UNIVERSIDADE DO EXTREMO SUL CATARINENSE – UNESC

CURSO DE CIÊNCIA DA COMPUTAÇÃO

GUSTAVO CARVALHO DE FARIAS

SERVIDOR PROXY DE CONEXÃO VOIP PARA COMPARTILHAMENTO DE COMUNICAÇÃO Wi-Fi

CRICIÚMA, NOVEMBRO DE 2009

GUSTAVO CARVALHO DE FARIAS

SERVIDOR PROXY DE CONEXÃO VOIP PARA COMPARTILHAMENTO DE COMUNICAÇÃO Wi-Fi

Trabalho de Conclusão do Curso apresentado para obtenção do Grau de Bacharel em Ciência da Computação da Universidade do Extremo Sul Catarinense.

Orientador: Prof. MSc. Rogério Casagrande

CRICIÚMA, NOVEMBRO DE 2009

GUSTAVO CARVALHO DE FARIAS

SERVIDOR PROXY DE CONEXÃO VOIP PARA COMPARTILHAMENTO DE COMUNICAÇÃO WIFI

Submetido ao corpo docente do Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense como um dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Profa. MSc. Ana Claudia Garcia Barbosa
Coordenadora do Curso de Ciência da Computação

Banca Examinadora:

Prof. MSc. Rogério Antônio Casagrande (UNESC)
Orientador

Prof. MSc. Paulo João Martins (UNESC)

Prof. MSc. Vilson Gruber (SATC)

Dedico esta conquista aos meus pais, a minha namorada e parceira e especialmente aos meus avós Irma Pescador Carvalho e Auzenir Guimarães Carvalho *in memorian* que mesmo ausente sempre me inspirou a seguir em frente. Enfim agradeço por estarem sempre do meu lado, dando apoio e motivação para que eu realizasse este sonho !!!

Quero agradecer primeiramente a Deus, aos meus familiares e a minha namorada que sempre contribuíram e me incentivaram nesta longa caminhada.

Aos professores do curso que proporcionaram momentos enriquecedores durante o período que comigo estiveram.

Aos meus amigos da faculdade e em especial aos professores e orientadores Daniel Pezzi e Rogério Casagrande pelas orientações e dedicação com que me incentivaram a realizar este trabalho, que com muita sabedoria, mostraram-me os melhores caminhos para cumprir minha meta.

Enfim, a todas as pessoas que contribuíram nesta fase da minha vida.

“O verdadeiro homem mede a sua força, quando se defronta com o obstáculo.”

Antoine de Saint-Exupéry

RESUMO

Este trabalho apresenta o desenvolvimento de um servidor *proxy* VoIP para o compartilhamento em uma rede Wi-Fi por meio do protocolo SIP, com o objetivo de possibilitar aos usuários, acesso ao serviço a partir de um dispositivo móvel. No seu desenvolvimento foram utilizadas técnicas de *design patterns Model View Controller* (MVC), para a construção do segmento web do projeto. Para disponibilizar esse serviço foram analisados alguns servidores de aplicação, sendo um dos requisitos o controle do protocolo SIP, utilizado no desenvolvimento do servidor *proxy*. O servidor de aplicação da *Sun Microsystems Glassfish* se mostrou adequado uma vez que possibilita o controle do SIP através de um complemento chamado *Sailfin*. O servidor *proxy* VoIP desenvolvido permite aos usuários o cadastro via web para a utilização de seu cliente VoIP por meio da rede Wi-Fi utilizando dispositivos móveis.

Palavras-chave: Proxy, VoIP, Wi-Fi, SIP, Java, J2EE.

ABSTRACT

This work presents the development of a VoIP proxy server for the sharing in a net Wi-Fi through the protocol SIP, with the objective of making possible the users, access to the service starting from a mobile device. In his development techniques of design were used patterns Model View Controller (MVC), for the construction of the segment web of the project. To make available this service, was analyzed some application servers, being one of the requirements the control of the protocol SIP, used in the development of the proxy server. The server of application of Sun Microsystems Glassfish was shown appropriate once it makes possible the control of SIP through a called complement Sailfin. The VoIP proxy server developed allows to the users the register through web for his customer's use VoIP through the net Wi-Fi using mobile devices.

Palavras-chave: Proxy, VoIP, Wifi, SIP , J2EE.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

API	*Application Program Interface*
DHCP	*Dynamic Host Configuration Protocol*
DNS	*Domain Name Server*
EJB	*Enterprise JavaBeans*
FISL	Fórum Internacional de Software Livre
FMC	*Fixe-Mobile Convergence*
GHZ	*GigaHertz*
HTTP	*HyperTranfer Text Protocol*
IDE	*Integrated Development Environment*
IEEE	*Institute of Electrical and Electronic Engineers*
IETF	*Internet Engineering Task Force*
J2EE	*Java 2 Platform, Enterprise Edition*
IP	*Internet protocol*
IMTC	*International Multimedia Teleconferencing Consortium*
ISDN	*Integrated Services Digital Network*
LAN	*Local Area Network*
LDAP	*Lightweight Directory Access Protocol*
MBPS	*Megabits*
MP	Módulos Processados
MVC	*Model View Controller*
PAM	*Pulse Amplitud Modulation*
PAN	*Personal Area Network*
PBX	*Private Branch Exchange*

PCM	*Pulse Code Modulation*
PSTN	*Packet Switched Telephone Network*
QoS	*Quality of Service*
RTP	*Real-Time Transport Protocol*
RTCP	*Real-Time Transport Control Protocol*
SCM	Serviços de Comunicação Multimídia
SIG	*Special Interest Group*
SIP	*Session Initiation Protocol*
SOA	*Service Oriented architecture*
SQL	*Structured Query Language*
STFC	Serviços Telefônicos Fixos Comutados
STFCs	Serviços de Telefonia Convencional
TCP	*Transfer Control Protocol*
TI	Tecnologia da Informação
UAC	*User Agent Client*
UAS	*User Agent Server*
UDP	*User Datagram Protocol*
VoIP	*Voice over IP*
XML	*Extensible Markup Language*
WAN	*Wide Area Network*
Wi-Fi	*Wireless Fidelity*
WLAN	*Wireless Local Area Network*

SUMÁRIO

1 INTRODUÇÃO

O *Voice over IP* (VoIP) é uma tecnologia de telecomunicação em expansão. De acordo com estudos do IDC Brasil (2007), em seu relatório de dez previsões para a área de Tecnologia da Informação (TI) dentro da América Latina.

O grande diferencial da telefonia VoIP é que apresenta custos muito mais baixos que os da telefonia convencional, principalmente por utilizar-se da rede de dados já existente (Internet) e por permitir a transmissão simultânea de um grande número de chamadas, mesmo em enlaces de capacidade relativamente pequena, devido às técnicas de compressão utilizadas (GONÇALVES; HOMMERDING, 2006).

A utilização VoIP exige a necessidade de *software* e/ou *hardware* adequados. Existem diversos dispositivos como, *voipPhones*, centrais telefônicas conectadas a rede IP e o próprio computador pessoal. Existem também diversos softwares de empresas que fornecem o serviço VoIP dentre eles os mais conhecidos são Skype, como serviço para usuário final e o Asterix como Central PBX para VoIP, que tem o projeto em *Open Source*.

Entretanto, estes dispositivos têm um custo considerável e não são muito populares entre os usuários domésticos, como também no ambiente empresarial. Sendo assim, a forma mais difundida de acesso a tecnologia VoIP é por meio do computador pessoal com conexão a Internet.

Aprimorar os dispositivos seja pelo *software* ou *hardware*, tendo como objetivo facilitar o acesso ao VoIP permitindo que um número maior de pessoas tenha acesso a essa tecnologia.

Um dispositivo que se mostra aderente a essa nova tecnologia é o celular, por ser uma tecnologia que faz parte do cotidiano e um meio de comunicação bem difundido. Alguns celulares trazem a tecnologia para acesso a *Wireless Fidelity* (Wi-Fi) que por sua vez é uma

tecnologia de acesso muito explorada em cyber cafés, shopping, aeroportos é até disponível nas praças em algumas cidades.

A partir da união dessas tecnologias os horizontes da telefonia móvel são ampliados. Essa expansão atual da telefonia móvel tem promovido recentes estudos sobre a convergência VoIP-Celular.

1.1 OBJETIVO GERAL

Desenvolver um servidor *proxy* para comunicação VoIP via dispositivos móveis em redes sem fio.

1.2 OBJETIVOS ESPECÍFICOS

Os objetivos específicos são :

a) estudar as técnicas para fornecimento de serviços de comunicação em ambiente cliente/servidor;
b) Implantar um servidor de aplicação um ambiente para o fornecimento do serviço de comunicação.
c) estudar *wifi* para transferência de voz.;
d) implementar um servidor *proxy*;
e) integrar as tecnologias voip, dispositivos móveis e redes sem fio.

1.3 JUSTIFICATIVA

A tecnologia VoIP ainda apresenta carências de clientes para a convergência com os aparelhos celulares, entretanto, se for adicionada a essas tecnologias a comunicação *Wi-Fi*, será criado um novo meio de interação entre elas.

Neste novo meio de interação com o celular, o serviço VoIP terá ampliado seus limites, já que o celular é uma tecnologia amplamente disseminada. Sendo assim ao utilizar o *Wi-Fi* do celular o VoIP mostra uma nova possibilidade de seu uso e de sua abrangência.

Essa nova possibilidade de uso adiciona a tecnologia VoIP uma parcela da população que não possuem computadores em casa, porém participam de um grande movimento de inclusão digital, gerada pela disseminação das lan-houses. Segundo Sérgio Vieira Branco Junior em uma palestra no oitavo Fórum Internacional de Software Livre (FISL 8.0), realizada no ano de 2007 em Porto Alegre, as lan-houses são as maiores responsáveis pela inclusão digital no Brasil agindo dentro de favelas e bairros da periferia.

Unir as tecnologias VoIP, celular e *Wi-Fi* a ação social gerada pelas lan-houses permite a uma parcela da população possa ter acesso ao baixo custo da comunicação VoIP.

Dar a oportunidade a esta parcela da população usufruir das tecnologias que forneçam um serviço a um baixo custo como, por exemplo, efetuar ligações de longa distância a um custo local.

1.4 ESTRUTURA DO TRABALHO

No Capitulo 2 são abordadas as redes de comunicação e suas topologias, sendo no Sub-capítulo 2.2 apresentado a arquitetura Cliente/Servidor e as características básicas dessa arquitetura.

Os protocolos utilizados no VoIP, como H.323 e o *Session Initiation Protocol* (SIP) e uma comparação entre eles, também apresentando uma introdução a topologia de rede serão apresentados no Sub-capitulo 2.3.

No Sub-capítulo 2.4 será apresentado o *Voice over IP* (VoIP), sua arquitetura, protocolos utilizados e suas mensagens de requisição e resposta e uma introdução a fluxo de chamadas.

Apresentando a conexão sem fio (*Wireless*) o Sub-capítulo 2.5 aborda principalmente a *Wireless Fidelity* (Wi-Fi).

2 REDES DE COMPUTADORES

As redes de comunicação de maneira geral e formada por um conjunto de módulos processados (MPs[1]) capazes de trocar informações e compartilhar recursos, interligados por um sistema de comunicação, conforme Colcher et. al. (2005).

Colcher et al (2005) também nos diz que o sistema de comunicação é formado por um conjunto de regras a fim de organizar a comunicação denominados protocolos e uma topologia que consiste na ligação entre os vários módulos processadores através de enlaces físicos (meio de transmissão).

2.1 TOPOLOGIA

A fim de viabilizar a troca e o compartilhamento de dados surgiram as LANs, preservando a independência dos MPs e permitindo a integração em ambientes de trabalho cooperativo. Deste modo pode ser caracterizar como LAN uma rede que permite a interconexão de MPs em uma região relativamente pequena. Entretanto isso é genérico já que essas distâncias atualmente podem variar de poucos metros a 25 km, muito embora as limitações técnicas utilizadas em redes locais não imponham tais limites a essas distâncias. (COLCHER, 2005)

[1] MPs se refere a qualquer dispositivo capaz de se comunicar através do sistema de comunicação, por exemplo, computador, telefone, máquina copiadora conforme Colcher et al (2005).

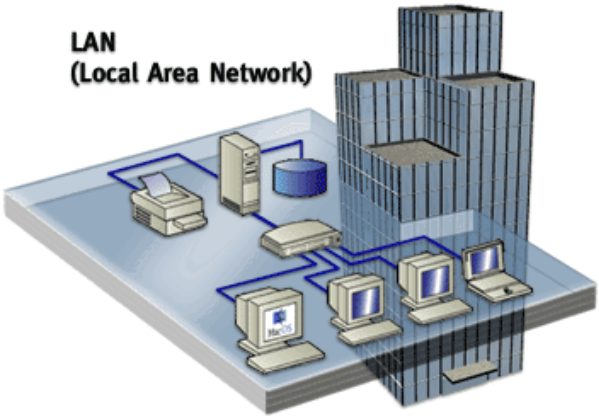


Figura 1. Exemplo de rede LAN
Fonte. COLCHER, S. (2005)

A Figura 1 ilustra um exemplo deste tipo de rede, onde e possível ver computadores clientes, um servidor de banco de dados e de impressão ligados por um *switch* ou *hub* formando um topologia tipo estrela[2]. Em geral essas redes são de responsabilidade da uma empresa ou instituição, e na Figura 1 isso fica demonstrado na representação de um edifício. Essa característica é a principal para distinguir uma LAN de uma *Wide Area Network* (WAN).

Redes WANs são redes geograficamente distribuídas, cuja abrangência alcança dimensões globais. Por possuírem um custo muito elevado como conseqüência da utilização de satélites, cabos submarinos, enlaces de microondas, essas redes são em geral públicas. Este sistema de comunicação, denominado sub-rede de comunicação é mantido e gerenciado, operado e de propriedade de grande concessionárias podendo elas ser públicas ou privadas (COLCHER, 2005).

[2] Em uma topologia tipo estrela cada nó é interligado a um nó central, este nó central age como centro de controle da rede interligando os demais nós escravos (COLCHER, 2005)

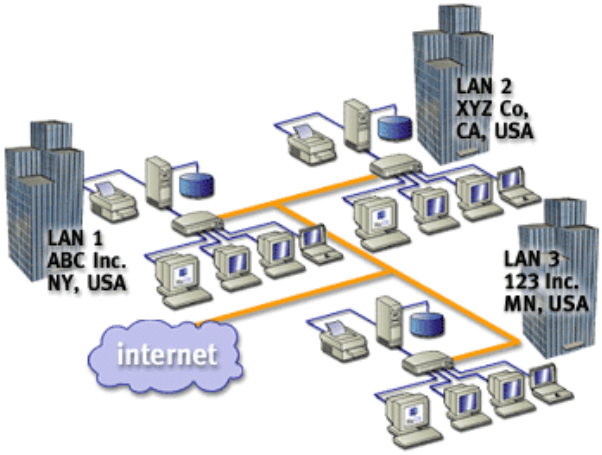


Figura 2. Exemplo de rede WAN
Fonte: COLCHER, S. (2005)

Na Figura 2 tem-se a visualização de uma rede interligando vários computadores, servidores e até mesmo LANs. Outro aspecto da imagem é a demonstração de duas situações, uma rede privada e outra de acesso público a Internet.

Na contra mão das grandes redes aparecem às redes *Personal Area Network* (PAN), que são muito similares as LANs, porém tem como característica a proximidade física tendo uma distância normalmente de metros. A PAN é normalmente composta de dispositivos de um único usuário como, por exemplo, celulares, computadores, impressoras.

Uma característica comum nesse tipo de rede e o uso de tecnologia sem fio ou *Wireless* como e normalmente conhecida. Na Figura 3 pode ser observado a aplicação dessa tecnologia com a utilização do *Bluetooth.*

Figura 3. Rede Wireless com Bluetooth
Fonte. www.rfidc.com

Com a tecnologia *Bluetooth* é possível conectar diversos dispositivos para formar uma rede PAN. Na Figura 3, é possível perceber isso, onde é demonstrado diversos dispositivos como, computadores, impressoras e PDAs todos interligados através do *Bluetooth*.

É bom salientar que as redes wireless é um conjunto de tecnologias de comunicação sem fio como Bluetooth, *Wireless Fidelity* (Wi-Fi) entre outras.

Esse conjunto de topologias LAN e WAN, somados ao meio de transmissão Wi-Fi, ampliam as possibilidades uma vez que o usuário dessas redes podem apartir de um celular no meio da rua possam imprimir um documento no seu escritório, para isso poder ocorrer de maneira adequada é necessário ainda adicionar mais um item, a arquitetura cliente/servidor, que no caso descrito acima atenderia disponibilizando um serviço de impressão e autenticação aos usuários, permitindo assim o perfeito funcionamento do sistema como um todo.

2.2 ARQUITETURA CLIENTE/SERVIDOR

A arquitetura cliente/servidor iniciou para atender uma nova realidade que surgiu com aumento significativo da potência dos computadores de mesa, que possibilitou a eles assumir maiores responsabilidades dentro de um ambiente computacional. Desta forma, foram desenvolvidos novos *softwares* para explorar essa capacidade computacional adicionando a ela a utilização de redes de computadores permitindo assim que um computador possa acessar dados de um ou vários sistemas em diferentes computadores, que possam estar fornecendo um serviço. Esse conjunto de tecnologias trabalhando juntas é denominada arquitetura cliente/servidor.

Nesta arquitetura, um processo do cliente faz solicitações ao processo servidor que atende essa solicitação (BOCHENSKI, 1995).

Na arquitetura cliente/servidor existe uma visão clássica da estrutura física onde fica melhor demonstrado na Figura 4.

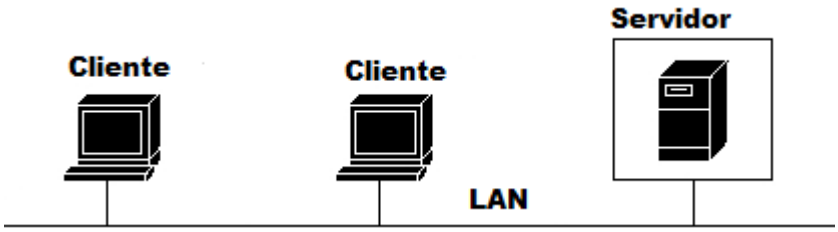


Figura 4. Estrutura física cliente/servidor
Fonte: BOCHENSKI, B. (1995)

Percebe-se na Figura 4 a existência de computadores clientes e um servidor interligados por uma *Local Area Network* (LAN), deste modo os clientes enviam suas solicitações ao servidor através da rede LAN, acessam dados compartilhados em outro computador cliente, que pode também fazer o papel de servidor.

Pode-se ter uma visão mais clara do que consiste uma arquitetura cliente/servidor definindo algumas características as quais foram descritas por Bochenski em 1995, quando

realizou uma enquete definindo as dez principais características em uma arquitetura cliente/servidor, as quais foram publicadas pela Computerworld em dezembro de 1995. As dez características mais citadas foram:

a) essa arquitetura deve consistir em um processo cliente e um processo servidor, podendo ser distintos um do outro, embora devam interagir totalmente;
b) o cliente e o servidor podem operar em diferentes plataformas de computadores, o que geralmente ocorre;
c) qualquer uma das plataformas pode ser atualizada, sem que exista a necessidade da atualizar a outra plataforma;
d) o servidor pode atender a vários clientes simultâneos, característica essa que também ocorre em algumas plataformas clientes/servidor, onde o cliente também acessa múltiplos servidores;
e) essa arquitetura também deve incluir qualquer tipo de capacidade de operar em rede;
f) uma parte significativa da lógica do aplicativo reside nos clientes;
g) em geral, a ação é iniciada no cliente, e não no servidor;
h) existe também uma interface amigável, geralmente no cliente;
i) um recurso característico mais não obrigatório é o *Structured Query Language* (SQL);
j) quando houver a disponibilidade do recurso SQL, também deve existir a proteção e segurança aos dados.

Porém, a maioria dos estudiosos que participaram da enquete concordam que somente as cinco primeiras características sejam fundamentais para uma arquitetura cliente/servidor.

A arquitetura cliente/servidor deve incluir algum tipo de capacidade de operar em rede. Essa característica é fundamental para a comunicação entre cliente e servidor como foi demonstrado na Figura 4.

Os serviços fornecidos por um servidor podem ser para vários, fins podendo-se listar algumas dessas funcionalidades, entre as quais algumas estão muito presentes no cotidiano.

Essas funcionalidades serão demonstradas por uma simulação passo a passo de como ocorre uma conexão com a Internet:

a) **passo 1**: o computador se conecta a rede e recebe um endereço IP fornecido pelo servidor *Dynamic Host Configuration Protocol* (DHCP)[3];

b) **passo 2**: o usuário digita seu nome e sua senha para acessar a rede do escritório essas informações são validades pelo servidor *Lightweight Directory Access Protocol* (LDAP)[4];

c) **passo 3**: ao abrir o navegador o usuário digita o endereço de um *site*, ao fazer o servidor de *Domain Name Server* (DNS)[5] resolvendo o endereço para um número IP;

d) **passo 4**: o servidor *proxy*[6] encaminhando o usuário à Internet para acessar o *site* desejado;

e) **passo 5**: o servidor *web* o qual hospeda o site desejado o serviço de disponibilidade desse conteúdo armazenado.

---

[3] O *Dynamic Host Configuration Protocol* (DHCP) é o serviço que oferece configuração dinâmica de rede aos clientes (ANDRADE; YAMAMOTO; 2003).

[4] O *Lightweight Directory Access Protocol* (LDAP) é um dos protocolos mais comumente utilizados em serviços de diretório. Diretório é uma base de dados cuja quantidade de consultas é significativamente superior à quantidade de outras transações. Os serviços de diretório são utilizados para manter conjuntos de atributos sobre determinados valores com contas de usuário, equipamentos de rede e etc. (ANDRADE; YAMAMOTO; 2003).

[5] O *Domain Name Server* (DNS), faz a tradução dos nomes de domínios em seus respectivos endereços IP's ou seja converte o endereço www.esmpresaA.com.br em um endereço IP tipo 172.146.34.1 (LIMA; 2001).

[6] Servidor Proxy, é o tipo de servidor que intercepta as requisições dos clientes e as executa por intermédio de outras requisições a outros servidores. Alguns *proxies* são implementados para *cache* e controle de acesso de conexões HTTP, onde o exemplo mais comum é o *Squid* (ANDRADE; YAMAMOTO; 2003).

Deste modo pode-se perceber o quanto é presente essa arquitetura. A listagem pode e vai aumentar conforme vai se avançando nas atividades como por exemplo, pode-se imprimir utilizando um servidor de impressão, ou falar ao telefone utilizando VoIP por meio de um servidor SIP.

Esses servidores se comunicam utilizando protocolos específicos para cada tipo de serviço. Estes protocolos são os que fornecem suporte e organizam uma rede de computadores.

2.3 PROTOCOLOS

Com o objetivo de reduzir a complexidade das redes, elas foram organizadas como se fosse uma pilha de camadas ou níveis, sobrepostos. O número de camadas, o nome, o conteúdo podem ser variados dependendo da rede analisada. As camadas em todas as redes têm a mesma função que é oferecer serviços à camada superior, isolando-a dos detalhes de implementação desses recursos oferecidos (TANENBAUM; 2003).

Com essa característica, as camadas vêm facilitar a vida dos usuários já que um determinado item de *software* ou *hardware* fornece um ou mais serviços, mantendo oculto os detalhes de seu estado interno e de seus algoritmos.

A camada *n* de uma máquina se comunica com a mesma camada de outra máquina. As regras e convenções utilizadas para essa comunicação são conhecidas como protocolo da camada *n*. Conforme Tanenbaum (2003), protocolo é um acordo entre partes que se comunicam, estabelecendo como se dará a comunicação.

Utilizando esse “acordo das partes que se comunicam”, é simplificado o desenvolvimento de software voltados a comunicação em rede como no caso do VoIP, onde

são utilizados vários protocolos, como IP, UDP, RTP, RTCP. Mas existem dois que definem como será feita a comunicação de voz propriamente dita que são H.323 e o SIP.

2.3.1 Protocolo H.323

Segundo Hersent (2002) o protocolo H.323 teve sua origem em 1995 baseada no protocolo H.320, que era um protocolo para comunicação de multimídia sobre redes *Integrated Services Digital Network* (ISDN). Com essa origem o H.323 tem um desempenho mais lento por exigir mais recursos, porém teve sua interoperabilidade facilitada com a ISDN.

Percebendo o potencial da voz sobre a rede IP, alguns fabricantes criaram o VoIP fórum. Essas empresas utilizando-se de seu prestigio e influência conseguiram integrar o VoIP fórum no *International Multimedia Teleconferencing Consortium* (IMTC). A partir dessa integração ao IMTC foram conquistadas alterações no protocolo que viabilizaram o uso do H.323 como padrão para um ambiente como a Internet.

O conjunto de recomendações que fazem parte do padrão H.323, descreve terminais, equipamentos, serviços e procedimentos para comunicações multimídia sobre *Internet Protocol* (IP) em tempo real. A Figura 5 apresenta os protocolos de sinalização para dados, voz e vídeo que fazem parte do padrão.

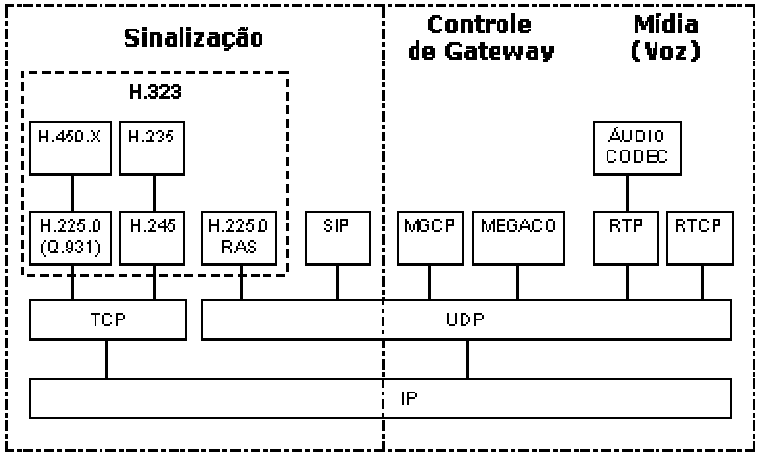


Figura 5. Pilha de protocolos do padrão H.323
Fonte: BERNAL F., H. (2008)

Tomando como base somente o que se refere à voz sobre IP, na Figura 5 percebe-se que consiste basicamente de três protocolos, além das especificações de *codecs* de áudio (LOURENÇO, R. 2007).

O protocolo RTP faz o encapsulamento das mídias, deixando-as adequadas a operar em conjunto com os protocolos de *real-time,* como o H.323. A utilização do RTP se torna importante na medida em que substitui o controle do TCP por uma solução mais simples, por fornecer somente informações importantes para os elementos de rede referentes à chamada, como informações de seqüência de pacotes, *timestamp* e tipo de mídia, proporcionando aos dispositivos a realização do controle de fluxo e de congestionamento.

Porém, o padrão não reserva recursos, e nem tem garantia de *Quality of Service* (QoS)[7]. Para isso foi desenvolvido um protocolo suplementar, efetuando um controle de entrega de dados e perda de pacotes, seu nome é RTCP. Ambos RTP e RTCP utilizam o protocolo UDP para envio das informações na rede (LOURENÇO, R. 2007).

O protocolo RTCP se baseia na transmissão periódica de pacotes para todos os participantes de uma sessão RTP, usando o mesmo mecanismo de distribuição dos pacotes de dados. Por meio da abertura de portas UDP, o RTCP provê um retorno da qualidade da distribuição de dados que pode ser utilizado para controle ou codificação adaptativa durante uma sessão RTP (LOURENÇO, R. 2007).

2.3.2 *Session Initiation Protocol* (SIP)

O SIP é outro padrão muito utilizado em VoIP, para substituir o H.323, e foi proposto pela *Internet Engineering Task Force* (IETF) (RFC 3261, 2002), e possui um

[7] *Quality of Service* (QoS), é um conjunto de procedimentos e técnicas com o objetivo de garantir um nível de qualidade à rede (REICHERT, 2004).

funcionamento relativamente simples em relação ao H.323. O SIP é utilizado simplesmente na sinalização de chamadas e conferência a fim de estabelecê-las.

Conforme Tarouco (2002) o protocolo SIP é utilizado para iniciar, modificar ou terminar sessões ou chamadas multimídia entre usuários e possui várias funcionalidades, dentre elas tem-se a localização de usuários, o estabelecimento de chamadas, o suporte a *unicast* ou *multicas*t, administração na participação de chamadas (transferências, conferência, entre outros) e possibilidade de participação de um usuário em terminal H.323, via *gatewa*y.

Também conforme Tauroco (2002) as sessões estabelecidas pelo SIP podem possuir diferentes tipos de dados, uma vez que sua configuração da sessão, alteração ou finalização são independentes do tipo de dado ou aplicação que está sendo utilizado na chamada.

Outra flexibilidade em relação aos tipos de dados que o SIP pode suportar, vem da sua origem que é baseada no protocolo HTTP e assim como ele, suporta o transporte de qualquer tipo de dados em seus pacotes, já que se utiliza de *MimeTypes* (*Multipurpose Internet Mail Extensions*).

2.3.2.1 Arquitetura do SIP

O SIP tem uma arquitetura baseada em cliente/servidor, por esse motivo possui somente métodos de requisição e resposta, outra característica herdada do HTTP e também observada no RTSP.

Essa arquitetura é constituída em duas partes (cliente/servidor): *User Agent e Network Serves.* Os User Agents podem ser um cliente (UAC – *User Agent Client*) ou um servidor (UAS – *User Agent Server*). Onde conforme Ferreira (2007) pode-se descrever cada um deles da seguinte maneira:

a) **UAC**: é o terminal SIP da estação final, ele funciona como um cliente no pedido de inicialização de sessão;

b) **UAS**: por sua vez, é caracterizado por responder ao pedido de sessão de um UAC.

Tendo essas características o *User Agent* tem uma arquitetura básica de cliente/servidor. As chamadas geradas pelo User Agent utilizando um endereço similar com a do email ou número de telefone. Tendo como exemplo o: SIP:31175@unesc.net. Desta maneira as URLs SIP são fáceis de associar a um endereço *email* do usuário.

Já o *Network Servers*, ou SIP Servers, são servidores maiores, e são divididos em três tipos, sendo os servidores: Proxy, Redirecionamento e Registro. E podendo-se descrever suas características individuas da seguinte maneira, segundo Gonçalves (2006):

a) ***proxy***: é um servidor que repassa a chamadas recebidas do *User Agent* para o próximo servidor que conhece melhor sobre a localização do cliente. Além disso, o servidor Proxy pode operar de dois modos: *stateful* (como circuto), *stateless* (como TCP). Operando como *stateful* ele pode "dividir" as chamadas por ordem de chegada, desta forma, pode haver várias extensões tocando, porém quando uma das extensões é atendida a chamada é transferida a ela, podendo dessa maneira, se ter um telefone SIP desktop, outra extensão no celular, e outro ponto e um telefone VoIP no escritório, todos eles respondendo pelo mesmo número ou usuário. Exemplo: 31175@unesc.net. Outra característica do servidor *Proxy* é que ele pode utilizar múltiplos métodos para resolver o pedido de endereço da solicitação, incluindo buscas de DNS, busca em base de dados ou transmitir ao servidor Proxy seguinte, entre outros;

b) **redirecionamento**: sua função é a resolução de nomes e localização do usuário, a partir de um pedido do *User Agent*, mas diferente do servidor Proxy ele não repassa os pacotes, mas sim, um pacote informando qual o próximo servidor, mapeando os endereços;
c) **registro**: este servidor recebe as requisições do *User Agent* de tipo *REGISTER*, armazenando ou atualização de localização dos usuários.

2.3.2.2 Funcionalidades do SIP

Tendo essa arquitetura cliente/servidor seu funcionamento básico é a realização de uma chamada a partir do cliente sendo atendida pelo servidor, ou em alguns casos pode ser envolvidos nessa chamada mais de um servidor. Para que essa chamada ocorra o protocolo tem que oferecer funções básicas, conforme Colcher (2005):

a) **conversão de nomes e localização de usuários:** envolve o mapeamento entre nomes de diferentes tipos de abstração, tais como nomes de um domínio e o nome de um usuário em um servidor Internet. Isso é necessário para que um determinado usuário que possui um nome qualquer possa ser convertido em um endereço IP, de modo que possa ser localizado a qualquer momento da ligação;
b) **negociação de configuração:** permite que um grupo de usuários defina que tipo de informação será trocada e seus respectivos parâmetros. O conjunto e o tipo dos dados que estão sendo enviados não precisam ser uniformes dentro de uma chamada. Como diferentes conexões ponto a ponto podem envolver diferentes tipos e parâmetros de dados. Muitos *codecs* são capazes de receber

diferentes tipos de codificações, sendo restritos ao enviar apenas um tipo de dado em cada fluxo;

c) **alteração de configuração:** torna possível a alteração, de maneira dinâmica, ou seja, durante a utilização de uma conexão por seus usuários, dos parâmetros definidos no momento do estabelecimento da conexão.

2.3.3 Comparativo H.323 E SIP

O *Session Initiation Protocol* (SIP) e o H.323 são padrões utilizados nas rotas das chamadas, nas sinalizações, troca de capacidades, controle de mídia e outros serviços adicionais. O principal diferencial do H.323 tem sido a sua interoperabilidade com o *Packet Switched Telephone Network* (PSTN) e sua disponibilidade de sistemas ou aplicações para *Desktop's* e em salas de videoconferência, possuindo um preço acessível e de qualidade confiável.(WANG; H., 2004)

Com o desenvolvimento da Internet e o incremento da comunicação de Voz sobre redes de pacotes, o SIP aparece como um protocolo mais simples já que foi desenvolvido especificamente para a internet, possuindo maior escalabilidade e flexibilidade. Seu funcionamento é baseado em troca de sinalização e mensagens similares à linguagem usada pelo protocolo HTTP. Ao invés de trocar sinalização baseada em bits, sua sinalização é baseada em texto, o que torna ela mais simples de compreender e implementar. Incorporando algumas experiências da Internet, o SIP tornou-se simples, aberto, compatível e com possibilidade de expansão (WANG; HU, 2004).

Desta maneira, é provável que o H.323 venha a ser utilizado como tecnologia de conferência para gerenciar serviços conferência/colaboração pelos próximos anos, e por outro

lado o SIP se tornando mais utilizado em aplicações voltadas a Internet (WANG; H., 2004). Existem diferenças entre os dois, as quais podem ser observadas na Tabela 1.

Tabela 1. Comparação entre o H.323 e o SIP.

Parâmetro	H.323	SIP
Padronizado pela	ITU-T	IETF
Complexidade do Protocolo	Complexo	Simples
Método de controle	Peer-to-peer	Cliente-Servidor
Formato das mensagens	ASN.1 (binário)	Texto (html)
Descrição das mídias suportadas	H.245	SDP
Equipamento de controle	H.323 Gatekeeper	Servidor SIP (proxy/ servidor de redirecionamento)
Capacidade de expansão	Pequena	Grande

Fonte: WANG; H. (2004)

Nas diferenças entre os protocolos demonstradas na Tabela 1, percebe-se que o SIP tem uma simplicidade maior de expansão, sendo uma arquitetura cliente/servidor e possui somente métodos de requisição e resposta, característica essa herdada da sua origem no HTML.

2.4 *VOICE OVER IP*

O conceito VoIP se inicia na década de 1990, quando surgiu o primeiro software comercial, o *Internet Phone* da VocalTec Communications, esse *software* permitia a troca de pacotes IP transportando amostras de voz entre computadores pessoais. Entretanto, nesse período a Internet era deficitária, e não fornecia uma qualidade de comunicação que se aproximasse à telefonia convencional.

Entretando, a tecnologia VoIP teve um rápida evolução em meados de 1998, quando algumas companhias já eram capazes de fornecer um serviço de internet com alguma qualidade, podendo assim se obter uma melhor qualidade no VoIP .

Essa evolução se confirma na atualidade, já que o VoIP é uma das tecnologias de telecomunicações de maior expansão, uma vez que possui custos baixos e usando técnicas de compressão para permitir uma transmissão simultânea de um grande número de chamadas mesmo num enlace relativamente pequeno. O VoIP consegue prover um serviço de melhor qualidade, que agora fica muito próximo à telefonia convencional.

VoIP de sinalização em meio a vários protocolos, está rapidamente se tornando o padrão no mercado para novas implementações já que agrega uma grande quantidade de serviços e possui uma estrutura relativamente simples, similar ao protocolo *Hyper Tranfer Text Protocol* (HTTP). Outra característica é a flexibilidade para desenvolvimento de novos produtos.

2.4.1 Arquitetura VoIP

O VoIP, segundo Hersent (2002), consiste na digitalização da voz, para isso, utiliza *codecs* apropriados. Esse processo de digitalização pode ser divididos em 3 etapas, conforme Colcher et. at. (2005): amostragem, quantização e compressão.

Durante a amostragem são capturadas amostras em pequenos intervalos de tempo, cada intervalo é determinado por um pulso de *clock* e a freqüência deste *clock* é chamada de taxa de amostragem. A quantização é onde os valores de *Pulse Amplitud Modulation* (PAM) obtidos são convertidos em valores discretos. Cada amostra é aproximada a um inteiro de *n bits*, produzindo-se os pulsos *Pulse Code Modulation* (PCM). O processo de compressão tem o objetivo de reduzir a quantidade de dados necessária para a representação da voz que foi

digitalizada, utilizando técnicas de remoção de informações redundantes e eliminação do silêncio, economizando assim a taxa de utilização da banda.

Após essas três etapas, o sinal digitalizado pode ser transportado através de pacotes IP. Com a utilização dos protocolos disponíveis para conexões de redes, é enviado pelo sistema a mensagem em pacotes, até chegarem ao destino, onde são recriadas e enviados ao dispositivo de som.

É necessário que esses dados sejam enviados entre dois pontos conhecidos, sendo assim necessária uma comunicação, onde o remetente passa ao destinatário uma mensagem, avisando o que é o conteúdo que esta sendo enviado e administrar a chamada. Este processo também é responsável pela sinalização. Sendo necessário um protocolo especial, dentre os quais o H.323 e *Session Initiation Protocol* (SIP).

Já o processo de transporte dos pacotes até o destino destaca-se com o protocolo *Real-Time Transport Protocol* (RTP). Conforme Tanenbaum (2003) consiste em um protocolo independente da camada de transporte e para que seus pacotes cheguem a um único destino (*unicast*) ou vários (*multicast*).

O RTP é complementado por um protocolo de controle chamado *Real-Time Transport Control Protocol* (RTCP) que possui a função de emitir periodicamente pacotes de controle para os participantes de uma sessão RTP, esses protocolos são encapsulados em um fluxo *User Datagram Protocol* (UDP).

O protocolo UDP opera de maneira contrária ao *Transfer Control Protocol* (TCP), deste modo não oferece funções de controle de erros e sequencialização o que é uma característica adequada às aplicações em tempo real, evitando assim a geração de “gagueira”, caso haja falha na entrega de alguns pacotes.

Por sua vez, os datagramas UDP são encapsulados em pacotes *Internet Protocol* (IP), como é mostrado na Figura **6**.

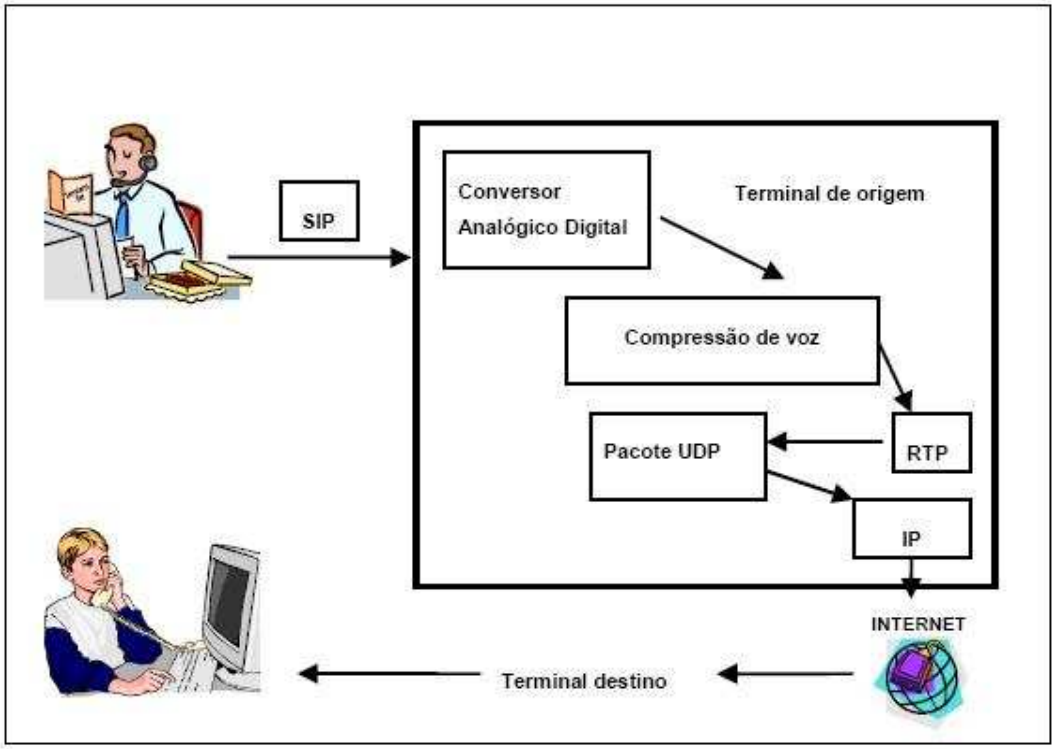


Figura 6. Fluxo Básico de dados de voz em um sistema VoIP
Fonte: BANDEIRA, A. (2007)

A Figura 6 demonstra os passos executados pelo SIP até chegar à internet, onde primeiro é efetuada a captura do sinal analógico e convertido em digital pelo conversor. Esse sinal digital e compactado utilizando *codecs*[8]. Após compactados esses dados são transportados pelo protocolo RTP que é encapsulado pelo protocolo UDP que por sua vez é encapsulado em pacotes IP para chegar a Internet e logo ao seu destino.

Outra característica do funcionamento do VoIP quando utilizado o protocolo SIP, que pode ser observado na Figura 7, é sua arquitetura cliente/servidor já que efetua apenas operações de requisição e resposta característica essa herdada do HTML.

[8] *Codecs* são responsáveis pela representação do sinal analógico em digital, de forma que seja utilizado um número reduzido de bits para o armazenamento e transmissão digital de voz ou vídeo (BANDEIRA, 2007)

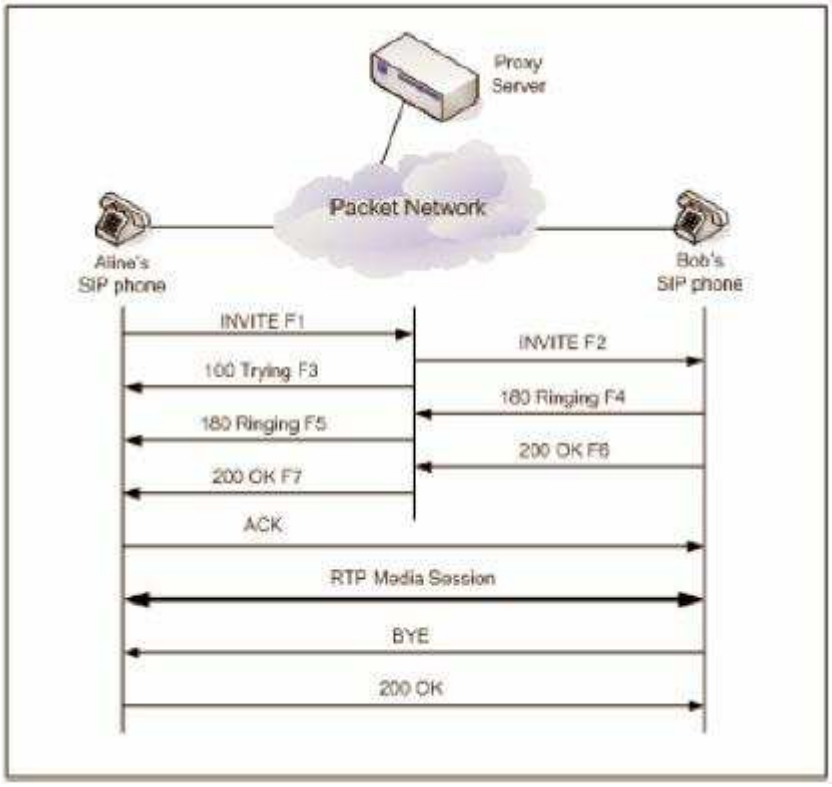


Figura 7. Conexão VoIP via SIP

Na Figura 7 é demonstrada a sinalização efetuada pelo VoIP onde o cliente Aline envia o convite ao servidor *proxy* que remete o convite ao cliente Bob e visível também o envio das sinalização de campainha (*ringing*), logo após a aceitação pelo cliente Bob o protocolo RTP é inicializado criando uma sessão de mídia para a comunicação da voz sobre IP, a ligação e finalizado pelo *BYE*. Este exemplo mostrado na Figura 7 utiliza como protocolo o SIP.

Na Figura 7 também é possível perceber que os dois telefones aparentemente são fixos, entretanto essa tecnologia se estende a computadores, e dispositivos móveis , utilizando como meio de transmissão a Wireless.

2.5 CONEXÃO SEM FIO (WIRELESS)

O termo Wireless significa "sem fio". A *Wireless* pode ser definir como uma tecnologia que disponibiliza a transmissão qualquer tipo de dados, como por exemplo som e imagens por meio de ondas de rádio. Essa tecnologia vem se desenvolvendo com o passar dos anos e com as exigências do mercado, e está se tornando um dos meios mais utilizados, uma vez que ela proporciona diversas funcionalidades tais como: consultar notícias; confirmar reservas em restaurantes ou hotéis, horários de vôos; verificar condições de tempo e trânsito; ler e-mail; fazer ou pagar compras feitas num shopping físico ou virtual, e tudo isso disponível em qualquer lugar ou hora, por meio de dispositivos móveis (celulares e PDA's). Essas funcionalidades demonstram que tem um grande potencial de exploração. (ZEINDIN; D., 2008)

Nos últimos anos houve grandes inovações no setor de comunicações sem fio, dentre elas, a tecnologia *Wireless Fidelity* (Wi-Fi) está sendo considerada o maior avanço no setor (ZEINDIN; D., 2008).

2.5.1 *Wireless Fidelity*

O Wi-Fi é o nome moderno dado a um conjunto de tecnologias sem fio que podem ser usadas para conectar tudo, desde computadores a utensílios de cozinha. É uma abreviação de “*Wireless Fidelity*”, que é um termo utilizado para descrever produtos que seguem o conjunto de padrões 802.11 o qual foi desenvolvido pelo *Institute of Electrical and Electronic Engineers* - IEEE. Dentre esses padrões o mais popular é o 802.11g, é compatível com o 802.11b e opera dentro da mesma faixa de 2,4 Ghz, alguns padrões podem transferir até 108 Mbps como o 802.11ª (ZEINDIN; D., 2008).

O padrão 802.11g ainda está sendo em desenvolvimento, tendo como objetivo o de operar a 22 Mb/s, mas ainda mantendo a compatibilidade com o padrão 802.11b. Outro padrão que está sendo elaborado é o 802.11e, que vai acrescentar gerenciamento de banda e melhorar os problemas causados por interferências tanto ao Wi-Fi como ao 802.11a.

Essas tecnologias em ainda em desenvolvimento possibilitam a expansão da tecnologia a novas fronteiras de velocidade e distância de modo a ampliar as aplicações baseadas nela.

3 SERVIDOR *PROXY* VoIP PARA COMPARTILHAMENTO DE WI-FI

O presente trabalho consiste no desenvolvimento de um Servidor *Proxy* VoIP com sua disponibilidade do serviço em redes Wi-Fi.

O servidor *proxy* tem o objetivo de permitir conexões simultâneas, tendo o controle sobre cada uma dessas conexões.

Esse controle se dá de maneira transparente ao UA (*User Agent*), desse modo podendo-se ter acesso ao áudio de cada uma dessas conexões, e também aos dados da conexão como, tempo, erros, taxa de transferência e todos os dados pertinentes à conexão analisada.

Buscando facilitar a obtenção dessas informações e simplificar o desenvolvimento foi utilizado um servidor de aplicação. A escolha dessa ferramenta é a parte principal do projeto.

3.1 METODOLOGIA

A metodologia para a realização do trabalho foi iniciada com o levantamento bibliográfico a respeito dos temas da pesquisa. Em seguida, para o cumprimento dos objetivos, foram realizadas as seguintes etapas:

a) pesquisa de um servidor de aplicação que atenda as necessidade de um servidor *Proxy* VoIP;
b) estudo do servidor de aplicação selecionado;
c) desenvolvimento do servidor utilizando as funcionalidades do servidor de aplicação;

d) definição de um ambiente Wi-Fi para testes da aplicação, e analise dos resultados.

O processo de desenvolvimento partiu da escolha do servidor de aplicação que abstraia os controles de rede e utilização de protocolos e portas, focando o desenvolvimento das funcionalidades do aplicativo.

3.1.1 A Escolha do Servidor de Aplicação

A escolha do servidor de aplicação a ser utilizado na implementação do servidor *proxy* foi realizada através da análise das características necessárias para o desenvolvimento da aplicação:

a) abstração do controle de rede e seus protocolos;

b) desenvolvimento em linguagem Java, por permitir a realização de alterações no código com mais facilidade, devido à estrutura orientada a objetos. Além disso, Java apresenta características de portabilidade, isto é, permite que “aplicativos rodem sem modificações sobre uma variedade de plataformas (diferentes tipos de computadores executando diferentes sistemas operacionais)” (DEITEL, Harvey M; 2003, p. 52), o que é desejável devido ao crescimento do uso de plataformas não baseadas em sistemas Microsoft, tais como o Linux.

c) funcionalidade, isto é, permitir a realização de chamadas de voz entre dois dispositivos UA.

d) utiliza o protocolo de comunicação SIP.

Dentre os servidores de aplicação analisados, o que possuia as características solicitadas era o *Glassfish* que utiliza o complemento *Sainfin* que faz o controle da comunicação via protocolo SIP.

3.1.2 Servidor de Aplicação *Glassfish*

O Servidor de aplicação *GlassFish*, é a implementação de referência Java EE, e possui código aberto que oferece desempenho, confiabilidade, produtividade e facilidade de uso. Por ser uma implementação J2EE ela é multi-plataforma, possibilitando que a aplicação seja disponibilizada através de qualquer plataforma, gratuita ou não. (Sun Microsystems, 2009).

O servidor de aplicação fornece vários recursos a fim de abstrair a complexidade de aplicações que serão disponibilizadas através da tecnologia J2EE. Outra de suas características que foi fundamental a sua escolha é a integração com a ferramenta de desenvolvimento *Netbeans*, simplificando o processo de desenvolvimento da aplicação.

Na Tabela 2. **Principais** são demonstradas as principais características do servidor *Glassfish* conforme a Sun Microsystems, 2009.

Tabela 2. Principais Características

Recurso	**Benefício**
Compatível com Java EE 5	Implementa as mais recentes tecnologias do Java EE 5, que ajudam a melhorar a eficiência do desenvolvedor.
Melhora a produtividade do Desenvolvedor	Aumenta a produtividade do desenvolvedor com APIs Java EE simplificadas e anotações que reduzem a quantidade de código que os desenvolvedores devem escrever.
Fundação para SOA	Suporta JAX-WS 2.0, JAXB 2.0 e Open ESB, fornecendo uma arquitetura aberta e extensível para colaboração entre tecnologias de integração e serviços web em uma arquitetura orientada a serviços (SOA)

Integração de Ferramentas	A IDE NetBeans com SOA suporta o desenvolvimento de aplicações Java EE 5 (incluindo módulo web e EJB 3.0) e fornece ferramentas de projeto visuais SOA para arquitetos e programadores.
Código Aberto	O código do *Sun Java System Application* Server 9.x é 100% derivado do servidor de aplicação *GlassFish.*

Fonte: SUN Microsystems, 2009.

O servidor *Glassfish* abstrair do desenvolver as necessidades de controle dos protocolos e suas portas de comunicação, uma vez que esse controle e simplificando através da interface de administração do próprio servidor, conforme Figura 8. **Console de Administração do *Glassfish***

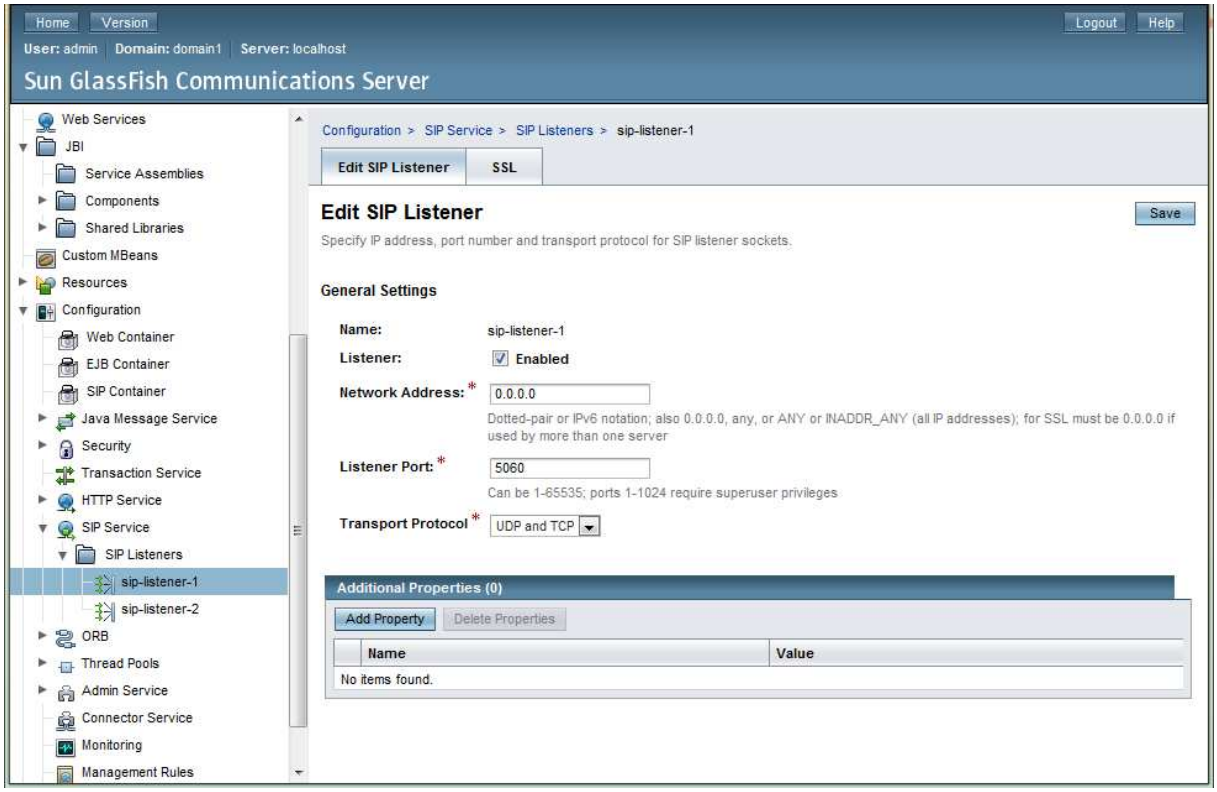


Figura 8. Console de Administração do *Glassfish*

Essa interface também permite a seleção do aplicativo para sua disponibilização através do servidor *Glassfish*, como por exemplo, uma aplicação SIP. Entretando para esse tipo de aplicação é necessário a instalação de um complemento chamando *Sailfin* o qual e responsável pela comunicação SIP dentro do servidor de aplicação *Glassfish.*

3.1.3 Projeto *Sailfin*

O *Sailfin* utiliza a tecnologia de *Servlets* mais especificamente *Sip Servlets* que contempla uma considerável parte do desenvolvimento de serviços de telecomunicações da próxima geração como VoiP, incorporando essas características ao servidor de aplicação *Glassfish*. Como demonstra a Figura 9.

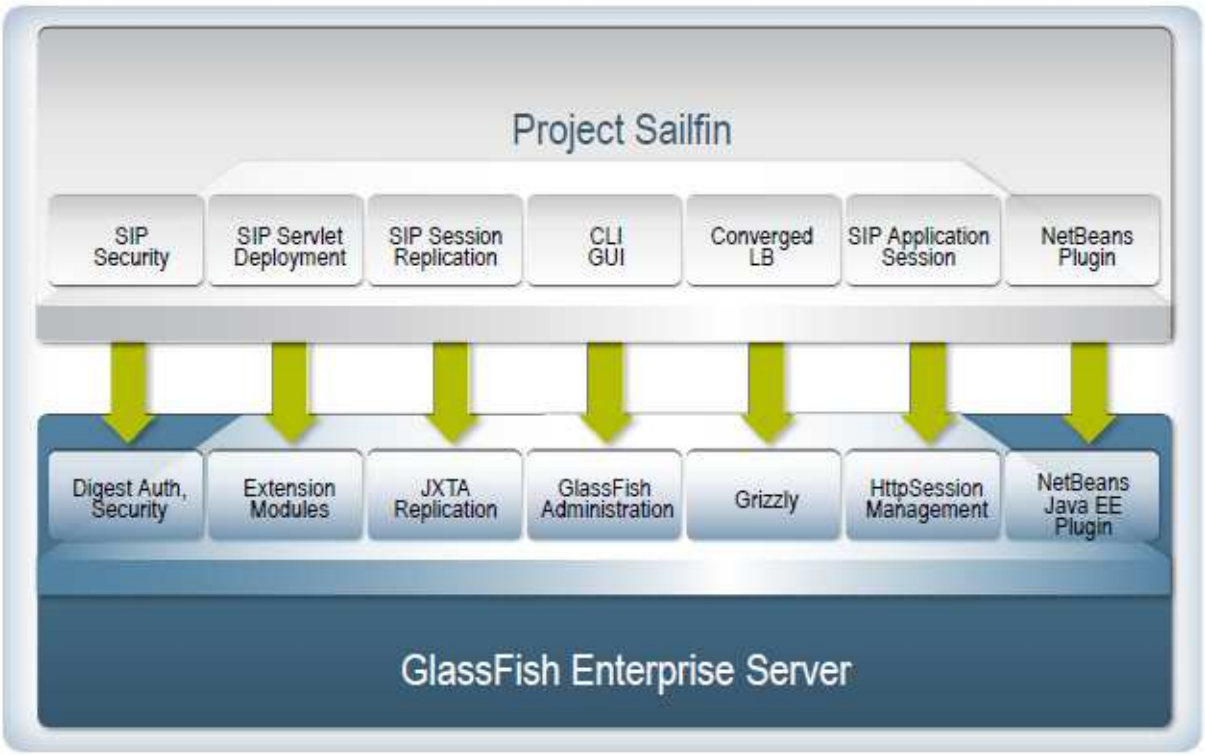


Figura 9. Integração entre o *Sailfin* e *Glassfish*
Fonte : Sun Microsystems, 2009.

Na Figura 9 é demonstrada a integração entre os servidores, onde por exemplo, mostra a integração do *Sip Application Session* do *Sailfin* ao *Http Session Management* do *Glassfish*, que faz o controle da requisições de conexão. Essa integração entre os servidores é fundamental para o alinhamento das ferramentas uma vez que o *Sailfin* estende as funcionalidades do *Glassfiss.*

3.2 DESENVOLVIMENTO

O desenvolvimento se iniciou na instalação do servidor de aplicação *Glassfish* e a implantação do complemento *Sailfin* e da ferramenta de desenvolvimento que também foi selecionada por possuir uma alta integração com o servidor de aplicação e o seu complemento.

Após a construção da base para desenvolvimento foi iniciado o processo de desenvolvimento propriamente dito. Durante os estudos do funcionamento do complemento *Sailfin* foi utilizado alguns exemplos para estudo da ferramenta. Os exemplos foram obtidos diretamente do site do projeto *Sailfin*, e serviram como base inicial para o desenvolvimento do projeto. Um dos exemplos utilizados, foi o *ClickToDial*.

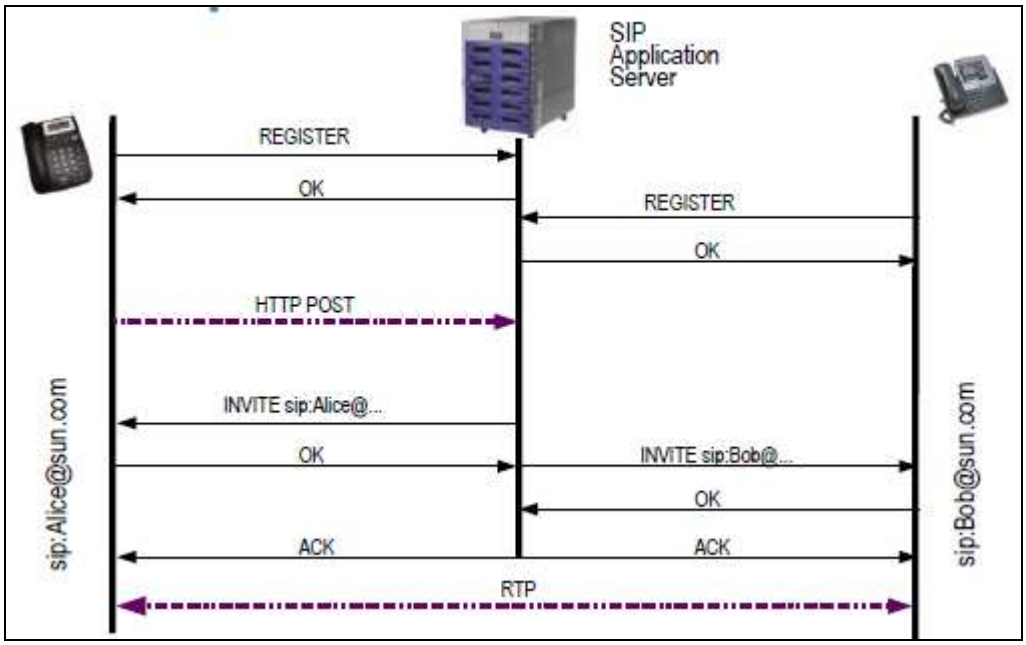


Figura 10. Exemplo *ClickToDial*
Fonte: sailfin.dev.java.net

Este exemplo permite a ligação entre dois clientes voip, iniciando a ligação através de um site na web, conforme apresentado na Figura 10. Onde os usuários,

Alice@sun.com e Bob@sun.com usando seus telefones voip se registram ao servidor Aplicação SIP e através de uma página, seleciona um usuário para efetuar sua ligação. Após a seleção via web e iniciada a requisição gerada através de uma chamada “HTTP POST”. O próximo evento e o envio de dois convites um para o usuário que gerou a solicitação de ligação e outro para o usuário a ser chamado. Seguindo os passos da Figura 10 após as aceitações dos convites por parte dos usuários e estabelecia a conexão através do RTP. Após efetuar o *deploy* do projeto dentro do servidor de aplicação, e execução de alguns testes, para estudar seu funcionamento. Esse exemplo ainda não possuía a comunicação que o projeto utiliza, o qual é efetuada através de um servidor *proxy*.

Porém esse exemplo possuía algumas características interessantes para serem aproveitadas como a interface para cadastramento de usuário através da *web*. Essa interface permitia o cadastramento e o *login* do usuário e uma listagem dos usuários cadastrados e seus estados de conexão podendo ser *online* ou *offline*.

3.2.1 Interface *Web*

A fim de desenvolver as funcionalidades, no que diz respeito ao *login* e ao cadastramento de usuários, foram reformuladas as interfaces e adicionados alguns campos.

No apêndice A são demonstradas as alterações, que lhe deram uma melhor aparência e o incremento do campo senha que não existia na versão inicial. Os botões “Criar Usuários Teste” tem a função de criar três usuários, Bob e Aline e Gustavo todas com a senha “1” de maneira automática a fim de minimizar o tempo de teste.

Seguindo com as implementações, o cadastro de usuários, teve adicionado os campos senha, nome e sobrenome, para uma melhor identificação do usuário. Também foi

desenvolvida a interface com as mesma características gráficas do *login* como pode ser visualizado no apêndice B.

Efetuadas as implementações nas interfaces e adicionados os campos, foi preciso desenvolver o comportamento do sistema, para adequar as novas necessidades de cadastro, para isso e necessário alterar a estrutura do banco de dados adicionando os novos campos. Porém ainda falta permitir o acesso a esses novos campos tanto para consulta quando para cadastro, deste modo foram remodeladas as seguintes classes, *ModelFacade.java* responsável pela interação do sistema com o banco de dados, também a classe modelo *Person.java*, essa por sua vez possui, os campos e suas definições, e ou validações quando necessário, a classe *LoginServlet.java* que é invocada através da *web* pela página *login.jsf*. Essa classe agora tem a funcionalidade de autenticação do usuário, utilizando o campo usuário e senha.

Uma vez que seja necessária a autenticação com senha o cadastro de usuário permite o cadastramento dos campos “usuário” e “senha”. Sendo assim a classe *NewUserServlet.java* permiti o cadastro desses campos, sendo essa classe invocada através da página da *web newUser.jsp*.

As interações entre as páginas *web* e os *servlets* foram implementadas utilizando o *Model View Controller* (MVC) *design pattern*. O MVC *design pattern* e exemplificado na Figura 11, onde é possível ver a divisão das três camadas, na camada *view* é executada a interação com o usuário, a *controller* possui as regras de negócio e a *model* possui a persistência com o banco de dados.

Exemplificando essas características dentro do projeto, fica a camada *view* sendo a página *web login.jsf* que é exibida no *browser*, o *LoginServlet.java* é o *controller* enquanto o *ModelFacade.java* é o *model* tendo a persistência com o banco de dados *proxySIP*, o mesmo ocorre *NewUserServlet.java* e *newUser.jsp*.

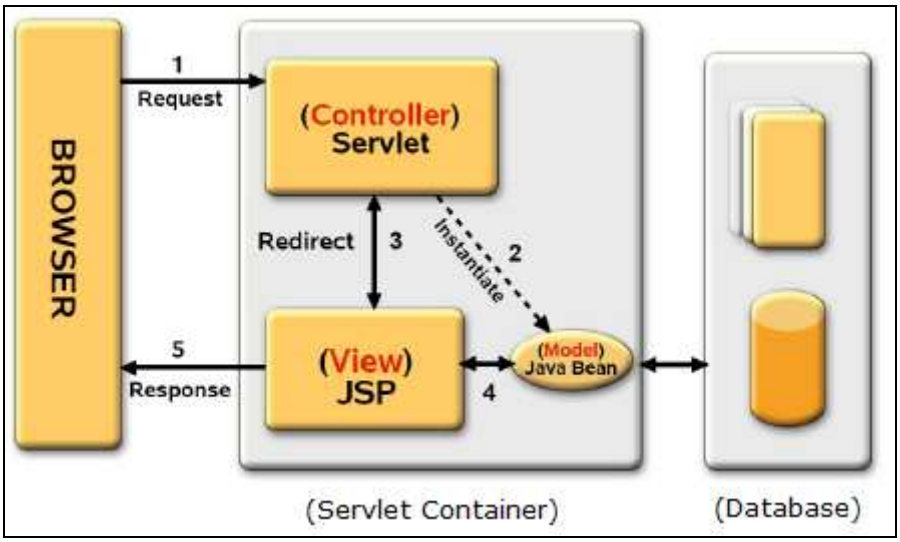


Figura 11. MVC *Design Pattern*
Fonte: http://www.cejug.org/

Foram criadas validações dentro da camada *controller* para não permitir cadastro de usuários com mesmo *login*, verificação da senha antes do cadastramento, e autenticação do usuário no momento do *login*, tanto pela a já existência do usuário, quanto pela validação da senha. Todas essas validações apresentam mensagens de erro para facilitar a correção do problema por parte do usuário. Na Figura 12 é demonstrado o fluxograma de *login* com a evolução do *login*, que pode chegar a passar pelo próprio cadastro caso o usuário ou senha inexistam. Esse fluxograma mostra a atual situação *web* até a página *main.jsp*.

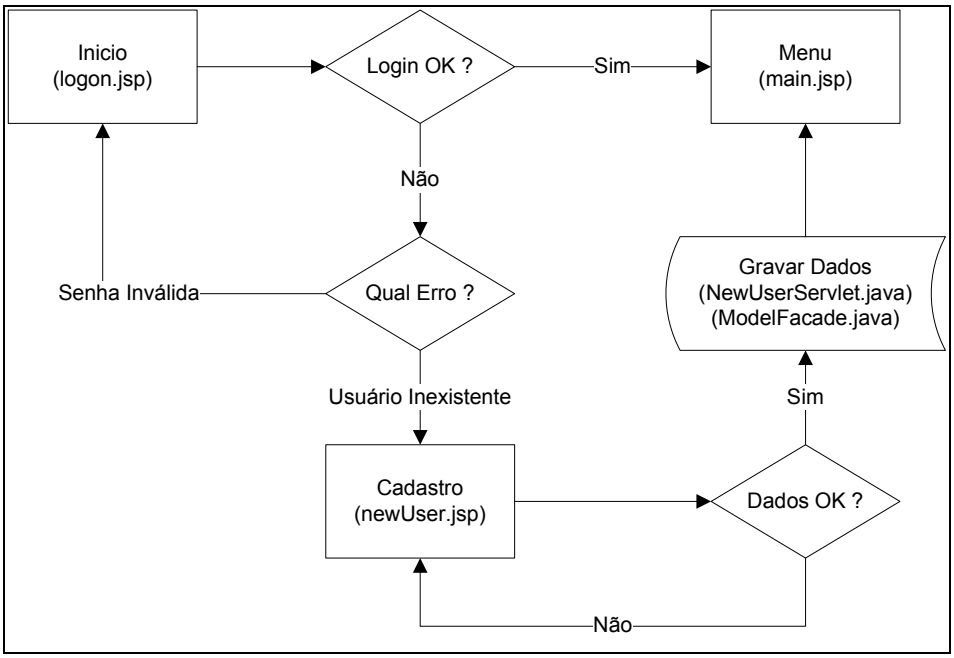


Figura 12. Fluxograma de *Login*

Na Figura 12 é possível observar as funcionalidades da interface web para o cadastramento, autenticação e listagem dos usuários conectados ou não. Na página *main.jsp* o usuário tem exibida uma listagens dos usuários cadastrados no sistema, com indicativo se estão *online* ou *offline*, conforme apêndice C.

Com essa listagem o usuário pode visualizar os usuários conectados e iniciar a comunicação VoIP fazendo utilização de um cliente UAC.

3.2.2 Registrando-se no Servidor SIP

Quando usuário efetua o uso de algum cliente UAC como, por exemplo, o *x-lite*, exibido na Figura 13. O Sistema ira verificar se o usuário está cadastrado e se a senha e válida.

Figura 13. Programa *X-Lite*

Validados usuário e senha é efetuado a atualização do campo *telephone* na tabela *Person*, campo esse que contém a string de conexão com usuário que agora está *online*. Na Tabela 3, pode-se verificar que o usuário Bob está conectado, e os demais usuários apesar de cadastrados estão *offline*, uma vez que não possuem o campo *Telephone* preenchido.

Tabela 3. Entidade *Person*

USERNAME	NAME	LASTNAME	PASSWORD	TELEPHONE
Gustavo	Gustavo	Carvalho	1	<null>
Bob	Bob	Silva	1	sip:Bob@192.168.1.100:5708;rinstance=0dbef1edda86d929
Alice	Alice	Silva	1	<null>

Esse processo iniciado pelo UAC, que efetua o registro do usuário através da autenticação e o preenchimento do campo *Telephone*. Isso ocorre quando o cliente UAC se conecta ao servidor previamente identificado nas configurações do mesmo e envia uma mensagem ao servidor através método *doRegister(SipServletRequest req)* que passa uma mensagem com os dados necessários para esse registro.

Essa mensagem possui a informação necessária para o registro e também serão utilizadas para o estabelecimento da conexão entre as partes envolvidas.

3.2.3 Manipulação de Mensagens

As mensagens enviadas ao servidor são manipuladas por *Sip Servlet*, os quais possuem métodos específicos para essa manipulação. Na conexão do UAC a sua primeira mensagem descrita na Figura 14 onde se tem uma visão dessa mensagem enviado ao servidor para o registro através do método *doRegister(SipServletRequest req).*

```
req = (com.ericsson.ssa.sip.SipServletRequestImpl) REGISTER sip:gustavo-pc SIP/2.0
Max-Forwards: 70
Content-Length: 0
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY, MESSAGE, SUBSCRIBE, INFO
Expires: 3600
To: "Bob Silva"<sip:Bob@gustavo-pc>
Contact: <sip:Bob@127.0.0.1:52268;rinstance=24019035f3e1ec53>
Cseq: 1 REGISTER
User-Agent: X-Lite release 1103k stamp 53621
Via: SIP/2.0/UDP 127.0.0.1:52268;branch=z9hG4bK-d8754z-713887130d3c022f-1---d8754z-;rport=52268;received=127.0.0.1
Call-Id: MGVjNjg2ZTJiYTRmMTZjZjM0ZGM5ODg3NjYxNTBhYjA.
From: "Bob Silva"<sip:Bob@gustavo-pc>;tag=c7207b4d
```

Figura 14. Mensagem Enviadas ao Servidor

Percebe-se na mensagem as informações já utilizadas, como *Contact*, essa usada para o preenchimento do campo *Telephone*. Outra informação é o *To* a qual foi utilizado na autenticação do usuário.

Para que a manipulação dessas mensagens seja através dos *Sip Servlet* corretos, é preciso descrever essa indicação dentro de um arquivo chamado *sip.xml*, esse arquivo contém as localizações dos métodos, ou seja em qual *Sip Servlet* esse método está descrito.

Este arquivo é demonstrando no Apêndice D, onde é possível visualizar o mapeamento dos métodos dentro dos *Sip Servlet* . A partir deste arquivo o servidor destina as

mensagens para os métodos dentro dos *Servlets* corretos. Os métodos iniciados pelos *Sip Request* e suas características estão exibidos na Tabela 4.

Tabela 4. Métodos de *SIP Request*

SIP Request	***Descrição***
doInvite()	Um cliente inicia enviando um convite de chamada a um participante.
doAck()	É confirmada a requisição do convite pelo cliente.
doBye()	A chamada é terminada pelo cliente origem ou destino.
doOptions()	Listagens das capacidades do servidor.
doCancel()	Cancela qualquer requisição pendente.
doRegister()	Registra o ciente com o acordo do servidor para o endereço *To* do cabeçalho.
doPrack()	Similar ao ACK, porém uma confirmação de provisionamento.
doSubscribe()	Subscribes the device for an event notification.
doNotify()	Notifica todos os *subscribers* de um evento.
doPublish()	Publica um evento para o servidor.
doInfo()	Envia informações no *middle* de uma *session* e não modifica o *status* da *session*.

Esses métodos descritos na Tabela 4 são endereçados pelo *xml* e invocados pelo UAS e UAC durante várias etapas da comunicação, desde o registro, até a finalização da comunicação. Entretanto nem todos os métodos tem a necessidade de implementação, neste projeto foram implementados os seguintes:

a) *doRegister* – faz a autenticação do usuário, e grava o campo *Telephone*;

b) *doBye* – Finaliza a sessão;

c) *doSuccessResponse* – Confirma o recebimento do *invite*;

d) *doErrorResponse* – Invalida a sessão por erro na resposta;

e) *doInvite* – Faz a chamada para inicio da sessão, buscando o contato dentro dos usuários cadastrados pela *web*.

3.2.4 Comunicação através do *Proxy Sip*

A comunicação se inicia após o a chamada efetuada pelo primeiro cliente, que digita em seu UA o usuário que deseja chamar, após isso o servidor recebe a requisição para sessão e efetuada uma série de etapas até chegar a comunicação. Essas etapas são demonstradas na Figura 15.

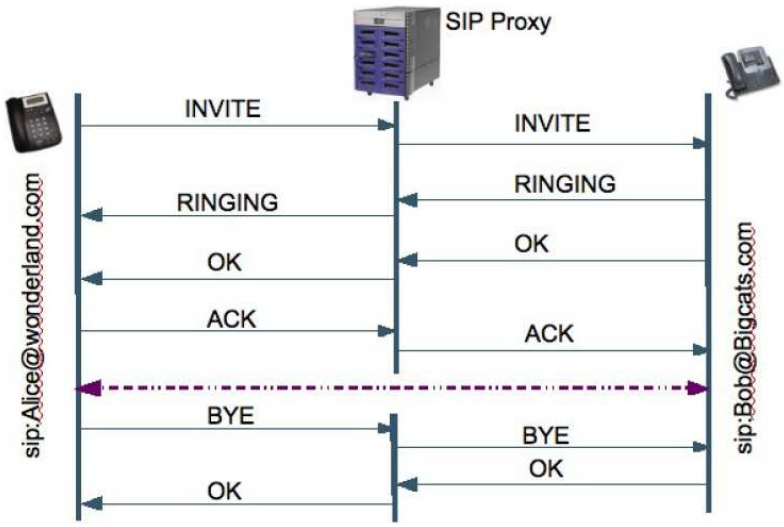


Figura 15. *Sip Proxy*

Na Figura 15 o cliente a usuária Alice faz o primeiro invite ao servidor proxy, note que no *ClicktoDial* exibido na Figura 10, essa etapa não existia pois o inicio da sessão se dava através de um *HTTP POST*, que faz a comunicação através de duas *request*, uma para o cliente origem e outra para o cliente destino. Entretanto utilizando o proxy a requisição *invite* e encaminhada ao servidor *Proxy* que destina essa mensagem para o *sip servlet* definido no *sip.xml* indicado com o *servlet* contém o método *doInvite().*

Esse processo e repetido outras vezes conforme as necessidades da sessão, na Figura 16 temos um diagrama de sequência exibindo a comunicação entre dois clientes.

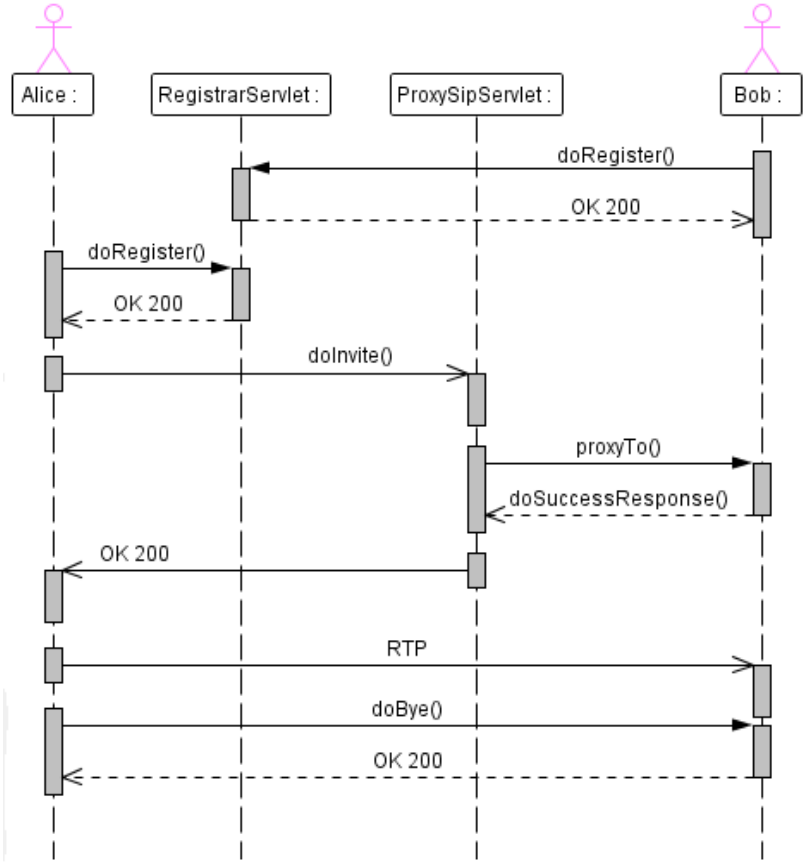


Figura 16. Diagrama de Sequência

Na diagrama são demonstrados alguns métodos principais utilizados durante o processo de estabelecimento até a finalização da sessão. Os dois clientes efetuam o registro através dos clientes UA utilizados por eles, em seguida a usuária Alice faz uma ligação ao usuário Bob, neste momento o cliente UA envia uma mensagem para o método *doInvite* para o servidor de aplicação, que verifica no *sip.xml* a qual *sip servlet* deve encaminhar a mensagem e a destina ao *RegistrarServlet*.

Ao receber essa mensagem o método *doInvite()*, envia o *telephone* destino ao método *proxyTo()*, isso envia um sinal sonoro no UA do Bob, que ao aceitar a ligação envia ao servidor uma mensagem destinada ao método *doSuccessResponse()*, que por suas vez envia um sinal de OK ao UA da Alice, e estabelecendo uma sessão entre os dois UA.

Ao final a usuária ao desligar envia uma mensagem ao servidor invocando o método *doBye()*, que e respondida com OK pelo usuário Bob.

CONCLUSÃO

Uma vez efetuada a implementação do servidor *proxy*, os testes de funcionamento dos processos contidos no servidor e efetuada uma avaliação dos resultados, foram obtidos um conjunto de informações que permitiram a apresentação das seguintes conclusões:

A arquitetura cliente/servidor foi utilizada, uma vez que a base da disponibilidade do serviço VoIP foi a utilização de um servidor de aplicação *Glassfish* e um componente chamado *SailFin* específico para comunicação SIP. Esse servidor foi selecionado seguindo uma série de critérios, e se mostrou muito aderente as necessidades para o desenvolvimento do servidor *proxy*.

O desenvolvimento do servidor *Proxy* teve parte de suas complexidades abstraídas pela utilização do servidor de aplicação, isso permitiu uma maior dedicação às funcionalidades. Entre suas funcionalidades a possibilidade de cadastramento do usuário via *web*, e sua autenticação durante a utilização do cliente VoIP, forneceu maior segurança na comunicação já que exige a utilização de senha, e não permite duplicação de usuários sendo disponíveis somente os usuários cadastrados na *web*.

A utilização do cliente VoIP foi facilitada já que este cliente pôde ser escolhido pelo usuário, com a única restrição de o software permitir o cadastro do servidor *proxy*. Isso permite a inclusão de clientes VoIP já existentes para dispositivos móveis, entretanto não são muitos os softwares com essa finalidade.

A integração do VoIP às redes sem fio ocorreu de maneira natural, já que o servidor atendeu a redes convencionais e sem fio, a qualidade da comunicação foi boa, já que apresentou baixo nível de ruído e poucas interrupções.

Esse resultado foi obtido durante testes efetuados entre a conexão de um cliente VoIP instalado no *iPaq* com acesso a rede *Wi-Fi* e um notebook também utilizando essa

mesma estrutura de rede. Foram feitas ligações a partir do *iPaq* e recebidas pelo notebook, como também o inverso. Isso permitiu obter os resultados observando a usabilidade da ferramenta.

Este projeto possibilitará um melhor resultado com a adição de novas funcionalidades como a conexão à um servidor *DNS* para a localização de usuários cadastrados em outros servidores *proxy*, a implementação de comunicação via mensagem de texto - IM (*instant messeger*) e a possibilidade de conexão desse servidor a telefonia convencional.

REFERÊNCIA

ANDRADE, Cid R.; YAMAMOTO, Marcelo A.; SALGUEIRO, Vicente. **Redes Cliente-Servidores.** São Paulo, 2003. Disponível em: < http://blog.cidandrade.pro.br/redes-cliente-servidor/>. Acesso em: 02 Jun. 2008.

BANDEIRA, Alessandra B. **Distribuição de Weibull na emulação de canal de redes WLAN para avaliação de VoIP**. 2007. 101 f. Trabalho de Conclusão de Curso (Pós-Graduação *Stricto-Sensu*) - Centro de Ciências Exatas, Ambientais e de Tecnologias, Pontifícia Universidade Católica de Campinas, 2007

BERNAL F., Huber. **Telefonia IP**. 2008.
http://www.teleco.com.br/tutoriais/tutorialtelip/pagina_3.asp Acessado em : 20 nov de 2009;

BOCHENSKI, B. **Implementando sistemas cliente/servidor de qualidade**. São Paulo: Makron, 1995

CANSADO, Jacinto C. **A Tecnologia Bluetooth Voltada para Aplicações com Sensores**. 2001. 31f. Trabalho de Conclusão de Curso (Graduação) - Sistemas Digitais, Escola Politécnica da Universidade de São Paulo, 2001.

COLCHER, Sérgio et al. **VOIP: Voz sobre IP**. Rio Janeiro: Campus, jan. 2005. 288p

DEITEL, Harvey M.; DEITEL, Paul J. **Java:** como programar. 4. ed. Porto Alegre: Bookman, 2003. 1386p.

GONÇALVES, André M.; HOMMERDING, Roberto. **Implementação didática de telefone VoIP por software utilizando protocolo SIP**. 2006. 62 f. Trabalho de Conclusão de Curso (Graduação) – Ciência Tecnologia em Eletrônica, Universidade Tecnológica Federal do Paraná, 2006.

LOURENÇO, Rogério B. **Protocolos VoIP para Redes Convergentes**. 2007. 150 f. Trabalho de Conclusão de Curso (Pós-Graduação *Stricto-Sensu*) - Centro Tecnológico, Engenharia de Telecomunicações da Universidade Federal Fluminense, 2007

HERSENT, O., GUIDE, D., PETIT, J-P. **Telefonia IP: Comunicação Multimida Baseada em Pacotes**. São Paulo: Editora Prentice Hall. 2002.

HES, Andre J; TEEFFELEN,Ronald. **Implementing Voice over IP.** Suíça, 2001. Disponível em: <http://www.upgrade-cepis.org/issues/2001/3/upgrade-VII.3.pdf>. Acesso em: 21 mar. 2008.

LIMA, Almir Wirth. **Telecomunicações multimídia**. Rio de Janeiro: Book Express, 2001

MATSUZAKA, Marcelo K.; HASEGAWA, Márcio G. **Simulação de Threads em Ambiente Multi-core**. 2008. 37 f. Trabalho de Conclusão de Curso (Graduação) – Instituto de Matemática, Departamento de Ciências da Computação, Universidade de São Paulo, 2008.

OLIVEIRA, R. A. Rabello, et al. **Caracterização de Topologias Dinâmicas no Bluetooth.** Departamento de Ciência da Computação, Universidade Federal de Minas Gerais. Disponível em: <http://www.dcc.ufmg.br/~rabelo/rabelo-wcsf2001.pz.gz>. Acesso: em 26 mai 2008.

REICHERT , Romeu H**. Transmissão De Voz Sobre Redes De Pacotes** ESTUDO DE CASO COM H.323 Novo Hamburgo, 2004.

RIBAS, Rodrigo G. **O VoIP e suas Soluções** . 2006. 51 f. Trabalho de Conclusão de Curso (Pós-Graduação *Lato-Sensu*) – Departamento de Computação Especialização em Desenvolvimento de Aplicações para Web, Universidade Estadual de Londrina.

SIQUEIRA, Ethevaldo. **Telecom World: a crise acabou**, http://www.estadao.com.br/rss/tecnologia/colunas/2003/out/19/25.htm. Acesso em: 21 mar de 2008.

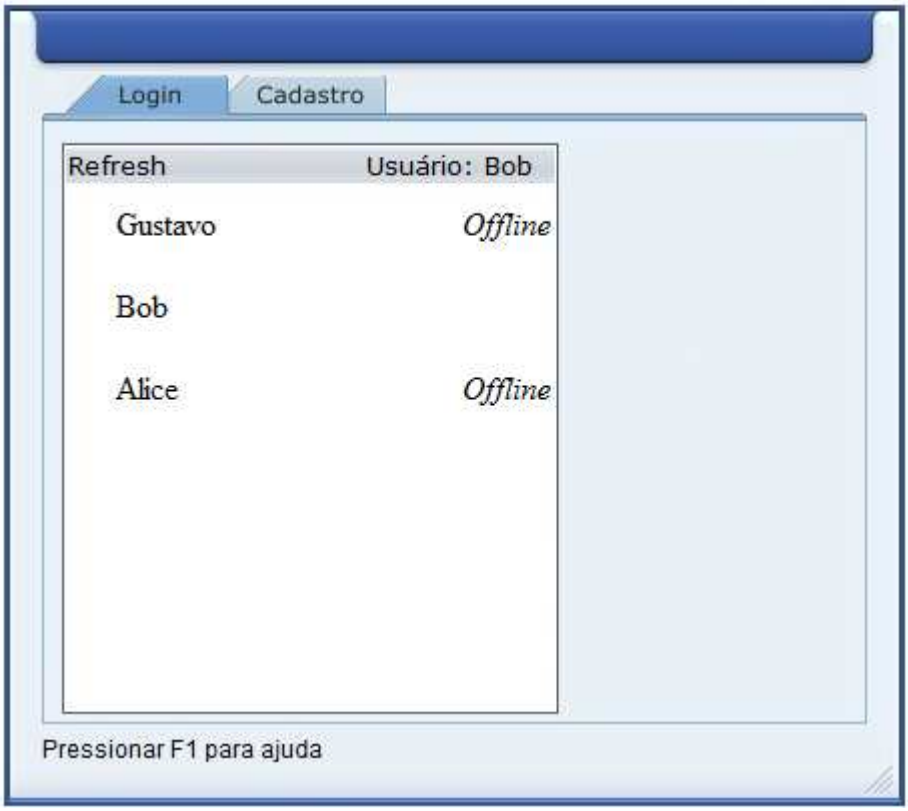
TANENBAUM, Andrew S. **Redes de Computadores.** Rio de Janeiro: Elsevier, 2003. 945p.
WANG, Runsheng; HU Xiaorui. **VoIP Development in China**. IEEE Computer Society, New York: v.37, n.9, p.30-37, set. 2004.

ZEINDIN, DENISE CARLA A. **A Tecnologia do Futuro Wi-Fi (*Wireless Fidelity*).** Blumenau, 2008.

APÊNDICE A. TELA DE *LOGIN* DE USUÁRIO

APÊNDICE B. CADASTRO DE USUÁRIOS

APÊNDICE C. MENU COM USUÁRIOS DISPONÍVEIS E SEUS *STATUS*.



APÊNDICE D. ARQUIVO SIP.XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sip-app PUBLIC "-//Java Community Process//DTD SIP Application
1.0//EN" "http://www.jcp.org/dtd/sip-app_1_0.dtd">
<sip-app>
   <display-name>Click to Dial application</display-name>

   <listener>
        <listener-class>clicktodial.sip.CallSipServlet</listener-class>
   </listener>

   <servlet>
      <servlet-name>CallSipServlet</servlet-name>
      <servlet-class>clicktodial.sip.CallSipServlet</servlet-class>
      <load-on-startup>1</load-on-startup>
   </servlet>

   <servlet>
      <servlet-name>RegistrarServlet</servlet-name>
      <servlet-class>clicktodial.sip.RegistrarServlet</servlet-class>
   </servlet>

   <servlet-mapping>
     <servlet-name>RegistrarServlet</servlet-name>
     <pattern>
         <or>
             <equal>
                 <var>request.method</var>
                 <value>REGISTER</value>
             </equal>
         </or>
     </pattern>
   </servlet-mapping>

   <servlet-mapping>
     <servlet-name>CallSipServlet</servlet-name>
     <pattern>
         <or>
             <equal>
                 <var>request.method</var>
                 <value>BYE</value>
             </equal>
             <equal>
                 <var>request.method</var>
                 <value>INVITE</value>
             </equal>
             <equal>
                 <var>response.method</var>
                 <value>OK</value>
             </equal>
         </or>
     </pattern>
   </servlet-mapping>
```

SERVIDOR PROXY DE CONEXÃO VOIP PARA COMPARTILHAMENTO DE COMUNICAÇÃO Wi-Fi

Gustavo Carvalho de Farias[1]

[1]Ciência da Computação – Universidade do Extremo Sul Catarinense (UNESC)
88.804-370 – Criciúma – SC – Brasil

gu.carvalho@gmail.com

Abstract. *This work presents the development of a VoIP proxy server for the sharing in a net Wi-Fi through the protocol SIP, with the objective of making possible the users, access to the service starting from a mobile device. In his development techniques of design were used patterns Model View Controller (MVC), for the construction of the segment web of the project. To make available this service, was analyzed some application servers, being one of the requirements the control of the protocol SIP, used in the development of the proxy server. The server of application of Sun Microsystems Glassfish was shown appropriate once it makes possible the control of SIP through a called complement Sailfin. The VoIP proxy server developed allows to the users the register through web for his customer's use VoIP through the net Wi-Fi using mobile devices..*

Resumo. *Este trabalho apresenta o desenvolvimento de um servidor proxy VoIP para o compartilhamento em uma rede Wi-Fi por meio do protocolo SIP, com o objetivo de possibilitar aos usuários, acesso ao serviço a partir de um dispositivo móvel. No seu desenvolvimento foram utilizadas técnicas de design patterns Model View Controller (MVC), para a construção do segmento web do projeto. Para disponibilizar esse serviço foram analisados alguns servidores de aplicação, sendo um dos requisitos o controle do protocolo SIP, utilizado no desenvolvimento do servidor proxy. O servidor de aplicação da Sun Microsystems Glassfish se mostrou adequado uma vez que possibilita o controle do SIP através de um complemento chamado Sailfin. O servidor proxy VoIP desenvolvido permite aos usuários o cadastro via web para a utilização de seu cliente VoIP por meio da rede Wi-Fi utilizando dispositivos móveis.*

1. Introdução

O *Voice over IP* (VoIP) é uma tecnologia de telecomunicação em expansão. De acordo com estudos do IDC Brasil (2007), em seu relatório de dez previsões para a área de Tecnologia da Informação (TI) dentro da América Latina.

O grande diferencial da telefonia VoIP é que apresenta custos muito mais baixos que os da telefonia convencional, principalmente por utilizar-se da rede de dados já existente (Internet) e por permitir a transmissão simultânea de um grande número de chamadas, mesmo em enlaces de capacidade relativamente pequena, devido às técnicas de compressão utilizadas (GONÇALVES; HOMMERDING, 2006).

A utilização VoIP exige a necessidade de *software* e/ou *hardware* adequados. Existem diversos dispositivos como, *voipPhones*, centrais telefônicas conectadas a rede IP e o

próprio computador pessoal. Existem também diversos softwares de empresas que fornecem o serviço VoIP dentre eles os mais conhecidos são Skype, como serviço para usuário final e o Asterix como Central PBX para VoIP, que tem o projeto em *Open Source*.
Entretanto, estes dispositivos têm um custo considerável e não são muito populares entre os usuários domésticos, como também no ambiente empresarial. Sendo assim, a forma mais difundida de acesso a tecnologia VoIP é por meio do computador pessoal com conexão a Internet.
Aprimorar os dispositivos seja pelo *software* ou *hardware*, tendo como objetivo facilitar o acesso ao VoIP permitindo que um número maior de pessoas tenha acesso a essa tecnologia.
Um dispositivo que se mostra aderente a essa nova tecnologia é o celular, por ser uma tecnologia que faz parte do cotidiano e um meio de comunicação bem difundido. Alguns celulares trazem a tecnologia para acesso a *Wireless Fidelity* (Wi-Fi) que por sua vez é uma tecnologia de acesso muito explorada em cyber cafés, shopping, aeroportos é até disponível nas praças em algumas cidades.
A partir da união dessas tecnologias os horizontes da telefonia móvel são ampliados. Essa expansão atual da telefonia móvel tem promovido recentes estudos sobre a convergência VoIP-Celular.

2.SIP

O SIP é outro padrão muito utilizado em VoIP, para substituir o H.323, e foi proposto pela *Internet Engineering Task Force* (IETF) (RFC 3261, 2002), e possui um funcionamento relativamente simples em relação ao H.323. O SIP é utilizado simplesmente na sinalização de chamadas e conferência a fim de estabelecê-las.
Conforme Tarouco (2002) o protocolo SIP é utilizado para iniciar, modificar ou terminar sessões ou chamadas multimídia entre usuários e possui várias funcionalidades, dentre elas tem-se a localização de usuários, o estabelecimento de chamadas, o suporte a *unicast* ou *multicas*t, administração na participação de chamadas (transferências, conferência, entre outros) e possibilidade de participação de um usuário em terminal H.323, via *gatewa*y.
Também conforme Tauroco (2002) as sessões estabelecidas pelo SIP podem possuir diferentes tipos de dados, uma vez que sua configuração da sessão, alteração ou finalização são independentes do tipo de dado ou aplicação que está sendo utilizado na chamada.
Outra flexibilidade em relação aos tipos de dados que o SIP pode suportar, vem da sua origem que é baseada no protocolo HTTP e assim como ele, suporta o transporte de qualquer tipo de dados em seus pacotes, já que se utiliza de *MimeTypes* (*Multipurpose Internet Mail Extensions*).

2.1. Arquitetura SIP

O SIP tem uma arquitetura baseada em cliente/servidor, por esse motivo possui somente métodos de requisição e resposta, outra característica herdada do HTTP e também observada no RTSP.
Essa arquitetura é constituída em duas partes (cliente/servidor): *User Agent e Network Serves.* Os User Agents podem ser um cliente (UAC – *User Agent Client*) ou um

servidor (UAS – *User Agent Server*). Onde conforme Ferreira (2007) pode-se descrever cada um deles da seguinte maneira:

c) **UAC**: é o terminal SIP da estação final, ele funciona como um cliente no pedido de inicialização de sessão;
d) **UAS**: por sua vez, é caracterizado por responder ao pedido de sessão de um UAC.

Tendo essas características o *User Agent* tem uma arquitetura básica de cliente/servidor. As chamadas geradas pelo User Agent utilizando um endereço similar com a do email ou número de telefone. Tendo como exemplo o: SIP:31175@unesc.net. Desta maneira as URLs SIP são fáceis de associar a um endereço *email* do usuário.

Já o *Network Servers*, ou SIP Servers, são servidores maiores, e são divididos em três tipos, sendo os servidores: Proxy, Redirecionamento e Registro. E podendo-se descrever suas características individuas da seguinte maneira, segundo Gonçalves (2006):

d) ***proxy***: é um servidor que repassa a chamadas recebidas do *User Agent* para o próximo servidor que conhece melhor sobre a localização do cliente. Além disso, o servidor Proxy pode operar de dois modos: *stateful* (como circuto), *stateless* (como TCP). Operando como *stateful* ele pode "dividir" as chamadas por ordem de chegada, desta forma, pode haver várias extensões tocando, porém quando uma das extensões é atendida a chamada é transferida a ela, podendo dessa maneira, se ter um telefone SIP desktop, outra extensão no celular, e outro ponto e um telefone VoIP no escritório, todos eles respondendo pelo mesmo número ou usuário. Exemplo: 31175@unesc.net. Outra característica do servidor *Proxy* é que ele pode utilizar múltiplos métodos para resolver o pedido de endereço da solicitação, incluindo buscas de DNS, busca em base de dados ou transmitir ao servidor Proxy seguinte, entre outros;
e) **redirecionamento**: sua função é a resolução de nomes e localização do usuário, a partir de um pedido do *User Agent*, mas diferente do servidor Proxy ele não repassa os pacotes, mas sim, um pacote informando qual o próximo servidor, mapeando os endereços;
f) **registro**: este servidor recebe as requisições do *User Agent* de tipo *REGISTER*, armazenando ou atualização de localização dos usuários.

2.2. Funcionalidades do SIP

Tendo essa arquitetura cliente/servidor seu funcionamento básico é a realização de uma chamada a partir do cliente sendo atendida pelo servidor, ou em alguns casos pode ser envolvidos nessa chamada mais de um servidor. Para que essa chamada ocorra o protocolo tem que oferecer funções básicas, conforme Colcher (2005):

d) **conversão de nomes e localização de usuários:** envolve o mapeamento entre nomes de diferentes tipos de abstração, tais como nomes de um domínio e o nome de um usuário em um servidor Internet. Isso é necessário para que um determinado usuário que possui um nome qualquer possa ser convertido em um endereço IP, de modo que possa ser localizado a qualquer momento da ligação;
e) **negociação de configuração:** permite que um grupo de usuários defina que tipo de informação será trocada e seus respectivos parâmetros. O

conjunto e o tipo dos dados que estão sendo enviados não precisam ser uniformes dentro de uma chamada. Como diferentes conexões ponto a ponto podem envolver diferentes tipos e parâmetros de dados. Muitos *codecs* são capazes de receber diferentes tipos de codificações, sendo restritos ao enviar apenas um tipo de dado em cada fluxo;

f) **alteração de configuração:** torna possível a alteração, de maneira dinâmica, ou seja, durante a utilização de uma conexão por seus usuários, dos parâmetros definidos no momento do estabelecimento da conexão.

3. *Voice Over IP*

O conceito VoIP se inicia na década de 1990, quando surgiu o primeiro software comercial, o *Internet Phone* da VocalTec Communications, esse *software* permitia a troca de pacotes IP transportando amostras de voz entre computadores pessoais. Entretanto, nesse período a Internet era deficitária, e não fornecia uma qualidade de comunicação que se aproximasse à telefonia convencional.

Entretando, a tecnologia VoIP teve um rápida evolução em meados de 1998, quando algumas companhias já eram capazes de fornecer um serviço de internet com alguma qualidade, podendo assim se obter uma melhor qualidade no VoIP .

Essa evolução se confirma na atualidade, já que o VoIP é uma das tecnologias de telecomunicações de maior expansão, uma vez que possui custos baixos e usando técnicas de compressão para permitir uma transmissão simultânea de um grande número de chamadas mesmo num enlace relativamente pequeno. O VoIP consegue prover um serviço de melhor qualidade, que agora fica muito próximo à telefonia convencional.

VoIP de sinalização em meio a vários protocolos, está rapidamente se tornando o padrão no mercado para novas implementações já que agrega uma grande quantidade de serviços e possui uma estrutura relativamente simples, similar ao protocolo *Hyper Tranfer Text Protocol* (HTTP). Outra característica é a flexibilidade para desenvolvimento de novos produtos.

3.1. Arquitetura VoIP

Na Figura 17 é demonstrada a sinalização efetuada pelo VoIP onde o cliente Aline envia o convite ao servidor *proxy* que remete o convite ao cliente Bob e visível também o envio das sinalização de campainha (*ringing*), logo após a aceitação pelo cliente Bob o protocolo RTP é inicializado criando uma sessão de mídia para a comunicação da voz sobre IP, a ligação e finalizado pelo *BYE*. Este exemplo mostrado na Figura 7 utiliza como protocolo o SIP.

Na Figura 17 também é possível perceber que os dois telefones aparentemente são fixos, entretanto essa tecnologia se estende a computadores, e dispositivos móveis , utilizando como meio de transmissão a Wireless.

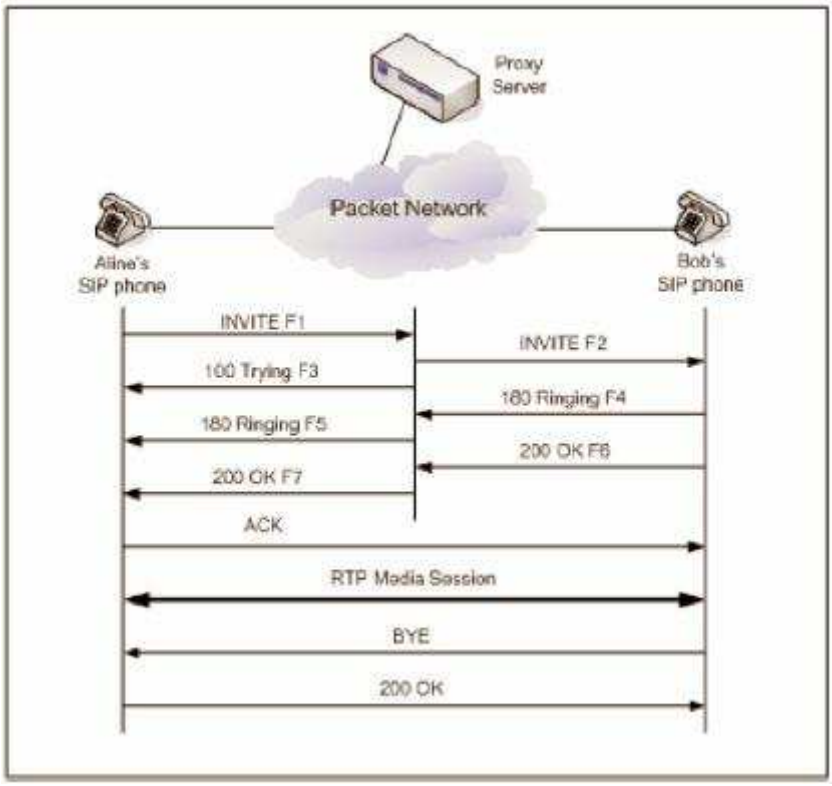


Figura 17. Conexão VoIP via SIP

4. Desenvolvimento do Servidor Proxy

O presente trabalho consiste no desenvolvimento de um Servidor *Proxy* VoIP com sua disponibilidade do serviço em redes Wi-Fi.

O servidor *proxy* tem o objetivo de permitir conexões simultâneas, tendo o controle sobre cada uma dessas conexões.

Esse controle se dá de maneira transparente ao UA (*User Agent*), desse modo podendo-se ter acesso ao áudio de cada uma dessas conexões, e também aos dados da conexão como, tempo, erros, taxa de transferência e todos os dados pertinentes à conexão analisada.

Buscando facilitar a obtenção dessas informações e simplificar o desenvolvimento foi utilizado um servidor de aplicação. A escolha dessa ferramenta é a parte principal do projeto.

4.1. Servidor de Aplicação Glassfish *e* Sailfin

O Servidor de aplicação *GlassFish*, é a implementação de referência Java EE, e possui código aberto que oferece desempenho, confiabilidade, produtividade e facilidade de uso. Por ser uma implementação J2EE ela é multi-plataforma, possibilitando que a aplicação seja disponibilizada através de qualquer plataforma, gratuita ou não. (Sun Microsystems, 2009).

O servidor de aplicação fornece vários recursos a fim de abstrair a complexidade de aplicações que serão disponibilizadas através da tecnologia J2EE. Outra de suas

características que foi fundamental a sua escolha é a integração com a ferramenta de desenvolvimento *Netbeans*, simplificando o processo de desenvolvimento da aplicação. O *Sailfin* utiliza a tecnologia de *Servlets* mais especificamente *Sip Servlets* que contempla uma considerável parte do desenvolvimento de serviços de telecomunicações da próxima geração como VoiP, incorporando essas características ao servidor de aplicação *Glassfish*. Como demonstra a Figura 18.

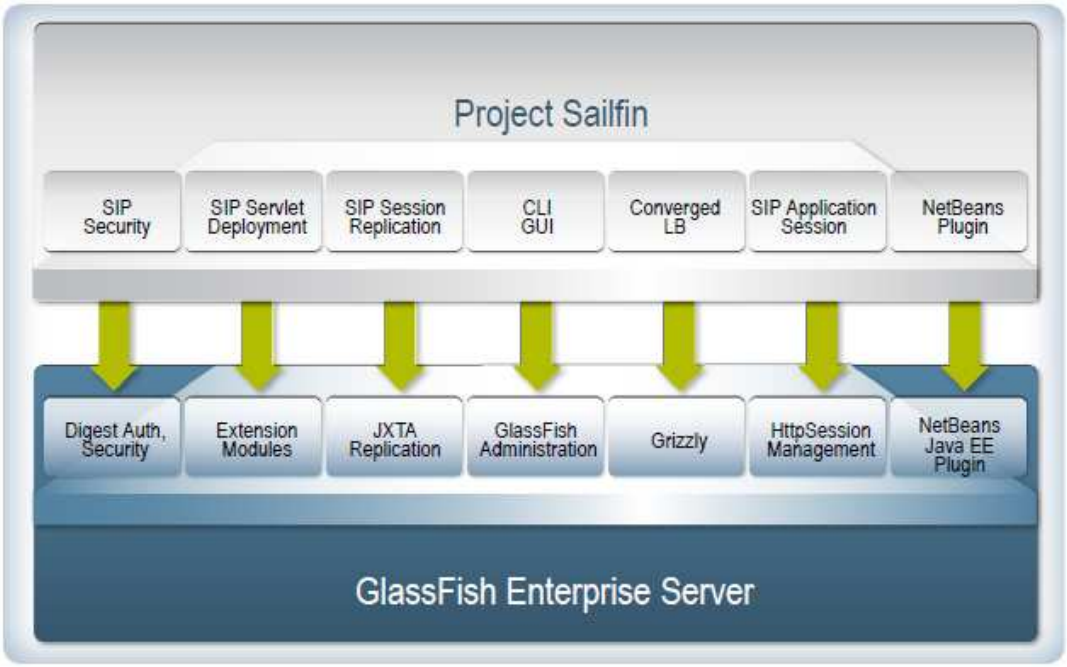


Figura 18. Integração entre o *Sailfin* e *Glassfish*

Na Figura 18 é demonstrada a integração entre os servidores, onde por exemplo, mostra a integração do *Sip Application Session* do *Sailfin* ao *Http Session Management* do *Glassfish*, que faz o controle da requisições de conexão. Essa integração entre os servidores é fundamental para o alinhamento das ferramentas uma vez que o *Sailfin* estende as funcionalidades do *Glassfiss.*

5. Desenvolvimento

O desenvolvimento se iniciou na instalação do servidor de aplicação *Glassfish* e a implantação do complemento *Sailfin* e da ferramenta de desenvolvimento que também foi selecionada por possuir uma alta integração com o servidor de aplicação e o seu complemento.

O desenvolvimento se iniciou na instalação do servidor de aplicação *Glassfish* e a implantação do complemento *Sailfin* e da ferramenta de desenvolvimento que também foi selecionada por possuir uma alta integração com o servidor de aplicação e o seu complemento.

5.1. Interface Web

A fim de desenvolver as funcionalidades, no que diz respeito ao *login* e ao cadastramento de usuários, foram reformuladas as interfaces e adicionados alguns campos.

Seguindo com as implementações, o cadastro de usuários, teve adicionado os campos senha, nome e sobrenome, para uma melhor identificação do usuário. Também foi desenvolvida a interface com as mesma características gráficas do *login*.

Efetuadas as implementações nas interfaces e adicionados os campos, foi preciso desenvolver o comportamento do sistema, para adequar as novas necessidades de cadastro, para isso e necessário alterar a estrutura do banco de dados adicionando os novos campos. Porém ainda falta permitir o acesso a esses novos campos tanto para consulta quando para cadastro, deste modo foram remodeladas as seguintes classes, *ModelFacade.java* responsável pela interação do sistema com o banco de dados, também a classe modelo *Person.java*, essa por sua vez possui, os campos e suas definições, e ou validações quando necessário, a classe *LoginServlet.java* que é invocada através da *web* pela página *login.jsf*. Essa classe agora tem a funcionalidade de autenticação do usuário, utilizando o campo usuário e senha.

Uma vez que seja necessária a autenticação com senha o cadastro de usuário permite o cadastramento dos campos "usuário" e "senha". Sendo assim a classe *NewUserServlet.java* permiti o cadastro desses campos, sendo essa classe invocada através da página da *web newUser.jsp*.

As interações entre as páginas *web* e os *servlets* foram implementadas utilizando o *Model View Controller* (MVC) *design pattern*. O MVC *design pattern* e exemplificado na Figura 19, onde é possível ver a divisão das três camadas, na camada *view* é executada a interação com o usuário, a *controller* possui as regras de negócio e a *model* possui a persistência com o banco de dados.

Exemplificando essas características dentro do projeto, fica a camada *view* sendo a página *web login.jsf* que é exibida no *browser*, o *LoginServlet.java* é o *controller* enquanto o *ModelFacade.java* é o *model* tendo a persistência com o banco de dados *proxySIP*, o mesmo ocorre *NewUserServlet.java* e *newUser.jsp*.

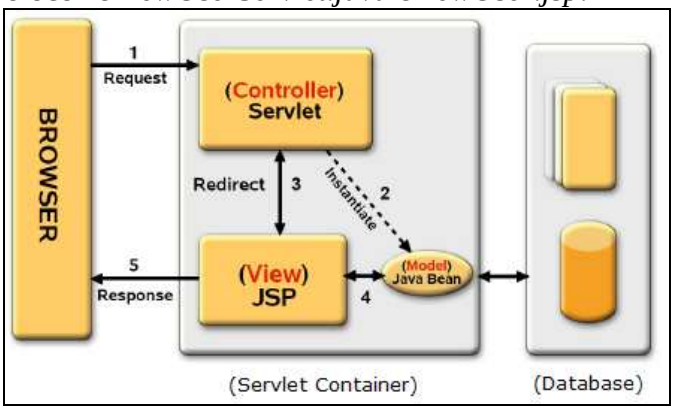


Figura 19. MVC Design Pattern

5.2. Registrando-se no Servidor SIP

Quando usuário efetua o uso de algum cliente UAC como, por exemplo, o *x-lite*, o sistema ira verificar se o usuário está cadastrado e se a senha e válida.

Validados usuário e senha é efetuado a atualização do campo *telephone* na tabela *Person*, campo esse que contém a string de conexão com usuário que agora está *online*. Na Tabela 5, pode-se verificar que o usuário Bob está conectado, e os demais usuários apesar de cadastrados estão *offline*, uma vez que não possuem o campo *Telephone* preenchido.

Tabela 5. Entidade Person

USERNAME	NAME	LASTNAME	PASSWORD	TELEPHONE
Gustavo	Gustavo	Carvalho	1	<null>
Bob	Bob	Silva	1	sip:Bob@192.168.1.100:5708;rinstance=0dbef1edda86d929
Alice	Alice	Silva	1	<null>

Esse processo iniciado pelo UAC, que efetua o registro do usuário através da autenticação e o preenchimento do campo *Telephone*. Isso ocorre quando o cliente UAC se conecta ao servidor previamente identificado nas configurações do mesmo e envia uma mensagem ao servidor através método *doRegister(SipServletRequest req)* que passa uma mensagem com os dados necessários para esse registro.
Essa mensagem possui a informação necessária para o registro e também serão utilizadas para o estabelecimento da conexão entre as partes envolvidas.

5.3. Comunicação Através do Proxy SIP

A comunicação se inicia após o a chamada efetuada pelo primeiro cliente, que digita em seu UA o usuário que deseja chamar, após isso o servidor recebe a requisição para sessão e efetuada uma série de etapas até chegar a comunicação. Essas etapas são demonstradas na Figura 20. Sip Proxy.

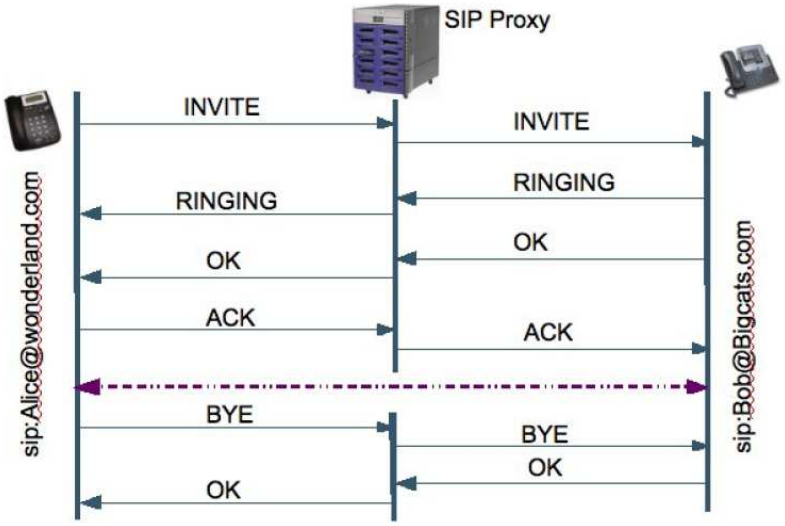


Figura 20. Sip Proxy

Na Figura 20. Sip Proxy o cliente a usuária Alice faz o primeiro invite ao servidor proxy, note que no *ClicktoDial*, essa etapa não existia pois o inicio da sessão se dava através de um *HTTP POST*, que faz a comunicação através de duas *request*, uma para o cliente origem e outra para o cliente destino. Entretanto utilizando o proxy a requisição *invite* e encaminhada ao servidor *Proxy* que destina essa mensagem para o *sip servlet* definido no *sip.xml* indicado com o *servlet* contém o método *doInvite().*
Esse processo e repetido outras vezes conforme as necessidades da sessão, na Figura 21 temos um diagrama de sequência exibindo a comunicação entre dois clientes.

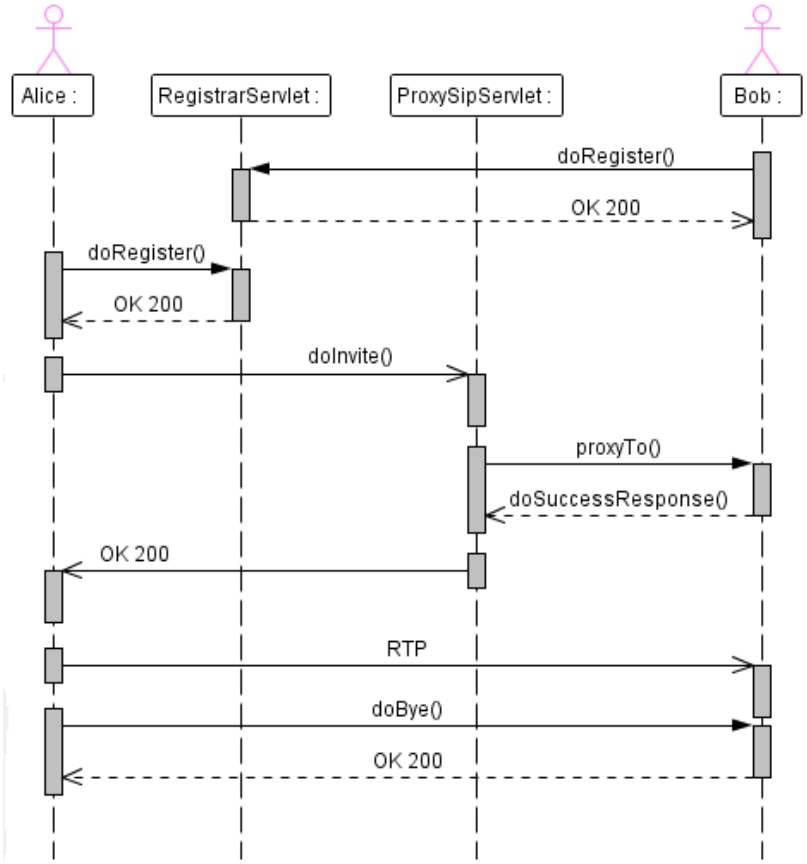


Figura 21. Diagrama de Sequência

Na diagrama são demonstrados alguns métodos principais utilizados durante o processo de estabelecimento até a finalização da sessão. Os dois clientes efetuam o registro através dos clientes UA utilizados por eles, em seguida a usuária Alice faz uma ligação ao usuário Bob, neste momento o cliente UA envia uma mensagem para o método *doInvite* para o servidor de aplicação, que verifica no *sip.xml* a qual *sip servlet* deve encaminhar a mensagem e a destina ao *RegistrarServlet*.
Ao receber essa mensagem o método *doInvite(),* envia o *telephone* destino ao método *proxyTo(),* isso envia um sinal sonoro no UA do Bob, que ao aceitar a ligação envia ao servidor uma mensagem destinada ao método *doSuccessResponse(),* que por suas vez envia um sinal de OK ao UA da Alice, e estabelecendo uma sessão entre os dois UA.
Ao final a usuária ao desligar envia uma mensagem ao servidor invocando o método *doBye(),* que e respondida com OK pelo usuário Bob.

6. Conclusão

Uma vez efetuada a implementação do servidor *proxy*, os testes de funcionamento dos processos contidos no servidor e efetuada uma avaliação dos resultados, foram obtidos um conjunto de informações que permitiram a apresentação das seguintes conclusões:

A arquitetura cliente/servidor foi utilizada, uma vez que a base da disponibilidade do serviço VoIP foi a utilização de um servidor de aplicação *Glassfish* e um componente chamado *SailFin* específico para comunicação SIP. Esse servidor foi selecionado seguindo uma série de critérios, e se mostrou muito aderente as necessidades para o desenvolvimento do servidor *proxy*.

O desenvolvimento do servidor *Proxy* teve parte de suas complexidades abstraídas pela utilização do servidor de aplicação, isso permitiu uma maior dedicação às funcionalidades. Entre suas funcionalidades a possibilidade de cadastramento do usuário via *web*, e sua autenticação durante a utilização do cliente VoIP, forneceu maior segurança na comunicação já que exige a utilização de senha, e não permite duplicação de usuários sendo disponíveis somente os usuários cadastrados na *web*.

A utilização do cliente VoIP foi facilitada já que este cliente pôde ser escolhido elo usuário, com a única restrição de o software permitir o cadastro do servidor *proxy*. Isso permite a inclusão de clientes VoIP já existentes para dispositivos móveis, entretanto não são muitos os softwares com essa finalidade.

A integração do VoIP às redes sem fio ocorreu de maneira natural, já que o servidor atendeu a redes convencionais e sem fio, a qualidade da comunicação foi boa, já que apresentou baixo nível de ruído e poucas interrupções.

Esse resultado foi obtido durante testes efetuados entre a conexão de um cliente VoIP instalado no *iPaq* com acesso a rede *Wi-Fi* e um notebook também utilizando essa mesma estrutura de rede. Foram feitas ligações a partir do *iPaq* e recebidas pelo notebook, como também o inverso. Isso permitiu obter os resultados observando a usabilidade da ferramenta.

Este projeto possibilitará um melhor resultado com a adição de novas funcionalidades como a conexão à um servidor *DNS* para a localização de usuários cadastrados em outros servidores *proxy*, a implementação de comunicação via mensagem de texto - IM (*instant messeger*) e a possibilidade de conexão desse servidor a telefonia convencional.

Referências

ANDRADE, Cid R.; YAMAMOTO, Marcelo A.; SALGUEIRO, Vicente. **Redes Cliente-Servidores.** São Paulo, 2003. Disponível em: < http://blog.cidandrade.pro.br/redes-cliente-servidor/>. Acesso em: 02 Jun. 2008.

BANDEIRA, Alessandra B. **Distribuição de Weibull na emulação de canal de redes WLAN para avaliação de VoIP**. 2007. 101 f. Trabalho de Conclusão de Curso (Pós-Graduação *Stricto-Sensu*) - Centro de Ciências Exatas, Ambientais e de Tecnologias, Pontifícia Universidade Católica de Campinas, 2007

BERNAL F., Huber. **Telefonia IP**. 2008.

http://www.teleco.com.br/tutoriais/tutorialtelip/pagina_3.asp Acessado em : 20 nov de 2009;

BOCHENSKI, B. **Implementando sistemas cliente/servidor de qualidade**. São Paulo: Makron, 1995

CANSADO, Jacinto C. **A Tecnologia Bluetooth Voltada para Aplicações com Sensores**. 2001. 31f. Trabalho de Conclusão de Curso (Graduação) - Sistemas Digitais, Escola Politécnica da Universidade de São Paulo, 2001.

COLCHER, Sérgio et al. **VOIP: Voz sobre IP**. Rio Janeiro: Campus, jan. 2005. 288p

DEITEL, Harvey M.; DEITEL, Paul J. **Java:** como programar. 4. ed. Porto Alegre: Bookman, 2003. 1386p.

GONÇALVES, André M.; HOMMERDING, Roberto. **Implementação didática de telefone**
VoIP por software utilizando protocolo SIP. 2006. 62 f. Trabalho de Conclusão de Curso
(Graduação) – Ciência Tecnologia em Eletrônica, Universidade Tecnológica Federal do Paraná, 2006.

LOURENÇO, Rogério B. **Protocolos VoIP para Redes Convergentes**. 2007. 150 f. Trabalho de Conclusão de Curso (Pós-Graduação *Stricto-Sensu*) - Centro Tecnológico, Engenharia de Telecomunicações da Universidade Federal Fluminense, 2007

HERSENT, O., GUIDE, D., PETIT, J-P. **Telefonia IP: Comunicação Multimida Baseada em Pacotes**. São Paulo: Editora Prentice Hall. 2002.

HES, Andre J; TEEFFELEN,Ronald. **Implementing Voice over IP.** Suíça, 2001.
Disponível em: <http://www.upgrade-cepis.org/issues/2001/3/upgrade-VII.3.pdf>. Acesso
em: 21 mar. 2008.

LIMA, Almir Wirth. **Telecomunicações multimídia**. Rio de Janeiro: Book Express, 2001

MATSUZAKA, Marcelo K.; HASEGAWA, Márcio G. **Simulação de Threads em Ambiente Multi-core**. 2008. 37 f. Trabalho de Conclusão de Curso (Graduação) – Instituto de Matemática, Departamento de Ciências da Computação, Universidade de São Paulo, 2008.

OLIVEIRA, R. A. Rabello, et al. **Caracterização de Topologias Dinâmicas no Bluetooth.** Departamento de Ciência da Computação, Universidade Federal de Minas Gerais.
Disponível em: <http://www.dcc.ufmg.br/~rabelo/rabelo-wcsf2001.pz.gz>. Acesso: em 26 mai 2008.

REICHERT , Romeu H**. Transmissão De Voz Sobre Redes De Pacotes** ESTUDO DE CASO COM H.323 Novo Hamburgo, 2004.

RIBAS, Rodrigo G. **O VoIP e suas Soluções** . 2006. 51 f. Trabalho de Conclusão de Curso (Pós-Graduação *Lato-Sensu*) – Departamento de Computação Especialização em Desenvolvimento de Aplicações para Web, Universidade Estadual de Londrina.

SIQUEIRA, Ethevaldo. **Telecom World: a crise acabou**,
http://www.estadao.com.br/rss/tecnologia/colunas/2003/out/19/25.htm. Acesso em: 21 mar de 2008.

TANENBAUM, Andrew S. **Redes de Computadores.** Rio de Janeiro: Elsevier, 2003. 945p.

WANG, Runsheng; HU Xiaorui. **VoIP Development in China**. IEEE Computer Society,
New York: v.37, n.9, p.30-37, set. 2004.

ZEINDIN, DENISE CARLA A. **A Tecnologia do Futuro Wi-Fi (*Wireless Fidelity*).** Blumenau, 2008.