UNIVERSIDADE DO ESTADO DO AMAZONAS - UEA

ESCOLA SUPERIOR DE TECNOLOGIA

ENGENHARIA DE COMPUTAÇÃO

JOÃO GUILHERME ALVES MARTINEZ

# DESENVOLVIMENTO DE UMA APLICAÇÃO VOIP BASEADA NO PROTOCOLO SIP

Manaus

2011

**JOÃO GUILHERME ALVES MARTINEZ**

# DESENVOLVIMENTO DE UMA APLICAÇÃO VOIP BASEADA NO PROTOCOLO SIP

Trabalho de Conclusão de Curso apresentado à banca avaliadora do Curso de Engenharia de Computação, da Escola Superior de Tecnologia, da Universidade do Estado do Amazonas, como pré-requisito para obtenção do título de Engenheiro de Computação.

Orientador: Prof. M. Sc. Jucimar Maia da Silva Júnior

Manaus

2011

***Universidade do Estado do Amazonas - UEA***

***Escola Superior de Tecnologia - EST***

*Reitor:*

***José Aldemir de Oliveira***

*Vice-Reitor:*

***Marly Guimarães Fernandes Costa***

*Diretor da Escola Superior de Tecnologia:*

***Mário Augusto Bessa de Figueirêdo***

*Coordenador do Curso de Engenharia de Computação:*

***Danielle Gordiano Valente***

*Coordenador da Disciplina Projeto Final:*

***Raimundo Corrêa de Oliveira***

*Banca Avaliadora composta por:* *Data da Defesa: 15/12/2011.*

***Prof. M.Sc. Jucimar Maia da Silva Júnior*** *(Orientador)*

***Prof. M.Sc. Jorge Luiz Silva Barros***

***Prof. M.Sc. Ernande Ferreira de Melo***

## CIP - Catalogação na Publicação

M385d MARTINEZ, João Guilherme Alves[12pt]
DESENVOLVIMENTO DE UMA APLICAÇÃO VOIP BASEADA NO PROTOCOLO SIP / João Guilherme Alves Martinez; [orientado por] Prof. MSc. Jucimar Maia da Silva Júnior - Manaus: UEA, 2011.

52 p.: il.; 30cm

Inclui Bibliografia

Trabalho de Conclusão de Curso (Graduação em Engenharia de Computação). Universidade do Estado do Amazonas, 2011.

CDU: 004

**JOÃO GUILHERME ALVES MARTINEZ**

# DESENVOLVIMENTO DE UMA APLICAÇÃO VOIP BASEADA NO PROTOCOLO SIP

Trabalho de Conclusão de Curso apresentado à banca avaliadora do Curso de Engenharia de Computação, da Escola Superior de Tecnologia, da Universidade do Estado do Amazonas, como pré-requisito para obtenção do título de Engenheiro de Computação.

**Aprovado em: 15/12/2011**

BANCA EXAMINADORA

---

**Prof. Jucimar Maia da Silva Júnior, Mestre**
*UNIVERSIDADE DO ESTADO DO AMAZONAS*

---

**Prof. Jorge Luiz Silva Barros, Mestre**
*UNIVERSIDADE DO ESTADO DO AMAZONAS*

---

**Prof. Ernande Ferreira de Melo, Mestre**
*UNIVERSIDADE DO ESTADO DO AMAZONAS*

# Agradecimentos

Primeiramente à Deus por sempre me guiar durante meu caminho.

Aos meus pais que sempre me permitiram tomar minhas próprias decisões a respeito do meu futuro, aconselhando, apoiando e ajudando.

Ao meu orientador, professor Jucimar Maia, pela paciência, compreensão, ajuda e direcionamento para que eu concluísse este trabalho.

À minha amiga Clarice e aos demais amigos, colegas e familiares que me acompanharam e ajudaram nessa conquista.

# Resumo

Este trabalho descreve o desenvolvimento de uma aplicação baseada no protocolo SIP para comunicação VoIP. O SIP tem como principal função realizar a sinalização entre as partes envolvidas antes de iniciar qualquer transação de dados de comunicação, ou seja, ele estabelece, configura ou encerra uma chamada. A aplicação realiza chamadas diretas entre dois clientes SIP, baseando-se pelos seus respectivos endereços IP. Ao final foram realizados testes para medir a qualidade das chamadas a partir da experiência do usuário.

# Abstract

This work describes the development of an application based on SIP protocol for VoIP communication. SIP has as its main function to perform signaling between the parties involved before any data communication transaction, therefore, it establishes, configures, or terminate a call. The application performs direct calls between two SIP clients, based on their respective IP addresses. At the end were performed tests to measure the quality of calls by the user's experience.

# Sumário

# Lista de Tabelas

# Lista de Figuras

# Capítulo 1

# Introdução

Com a evolução da Internet e o aumento cada vez mais rápido e abrangente da rede mundial que conecta cada vez mais pessoas, a proposta de digitalização das comunicações ganha força e dentre elas a comunicação por voz utilizando as redes de computadores no lugar da telefonia tradicional que seada na tecnologia da permutação de circuitos. A abordagem mais referenciada para possibilitar tal comunicação é o VoIP (Voz sobre IP), que se caracteriza pelo transporte da voz usando o protocolo IP.

O VoIP se torna possível e viavél pela interoperabilidade do protocolo IP no transporte de qualquer tipo de informação pela Internet, mas deve-se observar que ao utilizar-se de uma rede que não é exclusiva para o transporte de voz, surgem novos desafios para serem superados como por exemplo: Perda de pacotes, chegada não-ordenada de pacotes no destino, largura de banda mínima, compressão e descompressão da mídia, dentre outros.

Em todo processo de comunicação, antes de iniciar qualquer transação de dados é necessária a sinalização entre as partes envolvidas para estabelecer, configurar ou encerrar uma conexão. Um exemplo simples do processo da sinalização ocorre quando uma pessoa faz uma ligação telefônica para outra pessoa. Primeiro o telefone do destinatário começa a tocar indicando que alguém o está chamando, dando-lhe um sinal. Quando atende o telefone é iniciada a conversa. Ao final quando um dos dois desliga o telefone, o telefone da outra pessoa toca de uma maneira diferente sinalizando que a ligação foi encerrada.

Como forma de padronizar os protocolos e padrões da Internet, o *Internet Engineering Task Force* (IETF) define e publica uma série de padrões para os desenvolvedores. Dentre

diversos protocolos existentes para tal finalidade, destaca-se o SIP (Session Initiation Protocol) definido pela RFC 3261.

Segundo [Collins2000], por ser um protocolo simples, o SIP permite facilidade e agilidade no desenvolvimento de aplicações, além de acelerar o estabelecimento de chamadas e também permitir a transação de informações adicionais que possam ser utilizados para outros serviços adicionais.

## 1.1 Justificativa

A comunicação por VoIP é uma das formas mais difundidas de comunicação por voz pela Internet devido a sua interoperabilidade com o protocolo IP e pelo seu baixo custo por ser viabilizado pelas redes de computadores espalhadas e interconectadas pelo mundo inteiro ao contrário do sistema de telefonia tradicional que utiliza-se da permutação de circuitos.

Para efetuar qualquer comunicação, seja de telefonia ou até mesmo para a transferência de dados, é necessário que ambas as partes estejam cientes e pré-dispostas para receber e enviar os dados. Assim surgiram os protocolos de sinalização, para efetuar e tratar a conexão entre duas aplicações.

A aplicação desenvolvida neste trabalho utiliza-se do SIP como protocolo de sinalização e é desenvolvida na linguagem java, por ser multiplataforma e apresentar bibliotecas avançadas para manipulação de pacotes, captura e tratamento de áudio e criação de interfaces gráficas.

## 1.2 Objetivo

Desenvolver uma aplicação VoIP baseada no protocolo SIP utilizando a linguagem Java.

### 1.2.1 Objetivos Específicos

- Desenvolver módulo para gravação e captura de áudio;

- Desenvolver módulo para envio e recebimento de dados via sockets UDP;
- Obter um algoritmo para realizar a compressão e descompressão de áudio wave para codificação G711 $\mu$-*law* e vice-versa;
- Desenvolver módulo para encapsular os dados de áudio com os protocolos RTP/RCTP;
- Desenvolver módulo tratar sessões SIP;
- Desenvolver uma interface gráfica;

## 1.3 Metodologia

Para uma aplicação VoIP, é necessária a captura e tratamento do áudio. Para tal será utilizada a biblioteca Java Sound por ser simples e de fácil manipulação.

Para a manipulação, envio e recebimento dos pacotes de dados, serão utilizadas bibliotecas nativas do java.

Devido ao grande volume dos dados de áudio no formato *Wave*, será feita a compressão antes do envio e a descompressão após o recebimento desses dados. Será escolhido um formato para a compressão que seja simples e com implementações disponíveis em java.

Para maior controle e tráfego dos dados, os protocolos RTP (*Real-Time Transport Protocol*) e RCTP (*Real-Time Control Transport Protocol*) também serão utilizados para o encapsulamento dos dados de áudio durante a transmissão e recebimento dos pacotes.

Será desenvolvido o módulo SIP para a sinalização entre as partes para realizar, encerrar ou configurar a transação dos dados de voz.

Como um meio facilitador para o usuário, será criada uma interface gráfica utilizando a biblioteca Swing do Java com o intuito de melhorar a manipulação da aplicação pelos usuários.

Por fim serão realizados testes com aplicação realizando a comunicação entre duas máquinas em uma rede local e na Internet.

## 1.4 Organização do Trabalho

No capítulo 2 é feita uma visão geral sobre a importância do VoIP nas telecomunicações, os princípios básicos de funcionamento e os codecs para codificar os dados de áudio.

No capítulo 3 são apresentados os protocolos utilizados pela aplicação. Dentre eles o SIP, o SDP e o RTP.

No capítulo 4 é apresentado desenvolvimento e o modelo de classes da aplicação desenvolvida, além dos ambientes onde os testes foram realizados e os respectivos resultados obtidos.

Ao final do trabalho é apresentada uma conclusão referente ao aprendizado obtido com o desenvolvimento deste trabalho.

# Capítulo 2

# Voz sobre IP

Este capítulo aborda as vantagens do uso do protocolo IP na comunicação por voz, os parâmetros de qualidade necessários e apresenta algumas técnicas de codificação da voz como auxílio.

O termo Voz sobre IP, ou VoIP, é utilizado para designar o processo de comunicação por voz utilizando as redes baseadas no protocolo IP. Ao contrário da telefonia analógica tradicional, o VoIP oferece a possibilidade da convergência de dados de voz com outros tipos de dados em uma única tecnologia, a Internet.

## 2.1 Vantagens do protocolo IP na comunicação por voz

São várias as vantagens em se utilizar o protocolo IP para o transporte da voz ao invés do sistema de telefonia fixa. Dentre as principais vantagens estão:

- Baixo custo. As redes IP possuem um custo de instalação bem mais baixo do que o sistema tradicional de telefonia.
- Devido ao grande número de redes de computadores já instaladas em empresas e instituições, torna-se vantajoso aproveitar essa estrutura já pronta para fazer uso do VoIP.

- Pela falta de dependência das operadoras telefônicas, é possível utilizar softwares de arquitetura aberta, sem custo de licença ou taxas de serviço.
- Convergência de voz e dados em uma única rede, centralizando manutenção e gerência.
- Possibilidade de integrar serviços adicionais como atendimento ao cliente, chat, vídeo, correio de voz, reconhecimento de voz e etc.
- Grande disponibilidade e área de abrangência das redes IP.

## 2.2 Parâmetros de qualidade

Ao contrário da rede de telefonia tradicional, as redes IP trafegam diversos tipos de dados além da voz, gerando assim um tráfego de dados que pode comprometer a qualidade das chamadas. Antes de ser um potencial negócio para as comunicações por voz, o VoIP deve ser capaz de oferecer a mesma confiabilidade e qualidade de voz oferecida pela rede tradicional de telefonia pública.

A recomendação P.800 da ITU-T descreve um sistema de avaliação da qualidade da voz que varia de 1 a 5 chamada de *Mean Opinion Score* (MOS). Para determinar o ranking de uma determinada técnica em tal escala, um grupo de pessoas ouve amostras de voz e participam em conversações para ao final avaliar. No final a média das avaliações determina o ranking final daquela codificação. Como referência, para uma voz de qualidade espera-se um ranking de 4 ou superior.

O protocolo IP foi designado para o transporte de dados, ou seja, é tolerante a atrasos, desde que os dados cheguem de forma correta ao seu destino mesmo que para isso seja preciso retransmitir pacotes perdidos ou reorganizar os dados no destino. Mas para melhor avaliar a qualidade do transporte da voz usando o IP, é preciso olhar para cada parâmetro referente á essa transmissão.

Os atrasos são intoleráveis quando se trata do transporte de voz, porque quando esse se torna perceptível a comunicação pode ser comprometida já que uma das partes pode não compreender a fala do outro. Para isso surgiu a recomendação G.114 da ITU-T que

especifica que o *round-trip delay* (tempo para informação ir e retornar do destino) deve ser de no máximo 300ms para telefonia.

O *jitter* (variação do atraso) também é um problema que deve ser minimizado o máximo possível quando se fala em comunicações de tempo-real, porque mesmo com algum delay é possível fazer ajustes mas se torna complicado quando o mesmo fica variando.

Outro problema que surge é a perda de pacotes, que para uma boa transmissão deve ser evitada ao máximo mas ainda é tolerável desde que represente apenas 5% do total de pacotes enviados [Collins2000]. É muito mais válido para a comunicação perder alguns poucos pacotes do que iniciar um processo de reenvio, espera e reorganizaação da mídia que gera um *delay* perceptível.

## 2.3 Técnicas de codificação de voz

A codificação da voz é o processo onde uma onda digital constituída de 0s e 1s é construída para representar uma forma de onda analógica. Tal processo envolve a codificação do sinal analógico para o digital e vice-versa. As técnicas de codificação de voz se tornam importantes quando se fala em VoIP pois menos bits são utilizados para transmitir a voz reduzindo custo e largura de banda necessária.

Apesar de darem boas perspectivas, as técnicas diminuem a qualidade do sinal quanto menor for a largura de banda requerida ao final do processo, ou seja, existe uma relação (não necessariamente linear) entre a largura de banda que pode ser alcançada com uma codificação e a qualidade da voz que será transmitida. Os algoritmos utilizados também influenciam no resultado da qualidade de voz, podendo fornecer a mesma largura de banda que outras técnicas mas deteriorando poder de processamento.

Para gerar uma representação digital de uma forma de onda analógica primeiro é necessário pegar amostras discretas da onda e representar cada com uma quantia de bits, como demonstrado no exemplo da figura 2.1. Para recriar uma onda são necessárias infinitas amostras, mas a proposta das codificações é para se obter um número suficiente para posteriormente junto com alguns cálculos matemáticos recriar algo bem próximo do sinal original. O teorema de amostragem de Nyquist diz que um sinal pode ser reconstruído se estiver amostrado em no mínimo o dobro da máxima frequência. Por exemplo, para um

sinal de 4KHz é preciso no mínimo 8 mil amostras por segundo. A voz humana durante a fala varia de 300Hz a 3.8KHz, então assume-se um máximo de 4KHz e retira-se qualquer amostragem em frequências maiores do que isso.

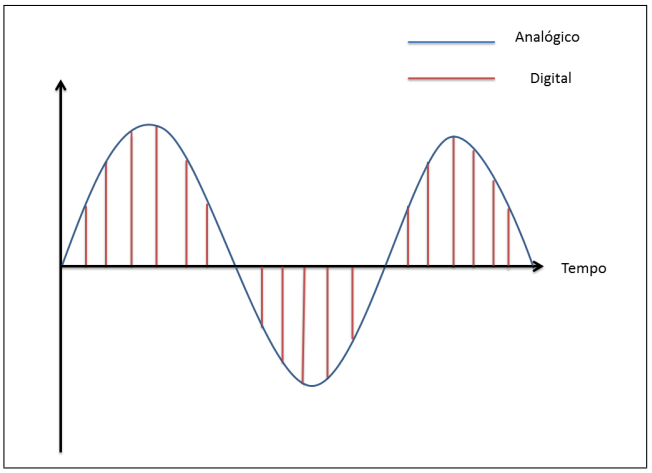


**Figura 2.1**: Exemplo de representação digital de um sinal

Um outro fator importante é a quantização, ou seja, quantos bits serão usados para representar cada amostra. Assumindo por exemplo que utilizemos 3 bits para representar cada amostra, estaremos limitados a 8 níveis, então num dado exemplo onde o sinal possui valor de 7.4 por exemplo, com a representação digital passa a ser 7, assim perdendo qualidade no sinal quando for reconstruído. Essa diferença entre a amostra analógica original e a amostra digital é chamada de ruído de quantização. Para contornar tal problema pode-se utilizar mais bits para representar melhor cada amostra, mas note que isso exigirá mais largura de banda para se que se possa transmitir devido ao aumento da qualidade.

Surgem dois efeitos ao se aplicar quantização uniforme no sinal: Muitos bits para representar cada amostra e pessoas que falam mais alto vão soar melhor do que os mais baixos porque o ruído de quantização é menor, por exemplo, uma amostra de valor 11.2 é representada por 11 (-1,8%) é bem menor que a diferença entre uma amostra de valor 2.2 para 2. Olhando tal efeito, pensou-se em aplicar quantização não-uniforme, ou seja usar uma diferença menor entre os níveis baixos e uma maior diferença para altos níveis, economizando assim largura de banda.

Antes de citar alguns codificadores de voz, é preciso diferenciá-los quanto a sua

categoria, que são de três tipos:

- *Waveform Codecs* - O sinal original é codificado e transmitido para seu destino onde lá é reconstruído. Geram saídas de alta qualidade e não são muito complexos mas por outro lado consomem muita largura de banda.

- *Source Codecs* - Traduz o sinal de entrada como um modelo matemático de como a voz é produzida e transfere tais parâmetros para seu destino, que faz as operações reversas utilizando os valores recebidos para reconstruir o sinal.

- *Vocoders* - Operam com poucos bits mas produzem sinais com sonoridade sintética. Mais utilizados para aplicações privadas e militares.

- *Hybrid Codecs* - Buscam mesclar as técnicas de modelagem com algumas amostras do sinal de entrada para melhor recriar os sinais em seu destino.

O G.711 é uma das codificações de voz mais utilizadas [Collins2000]. Esta dentro da categoria dos waveform codecs e é a técnica utilizada nas redes telefônicas. Por utilizar quantização não-uniforme utiliza 8 bits para cada amostra assim necessitando de uma taxa de 64kbps, chamado de *Pulse-Code Modulation* (PCM). Possui um MOS de 4.3.

O G.711 possui dois variantes: *A-law* - Utilizado na maioria dos países, dá prioridade para sinais de baixo nível; $\mu$*-law* - Utilizado na América do Norte, que dá mais níves quantizadores para sinais de alto nível.

Uma outra forma de codificar a voz consiste em enviar a diferença entre o valor atual do sinal e o valor anterior, para que o destino calcule o sinal resultante ao invés de recebê-lo. Tal técnica é chamada de *Differential* PCM (DPCM). També há uma versão mais avançada, a *Adaptative Differential* PCM (ADPCM), que além de transmitir a diferença, faz uma previsão da voz a partir da variação da mesma sobre o tempo, se tornando um método mais inteligente que consome menor largura de banda. A recomendação G.721 da ITU-T utiliza-se de ADPCM utilizando 32kbps, mas foi suspenso estando agora em substituição o G.726 que recebe *A-law* ou *mu-law* como entrada e converte para taxas de 16 a 40 kbps. Para 32kbps possui um MOS de 4.0. São algoritmos da categoria waveform codecs que não inserem delay.

*Analysis-by-Synthesis* (AbS) codecs são híbridos e utilizam-se da predição linear do modelo de filtro do trato vocal. Para cada excitação do sinal são utilizadas aquelas que mais se aproximam da onda original. O CELP implementa um filtro e contém um dicionário de vetores acústicos, onde cada um possui elementos que representam características do sinal de excitação. O Code-excited linear prediction (CELP) envia informações indicando os coeficientes do filtro e o ponteiro para o vetor escolhido, e no seu destino, que também possui o dicionário, é possível recriar o sinal com boa qualidade. O G.728 especifíca o *Low Delay-Code-Excited Linear Predictor* (LD-CELP), que é um codificador retroativo-adaptativo pois utiliza-se de amostras passadas para determinar os coeficientes do filtro. Opera utilizando 5 amostras por vez, ou seja, 5 amostras são necessárias para determinar um vetor dicionário e os coeficientes.

Como é necessário indicar o vetor escolhido no dicionário, é preciso enviar seu index. Como possuem 1024 índices, tal índice ocupa 10 bits no momento da transmissão. O G.728 ocupa uma largura de banda de 16kbps e apresenta MOS de 3.9. Seu *delay* introduzido é quase imperceptível.

O G.723.1 especifíca um codec de voz que pode operar a 6.3 ou 5.3 kbps. O codificador primeiro recebe o sinal quantizado por PCM e então opera em 240 amostras por vez, ou seja utilizando 30ms de voz por vez introduzindo assim um atraso de 30ms e mais 7.5ms do atraso do algoritmo. Mais cálculos e filtragens são feitas sobre o sinal, além de utilizar-se de um dicionário como auxílio. Sua principal vantagem é a baixa largura de banda necessária, mas o atraso que é inserido na codificação pode ser um problema.

O G.729 opera a 8kbps utilizando quadros de 10ms introduzindo 5ms de atraso resultando num atraso algoritmico de 15ms. É um codificador de implementação complicada que se utiliza de predição de coeficientes, índices de dicionário e parâmetros de ganho. Apresenta um MOS de 4.0.

No momento da escolha de um codificador para uma determinada aplicação, é preciso ponderar entre largura de banda exigida por cada um, atraso introduzido e qualidade do sinal, principalmente devido ao fato que o MOS é ranqueado em laboratório, enquanto que na prática real as redes por onde os pacotes irão trafegar podem inserir mais atrasos e induzir a perda de pacotes.

# Capítulo 3

# Protocolos

Toda e qualquer aplicação que opere na Web precisa trabalhar com protocolos para que a troca de dados na Internet ocorra com sucesso. Protocolos nada mais são do que padrões e regras a serem seguidas para a disposição e troca de dados.

Quando tratar-se da Internet, é necessário referenciar a *Internet Society*, uma organização sem fins lucrativos cujo objetivo é manter a Internet viva e em crescimento. São responsáveis pela padronização de protocolos, documentos, especificações e aprovação de novas tecnologias relacionadas á Internet.

Primeiro um padrão da Internet é divulgado como um rascunho, e até que esteja completo pode ser publicado como uma RFC com um número de identificação. O primeiro passo ocorre quando a RFC se torna um padrão proposto, que é estável, completo, bem compreendido e que atraiu interesse da comunidade, sem demonstrações ou implementações obrigatórias. O segundo passo acontece quando a RFC se torna um rascunho padrão. Para que isso aconteça é necessário que pelo menos duas implementações sejam demonstradas e que seja comprovada sua interoperabilidade. O passo final, quando a RFC se torna um padrão oficial, é quando a especificação já é madura, estável, legível e que possa ser implementada em larga escala sem problemas.

A figura 3.1 apresenta uma pilha de protocolos utilizados na aplicação e divididos pela camada de rede em que atuam.

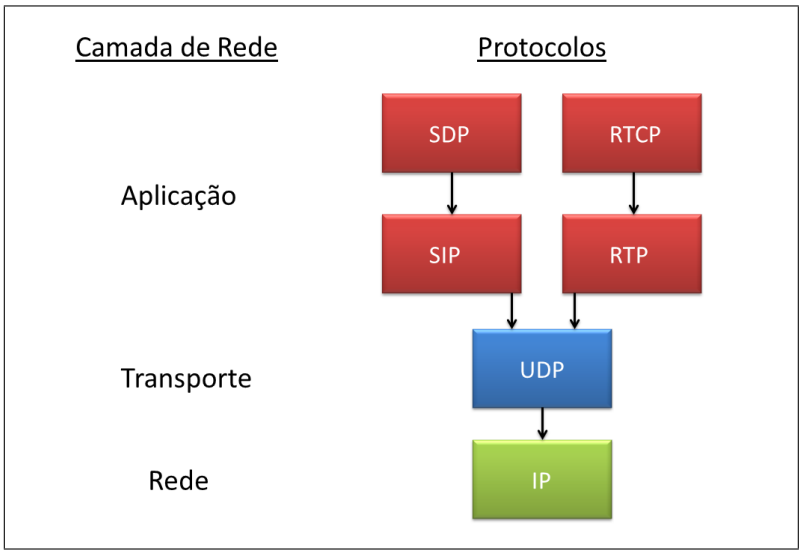


**Figura 3.1**: Pilha de protocolos

## 3.1 IP

A Internet pode ser referenciada como um conjunto de redes de computadores, públicas ou privadas. As redes públicas comunicam-se utilizando o protocolo IP, e as redes privadas para se comunicarem com as públicas também precisam utilizar o IP, por isso sua grande difusão e abrangência, além do fato de ter sido adotado pelas grandes desenvolvedoras de sistemas computacionais [Collins2000].

O protocolo IP é um protocolo de roteamento para encaminhar pacotes de dados. Dessa forma ele opera em conjunto com outros protocolos de camadas mais altas que utilizam-se dele para encaminhar seus pacotes, e também opera com protocolos de camadas mais baixas que lidam com a transmissão física dos dados. Tais camadas fazem referência ao modelo OSI para redes de computadores que define 7 camadas para a comunicação dos dados, onde cada camada encapsula e organiza os dados de acordo com sua funcionalidade, para que no seu destino, as mesmas camadas façam o trabalho reverso até que os dados sejam interpretados de maneira correta. O protocolo IP situa-se na camada de rede, para endereçar e garantir o roteamento dos pacotes.

Para que funcione, o protocolo IP adiciona ao pacote de dados um cabeçalho, que contém informações importantes sobre a origem e o destino desse pacote na Internet. O pacote de dados junto do cabeçalho é denominado de datagrama IP. Os roteadores utilizam

as informações nesse cabeçalho para encaminhar os pacotes pela rede. O IP não garante a entrega de seus pacotes pela rede, podendo estes se perderem ao longo do caminho.

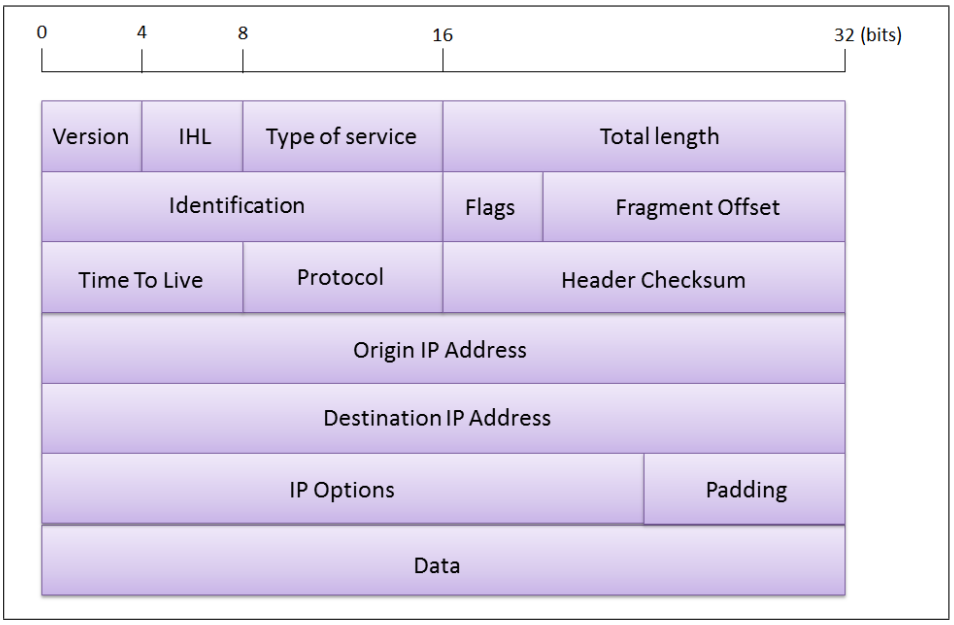


**Figura 3.2**: Cabeçalho IP

Na estrutura do cabeçalho, conforme a figura 3.2, vem a versão do protocolo IP, que no caso hoje utiliza-se a versão 4. Os campos *Identification, Flags* e *Fragment Offset* são utilizados quando o datagrama é preciso ser dividido por ser maior que a disponibilidade de banda de um link. O primeiro identifica o sub-pacote, o segundo serve para indicar se um datagrama pode ser dividido ou não e no caso de já ser um datagrama particionado se este é a última parte. O terceiro campo serve para indicar a posição de cada sub-pacote para que seja montado de forma correta.

O campo *Time to Live* indica por quanto tempo o pacote deve permanecer na rede antes de ser descartado. O campo protocolo indica o protocolo pelo qual este datagrama responde na camada superior, acima desta. Também estão presentes os endereços IP de origem e destino.

## 3.2 Protocolos da camada de Transporte

Na camada de transporte, o IP trabalha com um dos seguintes protocolos: TCP e UDP. O primeiro tem como principal função garantir o envio de todos os pacotes em sequência, sem omissão e sem erros. Para isso utiliza-se de confirmação de recebimento e retransmissão de pacotes. O UDP envia os dados para seu destino sem garantias de entrega, mas caso uma resposta a uma requisição não chegue em um tempo pré-determinado a aplicação pode retransmitir o pacote UDP, por isso cada um é identificado para que não ocorra duplicação de pacotes.

Ao transportar voz pelo IP, o protocolo UDP se torna uma melhor escolha, pelo fato de que numa conversa, a perda de alguns poucos pacotes não implica em uma perda perceptível na qualidade do serviço, mas é intolerante a atrasos, por isso o protocolo TCP não é favorável, pois sua política de confirmação e retransmissão de pacotes gera atrasos indesejáveis para a transmissão. Mesmo com a possível perda de pacotes ou chegada dos mesmos fora de ordem utilizando-se UDP, são casos mínimos em que algumas políticas de QoS podem ser aliados e ainda assim bem mais fortes do que utilizando-se o TCP [Collins2000].

A estrutura do cabeçalho UDP está demonstrado na figura 3.3.

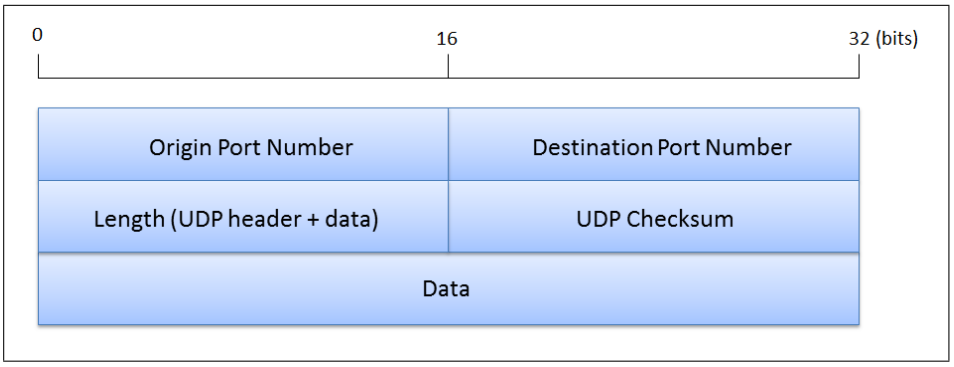


**Figura 3.3**: Cabeçalho UDP

### 3.2.1 RTP (Real-Time Transport Protocol) e RTCP (Real Time Control Protocol)

O RTP, descrito pela RFC 3550, que engloba também o RTCP (RTP Control Protocol), são protocolos para serviços de transporte de aplicações de tempo real, que operam numa camada acima do UDP, sendo então de grande auxílio para as falhas do UDP como perda de pacotes e ordenação dos mesmos pois fornecem dados como numeração sequencial, momento da saída do pacote de sua origem, dentre outros que poderão ser usados pelas camadas superiores.

O RTCP carrega dados relevantes à qualidade da transferência dos pacotes, como a perda e a ordem, operando em conjunto com o RTP que é responsável por carregar os pacotes de voz em si. Sempre que uma sessão RTP é aberta, uma RTCP também é, e utilizam comumente as portas 5004 e 5005.

Ao enviar pacotes de voz, o RTP anexa um cabeçalho para identificar o tipo de codificação de voz utilizada e então é enviado para o UDP para que este adicione seu cabeçalho, e que então passa para o IP e assim sucessivamente. A interpretação do tipo de codificação no cabeçalho RTP está especificada na RFC 3550 que relaciona o código com cada codificação.

O cabeçalho RTP inclui informações necessárias para a aplicação de destino reconstruir a amostra de voz. Os campos do cabeçalho são:

- *Version* (V) - Versão do protocolo RTP. Valor = 2.
- *Padding* (P) - Campo de um bit que indica se há um ou mais octetos descartáveis no fim do cabeçalho, já que este precisa ter 32 bits de tamanho, e em alguns casos seja preciso colocar bits inutilizados para preencher tal tamanho.
- *Extension* (X) - Um bit que indica se o cabeçalho é seguido por uma extensão, que pode ser adicionado de acordo com a necessidade de cada codificação.
- CSRC *Count* (CC) - Quatro bits para indicar o número dos source identifiers.
- *Marker* (M) - Marcador de um bit utilizado para algumas codificações como auxílio. Por exemplo no caso de algumas codificações que usam supressão de silêncio, este bit

pode servir como marcador para indicar o primeiro pacote após o silêncio.

- *Payload Type* (PT) - Sete bits que indicam a codificação utilizada.
- *Sequence Number* - Dezesseis bits que iniciam com um número randômico e são incrementados em um a cada pacote RTP entregue. Possibilita detectar a perda de pacotes ou pacotes que estão chegando fora de ordem.
- *Timestamp* - Trinta e dois bits que indicam o momento em que a primeira amostra foi gerada, fornecendo assim uma referência para que as amostras sejam reproduzidas de forma sincronizada e sejam feitos os cálculos sobre o jitter.
- *Synchronization Source* (SSRC) - Trinta e dois bits que indicam a fonte do stream (participante).
- *Contributing Source* (CSRC) - Trinta e dois bits que indicam o valor SSRC de um contribuinte para a sessão. É utilizado quando a stream RTP vem de um mixer, então é usado para identificar a fonte original por detrás do mixer.

A estrutura do protocolo está representado na figura 3.4.

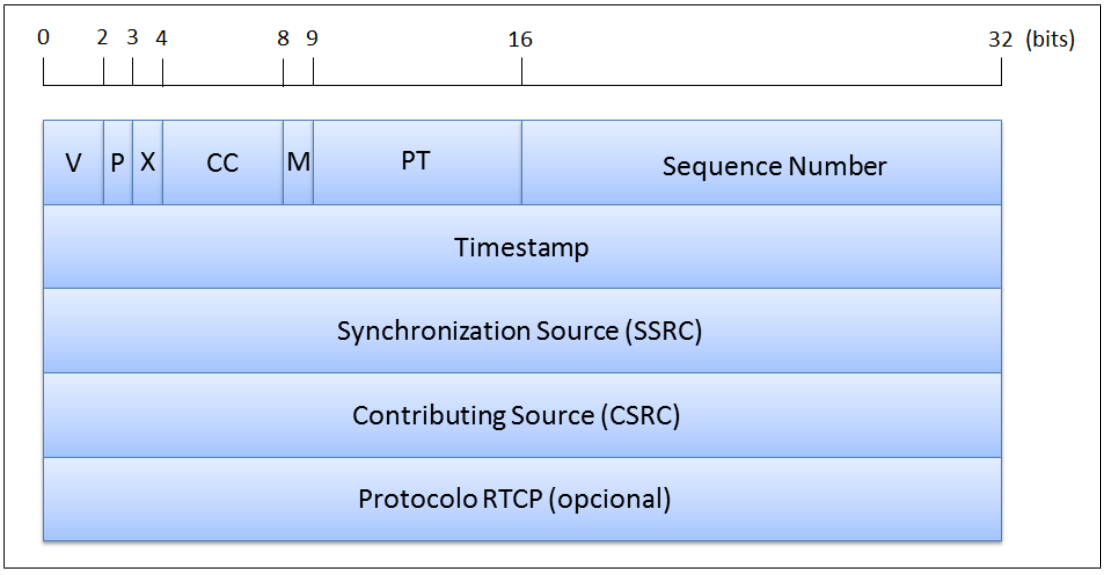


**Figura 3.4**: Estrutura do protocolo RTP

O RTP vem encaminhado pelo RTCP como protocolo auxiliar que habilita a troca periódica de informação entre os participantes de uma sessão com o objetivo de obter retorno relativo á qualidade. O RTCP define 5 diferentes tipos de pacotes.

- *Sender Report* (SR) - Para participantes ativos informarem estatísticas referentes à transmissão e recepção.
- *Receiver Report* (RR) - Para enviar estatísticas de recepção de membros que somente estão recebendo.
- *Source Description* (SDES) - Possui informações referentes ao participante. Particularmente possui o nome de identificação do participante, que difere do SSRC, já que este pode mudar no caso de um host reset, além de que uma fonte pode fornecer múltiplos streams como áudio e vídeo usando diferentes SSRC. Pode possuir outros dados como e-mail, telefone dentre outros.
- BYE - Indica o fim de uma participação na sessão.
- APP - Espaço destinado para que aplicações específicas utilizem caso necessário.

Os pacotes RTCP apesar de serem definidos de maneira individual são enviados em conjunto em um pacote composto, devido ao fato de que para quase toda transação todos os seus dados serão utilizados ou devem ser transacionados, mesmo que alguma das partes esteja vazia.

Quanto á estrutura de cada pacote, possuem os campos version e padding similar ao RTP. No pacote *Sender Report* (SR), o próximo campo é o *Report Counter* (RC) que diz quantos blocos relatórios estão anexados ao pacote de acordo com o número de participantes na conferência. Logo após vem o identificador do pacote RTCP, no caso 200 para o SR e um campo *lenght* com o tamanho total do pacote.

Para cada bloco relatório seguem alguns campos no pacote referentes a fonte de envio:

- SSRC da fonte.
- NTP *Timestamp* (padrão *Network Time Procotol*).
- RTP *Timestamp*, usa mesma lógica do pacote RTP.

- Número total de pacotes RTP enviados desde o início da sessão.*
- Número total de octetos *payload* enviados desde o inicio da sessão.*

*ambos são reiniciados se o SSRC da fonte for trocado.

Agora para os blocos de relatório seguem mais campos que são repetidos para cada participante:

- SSRC do participante.
- 8 bits indicando a fração de pacotes que foram perdidos desde o último relatório desse participante. A fração perdida é o número de pacotes perdidos dividido pelo número de pacotes esperados.
- Número de pacotes perdidos dessa fonte desde o início da sessão.
- Número de sequência do último pacote RTP recebido dessa fonte.
- Estimativa do *jitter*.
- 32 dos 64 bits do NTP *Timestamp* usados no último SR recebido da fonte. Indica se os relatórios dessa fonte estão sendo recebidos.
- Atraso desde recebimento do último SR.

A figura 3.5 representa a estrutura do protocolo RTCP.

O pacote RR é igual em estrutura ao pacote SR exceto seu código (201) e o fato de que somente possui seu SSRC como campo relativo a fonte de envio, já que este participante não envia dados à conferência.

O pacote SDES deve existir em todo pacote RTCP composto. Possui código 202. Diferente dos demais, possui ao invés do campo RC, possui o *Source Counter* (SC) que possui o número blocos de participantes que vão anexados ao pacote. Cada bloco possui o SSRC ou CSRC de cada e mais informações que são definidas pela RFC 3550 mas não sao obrigatórias exceto o CANONICAL NAME que é o identificador único para cada participante numa dada sessão.

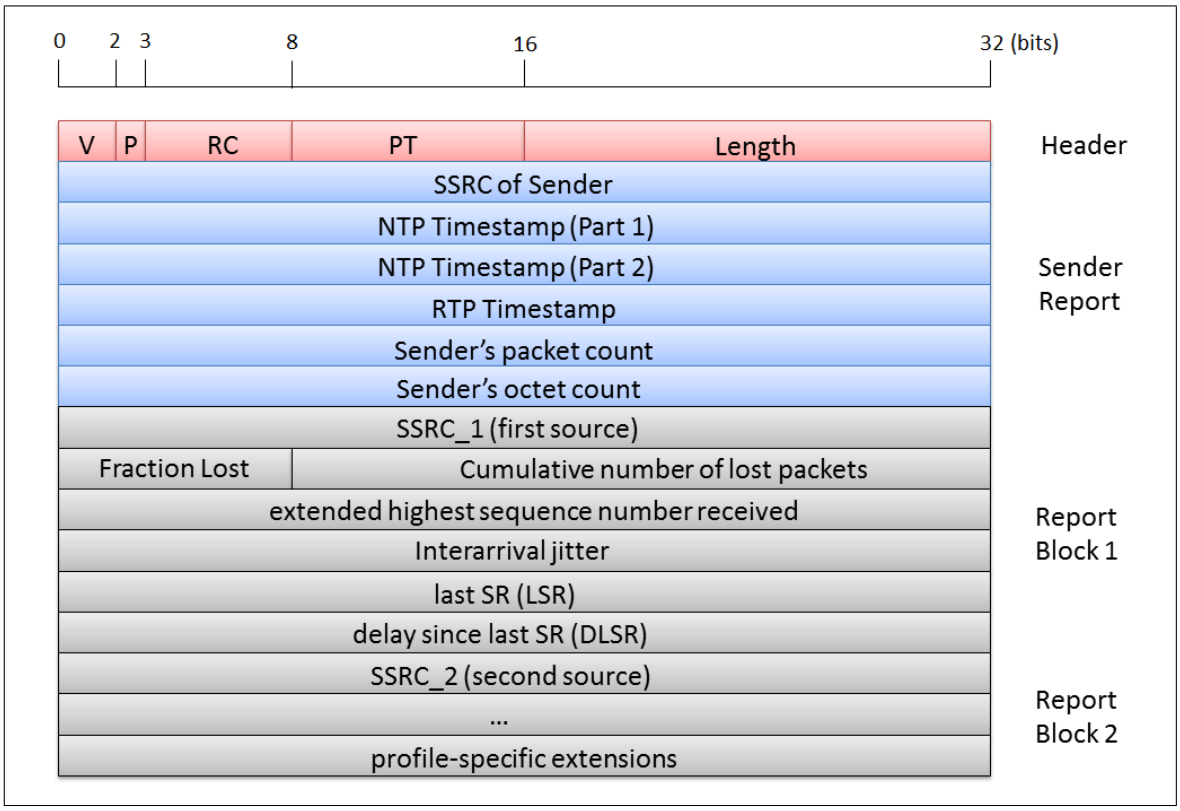


**Figura 3.5**: Estrutura do protocolo RTCP

O pacote BYE possui o campo SC para indicar quantos SSRC ele acompanha e ao final possui uma mensagem de texto para personalizar que deve ser precedida de seu tamanho no campo lenght. Possui código 203.

O pacote APP possui código 204 e como não é padronizado apresenta um nome em codificação ASCII e os dados.

## 3.3 SIP (Session Initiation Protocol)

Descrito na RFC 3261, é considerado um protocolo simples, mas que ganha força perante os desenvolvedores pelo simples fato que aplicações podem ser desenvolvidas mais rápido e estar disponível para os clientes mais cedo.

O SIP é um protocolo de sinalização que em conjunto com alguns outros protocolos tem como principal funcionalidade a configuração, modificação e encerramento de sessões multimídia pela Internet, e sua popularidade tem crescido tanto que é visto como o

futuro das aplicações VoIP. Foi criado para trabalhar em conformidade com qualquer outro protocolo de transporte que seja o encarregado pela mídia em si (exemplo: RTP), ou seja é trabalhado de forma separada da mídia.

No SIP estão definidas duas classes de entidades de rede: o cliente que envia as requisições SIP e o servidor que as responde, portanto trata-se de um protocolo cliente-servidor. As chamadas VoIP originam-se no cliente e terminam no servidor. Existem quatro tipos diferentes de servidor no SIP: servidor *proxy*, servidor de redirecionamento, servidor usuário-agente e servidor de registro.

- O servidor *proxy* lida com as requisições dos clientes ou encaminha para outros servidores, agindo como cliente e servidor.
- O servidor de redirecionamento retorna o endereço corrento do destinatário para o cliente que fez a requisição.
- O servidor usuário-agente recebe requisições e contacta o usuário. A resposta do usuário para o servidor resulta na resposta em nome do usuário.
- O servidor de registro recebe requisições de registro, assim o usuário pode deixar armazenado seu endereço onde está disponível.

### 3.3.1 Vantagens do SIP

O SIP se torna poderoso por sua simplicidade, buscando ater-se ao simples procedimento da sinalização, acelerando o estabelecimento das chamadas além de ter a capacidade de carregar informações adicionais, possibilitando assim que serviços melhores e mais completos sejam desenvolvidos passando a frente dos outros protocolos.

Existem algumas vantagens claras para se utilizar o SIP em uma aplicação VoIP.

- Faz parte do *Toolkit* da IETF, que projetou o protocolo focado no paradigma da Internet cumprindo seu papel e aproveitando-se de outros mecanismos da Internet para realizar tarefas adicionais, provendo assim alta flexibilidade já que sistemas SIP em conjunto com outro protocolos da Internet podem ser atualizados de maneira modular.

- Tem a responsabilidade somente de estabelecer uma sessão sem a necessidade de descrevê-la, ou seja, torna-se possível trabalhar com diversos protocolos para descrever uma sessão independente de seu tipo.
- Por atuar como um protocolo fim-a-fim focado na entrega do serviço de uma ponta a outra, torna-se eficiente já que os servidores SIP somente precisam rotear os pacotes sem a necessidade de processar os dados descritivos da sessão.
- Foi designado para que qualquer implementação possa interoperar com qualquer outra desenvolvida, tornando-se assim um protocolo genuinamente global.
- É altamante escalável já que a comunicação final entre as aplicações não precisa do servidor, permitindo que este manipule um grande número de sessões.
- A possibilidade de combinar serviços como navegação *web*, e-mail, vídeo-conferência e mensagens instantâneas.

### 3.3.2 Transações

O protocolo SIP é trabalha com requisições e respostas para realizar suas transações. As respostas possuem um *status code* para informar o resultado da requisição pela qual está respondendo. Tal código varia de 100 a 699, onde cada centena representa um tipo de resposta. As respostas de 100 a 199 são somente informativas e não há obrigatoriedade que ocorram numa transação.

- 1XX - Mensagem informativa
- 2XX - Mensagem de sucesso
- 3XX - Mensagem de redirecionamento
- 4XX - Mensagem de erro na requisição
- 5XX - Mensagem de erro do servidor
- 6XX - Mensagem de erro global

Junto com o *status code*, o protocolo carrega uma *reason phrase*, que consiste de uma frase informativa referente ao código da mensagem, por exemplo, para uma resposta 180 segue uma *reason phrase* que diz *'Ringing'*, para informar o usuário que no seu destino a aplicação está "chamando"pelo usuário.

As principais requisições SIP são de 6 tipos e acompanham um campo *method* que denota a funcionalidade de cada uma.

- INVITE - usado para convidar um usuário para uma sessão
- ACK - usado para confirmar o recebimento de uma resposta final
- OPTIONS - usado para perguntar ao servidor a respeito de suas funcionalidades
- BYE - usado para sair de uma sessão
- CANCEL - usado para cancelar transações pendentes
- REGISTER - usado para que o usuário informe sua localização atual para um servidor de registro.

Existem mais métodos de requisição no protocolo SIP mas integram a parte extendida do protocolo, não sendo importante para o funcionamento básico do protocolo.

Na figura 3.6 há um exemplo de uma transação SIP simples entre duas máquinas. Primeiro um usuário convida o outro enviando-o um INVITE. De forma automática no destino, a aplicação responde com uma resposta de código 180 *Ringing* para informar que está chamando pelo usuário. Este por sua vez ao aceitar a chamada responde com uma resposta 200 que caracteriza um OK de confirmação. Por fim o usuário que fez o convite confirma que recebeu a resposta enviando uma requisição ACK, e então ambos os usuários iniciam uma sessão de troca de mídia, como por exemplo uma sessão RTP.

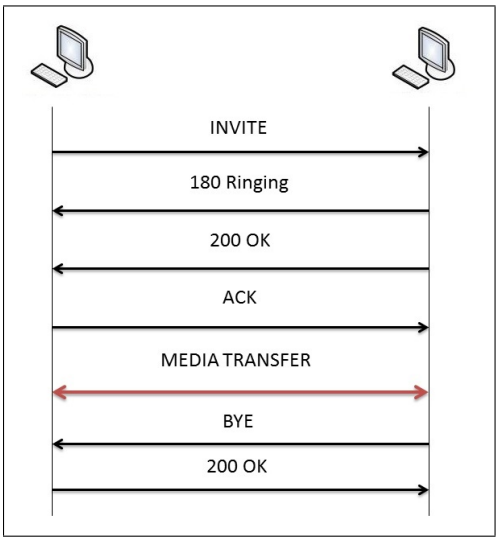


**Figura 3.6**: Fluxo de chamada simples

Ao final da sessão, quando algum dos participantes quiser abandonar a sessão, ele envia uma requisição BYE e o outro deve responder com uma resposta OK de confirmação finalizando assim a sessão.

Na figura 3.7 o usuário que enviou o INVITE agora decide não mais continuar a ligação e envia um CANCEL para o destino antes que esse atenda a chamada, mas de qualquer maneira ele precisa responder a requisição de cancelamento com um OK e deve então enviar uma resposta 487 *Transaction cancelled* a ser confirmado por um ACK. Esse é o fluxo padrão que deve ocorrer quando uma chamada é cancelada.

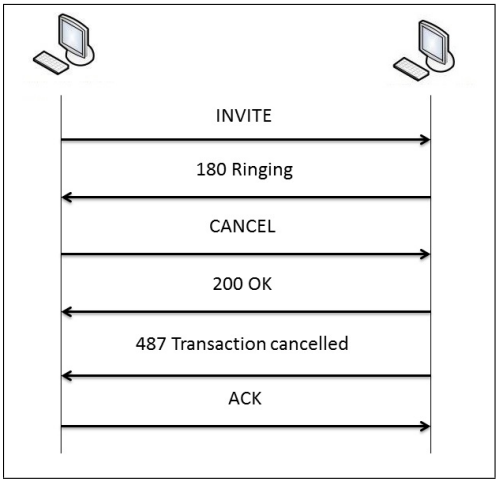


**Figura 3.7**: Fluxo de chamada cancelada

### 3.3.3 Sintaxe e Estrutura

A sintaxe do SIP é baseada em texto (codificação ISO 10646) e suas mensagens são requisições ou respostas. Cada mensagem possui a seguinte estrutura:

- *Request Line* ou *Status Line*
- *Headers*
- Linha em Branco
- *Message Body*

A primeira linha da mensagem define se a mensagem é uma requisição se vinher com uma *Request Line* ou uma resposta se vier com uma *Status Line*. A *Request Line* possui três elementos organizados da seguinte maneira:

*method SP Request-URI SP SIP-Version CRLF*

O *method* é o método definido pelo protocolo (INVITE, BYE e etc). SP é um espaço. Request-URI é o endereço SIP da entidade para onde está sendo direcionada a requisição. SIP-Version é a versão do protocolo. CRLF é um pulo de linha. Exemplo:

*INVITE sip:fulano@192.168.1.23 SIP/2.0*

O endereços no SIP são chamados de SIP URLs e identificam cada usuário possuindo a seguinte estrutura: *user@host*, semelhante a um endereço de e-mail mas que refere-se a um endereço SIP.

A *Status Line* é diferente da requisição, sendo organizada da seguinte maneira:

*SIP-Version SP status code SP reason-phrase CRLF*

Além de uma diferente ordem, o que muda em relação a *Request Line* é o fato de apresentar o *status code* da resposta junto com a *reason-phrase*. Exemplo:

*SIP/2.0 180 Ringing*

Os *Headers* carregam informações referentes à requisição
resposta e ao conteúdo da mensagem. Alguns são utilizados somente para requisições e outros para as respostas, e alguns para ambos. A sintaxe é simples, consiste somente do nome do *header*, dois pontos, o valor e ao final um pulo de linha. Exemplo:

*From: Fulano <sip:fulano@192.168.1.23>*

Na tabela 3.1 estão apresentados os principais *Headers* que são obrigatórios, acompanhados de sua funcionalidade.

| **Header** | **Função** |
|---|---|
| Call-ID | Identifica uma chamada, sendo criado a partir do primeiro INVITE e mantido durante todo o resto da transação |
| Contact | Fornece uma URL onde o usuário possa ser chamado |
| Content-Length | Informa o tamanho do *Message Body* |
| Content-Type | Informa o tipo de mídia contida no *Message Body* (obrigatório somente caso exista o *Message Body*) |
| CSeq | Serve para ordenar transações em uma sessão, para identificar as transações e diferenciar entre novas requisições e retransmissões |
| From | Indica o iniciador da sessão que enviou a primeira requisição |
| Max-Forwards | Serve para limitar o número de *proxies* ou *gateways* que podem encaminhar uma requisição para outro servidor |
| To | Indica o destino da requisição |
| Via | Indicar o caminho pelo qual passou a requisição e indica o caminho que deve ser seguido para rotear respostas |

**Tabela 3.1**: *Headers* obrigatórios do SIP

### 3.3.4 SDP (Session Description Protocol)

O protocolo SDP (RFC 4566) é utilizado em conjunto com o SIP atuando como seu *Message Body*, descrevendo as informações necessárias relativas á sessão de troca de mídia, bem como formatos, endereços e portas. Dentro de sua estrutura são carregados os parâmetros relativos à sessão e N blocos de parâmetros relativos à mídia a ser transferida onde N é o número de mídias diferentes que serão trocadas em uma mesma sessão.

A sintaxe do SDP se caracteriza pela seguinte maneira:

*campo=valor*

sem espaços, com letras minúsculas e terminados com um pulo de linha. Os campos relativos à sessão vem dispostos primeiro e são seguidos pelos campos relativos à mídia iniciando-se pelo campo (m=), assim é possível reconhecer a fronteira entre os dois tipos de campos.

Assim como qualquer outro protocolo, o SDP possui campos obrigatórios e campos opcionais. Na tabela 3.2 seguem os campos obrigatórios na ordem em que deve ser inseridos no protocolo.

| Sintaxe | Função | Obrigatório |
|---|---|---|
| v= | Versão do protocolo e também marca o início da descrição da sessão | X |
| o= | Origem da sessão | X |
| s= | Nome da sessão | X |
| i= | Texto informativo relacionado à sessão | |
| u= | Um endereço web para obter maiores informações sobre a sessão | |
| e= | Email do responsável pela sessão. Usado somente como campo de sessão | |
| p= | Telefone do responsável | |
| c= | Dados relativos à conexão | |
| b= | Especifíca a largura de banda necessária em kbps | |
| z= | Utilizada para especificar a timezone referencial para uma sessão | |
| k= | Campo para prover uma chave criptográfica que pode ser usada para criptografar a mídia | |
| a= | Campo para adicionar mais atributos relativos à mídia | |
| t= | Tempo de iniício e fim da sessão | X |
| r= | Especifíca o número e a frequência de vezes que uma sessão deve ser repetida | |
| m= | Indica o tipo de míidia, a porta de transporte, o protocolo de transporte e também marca o início da descrição da mídia | X |
| i= | Igual ao outro i mas agora relativo à mídia | |
| c= | Dados relativos à conexão. (igual ao anterior) | |
| b= | Especifíca a largura de banda necessária em kbps. (igual ao anterior) | |
| k= | Campo para prover uma chave criptográfica que pode ser usada para criptografar a mídia. (igual a anterior) | |
| a= | Campo para adicionar mais atributos relativos à mídia | |

**Tabela 3.2**: Campos do SDP

Alguns desses campos possuem subcampos, que devem ser dispostos separados por um

espaço simples dentro da declaração de cada campo. O primeiro deles é o *origin* (o), que apresenta seis subcampos:

- username - A identidade da origem em sua máquina host (Exemplo: joao)
- session ID - Identificação única criada pelo originador da chamada. Recomenda-se utilizar o NTP timestamp para não replicar tal número.
- version - Versão da sessão.
- network type - String que indica o tipo da rede. (Ex: IN = Internet)
- address type - Define o tipo de endereçamento. (Ex: IPv4)
- address - O endereço definido pelo tipo acima (Ex: 192.168.1.23)

O campo Media Information (m) também possui 4 sub-campos:

- media type - Tipo de mídia a ser transferida. (áudio, vídeo, dados)
- port - A porta usada para a troca da informação.
- transport - O protocolo de transporte utilizado.
- formats - Indica os códigos das mídias suportados pela aplicação corrente. Podem ser colocados mais de um separados por espaço, respeitando a ordem de prioridade em que são pré-dispostos.

Alguns outros campos também apresentam sub-campos, mas esses são os principais.

# Capítulo 4

# Desenvolvimento da aplicação

Neste capítulo serão apresentados os diagramas referentes a aplicaçzão desenvolvida, e também as telas da interface criadas para a interação com o usuário.

A aplicação foi desenvolvida utilizando-se a linguagem de programação Java por ser multiplataforma, sendo assim possível utilizá-la em sistemas operacionais diferentes, como por exemplo *Windows* e *Linux*.

## 4.1 Casos de Uso

A figura 4.1 apresenta os casos de uso que representam as funcionalidades da aplicação e estão descritos a seguir:

- **Fazer Login** - O usuário preenche seu nome e sua senha e clica no botão entrar.
- **Chamar** - O usuário preenche o nome sip e o endereço IP do destino e pressiona o botão Ligar.
- **Encerrar/Rejeitar Chamada** - O usuário caso deseje encerrar ou rejeitar uma chamada ativa, pressiona o botão desligar.
- **Atender Chamada** - Ao receber uma chamada o usuário é notificado e deve apertar no botão Atender para aceitar a chamada.

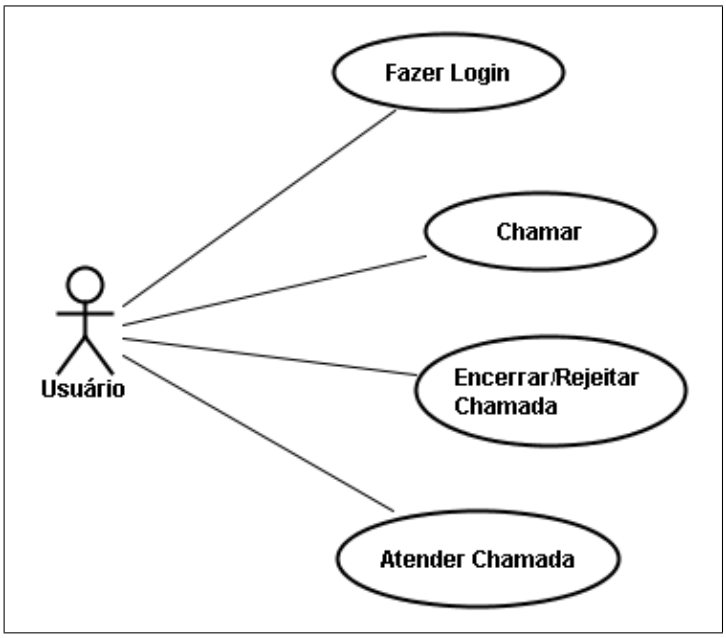


**Figura 4.1**: Diagrama de casos de uso

## 4.2 Interface

Na figura 4.2 que mostra a tela inicial , o usuário precisa fazer login com seu nome, que será utilizado para compor seu nome sip na forma *nome@IP*, e sua senha. A partir dai, o usuário já está conectado e pronto para receber e realizar chamadas.

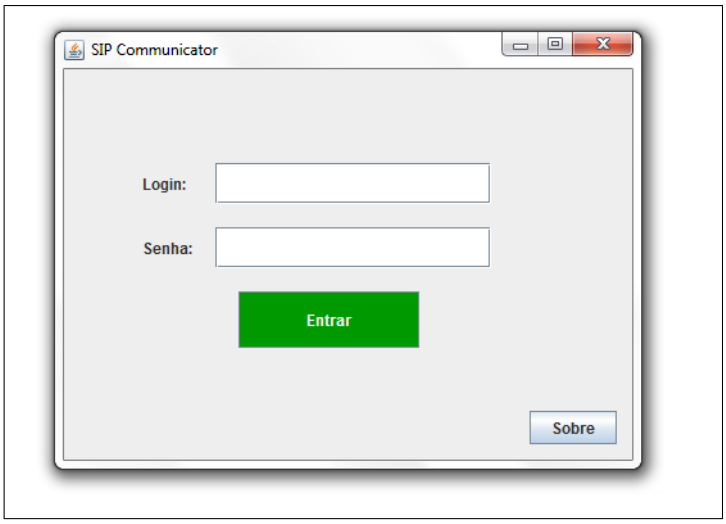


**Figura 4.2**: Tela Inicial

Na segunda tela, mostrada na figura 4.3, para fazer uma chamada é preciso digitar o

nome do destino e seu IP e então apertar em ”Chamar”.

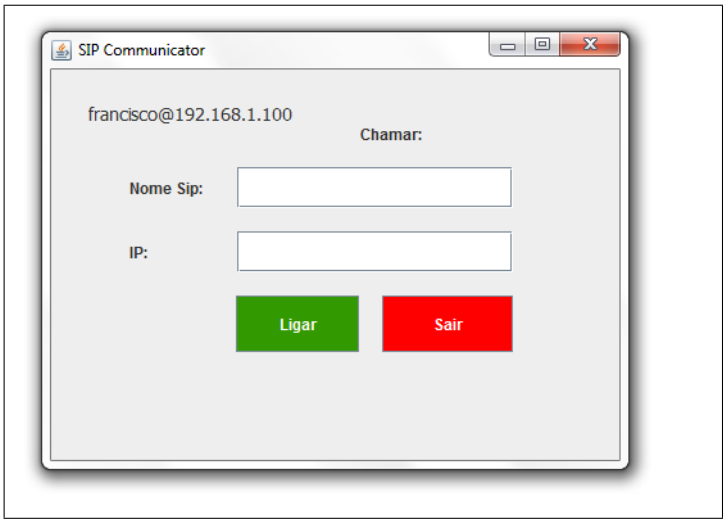


**Figura 4.3**: Tela 2

A partir do momento que se inicia uma sinalização, o usuário é levado para a tela mostrada na figura 4.4, que indica o destino com o qual está se conectando e sempre fornece um *feedback* a respeito do status da chamada. Caso o usuário deseje que a chamada seja encerrada, somente necessita apertar em ”Encerrar”.


**Figura 4.4**: Tela 3

Caso o usuário não esteja em uma chamada e esteja conectado, ele pode receber uma chamada como representado na figura 4.5. Caso isso aconteça, uma janela se abre

mostrando quem o está chamando e lhe permite decidir entre aceitar ou recusar a chamada.

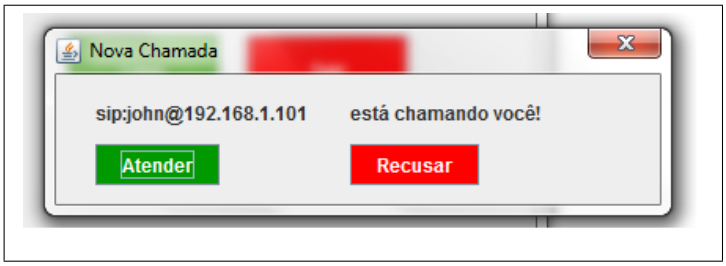


**Figura 4.5**: Janela ao receber uma chamada

## 4.3 Modelo de classes

A figura 4.6 apresenta o modelo de classes da aplicação e a tabela 4.1 descreve a atuação de cada uma no sistema de acordo com os objetivos específicos determinados para o trabalho.

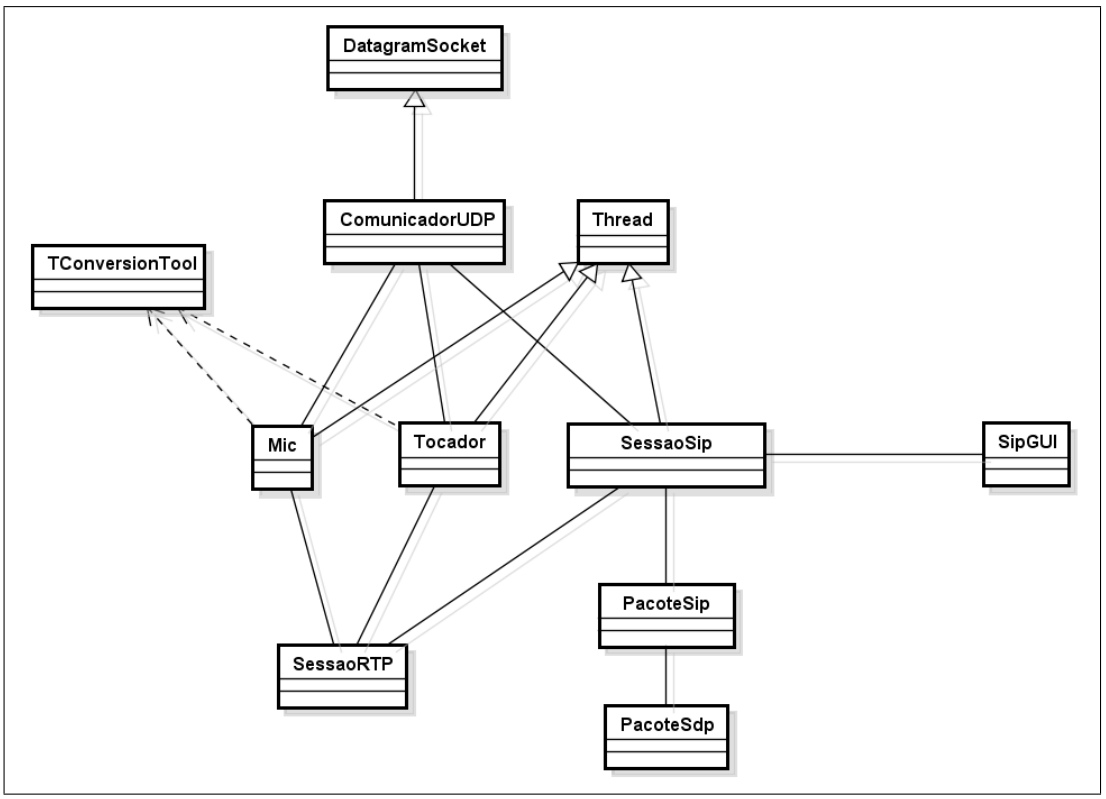


**Figura 4.6**: Diagrama de classes

| Classe | Funcionalidade |
|---|---|
| SipGUI | Renderiza todas as telas. |
| SessaoSIP | Inicializa e gerencia uma sessão SIP. É iniciada no momento em que o usuário faz login. |
| PacoteSIP | Manipula os dados do pacote SIP. |
| PacoteSDP | Manipula os dados do pacote SDP. |
| SessaoRTP | Inicializa e gerencia uma sessão RTP. É iniciada no momento em que inicia-se a troca de mídia. |
| Mic | Capta os dados de áudio do microfone durante uma sessão RTP. |
| Tocador | Sonoriza os dados de áudio recebidos na sessão RTP na saída de áudio do computador. |
| ComunicadorUDP | Engloba os dados em um pacote UDP e envia. |
| TConversionTool | Comprime e descomprime os dados de áudio para o formato G.711. |
| Thread | Classe nativa do Java que inicializa Threads. |
| DatagramSocket | Classe nativa do Java para criar e enviar pacotes UDP. |

**Tabela 4.1**: Funcionalidades das classes

## 4.4 Diagramas de Sequência

Nesta seção são apresentados alguns diagramas de sequência que representam a troca de mensagens entre as classes internamente.

### 4.4.1 Envio de mensagem SIP

A figura 4.7 exemplifica o envio de um INVITE quando o usuário realiza uma chamada. Após o usuário chamar outro, tal evento gera uma solicitação *enviarINVITE()* para a classe *SessaoSIP*, esta por sua vez solicita *criaRequest()* da classe *PacoteSIP* que retorna o pacote estrutado e pronto para envio que então é enviado para a classe *ComunicadorUDP* que o envia pela rede utilizando o protocolo UDP.

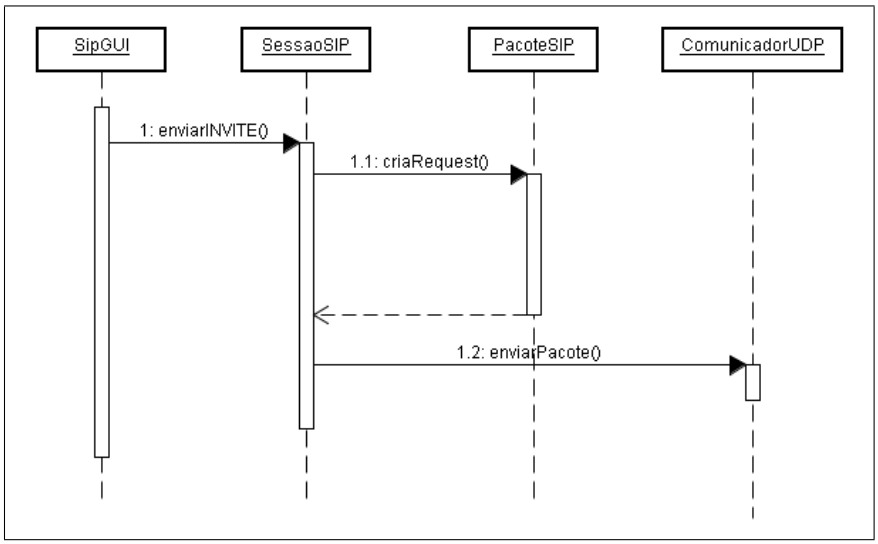


**Figura 4.7**: Diagrama de sequência para envio de um INVITE

### 4.4.2 Recebimento de mensagem SIP

A figura 4.8 exemplifica o comportamento da aplicação ao receber uma mensagem SIP. A classe *SessaoSIP* fica a espera de receber um pacote da classe *ComunicadorUDP* através do método *receberPacote()*. Assim que o pacote chega, o pacote é enviado para a classe *PacoteSIP* para que seja analisado, e então a classe *SipGUI* responsável pela interface fornece um *feedback* para o usuário.

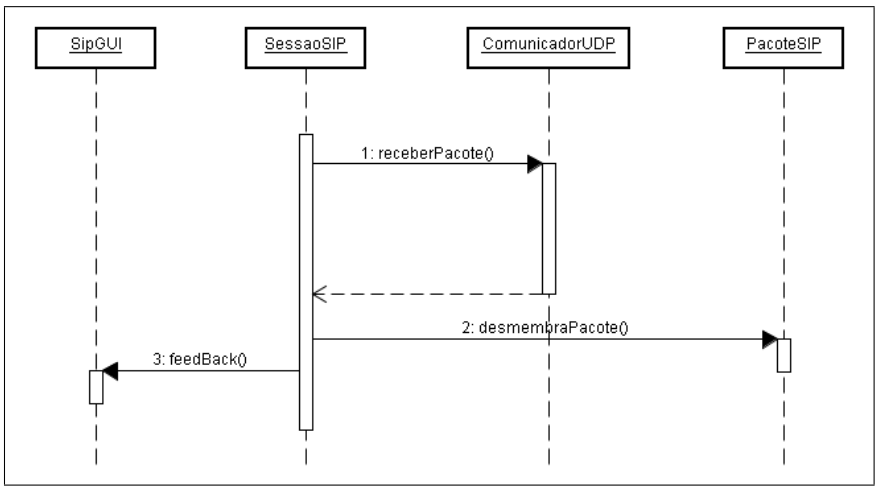


**Figura 4.8**: Diagrama de sequência para recebimento de mensagem SIP

## 4.5 Testes

Foram realizados alguns experimentos com a aplicação para comprovar sua eficácia em funcionamento. Os experimentos foram avaliados levando em consideração somente a Qualidade de Experiência (QoE) do usuário, uma medida subjetiva referente a experiência do usuário em relação ao serviço oferecido. Em termos práticos,o usuário faz uma avaliação do funcionamento da aplicação dentro de uma escala simples que varia de péssimo a ótimo.

### 4.5.1 Experimento 1

O primeiro experimento consistia em realizar uma chamada entre a aplicação desenvolvida e o Jitsi, uma aplicação de áudio, vídeo e chat que suporta os protocolos SIP, XMPP/Jabber, AIM/ICQ, Windows Live, Yahoo!, Bonjour e diversas outras ferramentas úteis. O Jitsi é um software livre de código aberto e disponínel sob os termos da LGPL (*Lesser General Public License*).

O ambiente do experimento foi uma rede local, entre duas máquinas conforme a figura 4.9.

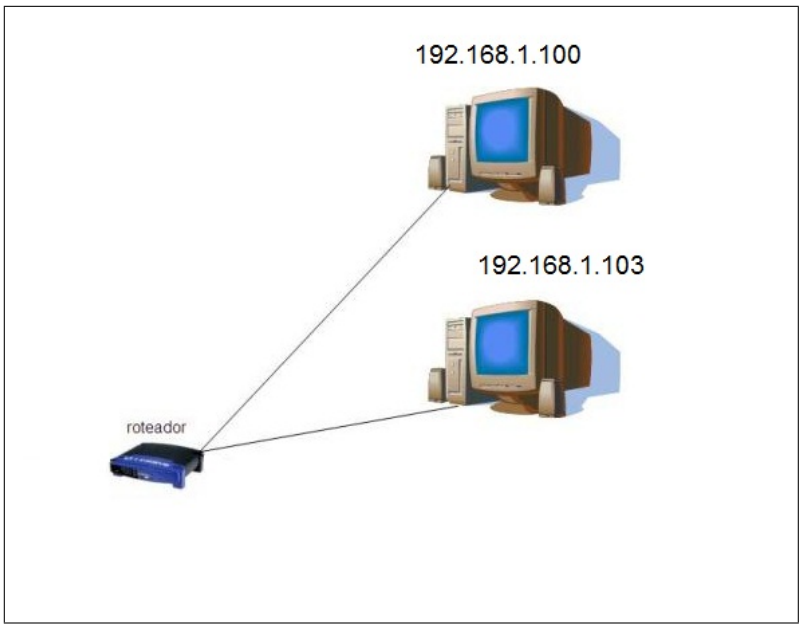


**Figura 4.9**: Ambiente de rede local

### 4.5.2 Experimento 2

O segundo experimento consistia em realizar uma chamada entre duas cópias da aplicação desenvolvida. O ambiente do experimento também foi uma rede local, entre duas máquinas.

Os dados referentes aos experimentos 1 e 2 estão representados na tabela 4.2.

| | **Máquina 1** | **Máquina 2** |
|---|---|---|
| **Processador** | Intel Pentium P6200 | Intel Core 2 Duo |
| **Memória RAM** | 4 GB | 4 GB |
| **Sistema Operacional** | Windows 7 64-bits | Linux Ubuntu 11.04 64-bits |
| **Máscara de Rede** | 255.255.255.0 | |

**Tabela 4.2**: Dados dos Experimentos 1 e 2

### 4.5.3 Experimento 3

O terceiro experimento consistia em realizar uma chamada entre a aplicação desenvolvida e o Jitsi, mas desta vez no ambiente da Internet, para assim verificar a operação da aplicação no ambiente real para a qual foi projetado seu uso, conforme mostrado na figura 4.10.

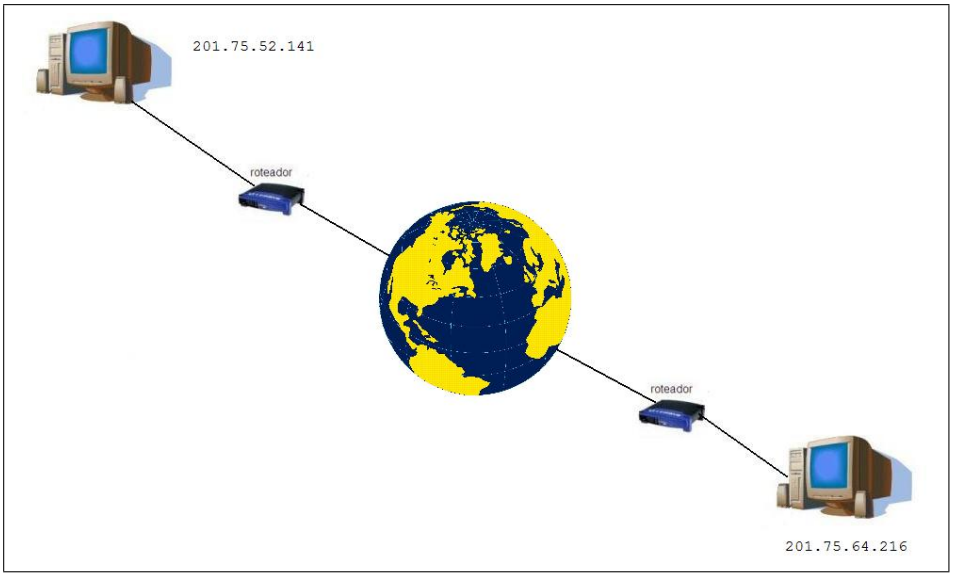


**Figura 4.10**: Ambiente da Internet

### 4.5.4 Experimento 4

O quarto e útimo experimento consistia em realizar uma chamada entre duas cópias da aplicação desenvolvida. O ambiente do experimento também foi a Internet.

Os dados referentes aos experimentos 3 e 4 estão representados na tabela 4.3. A figura 4.11 mostra o *Traceroute* de um pacote pela Internet até o destino dos experimentos 3 e 4. Na figura, a primeira coluna representa o número de roteadores por onde o pacote passou até chegar ao destino. As próximas três colunas mostram o *round-trip delay* em milisegundos de três pacotes diferentes para alcançar o destino. A última coluna mostra cada *host* por onde o pacote passou.

| | **Máquina 1** | **Máquina 2** |
|---|---|---|
| **Processador** | Intel Pentium P6200 | Intel Core 2 Duo |
| **Memória RAM** | 4 GB | 3 GB |
| **Sistema Operacional** | Windows 7 64-bits | Windows 7 64-bits |
| **Largura de Banda** | 5 Mbps | 10 Mbps |
| **Máscara de Rede** | 255.255.0.0 | |
| **Delay (média)** | 71ms | |

**Tabela 4.3**: Dados dos Experimentos 3 e 4

```
C:\Users\JG>tracert 201.75.111.3

Tracing route to c94b6f03.virtua.com.br [201.75.111.3]
over a maximum of 30 hops:

  1    24 ms   101 ms     1 ms  192.168.1.1
  2    12 ms    13 ms     8 ms  bacf9801.virtua.com.br [186.207.152.1]
  3    10 ms    11 ms    12 ms  c8bd581a.virtua.com.br [200.189.88.26]
  4    19 ms    17 ms    18 ms  c94b6f03.virtua.com.br [201.75.111.3]

Trace complete.
```

**Figura 4.11**: *Traceroute* do pacote pela Internet

### 4.5.5 Resultados

Em cada experimento foram realizadas 5 chamadas, e em 5 destas foi possível que os usuários realizassem a chamada conseguindo ter uma conversa. As pessoas que avaliaram a qualidade das chamadas reclamaram de um pequeno barulho que surge no início de cada

chamada mas que desaparece após um tempo. Quanto à qualidade da chamada em geral devido a algumas perdas de pacotes, as chamadas foram classificadas com qualidade Boa.

No terceiro e quarto experimento, devido ao fato das máquinas estarem em redes diferentes, recebem IP's diferentes, portanto foi necessário fazer uma modificação no código-fonte da aplicação informando manualmente o endereço IP externo para que a aplicação Jitsi pudesse direcionar de forma correta seus pacotes. E também foi necessário que os roteadores das extremidades fossem configurados para permitir que pacotes externos fossem direcionados às máquinas do experimento.

Nos experimentos realizados na Internet a qualidade das chamadas foi avaliada pelos participantes como ótima, sendo este um fato curioso já que espera-se que em rede local a qualidade seja sempre superior à Internet devido a curta distância e menor número de roteadores e *switch* entre as máquinas. A rede em que foram realizados os experimentos 1 e 2 apresenta muita instabilidade devido aos *proxys* e ao *firewall* que produzem *overhead* na rede congestionando o tráfego dos pacotes.

# Capítulo 5

# Conclusão

A aplicação desenvolvida neste trabalho apresenta o uso do protocolo SIP para realizar chamadas dentro de uma rede IP, tanto em redes locais como na Internet, demonstrando assim que é possível se utilizar deste para criar diversas aplicações livres, ofereçendo uma nova alternativa para comunicação frente à telefonia tradicional e aos sistemas proprietários.

Os resultados dos testes comprovam a eficácia da aplicação, não somente devido ao sucesso alcançado ao realizar chamadas entre cópias da mesma mas também pela comunicação bem sucedida com a aplicação Jitsi, que atuou como *benchmark* para alguns experimentos.

O desenvolvimento da aplicação demonstra como é possível e muitas vezes viável para algumas organizações ou empresas que desejam implantar uma rede de telefonia local, aproveitar a infra-estrutura já existente da Rede IP, que frequentemente já são encontradas nas empresas para acesso a Internet, e assim economizar muito em custos de implantação, taxas de operadoras e licenças de *softwares* proprietários.

## 5.1 Trabalhos Futuros

Algumas funcionalidades ficam como sugestões para trabalhos futuros:

- Integração com banco de dados para armazenar lista de contatos, histórico de

chamadas e usuários cadastrados (login e senha).

- Melhores recursos visuais.
- Controle de áudio (volume e configuração).
- Mais codificações suportadas.
- Implementação do RTCP.
- Capacitar comunicação com servidores SIP.

# Referências Bibliográficas


[rfc2002] (2002). *RFC 3261*. The Internet Society. http://www.ietf.org/rfc/rfc3261.txt.

[java2003] (2003). *Java Sound Programmer Guide*. Oracle.

[rfc2003] (2003). *RFC 3550*. The Internet Society. http://www.ietf.org/rfc/rfc3550.txt.

[rfc2006] (2006). *RFC 4566*. The Internet Society. http://tools.ietf.org/html/rfc4566.

[Camarillo2001] Camarillo, G. (2001). *SIP Demystified*. McGraw-Hill, first edition.

[Collins2000] Collins, D. (2000). *Carrier Grade Voice Over IP*. McGraw-Hill, first edition.

[da Silva Bezerra2005] da Silva Bezerra, R. M. (2005). Um estudo do protocolo sip e sua utilização em redes de telefonia móvel.

[da Silva Junior2003] da Silva Junior, J. M. (2003). Uma aplicação de voz sobre ip baseada no session initiation protocol. Master's thesis, Universidade Federal de Pernambuco.

[Gonçalves e Hommerding2006] Gonçalves, A. M. e Hommerding, R. (2006). Implementação didática de telefone voip por software utilizando protocolo sip.

[Marcondes e de Aguiar Rodrigues2002] Marcondes, C. A. C. e de Aguiar Rodrigues, P. H. (2002). Serviço robusto de web-to-dial baseado em sip e java applet.

[Nazario2003] Nazario, D. L. (2003). Protótipo de um sistema de telefonia ip para lans baseado no padrão sip.

[Singh and Schulzrinne2005] Singh, K. and Schulzrinne, H. (2005). Peer-to-peer internet telephony using sip.

# Capítulo 6

# Apêndice

## 6.1 Código-Fonte

Os códigos-fontes da aplicação estão disponíveis no CD que acompanha a monografia.