui a opção de notificar o recipiente de novas mensagens via *email*: de fato, a mensagem de áudio deve ser anexada ao *email*. (GOMILLION, 2005).

3.1.5 *Asterisk* é um sistema de voz sobre IP

Asterisk provê a habilidade de usar o protocolo IP para fazer chamadas telefônicas. Escolher usar o *Asterisk* não significa que só é possível utilizar voz sobre IP para fazer chamadas. De fato, muitas instalações do *Asterisk* não usam voz sobre IP. Mas muitos desses sistemas possuem capacidade de adicionar a opção de fazerem chamadas usando voz sobre IP facilmente, sem custos adicionais. (GOMILLION, 2005).

Muitas empresas possuem duas redes: uma para telefonia, e outra para computadores. E se fosse possível unir essas duas redes? Qual seria a economia com isto? A grande economia é a redução da sobrecarga sobre os funcionários da área de Tecnologia da Informação. Podem-se ter especialistas em computação e redes de computadores, e já que a parte de telefonia vai ser executada em um computador e sobre uma rede IP, o próprio conhecimento sobre computação e redes de computadores irá habilitar os funcionários a manter o sistema de telefonia. (GOMILLION, 2005).

Outro benefício é a economia com compra de equipamentos. Computadores vão, ao longo do tempo, ficando mais baratos, enquanto sistemas de telefonia proprietários tendem a manter aproximadamente um preço constante. Além disso, equipamentos como *switches*, roteadores e outros equipamentos de rede tendem a ficar mais baratos ao longo do tempo. (GOMILLION, 2005).

Voz sobre IP habilita a utilização em qualquer lugar, desde que se tenha uma conexão à Internet de banda larga. Isto significa que os funcionários podem ter acesso ao sistema de telefonia da empresa estando em casa, desde que possuam conexão à Internet de banda larga. Então, eles terão acesso a todos os serviços disponíveis na empresa, como correio de voz, chamadas de longa distância e chamadas direto a um ramal da empresa. (GOMILLION, 2005).

A figura 13 abaixo mostra a arquitetura do *Asterisk*:

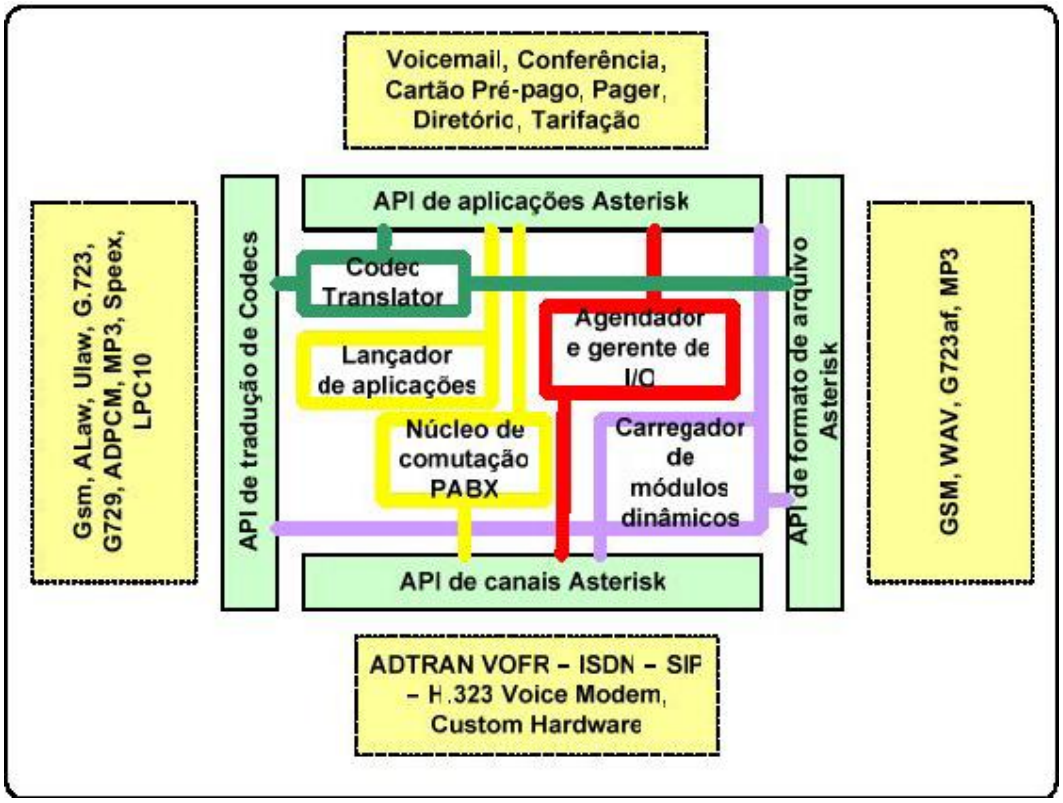

Figura 13: Arquitetura do *Asterisk.* (GONÇALVES, 2006).

3.2 FreeBSD

3.2.1 O que é FreeBSD

FreeBSD é um avançado sistema operacional compatível com arquiteturas x86 (incluindo Pentium e Athlon), amd64 (incluindo Opteron, Athlon64 e EM64T), UltraSPARC, IA-64, PC-98 e ARM. Plataformas adicionais estão em vários estágios de desenvolvimento.Ele é desenvolvido e mantido por um grande número de indivíduos.

FreeBSD é derivado do UNIX de *Berkeley,* um sabor de UNIX desenvolvido pelo *Computer System Research Group* na universidade da Califórnia em *Berkeley* e inicialmente lançado como *Berkeley Software Distribuition* (BSD) do UNIX[1].

[1] UNIX é uma marca registrada do *Open Group*, portanto legalmente, FreeBSD não pode ser chamado UNIX.

Assim como versões comerciais do UNIX, FreeBSD provê muitas características avançadas, incluindo:

- Multitarefa preemptivo: utiliza algoritmos de prioridade dinâmica para garantir um correto compartilhamento de recursos do computador entre as aplicações e os usuários.
- Sistema multi-usuário: vários usuários podem usar o sistema simultâneamente.
- Segurança: os registros de eventos de segurança são publicados pelo CERT (www.cert.org), uma organização renomada na área de segurança da computação.
- Confiabilidade: o sistema é usado por vários provedores de Internet em todo o mundo. Por exemplo, o *Yahoo*. Sistemas FreeBSD regularmente funcionam vários anos sem precisar serem reiniciados.
- Implementação completa da pilha de protocolos TCP/IP: isto significa que o FreeBSD provê interoperabilidade com outros sistemas que implementam a pilha de protocolos TCP/IP.
- Proteção de espaço de memória: garante que nem aplicações nem usuários interfiram no espaço de memória do outro. Se uma aplicação parar de funcionar, ela não afeta as outras aplicações que estão sendo executadas.
- Avançado sistema de instalação de programas, chamado *ports:* a coleção do *ports* do FreeBSD possui milhares de programas prontos para serem instalados e utilizados.
- Código fonte disponível e livre para uso: código fonte livre sobre termos da licença BSD, qualquer um pode alterar ou estudar o código fonte.

FreeBSD é baseado na versão 4.4BSD UNIX criado pelo *Computer System Research Group* na universidade da Califórnia em *Berkeley.* O projeto FreeBSD gasta muitas horas fazendo ajustes e testes no sistema para garantir uma ótima performance e confiabilidade. As características do FreeBSD, como performance e confiabilidade, são comparadas favoravelmente, com outros sistemas operacionais comerciais.

O crescimento da Internet se deu, principalmente, através do UNIX de Berkeley. A implementação original do TCP/IP, lançada em 1982, foi baseada no 4.4BSD, e praticamente toda implementação do TCP/IP atual foi copiada dele.

FreeBSD é um descendente do 4.4BSD e possui a pilha TCP/IP mais madura e confiável da atualidade. Isto o torna uma plataforma ideal para vários serviços de Internet, como servidores FTP, servidores WEB, servidores de correio eletrônico, servidores de DNS, *firewalls* e etc.

O desenvolvimento do FreeBSD é um trabalho de amor. Grandes empresas comerciais desenvolvem sistemas operacionais e cobram caro por eles. O projeto FreeBSD desenvolve um sistema operacional de qualidade profissional e libera-o grátis.Esta não é a única diferença.

3.2.2 Motivo da escolha do FreeBSD

Escolhi utilizar o FreeBSD como sistema operacional para a instalação do *Asterisk* por vários motivos. Um deles é a questão da familiaridade com o sistema operacional. Trabalho com FreeBSD desde 2001 e portanto tenho um bom conhecimento do sistema e seus recursos. Outra questão é a estabilidade, performance e capacidade de recuperação após falhas que o FreeBSD possui, poucas vezes tive problemas com recuperação de dados após falhas de energia. Um outro motivo é a questão de instalação de programas. O FreeBSD possui um sistema chamado *ports* que possibilita a instalação de pacotes através de seu código fonte, compilando na hora o programa e suas dependências. Como é compilado na hora através do código fonte podemos passar parâmetros de otimização e de controle conforme a necessidade. Uma característica muito útil do *ports* é que ele resolve as dependências automaticamente, o que facilita e muito a instalação de programas no FreeBSD. Outro motivo, e um dos mais importantes, é que Jim Dixon, do projeto *Zapata Telephony*, que sabia que as placas de telefonia eram caras devido ao DSP (*Digital Signal Processor*), começou a realizar algumas experiências em seu computador pessoal com a placa *Mitel39000C "ISDN Express Development Card*", escrevendo um driver para FreeBSD. Após isso, Mark Spencer, o criador do *Asterisk*, começou a portar o código para que o mesmo funcionasse em Linux. Portanto o a versão original do *driver* foi desenvolvido para o FreeBSD e isso influenciou na escolha do mesmo como sistema operacional para este projeto.

3.3 Porta FXO

O *Asterisk* possui suporte a tecnologias tradicionais de telefonia analógicas e digitais, como sinalizações ISDN, R2, *Foreign eXchange Station* (FXS) e *Foreign eXchange Office* (FXO) sendo que as placas e equipamentos são de baixo custo se comparado aos equipamentos de PABX convencionais.

3.3.1 Canais FXO e FXS

A diferença entre um canal FXO e um canal FXS é simplesmente qual ponta final da conexão provê o tom de discagem. Uma porta FXO não gera tom de discagem, ela recebe tom de discagem. Um exemplo comum é o tom de discagem gerado pela companhia de telefonia. Uma porta FXS gera tanto, o tom de discagem, como também o aumento da tensão para alertar a estação do usuário de uma ligação chamando. Ambas as interfaces provêem comunicação bi-direcional. (MEGGELEN et al, 2005).

Se o servidor *Asterisk* possuir uma porta FXO, pode-se conectar uma linha telefônica nesta porta. O *Asterisk* pode utilizar essa linha para efetuar e receber ligações telefônicas. Se o servidor *Asterisk* possuir uma porta FXS, pode-se conectar um telefone analógico no servidor e este telefone pode ser utilizado para atender ou efetuar chamadas telefônicas. (MEGGELEN et al, 2005).

As portas FXO e FXS são definidas, na configuração, pela sinalização que elas usam, como o oposto ao tipo físico que elas são. Assim, uma porta FXO será definida na configuração utilizando sinalização FXS, e uma porta FXS será definida utilizando sinalização FXO. Isto parece confuso até entender as razões para isto. Placas FX_ são nomeadas não de acordo ao que elas são, mas sim ao que é conectado a elas. Assim, uma placa FXS é uma placa que conecta a uma estação (por exemplo: um telefone analógico). Sendo assim, para que a estação funcione a placa FXS deve se comportar como uma central para a estação e utilizar a sinalização FXO para que a estação consiga se comunicar. Similarmente, uma placa FXO conecta a uma central (geralmente a central da empresa de telefonia pública), portanto ela deve se comportar como uma estação e usar a sinalização FXS. O modem é um exemplo clássico de um dispositivo FXO. (MEGGELEN et al, 2005).

Para este projeto foi utilizada uma placa FXO modelo X100P com *chipset* Motorola. Esta placa, que na verdade é um modem, possui *drivers* adaptados para utilizar a placa como um único dispositivo FXO.

A figura 14 abaixo mostra a placa utilizada no projeto:

Figura 14: Placa clone da X100P utilizada para comunicação com a PSTN.

Para que a placa citada acima seja reconhecida e suportada pelo FreeBSD e pelo *Asterisk*, é preciso instalar os *drivers* da placa. Esses *drivers* são instalados através do projeto zaptel. Para a instalação do projeto zaptel no FreeBSD os seguintes passos foram executados:

```
# cd /usr/ports/misc/zaptel
# make install clean
```

Pronto, o projeto zaptel está instalado. Porém, o projeto zaptel não possui um *tone zone* para o Brasil. Para adicionar um *tone zone* para o Brasil o arquivo

zonedata.c foi editado e foram acrescentadas as linhas referentes ao suporte ao *tone zone* Brasil, abaixo seguem os passos que foram executados para realizar essa tarefa:

```
# cd /usr/ports/misc/zaptel/work/zaptel-bsd-1.0/ztcfg
# vi zonedata.c
```

As seguintes linhas foram acrescentadas ao arquivo, logo abaixo da última zona China:

```
  { 21, "br", "Brazil", { 1000, 4000 },
  {
     { ZT_TONE_DIALTONE, "425" },
     { ZT_TONE_BUSY, "425/250,0/250" },
     { ZT_TONE_RINGTONE, "425/1000,0/4000" },
     { ZT_TONE_CONGESTION, "425/250,0/250,425/750,0/250" },
     { ZT_TONE_CALLWAIT, "425/50,0/1000" },
     { ZT_TONE_DIALRECALL, "350+440" },
     { ZT_TONE_RECORDTONE, "425/250,0/250" },
     { ZT_TONE_INFO, "950/330,1400/330,1800/330" },
     { ZT_TONE_STUTTER, "350+440" } },
  },
```

Após acrescentar as linhas acima e salvar o arquivo, falta só instalar novamente o software ztcfg com as alterações efetuadas. Para isso os seguintes comandos foram executados:

```
# make && make install
```

Pronto, o projeto zaptel está instalado e com suporte ao *tone zone* do Brasil.

Para que os *drivers* sejam carregados na inicialização do servidor *Asterisk*, é preciso adicionar a seguinte linha no arquivo /etc/rc.conf:

```
zaptel_enable=”YES”
```

O próximo passo é configurar o arquivo zapata.conf , que tem por finalidade controlar todas as portas reconhecidas pelos módulos zaptel.

Para o funcionamento deste projeto, as seguintes linhas foram acrescentadas ao arquivo /usr/local/etc/asterisk/zapata.conf :

```
[channels]
busydetect=4
busycount=5
context=from-analog
signalling=fxs_ks
callerid=asreceived
echocancel=yes
channel => 1
```

Os parâmetros utilizados na seção *channels* são descritos a seguir:

- busydetect: especifica quantos tons de ocupado antes de enviar um sinal de *Hangup*.
- busycount: especifica a intensidade do sinal de ocupado, na maioria dos países é 500msec ligado e 500msec desligado.
- context: especifica o contexto onde a porta FXO será incluída no arquivo extensions.conf.
- signalling: especifica a sinalização que a porta FXO irá receber.
- callerid: especifica o identificador de quem está chamando, esta opção depende muito da operadora de telefonia e do hardware utilizado. Nesta caso a opção *asreceived* utilizará o identificador enviado pela operadora de telefonia.
- echocancel: especifica se será utilizado algoritmos de cancelamento de eco.
- channel: especifica qual canal a ser utilizado na placa. Nesta caso só temos um canal, uma porta FXO.

O arquivo possui mais opções de configuração, porém não são relevantes para o desenvolvimento deste projeto. O arquivo zapata.conf completo encontra-se no anexo 1.

3.4 Instalação e Configuração do *Asterisk*

O *Asterisk* foi instalado via *ports*. Antes de efetuar a instalação do *Asterisk*, primeiro foi efetuada a atualização do *ports* para garantir a utilização da versão mais recente do *Asterisk* disponível no *ports*. Para isso foram utilizados os seguintes comandos:

portsnap fetch extract (necessário somente na primeira vez após a instalação do sistema operacional).
portsnap fetch update

Após a atualização do *ports* foi efetuada a instalação do *Asterisk* utilizando os seguintes comandos:

cd /usr/ports/net/asterisk
make -DWITHOUT_SQLITE -DWITHOUT_H323 -DWITHOUT_POSTGRES -DWITHOUT_RADIUS -DWITH_ZAPTEL install clean

Os parâmetros passados ao comando *make* acima são descritos a seguir:

- -DWITHOUT_SQLITE: Desabilita o suporte a SQLITE.
- -DWITHOUT_H323: Desabilita o suporte ao protocolo H323. Atualmente o mais utilizado é o protocolo SIP.
- -DWITHOUT_POSTGRES: Desabilita o suporte ao banco de dados PostGreSQL.
- -DWITHOUT_RADIUS: Desabilita o suporte a RADIUS.
- -DWITH_ZAPTEL: Habilita o suporte ao projeto Zaptel.

Após os comandos citados acima o *Asterisk* foi instalado e estava pronto para ser utilizado.

Para que o servidor *Asterisk* iniciasse durante a inicialização do computador, a seguinte linha foi adicionada ao arquivo /etc/rc.conf :

asterisk_enable=”YES”

Após a instalação do *Asterisk*, foi instalado o servidor de banco de dados MySQL. O MySQL é utilizado no projeto para armazenar as informações de bilhetagem do *Asterisk*. O padrão do *Asterisk* é armazenar essas informações em arquivos texto no formato CSV (separado por vírgulas). Por motivos de melhor gerenciamento e acesso às informações foi escolhido o armazenamento em banco de dados.

Para a instalação do servidor MySQL foi utilizado o comando abaixo:

```
# cd /usr/ports/databases/mysql50-server
# make install clean
```

Após executar os comandos acima o servidor MySQL foi instalado. Para que o servidor MySQL iniciasse durante a inicialização do computador a seguinte linha foi adicionada ao arquivo /etc/rc.conf :

mysql_enable=”YES”

O servidor MySQL, por padrão, após a instalação não tem senha configurada para o usuário administrador (root), para isso os comandos abaixo foram executados para criar essa senha:

```
# /usr/local/etc/rc.d/mysql.sh start (iniciando o servidor MySQL)
# mysqladmin –u root password ceub (atribuindo a senha “ceub” ao usuário root)
```

Pronto. O servidor MySQL está instalado e configurado.

Por motivos de licença do MySQL, o *Asterisk* não inclui em seu pacote o suporte padrão ao armazenamento das informações de bilhetagem no banco de dados

MySQL. Para solucionar esse problema foi instalado o pacote *asterisk-addons*, que provê suporte a vários serviços no *Asterisk* e entre eles o suporte ao MySQL.

Para a instalação do pacote *asterisk-addons* foram executados os seguintes comandos:

```
# cd /usr/ports/net/asterisk-addons
# make -DWITH_SAMPLE_CONFIG install clean
```

O parâmetro (-DWITH_SAMPLE_CONFIG) passado ao comando *make* acima habilita a instalação dos arquivos de configurações usados como exemplos.

Para que o *Asterisk* reconheça que o servidor MySQL está disponível para utilização, é necessário carregar o módulo de suporte ao MySQL durante a inicialização do *Asterisk*. Para isso a seguinte linha foi adicionada no arquivo /usr/local/etc/asterisk/modules.conf:

```
load => cdr_addon_mysql.so
```

O arquivo modules.conf completo encontra-se no anexo 2.

Após a instalação do pacote *asterisk-addons*, o *Asterisk* já estava pronto para armazenar as informações de bilhetagem no banco de dados MySQL.

Para armazenar essas informações é necessário uma base de dados com uma estrutura definida pelo *Asterisk*, com as tabelas no formato correto para o armazenamento. Para criar essa tabela foi criado um arquivo chamado *cdr.sql* com comandos SQL para criação da base de dados e das tabelas.

Abaixo, segue o conteúdo deste arquivo:

```
create database asteriskdrdb;
set storage_engine=myisam;

use asteriskcdrdb;
/*!40101 SET
@OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET
@OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
```

```
/*!40101 SET
@OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS,
UNIQUE_CHECKS=0 */;
/*!40014 SET
@OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

DROP TABLE IF EXISTS `cdr`;
CREATE TABLE `cdr`(
  `calldate` datetime NOT NULL default '0000-00-00 00:00:00',
  `clid` varchar(80) NOT NULL default '',
  `src` varchar(80) NOT NULL default '',
  `dst` varchar(80) NOT NULL default '',
  `dcontext` varchar(80) NOT NULL default '',
  `channel` varchar(80) NOT NULL default '',
  `dstchannel` varchar(80) NOT NULL default '',
  `lastapp` varchar(80) NOT NULL default '',
  `lastdata` varchar(80) NOT NULL default '',
  `duration` int(11) NOT NULL default '0',
  `billsec` int(11) NOT NULL default '0',
  `disposition` varchar(45) NOT NULL default '',
  `amaflags` int(11) NOT NULL default '0',
  `accountcode` varchar(20) NOT NULL default '',
  `uniqueid` varchar(32) NOT NULL default '',
  `userfield` varchar(255) NOT NULL default '',
  KEY `calldate` (`calldate`),
  KEY `dst` (`dst`),
  KEY `accountcode` (`accountcode`)
);
```

```
/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET
CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET
CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET
COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;
```

Após a criação do arquivo, faltava então executar esses comandos SQL dentro do servidor MySQL. Para isso o seguinte comando foi utilizado:

```
# mysql –u root –p < cdr.sql
```

Após digitar a senha do usuário root, os comandos SQL foram executados com sucesso e a base de dados foi criada.

Toda a estrutura necessária para o armazenamento das informações de bilhetagem do *Asterisk* está criada e pronta para ser utilizada. Para isso o *Asterisk* deve ser configurado para utilizar o servidor MySQL para armazenamento dessas informações. Para tal, dois arquivos devem ser alterados para ativar esse suporte. O primeiro deles é o arquivo /usr/local/etc/asterisk/cdr_mysql.conf. Neste arquivo são definidos parâmetros para conexão com o banco de dados MySQL.

Abaixo, segue o conteúdo desse arquivo:

```
[global]
hostname=localhost
dbname=asteriskcdrdb
table=cdr
password=ceub
user=root
port=3306
```

sock=/tmp/mysql.sock

O parâmetro hostname define qual o hostname o servidor onde o MySQL está rodando.

O parâmetro dbname define qual a base de dados será utilizada.

O parâmetro table define qual a tabela será utilizada.

O parâmetro password define a senha para conexão.

O parâmetro user define o usuário utilizado para conexão.

O parâmetro port define a porta para conexão com o servidor MySQL.

O parâmetro sock define o socket para conexão com o servidor MySQL.

O próximo, e último, arquivo a ser alterado para que o *Asterisk* possa utilizar o MySQL para armazenar as informações de bilhetagem é o arquivo /usr/local/etc/asterisk/cdr.conf. Abaixo, segue os campos adicionados ao arquivo para provê suporte ao MySQL:

[mysql]
mysqlcfg => /usr/local/etc/asterisk/cdr_mysql.conf

O parâmetro mysqlcfg define o caminho do arquivo com as configurações necessárias para conexão com o servidor MySQL.

O arquivo possui mais informações, porém não são relevantes para o funcionamento do projeto. O arquivo cdr.conf completo encontra-se no anexo 3.

3.5 Configuração de Ramais SIP no Asterisk

Para entender como funcionam os ramais SIP no *Asterisk*, é preciso conhecer o que é um plano de discagem no *Asterisk*.

3.5.1 Plano de discagem

É o lugar onde se configura o comportamento de todas as conexões, considerando o plano mestre do controle de fluxo ou de execuções para todas as operações, define como as ligações entrantes e saintes serão tratadas e distribuídas.

O *Asterisk* possui um plano de discagem extremamente poderoso e flexível, o que é um grande diferencial em relação aos sistemas de PABX proprietários.

Algumas das vantagens do plano de discagem do *Asterisk* são citadas abaixo:

- Segurança, restringindo os ramais para ligações DDD, DDI e celular;
- Autenticação, solicitando senha para realizar chamadas;
- Roteamento, distribuindo as chamadas para rotas de menor custo;
- Entroncamento com operadoras VoIP, PABX convencionais, tronco de celular;
- URA rica e flexível;
- Regras por horários, definindo fluxos de acordo com a hora;
- Macros, definindo seqüência de comandos para determinadas situações.

A configuração dos planos de discagem são efetuadas no arquivo /usr/local/etc/asterisk/extensions.conf.

Para o projeto foram criados os seguintes planos de discagem:

```
[from-sip-internal]
exten => _XXXXXXXX,1,Dial(Zap/1/${EXTEN})
exten => _XXXXXXXX,2,Congestion
exten => _XXXXXXXX,3,Hangup
exten => 1945,1,Dial(SIP/1945,1)
exten => 1946,1,Dial(SIP/1946,1)


[from-analog]
exten => s,1,Answer()
exten => s,2,Dial(SIP/1945,30)
```

```
exten => s,4,Hangup
exten => i,1,Hangup
exten => h,1,Hangup
```

Acima foram criados dois contextos, *from-sip-internal* e *from-analog*. Primeiro é preciso entender o que é um contexto. De uma maneira bem simples, um contexto é um grupo de extensões. Cada extensão deve existir dentro de um contexto. Existem outras aplicações para contextos do que somente agrupar extensões, porém não são relevantes para este projeto.

A forma geral para definir uma extensão no arquivo extensions.conf é:

```
exten => númerodaextensão,prioridade,ação
```

Abaixo segue a descrição das extensões criadas para o contexto *from-sip-internal*:

- exten => _XXXXXXXX,1,Dial(Zap/1/${EXTEN}): esta extensão permite efetuar ligação para qualquer número (o "_" indica ao *Asterisk* que os próximos caracteres não serão interpretados literalmente. Os "X" funcionam como coringas (*wildcards*) que indica ao *Asterisk* que qualquer número de 0 a 9 será aceito. Esta extensão utiliza o dispositivo Zap/1, que é a porta FXO configurada no canal 1, portanto podendo efetuar ligações utilizando a PSTN.
- exten => _XXXXXXXX,2,Congestion: esta extensão define se caso o número discado na primeira extensão está ocupado, ou a linha telefônica utilizada pelo canal 1 está ocupada .
- exten => _XXXXXXXX,3,Hangup: caso a linha esteja ocupada, como verificado na extensão 2, esta extensão irá desligar a chamada.
- exten => 1945,1,Dial(SIP/1945,1): esta extensão irá discar para o ramal SIP 1945 caso seja efetuada uma ligação para o número 1945.
- exten => 1946,1,Dial(SIP/1946,1): esta extensão irá discar para o ramal SIP 1946 caso seja efetuada uma ligação para o número 1946.

Abaixo segue a descrição das extensões criadas para o contexto *from-analog:*

- exten => s,1,Answer(): esta extensão serve para que o *Asterisk* capture a chamada.
- exten => s,2,Dial(SIP/1945,30): após capturar a chamada, ele irá transferir a chamada para o ramal SIP 1945 por 30 segundos.
- exten => s,4,Hangup: caso o ramal SIP 1945 não atenda em 30 segundos, a chamada será desligada.
- exten => i,1,Hangup: caso seja discado um número inválido (“i”), o Asterisk irá desligar a chamada.
- exten => h,1,Hangup: esta extensão é recomendada para quando for desligada uma ligação, seja efetuada a remoção correta das rotas das ligações no *Asterisk*.

O arquivo possui mais opções de configurações, porém não são relevantes para esta parte do projeto. O arquivo extensions.conf completo encontra-se no anexo 4.

Após criar os planos de discagem, foram criados os seguintes ramais SIP no arquivo /usr/local/etc/asterisk/sip.conf:

```
[1945]
type=friend
username=1945
secret=ceub
canreinvite=no
disallow=all
callerid="Danilo Lara"
host=dynamic
context=from-sip-internal
nat=never
allow=ulaw
```

```
[1946]
type=friend
username=1946
secret=ceub
canreinvite=no
disallow=all
callerid="Ata"
host=dynamic
context=from-sip-internal
nat=never
allow=ulaw
```

Abaixo segue a descrição dos parâmetros utilizados para os ramais SIP:

- type: define o tipo de usuário. Existem três tipos: *user*, *peer* e *friend*. O tipo *user* só pode efetuar chamadas. O tipo *peer* só pode receber chamadas. O tipo *friend* pode efetuar e receber chamadas.
- username: define o nome de usuário para autenticação.
- secret: define a senha para o usuário.
- canreinvite: define se os dispositivos poderão efetuar o *reinvite* (maiores detalhes sobre a opção *reinvite* estão na seção 3.6.1)
- disallow: desabilita todos os *codecs*.
- callerid: define a identificação de chamada do usuário. Geralmente utilizado o nome do usuário que utiliza o ramal. Por exemplo: Danilo Lara.
- host: define o endereço do *host* do usuário. Pode ser um endereço estático ou dinâmico.
- context: define qual contexto o usuário será incluído no arquivo extensions.conf.
- nat: define se o dispositivo SIP está atrás de um NAT ou não.
- allow: define qual *codec* será permitido a utilização.

Após adicionar as linhas acima no arquivo sip.conf , os dois ramais SIP necessários para o projeto foram criados e estavam prontos para serem utilizados.

O arquivo possui mais opções de configurações, porém não são relevantes para esta parte do projeto. O arquivo sip.conf completo encontra-se no anexo 5.

3.6 Bilhetagem no *Asterisk*

O *Asterisk* guarda um completo registro das chamadas efetuadas e recebidas, conhecido como CDR (*Call Detail Records*). Essas informações podem ser armazenadas em um arquivo texto ou em um banco de dados. Usando essas informações, é possível monitorar a utilização do sistema *Asterisk*, buscando por padrões ou anomalias que possam causar impactos ou problemas aos negócios. (GOMILLION, 2005).

É possível comparar essas informações com as informações contidas na conta telefônica enviada pela empresa de telefonia e verificar se elas estão corretas ou não. Essas informações permitem que seja feita uma análise do tráfego de chamadas, gerar relatórios dos dez números mais chamados e etc.

Mais do que isso, é possível verificar qual o tempo de cada chamada. Pode-se verificar quantas chamadas um agente (ramal) específico, no caso de um *call center*, atende e comprar com a média dos outros agentes. Pode-se ainda, identificar abusos no uso do serviço de chamada de longa distância. Enfim, as possibilidades de uso das informações das chamadas registradas pelo *Asterisk* são várias. (GOMILLION, 2005).

Por padrão, o *Asterisk* inclui um módulo chamado *cdr_csv*. Após a instalação, o *Asterisk* registra as informações de todas as chamadas entrantes e saintes. Essas informações são armazenadas em um arquivo texto separadas por vírgula, o chamado formato CSV (*Comma-Separated Value*). Este arquivo CSV é localizado no diretório /var/log/asterisk/cdr-csv. Todas as informações estão no arquivo *Master.csv* , e alguns canais podem ser configurados para armazenar essas informações em outros arquivos. (GOMILLION, 2005).

A vantagem de utilizar um arquivo CSV é a simplicidade. Basta apenas compilar e instalar o *Asterisk* que essa função já está disponível. Nenhuma configuração adicional é necessária, nenhum tráfego adicional é gerado na rede e nenhum outro serviço precisa ser instalado no servidor.

As informações de CDR armazenadas pelo *Asterisk* são as seguintes:

- accountcode: definido se a aplicação *SetAccount*() for utilizada ou se for configurada para um canal no arquivo de configuração do canal, exemplo: arquivo sip.conf.
- src: quem efetuou a chamada.Uma string de até 80 caracteres.
- dst: o destino da chamada.
- dcontext: o contexto de destino
- clid: identificação do chamador.
- channel: canal utilizado por quem efetua a chamada.
- dstchannel: canal utilizado para o destino da chamada.
- lastapp: a última aplicação executada no canal.
- lastdata: o último argumento para a última aplicação no canal.
- start: início da chamada.
- answer: quando a chamada foi atendida.
- end: o final da chamada.
- duration: o tempo de duração da chamada, em segundos. Diferença entre o tempo de início e o término.
- billsec: tempo permanecido na chamada desde quando atendida até ser desligada, em segundos.
- disposition: o que aconteceu com a chamada. (atendida, não atendida, ocupado).
- amaflags: quais *flags* foram utilizadas (documentação, bilhetagem, ignorar, etc.) especificadas basicamente por canal.
- userfield: nome do usuário definido pelo comando *SetCDRUserField*.

Utilizar o formato CSV para armazenar as informações de CDR é simples, porém não é nada simples de entender e consultar essas informações. A maneira mais fácil de fazer isso é armazenando as informações de CDR em um bando de dados.

O *Asterisk*, nativamente, suporta armazenar as informações de CDR em SQLite, PostGreSQL e unixODBC. O suporte ao MySQL, por questões de licença do MySQL, é fornecido separadamente, com a instalação do pacote *asterisk-addons*. Para este projeto foi escolhido o armazenamento das informações de CDR em um banco de dados, o MySQL. A escolha pelo MySQL se deu pelo fato de ele ser mais "leve" e simples de configurar do que o PostGreSQL e também porque a linguagem

de programação utilizada no projeto possui um suporte melhor e mais simples ao MySQL.

3.6.1 Desafios para o armazenamento de CDR

O *Asterisk* irá armazenar as informações sobre as chamadas que passam por ele, porém ele não poderá armazenar informações que não são informadas. Por exemplo, se existirem dispositivos IP cuja opção de *reinvite* esteja habilitada, após o *Asterisk* efetuar a conexão da chamada, os dispositivos não irão mais precisar da assistência do *Asterisk*. Se esses dispositivos irão fornecer ou não as informações das chamadas, o *Asterisk* não tem como controlar isso. Portanto, se as informações de CDR são importantes, certifique-se que os dispositivos IP não estarão habilitados a efetuar o *reinvite*.

3.7 Interface WEB

Com o intuito de facilitar a administração do *Asterisk* foi desenvolvida uma interface WEB que permite a criação, alteração e consulta a informações sobre os ramais disponíveis no servidor *Asterisk*.

A principal vantagem em utilizar a interface WEB é que o usuário responsável pelo IPBX não precisa conhecer o funcionamento interno do *Asterisk*, como os ramais funcionam e como configurá-los, tudo isto é feito de maneira bem simples através da interface WEB onde o usuário simplesmente escolhe as opções disponíveis de acordo com padrões pré-definidos. Com isso qualquer usuário pode adicionar, alterar e consultar informações sobre os ramais, desde que possua acesso à interface WEB. Desta maneira, o administrador do servidor *Asterisk* não é obrigado a estar disponível a qualquer momento para efetuar uma simples alteração nos ramais, por exemplo, podendo até sair de férias sem ter que ficar à disposição da empresa.

Outra vantagem em utilizar a interface WEB é que o usuário não precisa acessar o console do servidor *Asterisk*, minimizando assim prováveis problemas de segurança.

Para o desenvolvimento do código da interface WEB foi utilizado o *framework Ruby on Rails*. A linguagem de programação utilizada foi o *Ruby*. O

principal motivo da escolha do *Ruby* como linguagem de programação é a sua simplicidade de leitura e entendimento do código gerado. *Ruby* é uma linguagem de *script* orientada a objetos que vem crescendo muito rapidamente nos últimos anos. A utilização do *framework Ruby on Rails* permite um desenvolvimento para sistemas via WEB rápido e simples, possibilitando um ganho de produtividade considerável.

Hoje em dia o novo conceito de WEB 2.0, várias tecnologias tem surgido para suprir as novas necessidades do mercado. Interatividade, integração, contribuição são as novas palavras da nova internet (WEB 2.0).

A Adobe tem a tecnologia do *Flash*, que está no mercado há mais de 10 anos, o qual ficou conhecido por fazer sites com animações inovadoras. O problema era que ele não supria as necessidades do mercado de aplicativos ricos (RIA ou WEB 2.0), mesmo tendo os componentes e comandos específicos para manipulação de informações. Com isso eles resolveram tirar o estigma de programa para fazer somente animações e criaram o *Flex* que é focado em aplicativos ricos para internet.

Atualmente mais de 95% dos computadores conectados na internet tem o *Flash Player* instalado e estão aptos para rodar tanto sites/aplicativos feitos com a plataforma *Flash*.

Para a camada de visualização foi utilizada a tecnologia Adobe *Flex* 2. O Adobe *Flex* 2 foi utilizado para o *design* das telas da interface WEB.

4 TESTES E RESULTADOS

4.1 Criação e Alteração de ramais via WEB

Esta é uma tarefa simples de ser feita no servidor *Asterisk*, porém somente usuários que conheçam como funcionam os ramais no *Asterisk*, ou seja, sua arquitetura e tipos de ramais suportados poderiam completar esta tarefa. Além disto, toda a configuração é feita através de arquivos texto diretamente na *console* de administração do servidor FreeBSD, o que para muitos usuários não acostumados com sistemas Unix é uma tarefa extremamente complicada. Com o desenvolvimento de uma interface WEB, como a desenvolvida neste projeto, isto se torna fácil, simples e rápido de ser feito, bastando apenas poucos cliques com o mouse para efetuar qualquer alteração, bastando apenas um simples treinamento das opções disponíveis e quais passos devem ser executados para completar esta tarefa.

Para a criação e alteração de ramais via WEB, o usuário precisará de uma senha de acesso à interface. Essa senha é cadastrada no banco de dados MySQL que roda no servidor *Asterisk*. O anexo 6 contém um script SQL que cria a base de dados de autenticação de usuários. O usuário padrão criado pelo script SQL é o seguinte:

Usuário: admin
Senha: admin

Essa senha pode ser alterada através do módulo de cadastro de usuários. Além de possibilitar a troca da senha é possível também acrescentar novos usuários de acordo com a necessidade de administração do PBX.

Para criar a base de dados de autenticação basta importar o script SQL contido no anexo 6 para o servidor MySQL.

Após criada a base de dados de autenticação, a interface já está pronta para ser utilizada.

Para acessar a interface WEB basta acessar o seguinte endereço no navegador de internet: http://192.168.1.2:3000/bin/asterisk.html

Após acessar o endereço citado acima, a seguinte tela será exibida:

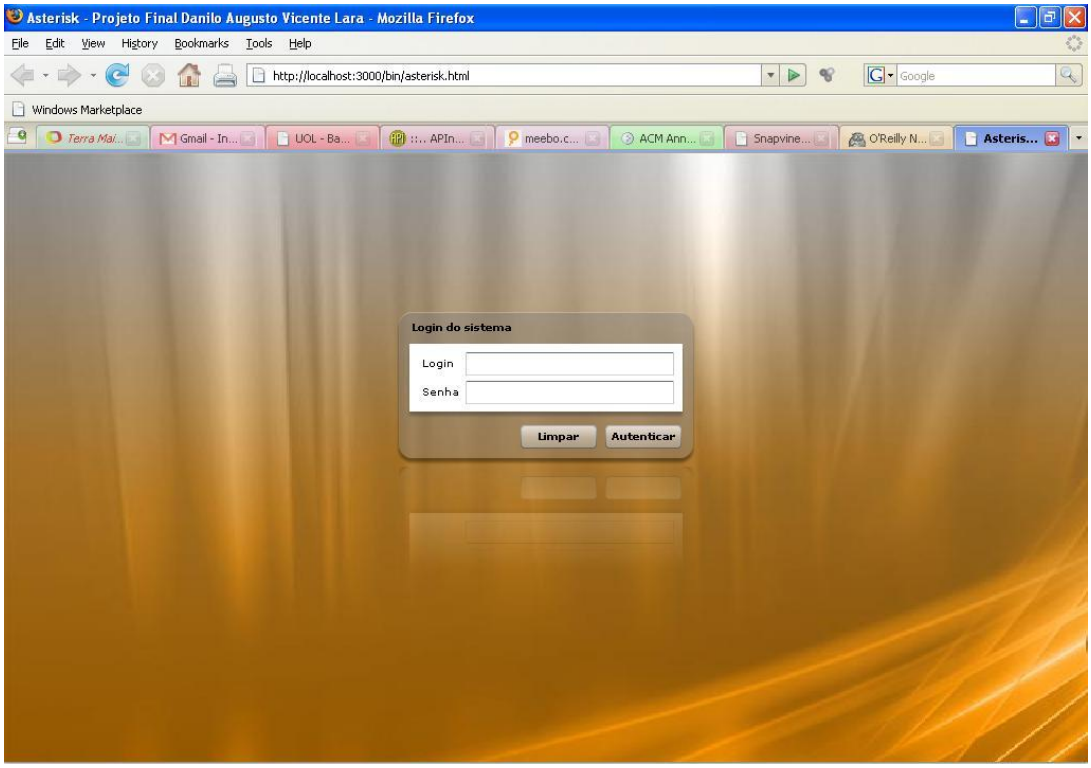

Figura 15: Tela de login do sistema

Após preencher os campos login, senha e clicar em autenticar o usuário estará, caso forneça as informações de credenciais corretas, autenticado e está autorizado a usar o sistema.

Após efetuar a autenticação, a seguinte tela será exibida:

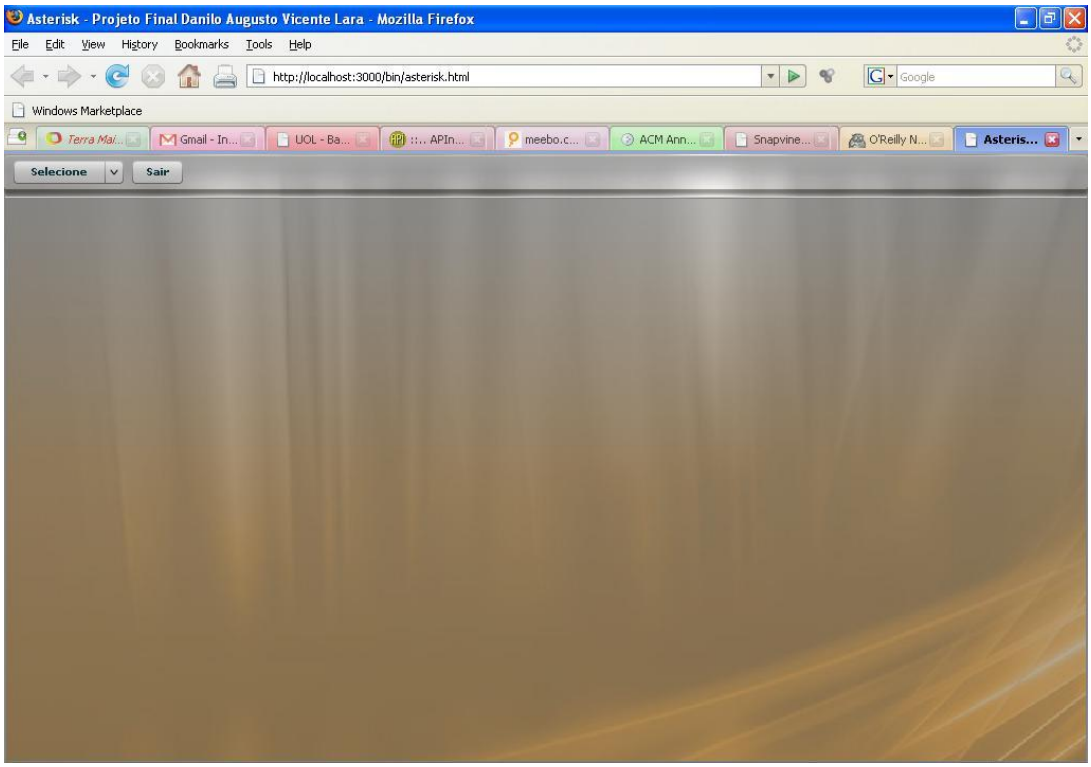

Figura 16: Tela após efetuar a autenticação

Para efetuar cadastramento de novos ramais ou alteração de ramais já existentes, deverá ser selecionado o menu de administração de ramais, clicando na seta para baixo ao lado do botão “Selecionar”. As seguintes opções serão exibidas:

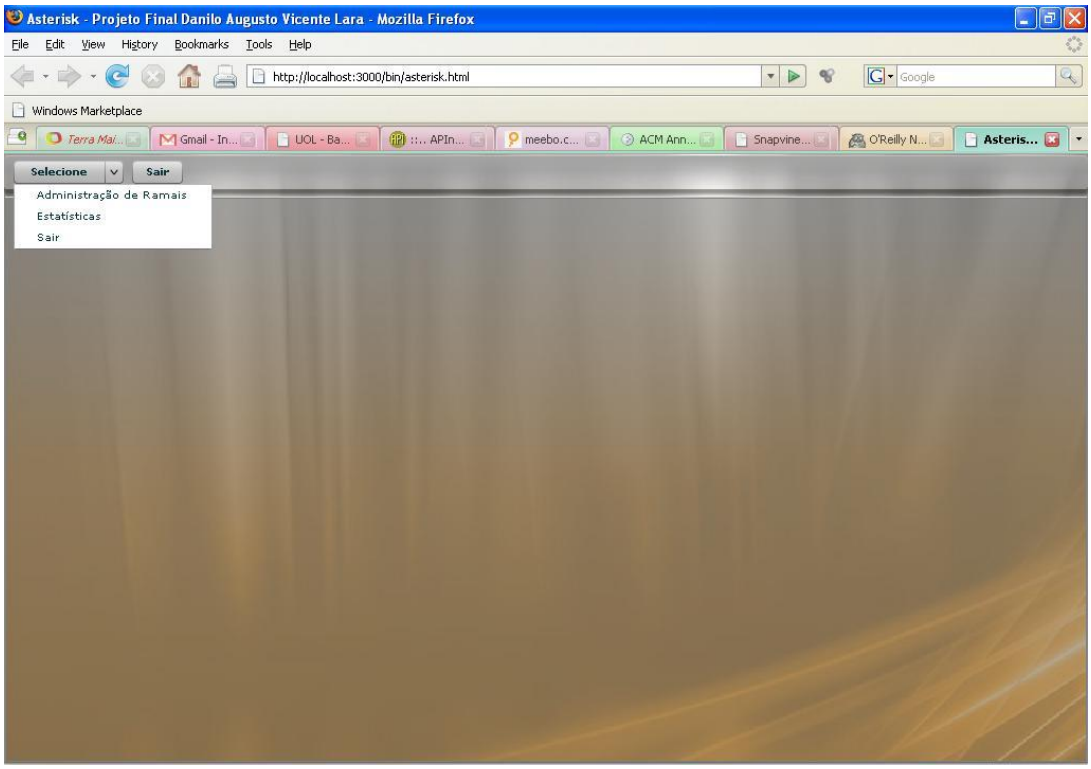

Figura 17: Menu de opções.

Para a criação e alteração de ramais, deverá ser selecionada a opção "Administração de Ramais". Após selecionar a opção "Administração de Ramais", a seguinte tela será exibida:

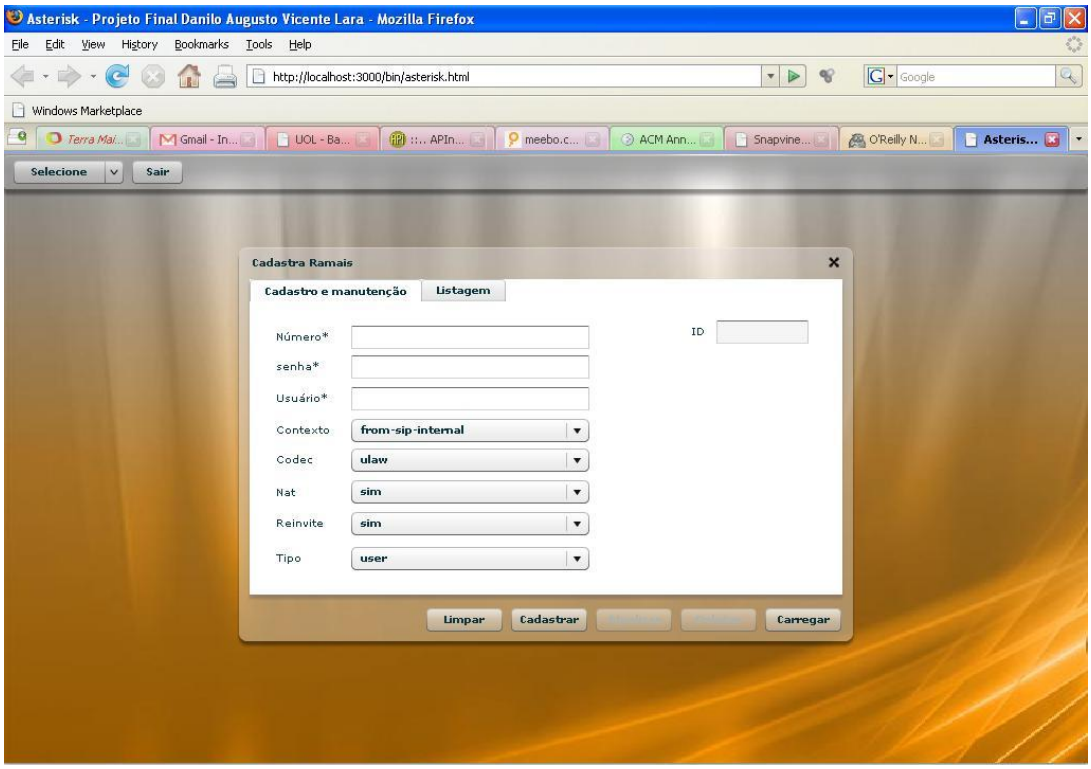

Figura 18: Tela de cadastramento e manutenção de ramais.

Nesta tela, o usuário terá a opção de cadastrar novos ramais. Os campos marcados com um "*" são de preenchimento obrigatório. Após preencher os campos necessários, para criar o novo ramal o usuário deve clicar no botão "Cadastrar". Será exibida uma tela dizendo se a operação foi concluída com sucesso ou não. Caso não tenha sido concluído com sucesso, os campos que estiverem com problemas serão marcados em vermelho, bastando passar o mouse sobre o campo para verificar qual foi o erro encontrado.

Nesta mesma tela existe a opção "Listagem", onde todos os ramais cadastrados através da interface WEB serão listados. A figura 19 mostra um exemplo da tela com a listagem dos ramais cadastrados.

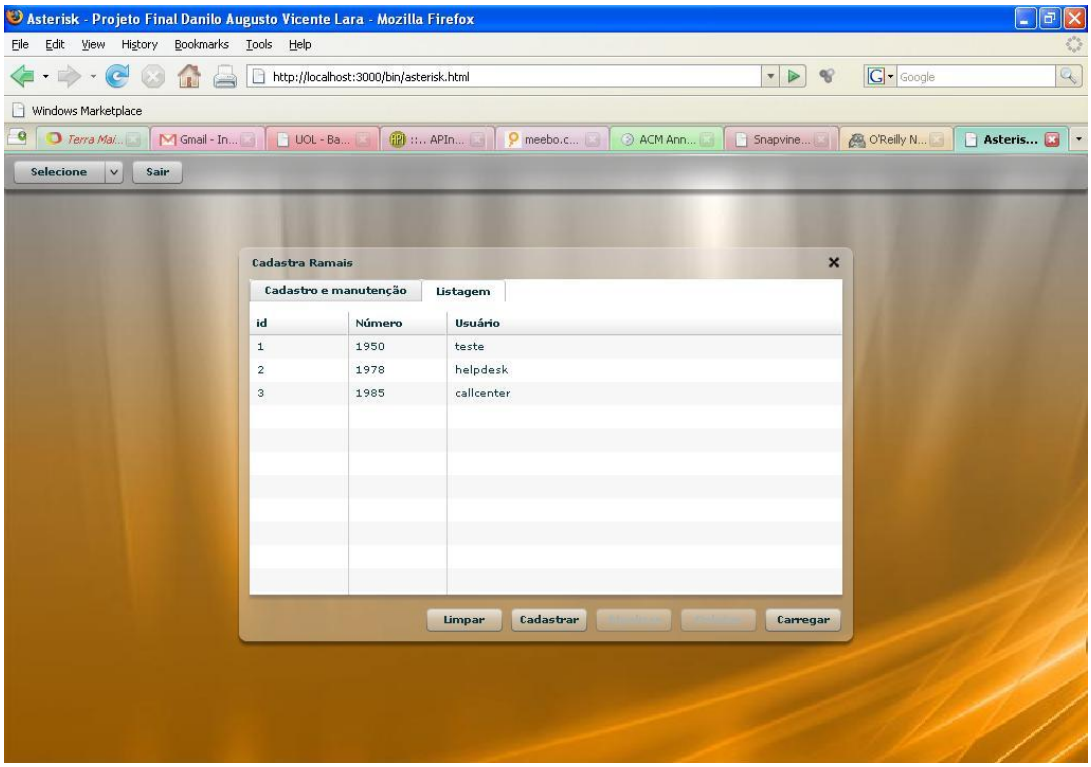

Figura 19: Listagem dos ramais cadastrados.

Para alterar um ramal já existente, o usuário deverá selecionar o ramal desejado na tela de listagem clicando em cima do ramal que se deseja alterar. A figura 20 mostra a tela que será exibida após clicar em um ramal na tela de listagem de ramais.

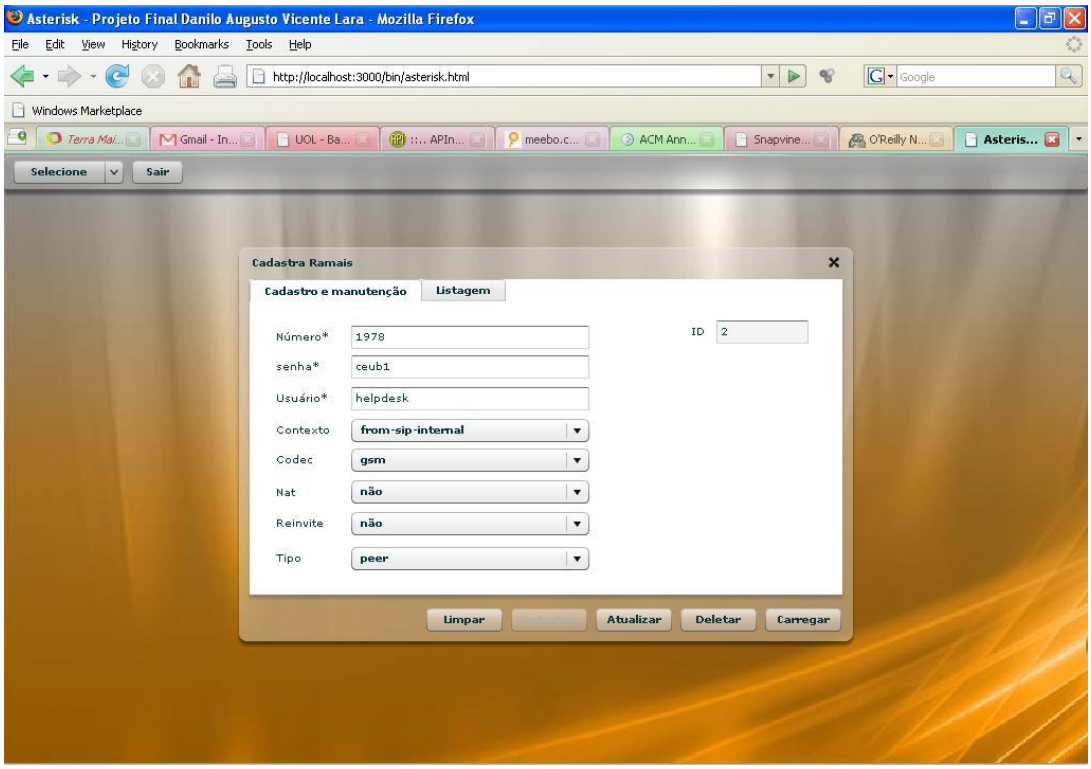

Figura 20: Tela de alteração de ramal.

O ramal selecionado estará disponível para alteração, podendo ser alterado qualquer campo disponível na tela de “Cadastro e manutenção”. Após efetuar as alterações, o usuário deverá clicar no botão “Atualizar” para gravar as alterações efetuadas.

Depois de criar ou alterar os ramais já existentes, o usuário deve utilizar o menu “Gerar arquivo” para poder gerar os arquivos com as informações dos ramais no formato que o *Asterisk* reconhece e clicar no botão “Gerar arquivo” e depois clicar no botão “Reiniciar” para que as alterações sejam efetuadas. Este menu foi disponibilizado com o intuito de tornar o sistema independente de sistema operacional, podendo ser executado tanto em FreeBSD quanto em Linux, bastando apenas que o usuário administrador informe o caminho correto dos arquivos solicitados pela Interface.

O código fonte utilizado para a criação da aplicação WEB se encontra no anexo 7.

4.2 Consulta a informações de bilhetagem via WEB

A grande contribuição desta parte da interface é a captura e organização das informações sobre as chamadas processadas pelo servidor *Asterisk*. A extração destas informações sem o auxílio de uma ferramenta como esta que foi desenvolvida é um pouco complexa inclusive para quem conhece bem como o *Asterisk* funciona, imagina para quem não conhece. Para simplificar essa consulta, a interface fornece opções de campos a serem exibidos, bem como gráficos com informações de cada ramal, o que gera uma visualização simples e direta das informações.

O *Asterisk* guarda registros de todas as chamadas efetuadas e recebidas por ele. Para este projeto essas informações foram armazenadas em um banco de dados para facilitar a consulta e gerenciamento.

Para ter acesso à essas informações, o usuário deverá selecionar o módulo "Estatísticas", disponível ao clicar na seta para baixo ao lado do botão "Selecione".

Após selecionar o módulo de estatísticas, a seguinte tela será apresentada:

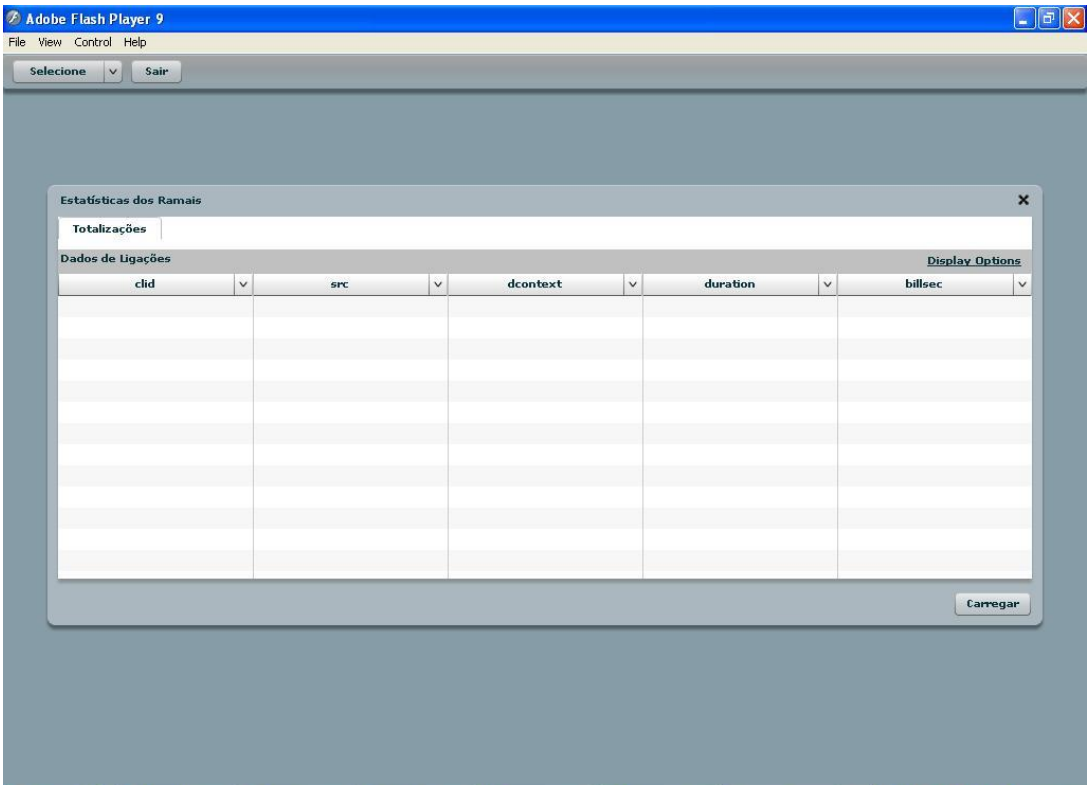

Figura 21: Tela principal do módulo de estatísticas.

Nesta tela o usuário pode verificar as informações sobre as chamadas efetuadas e recebidas. Para verificar as informações referentes a um ramal específico, basta selecionar o ramal e depois clicar na seta para baixo ao lado do campo desejado e escolher a opção agrupar (Group). Existe também a opção de organizar as informações em ordem de data crescente ou decrescente.

O módulo de estatísticas possui a facilidade de customização das informações apresentadas, onde o usuário pode selecionar quais campos ele deseja que sejam mostrados na interface. Para poder escolher esses campos o usuário deve clicar em "Display Options", no canto superior direito da interface.

Após clicar em "Display Options", a seguinte tela será apresentada:

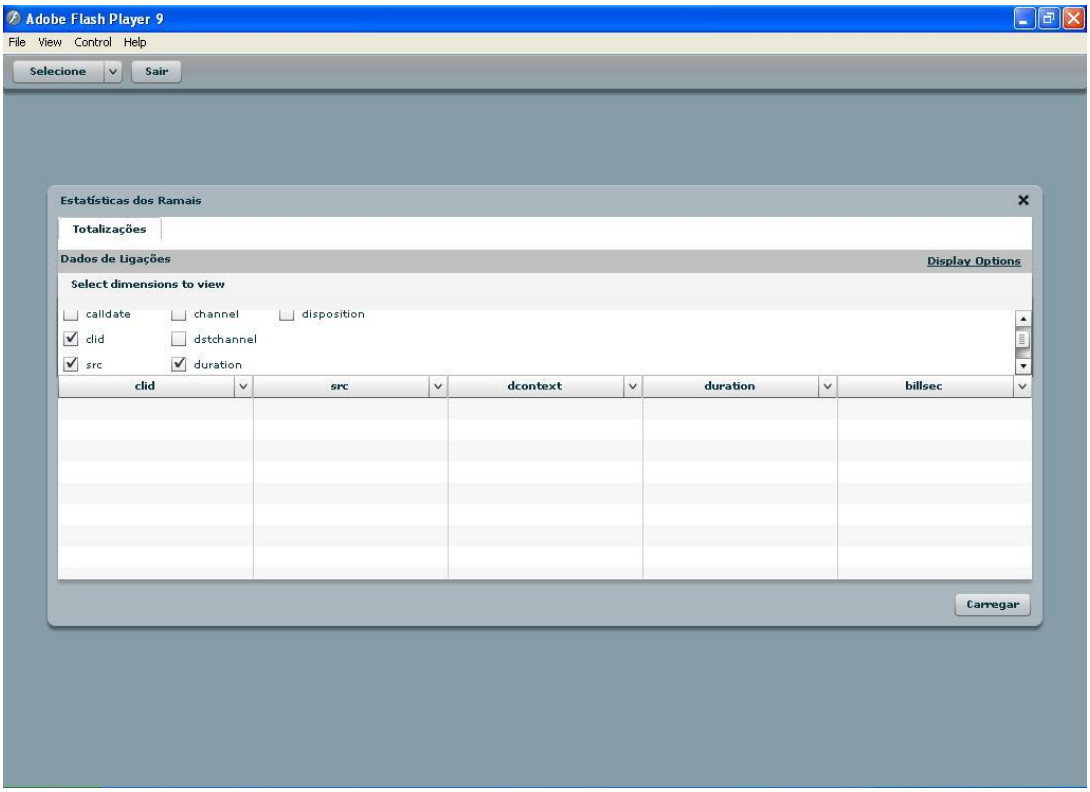

Figura 22: Tela de opções para a visualização.

Serão mostrados os campos que o *Asterisk* armazena com informações das chamadas. Os campos podem ser marcados ou desmarcados de acordo com a informação que se deseja obter.

4.3 Comunicação com a PSTN através de um *Soft Phone*

Para esta parte do projeto, a maior dificuldade foi a escolha do modem a ser utilizado para esta tarefa. Foi testado um modem ISA US Robotics Hardmodem, conhecido por ser um modem muito bom e de fácil reconhecimento pelos sistemas operacionais modernos. Porém após pesquisar sobre o hardware suportado pelo *Asterisk* foi descoberto que o mesmo só reconhece modems PCI e com chipsets Intel Ambient 3200 ou Motorola. Essas informações podem ser encontradas no site http://www.voip-info.org. Após escolher o modem, outra dificuldade foi encontrar locais que vendiam este modem, que foi comprado no site Mercado Livre (http://www.mercadolivre.com.br).

Para a comunicação com a rede pública de telefonia foi utilizada uma placa que fornece uma porta FXO, a placa clone da X100P que custa em média cem reais, conforme mostra a figura 23.

Figura 23: Placa clone da X100P.

Com a utilização desta placa o *Asterisk* reconhece que existe uma porta FXO disponível para utilização. Após conectar a linha telefônica na placa, o *Asterisk* já está habilitado a efetuar e receber ligações através da rede de telefonia pública. A figura 24 mostra o diagrama utilizado para esta parte do projeto.

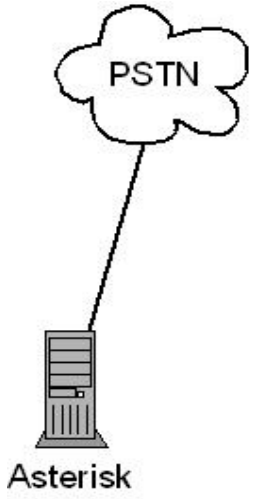

Figura 24: Diagrama de conexão com a PSTN.

Após instalar e configurar a placa que disponibiliza uma porta FXO para o servidor *Asterisk*, foram efetuados testes de comunicação com a rede pública de telefonia através de um *soft phone* instalado em outro computador. O *soft phone* utilizado foi o X-Lite (versão gratuita) da X-TEN. No *soft phone* foi configurado o ramal SIP a ser utilizado para efetuar e receber as ligações. Além disto, o *soft phone* foi configurado para utilizar o *codec* ulaw e foram desabilitados todos os outros *codecs*, o que permitiu um ganho de desempenho na inicialização do *soft phone*. A figura 25 mostra a interface do *soft phone* X-Lite.

Figura 25: Soft phone X-Lite.

As configurações efetuadas no *soft phone* constam no anexo 8.

A qualidade da voz entre a comunicação do *soft phone* com a rede de telefonia pública foi satisfatória.

4.4 Comunicação com um *ATA Phone* através de um *Soft Phone*

Para a comunicação com um telefone analógico através do *Asterisk*, é preciso que algum dispositivo faça a conversão do sinal analógico para digital. Para este projeto o dispositivo utilizado foi um ATA (*Analog Telephone Adapter*). O ATA utilizado foi um do modelo Linksys PAP2, conforme mostrado na figura 26. Este modelo suporta o protocolo SIP, que é o protocolo utilizado pelos ramais neste projeto.Uma dificuldade encontrada com a utilização deste ATA é que este modelo, o

Linksys PAP2 é um modelo bloqueado para funcionar somente na operadora VoIP Vonage, dos Estados Unidos, portanto para utilização com qualquer operadora VoIP deve ser utilizado o modelo Linksys PAP2-NA, que é um modelo que não possui este bloqueio. Para o projeto é possível efetuar uma ligação do *soft phone* para o ATA, porém não é possível ligar do ATA para o *soft phone* por causa do bloqueio do equipamento.

Figura 26: ATA Linksys PAP2.

As configurações efetuadas no ATA para que o mesmo funcionasse com o ramal SIP desejado constam no anexo 9.

Após configurar o ATA, o cenário mostrado na figura 27 foi montado.

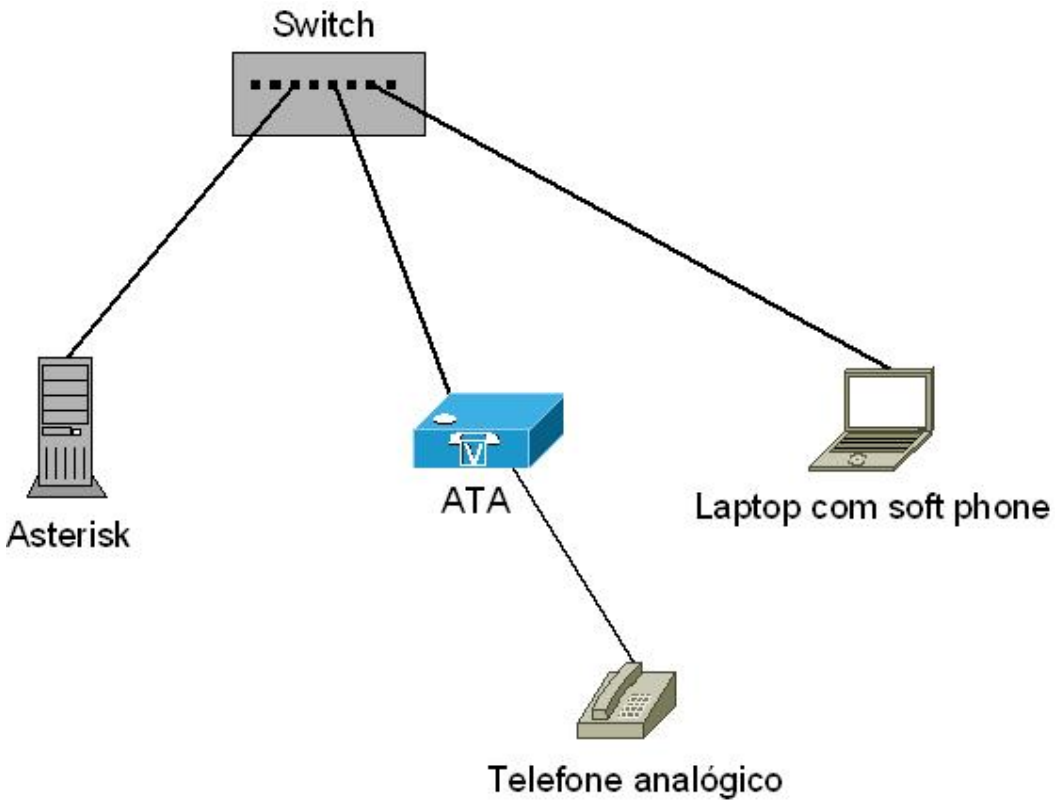

Figura 27: Cenário utilizado para comunicação entre o telefone analógico e o *soft phone*

No cenário acima, o dois ramais SIP utilizados foram configurados em seus respectivos telefones e registrados no servidor *Asterisk*. Quando o *soft phone* e o ATA são ligados, os mesmos efetuam o processo de registro do ramal SIP que está configurado no equipamento, só após esse registro é que o servidor *Asterisk* fará o roteamento das chamadas destinadas a estes ramais.

A qualidade da voz entre a comunicação do *soft phone* com o telefone analógico, através do ATA, foi satisfatória.

5 CONCLUSÃO

De acordo com as pesquisas e testes realizados conclui-se que a administração do *Asterisk* pode ser facilitada. A arquitetura modular do *Asterisk* torna o sistema bastante flexível e poderoso.

Conclui-se também que é possível utilizar o *Asterisk* para prover comunicação com a rede de telefonia pública, utilizando hardware simples e de baixo custo.

Outro item que foi verificado neste projeto é a possibilidade de utilizar telefones analógicos juntamente com o *Asterisk*, utilizando um dispositivo para efetuar a conversão do sinal analógico para digital. Isso permite que os telefones antigos possam ser utilizados como ramais de um IPBX, evitando assim gastos com aquisição de telefones IP.

Outro ponto que merece ser destacado é a não dependência de empresas de suporte e manutenção a PABXs proprietários. Qualquer alteração no IPBX pode ser realizada por um usuário que possua acesso ao sistema e conheça seu funcionamento.

Outro item que merece ser destacado é a viabilidade do *Asterisk*, que ficou provado durante o projeto que o mesmo é um software que permite criar um *gateway* VoIP ou um PABX comum com vários recusros e opções muitas vezes não encontradas em sistemas proprietários. Além disto, o *Asterisk* possui grande documentação disponível e uma grande comunidade de suporte e de usuários, inclusive grandes empresas o utilizam.

A tecnologia de voz sobre IP deverá crescer consideravelmente nos próximos anos, permitindo maior integração de serviços e redes de comunicações. O projeto mostrou a integração entre a rede de telefonia e a rede de dados utilizando um *gateway* para prover essa integração, o *Asterisk*.

Propõem-se como projetos futuros:

- A implementação de um IPBX simulando a interligação entre matriz e filial através do protocolo IAX (*Inter-Asterisk Exchange*);
- Criação de outros módulos para a interface WEB para configuração de, por exemplo, o sistema de música em espera e correio de voz;
- Criação de um módulo para definir níveis de privilégios para os usuários, permitindo criar usuários com autorização para algumas tarefas somente;
- Testes com outros equipamentos ATA para verificar sua compatibilidade com o *Asterisk*.
- Estudo comparativo entre os *codecs* suportados pelo *Asterisk*, analisando qual possui melhor qualidade de voz e qual utiliza menos largura de banda.

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. Rio de Janeiro: Elsevier, 2005.

WALLINGFORD, Ted. *Switching to VoIP*. 1. ed. Sebastopol: O'Reilly Media Inc, 2005.

GOMILLION, David; DEMPSTER, Barrie. *Building Telephony Systems with Asterisk*: An easy introduction to using and configuring Asterisk to build feature-rich telephony systems for small and medium businesses. Olton: Packt Publishing Ltd, 2005.

MEGGELEN, Jim Van; SMITH, Jared; MADSEN, Leif. *Asterisk*: The Future of Telephony. 1. ed. Sebastopol: O'Reilly Media Inc, 2005.

LEHEY, Greg. *The Complete FreeBSD*: Documentation from Source. 4. ed. 1. ed. Sebastopol: O'Reilly Media Inc, 2003.

OLIVEIRA JÚNIOR, Eustáquio Rangel de. *Ruby*: Conhecendo a linguagem. Rio de Janeiro: Brasport, 2006.

THOMAS, Dave; FOWLER, Chad; HUNT, Andy. *Programming Ruby*: The Pragmatic Programmers' Guide. 2. ed. Dallas: Pragmatic Bookshelf, 2004.

THOMAS, Dave; HANSSON, David; BREEDT, Leon; CLARK, Mike; DAVIDSON, James Duncan; GEHTLAND, Justin; SCHWARZ, Andreas. *Agile Web Development with Rails*. 2. ed. Dallas: Pragmatic Bookshelf, 2006.

MILANI, André. *MySQL*: Guia do Programador. São Paulo: Novatec Editora, 2006.

SOARES, Luiz Fernando Gomes; LEMOS, Guido; COLCHER, Sérgio. Redes de Computadores: Das LANs, MANs e WANs às Redes ATM. 2. ed. Rio de Janeiro: Elsevier, 1995.

REIS, Maurício Caruzo. *Eletrônica de Telefone Residencial Celular*. Caraguatatuba-SP: Letron, 2003.

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

GONÇALVES, Flávio Eduardo de Andrade. *Asterisk Guia de Configuração*: Como Construir e Configurar um PBX com Software Livre. 4. ed. Florianópolis: V.Office, 2005.

GONZALEZ, Felipe Nogaroto. Estudo e Implementação de Solução de Voz sobre IP baseadas em Softwares Livres. 2007. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informações) – Instituto Superior Tupy, Joinville. 2007.

TORRES, Gabriel; LIMA, Cássio. *Como o protocolo TCP/IP Funciona – Parte 1*. Disponível em: < http://www.clubedohardware.com.br/artigos/1351>. Acesso em: 18 out. 2007.

DUARTE, Otto. *NAT – Network Address Translation*. Disponível em: <http://www.gta.ufrj.br/grad/01_2/nat/index.htm>. Acesso em: 18 out. 2007.

GALEA, Nick. *Dez Razões para Mudar para PABX IP*. Disponível em: <http://www.cliconnect.com/br/Artigos/DezrazoesparamudarPabxIP.html>. Acesso em: 23 out. 2007.

COMER, Douglas E. *Internetworking with TCP/IP Volume 1*: Principles, Protocols, and Architecture. 5. ed. New Jersey: Prentice Hall, 2005.

Site oficial do voip-info.org. http://www.voip-info.org

ANEXO 1 – ZAPATA.CONF

Disponível em cd-rom.

ANEXO 2 – MODULES.CONF

Disponível em cd-rom.

ANEXO 3 – CDR.CONF

Disponível em cd-rom.

ANEXO 4 – EXTENSIONS.CONF

Disponível em cd-rom.

ANEXO 5 – SIP.CONF

Disponível em cd-rom.

ANEXO 6 – SCRIPT SQL PARA CRIAÇÃO DA BASE DE DADOS DE AUTENTICAÇÃO

Disponível em cd-rom.

ANEXO 7 – CONFIGURAÇÕES DO *SOFT PHONE* X-LITE

Disponível em cd-rom.

ANEXO 8 – CONFIGURAÇÕES DO ATA

Disponível em cd-rom.

APÊNDICE A – CÓDIGO FONTE DA APLICAÇÃO DESENVOLVIDA

Disponível em cd-rom.