ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO

ALEXANDRE DERVAZI CARVALHO

DIOGO SATORU AIHARA

THIAGO DANIEL JORGE FERNANDES

FRAMEWORK VOIP

SÃO PAULO

2005

ALEXANDRE D. CARVALHO
DIOGO S. AIHARA
THIAGO D. J. FERNANDES

FRAMEWORK VOIP

PROJETO DE FORMATURA
EPUSP
2005

Alexandre D. Carvalho
Diogo S. Aihara
Thiago D. J. Fernandes

Framework VOIP

Relatório Final do Projeto de Formatura apresentado como requisito essencial para a obtenção de título de Engenheiro Eletricista pela Escola Politécnica da Universidade de São Paulo.

Orientador: Prof. Dr. Reginaldo Arakaki

São Paulo
2005

Palavras-chave: Voz Sobre IP (VOIP), User Datagrame Protocol (UDP), Transfer Control Protocol (TCP), Internet Protocol (IP), Session Initiation Protocol (SIP), Session Description Protocol (SDP), Real-time Transport Protocol (RTP), Real-time Transport Control Protocol (RTCP), Quality of Service (QoS), Framework, workflow

FOLHA DE APROVAÇÃO

Alexandre D. Carvalho

Diogo S. Aihara

Thiago D. J. Fernandes

Relatório Final do Projeto de Formatura apresentado como requisito essencial para a obtenção de título de Engenheiro Eletricista pela Escola Politécnica da Universidade de São Paulo.

Aprovado em:

Banca Examinadora

Prof. Dr. _____

Instituição: _____ Assinatura: _____

Prof. Dr. _____

Instituição: _____ Assinatura: _____

Prof. Dr. _____

Instituição: _____ Assinatura: _____

Prof. Dr. _____

Instituição: _____ Assinatura: _____

Prof. Dr. _____

Instituição: _____ Assinatura: _____

Prof. Dr. _____

Instituição: _____ Assinatura: _____

Dedicatória

Dedicamos este trabalho aos nossos pais pelo apoio que nos foi dado durante o percurso de nossas vidas e à oportunidade que nos foi dada.

Dedicamos também, uns aos outros, autores deste trabalho. Pelas noites em claro e força que fazia com que nos encorajássemos mutuamente para finalizá-lo da melhor maneira possível.

À Cristina, companheira de Alexandre que o manteve nos trilhos em momentos de desespero.

Ao aluno André Bertolace, Bertola, que participou ativamente no desenvolvimento deste trabalho e que nos trocou em favor de oportunidades no Velho Mundo no momento de colher os frutos que plantamos.

À Jorge Aihara, que não poderá estar presente no momento de apresentação deste projeto, mas que estará acompanhando em uma posição privilegiada.

À mulheres como Angelina Jolie, que nos mostraram porque devemos desenvolver projetos audaciosos e buscar a riqueza material.

E, principalmente, ao Monty Python, que nos ensinou a sempre olhar para o lado bom da vida.

RESUMO

Este trabalho tem por objetivo a apresentação e documentação do Projeto de Formatura, requisito necessário para a graduação na área de Engenharia Elétrica pela Escola Politécnica da Universidade de São Paulo (POLI-USP). As informações aqui apresentadas são resultado de um trabalho executado durante o período de um ano, onde no primeiro semestre foi definido o escopo do projeto e, na segunda metade, foram realizados um aprimoramento do projeto e sua execução.

O tema abordado neste projeto é a comunicação de voz sobre IP (VOIP), onde o objetivo é desenvolver um conjunto de bibliotecas funcionais baseadas neste serviço, de modo a prover uma plataforma de desenvolvimento que possibilite a criação de aplicações com um maior nível de abstração e menor alocação de recursos.

Para isso foram realizados estudos sobre o mercado do produto, sua expansão e panorama no cenário brasileiro, realizando posteriormente a análise das tecnologias usualmente utilizadas e consumação de uma modelagem do processo. Como resultado obtivemos um produto voltado a atender às expectativas do mercado, prioritariamente o corporativo.

ABSTRACT

This report aims to present and document our Graduation Project, a primary requirement to graduate on Electric Engineering at Escola Politécnica da Universidade de São Paulo (POLI-USP). All the information included in this document results from a work developed during the period of one year. The first half was expended on scope definition and, on the second half, the project was refined and executed.

The subject covered in this project was voice over IP communication (VoIP), where the goal was to develop a functional library package based on this kind of service, in a way to provide a developing platform that enables creating applications with a higher level of abstraction and lower resources allocation.

For that end, studies have been made on market demands, its expansion and national scenario overview, to later accomplish an analysis on common used technologies and achieve a process modeling, reaching a product that meets market expectations, specially the corporate one.

Lista de Ilustrações

Lista de Tabelas

Lista de Siglas

TCP/IP: *Transfer Control Protocol / Internet Protocol*: conjunto de protocolos que regem a comunicação atual na internet.

UDP: *User Datagram Protocol*: protocolo responsável por envio não confiável de informações por uma rede IP. Geralmente utilizado para envio de dados que não tenham a necessidade de ter a entrega garantida no destino. Por este mesmo motivo, o envio de dados via UDP tende a ser mais rápido, por evitar confirmações de envio e medidas para retransmissão de dados. É utilizado para envio de dados de voz e vídeo, que necessitam de velocidade e não necessitam que 100% dos dados enviados cheguem ao destino (realizam interpolações para pacotes que não chegam).

SIP: *Session Initiation Protocol*: protocolo utilizado na sinalização de conversas de voz e vídeo sobre IP

SDP: *Session Description Protocol*: protocolo utilizado em conjunto com o SIP, para fornecer informações a respeito das compactações envolvidas nas conversas de voz e vídeo sobre IP

RTP: *Real-time Transport Protocol*: protocolo utilizado para envio de pacotes de voz e vídeo com o intuito de transformar o tráfego não-determinístico das redes IP (geralmente, tráfegos feitos via UDP para maior velocidade), como por exemplo, a Internet em uma comunicação em tempo real, reorganizando os pacotes que chegam fora de ordem e realizando interpolações para suprir a falta dos que por algum motivo não chegam ao destino.

RTCP: *Real-time Transport Control Protocol*: utilizado para auxiliar na Qualidade de Serviço da comunicação dos dados, fornecendo dados estatísticos a respeito das transmissões efetuadas pelo RTP.

QoS (*Quality of Service)*: Qualidade de serviço – conceito relacionado à uma busca por serviços entregues de forma satisfatória (com boa qualidade) para o usuário.

WiMAX: Worldwide Interoperability for Microwave Access. Define novo paradigma de conexão de banda larga, com o objetivo de fornecer a baixo custo e grandes distâncias.

G.723: Algoritmo utilizado para compressão de voz e supressão de silêncio de um sinal digital cuja variante mais conhecida é o G-723.1, que consegue converter um sinal PCM de entrada de 64 kbps em uma saída que pode variar de 5,3 a 6,4 kbps.

Codec: Significa Codificador/Decodificador (um termo de telecomunicações) ou Compressor/Descompressor (um termo de computação). Um codec é uma parte do hardware ou software que comprime e descomprime vídeo e/ou áudio digital.

GSM: (Global System for Mobile Communications) GSM é um standard internacional de comunicações digitais celulares. A família de sistemas GSM inclui o GSM 1800 e GSM 900.

Há diferentes fases de desenvolvimento do sistema GSM. Os telefones GSM podem ser neste momento conformes com a fase 1 ou fase 2. GSM eram originalmente as iniciais de "Groupe Speciale Mobile" tendo sido posteriormente alterado para "Global System for Mobile Communications".

Gateway: Servidor que realiza a interface entre dois protocolos de comunicação diferentes, fazendo a transcodificação da sinalização e informação.

http: (Hypertext Transfer Protocol) Um protocolo usado na Internet por browsers Web para transportar texto e gráficos.

API: Application Program Interface – conjunto de métodos padronizados que a aplicação utiliza para efetuar seus serviços.

JAIN: Java API for Integrated Networks – conjunto de APIs feitas em Java que estão voltadas para aplicações de telecomunicações.

NIST: National Institute of Standards and Technology – um instituto do governo norte-americano que trabalha em pareceria com empresas para fornecer tecnologias e padrões.

Jitter: Em voz sobre IP, refere-se ao efeito de perda de ordenação de pacotes.

P2P: Modo de comunicação que envolve diretamente dois pontos de uma rede.

SUMÁRIO

1. Introdução

1.1. Objetivo

A proposta consiste em uma modelagem de soluções em um ambiente corporativo, tendo como enfoque a elaboração de uma infra-estrutura para desenvolvimento de aplicativos, com o intuito de prover ferramentas de comunicação interna utilizando telefonia em uma rede IP.

Esta infra-estrutura tem por finalidade reduzir a quantidade de recursos envolvidos no desenvolvimento de aplicativos na área de abordagem da mesma, fornecendo um ambiente de programação, com um maior nível de abstração.

A fim de validar a eficácia da modelagem e da infra-estrutura, será escolhido o caso de comunicação ponto a ponto (P2P) que possui as funcionalidades básicas necessárias aos demais casos de uso mencionados neste trabalho.

Do ponto de vista comercial, esta é uma solução interessante, pois o framework possibilitará maior flexibilidade ao usuário ao implementar soluções baseadas em voz sobre IP que resolvam o seu problema em especial.

1.2. Motivação

1.2.1. Histórico

Quando ao final dos anos 90 surgiram os primeiros papers, as primeiras soluções e reportagens sobre esta então nova tecnologia, não se imaginava o impacto que estava por vir.

O VoIP vem quebrando previsões e paradigmas a cada dia e novos produtos são produzidos quase que incessantemente. São infindáveis exemplos, desde softwares em que o usuário utiliza seu próprio computador, passando por adaptadores e gateways, que fazem a interface entre o telefone usual e um ponto de rede, até prestadores de serviço que chaveiam através de seu sistema uma ligação normal de um cliente para a rede VoIP transparentemente.

Talvez o principal atrativo desta tecnologia seja o fato de ser extremamente barata, principalmente no que diz respeito a ligações de longa distância. Provedores de serviços oferecem tarifas em média 80% menores que as concorrentes operadoras de telefones fixos tradicionais. Essa economia deixa evidente o crescente interesse que tanto empresas quanto particulares desenvolveram com relação ao VoIP como forma de comunicação.

Podemos traçar a origem do VoIP como conceito desde o surgimento de aplicativos de *instant messaging* tais como ICQ, Messenger, dentre outros. Foi o inicio da utilização da

Internet como rede de comunicação real time. Logo, estes softwares ofereciam conversas via voz – mesmo que ainda incipiente – dando vazão à nova possibilidade criada e que dá nome a tecnologia – a transmissão de voz sobre IP.

Com o VoIP, o conceito de comunicação se amplia em relação a telefonia convencional. Novos serviços podem ser oferecidos com a integração das tecnologias.

A VocalTec, empresa de origem israelense, foi a primeira a lançar um software com essa tecnologia, longe dos padrões atuais, porém sendo pioneira e acreditando na tecnologia. Algumas outras tentativas se sucederam até o Skype iniciar suas atividades e oferecer uma solução que hoje atinge boa parte dos países do mundo com tarifas muito menores que operadoras de telefonia fixa.

1.2.2. Projeções e tendências de mercado mundial

O mercado do VoIP ainda se encontra em franca expansão, e diversas novas soluções e conceitos estão sendo produzidos constantemente. Com o futuro advento das novas tecnologias de transmissão de redes, das quais para este caso se destaca o WiMAX, que possui em seu protocolo 802.16e uma plataforma favorável (priorização de pacotes e QoS, por exemplo) a distribuição de mídia via rede IP, inclusive dando suporte para novas tecnologias, as possibilidades tendem a crescer. A conectividade à Internet segue inexoravelmente a caminho de se tornar anytime-anywhere, o que significa estar sempre conectado da melhor maneira possível não importando onde se esteja.

O oficial do ministério japonês, Junko Koizumi, anunciou em nota oficial a confirmação dos planos de seu país de implantar uma rede de telefonia móvel baseada em voz sobre IP até o final de 2007, com banda de 15 megabits por segundo, o que significa mil vezes a atual capacidade da rede celular 3G instalada atualmente na ilha (fonte: *The Inquirer*, 13/10/05).

Por outro lado, apesar de toda a projeção positiva, ainda está em pauta a discussão no que diz respeito à substituição da atual rede de telefonia. Ainda muitas questões regulatórias, como por exemplo, a utilização e estabilidade do serviço de emergência 911 americano, geram polêmica e dúvidas nesse sentido. Isso porque no caso de telefonia fixa, ao realizar a chamada o sistema consegue localizar exatamente sua origem. No VoIP, o assinante pode utilizar o serviço de qualquer ponto de acesso, ou seja, não necessariamente existe um ponto geográfico associado a linha telefônica. O FCC (*Federal Communications Commission*) está regulamentando regras para prestadores de serviço de forma a aliviar este problema. (Fonte: *FCC*, 06/03/05)

1.2.3. Panorama no Brasil

No Brasil, o mercado de telefonia VoIP ainda está se desenvolvendo. Hoje ele atinge uma pequena quantidade de pessoas, quantidade que vem crescendo bastante com o surgimento de prestadoras de serviços nacionais, o que atrai o comprador que tem receios de utilizar um serviço estrangeiro, pagando em moeda internacional.

Um exemplo é a HipTelecom, que entrou no mercado no último ano e vende o serviço em três formatos:

- Cartões pré-pagos que podem ser utilizados em qualquer telefone fixo, utilizando um prefixo de operadora para realizar o chaveamento para sua rede.
- Gateway que pode ser instalado na interface de rede e no telefone usual fixo, de forma a utilizar diretamente o aparelho como um telefone VoIP.
- Software que permite utilizar o serviço, fazendo chamadas e utilizando recursos do próprio PC do assinante.

A Brasil Telecom, por exemplo, começa também a oferecer esses mesmos serviços a partir de dezembro de 2005.

Podemos perceber um interesse intenso do mercado brasileiro, que agora começa a se expandir criando novas oportunidades e espaço para novos produtos que se adeqüem não só a realizar suas funções como também adaptar-se e gerar novas soluções para a nossa realidade. Isso já é observável no que diz respeito ao ambiente corporativo, não só no Brasil como no mundo todo. Este ambiente é extremamente rentável e sempre em busca de ferramentas que não só barateiem o custo operacional, como proponham vantagens e oportunidades de integrar e melhorar processos, facilitar comunicação, aumentar produtividade e encontrarem falhas antes que se tornem problemas reais. Nesse contexto se insere este trabalho.

1.3. Organização

Este documento está dividido da seguinte maneira:

- **Introdução:** Contextualização e organização do trabalho
- **Aspectos Conceituais:** Descrição das tecnologias estudadas para a realização do projeto.
- **Especificação do Projeto de Formatura:** Definição de escopo e desenvolvimento teórico do sistema.
- **Metodologia:** Proposta de trabalho, definição de estratégia de ataque ao problema proposto.

- **Projeto e Implementação:** Definição das ferramentas, modelagem e descrição do produto implementado.
- **Testes e Avaliação:** Metodologia e definição de testes.
- **Considerações Finais:** Considerações do grupo a respeito do projeto.
- **Bibliografia:** Referências bibliográficas utilizadas no decorrer do trabalho.

2. Aspectos Conceituais

2.1. Estudo dos protocolos

A comunicação de dados em uma conversa utilizando Voz sobre IP utiliza, além dos protocolos TCP, UDP e IP, os seguintes protocolos:

- Session Initiation Protocol (SIP): responsável pela sinalização da conversa como, por exemplo, decidir se um usuário está disponível ou não, fazer um telefone tocar, encerrar uma chamada, entre outros.
- Session Description Protocol (SDP): responsável por decidir os parâmetros da chamada, como por exemplo: nome e protocolo utilizado para compactação da voz.
- Real-time Transport Protocol (RTP): responsável por garantir um mínimo de controle da transmissão dos pacotes de voz, atuando neste caso como um aditivo ao UDP para o transporte da voz, ordenando os dados em tempo real, evitando um embaralhamento da voz (os pacotes de voz podem chegar em ordem diferente da original). Considerando que uma conversa continua inteligível mesmo com alguns trechos pequenos (aprox. 35ms) faltando, o uso do UDP+RTP torna-se vantajoso com relação ao uso do TCP, que força que todos os pacotes enviados cheguem ao destino e isso geraria um atraso indesejável.
- Real-time Transport Control Protocol (RTCP): serve como um aditivo ao RTP, suprindo a necessidade de monitoramento das chamadas, fornecendo relatórios sobre a qualidade do serviço.

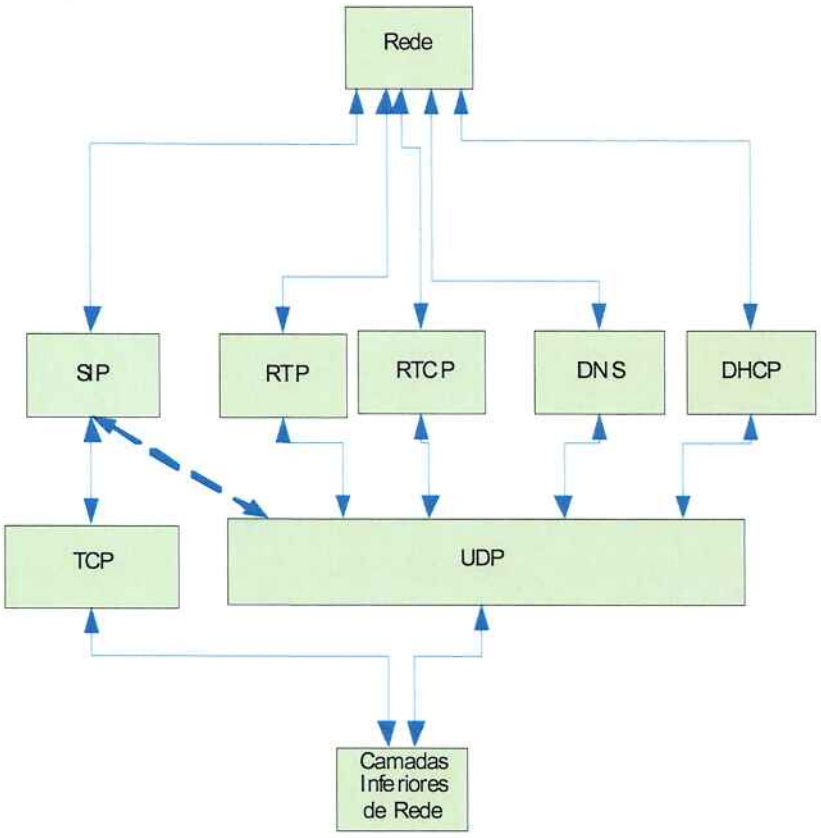

Figura 1 - Protocolos de Rede

2.2. *Tecnologias envolvidas*

2.2.1. Sinalização

O SIP é um protocolo da camada de aplicação cuja finalidade é iniciar, modificar e terminar sessões multimídia tais como conversas telefônicas pela Internet.

Tal qual o HTTP, o SIP é baseado em texto, o que facilita a implementação em programação orientada a objeto, permite um debug mais fácil e um melhor acoplamento com outros protocolos, como o próprio http. Estas características o tornam um protocolo flexível e extensível. Alem disso, foi criado para ser simples e eficiente, o que levou a um protocolo com pouca troca de sinalização, no qual apenas as funções básicas são executadas (criar, modificar e terminar).

Outra característica que demonstra a flexibilidade do SIP consiste no fato de ser independente da camada de transporte, ou seja, ele pode rodar em cima de TCP ou UDP, ficando esta decisão a cargo do usuário ou programador. No entanto, apesar desta possibilidade é comum utilizar o UDP, pois permite maior controle do atraso e retransmissão de pacotes.

Para estabelecer ou modificar uma chamada, o SIP dispõe de cinco processos: localização, disponibilidade e capacidade do usuário e inicialização e controle da sessão. Estes processos são feitos por meio alguns métodos de requerimento, citados e explanados abaixo:

Invite: Convida o usuário para a sessão

ACK: Usado para facilitar a troca de mensagens durante o Invite

Options: Solicita informações sobre a capacidade do servidor e dos usuários

Bye: Encerra a sessão

Register: Registra o usuário em sua localização atual

Info: Usado no meio da sinalização para requisição de informações

E como usualmente ocorre em sistemas com troca de sinalização, existem os códigos de status, que servem para informar a situação do requerimento, são eles:

1xx: Informacional: requisição recebida, processo em andamento.

Exemplo: 100 – Trying, 180 – Ringing.

2xx: Sucesso na execução da requisição.

Exemplo: 200 – OK.

3xx: Redirecionamento, outras ações devem ser tomadas para executar o requerimento.

Exemplo: 302 – Moved Temporarily.

4xx: Erro no cliente.

Exemplo: 404 – Not found.

5xx: Erro no servidor.

Exemplo: 501 – Not implemented.

6xx: Falha global.

Exemplo: 603 – Decline.

Antes de entrar em maiores detalhes quanto ao cabeçalho e o diagrama de sinalização, é importante abordar a arquitetura de rede considerada pelo protocolo. Cada elemento na arquitetura de rede do SIP é dividido em duas partes que são chamadas *User Agent Server* (UAS), que é responsável por receber requisições e enviar respostas e *User Agent Client* (UAC), que é responsável por enviar requisições e receber respostas.

Além disso, na arquitetura, temos as entidades cliente, que é um terminal utilizado por um usuário e servidor, que são entidades que atuam segundo funções específicas. Os servidores dividem-se em três tipos: servidor proxy, servidor de redirecionamento e servidores de registro. Servidores de proxy são servidores na camada de aplicação, responsáveis por receber a requisição e determinar para onde encaminhar, já os servidores de

agente de usuário são responsáveis por determinar o que responder ao usuário. Servidores de redirecionamento mapeiam a localização do usuário, e retorna a informação ao proxy, de modo que este possa encaminhar a requisição ao destinatário, mesmo com ele em outro domínio. E por ultimo, o servidor de registro torna disponível o registro dos usuários na rede, esta atividade é realizada por meio do comando Register. Vale ressaltar que todas estas divisões são lógicas, podendo todos os servidores encontrar-se fisicamente no mesmo lugar.

Para exemplificar a divisão em *User Agents* e sua junção para formar os elementos cliente e servidor, seguem abaixo duas figuras.

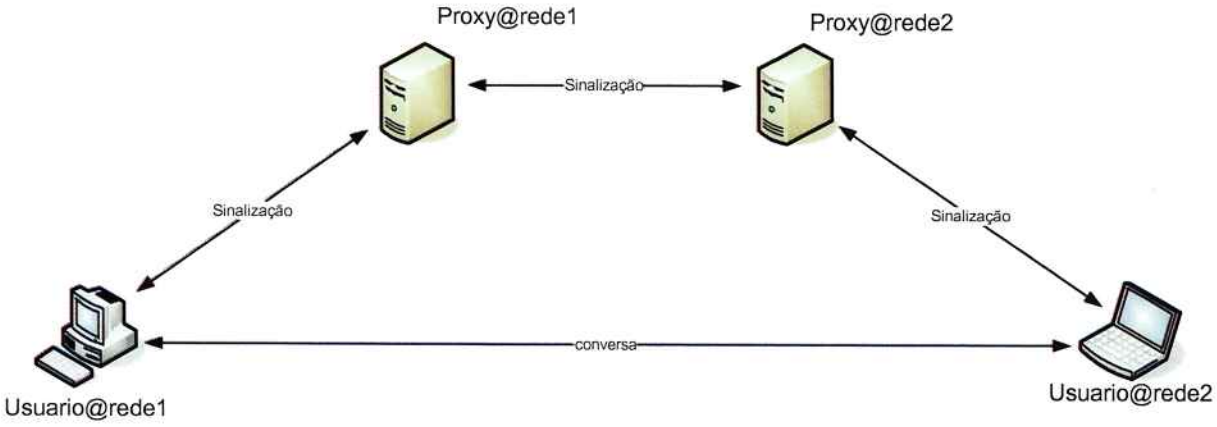

Figura 2 - Elementos da arquitetura de rede do SIP

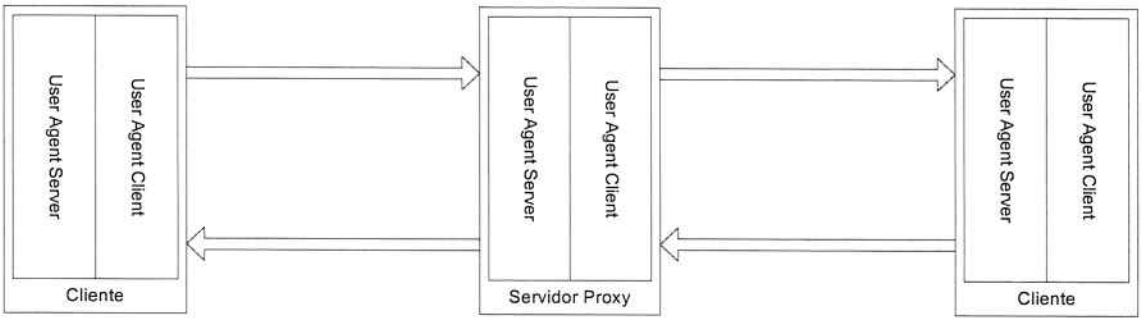

Figura 3 - User Agent Server e User Agent Client

No exemplo abaixo, podemos analisar a troca de sinalização para o estabelecimento da chamada entre dois usuários que estão em domínios diferentes. O processo pode ser descrito como:

User1 envia a requisição *invite* ao *User2* inicialmente pertencente ao mesmo domínio. O cabeçalho da mensagem segue o padrão abaixo:

INVITE sip: user2@rede1.com SIP/2.0

Via: SIP/2.0/UDP 127.0.0.1/5060

From: User1<sip:user1@rede1.com>;tag=740924

To: User2<sip:user2@rede1.com>

Call-ID:123456789abcd@127.0.0.1

Cseq: 1 INVITE
Contact: <sip:User1@127.0.0.1>
Content-Type: application/sdp
Content-Length: 142
(corpo da mensagem)

O provedor proxy responde 100 (Trying) ao *User1* e encaminha o *invite* ao servidor de redirecionamento, o qual responde 302 (User Moved Temporarily), indicando que o *User2* não encontra-se neste domínio e encaminha qual o domínio atual do *User2*. A mensagem é do seguinte tipo:

SIP/2. 302 Moved Temporarily
Via: SIP/2.0/UDP 127.0.0.1/5060
From: User1<sip:user1@rede1.com>;tag=740924
To: User2<sip:user2@rede1.com>
Call-ID:123456789abcd@127.0.0.1
Cseq: 1 INVITE
Contact: <sip:User2@rede2.com>
Content-Length: 0

Ao receber esta mensagem, o servidor retorna o comando ACK ao servidor de redirecionamento e reencaminha o *invite* ao servidor do domínio rede2.com. Este por sua vez encaminha a mensagem ao *User2* e responde 100 (Trying) ao servidor rede1.

Esta e as outras mensagens são encaminhadas ao *User2* e as respostas repassadas ao *User1*, até se estabelecer a chamada e iniciar-se a sessão RTP.

Por último, para encerrar a chamada um dos usuários encaminha um *bye* ao outro usuário e este responde 200 (OK).

As figuras Figura 4 e Figura 5 ilustram a troca de mensagens descritas acima.

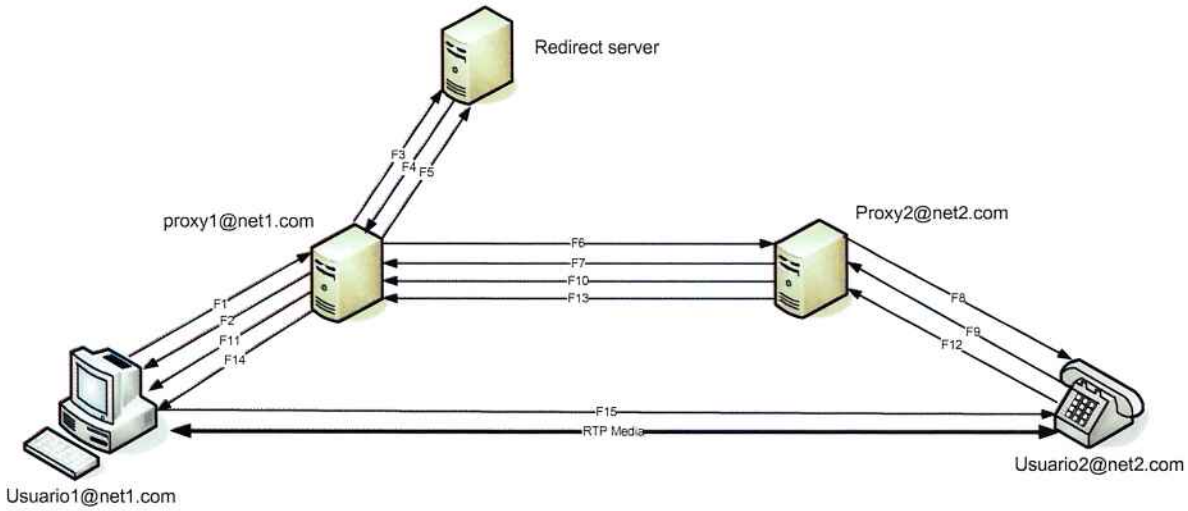

Figura 4 - Fluxo de Sinalização

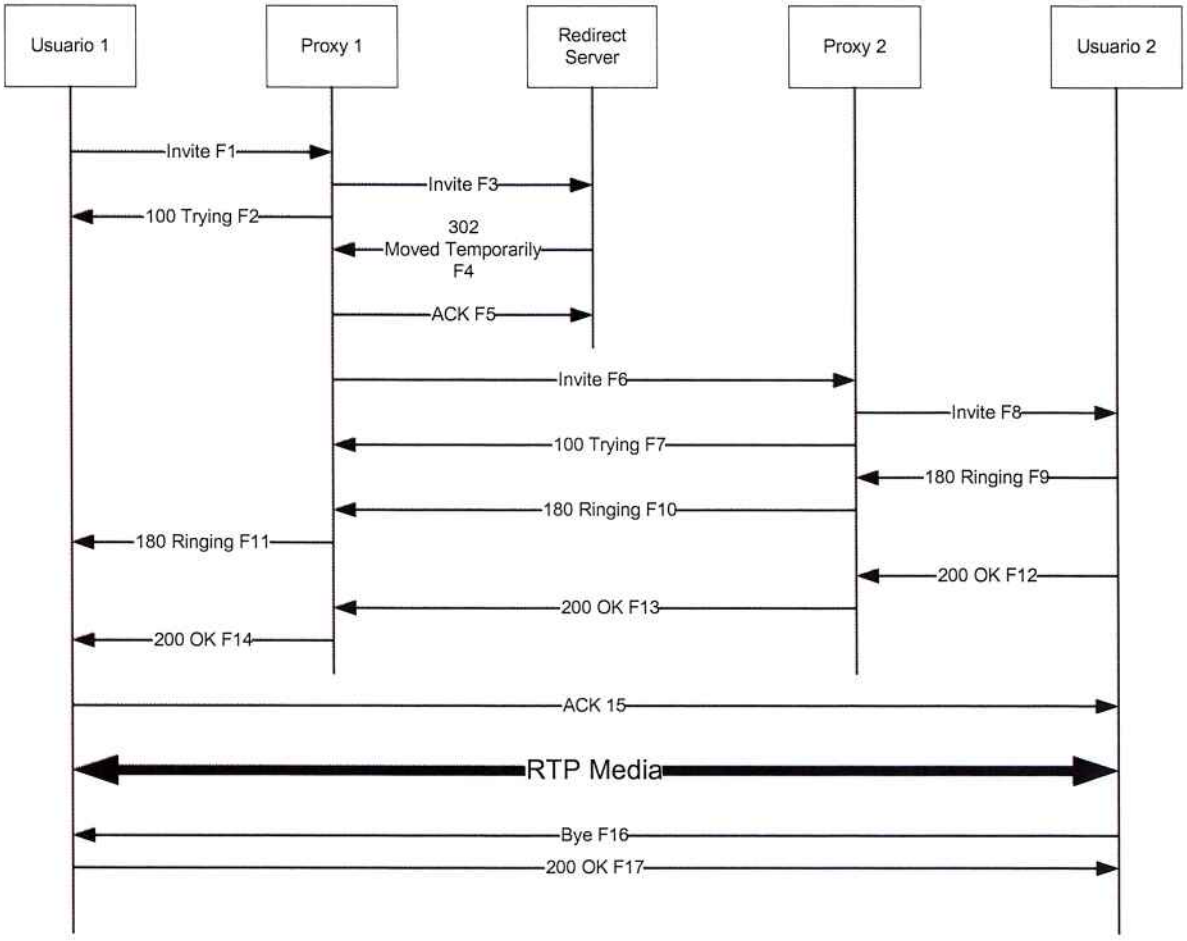

Figura 5 - Diagrama esquemático do Fluxo de Sinalização

2.2.2. Transporte em tempo real (RTP)

O RTP é um protocolo de transporte de dados em tempo real que permite aplicações de áudio e vídeo interativo. Suas principais características são a identificação do tipo de dados

(payload), numeração seqüencial, timestamping e monitoração de entregas. Ele não possui em sua estrutura, nenhum tipo de confirmação de entrega, porém sua numeração permite que sejam recompostas as informações mesmo com perdas de pacote, permitindo a reconstrução de dados que dependem da ordenação temporal.

Em rede IP, o RTP usualmente é transportado pelo UDP, o qual possui mecanismos básicos de checagem de erro e multiplexação de serviços, características suficientes e até mesmo mais apropriadas. O TCP com seu formato que garante a transmissão dos dados com verificação de erro, confimação de entrega e retransmissão além de serem não ideais para a aplicação, mascaram o estado da rede e impedem a análise correta quanto a qualidade do serviço.

Ele também possui um protocolo de controle, baseando-se no conceito de que a aplicação é a entidade mais indicada para detectar e resolver possíveis problemas que possam ocorrer como excesso de perdas de pacote e atraso na entrega, dentre outros. Nesse contexto, existe o RTCP que com o envio periódico de pacotes, consegue realizar uma estatística bem razoável quanto a situação da transmissão e QoS. Com isso, fornece parâmetros para a aplicação reagir de maneira coerente para resolver algum destes problemas potenciais.

Seu conceito foi idealizado visando suprir a necessidade de um protocolo para um cenário de conferência multimídia, porém sem estar limitado apenas a isto. Outras aplicações tais como armazenamento contínuo de dados e aplicações de controle e medição de parâmetros de rede.

Uma sessão RTP consiste em duas ou mais pessoas que se comunicam utilizando o protocolo que usualmente pode ser dividida em dois casos gerais, unicast e multicast. No unicast, dois pontos se conectam diretamente, ponto a ponto. No multicast, se a arquitetura da rede suportar, diversos pontos se conectam utilizando um IP multicast para o envio e recepção dos pacotes. Também pode se trabalhar com protocolos de rede distintos, como IP, ATM utilizando tradutores que interligam as redes, ficando o RTP sobre elas com total compatibilidade.

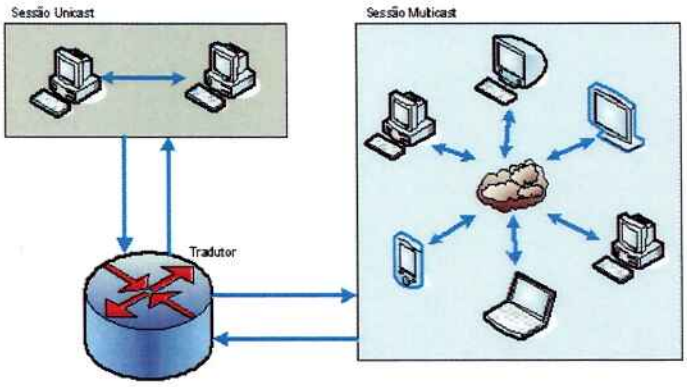

Figura 6 – Tradução unicast – multicast

2.2.2.1. Principais definições do RTP:

- **RTP Payload:** Dados transportados via RTP em um pacote.
- **Pacote RTP:** Pacote de dados contendo o cabeçalho e o payload.
- **Pacote RTCP:** Pacote de controle que consiste em um cabeçalho fixo e elementos estruturados que variam em função do tipo de pacote.
- **Endereço de transporte:** Combinação de um endereço de rede e uma porta que caracterize o destino (como um endereço IP e uma porta UDP, por exemplo).
- **Sessão RTP:** Uma associação num conjunto de participantes que se comunicam por meio do protocolo RTP. Um elemento da sessão pode possuir e diferenciar diversas sessões RTP fazendo uso de diferentes pares de endereços de transporte.
- **Synchronization Source (SSRC):** A fonte de um fluxo de pacotes RTP, identificado por um valor numérico de 32 bits contido no cabeçalho RTP de forma a não depender do endereço de rede. Este valor é escolhido aleatoriamente, para que seja único por fonte numa sessão RTP. Devem ainda ser definidos SSRCs distintos quando uma mesma fonte gera fluxos RTP diferentes.
- **Contributing Source (CSRC):** Fonte de um fluxo de pacotes RTP que contribui para formar um fluxo combinado gerado por um mixer, como será mais detalhado adiante.
- **End system:** Uma aplicação que gera o conteúdo a ser enviado em pacotes RTP ou consome o conteúdo dos pacotes RTP recebidos.
- **Mixer:** Um sistema intermediário que recebe pacotes RTP de uma ou mais fontes, possivelmente modifica o formato dos dados, combina os pacotes de alguma forma e os encaminha em um novo pacote RTP.
- **Translator:** Um sistema intermediário que encaminha pacotes RTP mantendo suas fontes de sincronismo intactas.

2.2.2.2. Formato do pacote RTP

O pacote RTP possui tipicamente o formato da figura abaixo. Uma sucinta descrição de cada campo e sua funcionalidade é apresentada a seguir.

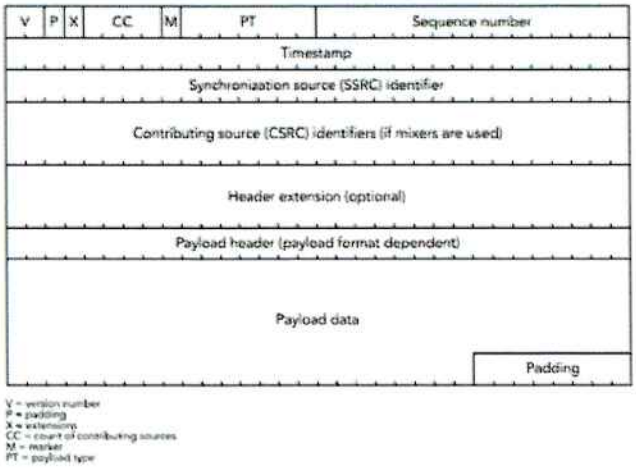

Figura 7 - Formato do pacote RTP

- **version (V 2 bits):** Este campo identifica a versão do protocolo. Baseado na RFC3550, este campo possui atualmente o valor 2.
- **padding (P 1 bit):** Este campo define, se setado, que existem campos adicionais (um ou mais octetos) ao final do pacote, porém que não pertencem ao payload. Isso é usado quando se utiliza codificação dos dados que precisam de um tamanho fixo de informação para seu algoritmo.
- **extension (X 1 bit):** Se setado, uma extensão de header definida deve seguir logo após o header usual. Este header extendido possui uma forma regular definida pela normalização do protocolo.
- **CSRC count (CC 4 bits):** Este número defini quantos CSRC existem no pacote.
- **marker (M 1 bit):** Em princípio este campo serve para marcar eventos importantes. Este campo pode tanto ser utilizado quanto desabilitado, sendo que no segundo caso, campo payload type aumenta para 8 bits.
- **sequence number (16 bits):** Este número é inicialmente gerado aleatório e conforme pacotes são enviados, ele é incrementado. O valor deste campo serve para determinar perdas de pacote e efetuar seu seqüenciamento.
- **payload type (PT 7 bits):** Este campo identifica o formato do payload e determina sua interpretação pela aplicação.

- **SSRC (32 bits):** O campo SSRC identifica a fonte de sincronismo. Ele deve ser escolhido randomicamente com a intenção de não existirem duas fontes dentro da mesma sessão RTP.

2.3. Tecnologias envolvidas – implementação

Neste projeto procuramos explorar as competências aprendidas na faculdade:

- Software: modelagem orientada a objetos (UML), gestão de projetos e desenvolvimento.
- Desenvolvimento de sistema: definição de processos, requisitos, elaboração de arquitetura e especificação de sistema.
- Redes: protocolos, transmissão de dados.
- Fundamentos de computação: modelagem baseada em autômatos, linguagens, gramáticas, máquinas de estado finito.
- Processamento de sinais: filtros adaptativos, compactação e descompactação de sinais.
- Outros: visão de futuras áreas de atuação, potenciais mercados consumidores.

No que diz respeito à implementação, optamos pelo uso da linguagem Java, pois possui um forte conjunto de soluções open source, que nos serviram de base para estudo e implementação, além da portabilidade fornecida por esta linguagem, o que nos possibilitou visualizar futuras implementações do nosso projeto em outras plataformas, como por exemplo, dispositivos móveis. Como ambiente de desenvolvimento, escolhemos a ferramenta eclipse por esta propiciar um ambiente de programação eficiente e com grandes recursos. Alem dos recursos nativos também é possível agregar outras funcionalidades através de plug-in, como jigloo que proporciona excelentes recursos para o desenvolvimento de interfaces em java.

Infra-estruturas voltadas à implementação dos protocolos usados em transmissão de voz sobre IP, JainSip na sinalização e JMF no processamento e transmissão de voz, também foram incorporadas a este trabalho. Estes serão apropriadamente descritos nos itens (5.2) e (5.3).

3. Especificação do Projeto de Formatura

3.1. Definição do escopo do projeto

Tendo em vista que o projeto é baseado em serviços de voz sobre IP, escolhemos o caso da comunicação entre dois usuários por duas razões. Primeiramente serve como ponto de partida para entender o funcionamento dos protocolos já citados e em segundo lugar ele compõe as funcionalidades básicas de operação necessárias para todos os outros processos, dentre os quais podemos destacar:

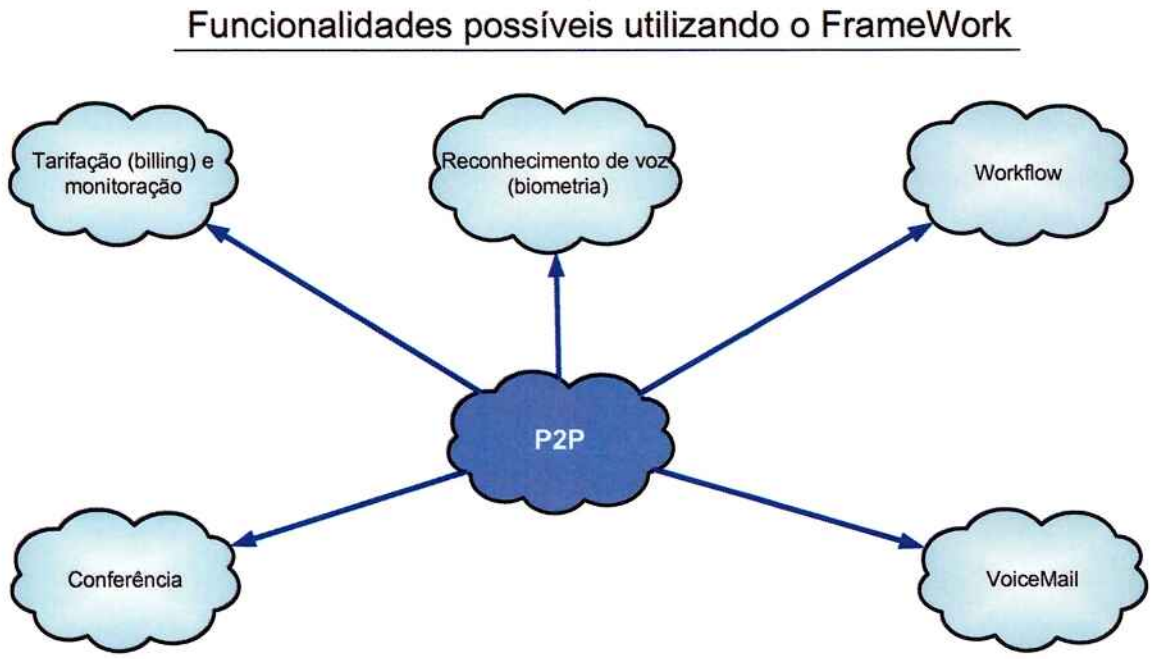

Figura 8 - Possíveis módulos do Framework

- Áudio-conferência: em uma primeira análise, uma evolução do caso da conversa entre duas pessoas. No entanto demanda uma elaboração de como será feita a transferência de pacotes de voz e sinalização entre os participantes.
- Tarifação: caso de uso que envolve o uso de um sistema capaz de "logar" as ações e características da utilização do serviço de modo a possibilitar a tarifação de acordo com a configuração e vontade de cada cliente.
- Monitoração: da mesma maneira descrita acima, podemos utilizar estes logs gerados para obter relatórios da qualidade de serviço (QoS) e qualquer outro dado de interesse.
- Reconhecimento de voz: o framework também poderia prever a utilização de reconhecimento de voz para questões de segurança, autenticação e chaves de acesso. Apesar do fato deste caso não ser o foco deste trabalho, ele ilustra a possibilidade de se acoplar outras funcionalidades ao sistema.

- Workflow: poder-se-ia também integrar um módulo de workflow que utilizasse os serviços de conferência, redirecionamento, voice-mail, dentre outros, de forma a prover uma grande capacidade de comunicação interna.

3.2. Estudo do Caso de Uso P2P

3.2.1. Descrição do Processo

Uma conversa telefônica entre duas pessoas inicia-se por um dos lados (pessoa1) manifestando a vontade de iniciar uma conversa. A partir de então, o outro lado (pessoa2) recebe um aviso indicando que ela está sendo requisitada para uma conversa. Esta pode então aceitar ou recusar a solicitação. Aceitando, começa então a fase de troca de mensagens (voz) entre as duas pessoas. Assim que um dos lados desejar terminar a conversa, deve-se então requisitar o fim da conexão, terminando então a conversa.

Além disso, quando a pessoa2 não aceita a chamada, esta passará por um tratamento que enviará a resposta à pessoa1, baseando-se em configurações pré-estabelecidas. Tais tratamentos podem ser: enviar sinal de ocupado, transferir para caixa postal ou redirecionar para outro número.

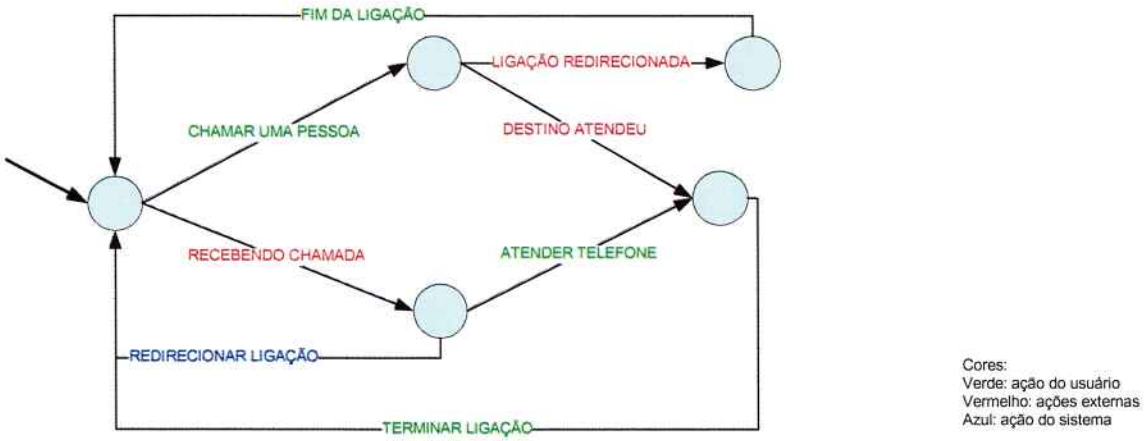

Figura 9 - Processo de Conversa P2P

3.2.2. Requisitos funcionais

- Desenvolvimento de uma infra-estrutura que propicie fácil desenvolvimento de sistemas que utilizem tecnologia de voz sobre IP como parte de seu negócio.
- Fornecer comunicação de voz entre dois usuários.
- Fornecer registro dos eventos ocorridos durante o processo de comunicação.

- Fornecer registros que informem parâmetros de qualidade da rede, como jitter, banda, latência, perda de pacote e atraso.

3.2.3. Requisitos não funcionais

- Segurança: O sistema deve garantir a integridade das transmissões tanto no que diz respeito a entrega quanto a confidencialidade.
- Disponibilidade: Tendo em vista a natureza da aplicação, deve-se garantir uma disponibilidade de 99%.
- Confiabilidade: As informações trafegadas na rede devem manter alto nível de fidelidade, ou seja, as vozes transmitida e reproduzida devem ser idealmente iguais.
- Transparência: Os usuários do *framework* não necessitam ter conhecimento avançado sobre os processos e protocolos envolvidos na comunicação de voz sobre IP.

3.2.4. Arquitetura Adotada

A arquitetura aqui adotada visa não apenas a existência de dois usuários, mas sim a existência de vários usuários que se comunicam entre si utilizando conversas P2P. Além disso, a existência de servidores permite expandir a variedade de serviços prestados, tais como tarifação (*billing*), audio-conferência, caixa postal centralizada e redirecionamento caso o usuário se encontre em outro local definido.

Os componentes envolvidos nesta arquitetura são:

- Cliente: equipamento que o usuário manipula para efetuar (e receber) chamadas
- Servidor: máquina que concentra serviços de tratamento de requisições tais como registro de usuários, tarifação, redirecionamento e localização do usuário.

Estes componentes realizam a comunicação entre eles utilizando redes TCP/UDP/IP, juntamente com outros protocolos de tecnologia de voz sobre IP (SIP, SDP, RTP, RTCP) que possibilitam a comunicação entre os usuários.

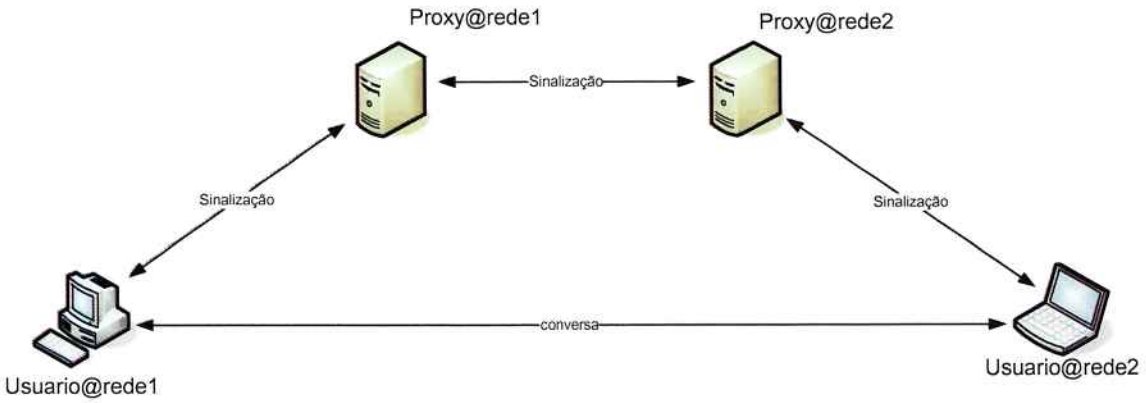

Figura 10 - Arquitetura de Comunicação P2P

3.2.5. Modelagem do Sistema

3.2.5.1. Cliente

O cliente é composto por três blocos denominados Controlador de Conexões, Máquina de Estados de Conexão (MEC) e Máquina de Estados de Redirecionamento (MER), ligados conforme figura abaixo:

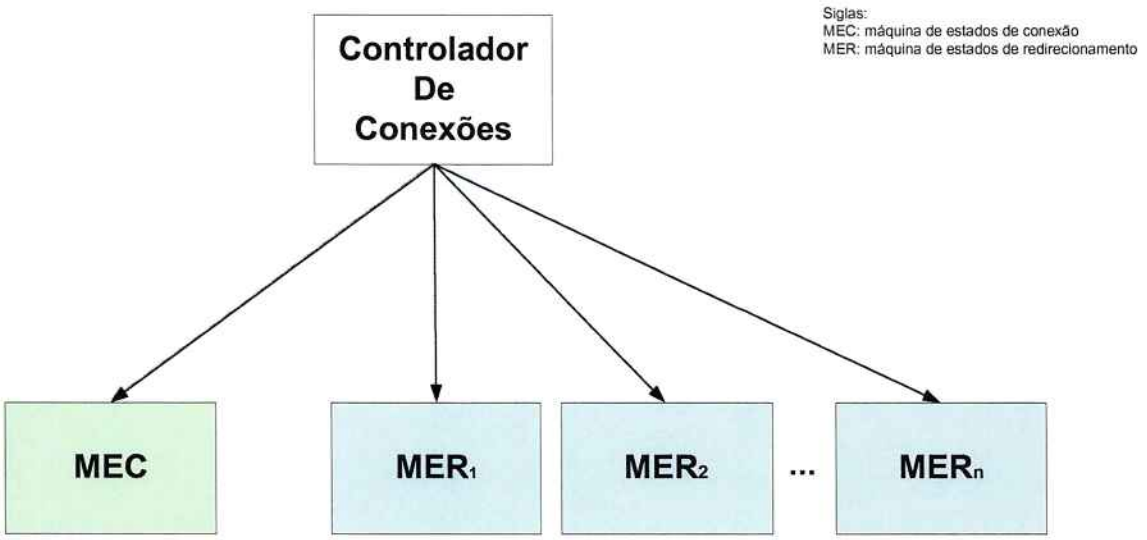

Figura 11 - Diagramas de Blocos do Cliente P2P

Destes blocos, o Controlador de Conexões tem por finalidade gerenciar o estado atual do cliente, e tomar as decisões necessárias conforme as requisições são recebidas, criando e delegando ações para as máquinas MEC e MER conforme diagrama abaixo:

A MEC, responsável por coordenar a troca de mensagens cliente-servidor (protocolo de sinalização – SIP) referentes a uma conversa em andamento, realiza essa troca de mensagens conforme a figura abaixo.

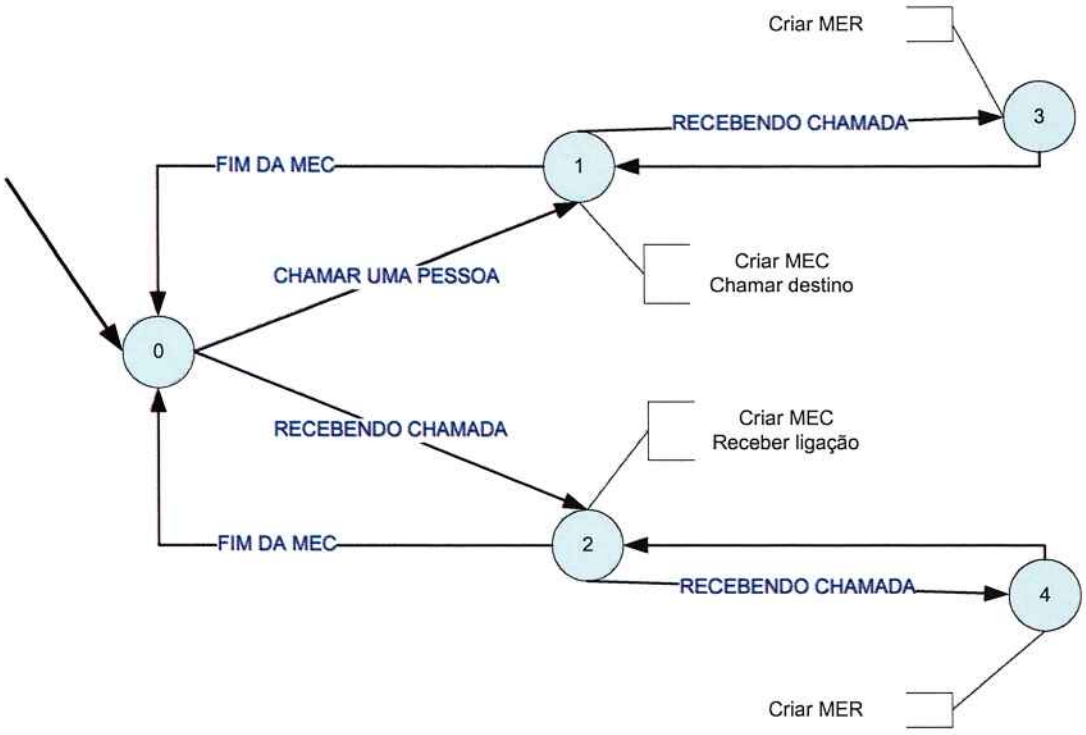

Figura 12 – Máquina de Estados do Gerenciador de Conexões – P2P

Vale lembrar que, apesar de, a princípio existir apenas uma instância da MEC no cliente, poderão existir mais de uma instância simultânea se for considerado o caso de uso do recurso de chamada em espera, que seria uma extensão ao caso estudado.

Já a MER, dada a sua natureza, é uma máquina que pode ter mais de uma instância simultânea, visto que para ser instanciada, basta que seja feita uma requisição de conexão quando o cliente está indisponível.

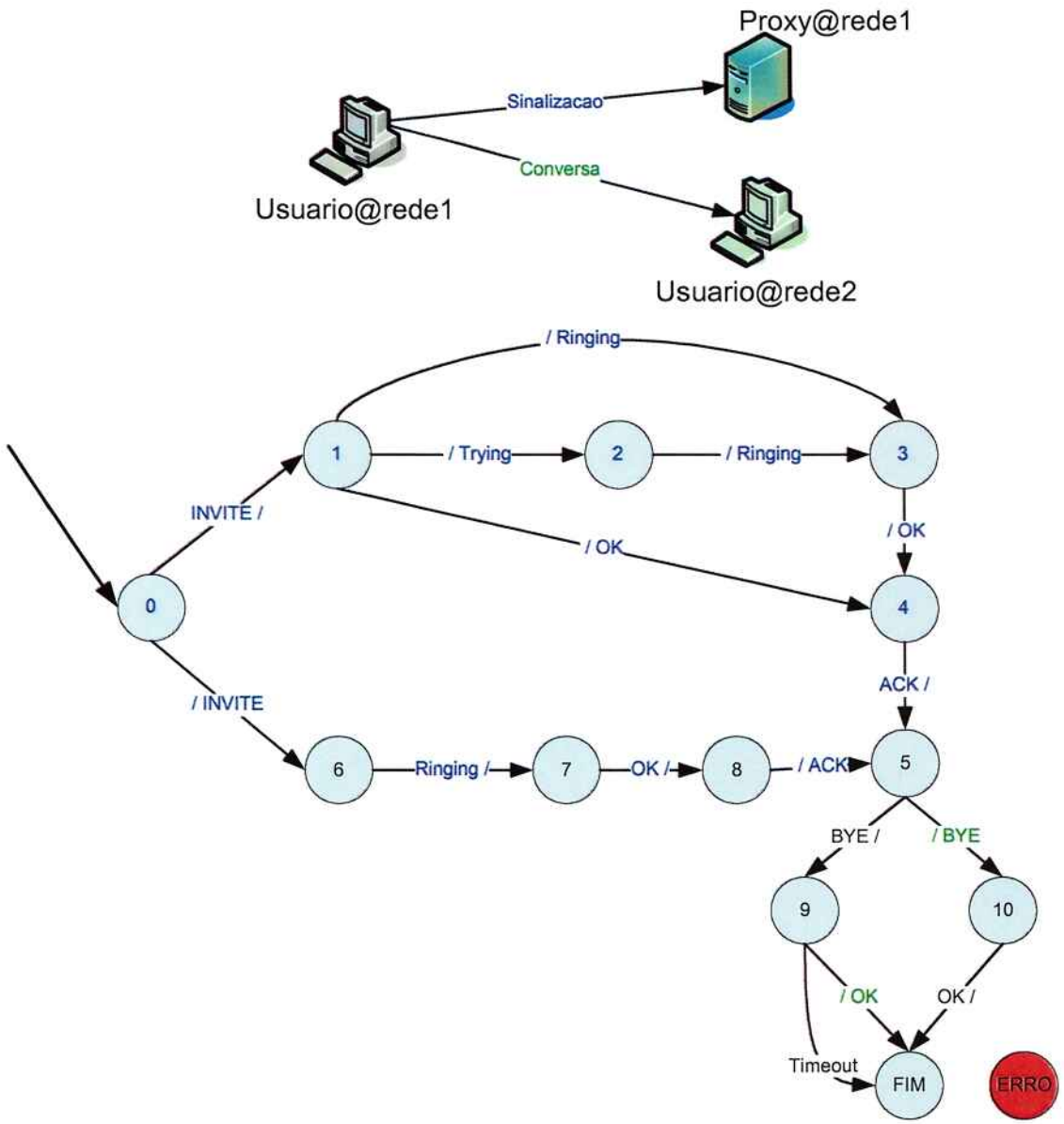

Figura 13 - Máquina de Estados da MEC – P2P

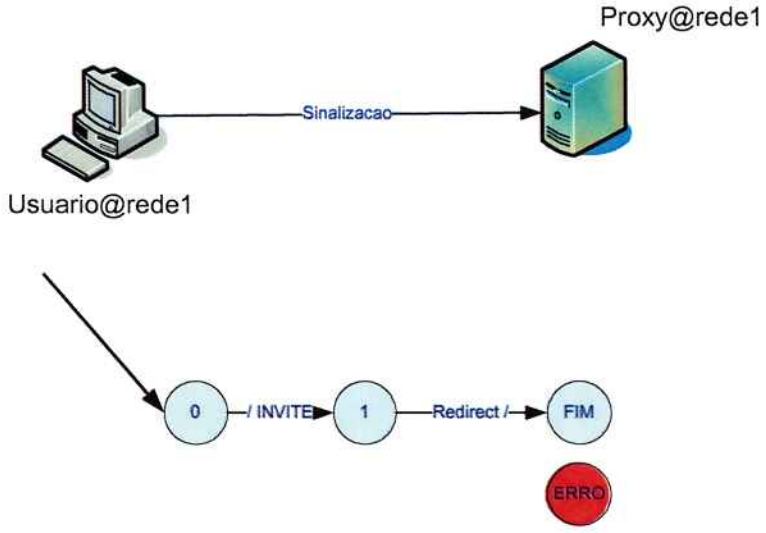

Figura 14 - Máquina de Estados da MER - P2P

3.2.5.2. Servidor

O servidor é composto por três blocos denominados Controlador de Conexões, Máquina de Estados de Conexão Servidor (MECS) e Máquina de Estados de Redirecionamento Servidor (MERS), ligados conforme figura abaixo:

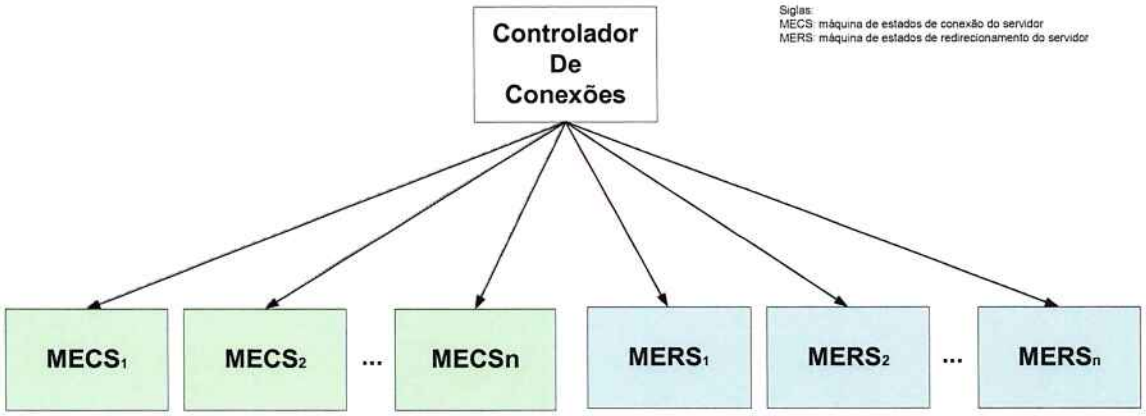

Figura 15 - Diagrama de Blocos - Servidor

O Controlador de Conexões tem por finalidade gerenciar as informações de cada cliente, e tomar as decisões necessárias conforme as requisições são recebidas, criando e delegando ações para as máquinas MECS e MERS conforme diagrama abaixo:

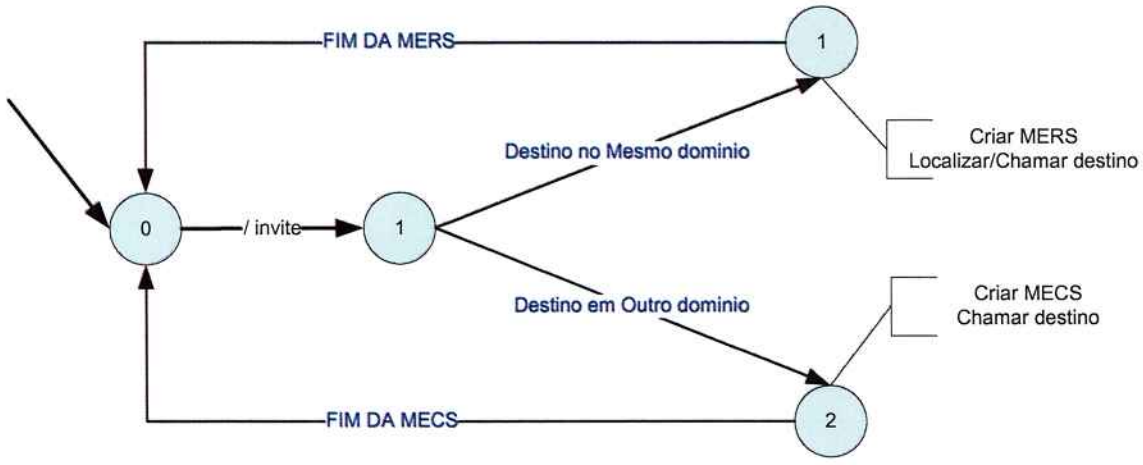

Figura 16 - Máquina de Estados do Gerenciador de Conexões- Servidor

A MECS, responsável por coordenar a troca de mensagens cliente-servidor e servidor-servidor (protocolo de sinalização – SIP) referentes a uma requisição de um usuário, é instanciada quando o destino procurado pertence a outro domínio e realiza troca de mensagens conforme a Figura 17 - Máquina de Estados da MECS.

A MERS, responsável por coordenar a troca de mensagens cliente-'servidor proxy' e 'servidor proxy'-'redirect server' (protocolo de sinalização – SIP) referentes a uma requisição, é instanciada quando o destino pertence a este 'servidor proxy'.

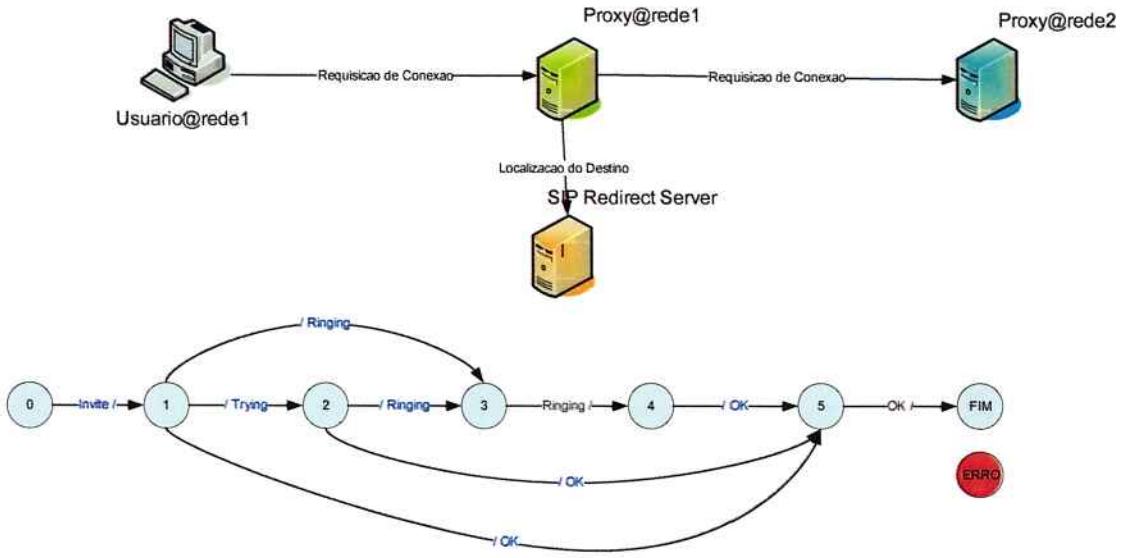

Figura 17 - Máquina de Estados da MECS

O servidor 'redirect server' é responsável por determinar em que domínio um determinado usuário esta logado.

Vale salientar, que podem existir varias instancias de MECS e MERS, pois estas são instanciadas pelo servidor quando este recebe uma requisição.

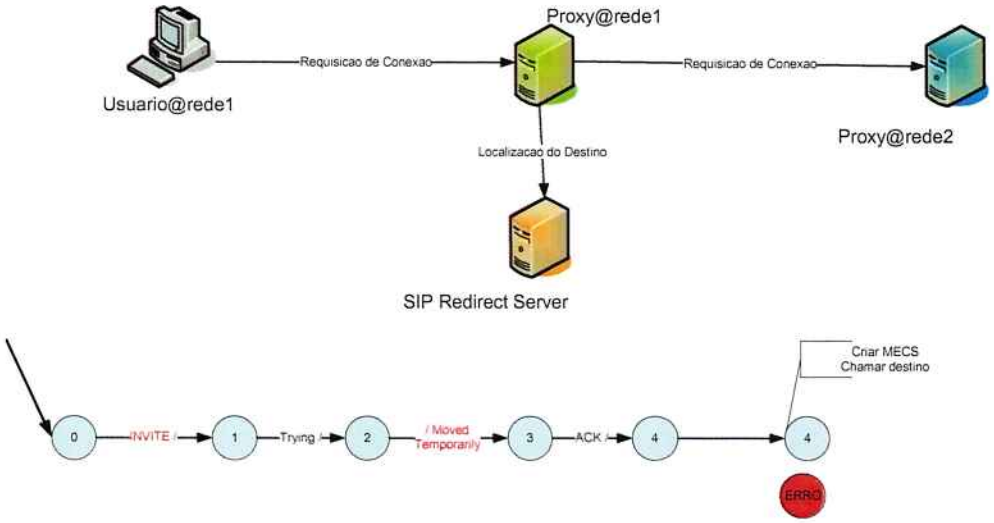

Figura 18 - Máquina de Estados da MERS

4. Metodologia

4.1. Proposta inicial

Em nossa proposta inicial, almejávamos modelar e desenvolver uma infraestrutura de um *framework* VoIP que fornecesse uma plataforma para desenvolvimento de aplicativos baseados no serviço de voz sobre IP com maior abstração e facilidade de implementação. Essa proposta fora elaborada levando-se em consideração a análise realizada pelo grupo a respeito da utilização e projeção que esta tecnologia apresentava.

Para validar a modelagem e idealização do sistema, abordamos o caso P2P que resume a funcionalidade comum às demais funcionalidades que podem ser implementadas com o *framework* tais como conferência, voice-mail, billing, biometria e ferramenta de gerenciamento de workflow.

O caso P2P é comum, pois reúne todas as funcionalidades básicas inerentes ao sistema de comunicação de voz sobre IP. Ele inclui os serviços de sinalização, processamento e transporte da voz.

Também seriam geradas estatísticas e informações relevantes sobre as chamadas como jitter, banda, perda de pacotes e atrasos – parâmetros usados para determinar a qualidade de serviço (QoS) – como também tempo de conversação, horário das chamadas dentre outras informações. Com isso, obtém-se um quadro completo sobre a utilização do serviço e possibilitando a criação de regras de cobrança (billing).

4.2. Alterações de proposta de trabalho

Na primeira apresentação realizada à banca do departamento de computação e sistemas digitais (PCS), nos foi recomendado que reduzíssemos o escopo e objetivos finais do projeto de formatura. O motivo alegado foi o grande número de atividades que a proposta inicial demandaria contraposto ao tempo reservado para sua realização.

Outro fator de relevância que acabou ocasionando redução de escopo foi a saída inesperada de um integrante do grupo devido a uma oportunidade de trabalho no exterior.

Associado a estes, também enfrentamos dificuldades no entendimento dos protocolos, pois são soluções extremamente complexas e extensas devido ao fato de serem soluções completas de mercado, indo muito além do que o projeto demanda.

A soma destes fatores resultou na redução geral do projeto, de modo que tivemos que escolher módulos que fossem mais relevantes na validação da proposta sem perda

significativa de valor em relação ao idealizado inicialmente, tendo em vista o foco do projeto consiste na modelagem.

O resultado dessa redução reflete na nova proposta final apresentada em (4.3).

4.3. Proposta final de plano de trabalho

A proposta final do *framework* consiste na modelagem e implementação das classes do caso de uso P2P, que é formado pela sinalização (SIP) direta entre dois usuários, sem utilizar servidor Proxy como apresentado na arquitetura (3.2.5), codificação/decodificação da voz em um padrão atrelado ao JMF e na transmissão dos pacotes de dados pela rede (RTP).

Além disso, também fazem parte a implementação das classes da aplicação P2P, tais como a interface homem-máquina e a lógica de negócio envolvida. Como supracitado, estas classes utilizam a infra-estrutura criada pelo *framework*.

4.4. Metodologia da criação do projeto

O projeto foi feito a partir do conceito de *framework*, que é um conjunto de classes e interfaces que servem como um elemento para ser incluído em outros projetos maiores.

Para tanto, uma seqüência de ações foi criada para visualizar como o projeto final deveria ser:

- Analisar o campo de voz sobre IP.
- Traçar os limites de atuação do *framework* que viria a ser projetado.
- Analisar as soluções pré-existentes no mercado, que poderiam ser integradas com o *framework*, isto é, produtos que auxiliariam no desenvolvimento do projeto, servindo como base de sustentação, realizando a comunicação com as camadas inferiores de rede. Dentre os encontrados, dois foram utilizados no projeto: *Java API for Integrated Networks* – SIP (*Jain-SIP*) e *Java Media Framework (JMF)*.
- Uma vez decidido que os dois pacotes seriam utilizados, foi necessário estudar estes pacotes, com o intuito de entender suas características e modo de utilização.
- Montar uma prova de conceito, que teria a função de demonstrar que o *framework* modelado é funcional.
- Implementar um caso de uso para ilustrar o funcionamento da modelagem.

4.5. Componentização

O processo compõe-se de três grandes blocos: P2P, serviços e monitoração. O P2P possibilita a conexão e a conversa entre dois usuários e gera logs de conexão e conversa que

são armazenados para serem processados posteriormente. O bloco de serviços agrega funcionalidades ao sistema e também gera logs para o sistema de monitoração. Já o bloco de monitoração utiliza os registros de log para realizar a tarifação e serviços de QoS.

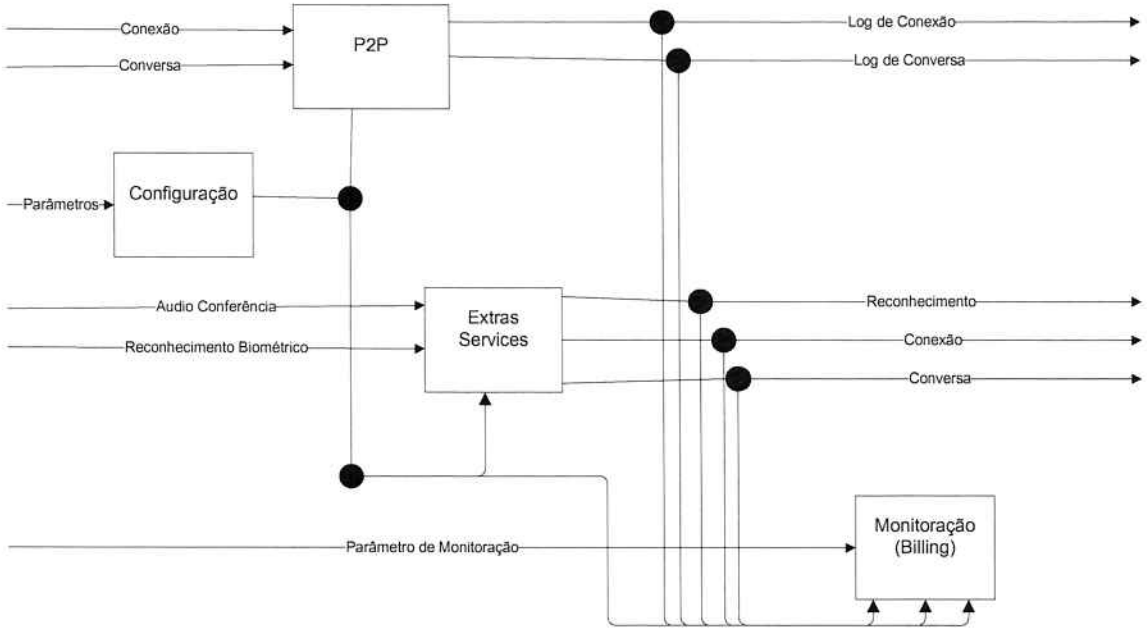

Figura 19 - Processo Operacional

A componentização, por sua vez, agrega classes com funcionalidades semelhantes criando agrupamentos que facilitam a organização e distribuição do sistema. Eles foram classificados da seguinte maneira:

- Módulo de dados de Usuário: Banco de dados que contem informações dos usuários do sistema.
- Módulo de Conexão: Responsável por estabelecer a conexão entre usuários. Acessa dados do banco de dados e troca informações com o módulo de conversa.
- Módulo de Conversa: Encarregado de realizar a troca efetiva dos dados da comunicação. O módulo de conexão tem o controle e opera sobre este módulo.
- Módulo de Log: Ambos os módulos de conexão e conversa geram saídas que são capturadas e filtradas pelo módulo de log de forma a gerar informações relevantes para a tarifação e qualidade.

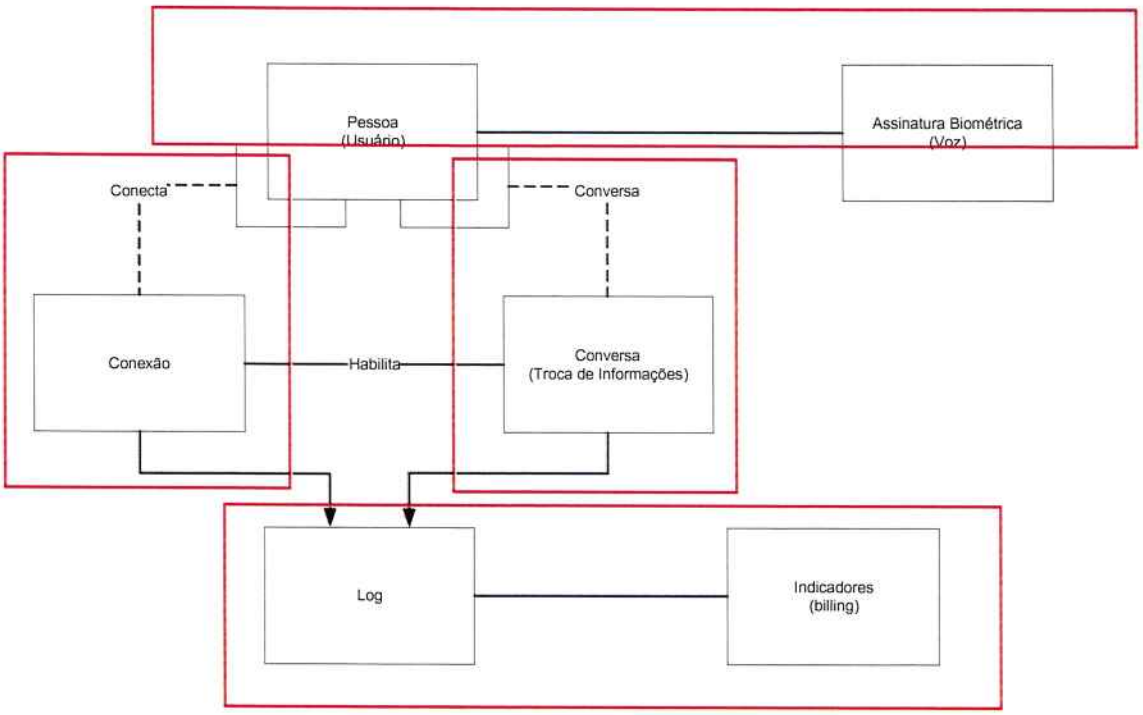

Figura 20 – Componentização

4.6. Pontos de checagem do andamento do projeto

Para que o projeto não trilhasse caminhos indesejados, foi planejado que fossem utilizados pontos de checagem mensais, que, apesar de possuírem metas especificadas no começo do projeto (vide cronograma), tiveram as metas alteradas dinamicamente, para atender as necessidades que foram surgindo ao decorrer do projeto.

Tabela 1 - Pontos de checagem

	Mai	Jun	Jul	Ago	Set	Out	Nov
Definição das funcionalidades							
Modelagem							
Prova de conceito							
Implementação parcial							
Testes							

4.7. Plano de recuperação

No final de Julho, houve o primeiro ponto de checagem com resultados indesejados – a prova de conceito se fez mais complexa do que o esperado, por causa da dificuldade de compreensão do funcionamento dos dois pacotes das camadas inferiores. Como não existiam

muitas alternativas, o caminho escolhido foi prolongar o período de dedicação à prova de conceito, até o final de Agosto, aproximadamente.

Já no início de Novembro, a implementação acabou não chegando ao ponto desejado, que era um cliente funcional, tendo concluído aproximadamente 70% do esperado, faltando a conclusão e integração das partes implementadas. Foi montado então um cronograma para conciliar as tarefas corriqueiras com o projeto de formatura, de forma a recuperar o andamento do projeto e montar um produto completo até a data da entrega. O cronograma produzido então, encontra-se no item cronograma de atividades.

5. Projeto e Implementação

5.1. Projeto

A modelagem do *framework* visa à idealização de uma camada que não se limite apenas à comunicação de voz sobre IP, como também serviços que sejam interessantes no meio corporativo. Sistemas como tarifação e monitoração, utilização de reconhecimento de voz (biometria) como instrumento de segurança e suporte para o desenvolvimento de um ambiente automatizado de *workflow* são exemplos de serviços que podem ser agregados ao framework, pois são itens de relevância para este nicho de mercado.

Em linhas gerais, o framework será um sistema flexível, que facilitará a implementação de soluções que utilizem voz sobre IP como ferramenta de comunicação, sendo composto de diversos módulos que agregam funcionalidades ao sistema. A figura abaixo ilustra a posição do framework entre as camadas de desenvolvimento de um projeto.

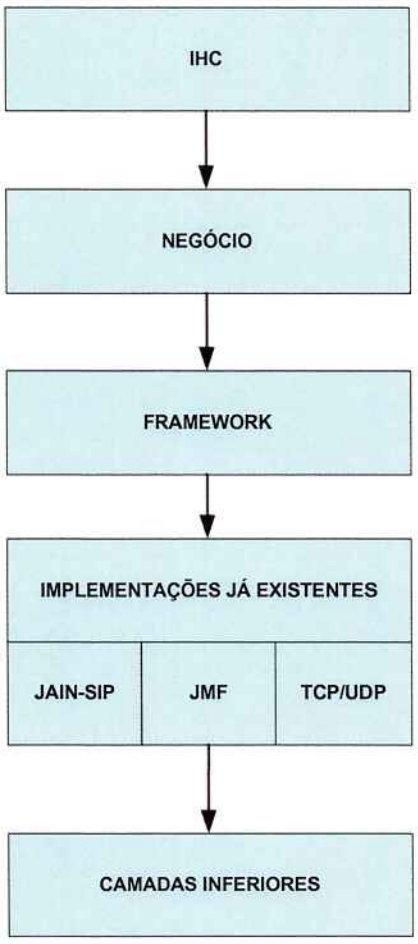

Figura 21 - Camadas de projeto

Devido à característica modular da modelagem do framework, este prevê a adição de novas funcionalidades além das modeladas neste trabalho, o que a torna uma solução muito poderosa, na medida em que muitos serviços podem ser incorporados, acompanhando a característica dinâmica do mercado, como novos protocolos de comunicação, padrões de codificação e diferentes plataformas.

Para atender estes requisitos, se faz necessária, para a implementação, a utilização de uma linguagem orientada a objetos (modular), flexível e portável e que provenha uma infra-estrutura básica da tecnologia VoIP, como protocolos (SIP, RTP) e padrões de codificação (G.723, entre outros).

Um sistema de VoIP pode ser divido em blocos, que são:

- Aplicação: contém a lógica de negócio do produto criado, além da interface homem-máquina envolvida na mesma;
- Framework contém a lógica criada para aumentar o nível de abstração dos itens já existentes que trabalham com VoIP. É o escopo deste projeto;
- Áudio (JMF): responsável pela codificação e decodificação dos sinais de voz, além do transporte em tempo real dos mesmos;
- Sinalização (JAIN SIP): bloco responsável pela codificação, decodificação e transmissão dos pacotes relacionados à sinalização;
- JRE: outras classes que são provenientes da máquina virtual do Java.

Um diagrama representando estes blocos e suas interconexões segue abaixo:

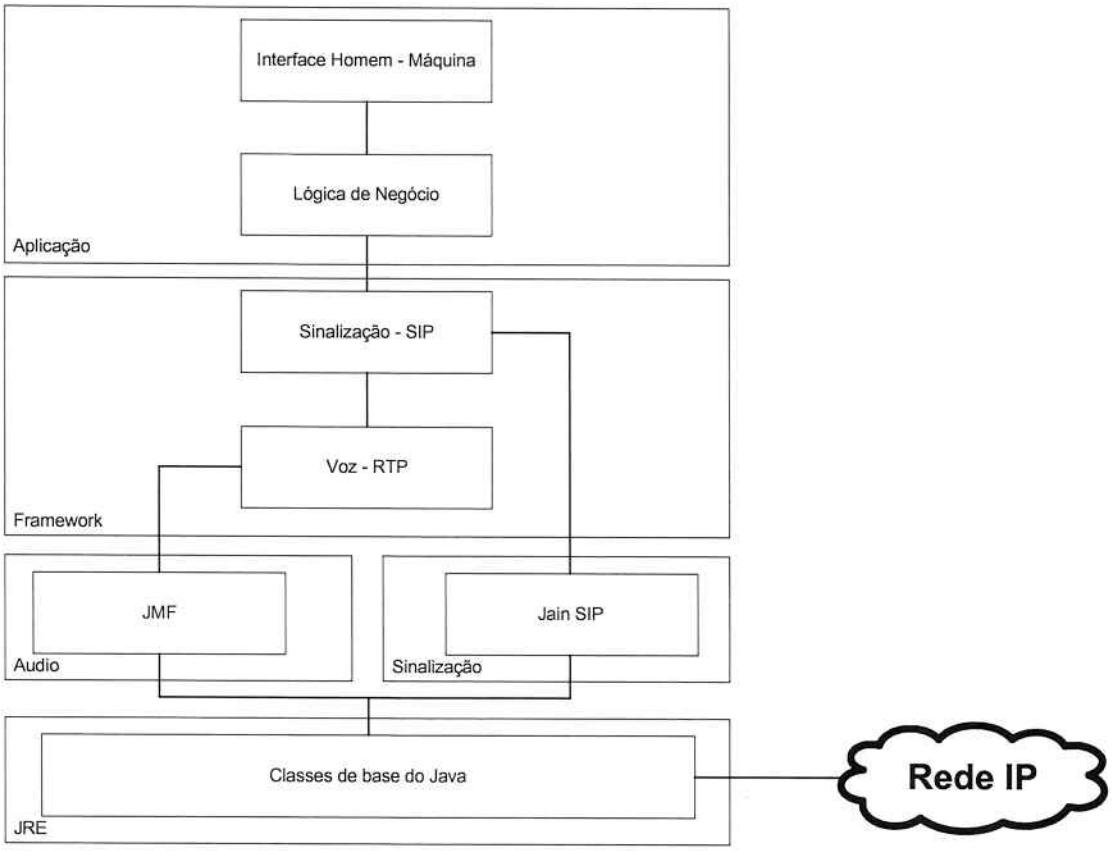

Figura 22 - Divisão em blocos de um sistema de VoIP

5.2. *Sinalização*

5.2.1. JAIN SIP

O JAIN SIP é um conjunto de classes que fornece as funcionalidades básicas relacionadas ao protocolo SIP em baixo nível.

No caso, foi utilizado um pacote denominado NIST SIP, que é uma versão fornecida pelo NIST, contendo uma implementação de referência do JAIN SIP, com exemplos de utilização.

Entre os serviços fornecidos pelo JAIN SIP, temos:

- Prover métodos para formatar e enviar as mensagens do protocolo SIP;
- Realizar análise das mensagens SIP recebidas e prover acesso aos campos das mensagens de forma padronizada para Java;
- Chamar métodos para os eventos significativos do protocolo, por exemplo: nova mensagem recebida e timeout de envio;
- Gerenciar as transações e diálogos do protocolo SIP quando pedido pelo framework.

5.2.2. Arquitetura do JAIN SIP

O JAIN SIP é composto por um conjunto classes e interfaces, sempre existindo em pares contendo uma interface e uma classe que a implementa. As principais classes são as seguintes:

- SipStack: É o elemento mais próximo da camada de rede, estando relacionado com a arquitetura do protocolo SIP, implementando a especificação da mesma. Também possui a função de gerenciamento do JAIN SIP, sendo o ponto concentrador dos outros elementos;
- ListeningPoint: Responsável por representar o par (protocolo, porta), sendo assim uma representação de um ponto de escuta de recebimento de dados;
- SipProvider: Gerencia as mensagens enviadas e recebidas pelo JAIN SIP, realizando chamada de métodos no SIPListener quando chegam mensagens e é por ele que o SIPListener realiza o envio de mensagens;
- SipListener: Interface que diz respeito à aplicação, provendo as declarações dos métodos necessários para a comunicação com o protocolo. Quando é desenvolvido um aplicativo utilizando o JAIN SIP, a classe que está em contato com o pacote deve implementar esta interface. No nosso caso, o controlador de conexões é quem implementa a interface SipListener, como veremos em (5.4).

  Um diagrama demonstrando o relacionamentro entre estas entidades segue abaixo.

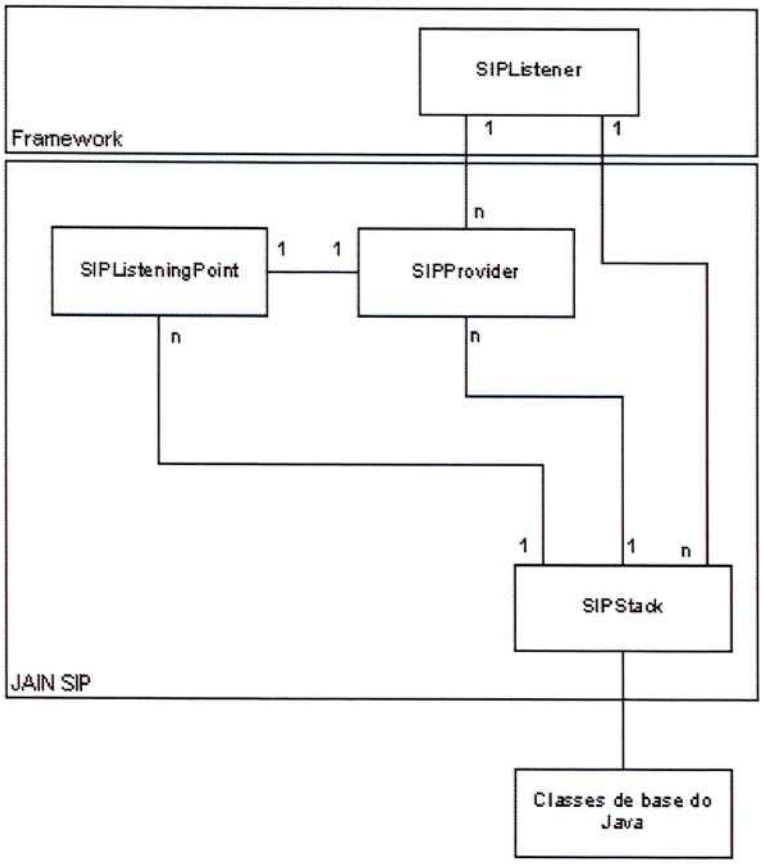

Figura 23 – Principais classes do JAIN SIP

Além disso, existem outras classes que servem de apoio às classes supracitadas:

- SipFactory: é utilizada para criação de objetos da classe SipStack;
- Transaction, ClientTransaction e ServerTransaction: classes que representam uma transação no protocolo SIP, que é um conjunto de mensagens trocadas entre uma requisição e a resposta final (não-provisórias);
- Dialog: classe que representa um diálogo, que é um conjunto de todas as mensagens trocadas em uma conexão ponto a ponto, podendo possuir quantas transações forem necessárias.

Além disso, existem outras classes de menor importância, voltadas para os atributos das mensagens trocadas.

Um diagrama ilustrando a relação entre requisições, respostas, transações, diálogos e capacidades de atuação da aplicação neles segue abaixo.

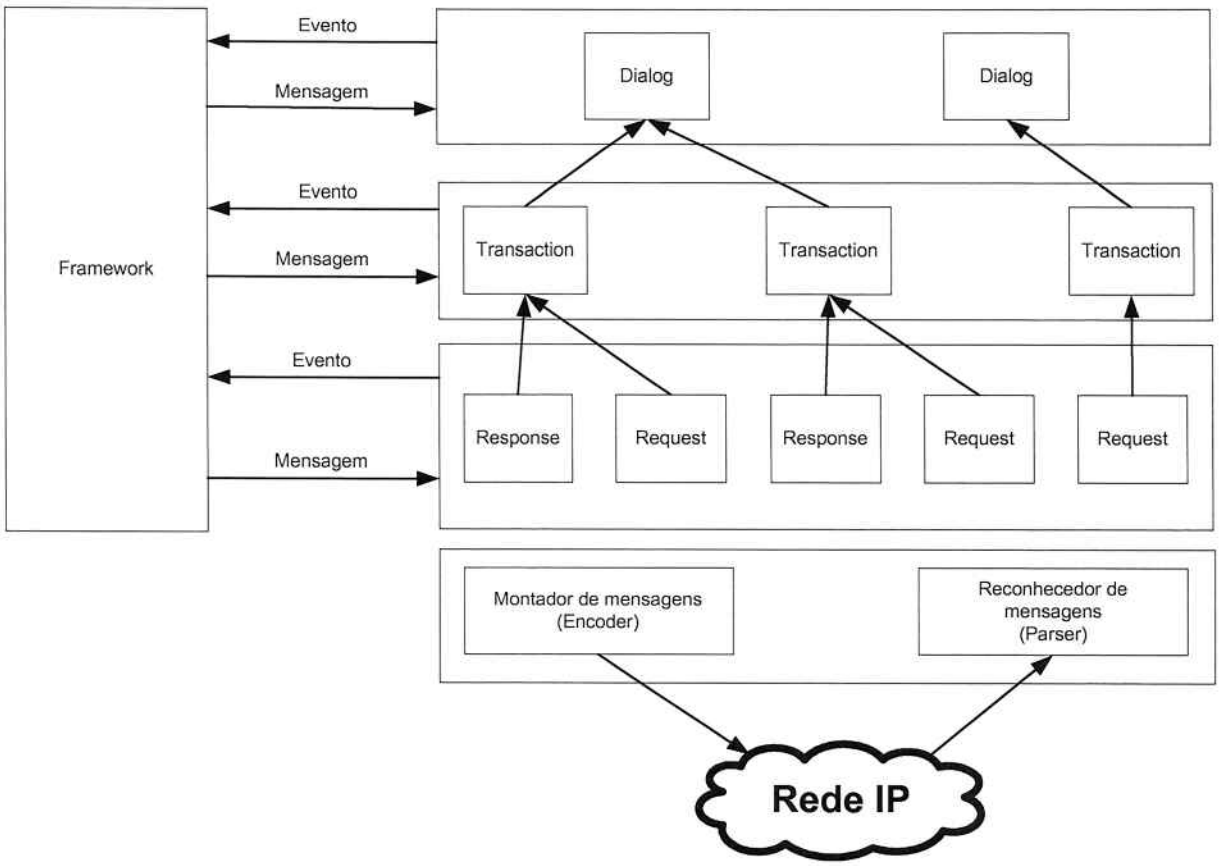

Figura 24 - Transações e Diálogos

O JAIN SIP também define uma lista de deveres que o framework deve atender para interoperar de maneira satisfatória com esta API. São estas:

- O framework deve implementar a Interface SIPListener para poder comunicar com o JAIN SIP
- O envio de mensagens SIP deve ser feita por chamada de métodos da interface SIPProvider.
- O recebimento de mensagens SIP é feito por chamada de eventos na interface SIPListener.

5.3. Processamento e transporte da voz

5.3.1. Processamento e transporte de áudio no JMF

O framework JMF é uma ferramenta desenvolvida pela *Sun Microsystems* que propõe uma suíte de soluções para manipulação, tratamento e transporte de mídia. Ele é utilizado neste projeto para realizar a captura, processamento, transporte via protocolo RTP e reprodução do sinal de voz, conforme o fluxo descrito abaixo.

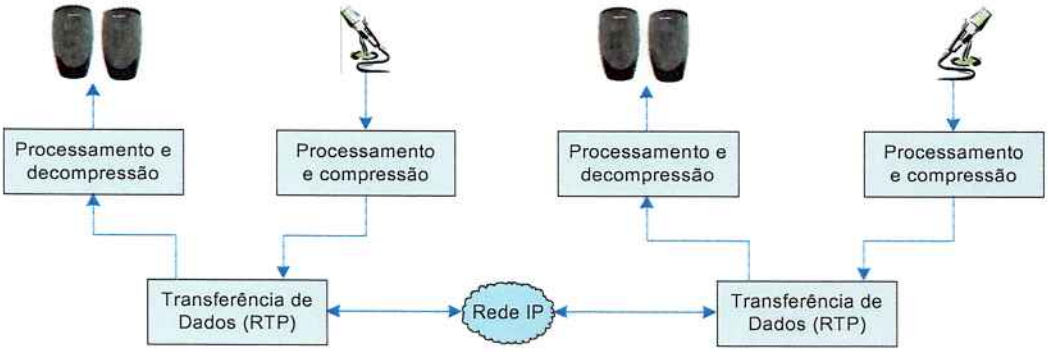

Figura 25 – Fluxo da voz

O processo envolvido em cada uma das etapas do fluxo será detalhado neste item para estender a compreensão de sua utilização no nosso sistema.

5.3.2. O modelo de processos de mídia no JMF

O framework possui um modelo geral que descreve seu funcionamento em três etapas:

- ***Input:*** Consiste na recepção de dados, independente de sua origem. Pode ser proveniente de um dispositivo de captura (microfone), um arquivo de entrada ou dados que lhe foram enviados pela rede.
- ***Process:*** Realiza o processamento das informações recebidas na etapa de Input. As ações usuais nesta etapa seriam realizar a aplicação de algum filtro de efeito, a compressão ou descompressão dos dados ou então convertê-los para outro formato.
- ***Output:*** Constitui na saída de dados. Uma vez realizado o processamento desejado e adequado, o resultado pode ser apresentado ao usuário via caixas de som, salvo em algum arquivo ou enviado pela rede, podendo servir então de input para seu receptor.

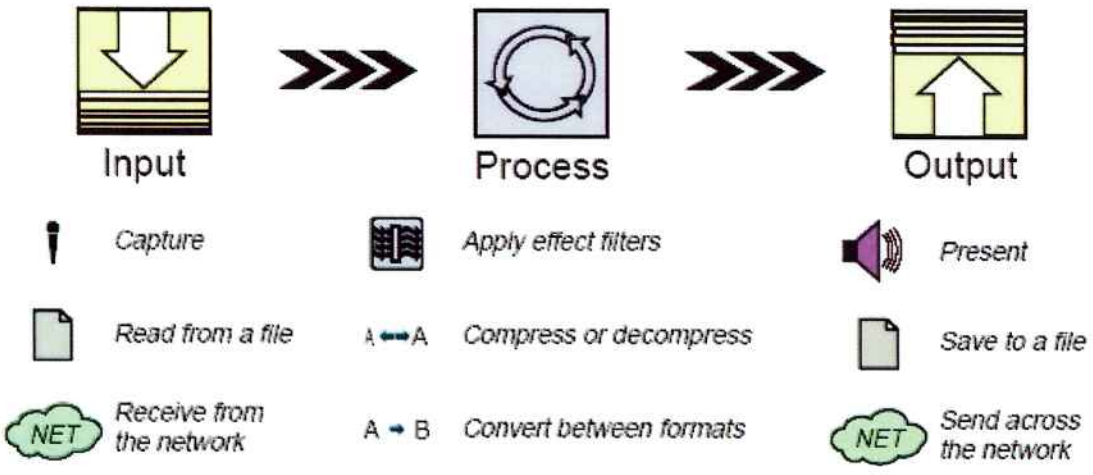

Figura 26 – Modelo de processamento de mídia utilizando o JMF

Com essa descrição do processo, podemos generalizar o funcionamento do módulo de processamento e transporte da voz, combinando origens, destinos e processamentos dentre as três etapas, fechando o ciclo funcional desta parte do Framework VoIP.

5.3.3. O Input

O *input* no sistema de comunicação ocorre em dois momentos. Um deles, na captura do dispositivo de microfone do usuário para processamento, outro no momento em que o usuário recebe os pacotes de voz por RTP pela rede.

Captura de áudio pelo dispositivo

A captura do áudio é um processo composto pelos seguintes passos:

Configuração de parâmetros de amostragem, tais como codificação, bits por amostra, freqüência de amostragem e número de canais.

Identificar o dispositivo de captura, usualmente um microfone.

Estes passos definem como o áudio deve ser inicialmente processado em sua aquisição, preparando para o estágio de processamento onde esta amostragem será codificada e formatada para o envio de pacotes RTP. Esta passagem é feita gerando um *datasource* a partir das definições citadas acima que age como um buffer para a próxima etapa.

Recebimento de pacotes RTP

Ao se receber dados via RTP, é gerado um evento de um novo *stream* de entrada. É criado então um *datasource* a partir deste *stream* de pacotes RTP recebidos, que posteriormente é repassado para uma entidade de reprodução de áudio, servindo de entrada para a fase de *output*.

5.3.4. O Processamento

Esta etapa é composta também por duas origens, porém totalmente análogas em seu tratamento. O sinal pode ser recebido oriundo da aquisição da voz, caso no qual deve ser compactado ou do recebimento de pacotes RTP, devendo ser então descompactado.

O sinal é processado a partir de seu *DataSource*, realizando a compressão ou a descompressão no padrão estabelecido nesta mesma etapa. Diversos padr'ões são suportados pelo JMF, apesar de no software exemplo desenvolvido ser utilizado apenas o GSM a titulo ilustrativo. Este parâmetro pode ser trivialmente incorporado expandindo a flexibilidade e customização necessárias ao Framework VoIP.

Abaixo a lista com os padrões suportados com suas principais características:

Format	Content Type	Quality	CPU Requirements	Bandwidth Requirements
Mu-Law	AVI QuickTime WAV RTP	Low	Low	High
ADPCM (DVI, IMA4)	AVI QuickTime WAV RTP	Medium	Medium	Medium
GSM	WAV RTP	Low	Low	Low
G.723.1	WAV RTP	Medium	Medium	Low

Figura 27 - Padrões suportados pelo JMF para codificação da voz

5.3.5. O Output

No processo de *output*, existem dois processos equivalentes aos de *input*. Num primeiro momento, temos a saída dos pacotes RTP, que são transmitidos para a rede. Num outro, acontece a reprodução do sinal de voz recebido e processado no sistema de áudio do dispositivo.

- **Envio de pacotes RTP**

Nesta etapa posterior ao processamento e compactação do sinal de voz, monta-se então um *datasource* que alimenta um *stream* de dados a serem enviados. Eles são empacotados em pacotes RTP e enviados aos nós devidos. Esse *stream* é controlado pelo JMF que continua a se alimentar da saída das etapas anteriores automaticamente após iniciado a transmissão.

- **Reprodução da voz no sistema de áudio**

Com os dados recebidos e decodificados, ocorre o último processo de interesse, que é a reprodução da voz no dispositivo de áudio do sistema. Para isso, um *player* é então criado à partir do *datasource* originário da etapa de processamento para reproduzir continuamente o sinal.

5.3.6. Descrição das classes do JMF

As duas entidades que devem ser criadas para realizar os processos descritos acima são um transmissor e um receptor. As classes que as compõem são pertencentes ao JMF, o qual fornece objetos necessários para realizá-las. Dividiremos então em três grupos funcionais

que contém todas as classes e interfaces utilizadas, sendo eles captura e reprodução de áudio, compressão e descompressão do sinal de voz e transmissão e recepção dos dados.

5.3.6.1. Captura e reprodução de áudio

- **Captura**
  - **AudioFormat:** Encapsula as informações de formato dos dados de áudio. Seus atributos incluem a taxa de amostragem, bits por amostra e número de canais.
  - **CaptureDeviceInfo:** A CaptureDeviceInfo contém informação sobre um dispositivo de captura.
  - **MediaLocator:** Esta descreve a localização do conteúdo de mídia. Cria um caminho da localização da mídia transparente para as classes que podem utilizá-la como entrada.
- **Reprodução**
  - **MediaHandler:** Uma interface para objetos que lêem e manipulam conteudo de mídia provenientes de um *datasource*.
  - **Player:** É um *MediaHandler* para trabalhar e controlar dados variantes no tempo. Ela extende a interface Controller e provê métodos para obter componentes gráficos de interface (Java AWT), controles de processamento e trabalhar outros Controllers.

5.3.6.2. Processamento do sinal de voz

- **Processor:** Define um modulo para processor e controlar dados de mídia variantes no tempo. Extende a classe Player, porém ao invés de processar dados como "caixa preta", ela fornece meios para controlar o processamento.
- **TrackControl:** Interface fornecida pela Processor para consultar, controlar e manipular dados de *tracks* individuais contidos em um lote de mídia.
- **Format:** Abstrai um formato de mídia, sem carregar nenhum parâmetro específico ou informações temporais.

5.3.6.3. Transmissão dos dados

- **RTPManager:** É a classe que cria, controla e manipula uma sessão RTP, enviando e recebendo dados, fornecendo controles dentre outras funcionalidades.
- **ReceiveStreamEvent:** Este evento notifica um receptor de todos os eventos que são recebidos numa particular ReceiveStream. Isso permite que o usuário capture detalhes de todos os *streams* deste tipo enquanto eles transitam através de vários estados. ReceiveStreamEvents podem ser dos tipos: ActiveReceiveStreamEvent,

ApplicationEvent, InactiveReceiveStreamEvent, NewReceiveStreamEvent, PayloadChangeEvent, ReceiveStreamMappedEvent, TimeoutEvent.

- **SessionEvent:** São eventos que pertencem a Session como um todo e não pertencem a ReceiveStream em particular ou a um participante remoto. SessionEvent podem ser do tipo: NewParticipantEvent ou LocalCollisionEvent.

5.4. O Framework

5.4.1. Classes e interfaces internas

As classes internas ao framework e suas interfaces para implementação de aplicativos que nela se baseiam estão listadas abaixo:

5.4.1.1. Processamento e transporte

- **Transmitter:** Classe que utiliza o JMF para realizar a captura, processamento e transmissão da voz.
  - Transmitter(InetAddress p_IpAddress, int p_Port): Construtor do transmissor. Seta o par endereço IP e porta de destino para envio do stream de dados.
  - initMicrophone(): Define a localização do input de voz e inicia a sua captura.
  - createAudioProcessor(): Cria o processador de áudio que irá codificar o sinal de voz adquirido.
  - createAudioManager(): Cria e inicia o manager da sessão de áudio RTP, iniciando também o envio do stream de dados.
- **ITransmitter:** Interface que contém os métodos de chamada do transmissor.
  - startTransmitter(): Inicia a transmissão.
  - stopTransmitter(): Finaliza a transmissão.
- **Receiver:** Classe que utiliza o JMF para realizar o recebimento, decodificação e reprodução da voz. Também é responsável por ouvir eventos de sessão como novo participante, fim de conexão, novo stream, dentre outros.
  - Receiver(InetAddress p_IpAddress, int p_Port) : Construtor do receptor. Seta o par endereço IP e porta de destino nos quais aguardará streams de dados.
  - update(SessionEvent evt): Trata eventos pertencentes a sessão de RTP, ou seja, que pertencem a sessão global e não a um usuário específico. O caso padrão é um novo participante.
  - update(ReceiveStreamEvent evt): Trata eventos de recepção de stream, no caso específico de cada participante.

- **IReceiver:** Interface que contém os métodos de chamada do receptor.
    - startReceiver(): Inicia a recepção.
    - stopReceiver(): Finaliza a recepção.

5.4.1.2. Sinalização

- **SipListener:** Interface proveniente do JAIN SIP como elemento integrador do framework ao JAIN SIP
    - processRequest(RequestEvent requestEvent): evento gerado pelo JAIN SIP informando que chegou uma nova requisição.
    - processResponse(ResponseEvent responseEvent): evento gerado pelo JAIN SIP informando que chegou uma resposta.
    - processTimeout(TimeoutEvent timeoutEvent): evento gerado pelo JAIN SIP informando que ocorreu um timeout de envio de mensagem.
- **Controlador:** Implementação de SipListener
    - usage(): Informa mensagem de erro especificada;
    - shutDown(): Destrói os objetos correntes e encerra a instância do objeto;
    - init(): Realiza os ajustes necessários para a execução do objeto; chamar(): Realiza uma nova chamada;
    - atender(): Atende uma chamada;
    - desligar(): Desliga uma chamada em andamento.
- **MEC:** Máquina de Estados de Conexão – codificação da MEC citada anteriormente no texto
    - usage(): Informa mensagem de erro especificada;
    - processGUI(int comando, Properties properties): processa as informações provenientes da interface gráfica
    - init(): Realiza os ajustes necessários para a execução do objeto;
    - transitarEstado(): Realiza a transição dos estados no autômato;
    - acaoSemantica0() a acaoSemantica12(): Execução das ações atreladas aos estados.
- **ISinalizacao:** Interface de comunicação do framework com a aplicação.
    - chamar(): Realiza uma nova chamada;
    - atender(): Atende uma chamada;
    - desligar(): Desliga uma chamada em andamento.

- **MyRouter:** Especifica o destino padrão das mensagens SIP, isto é, especifica o servidor proxy, caso este exista.

5.4.1.3. GUI

- **IGui**
  - chamando(String remetente): Passa para a camada da aplicação informações a respeito da pessoa que está chamando;
  - conversaIniciada(): Informa a camada de aplicação que uma nova conversa foi iniciada;
  - recebeuDesligar(): Informa que o outro lado da conversa pediu para encerrar a chamada;
  - conversaEncerrada(): Informa que a conversa atual foi encerrada;
  - adicionaNome(String name): Informa o nome do novo participante para a camada de aplicação;
  - removeNome(String name): Informa a remoção do nome da lista de participantes para a camada de aplicação;

5.4.2. Caso de uso P2P

- **AppP2P**
  - initSIP(): inicializa a controladora de estados;
  - atender(): atende uma ligação;
  - chamar(): realiza uma ligação;
  - chamando(String remetente): Informa que o software está recebendo uma ligação;
  - conversaIniciada(): Informa que a conversa foi iniciada com sucesso;
  - recebeuDesligar(): Informa que recebeu um pedido para o computador;
  - encerrarConversa(): Encerra a conversa atual;
  - conversaEncerrada(): Informa que a conversa foi encerrada com sucesso;
  - adicionaNome(String nome): Adiciona um nome ao campo de participantes;
  - removeNome(String nome): Remove o nome da lista de participantes.

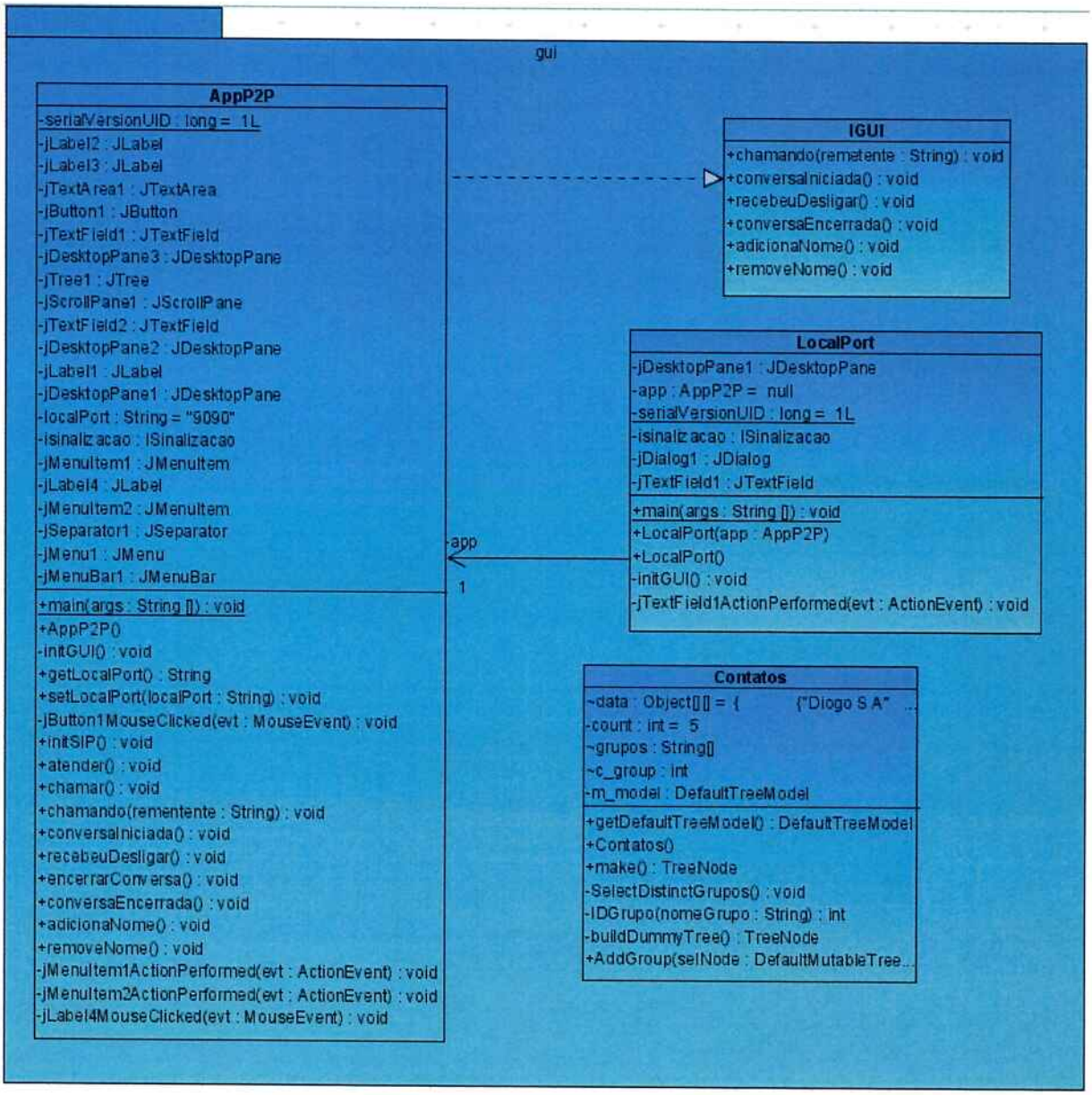

Figura 28 - Diagrama de Classes da aplicação P2P

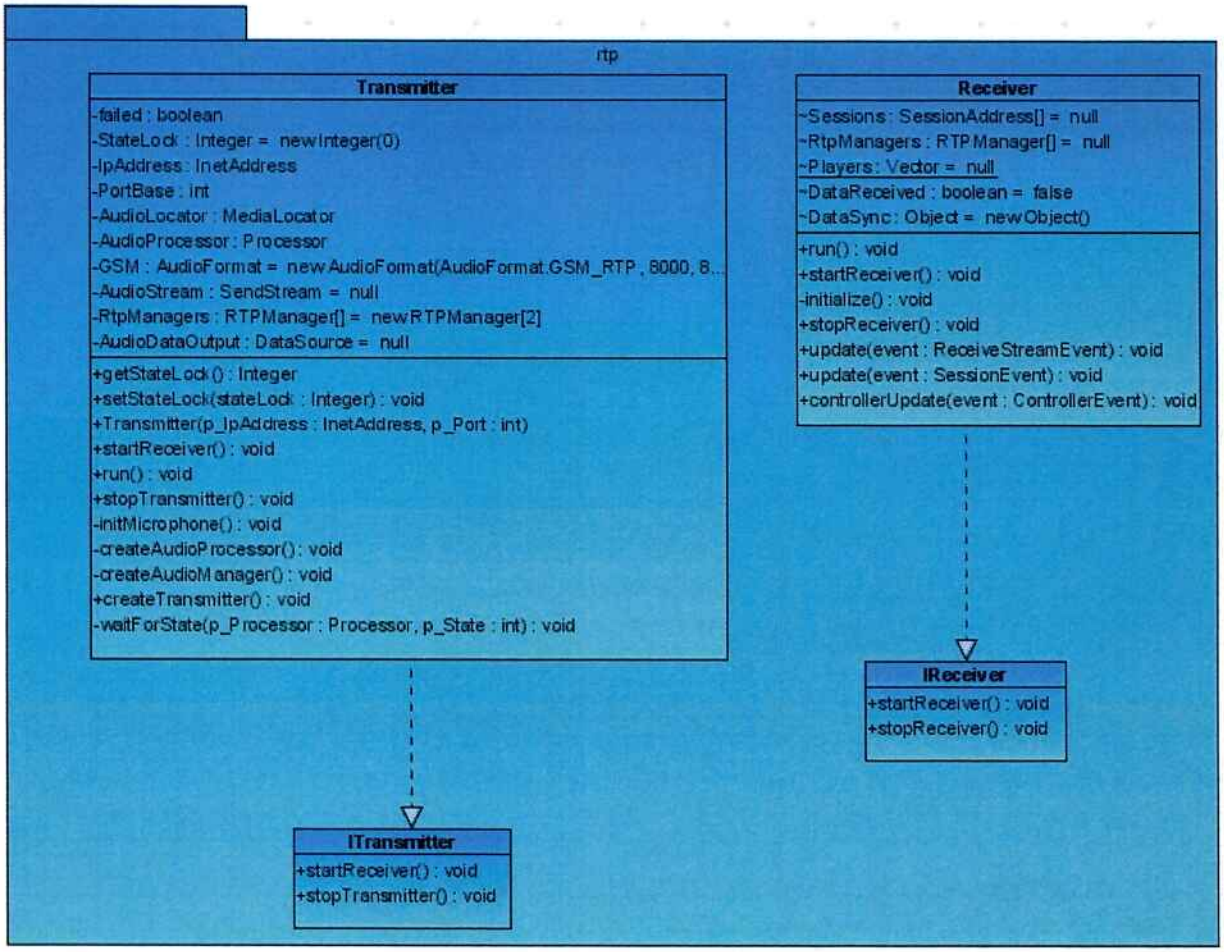

Figura 29 - Diagrama de Classes do RTP

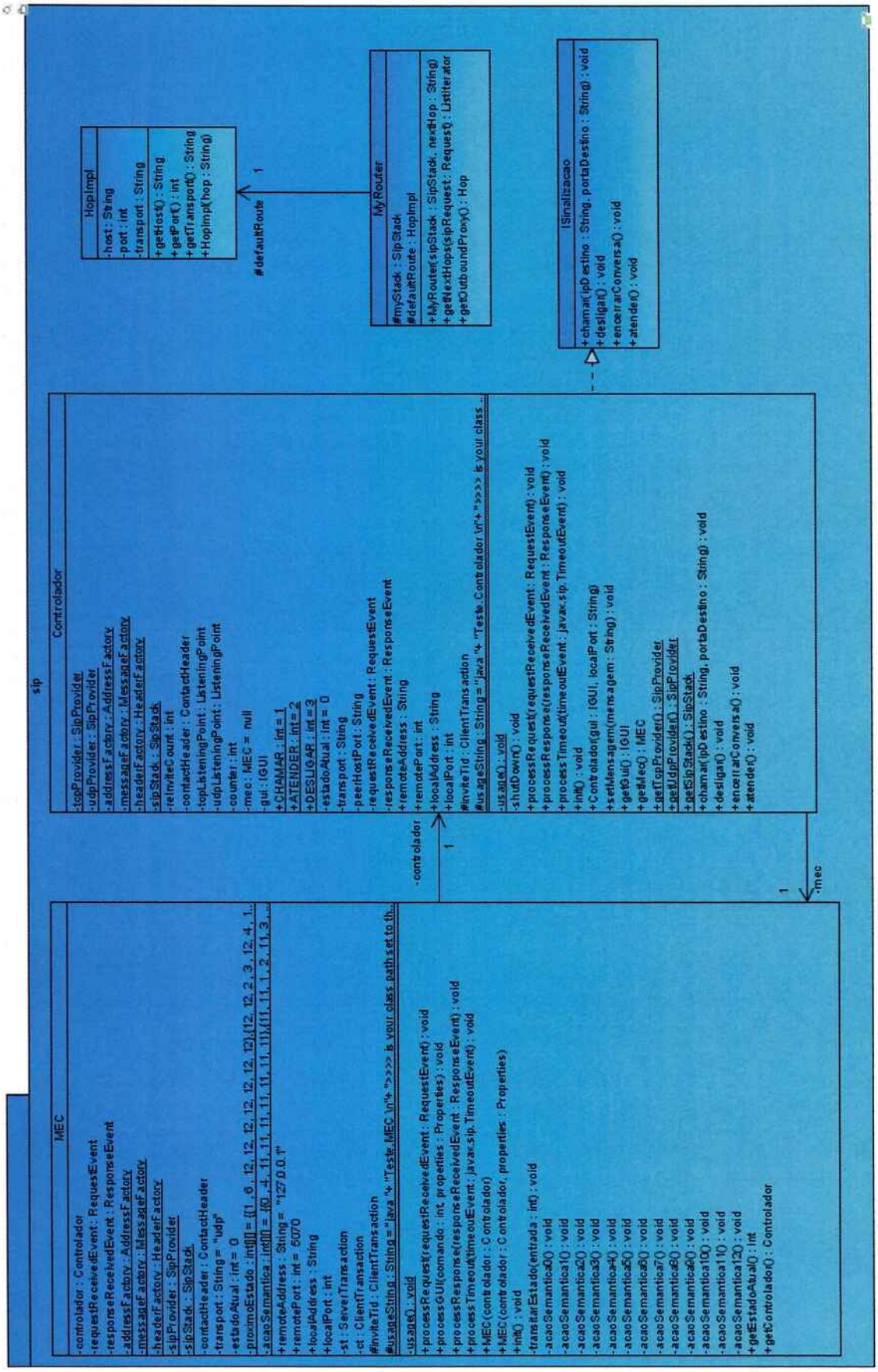

Figura 30 - Diagrama de Classes do SIP

5.5. Resultados esperados

Espera-se que tenhamos uma versão funcional da aplicação de conversa P2P, com sua respectiva modelagem. Além disso, esta aplicação deve estar divida em módulos de modo que seja reconhecível a existência do *framework* modelado, atendendo então aos requisitos funcionais.

Quanto aos requisitos não funcionais, estes são garantidos idealmente pelas implementações já existentes dos protocolos de sinalização, codificação, decodificação e transporte de áudio.

5.6. Resultados alcançados

Neste trabalho obtivemos a modelagem e implementação do *framework* que supre as funcionalidades do caso de uma comunicação entre duas pessoas (P2P).

A fim de validar o *framework*, foi desenvolvida uma aplicação que utiliza o *framework* como infra-estrutura base. A implementação foi realizada com sucesso e será demonstrada na apresentação do projeto.

Desta forma, podemos afirmar que, partindo da premissa que nos orientou, obtivemos exito total no desenvolvimento do projeto, atingindo de maneira extremamente satisfatória nossos objetivos e preparando campo para um futuro desenvolvimento de toda arquitetura idealizada.

6. Testes e Avaliação

6.1. Necessidade de Testes

Com o intuito de assegurar a qualidade do software criado, devem ser realizados testes de funcionamento do sistema de acordo com os casos de uso, pois apontam possíveis falhas e suas prioridades, para que estas sejam avaliadas e corrigidas.

6.2. Levantamento das características a serem testadas – plano de testes

Para testar o software, as seguintes características foram escolhidas:

1. Interface Homem – Máquina: navegação do sistema;
2. Sinalização: capacidade de estabelecer e fechar conexão com uma outra instância do software que está em execução em outro computador;
3. Áudio: codificação e decodificação de sinais de áudio;
4. Transmissão de dados: transmissão e recepção de dados.

6.3. Formas de realização do teste

Para realizar o teste do sistema, as ações listadas nos casos de teste serão executadas conforme especificado e, existirão saídas em tela, em log e em um capturador de pacotes, que vai servir para examinar possíveis problemas no que diz respeito a comunicação entre os dois clientes.

6.4. Resultados dos testes

Teste no. **01**	*Requisito testado:* **Inicialização do sistema**
Funcional: ☐ *Não funcional:* ☒	*Observações:* Este teste se torna interessante a partir do fato de que a inicialização de aplicativos em Java ocorre em uma maneira diferente do usual. Como pré-requisito, se faz necessária a instalação do Java Media Framework (JMF)

Seqüência para verificação do caso comum	*Comportamento esperado do software*	*Testado*
1. Realizar um duplo clique no arquivo .jar que contém o projeto	O sistema deverá inicializar, mostrando sua tela principal	☒

Seqüência para verificação de exceções	*Comportamento esperado do software*	
*1.*Realizar a inicialização sem a instalação prévia do JMF	O sistema deverá emitir mensagem de erro	☒

☒ *Aprovado* ☐ *Reprovado*	*Comentários* Foi exibida uma mensagem de erro padrão Java - exception.

Cientes:

Avaliadores *Apresentador*

Teste no. **02**	*Requisito testado:* **Mudança de designação de porta local**
Funcional: ☒ *Não funcional:* ☐	*Observações:* A porta local padrão do aplicativo é 9090. Para uitilizar uma porta distinta desta, é necessária a execução desta funcionalidade.

Seqüência para verificação do caso comum	*Comportamento esperado do software*	*Testado*
1. Na tela principal, clicar no menu "File" e escolher a opção "Local Port".	O sistema deverá apresentar uma janela com uma caixa de texto para a edição deste atributo.	☒
2. Digitar o novo valor para a porta local e apertar a tecla [Enter] para confirmar.	O sistema irá fechar a janela de edição, atualizando o valor deste atributo.	☒

Seqüência para verificação de exceções	*Comportamento esperado do software*	

	Comentários
☒ *Aprovado*	A alteração de porta local ocorreu sem problemas.
☐ *Reprovado*	

Cientes:	
Avaliadores	*Apresentador*

Teste no. **03**	*Requisito testado:* **Realização de chamada**
Funcional: ☒ *Não funcional:* ☐	*Observações:* Caso de uso principal do aplicativo, descrevendo a realização de uma chamada de voz sobre IP.

Seqüência para verificação do caso comum	*Comportamento esperado do software*	*Testado*
1. Na tela principal, digitar o endereço e porta do destino a ser chamado nas caixas de texto localizadas na parte inferior da tela.	O sistema deverá atualizar os valores de endereço e porta de destino, sem realizar nenhuma operação extra.	☒
2. Acionar o botão "Chamar" para iniciar a chamada.	O sistema irá então inicializar o processo de chamada de uma pessoa para uma conversa.	☒
3. Iniciar a conversa.	O sistema deverá inicializar os módulos de processamento, tratamento e transporte de voz, permitindo então que se realize a conversa a partir do momento que receber um ok por parte do outro lado da conversa em questão.	☒

Seqüência para verificação de exceções	*Comportamento esperado do software*	
1. Tentar realizar uma chamada em uma rede com problemas.	O sistema irá mostrar uma mensagem de erro com uma descrição breve do problema reportado.	☒

	Comentários
☒ *Aprovado* ☐ *Reprovado*	O sistema não transita entra estados até receber respostas do destinatário ou cancelamento do usuário.

Cientes:

Avaliadores *Apresentador*

Teste no. **04**	*Requisito testado:* **Recebimento de chamada**
Funcional: ☒ *Não funcional:* ☐	*Observações:* Caso de uso que descreve o recebimento de uma chamada , atuando junto com o caso de uso de realização de uma chamada.

Seqüência para verificação do caso comum	*Comportamento esperado do software*	*Testado*
1. Na tela principal, clicar no botão relacionado ao atendimento de uma chamada, quando o usuário está recebendo-a.	O sistema deverá inicializar os módulos de processamento, tratamento e transporte de voz, permitindo então que se realize a conversa.	☒

Seqüência para verificação de exceções	*Comportamento esperado do software*	

☒ *Aprovado* ☐ *Reprovado*	*Comentários* O sistema inicializou com sucesso as conversas de teste.

Cientes: *Avaliadores* *Apresentador*

Teste no. **05**	*Requisito testado:* **Encerramento de uma chamada**
Funcional: ☒ *Não funcional:* ☐	*Observações:* Caso de uso que descreve a etapa final de uma chamada.

Seqüência para verificação do caso comum	*Comportamento esperado do software*	*Testado*
1. Na tela principal, clicar no botão relacionado ao encerramento de uma chamada durante a chamada..	O sistema deverá finalizar a comunicação por voz e realizar a troca necessária de mensagens com o outro lado da conversa.	☒

Seqüência para verificação de exceções	*Comportamento esperado do software*

	Comentários
☒ *Aprovado*	O sistema finalizou com sucesso as conversas de teste.
☐ *Reprovado*	

Cientes:

Avaliadores *Apresentador*

Teste no. **06**	*Requisito testado:* **Cancelamento de uma requisição de chamada**
Funcional: ☒ *Não funcional:* ☐	*Observações:* Caso de uso que descreve o cancelamento de uma requisição de chamada, no período de tempo entre a criação da requisição e a inicialização da conversa.

Seqüência para verificação do caso comum	*Comportamento esperado do software*	*Testado*
1. Na tela principal, clicar no botão relacionado ao cancelamento de uma requisição de uma chamada enquanto a conversa não inicia.	O sistema deverá finalizar a requisição de chamada, abortando a comunicação.	☒

Seqüência para verificação de exceções	*Comportamento esperado do software*	

	Comentários
☒ *Aprovado*	O sistema finalizou com sucesso as requisições de conversas de teste.
☐ *Reprovado*	

Cientes:

Avaliadores *Apresentador*

Teste no. **07**	*Requisito testado:* **Alteração dos parâmetros da conversa**
Funcional: ☒ *Não funcional:* ☐	*Observações:* Caso de uso que descreve a configuração dos parâmetros que coordenam o processamento dos sinais de voz

Seqüência para verificação do caso comum	*Comportamento esperado do software*	*Testado*
1. Na tela principal, clicar no menu "File" e depois "Rtp Options"	O sistema deverá mostrar uma janela contendo as opções de ajuste dos parâmetros de configuração de áudio.	☒
2. Alterar os parâmetros disponíveis.	O usuário deve ajustar os parâmetros, como bitrate, algoritmo de compactação, taxa de amostragem.	☒
3. Salvar as alterações.	O usuário deve clicar no botão "Sair" para salvar as alterações	☒

Seqüência para verificação de exceções	*Comportamento esperado do software*	

	Comentários
☒ *Aprovado*	O sistema configurou com sucesso as opções disponíveis.
☐ *Reprovado*	

Cientes:	
Avaliadores	*Apresentador*

7. Considerações Finais

As impressões finais que ficam a respeito do trabalho são bem positivas. Primeiramente, podemos assumir que o trabalho foi realizado com sucesso, o que implica em dizer que o objetivo especificado foi cumprido. Apesar de alguns problemas que acabaram ocorrendo, uns por inexperiência, outros por forças maiores, conseguimos suplantar estas dificuldades, realizando um projeto com seriedade, sempre buscando o melhor resultado.

Obtivemos também uma visão ampla do mercado de voz sobre IP e como ele está imerso no novo paradigma de comunicação que se desenvolve no mundo. Observar como esta tecnologia vem se desenvolvendo e ganhando papel importante no processo de modernização das tendências de comunicação nos coloca em uma posição privilegiada tanto como produtores quanto como consumidores de tecnologia.

O conhecimento desta tecnologia proveniente do estudo realizado no projeto permitirá nos inserir neste mercado que vêm cada vez mais criando oportunidades e oferece uma vasta gama de desafios a serem vencidos.

Nossa proposta apresentada neste trabalho, quando foi idealizada, era tida como inédita em sua natureza. Hoje, grandes empresas, tais como Microsoft, vêm investindo neste tipo de solução nos mostrando que temos capacidade de não só atuar passivamente como profissionais, mas também ativamente, lançando tendências a partir de um estudo bem realizado.

Finalmente, terminamos o projeto com um balanço positivo e com uma visão de projeto melhor desenvolvida, atuando desde a concepção, realização de estudos de viabilidade, gerência de projeto e recursos, escolha de tecnologias e desenvolvimento do produto, complementando a formação acadêmica a qual estamos concluindo com este trabalho.

8. Bibliografia

[1] Jeszensky, Paul Jean Etienne, *Sistemas Telefônicos*, Manole, 2004.

[2] Fujimoto, Rogério Takashi, Análise de QoS para aplicações de VoIP em rede de nova geração, PTC-EPUSP, 2004.

[3] Zhang Y., *SIP-based VoIP network and its interworking with the PSTN*, IEEE Electronics and Communication Engineering Journal, December 2002

[4] Singh, A., Mahadevan P., Acharya A., Shae Z., *Design and Implementation of SIP Network and Client Services*, IEEE, 2003.

[5] Glasmann J., Kellerer W., Muller H., *Service Development and Deployment in H.323 and SIP*, IEEE, 2001.

[6] RFC 3550 – "*RTP: A Transport Protocol for Real-Time Applications*", 2003

[7] *Java Media Framework – API Guide*, Sun Microsystems, 1999.

[8] Sun Microsystems – http://www.sun.com/.

[9]NIST SIP JavaDoc – http://www-x.antd.nist.gov/proj/iptel/src/nist-sip/jain-sip/docs/api/overview-summary.html

[10] RFC 3261 – "Sip: Session Initiation Protocol", 2002.