UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO – BACHARELADO

PROTÓTIPO DE FERRAMENTA PARA MAXIMIZAÇÃO DA QUALIDADE DE SERVIÇO NO TRÁFEGO DE *STREAMING* DE ÁUDIO EM TEMPO REAL

ADILSON HASCKEL

BLUMENAU
2005

2005/2-01

ADILSON HASCKEL

PROTÓTIPO DE FERRAMENTA PARA MAXIMIZAÇÃO DA QUALIDADE DE SERVIÇO NO TRÁFEGO DE *STREAMING* DE ÁUDIO EM TEMPO REAL

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciências da Computação — Bacharelado.

Prof. Francisco Adell Péricas, Ms – Orientador

BLUMENAU
2005

2005/2-01

PROTÓTIPO DE FERRAMENTA PARA MAXIMIZAÇÃO DA QUALIDADE DE SERVIÇO NO TRÁFEGO DE *STREAMING* DE ÁUDIO EM TEMPO REAL

Por

ADILSON HASCKEL

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Francisco Adell Péricas, Ms – Orientador, FURB
Membro:	Prof. Paulo Fernando da Silva, Ms – FURB
Membro:	Prof. Sérgio Stringari, Ms – FURB

Blumenau, 07 de dezembro de 2005

Dedico este trabalho a minha família e amigos que sempre me apoiaram nos momentos decisivos da graduação.

AGRADECIMENTOS

Aos meus pais, Arlindo e Albertina e meu irmão Alex pelo apoio nos momentos mais difíceis.

Aos meus amigos de graduação pela parceria ao longo destes últimos anos, principalmente ao André Luís Garlini.

Ao meu orientador, Francisco Adell Péricas, pelo precioso auxílio no desenvolvimento deste trabalho.

A vida não é uma série de imagens que mudam à medida que se repetem.

Andy Warhol

RESUMO

O presente trabalho descreve o desenvolvimento de um protótipo de ferramenta que visa fornecer qualidade de serviço no tráfego de *streaming* de áudio em tempo real, ajustando-se às condições da rede, permitindo garantir os requisitos mínimos de qualidade de serviço definidos pela aplicação. Apresenta-se a especificação e implementação do protótipo desenvolvido, que utiliza para o transporte multimídia o protocolo *Real-time Transport Protocol* (RTP) e para as ações de controle da transmissão o *Real-time Control Protocol* (RTCP).

Palavras-chave: *Streaming* multimídia. Qualidade de serviço. Tempo real. RTP. RTCP.

ABSTRACT

The present work describes the development of a tool archetype that aims to supply the quality of service in the traffic of streaming audio in real time, adjusting the its conditions to the net, guaranteeing the minimum requirements of quality of service defined by the application. It is presented the specification and implementation of the developed archetype, that uses for the multimedia transport the Real-time Transport Protocol (RTP) and for the actions of control transmision the Real-time Control Protocol (RTCP).

Key words: Streaming multimedia. Quality of service. Real time. RTP. RTCP

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

APP - *Application-defined*

CC - *Contributing source identifiers Count*

CD - *Compact Disc*

CNAME - *Canonical Name*

DLSR - *Delay since Last Sender Report*

FPS - *Frames Per Second*

FURB - Fundação Universidade Regional de Blumenau

GSM - *General Special Mobile*

HDTV - *High-definition Television*

HTTP - *Hypertext Transfer Protocol*

IP - *Internet Protocol*

ISDN - *Integrated Services Digital Network*

ISO - *International Organization for Standardization*

ITU-T - *International Telecommunications Union*

J2SE - *Java 2 Standard Edition*

JMF - *Java Media Framework*

JPEG - *Joint Photographic Experts Group*

Kbps - *Kilobits per second*

KHz - *Kilohertz*

LSR - *Last Sender Report timestamp*

MB - *Megabytes*

Mbps - *Megabits per second*

MJPEG - *Motion Joint Photographic Experts Group*

MP3 - *Moving Pictures Expert Group Layer 3*

MPEG - *Moving Pictures Expert Group*

NTP - *Network Time Protocol*

PCM - *Pulse Code Modulation*

PT - *Payload Type*

QoS - *Quality of Service*

RF - Requisito Funcional

RFC - *Request For Comments*

RNF - Requisito Não Funcional

RR - *Receiver Report*

RSVP - *Resource Reservation Protocol*

RTP - *Real-time Transport Protocol*

RTCP - *Real-time Control Protocol*

RTSP - *Real-time Streaming Protocol*

SDES - *Source Description*

SIF - *Standard Interchange Format*

SR - *Sender Report*

SSRC - *syncronization source identifier*

TCP - *Transmission Control Protocol*

UDP - *User Datagram Protocol*

UML - *Unified Modeling Language*

VCR - *Video Cassete Recorder*

SUMÁRIO

1 INTRODUÇÃO

As aplicações multimídia obtiveram uma posição de destaque no atual cenário das redes de computadores, principalmente na internet. A tecnologia de mídia *streaming*, ou seja, a transmissão de fluxos contínuos de áudio e vídeo proporcionou novas formas de comunicação e de entretenimento em tempo real. Entre estas se destacam as aplicações de audioconferência, videoconferência e de transferência de mídia armazenada em tempo real.

Com o surgimento das aplicações multimídia observou-se a necessidade de rever a arquitetura de rede existente, devido às características especiais destas aplicações, que são muito diferentes das aplicações tradicionais da internet. Conforme Melo (2001, p. 1), rigidez no atraso fim a fim (origem e destino da transmissão) e tolerância a perdas mínimas de pacotes na transmissão são as principais características das aplicações multimídia.

Para suprir as necessidades destas aplicações, foram criados protocolos especiais para a transmissão multimídia. Estes protocolos possuem informação de tempo, essencial para aplicações de mídia *streaming*. Um exemplo desta classe de protocolos é o RTP.

Segundo Tschöke (2001, p. 23), o RTP não fornece reserva de recursos da rede e nem fornece garantia na qualidade de serviço da transmissão em tempo real, apenas o monitoramento dos dados entregues. Esta falta de qualidade de serviço é um sério problema na transmissão multimídia. Excesso de perdas de pacotes, atrasos e congestionamento da rede durante a transmissão são fatores que poderão afetar a integridade da mídia recepcionada por uma aplicação cliente.

Tendo em vista as questões apresentadas, este trabalho propõe a criação de uma ferramenta que vise maximizar a qualidade de serviço no tráfego de *streaming* de áudio em tempo real. A ferramenta adaptará a taxa de transmissão do fluxo de áudio, baseando-se nas condições da rede, que variam em função de perda de pacotes, atraso na transmissão e a

variação deste atraso. Dois aplicativos, um servidor e outro cliente, irão compor a ferramenta. O aplicativo servidor será responsável pela transmissão da mídia previamente armazenada e pelas funções de monitoração e adaptação. O cliente efetuará a solicitação da mídia para apresentação. O protocolo RTP será utilizado para o transporte fim-a-fim das mídias e para o levantamento da qualidade de serviço de recepção do cliente será utilizado o protocolo RTCP.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é desenvolver um protótipo de ferramenta que visa maximizar a qualidade de serviço no tráfego de *streaming* de áudio em tempo real, gerado por um aplicativo que será executado em um servidor.

Os objetivos específicos do trabalho são:

a) possibilitar a transmissão da mídia solicitada para visualização em um aplicativo cliente;
b) efetuar o monitoramento dos pacotes de controle do protocolo RTP para determinar o estado da rede através de seus parâmetros;
c) permitir a adaptação da taxa de transmissão do fluxo de áudio conforme os parâmetros da rede;
d) utilizar o protocolo RTP para a transmissão fim-a-fim do *streaming* multimídia.

1.2 ESTRUTURA DO TRABALHO

O trabalho está organizado da seguinte forma:

a) O primeiro capítulo apresenta uma introdução sobre o assunto e os objetivos do trabalho, a fim de fornecer ao leitor as informações necessárias para o

entendimento do assunto e do projeto a ser desenvolvido;

b) O segundo capítulo apresenta alguns conceitos, técnicas e protocolos que fazem parte da fundamentação teórica do trabalho;
c) O terceiro capítulo descreve o desenvolvimento do trabalho, abordando questões referentes à especificação e implementação do projeto;
d) O quarto e último capítulo são reservados para as conclusões do trabalho e para a algumas sugestões de trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentados conceitos, técnicas, protocolos e trabalhos correlatos relacionados ao projeto desenvolvido.

2.1 APLICAÇÕES MULTIMÍDIA NA ARQUITETURA INTERNET

As aplicações multimídia são constituídas da integração de diferentes tipos de mídias, como som, vídeo e texto. A transferência destes dados na internet pode ser classificada como em tempo real e não-real, e a questão temporal está relacionada com o momento em que a mídia é apresentada no receptor da aplicação (NAUGLE, 2001, p. 300).

Tempo não-real é a transferência de dados sem considerar o fator tempo, ou seja, os dados são inteiramente entregues para o receptor e podem ser exibidos numa data posterior. Alguns exemplos de tempo não-real são as aplicações tradicionais da internet, como o *email* e os navegadores *web*.

Ao contrário das aplicações de tempo não-real, o tempo é fundamental para a semântica das aplicações de tempo real. Segundo Naugle (2001, p. 300), tempo real vem a ser a capacidade de visualizar (ver ou ouvir) a mídia à medida que os dados são transferidos, por meio de *download*, da estação de rede provedora do serviço até a estação receptora dos dados.

Alguns exemplos de tempo real são: a visualização de um clipe de vídeo à medida que este é transferido; a distribuição dos dados capturados por uma câmera ao vivo em um discurso pelos servidores de vídeo a outras estações para exibição imediata; e as vídeo-conferências onde diversos participantes em localidades distintas podem interagir entre si.

As aplicações multimídia possuem duas características bem definidas que se contrapõem às aplicações tradicionais da internet. Conforme Kurose e Ross (2003, p. 378), as

aplicações multimídia são altamente sensíveis ao atraso e tolerantes a perdas ocasionais de pacotes na transmissão. Estas perdas causam apenas pequenas variações na recepção do áudio e vídeo, que podem ser parcialmente ou totalmente ocultas. Por outro lado, nas aplicações tradicionais, longos atrasos são incômodos, porém não prejudiciais ao resultado final, já a perda de pacotes afeta a integridade dos dados.

2.1.1 Classes de aplicações multimídia

Segundo Péricas (2003, p. 112), as aplicações multimídia são divididas em três classes: *streaming* de áudio e vídeo armazenado, *streaming* de áudio e vídeo em tempo real e *streaming* de áudio e vídeo interativo em tempo real.

Na classe de *streaming* de áudio e vídeo armazenado, os arquivos de mídia estão previamente armazenados em aplicações servidoras. Estas aplicações possuem duas funções básicas que são receber as requisições de solicitação de arquivos das aplicações clientes e efetivamente enviar o arquivo solicitado para visualização dos clientes à medida que os dados forem recebidos. Os clientes podem controlar a mídia com as funções de avançar, retroceder, pausar e navegar por meio do protocolo *Real-time Streaming Protocol* (RTSP).

A classe de *streaming* de áudio e vídeo em tempo real tem basicamente a mesma idéia das transmissões de rádio e televisão convencionais, só que utilizando a internet como meio de difusão. Os usuários podem receber a programação ao vivo das transmissões, independente da sua localização geográfica.

Na classe das aplicações de *streaming* de áudio e vídeo interativo em tempo real, os usuários podem comunicar-se uns com os outros em tempo real. A interação entre os usuários pode ocorrer por meio de áudio, denominado de telefonia da internet, ou por vídeo, através da videoconferência e videofonia da internet.

2.1.2 Compressão de áudio e vídeo

Para possibilitar a transmissão de áudio e vídeo em uma rede de computadores é necessário que estes sejam digitalizados e comprimidos. A digitalização é uma necessidade primária, pois é necessário converter os sinais analógicos em seqüência de *bits*, que é a forma como os computadores transmitem a informação. A compressão multimídia é importante, devido à redundância oriunda dos sinais digitalizados de áudio e vídeo. Estas redundâncias tornam a mídia inviável para a transmissão multimídia, pois consomem uma significativa capacidade de armazenamento e de largura de banda.

Goulart (1998, p. 11) ilustra um exemplo de como a compressão pode reduzir significativamente o tamanho de um arquivo multimídia. Seja um vídeo (áudio associado), com duração de 15,4 segundos, com os seguintes atributos: áudio capturado a 22,05 *kilohertz* (KHz) canal mono e 8 *bits* por amostra; vídeo com tamanho de quadro (*frame*) de 320X240 *pixels*, capturado a uma taxa de 25 *frames per second* (fps) e utilizando 24 *bits* para cores. O arquivo gerado sem compressão com a configuração descrita anteriormente chega a um tamanho de 89,4 *megabytes* (MB). Utilizando técnicas de compressão, o tamanho do arquivo pode ser reduzido para apenas 3 MB, que é um valor condizente com as atuais tecnologias de armazenamento e os *links* de transmissão.

Para áudio, a técnica básica de codificação digital é o *Pulse Code Modulation* (PCM), utilizada principalmente em sistemas telefônicos. A freqüência de amostragem adotada é de 8.000 amostras da amplitude do sinal por segundo. Estas amostragens são quantizadas em 256 níveis, sendo utilizado 8 *bits* para representação. O PCM necessita de um canal de 64 *kilobits per second* (Kbps) para transmissão do sinal digitalizado, já que por segundo são geradas 8.000 amostras de 8 *bits*.

A codificação PCM é apenas uma técnica de digitalização de áudio, a qual não

costuma ser utilizada em transmissões multimídia na internet. Para este fim, existem outros inúmeros formatos que utilizam técnicas de compressão de voz, entre estes se destacam: *General Special Mobile* (GSM), G.729, G.723.3 e o *Moving Pictures Expert Group* (MPEG) nível 3, mais conhecido como MPEG *audio layer* 3 (MP3).

Segundo Kurose e Ross (2003, p. 383), o MP3 utiliza um padrão de compressão complexo, com máscara psicoacústica, redução de redundância e *buffer* com reservatório de *bits*. A compressão do áudio pode ser a uma taxa variável de *bits*, sendo 128 Kbps a taxa mais utilizada, causando pouquíssima degradação ao som. A qualidade obtida é comparável à música estéreo de um *Compact Disc* (CD) e pode ser fragmentado em pedaços reproduzíveis de música, possibilitando a transmissão *streaming* pela internet.

A necessidade de compressão do vídeo é maior que a do áudio, pois trata de uma seqüência de imagens tipicamente capturadas a taxas de 24 ou 30 quadros por segundo. Entre as técnicas de compressão de vídeo, as mais difundidas são as codificações MPEG, *Joint Photographic Experts Group* (JPEG) e H.261.

Segundo Goulart (1998, p.16) o MPEG possui vários níveis de codificações para vídeo que foram definidos pelo grupo *International Organization for Standardization* (ISO). Estes níveis são os seguintes:

a) MPEG-1: destinado ao armazenamento de seqüência de áudio sincronizado, com qualidade de *Video Cassete Recorder* (VCR) sobre CD. Opera a taxa de 1,2 *megabits per second* (Mbps) com imagem *Standard Interchange Format* (SIF). A razão de compressão é da ordem de 26:1;
b) MPEG-2: é definido para a televisão com qualidade de estúdio e canais de áudio com qualidade de CD com taxa de 4 a 6 Mbps. Pode manipular resoluções que chegam a 720 X 480 *pixels* para sinal de luminância e 360X480 *pixels* para o espaço da cor;

c) MPEG-3: codificação e compressão para *High-definition Television* (HDTV). Entretanto, o MPEG-2 conseguiu absorver esta tecnologia e o resultado disto, foi o cancelamento deste nível;

d) MPEG-4: é um padrão muito robusto para videoconferência, opera a baixas taxas de transmissão que variam de 4,8 a 64 Kbps. A videofonia e a videoconferência são seus principais campos de atuação.

O padrão JPEG, a princípio, foi desenvolvido para a codificação de imagens estáticass. Contudo, utilizando-se de uma abordagem denominada *Motion* JPEG (MJPEG), através do emprego de um *hardware* especializado, é possível codificar e decodificar uma série de imagens em tempo real realizando assim vídeo em movimento. O MJPEG, entretanto não é considerado um padrão para vídeo.

O H.261 (recomendação *International Telecommunications Union* (ITU-T) H.261) consiste em um padrão de codificação e compressão de vídeo, que inicialmente foi projetado para a utilização em videoconferência sob redes *Integrated Services Digital Network* (ISDN). A largura de banda de comunicação consumida pode variar entre 64 Kbps e 2 Mbps, medidas em intervalos de 64 Kbps. Esta técnica é também conhecida como “px64” onde "p” pode variar de 1 até 30.

Um resumo das técnicas de compressão de vídeo MPEG, JPEG e H.261 é apresentado no Quadro 1. É interessante observar que os três padrões foram concebidos por grupos e objetivos distintos.

Codinome	Designação	Grupo de padrões	Razão de compressão
JPEG	Compressão digital e codificação de imagens paradas com tonalidade contínua	Grupo de especialista em fotografias	15:1 (aplicações com quadros parados cheios)
H.261, px64	Codificador/decodificador para serviços audiovisuais no px 64 Kbps	Grupo especialista em codificação para transmissão visual sobre linhas telefônicas	100:1 até 2000:1 (telecomunicações baseadas em vídeo)
MPEG	Codificação de figuras em movimento e áudio associado	Grupo especialista em figuras em movimento	200:1 (aplicações intensas em movimento)

Fonte: adaptado de Goulart (1998, p. 17)

Quadro 1 – Padrões de compressão multimídia

2.2 TRANSMISSÃO DE *STREAMING* MULTIMÍDIA

Em aplicações de rede multimídia que disponibilizam áudio e vídeo sob a forma de *streaming*, o servidor aguarda por requisições das aplicações clientes. O formato da requisição varia conforme o ambiente onde estiver sendo executada a aplicação: no caso da navegação *web*, é efetuada uma solicitação *get* do protocolo de aplicação *Hypertext Transfer Protocol* (HTTP). Em um ambiente cliente/servidor, a requisição é feita através de mensagem via um canal de comunicação denominado soquete.

Após receber a solicitação do cliente, o servidor segmenta o *streaming* para ser enviado à rede. O transporte destes segmentos é normalmente realizado por um protocolo especial para tráfego multimídia, o RTP.

Conforme Tanenbaum (2003, p. 724), o uso do popular protocolo de tranporte *Transmission Control Protocol* (TCP) não é recomendado para as aplicações multimídia em tempo real, pois se ocorrer algum erro, o último pacote é retransmitido, introduzindo um intervalo inaceitável na reprodução do áudio ou vídeo. Por este motivo, o RTP normalmente é disposto em camadas sobre o protocolo de transporte *User Datagram Protocol* (UDP), que

não efetua a correção de erros e conseqüentemente a retransmissão de pacotes.

Depois de ser transportado pela rede, cada porção de dados é recebida na aplicação cliente (reprodutor de mídia), armazenada em *buffer* e encaminhada para a reprodução. O uso de um *buffer* é necessário para efetuar a sincronização temporal da mídia. Um reprodutor de mídia basicamente tem a incumbência de administrar uma interface gráfica com o usuário, tratar erros de transmissão, descompressão da mídia e eliminar flutuação temporal.

A interface gráfica de usuário consiste na área de visualização da mídia, onde através de controles há uma interação com o usuário. Estes controles em geral englobam comandos de volume, pausa/reinício e avanço temporal. Alguns reprodutores ainda oferecem recursos especiais, através de *plug-ins* como a integração com *browsers* e a personalização da interface gráfica com a incorporação dos denominados *skins.*

Os erros de transmissão, geralmente vinculados à perda de pacotes por congestionamentos da rede, podem ser recuperados pelos reprodutores de mídia de diversas formas. As práticas mais adotadas são: reconstrução dos pacotes perdidos por meio da transmissão de pacotes redundantes, solicitação explícita do cliente para retransmissão de pacotes perdidos e interpolação de dados que faltam receber nas mensagens recebidas.

A descompressão da mídia é necessária durante a sua reprodução, pois o áudio ou vídeo quase sempre é comprimido na transmissão para economizar espaço de armazenamento e largura de banda da rede.

Outra função do reprodutor de mídia, a eliminação da flutuação temporal ou variação do atraso (*jitter*), visa eliminar a variação do tempo que ocorre entre a origem e o destino da transmissão de dados pela rede. Como o áudio ou vídeo necessitam ser reproduzidos a uma taxa constante (taxa de codificação), o *jitter* deve ser removido e compensado. Isto é possível através do armazenamento local de pequena quantidade de pacotes antes de serem reproduzidos.

Atualmente os reprodutores de mídia *streaming* mais conhecidos e utilizados são o *Real One Player*, o Winamp e o *Windows Media Player*, não necessariamente nesta ordem.

2.3 PROTOCOLO RTP

O protocolo RTP é o padrão de mercado para o transporte de fluxo multimídia e foi definido originalmente na *Request for Comments* (RFC) 1889 juntamente com o protocolo de controle RTCP. Held (2000, p. 75, tradução nossa) afirma: “O RTP foi desenvolvido para ser um serviço de entrega fim a fim de dados com características de tempo real, como o áudio e o vídeo interativo”.

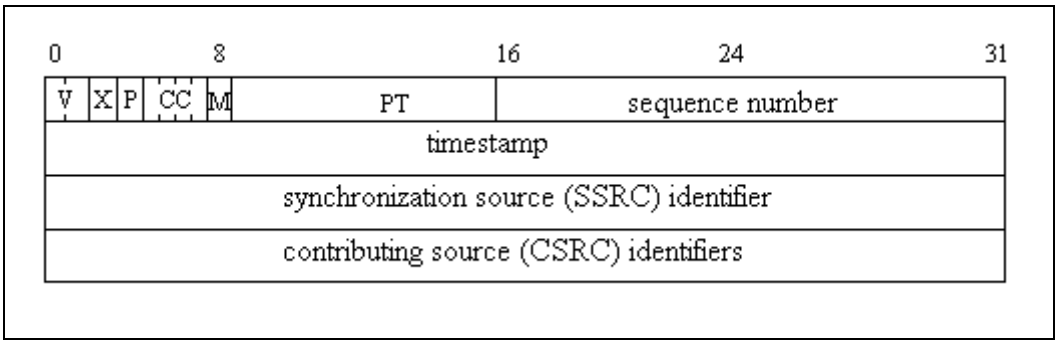
O RTP opera tanto em redes *multicast* quanto em redes *unicast*. Para a utilização em redes *multicast* (típico cenário de videoconferência), é necessário que haja suporte do *backbone*.

Para efetuar o transporte multimídia de forma satisfatória, o RTP disponibiliza para seus pacotes marcação de tempo, numeração de seqüência de pacotes, identificação de fonte de sincronização, monitoramento de entrega (através do RTCP) e identificação do tipo de codificação de áudio ou vídeo transportado. A identificação de sincronização é necessária, pois quando se transporta um vídeo, por exemplo, é necessário manter uma sincronização temporal entre os fluxos de áudio e vídeo, objetivando manter a harmonia da mídia.

A posição do RTP na pilha de protocolos é curiosa, pois ele está diretamente associado ao programa aplicativo (camada de aplicação), mas sendo independente das aplicações, fornecendo apenas recursos de transporte (camada de transporte). Tendo em vista esta estrutura, Tanenbaum (2003, p. 563) afirma: “Talvez a melhor descrição do RTP seja como um protocolo de transporte implementado na camada de aplicação”.

2.3.1 Formato do cabeçalho RTP

Segundo Costa (2004, p. 8), um pacote RTP é um pacote de dados que possui um cabeçalho fixo, uma lista de fontes de contribuição e um *payload* (codificação da mídia). Um pacote da camada de transporte que encapsula o RTP (normalmente o UDP) pode conter um ou mais pacotes RTP, dependendo da forma de encapsulamento empregada. O formato do cabeçalho RTP é apresentado na Figura 1. Os primeiro doze bytes estão sempre presentes, enquanto os identificadores de fontes de contribuição são usados em aplicações específicas.



Fonte: adaptada de Liesenborgs (2000)

Figura 1 – Formato do cabeçalho RTP

Os dois primeiros *bits* do cabeçalho (V) são reservados para o identificador da versão do protocolo, atualmente na versão 2. Com apenas 2 *bits* para a versão do protocolo, supostamente o RTP não suportaria a representação de todas as suas futuras versões. Considerando este problema, Peterson e Dave (2004, p. 328), afirmam que caso seja necessário uma nova versão do protocolo poderia se considerar uma mudança no formato do cabeçalho. Por exemplo, um novo cabeçalho RTP com valor 3 no campo de versão poderia ter um campo representativo de “subversão” em outro lugar no cabeçalho.

O próximo *bit* (P) é o *bit* de *padding* que caso estiver preenchido, há um ou mais *bytes* de preenchimento no final do pacote. O preenchimento com bytes adicionais pode ser necessário, caso algum mecanismo de criptografia esteja sendo empregado e que necessite de

um tamanho mínimo de pacote. A quantidade de bytes de preenchimento a serem ignorados na transmissão é indicada no último byte de preenchimento.

O *bit* de extensão (X) é utilizado para indicar se há a presença de um cabeçalho de extensão procedendo ao cabeçalho fixo do RTP. A definição deste cabeçalho adicional ficaria a cargo de uma aplicação específica que definiria as informações relevantes à implementação em questão.

O campo *Contributing source identifiers Count* (CC) é composto por 4 *bits* e consiste num contador de origens de contribuição presentes após o cabeçalho fixo. Este conceito é apresentado na explicação do campo c*ontributing source identifiers* (CSRC). Após o campo CC é definido um *bit* de marcação (M) específico da aplicação. Segundo Tanenbaum (2003, p. 565), este *bit* pode ser utilizado para marcar o início de um quadro de vídeo, o começo de uma palavra em um canal de áudio ou ainda qualquer outro elemento significativo para a aplicação.

Os próximos 7 *bits* do cabeçalho são reservados para o campo *payload type* (PT) que consiste em um identificador do formato contido no pacote RTP. Segundo Costa (2004, p.9), um perfil informa um mapeamento estático padrão entre códigos de tipos de *payload* e os formatos de codificação (*codecs*) da mídia. Como todo pacote RTP apresenta este campo, a aplicação poderá alterar a codificação da mídia durante a transmissão, com base em informações sobre a qualidade da aplicação e o estado da rede. A Tabela 1 apresenta alguns dos formatos de mídia suportados no perfil padrão do protocolo RTP.

Tabela 1 – Alguns tipos de mídia suportados pelo perfil padrão do RTP

Código do Tipo	**Áudio/Vídeo (A/V)**	**Formato**	**Taxa de amostragem (KHz)**	**Vazão de banda de rede (Kbps)**
0	A	PCM	8	64
1	A	1016	8	4,8
3	A	GSM	8	13
7	A	LPC	8	2,4
9	A	G.722	8	48-64
14	A	áudio MPEG	90	-
15	A	G.728	8	16
26	V	*Motion* JPEG	90	-
31	V	H.261	90	-
32	V	vídeo MPEG 1	90	-
33	V	vídeo MPEG 2	90	-

Fonte: adaptado de Kurose e Ross (2003, p. 401)

Após o *payload type*, o cabeçalho RTP reserva 16 *bits* para o campo *sequence number* (número de seqüência). Este campo é incrementado em uma unidade a cada pacote RTP enviado pela aplicação e pode ser utilizado pelo receptor para determinar perdas de pacotes ao longo de uma transmissão, tornando possível a ocultação de dados perdidos.

O campo *timestamp* (marca de tempo), possui um comprimento de 32 *bits*. Ele marca o instante de amostragem do primeiro byte do pacote RTP. As marcas de tempo podem ser utilizadas pelo receptor para remover a variação do atraso introduzida pela rede e estabelecer uma recepção sincronizada da mídia. A marca de tempo tem origem num relógio de amostragem no remetente.

Kurose e Ross (2003, p. 402) explicam através de um exemplo como é determinada a marcação de tempo: para áudio, o relógio de marcação de tempo é incrementado em uma unidade para cada período de amostragem (por exemplo, a cada 125 milissegundos para um relógio de amostragem de 8 KHz). Desta maneira se a aplicação gerar porções de 160 amostras codificadas, então a marca de tempo aumentará em 160 para cada pacote RTP, enquanto a fonte estiver ativa. O relógio de marca de tempo segue aumentado a uma taxa constante, mesmo que a fonte encontre-se inativa.

O cabeçalho fixo do RTP termina com 32 *bits* destinados ao campo *syncronization source identifier* (SSRC). Este campo identifica a fonte de sincronização responsável pelo fluxo que está sendo transmitido. De modo geral, cada fonte de uma sessão RTP possui um SSRC distinto, que é gerado de maneira aleatória. Um exemplo disto é a transmissão de um vídeo com áudio associado, ou seja, dois canais, onde cada fluxo terá o seu SSRC.

O cabeçalho RTP pode ainda conter 16 itens de 32 *bits* cada, referentes ao campo *contributing source identifiers* (CSRC). Este campo indica as fontes participantes de uma mídia "mixada". Este tipo de mídia consiste na junção de fluxos de diferentes fontes de sincronização.

2.3.2 Sessão RTP

Segundo Costa (2004, p. 12), uma sessão RTP consiste em uma associação entre um conjunto de participantes comunicando-se via RTP. É identificada por um par de endereços de transporte, cada um com um endereço de rede e uma porta de dados (RTP) ou de controle (RTCP).

Segundo IETF – RFC 1889 (1996), a transmissão de áudio e vídeo deve ser efetuada em sessões distintas. Estas podem ser associadas através da identificação de remetente presente nos pacotes RTCP do emissor. O sincronismo temporal entre áudio e vídeo, também é possível devido ao cruzamento de informações de tempo dos pacotes RTP e RTCP.

A manutenção de fluxos diferentes de mídia numa mesma sessão RTP pode desencadear uma série de problemas:

a) caso ocorra mudança no tipo de *payload* de uma das mídias durante a sessão, não se saberá ao certo se foi a codificação de áudio ou de vídeo que foi alterada, tendo em vista que ambas possuem o mesmo identificador SSRC;

b) o identificador SSRC somente descreve um escopo de temporização e de número de seqüência;
c) os relatórios de receptor e de emissor do RTCP indicam apenas uma fonte SSRC;
d) impossibilidade da combinação de mídias incompatíveis em um perfil RTP que utilize misturadores.

2.4 PROTOCOLO RTCP

O RTCP fornece um fluxo de controle dos dados transmitidos em tempo real pelo protocolo RTP. Os pacotes RTCP são transmitidos periodicamente por todos participantes de uma sessão e utilizam os mesmos mecanismos de comunicação dos dados, sendo multiplexados em endereços de transporte distintos.

Segundo Peterson e Dave (2004, p. 330), as principais funções relacionadas ao fluxo de controle são:

a) oferecer uma estimativa do estado da aplicação e da rede, tendo em vista a análise da qualidade dos dados transmitidos e recepcionados;
b) correlacionamento e sincronização de diferentes fluxos de mídia provenientes de um mesmo emissor;
c) transporte da identidade do emissor para a interface com o usuário.

A primeira função é a mais importante, pois pode ser útil no diagnóstico de problemas da rede, geralmente oriundos de congestionamentos que causam excesso de perdas de pacotes e atrasos na transmissão. Através da análise dos dados de controle fornecidos pelo RTCP, uma aplicação poderá adaptar a sua taxa de transmissão, se adequando às condições atuais da rede. Uma forma de adaptação muito comum é adotar um esquema de codificação de mídia que ocupe menos largura de banda quando a rede estiver congestionada, ou enviar um fluxo

de qualidade mais alta quando houver pouco congestionamento.

O correlacionamento de fluxos é possível, pois nos pacotes RTCP existe uma identificação persistente para uma fonte RTP chamada de *canonical name* (CNAME). Diferentemente do SSRC que pode ser alterado por conflitos ou problemas na aplicação o CNAME é o mesmo enquanto até o término da sessão.

A terceira função é opcional e é utilizada comumente em conferências para manter-se uma cobertura mínima de uma sessão. Informações como nome, email e telefone dos participantes podem ser registrados nos pacotes RTCP.

O RTCP define diversos tipos de pacotes e relatórios como:

a) *sender report* (SR): consiste em um relatório enviado por emissores ativos em uma sessão, relatando as estatísticas de transmissão e recepção;
b) *receiver report* (RR): é enviado por receptores não ativos em uma sessão e é utilizado para relatar as estatísticas de recepção;
c) *source description* (SDES): pacote que contém o CNAME e outras informações descritivas relacionadas a uma fonte RTP;
d) *bye*: pacote indicativo de fim de sessão;
e) *application-defined* (APP): pacote com funções de controle específicos da aplicação.

2.4.1 Formato dos pacotes RTCP

Para identificar o tipo de pacote, o cabeçalho RTCP possui um campo denominado *packet type* (PT), onde é registrado um código específico para cada tipo. O código 200, por exemplo, identifica um SR.

Os principais pacotes do protocolo RTCP são os relatórios SR e RR, pois trazem

informações relevantes sobre a qualidade dos dados transmitidos. O formato destes dois pacotes é muito semelhante, ambos possuem um bloco de recepção de dados (*reception report block*) para cada fonte relacionada desde o último relatório.

Com relação ao RR, o SR possui campos extras de informação que são os seguintes:

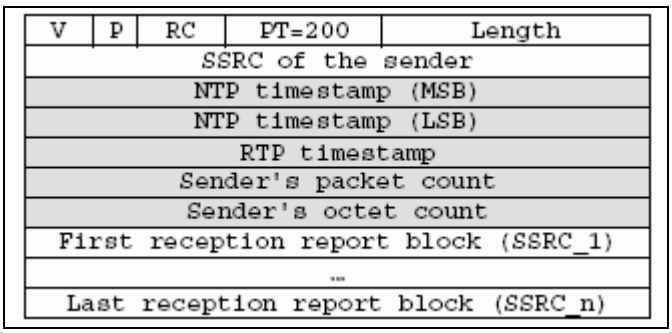
a) *Network Time Protocol* (NTP) *timestamp*: marcação de tempo com a hora do dia em que relatório foi gerado;

b) RTP *timestamp*: marcação de tempo no formato RTP correspondente ao tempo que o pacote foi gerado;

c) *sender's packet count:* contagens cumulativas de pacotes e bytes enviados pelo emissor desde o início da transmissão.

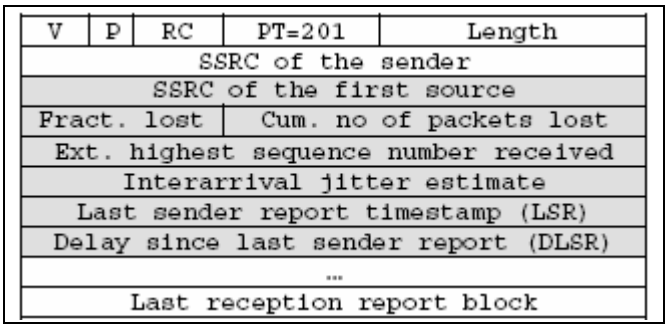
Segundo Peterson e Dave (2004, p. 331), as marcações de tempo são utilizadas para permitir a sincronização de diferentes fluxos de mídia provenientes de uma mesma fonte. Desta forma, mesmo que os fluxos utilizem diferentes granularidades na temporização de dados RTP, é possível converter a hora do dia em marcações de tempo RTP já que a chave de conversão é fornecida.

A Figura 2 apresenta o formato do pacote SR e a Figura 3 ilustra o pacote RR.



Fonte: Koistinen (2005)

Figura 2 – Formato do pacote SR



Fonte: Koistinen (2005)

Figura 3 – Formato do pacote RR

Cada bloco de recepção de dados apresenta as seguintes informações referentes à fonte RTP em questão:

a) SSRC da fonte transmissora de dados, cujas informações são descritas por este bloco;

b) *fraction lost*: representa a fração de pacotes de dados perdidos pela fonte desde o último relatório. Para o cálculo desta estatística é efetuada a comparação do número de pacotes recebidos com o número de pacotes esperados; este último é determinado a partir dos números de seqüência RTP;

c) *cumulative number of packet lost*: representa o número total de pacotes perdidos pela fonte desde o início da transmissão;

d) *extended highest sequence number received*: este campo consiste no número de seqüência mais alto recebido pela fonte RTP em questão. É estendido para 32 *bits* para poder armazenar a contagem de ciclagem do número de seqüência, ou seja, quantas vezes os 16 *bits* do número de seqüência teve que ser iniciado;

e) *interarrival jitter estimate*: representa o *jitter* estimado entre chagadas para a fonte. Pode ser calculado, através da comparação do espaçamento entre chegadas dos pacotes recebidos com o espaçamento esperado no tempo de transmissão;

f) *last sender report timestamp* (LSR): é a última marcação de tempo real recebida

através do RTCP para esta fonte;

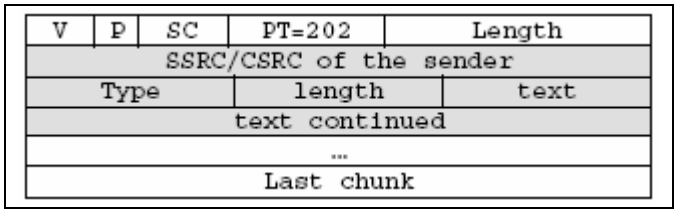
g) *delay since last sender report* (DLSR): representa o atraso desde o último relatório do emissor recebido via RTCP por esta fonte.

Para armazenar as informações de descrição de fonte, o RTCP utiliza os pacotes SDES. São compostos por um cabeçalho fixo padrão, o SSRC do emissor, CNAME e por blocos de itens opcionais de descrição.

O formato padronizado do CNAME é *user@host*, onde *user* representa o nome de usuário e *host* o nome de domínio qualificado da máquina emissora. Conforme Peterson e Dave (2004, p. 331), o formato grande e variável utilizado na representação do CNAME, não seria a forma ideal para representar o SSRC, uma vez que este é enviado com cada pacote de dados e deve ser processado em tempo real. Desta forma o relacionamento entre CNAMEs e os valores dos identificadores SSRC é dado através de mensagens RTCP periódicas, o que possibilita um formato numérico simples e eficiente para o SSRC.

Os blocos de itens de descrição opcionais do SDES podem incluir informações de interesse para os usuários, como o nome real da fonte RTP emissora, *email*, telefone e localização geográfica. Estes itens do SDES não são importantes para a operação do RTP e RTCP, vistos que estes necessitam apenas do CNAME para efetuar o relacionamento de fluxos de um mesmo emissor.

A Figura 4 ilustra o formato do pacote SDES.



Fonte: Koistinen (2005)

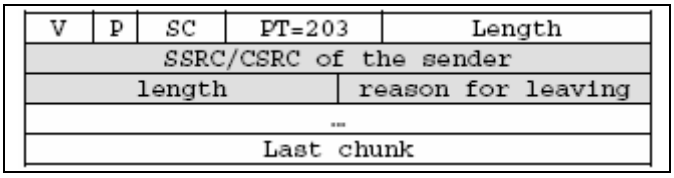
Figura 4 – Formato do pacote SDES

Cada item de descrição é representado por um código pré-estabelecido através do

campo *type* (CNAME é representado pelo código 1), seguido pelo tamanho do item e pela informação propriamente dita.

Para finalizar uma sessão RTP, uma fonte transmissora envia uma mensagem *bye* RTCP, indicando aos receptores que a fonte não está mais ativa. Nesta mensagem é indicado no campo *reason for leaving*, o motivo da saída.

O formato do pacote *bye* é apresentado na Figura 5.

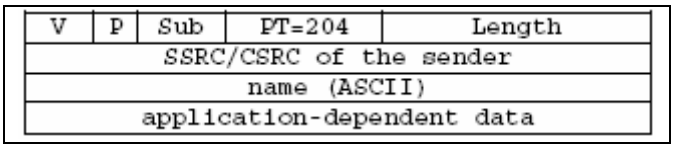


Fonte: Koistinen (2005)

Figura 5 – Formato do pacote *bye*

Para possibilitar funções de controle específicas e para o uso experimental de uma aplicação o RTCP oferece o pacote APP. O campo mais relevante do cabeçalho APP é o *name* que indica a qual grupo de mensagens o pacote pertence.

A Figura 6 mostra o formato de um pacote APP:



Fonte: Koistinen (2005)

Figura 6 – Formato do pacote APP

2.5 QUALIDADE DE SERVIÇO EM APLICAÇÕES MULTIMÍDIA

Em aplicações de redes de computadores, o termo qualidade de serviço ou *Qualitiy of Service* (QoS) pode ser observado de duas formas: do ponto de vista da aplicação ou da rede. A QoS da aplicação é o oferecimento de serviços que atendam as expectativas do usuário com

relação ao tempo de resposta da aplicação e da qualidade, como por exemplo fidelidade na imagem ou som sem congelamento ou ruídos. Já a QoS da rede depende das necessidades de cada aplicação e é mensurada por parâmetros que indicam o desempenho da rede (REDE NACIONAL DE ENSINO E PESQUISA, 2004).

Os principais parâmetros de QoS da rede são:

a) atraso fim a fim: é o tempo entre o envio e a recepção de uma mensagem entre dois nós;
b) variação do atraso (*jitter*): consiste numa distorção ocasionada nos tempos de chegada entre pacotes, comparados aos tempos originais de transmissão;
c) perda de pacotes: consiste no números de pacotes perdidos em uma transmissão em um certo período de tempo;
d) largura de banda e vazão: são medidas de capacidade da transmissão de dados. A largura de banda é a capacidade máxima teórica de transmissão em uma conexão. A vazão representa o volume de dados que está trafegando entre dois nós da rede em um determinado período.

Em aplicações multimídia que utilizam o protocolo RTP para o transporte de dados, os processos relacionados à qualidade de serviço da rede devem ser implementados na camada de aplicação. Isto porque o RTP não utiliza nenhum mecanismo de reserva de recursos e de garantia da qualidade de serviço dos dados transportados.

As aplicações que utilizam mecanismos de QoS e que adaptam o seu funcionamento as variações de desempenho do ambiente de rede, utilizam basicamente dois processos: monitoração e adaptação de QoS.

Segundo Lunardi e Dotti (2001), o processo de monitoração de QoS consiste na observação de eventos relacionados aos requisitos de qualidade de serviço da aplicação, reportando parâmetros de desempenho da rede e tornando possível identificar o estado

corrente. Este processo, além da observação de eventos, têm a incumbência de encaminhar mensagens de notificação para os módulos que tratam da adaptação de QoS.

Quando utilizado o RTP, o módulo de monitoração de QoS da aplicação fará a análise dos relatórios SR e RR do pacotes RTCP, enviados periodicamente. Estes relatórios trazem estatísticas muito úteis na determinação do estado da rede e da necessidade de adaptações da aplicação.

O outro processo necessário para a implementação de QoS em aplicações multimídia é o de adaptação de QoS. Segundo Lunardi e Dotti (2001), este processo tem a função de ajustar a aplicação às mudanças constantes das condições da rede. Normalmente, o processo de adaptação multimídia acontece como resultado de notificações emitidas pelo mecanismo de monitoração de QoS. Ações corretivas são tomadas, baseadas nos parâmetros da rede e nos requisitos definidos pela aplicação.

Um típico módulo de adaptação engloba um conjunto de políticas adaptativas. Estas políticas consistem em um conjunto de estratégias que visam assegurar pelo menos um nível mínimo de QoS para a aplicação. Os mecanismos de adaptação de QoS, normalmente são implementados na aplicação emissora e podem estar relacionados ao transporte de dados e a aplicação. O controle da taxa de transmissão e do tamanho de buffer de recebimento (no transporte) e adaptação da taxa de codificação de um vídeo (na aplicação) são exemplos de mecanismos que podem ser empregados.

2.6 TRABALHOS CORRELATOS

Em Lunardi e Dotti (2001), é descrita uma camada de adaptação para incorporar QoS a aplicações multimídia da internet. A camada de adaptação consiste em uma API que fornece funções que encapsulam o processo de monitoração dos parâmetros de QoS da rede através do

controle de pacotes RTCP. A API desenvolvida, além de um módulo de monitoração, possui um módulo de adaptação que, baseado nas informações da adaptação, aplica algumas políticas para evitar a degradação da mídia, como mudança nas taxas de transmissão e modificação do tamanho de *buffer* da aplicação cliente.

Gomes et al. (2000) apresenta uma avaliação de um sistema distribuído de criação e apresentação de documentos multimídia. O objetivo do trabalho é determinar um ambiente propício para criação e recuperação de documentos multimídia adaptativos em redes corporativas. A aplicação desenvolvida nos testes é baseada no modelo cliente/servidor e utiliza o protocolo RTP na transmissão dos fluxos multimídia. O lado cliente da aplicação, além da visualização do documento, é responsável pela solicitação de adaptação semântica da mídia. A solicitação de adaptação ao servidor é baseada nos valores de QoS dos pacotes RTCP.

Em Tschöke (2001), é descrita uma ferramenta cuja função é a criação de *streaming* para a distribuição de sinais de vídeo pela internet. Para efetivação da transmissão, a ferramenta utiliza as tecnologias de protocolos para aplicações multimídia, como o RTP, RTCP, RTSP e *Resource Reservation Protocol* (RSVP). Este último é responsável pela reserva de recursos da rede.

Em Koliver (2001), a adaptação de QoS para sistemas multimídia é baseada em controle nebuloso. O controlador nebuloso determina a nova taxa de *bits* a ser enviada pelo remetente na transmissão da mídia. A base para esta nova taxa é dada por informações de realimentação do controlador diante da análise dos parâmetros da rede.

3 DESENVOLVIMENTO DO TRABALHO

O presente capítulo descreve a especificação, a implementação e os testes do protótipo de ferramenta referentes às ações adaptativas de qualidade de serviço. Na primeira seção são descritos os requisitos da ferramenta. A segunda seção mostra a especificação do protótipo através dos diagramas de casos de uso, atividades e de classe. A terceira seção descreve a implementação, as técnicas e ferramentas utilizadas e a operacionalidade da implementação através de um caso de uso. Na quarta e última seção estão descritos os resultados obtidos e as discussões referentes aos trabalhos correlatos apresentados na fundamentação teórica.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Nesta seção são descritos os principais requisitos do protótipo. Os requisitos são classificados como funcional (RF) e não-funcional (RNF).

O protótipo de ferramenta deverá:

a) permitir que o aplicativo cliente solicite a transmissão de um arquivo de mídia armazenado no aplicativo servidor (RF);
b) efetuar, através servidor, a transmissão da mídia solicitada pelo cliente (RF);
c) possuir um módulo de monitoração da qualidade de serviço da rede no servidor (RF);
d) possuir um módulo de adaptação no servidor da ferramenta para efetuar a alteração na taxa de transmissão ou sinalização da mudança do tamanho do buffer quando os parâmetros de qualidade de serviço definidos para a aplicação não forem atendidos (RF);
e) ter uma área para reprodução da mídia no aplicativo cliente (RF);

f) possuir uma área no aplicativo cliente que informe as estatísticas referentes à transmissão em andamento, necessário para verificar os efeitos do processo de adaptação (RF);

g) utilizar os protocolos RTP e RTCP para transmissão multimídia fim a fim e controle da transmissão, respectivamente (RNF);

h) ser implementado na plataforma Java através do ambiente Netbeans 4.0 (RNF).

3.2 ESPECIFICAÇÃO

Nesta seção é apresentada a especificação do protótipo, realizada através da *Unified Modeling Language* (UML). Os diagramas de casos de uso, atividade e de classes foram utilizados na especificação.

3.2.1 Diagramas de casos de uso

Segundo Furlan (1998), o diagrama de casos de uso tem como propósito descrever de forma conceitual a estrutura de um sistema. A Figura 7 apresenta os casos de usos referentes ao ator usuário do aplicativo cliente.

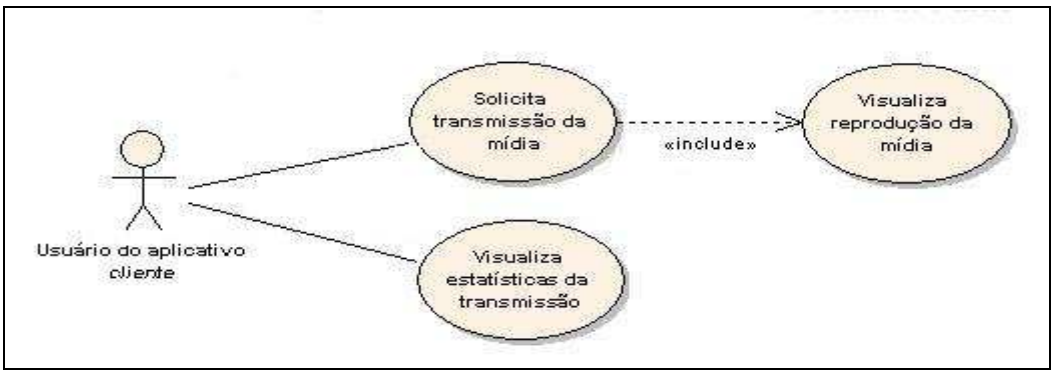


Figura 7 – Casos de uso do usuário do aplicativo cliente

O Quadro 2 descreve o caso de uso solicita transmissão da mídia do usuário do aplicativo cliente.

SOLICITA TRANSMISSÃO DA MÍDIA
Descrição: o usuário do aplicativo cliente solicita a transmissão de um arquivo de mídia armazenado no servidor. **Ator principal:** usuário do aplicativo cliente **Cenário principal:** solicita transmissão a) o usuário informa o IP do servidor, o nome do arquivo de áudio ou vídeo e a partir de qual sessão RTP deseja receber o *streaming* multimídia e em seguida clica no botão iniciar; b) o aplicativo cliente valida os dados informados e estabelece um canal de comunicação exclusivo com o servidor para troca de mensagens de controle; c) o aplicativo cliente aguarda o recebimento da mídia a partir da sessão RTP especificada **Cenário de exceção:** servidor não disponível Caso não seja possível estabelecer uma conexão com o servidor é emitida uma mensagem de erro: "Servidor desconhecido ou não disponível!". **Pós-condição:** o aplicativo cliente está apto a receber a mídia solicitada e iniciar a sua reprodução.

Quadro 2 – Caso de uso solicita transmissão da mídia

O Quadro 3 descreve o caso de uso visualiza reprodução da mídia, sendo este um *include* de solicita transmissão da mídia.

VISUALIZA REPRODUÇÃO DA MÍDIA
Descrição: o usuário do aplicativo cliente visualiza a reprodução da mídia. **Ator principal:** usuário do aplicativo cliente **Cenário principal:** visualiza reprodução a) o usuário visualiza a reprodução da mídia; b) a reprodução do áudio ou vídeo pode ser interrompida temporariamente ou em definitivo, ocasionando o encerramento da transmissão nas sessões RTP. **Pós-condição:** foi visualizado a mídia no aplicativo cliente e encerrado a sessão RTP, estabelecida pelo servidor para a transmissão do *streaming*.

Quadro 3 – Caso de uso visualiza reprodução da mídia

O Quadro 4 descreve o caso de uso visualiza estatísticas da transmissão referente ao usuário do aplicativo cliente.

VISUALIZA ESTATÍSTICAS DA TRANSMISSÃO
Descrição: permite ao usuário do aplicativo cliente visualizar as estatísticas da transmissão. **Ator principal:** usuário do aplicativo cliente **Pré-condição:** o aplicativo cliente deve estar recebendo *streaming* a partir da sessão RTP especificada. **Cenário principal:** visualiza estatísticas da transmissão O usuário clica no botão estatísticas e visualiza as estatísticas globais da transmissão e as específicas das sessões RTP, referentes aos parâmetros de QoS da rede.

Quadro 4 – Caso de uso visualiza estatísticas da transmissão

A Figura 8 ilustra os casos de uso do aplicativo cliente no servidor.

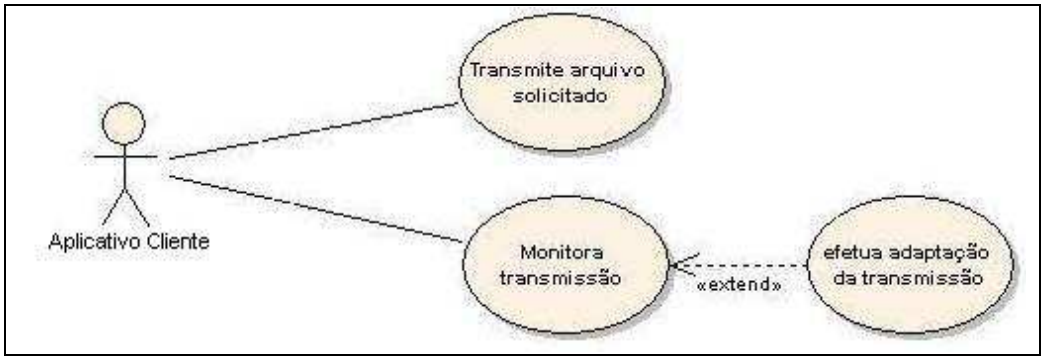


Figura 8 – Casos de uso do aplicativo cliente no servidor

O Quadro 5 descreve o caso de uso transmite arquivo.

TRANSMITE ARQUIVO

Descrição: servidor RTP efetua a transmissão do arquivo solicitado pelo aplicativo cliente.

Ator principal: aplicativo cliente

Pré-condição: o cliente deve ter efetuado a solicitação de transmissão de um arquivo.

Cenário principal: transmite arquivo

a) o servidor recebe a solicitação de transmissão de arquivo e estabelece um canal de comunicação exclusiva com o cliente para a troca de informações de controle;

b) as informações de requisição de cada cliente são apresentadas na tela principal do aplicativo;

c) o servidor busca o arquivo solicitado em disco, processa as faixas da mídia em diferentes sessões RTP, associa a sessão a um monitor e um adaptador de qualidade de serviço específico e por fim cria os *streams* de saída para posterior envio à rede.

Cenário de exceção: arquivo não localizado

Caso o arquivo solicitado não seja localizado, o servidor encaminha pelo canal de comunicação a seguinte mensagem ao aplicativo cliente: "Arquivo solicitado não pode ser localizado!".

Cenário de exceção: erro no estabelecimento das sessões RTP.

Caso ocorra algum erro interno na criação dos *streams* ou no estabelecimento das sessões RTP é enviada ao canal de comunicação a seguinte mensagem informativa para o aplicativo cliente: "Transmissão não pode ser iniciada, erro no processamento das sessões RTP!". **Pós-condição:** foi efetuada a transmissão do arquivo solicitado sob a forma de mídia *streaming* e liberado o canal de comunicação, assim como as sessões RTP utilizadas durante o processo de transmissão.

Quadro 5 – Caso de uso transmite arquivo

No Quadro 6 é apresentado o caso de uso monitora transmissão.

MONITORA TRANSMISSÃO
Descrição: permite a monitoração da transmissão baseado nas informações dos parâmetros de qualidade de serviço contidos nos relatórios de recepção RTCP enviado pelos clientes. **Ator principal:** aplicativo cliente **Pré-condição:** a transmissão de *streaming* deve ter sido iniciada. **Cenário principal:** monitora transmissão a) a cada recepção de um relatório RTCP enviado por um participante, o servidor analisa os parâmetros de qualidade de serviço que indicam como o cliente está recebendo os dados; b) baseado na informação de fração de pacotes perdidos, é calculado a taxa de perdas na transmissão para determinação do nível de tráfego da rede; c) as informações referentes aos parâmetros de qualidade de serviço são encaminhadas ao adaptador específico da sessão, juntamente com a descrição do nível de carga da rede; d) as informações coletadas nos relatórios RTCP são organizadas em nível de sessão e de participante e podem ser visualizadas pelos usuários do aplicativo servidor, sendo atualizadas permanentemente durante a transmissão. **Pós-condição:** foram coletados os parâmetros de qualidade de serviço e determinado o nível de tráfego da rede.

Quadro 6 – Caso de uso monitora transmissão

O Quadro 7 descreve o caso de uso efetua adaptação da transmissão, sendo este uma extensão do caso de uso monitora transmissão.

EFETUA ADAPTAÇÃO DA TRANSMISSÃO
Descrição: permite verificar se os requisitos mínimos de qualidade de serviço definidos pela aplicação foram atendidos e adotar medidas de adaptação da mídia às condições da rede.
Ator principal: aplicativo cliente
Cenário principal: efetua adaptação da transmissão
a) o módulo adaptador do servidor recebe os dados referentes à qualidade de serviço obtidos e determinados no processo de monitoração e compara com os parâmetros definidos pela aplicação.
b) caso os requisitos mínimos não forem atendidos, o módulo define uma ação adaptativa apropriada para o ajuste da transmissão as condições da rede. Esta ação pode ser a mudança do tipo de codificação transmitido ou a sinalização para a mudança do tamanho do buffer no recebedor dos dados.
Pós-condição: uma ação adaptativa foi efetuada, caso os requisitos mínimos da aplicação não foram atendidos.

Quadro 7 – Caso de uso efetua adaptação da transmissão

A Figura 9 ilustra os casos de uso do usuário do aplicativo servidor.

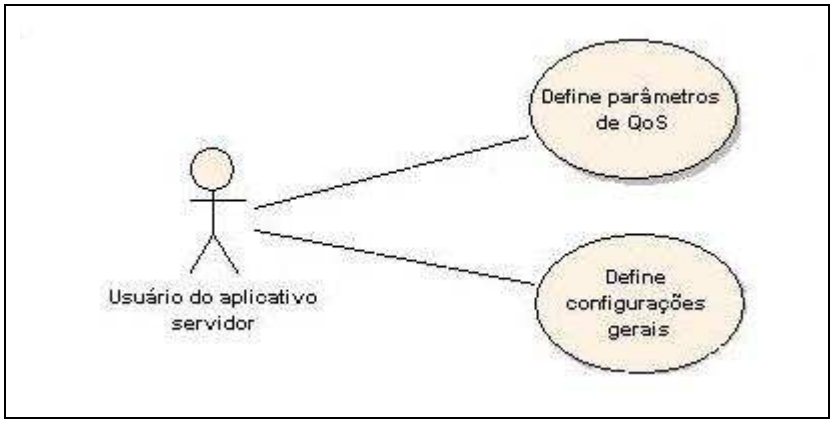


Figura 9 – Casos do usuário do aplicativo servidor

O Quadro 8 descreve o caso de uso Define parâmetros de QoS.

DEFINE PARÂMETROS DE QoS
Descrição: permite ao usuário do aplicativo servidor definir os parâmetros de QoS utilizados nos módulos de monitoração e adaptação do servidor. **Ator principal:** usuário do aplicativo servidor. **Cenário principal:** define parâmetros de QoS. a) o usuário clica no item "Definir QoS" do menu "Definições"; b) o usuário define os valores para os parâmetros de QoS que são: variação do atraso (jitter), fração de pacotes perdidos e limite máximo da vazão de banda. Também são informados os valores das variáveis utilizadas pelo módulo de monitoração: peso da taxa de perdas anterior, limite máximo da rede descarregada e limite mínimo da rede congestionada.

Quadro 8 – Caso de uso Define parâmetros de QoS

O Quadro 9 descreve o caso de uso Define configurações gerais.

DEFINE CONFIGURAÇÕES GERAIS
Descrição: permite ao usuário do aplicativo servidor definir as configurações gerais do servidor. **Ator principal:** usuário do aplicativo servidor. **Cenário principal:** define configurações gerais. a) o usuário clica no item "Configurações Gerais" do menu "Definições"; b) o usuário informa o caminho de localização do repositório de mídia e os campos de descrição para os pacotes SDES do protocolo RTCP que são: nome, email, localidade, telefone, privado e ferramenta e notas; c) o usuário clica no botão "Confirmar" e o aplicativo valida os dados informados.

Quadro 9 – Caso de uso Define configurações gerias

3.2.2 Diagrama de atividades

Nesta seção são demonstrados os diagramas de atividades do protótipo que ilustram as atividades executadas desde a solicitação da mídia por parte do cliente até a transmissão, monitoração e adaptação, tarefas estas desempenhadas pelo servidor.

Os diagramas apresentados mostram o funcionamento geral do protótipo, onde a integração entre cliente e servidor é dada pela trocas de mensagens de controle do protocolo RTCP ou pelo canal de comunicação estabelecido no início da transmissão.

Para representar o aplicativo cliente foram desenvolvidos dois diagramas, um denominado “atividades do cliente RTP” que ilustra o processo de solicitação da mídia, criação das sessões RTP e recebimento dos *streams*. O outro diagrama consiste numa sub-atividade do diagrama anterior, denominado “sub-atividade reprodução da mídia” que mostra as atividades executadas no processo de reprodução da mídia.

O aplicativo servidor é representando pelo “diagrama de atividades do servidor RTP”, que realiza o processamento das requisições, criação das sessões RTP e envio dos *streams* à rede e pelas sub-atividades “monitora sessão RTP” e “transmite mídia”.

A Figura 10 mostra as atividades do cliente RTP. Este diagrama mostra basicamente o processo de coleta e envio das informações de requisição ao servidor, estabelecimento das sessões RTP, recepção e reprodução dos *streams* na sessão RTP correspondente.

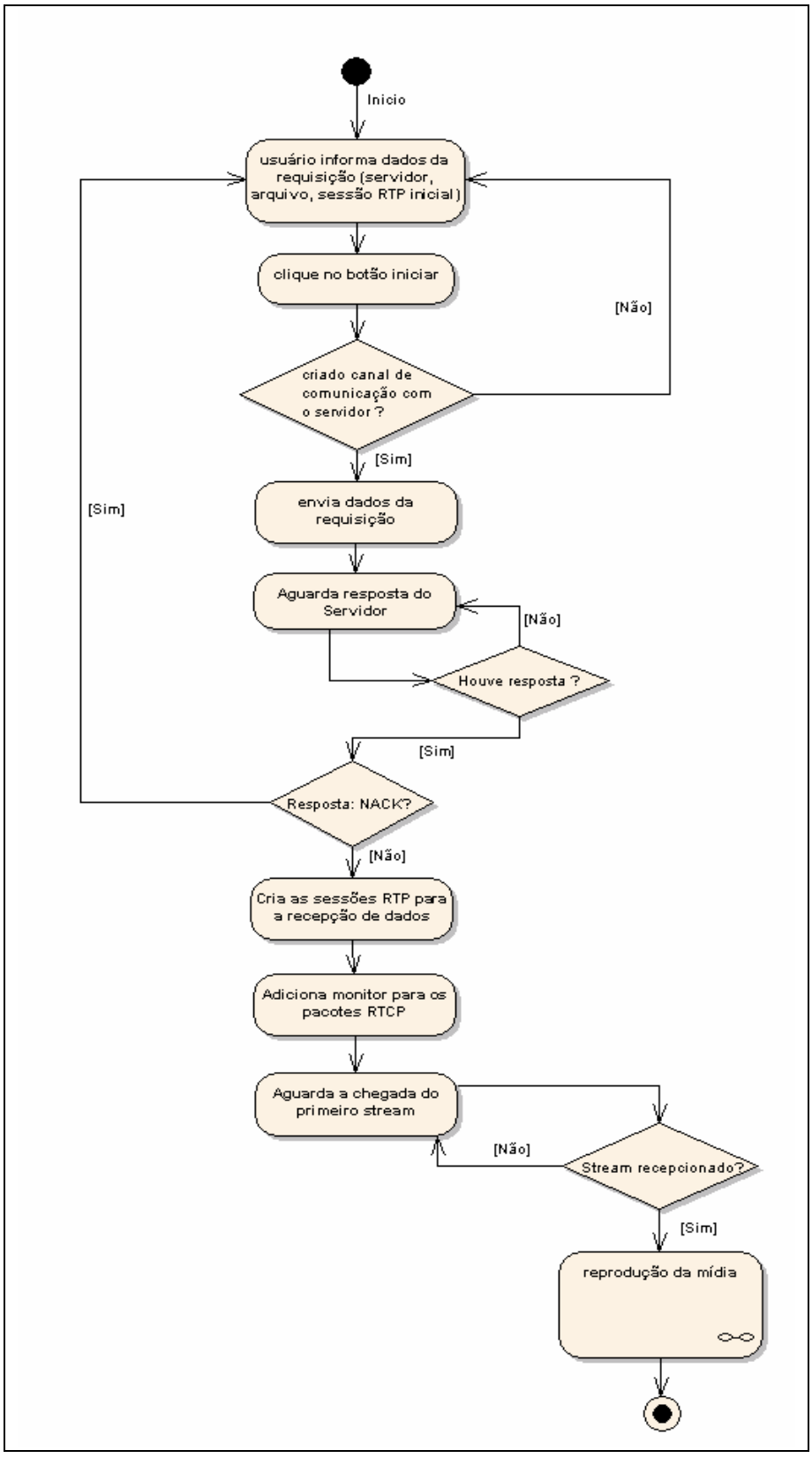


Figura 10 – Diagrama de atividades do cliente RTP

A reprodução da mídia no cliente RTP é ilustrada através da sub-atividade

representada através da Figura 11. É criado um *player* a cada novo *stream* identificado na sessão RTP, efetuado a reprodução da mídia e recepção dos pacotes RTCP.

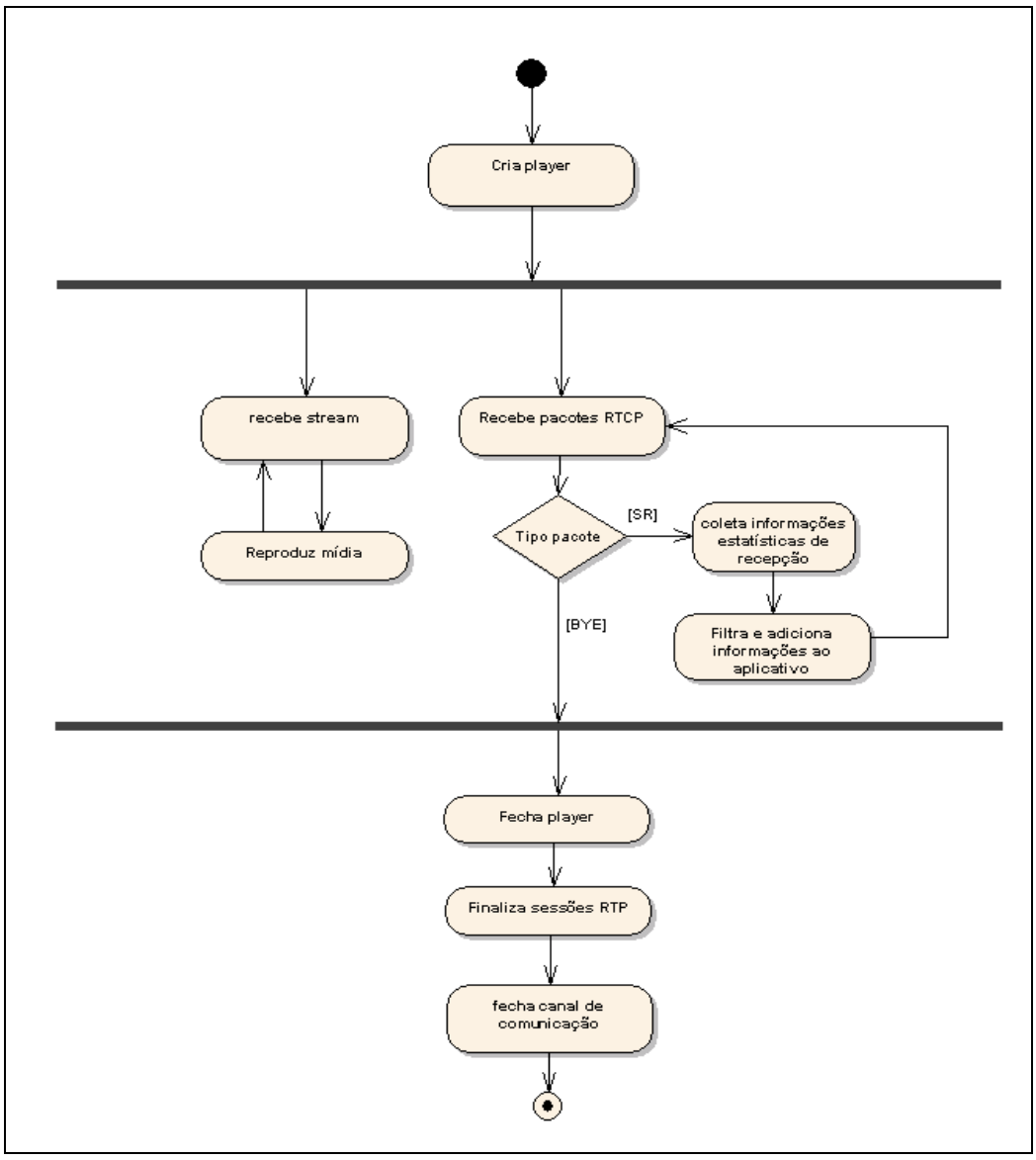


Figura 11 – Sub-atividade reprodução da mídia

No aplicativo servidor são definidos os diagramas de atividades, servidor RTP e suas respectivas sub-atividades transmite mídia e monitora sessão RTP. A Figura 12 representa o diagrama de atividades servidor RTP, onde o servidor aguarda as requisições do cliente, cria o canal de comunicação, processa a mídia e cria as sessões RTP para cada faixa antes de enviar

o *stream* à rede. De forma paralela é efetuada a transmissão e monitoração da sessão RTP.

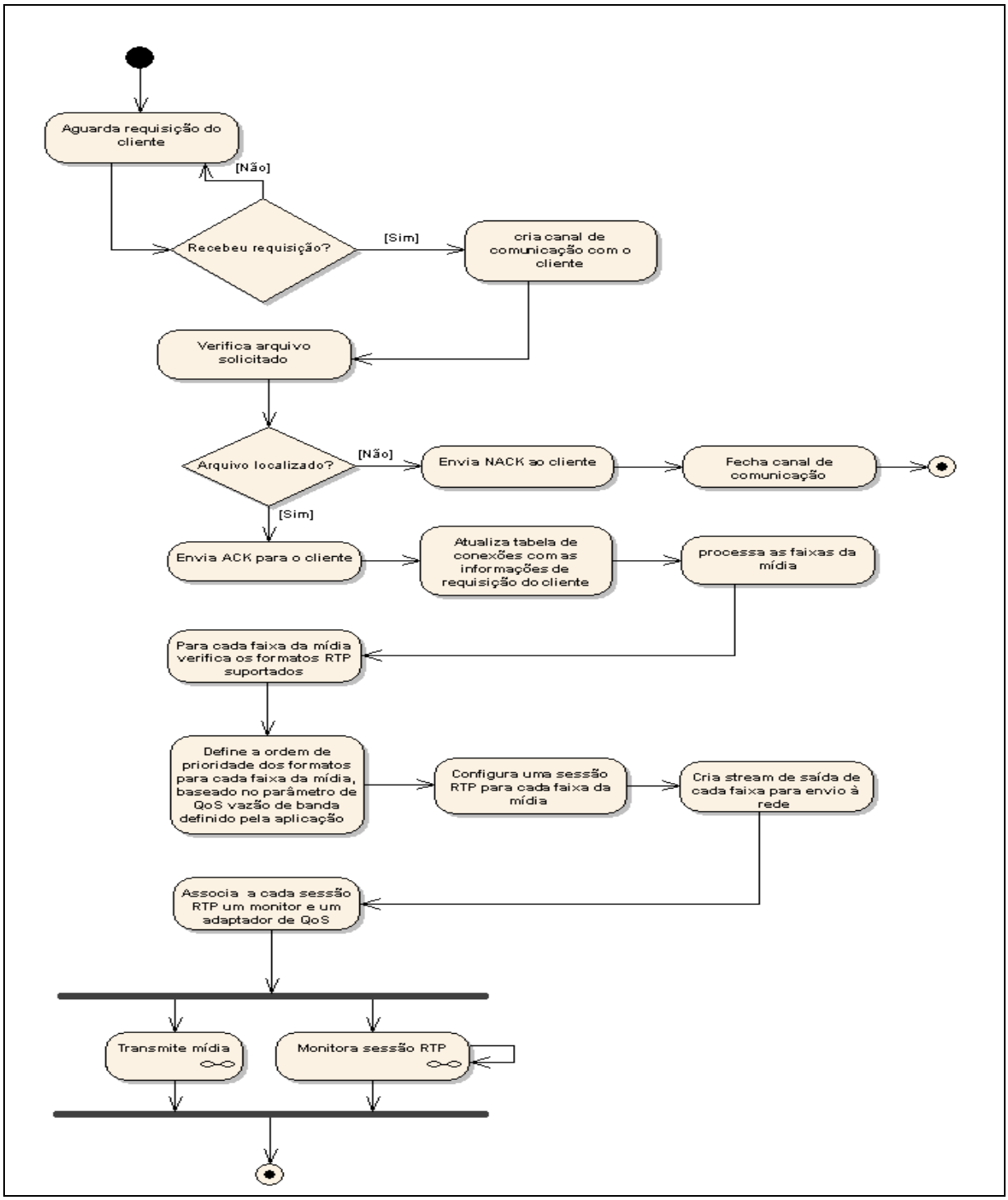


Figura 12 – Diagrama de atividades do servidor RTP

A sub-atividade monitora transmissão RTP é representada pela Figura 13.

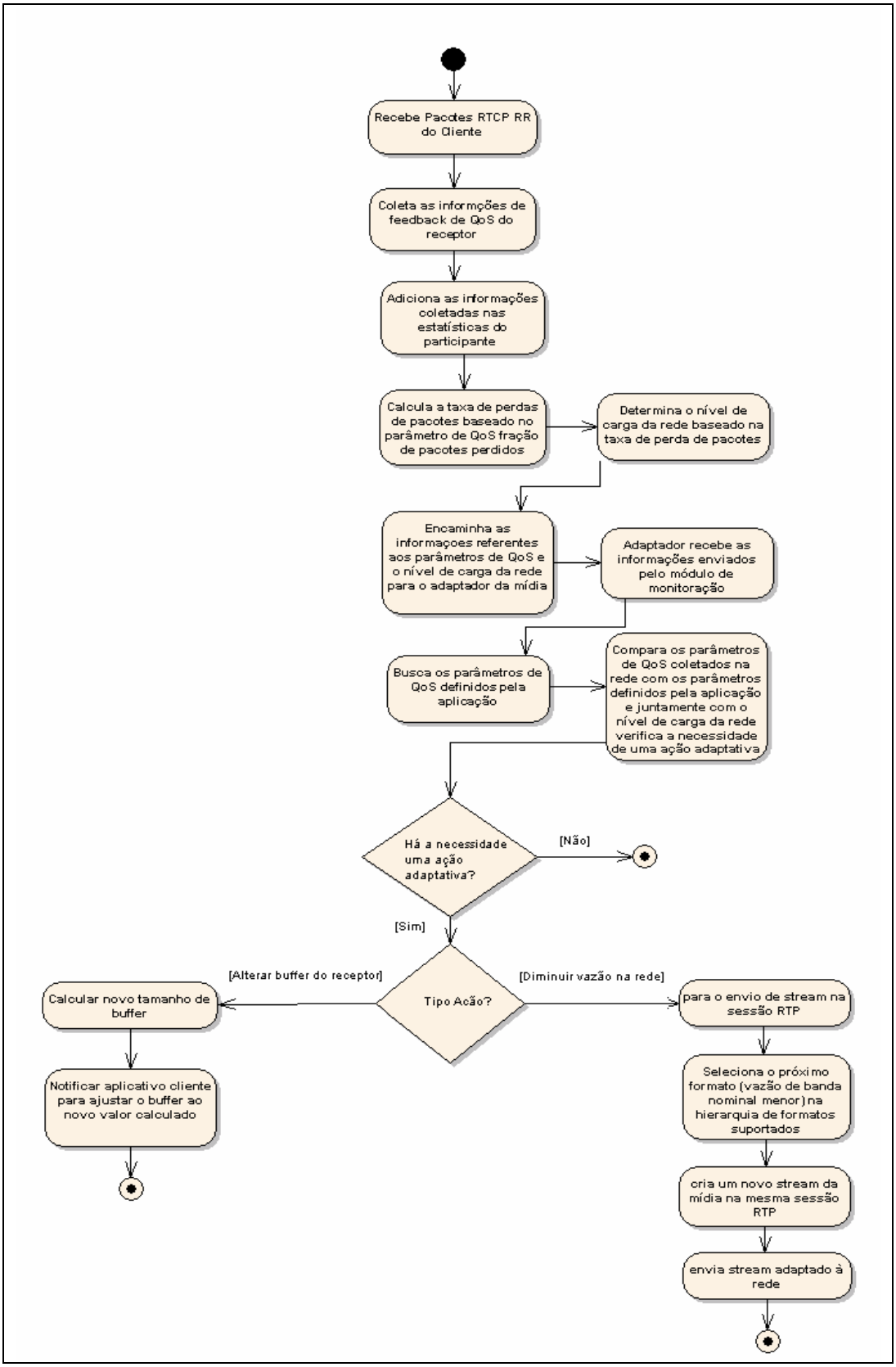


Figura 13 – Sub-atividade monitora sessão RTP

Nesta sub-atividade são coletadas as informações de *feedback* de QoS enviadas pelo

cliente RTP. Com base no parâmetro fração de pacotes perdidos é determinada a taxa de perda de pacotes da transmissão e por sua vez o nível de carga da rede. Os parâmetros de QoS da rede, juntamente com os dados determinados no processo de monitoração são encaminhados ao módulo de adaptação de QoS que determina a execução ou não de uma ação adaptativa.

Na Figura 14 é representada a sub-atividade transmite mídia, que efetua o envio do *stream* à rede. Ao finalizar o processo de transmissão é fechado o canal de comunicação com o cliente e removido o mesmo da tabela de conexões do servidor.

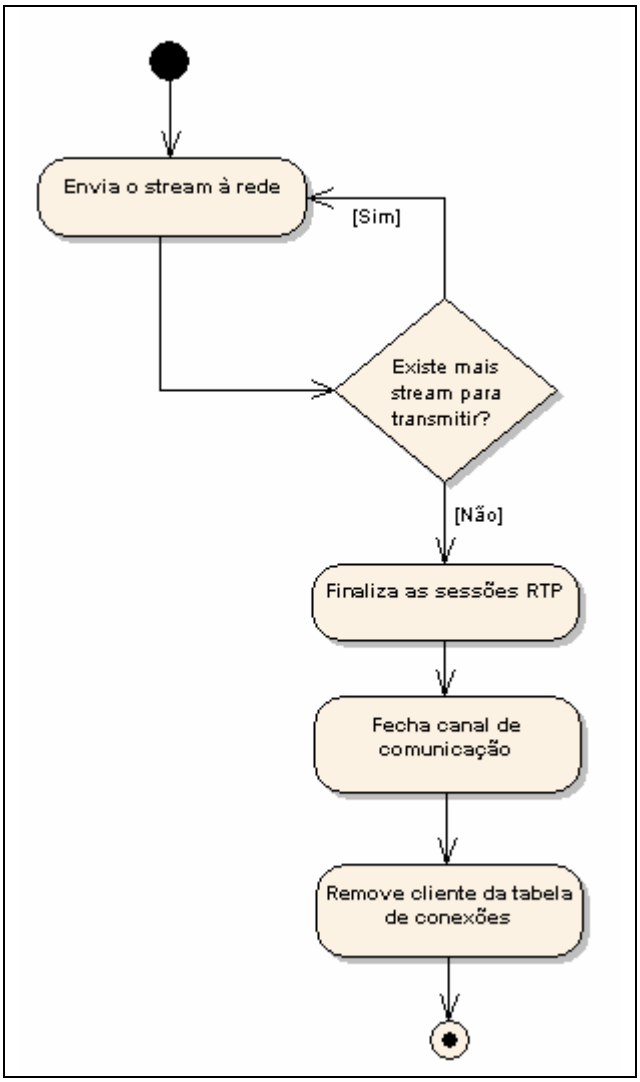


Figura 14 – Sub-atividade transmite mídia

3.2.3 Diagrama de classes

Nesta seção são especificados os diagramas de classe do aplicativo servidor e do cliente, destacando as principais classes do protótipo. Em um segundo momento é descrito um breve comentário a respeito das funcionalidades e características de cada classe.

A Figura 15 ilustra o diagrama de classes do servidor:

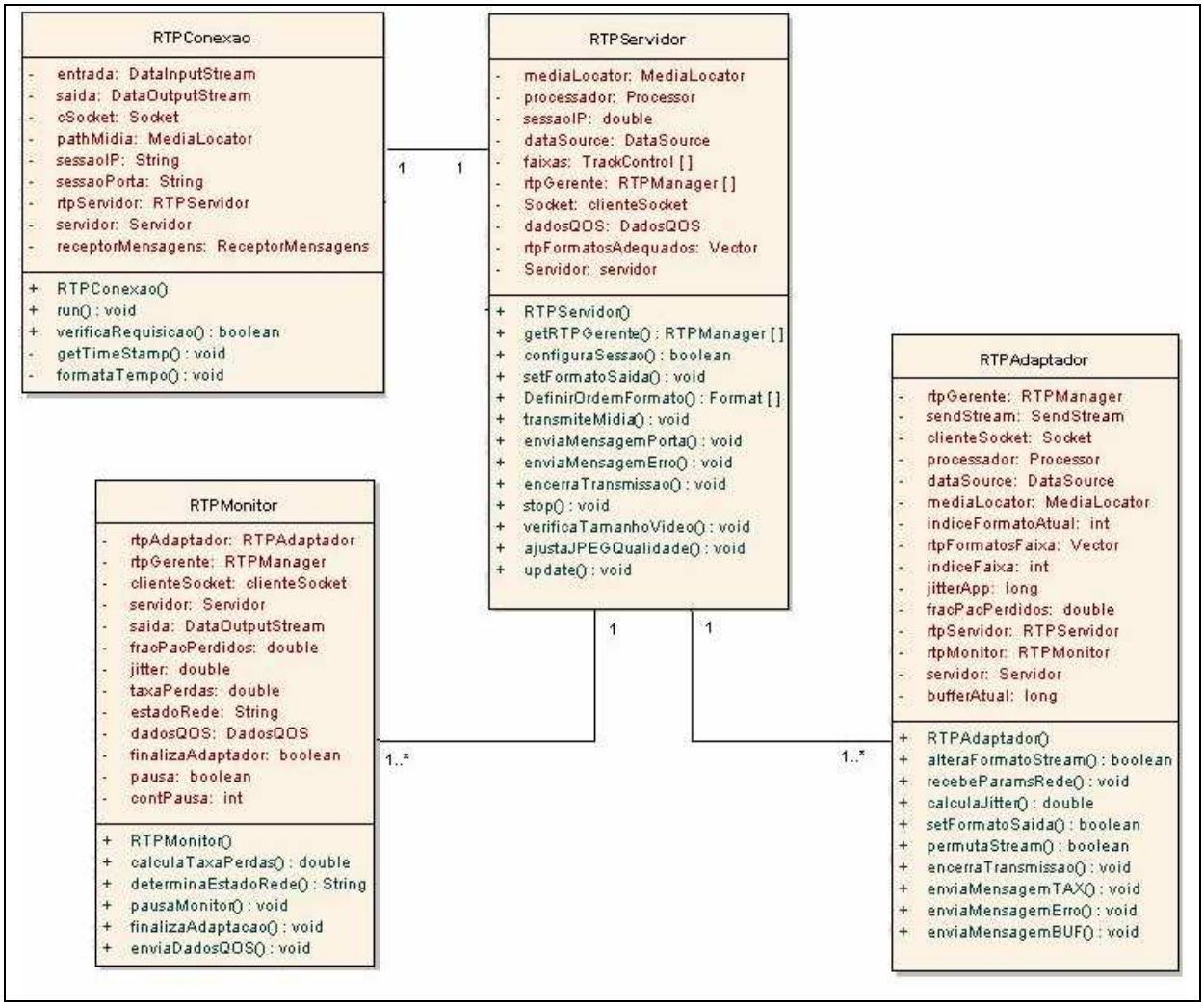


Figura 15 – Diagrama de classes do servidor

O servidor é composto pelas seguintes classes:

a) RTPConexao: é uma *thread* criada a cada solicitação de transmissão de arquivo. Um objeto desta classe verifica as informações de requisição e certifica-se que o

arquivo solicitado realmente existe no servidor;

b) RTPServidor: instanciado por um objeto RTPConexao, é responsável pelos processamento da mídia, configurando uma sessão e um gerente de sessão para cada faixa da mídia, define o formato inicial do *stream* e, no caso de áudio, a ordem dos formatos suportados adequados em ordem decrescente de vazão nominal de banda. É esta classe que efetivamente enviará o *streaming* à rede;

c) RTPMonitor: instanciado por um objeto RTPServidor, é responsável pelo monitoramento de uma sessão RTP, coletando as informações de QoS da rede, obtidas através dos pacotes RTCP RR, enviados periodicamente pelo cliente. É calculada a taxa de perdas de pacotes e determinado o estado de carga da rede;

d) RTPAdaptador: instanciado por um objeto RTPServidor, esta classe compara os parâmetros de QoS definidos pela aplicação com os obtidos na rede e na análise do módulo de monitoração. Caso os requisitos mínimos de qualidade de serviço da aplicação não forem atendidos é executada uma ação adaptativa, como mensagem de mudança do tamanho do buffer no receptor e alteração do formato do *stream* na mesma sessão RTP. Ambas as ações são efetuadas apenas para áudio.

A Figura 16 ilustra o diagrama de classes do cliente.

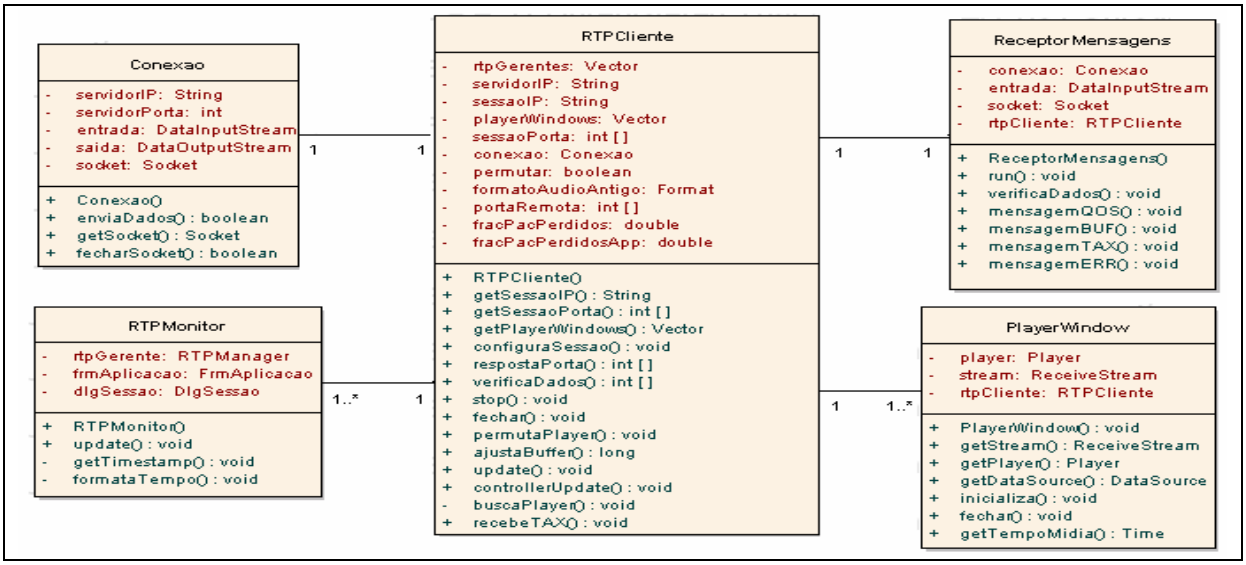


Figura 16 – Diagrama de classes do cliente

O cliente é composto pelas seguintes classes:

a) Conexao: classe responsável pela criação do canal de comunicação e pelo envio das informações de requisição da mídia como nome do arquivo e os endereços IP e de porta da sessão RTP inicial;

b) RTPCliente: é a principal classe do aplicativo cliente, responsável pela criação e configuração das sessões RTP onde serão recebidos os *streams* enviados pelo servidor. Cada sessão RTP possui um gerente de sessão e recebe eventos remotos como a identificação de um novo *stream* na sessão, mudança do tipo da mídia, e fim de sessão (pacote RTCP *bye*). Quando ocorre a mudança do tipo da mídia o objeto RTPCliente efetua a permutação de *player*, assim como o ajuste do tamanho do *buffer*, no recebimento de uma mensagem deste tipo enviada pelo servidor;

c) RTPMonitor: é instanciado por um objeto RTPCliente e recebe os pacotes RTCP SR enviados pelo servidor que possuem estatísticas de transmissão da sessão RTP;

d) *PlayerWindow*: é instanciado por um objeto RTPCliente e trata-se de uma classe que irá criar um *player* para a reprodução da mídia. Este *player* é constituído de

componentes visuais para o controle da mídia e uma área para exibição de conteúdo no caso de vídeo;

e) ReceptorMensagens: consiste em uma *thread* de recebimento de mensagens de controle do servidor, estas mensagens podem indicar um novo valor de *buffer*, informações de QoS da rede, erros no servidor e mudança de formato RTP.

3.3 IMPLEMENTAÇÃO

Nesta seção é mostrada em detalhes a implementação do protótipo, onde são descritas a técnicas e ferramentas utilizadas e a operacionalidade da implementação através de um estudo de caso. Para finalizar, são comentados os resultados obtidos a partir do estudo de caso.

3.3.1 Técnicas e ferramentas utilizadas

Para a implementação do protótipo foi utilizada a plataforma Java 1.5.0 e o ambiente de desenvolvimento Netbeans 4.0. Uma extensão da plataforma *Java 2 Standard Edition* (J2SE) foi instalada, a API *Java Media Framework* (JMF). A especificação do protótipo foi realizada através da ferramenta *Enterprise Architect* (SPARXS, 2005).

3.3.1.1 API JMF

Conforme Deitel e Deitel (2003, p. 1113), a JMF é uma API da plataforma Java para multimídia, podendo ser utilizada para a criação de aplicativos Java que reproduzem, editam e capturam os mais populares tipos de mídia.

A JMF encontra-se na versão 2.0 especificada pelas empresas IBM e Sun. Esta especificação possui uma implementação de referência denominada JMF 2.1.1, posteriormente atualizada para a versão 2.1.1e.

A transmissão de mídia com a JMF é dada através do RTP, por isso é definido na JMF um subconjunto de API's exclusivas para o tratamento do RTP e RTCP. A arquitetura de alto nível da JMF é composta pela aplicação do usuário e a API JMF que engloba a JMF RTP API.

A JMF facilita o processo de desenvolvimento de aplicativos que efetuam transmissão multimídia sob a forma de *streaming,* pois não é necessário se preocupar com a montagem dos pacotes RTP e a geração dos pacotes de controle RTCP, utilizados para estatísticas de QoS.

3.3.1.2 Implementação do servidor RTP

O Quadro 10 mostra o código do método *transmiteMidia* da classe *RTPServidor*. Este método é responsável pela criação das sessões RTP e dos *streams* para posterior envio a rede. Algumas classes e interfaces da JMF são utilizadas neste método como *Processor*, *DataSource*, *SessionAddress*, *SourceDescription* e *SendStream*.

A partir de um *Processor* que controla o processamento da mídia, obtém-se um *DataSource* que armazena os dados propriamente ditos. Para cada faixa da mídia é instanciado um objeto de *RTPManager* para efetuar o gerenciamento das sessões RTP e criar os *streams* de saída (*SendStream*) a partir do *DataSource*. As sessões RTP são representadas pelos objetos *SessionAddress* que encapsulam os endereços IP e de porta da sessão.

```
//método responsável pela criação e transmissão de streams
public synchronized boolean transmiteMidia(){
        //obtém uma fonte de dados que representa a mídia
       dataSource = processador.getDataOutput();
        if (dataSource == null) {
            JOptionPane.showMessageDialog(null, "Não há DataSource para a mídia",
                                          "RTPServidor", JOptionPane.ERROR_MESSAGE);
            enviaMensagemErro("Mensagem do Servidor: Não há DataSource para a mídia");
            return false;
        }
        String ParamQOSApp[] = new String[3];
        int [] portas;
        Format formato = null;
        SessionAddress sessaoLocal[] = new SessionAddress[1] , sessaoRemota;
        SendStream sendStream;
        //cria buffers para obter as faixas da mídia
        PushBufferDataSource bufferDatasource = ( PushBufferDataSource ) dataSource ;
        PushBufferStream bufferStream[] = bufferDatasource.getStreams() ;
        rtpGerente = new RTPManager[ bufferStream.length ] ;
        portas = new int[bufferStream.length];
```

```
    try {
            ip = InetAddress.getByName(sessaoIP);
            int portaAux = sessaoPorta + 10;
            for (int i = 0 ; i < bufferStream.length ; i++){
                portas[i] = PoolPortas.getPorta();
                if ((portas[i] == sessaoPorta) || (portas[i] == portaAux)){
                    portas[i] = PoolPortas.getPorta();
                }
            }
            if (ip.equals(InetAddress.getLocalHost())){
                for (int i = 0 ; i < portas.length ; i++){
                    if ((portas[i] == sessaoPorta) || (portas[i] == portaAux))
                        portas[i] = portas[i] + 3000;
                }
            }
            for ( int i = 0 ; i < bufferStream.length ; i++ ){
                formato = bufferStream[i].getFormat();
                //obtém os parâmetros de QoS da aplicação
                if (formato instanceof AudioFormat){
                    ParamQOSApp[0] = dadosQOS.getJitAudio();
                    ParamQOSApp[1] = dadosQOS.getFracAudio();
                }
                DadosGerais dadosGerais = new DadosGerais();
                //cria um gerente de sessão para cada stream da mídia
                rtpGerente[i] = RTPManager.newInstance();
                //configura os campos de descrição para os campos SDES do RTCP
                SourceDescription [] srcDesc = new SourceDescription[8];
                srcDesc[0] = new SourceDescription(SourceDescription.SOURCE_DESC_CNAME,
                        SourceDescription.generateCNAME(), 1, false);
                srcDesc[1] = new SourceDescription(SourceDescription.SOURCE_DESC_NAME,
                        dadosGerais.getNome(), 1, false);
                srcDesc[2] = new SourceDescription(SourceDescription.SOURCE_DESC_EMAIL,
                        dadosGerais.getEmail(), 1, false);
                srcDesc[3] = new SourceDescription(SourceDescription.SOURCE_DESC_PHONE,
                        dadosGerais.getFone(), 1, false);
                srcDesc[4] = new SourceDescription(SourceDescription.SOURCE_DESC_LOC,
                        dadosGerais.getLoc(), 1, false);
                srcDesc[5] = new SourceDescription(SourceDescription.SOURCE_DESC_NOTE,
                        dadosGerais.getNotas(), 1, false);
                srcDesc[6] = new SourceDescription(SourceDescription.SOURCE_DESC_PRIV,
                        dadosGerais.getPriv(), 1, false);
                srcDesc[7] = new SourceDescription(SourceDescription.SOURCE_DESC_TOOL,
                        dadosGerais.getFerramenta(), 1, false);
                //definição das sessões RTP local (servidor) e remota (cliente)
                sessaoPorta += (10 * i);
                sessaoLocal[0] = new SessionAddress(InetAddress.getLocalHost(), portas
[i]);
                sessaoRemota = new SessionAddress(ip, sessaoPorta);
                rtpGerente[i].addSendStreamListener(this);
                rtpGerente[i].initialize(sessaoLocal, srcDesc, 0.05, 0.25, null);
                rtpGerente[i].addTarget(sessaoRemota);
                //o gerente de sessão RTP cria um stream a partir do data source da
mídia
                sendStream = rtpGerente[i].createSendStream(dataSource, i);
                //inicia a transmissão do stream
                sendStream.start();
```

```
//instancia os objetos de monitoração e adaptação da transmissão
                RTPAdaptador rtpAdaptador = new RTPAdaptador(this, servidor, rtpGerente
[i],
                      mediaLocator, sendStream, rtpFormatosAdequados, i, ParamQOSApp,
clienteSocket);
                RTPMonitor rtpMonitor = new RTPMonitor(servidor, rtpGerente[i],
rtpAdaptador, clienteSocket);
                //adiciona um ouvinte para eventos remotos (recebimento de pacotes RTCP
dos clientes)
                rtpGerente[i].addRemoteListener(rtpMonitor);
            }
            processador.start();
            enviaMensagemPorta(portas);
        } catch(...)
            ....consistências
        return true;
    }
```

Quadro 10 – Código do método transmiteMidia da classe RTPServidor

O processo de monitoração da transmissão e da coleta dos pacotes RTCP RR enviados pelo cliente pode ser visto no Quadro 11 que mostra um trecho de código da classe RTPMonitor. Esta classe implementa a interface *RemoteListener*, o que torna possível receber eventos dos participantes remotos em uma sessão RTP.

O método *update* é invocado quando um evento remoto ocorre e caso este evento seja uma instância de *ReceiverReportEvent*, o programa obtém os *feedbackReports* que representam os *reports blocks* dos pacotes RTCP. É a partir do *feedbackReport* que obtém-se os parâmetros de QoS da rede que demonstram como o receptor está recebendo os dados.

Além dos parâmetros de QoS obtidos nos RR é calculado a taxa de perdas ao longo da transmissão e determinado o nível de carga da rede. A taxa de perdas é calculada com base no parâmetro fração de perdas de pacotes e levando em consideração o peso que se dá a taxa de perdas anterior, este valor pode variar de 0 a 1. Após calculada a taxa de perdas é determinada o nível de carga da rede que pode ser descarrega, carregada ou congestionada. Os limites da taxa de perdas de pacotes podem ser parametrizados para determinar o estado da rede. Um exemplo seria de 0% a 2 % rede descarregada, a partir de 2% até 4% rede carregada e acima de 4% rede congestionada.

O processo de monitoração ocorre a cada intervalo de recepção de um pacote RTCP

RR que ocorre em média a cada 5 segundos. As informações obtidas e determinadas na monitoração são encaminhadas para o objeto *RTPAdaptador* da sessão, onde será feita a comparação entre os parâmetros de QoS da rede e os determinados pela aplicação para definir a execução ou não de uma ação adaptativa.

```
//método que recebe eventos remotos
public void update(RemoteEvent evento){
        //recebimento de pacote RTCP RR
        if (evento instanceof ReceiverReportEvent) {
            if (pausa == false){
                ReceiverReport receiverReport = ((ReceiverReportEvent) evento).getReport
();
                if( receiverReport != null) {
                    //obtém um objeto que representa o participante remoto
                    Participant participant = receiverReport.getParticipant();
                    if( participant != null) {
                        //obtém os reports block do pacote RTCP
                        Vector feedbackReports = receiverReport.getFeedbackReports();
                        //extrai os parâmetros de QoS relevantes
                        if (feedbackReports.size() > 0){
                            Feedback feedback = (Feedback)feedbackReports.get(0);
                            dadosQOS = new DadosQOS();
                            fracPacPerdidos = feedback.getFractionLost()/256.0;
                            long jitterRTP = feedback.getJitter();
                            jitter = rtpAdaptador.calculaJitter(jitterRTP);
                            taxaPerdas = calcularTaxaPerdas(fracPacPerdidos);
                            estadoRede = determinarEstadoRede(taxaPerdas);
                            enviaDadosQOS(feedback);
                            servidor.reportRTCP(rtpGerente, receiverReport, feedback,
jitter, taxaPerdas, estadoRede);
                            if (finalizaAdaptador == false){
                                //envia as informações para o módulo adaptador
                                rtpAdaptador.recebeParamsRede(this, estadoRede, jitter,
fracPacPerdidos);
                            }
                        }
                    }
                }

            } else{
                contPausa++;
                if (contPausa > 3){
                    contPausa = 0;
                    pausa = false;
                }
            }
        }

    }
```

```
 public double calcularTaxaPerdas(double fracPacPerdidos){
        if (dadosQOS.getPesoTaxa() != null){
            double pesoTaxa = (Double.parseDouble(dadosQOS.getPesoTaxa())/100);
            double taxa  = ((1 - pesoTaxa) * this.taxaPerdas) + pesoTaxa *
fracPacPerdidos;
            return taxa;
        }
        return 0;
    }

    public String determinarEstadoRede(double taxa){
        if ((dadosQOS.getRedeDesc() != null)  && (dadosQOS.getRedeCong() != null)){
            double redeDesc = (Double.parseDouble(dadosQOS.getRedeDesc())/100.0);
            double redeCong = (Double.parseDouble(dadosQOS.getRedeCong())/100.0);
            if (taxa  < redeDesc)
                return "descarregado";
            else
                if (taxa < redeCong)
                    return "carregado";
            return "congestionado";
        }
        return null;
    }
```

Quadro 11 – Trecho de código da classe RTPMonitor

O processo de adaptação através do método *recebeParamsRede* da classe *RTPAdaptador* recebe as informações encaminhadas pelo módulo de monitoração e faz algumas verificações e análise caso a mídia seja áudio:

a) necessidade de alteração do *buffer* no receptor: neste caso o parâmetro de QoS analisado é o *jitter* e se o valor obtido na rede for maior que o definido pela aplicação é enviado uma mensagem com o novo valor de *buffer* a ser ajustado pelo aplicativo cliente;

b) alteração do tipo de codificação da mídia (diminuição da taxa de transmissão): para esta ação adaptativa é verificado o nível de carga da rede e a fração de pacotes RTP perdidos. A ação é disparada caso a rede esteja “congestionada” e a fração de pacotes perdidos da rede seja maior que a definida pela aplicação. Caso haja a necessidade de alteração da codificação é necessário parar o *SendStream* atual e quando *Processor* da mídia estiver configurado definir o novo formato de codificação da mídia. Previamente foi definida a ordem dos formatos RTP em ordem decrescente de vazão nominal de banda de rede. Esta ordenação garante

que ocorra a diminuição na taxa de transmissão a cada execução da ação adaptativa. E por fim quando o *Processor* estiver no estado realizado o *RTPManager* da sessão cria um novo *SendStream* e envia-o à rede na mesma sessão RTP do *stream* original.

O processo de adaptação descrito anteriormente pode ser visto através de um trecho de código da classe *RTPAdaptador* no Quadro 12.

```
//método que recebe as informações de QoS
public void recebeParamsRede(RTPMonitor rtpMonitor, String estadoRede, double jitter
double fracPacPerdidos){
        this.rtpMonitor = rtpMonitor;
        if (tipo == "audio") {
            if ((jitter > jitterApp) && (bufferAtual < 1000))  {
                bufferAtual = bufferAtual + 100;
                enviaMensagemBUF(bufferAtual, jitter, jitterApp);
            }
        }
        RTPManager [] rtpGerentes =  rtpServidor.getRTPGerente();
        if (!(rtpGerentes.length > 1)){
            if (tipo == "audio") {
                if (estadoRede == "congestionado"){
                    if (fracPacPerdidos > fracPacPerdidosApp){
                        //diminuir taxa de transmissão permutar stream
                        if (indiceFormatoAtual < (((Format []) rtpFormatosFaixa.get
(indiceFaixa)).length)-1) {
                            enviaMensagemTAX(fracPacPerdidos, fracPacPerdidosApp);
                            alteraFormatoStream();
                            rtpMonitor.pausaMonitor();
                        } else {
                            rtpMonitor.finalizaAdaptacao();
                        }
                    }
                }
            }
        }
    }

   //método que inicia a ação de adaptação
   public boolean alteraFormatoStream(){
        try {
            //para o stream antigo
            sendStream.stop();
            //cria um processador para a mídia
            processador = Manager.createProcessor(mediaLocator);
            processador.addControllerListener(new ProcessaEventos());
            processador.configure();
        } catch(IOException e){
            JOptionPane.showMessageDialog(null, e.getMessage(), "RTPServidor",
JOptionPane.ERROR_MESSAGE);
            enviaMensagemErro("Mensagem do Servidor: " + e.getMessage());
            return false;
        } catch(NoProcessorException e){
            JOptionPane.showMessageDialog(null, e.getMessage() , "RTPServidor",
JOptionPane.ERROR_MESSAGE);
            enviaMensagemErro("Mensagem do Servidor: " + e.getMessage());
            return false;
        }
        return true;
    }
```

```
//método que define o novo formato do stream
   public boolean setFormatoSaida(){
        processador.setContentDescriptor( new ContentDescriptor
(ContentDescriptor.RAW_RTP));
        TrackControl faixas [] = processador.getTrackControls();
        Format rtpFormatos[];
        for (int i = 0 ; i < faixas.length; i++){
            if (faixas[i].isEnabled()){
                //obtém os formatos adequados que não foram utilizados
                rtpFormatos = (Format [])rtpFormatosFaixa.get(i);
                if (rtpFormatos.length > 0){
                    if (i == indiceFaixa){
                        indiceFormatoAtual++;
                        //seta o novo formato para a faixa
                        faixas[i].setFormat(rtpFormatos[indiceFormatoAtual]);
                    }
                }
            }
        }
        return true;
    }

    //método que cria e envia o novo stream à rede
    public boolean permutaStream(){
        DataSource dataSource = processador.getDataOutput();
        if (dataSource == null){
            return false;
        }
        try {
            //cria um novo stream
            sendStream = rtpGerente.createSendStream(dataSource, 0);
            sendStream.start();
        } catch(IOException e){
            JOptionPane.showMessageDialog(null, e.getMessage() , "RTPServidor",
JOptionPane.ERROR_MESSAGE);
            enviaMensagemErro("Mensagem do Servidor: " + e.getMessage());
            return false;
        } catch(UnsupportedFormatException e){
            JOptionPane.showMessageDialog(null, e.getMessage() , "RTPServidor",
JOptionPane.ERROR_MESSAGE);
            enviaMensagemErro("Mensagem do Servidor: " + e.getMessage());
            return false;
        }
        processador.start();
        return true;
    }

    //classe tratadora dos eventos do Processor
        private class ProcessaEventos extends ControllerAdapter{

            public void configureComplete(ConfigureCompleteEvent e){
                setFormatoSaida();
                processador.realize();
            }

            public void realizeComplete(RealizeCompleteEvent e){
                permutaStream();
            }

            public void endOfMedia(EndOfMediaEvent e){
                encerraTransmissao();
            }
```

Quadro 12 – Trecho de código da classe RTPAdaptador

3.3.1.3 Implementação do cliente RTP

A principal classe do cliente RTP é a *RTPCliente*, um objeto desta classe implementa a interface JMF *ReceiveStreamListener*, o que torna possível receber eventos remotos relacionados a recepção de mídia *streaming* em uma sessão RTP. Um trecho de código da classe *RTPCliente* pode ser visto no Quadro 13.

Através do método *update* se pode capturar um *ReceiveStreamEvent*, o programa está preparado para tratar os eventos *NewReceiveStreamEvent*, *RemotePayloadChangeEvent* e *ByeEvent*.

NewReceiveStreamEvent é um evento gerado quando um novo *stream* RTP está sendo recepcionado na sessão RTP. Quando um evento deste tipo ocorre é criado uma *Player* para o controle da reprodução da mídia. A *interface Player* e o *stream* são utilizados na criação de uma instância de um objeto da classe *PlayerWindow* que provê controles e componentes visuais próprios para a visualização e reprodução da mídia.

O evento *RemotePayloadChangeEvent* ocorre quando ao longo de uma transmissão o emissor dos dados altera o formato RTP do *streaming*. Nestes casos é necessário fechar o *Player* atual e efetuar a sua permuta.

Um evento *ByeEvent* é enviado pelo emissor dos dados quando este encerra a sessão RTP. Todo objeto *PlayerWindow* é finalizado, resultando no encerramento da reprodução da mídia.

```
//método que recebe eventos remotos
public synchronized void update(ReceiveStreamEvent evento){
        RTPManager rtpGer = (RTPManager) evento.getSource();
        Participant participante = evento.getParticipant();
        ReceiveStream stream = evento.getReceiveStream();
```

```
        //novo stream recepcionado
        if (evento instanceof NewReceiveStreamEvent){
            try {
                stream = ((NewReceiveStreamEvent)evento).getReceiveStream();
                linha = "\n\nNovo stream identificado na sessão RTP";
                linha += "\nSSRC: " + formataLong(stream.getSSRC());
                DataSource dataSource = stream.getDataSource();
                PushBufferDataSource bufferDatasource = ( PushBufferDataSource )
dataSource ;
                PushBufferStream bufferStream[] = bufferDatasource.getStreams();
                for (int i = 0 ; i < bufferStream.length ; i++){
                    Format formato = bufferStream[i].getFormat();
                    if (formato instanceof AudioFormat) {
                        formatoAudioAntigo = formato;
                        linha+= "\nStream de áudio ( " + formatoAudioAntigo + " )";
                    } else {
                        linha+= "\nStream de vídeo ( " + formato + " )";
                    }
                }
                frmAplicacao.adicionaLinha(linha);
                RTPControl controle  = (RTPControl)dataSource.getControl
("javax.media.rtp.RTPControl");
                //cria Player para controle do stream
                Player player = Manager.createPlayer(dataSource);
                if (player == null){
                    frmAplicacao.adicionaLinha("\nGerado player nulo");
                    return;
                }
                player.addControllerListener(this);
                //cria objeto que irá reproduzir a mídia (componente visual)
                PlayerWindow playerWindow = new PlayerWindow(this, player, stream);
                playerWindows.addElement(playerWindow);
                player.realize();
            } catch (IOException e) {
                frmAplicacao.adicionaLinha("\nProblemas no acesso ao DataSource");
                JOptionPane.showMessageDialog(null, e.getMessage(), "RTPCliente",
JOptionPane.INFORMATION_MESSAGE);
            } catch(NoPlayerException e){
                frmAplicacao.adicionaLinha("\nNão há player disponível para a reprodução
da mídia");
                JOptionPane.showMessageDialog(null, e.getMessage(), "RTPCliente",
JOptionPane.INFORMATION_MESSAGE);
            }

        //mudança no formato da mídia
        } else if (evento instanceof RemotePayloadChangeEvent) {
            long valor;
            String ssrc;
            valor = stream.getSSRC();
            ssrc = formataLong(valor);
            Time time;
            if (playerWindows != null) {
                permutar = true;
                for (int i = 0 ; i < playerWindows.size(); i++){
                    PlayerWindow playerWindow = (PlayerWindow)playerWindows.elementAt
(i);
```

```
if (playerWindow != null){
                        //ao fechar o Player é iniciado o processo de permuta de Player
                        playerWindow.fechar();
                        if (playerWindow.getStream().getSSRC() == valor) {
                            time = playerWindow.getTempoMidia();
                            DataSource dataSource = playerWindow.getDataSource();
                            PushBufferDataSource bufferDatasource = (
PushBufferDataSource ) dataSource ;
                            PushBufferStream bufferStream[] =
bufferDatasource.getStreams();
                            for (int j = 0 ; j < bufferStream.length ; j++){
                                Format formato = bufferStream[i].getFormat();
                                String linha;
                                linha =  "Tempo: " + time.getSeconds();
                                linha += "\nSSRC: " + ssrc;
                                linha += "\nTipo: Alteração do tipo da mídia";
                                if (formato instanceof AudioFormat){
                                    linha += "\nCodificação antiga: " +
formatoAudioAntigo;
                                    formatoAudioAntigo = bufferStream[i].getFormat();
                                    frmAplicacao.adicionaLinha("\nMudança de Payload de
áudio, novo formato " +  bufferStream[i].getFormat());
                                }
                                linha += "\nCodificação Nova: " + bufferStream
[i].getFormat();
                                linha += "\nMotivo: rede congestionada";
                                linha += "\n        Fração pacotes perdidos definidos
pela aplicação (%): " +  (fracPacPerdidosApp * 100);
                                linha += "\n        Fração pacotes perdidos da rede (%):
" + (fracPacPerdidos * 100)  + "\n\n";
                                DlgAdaptacao  dlgAdaptacao = dlgSessao.getDlgAdaptacao
();
                                if (dlgAdaptacao != null) {
                                    dlgAdaptacao.adicionaLinha(linha);
                                }
                            }
                        }
                    }
                }
            }

        } else if (evento instanceof ByeEvent){
            frmAplicacao.adicionaLinha("\n\nMensagem BYE enviado pelo servidor (término
da sessão)");
            PlayerWindow playerWindow = buscaPlayer(stream);
            if (playerWindow != null) {
                playerWindow.fechar();
                playerWindows.removeElement(playerWindow);
            }
            fechar(rtpGer);
        }
    }
```

Quadro 13 – Trecho de código da classe RTPCliente

3.3.2 Operacionalidade da implementação

Esta seção descreve a funcionalidade do protótipo através de um estudo de caso onde o

cliente RTP solicita ao servidor a transmissão do arquivo de áudio Carmina_Burana.wav em uma rede local. A Figura 17 ilustra o ambiente de rede do estudo de caso.

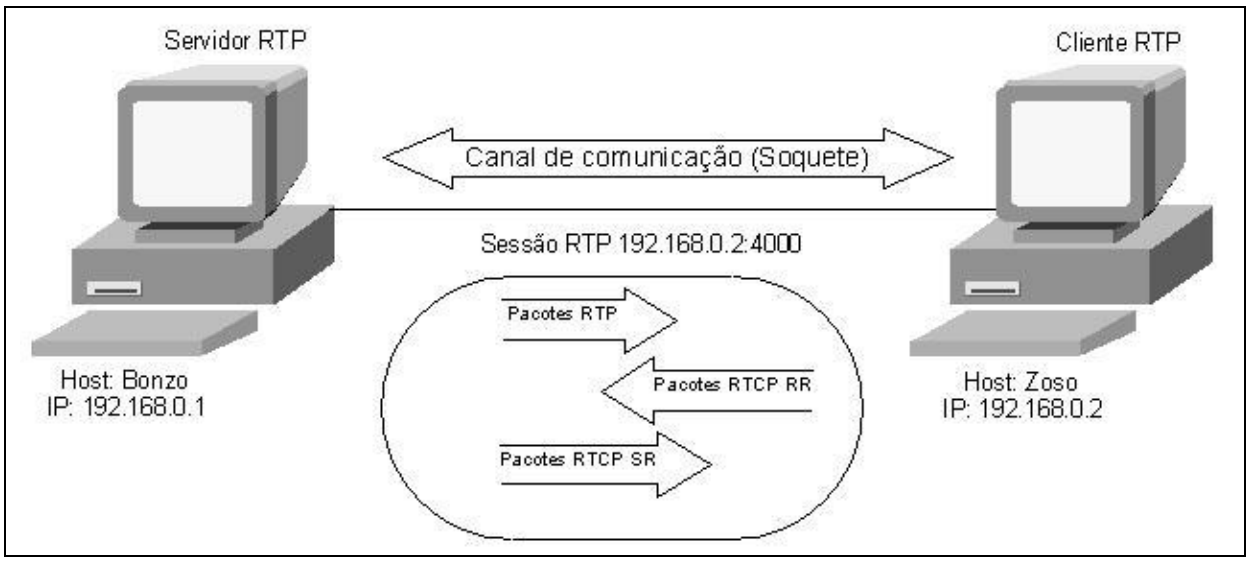


Figura 17 – Ambiente de rede do estudo de caso

3.3.2.1 Configurações e definições do servidor RTP

O servidor RTP está localizado em uma rede local com o seguinte endereço IP 192.168.0.1. A tela principal da ferramenta pode ser vista na Figura 18 e mostra os detalhes de conexão dos clientes quando há sessões RTP ativas. Estas informações são constituídas pelo *host* , IP, marcação da hora de conexão, endereço da sessão RTP e o nome do arquivo solicitado.

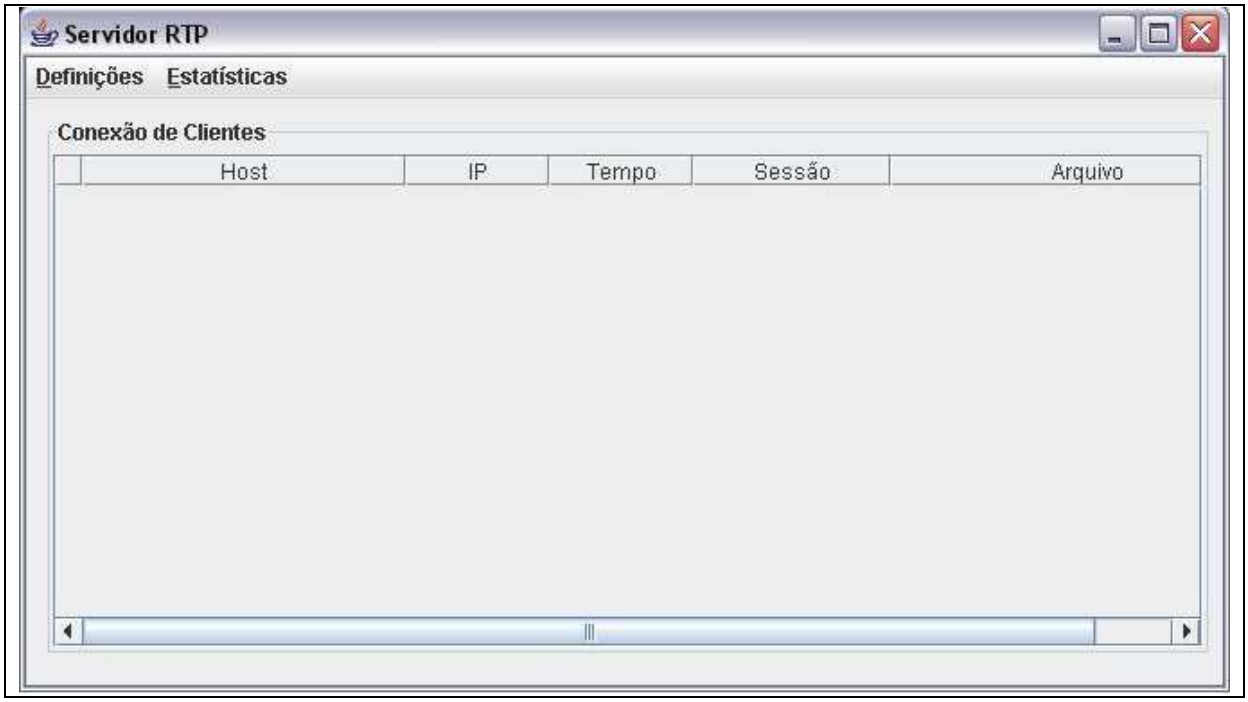


Figura 18 – Tela principal do servidor RTP

Para o servidor tornar-se apto a receber conexões e transmitir mídia com qualidade de serviço é necessária à definição dos parâmetros de QoS da aplicação e da definição de informações relevantes para os módulos de monitoração e adaptação do protótipo. Outros parâmetros devem ser preenchidos como o diretório de localização da mídia no servidor e os campos de identificação do servidor que irão compor os relatórios RTCP SDES enviados ao longo de uma sessão RTP.

Para parametrizar as informações de localização e de identidade do servidor o usuário deve clicar no menu “Definições” e após no item “Configurações Gerais”. O protótipo irá apresentar a tela de configurações gerais do servidor RTP, como pode ser visto na Figura 19.

Figura 19 – Tela de configurações gerais do servidor RTP

Os parâmetros de QoS que devem ser informados pelo usuário são referentes a mídia de áudio e são os seguintes: valor máximo da variação do atraso (*jitter*) em milissegundos, máximo da fração de pacotes perdidos e o limite máximo para a vazão de banda, medido em Kbps. Este último parâmetro é utilizado para definir o formato inicial da mídia, cuja vazão nominal de banda atenda as expectativas da aplicação.

Para o módulo de monitoração efetuar o cálculo da taxa de perdas de pacotes ao longo da transmissão de uma sessão RTP e determinar o nível de carga da rede é necessário que o usuário informe o peso da taxa de perdas anterior, o limite máximo para rede descarregada e o limite máximo para a rede congestionada.

Para efetuar o preenchimento das informações de QoS da aplicação e das definições do módulo de monitoração é necessário acessar o menu “Definições” e clicar no item “Definir QoS”, o protótipo apresentará a tela denominada “Parâmetros de QoS”, como pode ser observado na Figura 20.

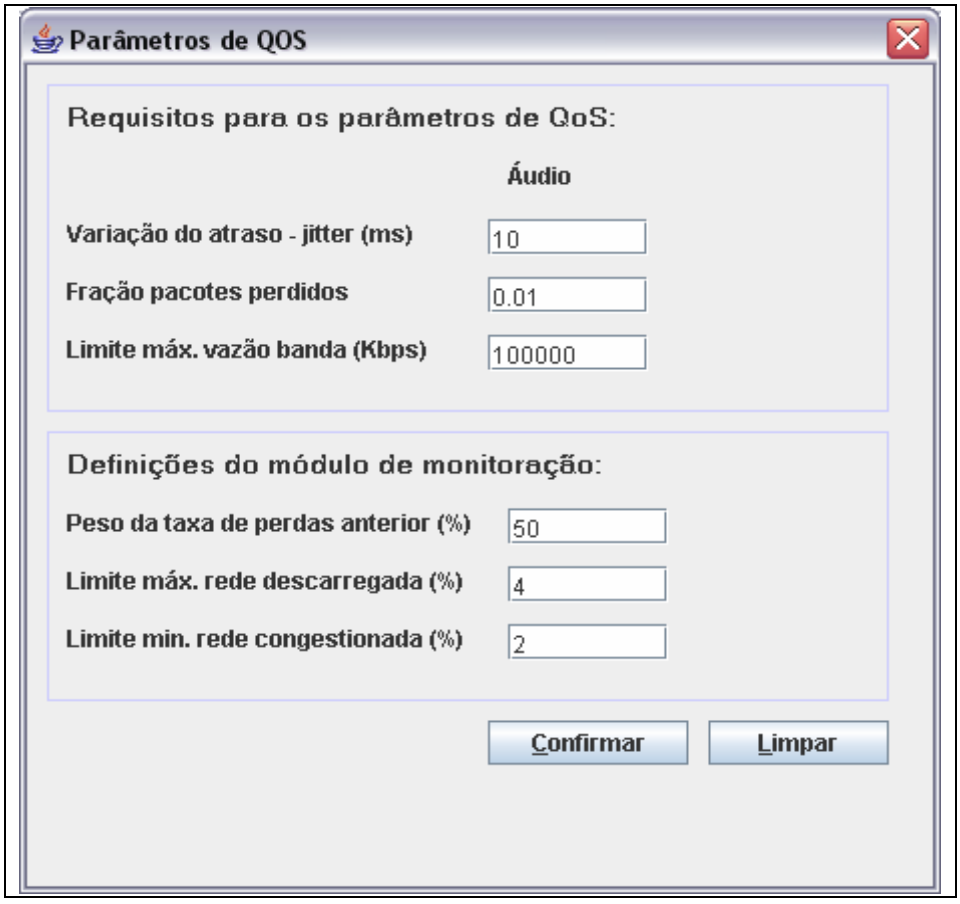


Figura 20 – Tela de definição dos parâmetros de QoS da aplicação

3.3.2.2 Informações de requisição do cliente RTP

O cliente RTP utilizado neste estudo de caso encontra-se na mesma rede local do servidor RTP e seu endereço IP é 192.168.0.2. Para solicitar a transmissão de um arquivo armazenado no servidor através de uma sessão RTP é necessário preencher algumas

informações, tais como: endereço IP do servidor, tipo da mídia (áudio ou vídeo, necessário para definir a quantidade de sessões RTP, sendo uma para áudio e duas para vídeo), nome do arquivo, endereço e porta inicial da sessão RTP. A Figura 21 mostra a tela de solicitação de uma sessão RTP, que pode ser acessada através do menu “Arquivo” e item “Abrir Sessão RTP”. Os campos preenchidos já estão adequados para este estudo de caso e para efetivar a solicitação do arquivo o usuário deve pressionar no botão “Iniciar”.

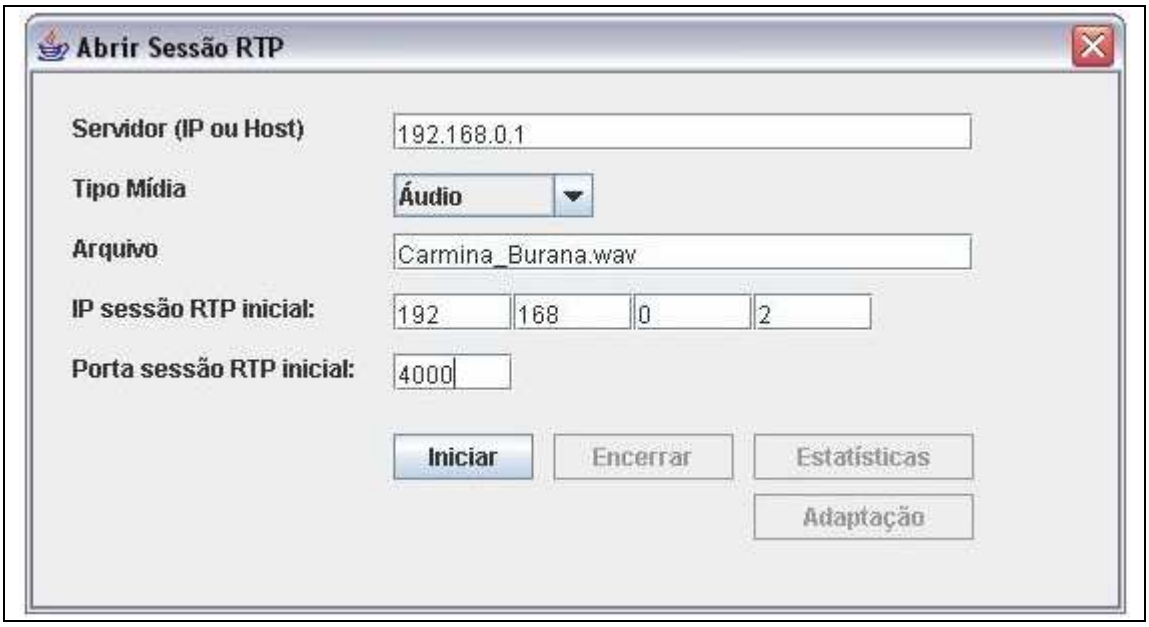


Figura 21 – Tela de solicitação de arquivo em uma sessão RTP

3.3.2.3 Processo de transmissão e recepção da mídia

Ao receber a solicitação de uma mídia o servidor RTP adiciona as informações de solicitação e do cliente na tabela de conexões na tela principal do protótipo, conforme ilustra a Figura 22.

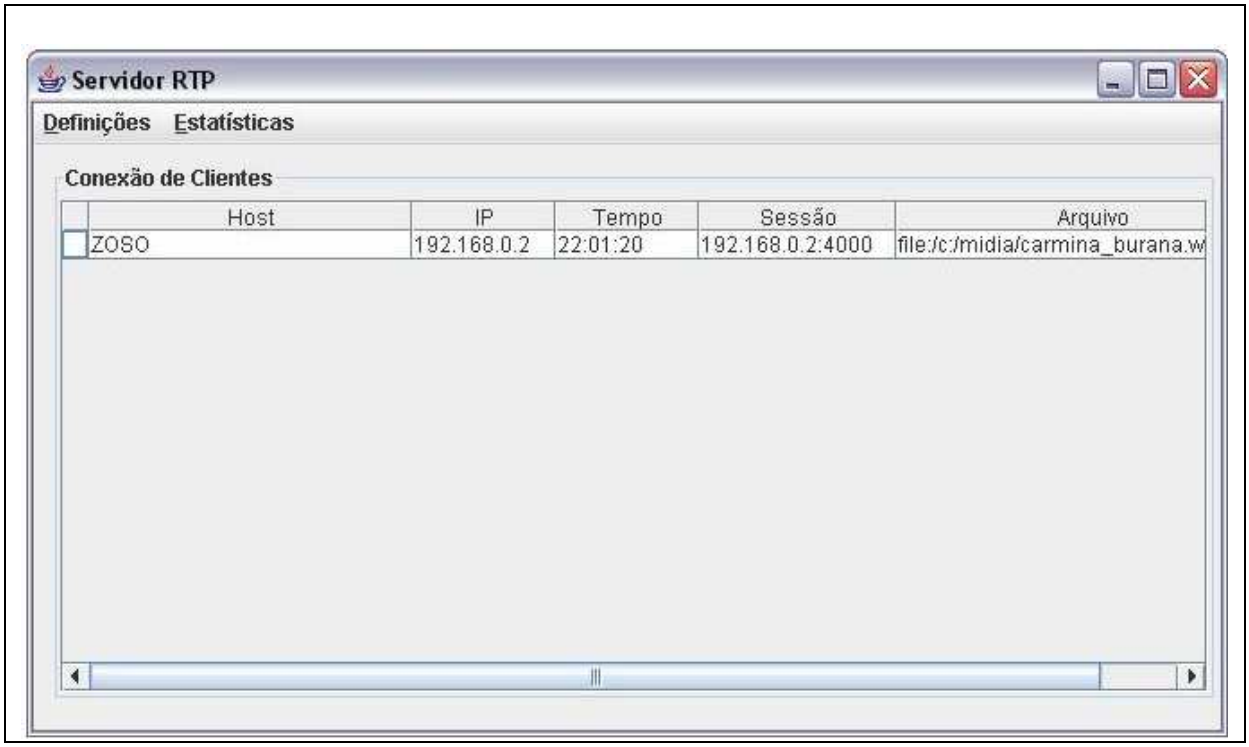


Figura 22 – Tabela de conexões de cliente

Após processada a requisição e estabelecida a sessão RTP o servidor está apto a receber as informações de *feedback* que são encontradas nos pacotes RTCP RR enviadas pelo cliente RTP. Além destas informações o servidor registra algumas estatísticas de transmissão específicas para cada sessão, como por exemplo, o total de bytes enviados e de pacotes RTP e RTCP enviados. A taxa de perdas de pacotes calculada, assim como o nível de carga corrente da rede também é registrada. Todos estes dados são atualizados a cada recepção de um pacote RTCP RR. A Figura 23 ilustra a tela de estatísticas de QoS dos participantes, que pode ser acessada pelo menu “Estatísticas” e item “QoS participantes” do servidor RTP. Para cada participante (cliente RTP) é adicionada uma aba que é identificada pelo CNAME do participante.

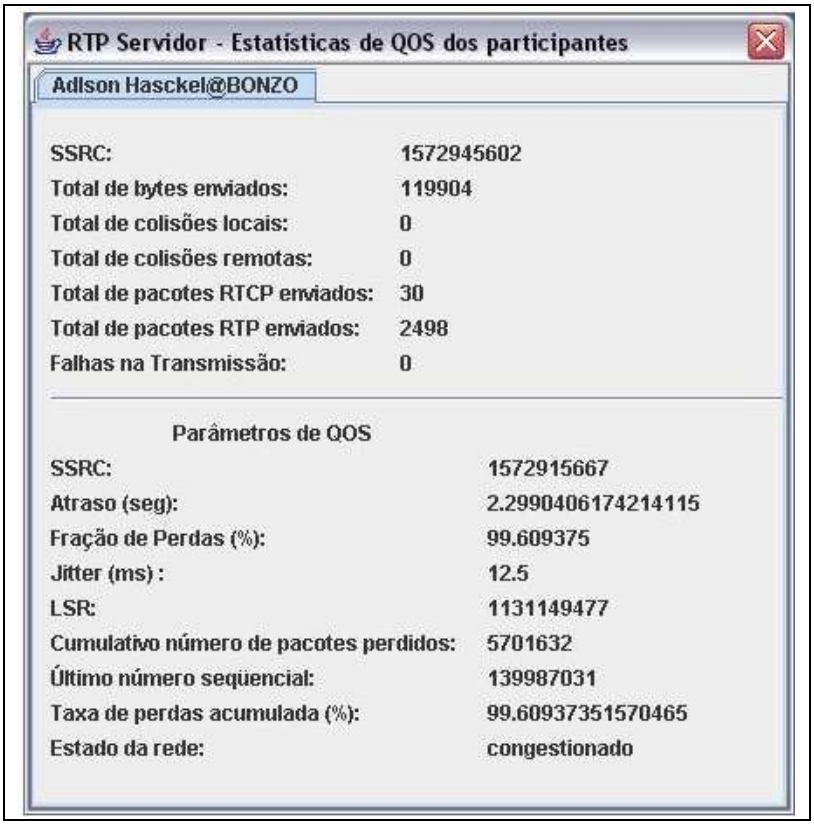


Figura 23 – Tela de estatísticas de QoS dos participantes

Quando o aplicativo cliente recebe o *stream* na sessão RTP é criado um *player* para efetuar a reprodução da mídia. Através do *player* pode-se também acessar algumas propriedades da transmissão e da codificação da mídia, como é o caso do tipo de codificação e vazão de banda de rede consumida. A Figura 24 mostra o *player* gerado para este caso de uso.

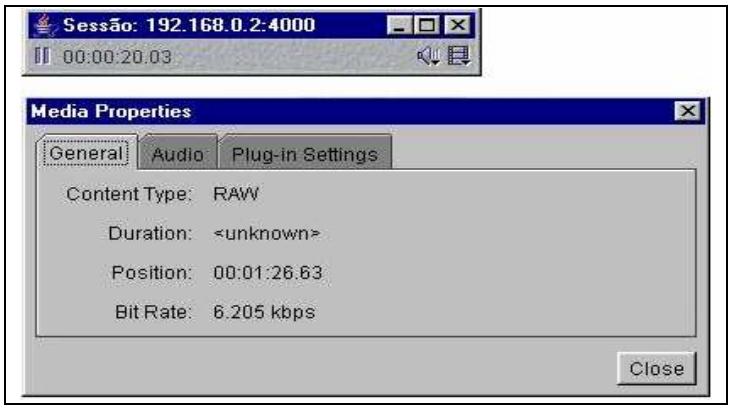


Figura 24 – *Player* e propriedades da mídia

Na tela principal do cliente RTP é registrado as configurações da sessão RTP em

execução como os endereços da sessão local e remota, o tipo da codificação do *stream* recebido e a marcação de tempo de chegada de cada SR RTCP enviado pelo servidor. A tela principal do cliente RTP com uma sessão ativa pode ser vista através da Figura 25.

Figura 25 – Tela principal do cliente RTP com uma sessão ativa

O usuário do aplicativo cliente poderá acessar as informações estatísticas de recepção e de QoS através do botão “Estatísticas”. Estas estatísticas estão divididas em três grupos: globais, sessão e qualidade de serviço.

As estatísticas globais trazem dados quantitativos de recepção na sessão como, por exemplo, os totais de bytes recebidos, pacotes RTP e RTCP.

Nas estatísticas de sessão é registrado o participante local e remoto da sessão, ou seja, o cliente e o servidor RTP respectivamente, assim como os campos de descrição dos pacotes SDES do participante remoto. Este último servindo como forma de identificação do servidor RTP.

E o último grupo de estatísticas é o de qualidade de serviço, que traz as mesmas informações apresentadas no servidor RTP no que diz respeito aos parâmetros de QoS da rede. A Figura 26 ilustra a tela de estatísticas de recepção.

Figura 26 – Tela de estatísticas de recepção

Outra funcionalidade do cliente RTP é o *log* de adaptação que pode ser acessado enquanto uma sessão estiver ativa através do botão “Adaptação”. Quando o servidor RTP efetua algum processo de adaptação na sessão, seja notificação para alteração de *buffer* ou alteração da codificação da mídia a ferramenta registra algumas informações úteis para o usuário. Estas informações são: o tempo do *player* em que ocorreu a adaptação, o SSRC da fonte emissora, o tipo de adaptação, o tamanho do *buffer* antigo ou a especificação da codificação antiga, o tamanho do *buffer* novo ou a especificação da codificação nova e o

motivo da adaptação. A Figura 27 ilustra a tela de *log* de adaptação para o caso de uso em questão. O *log* registra três adaptações ocorridas na transmissão, aos 18 segundos de reprodução foi alterado o tamanho do *buffer* de 500 para 600 ms e também o formato da codificação que passou para ULAW/RTP, e por fim aos 20 segundos a alteração do tamanho do buffer de 600 para 700 ms.

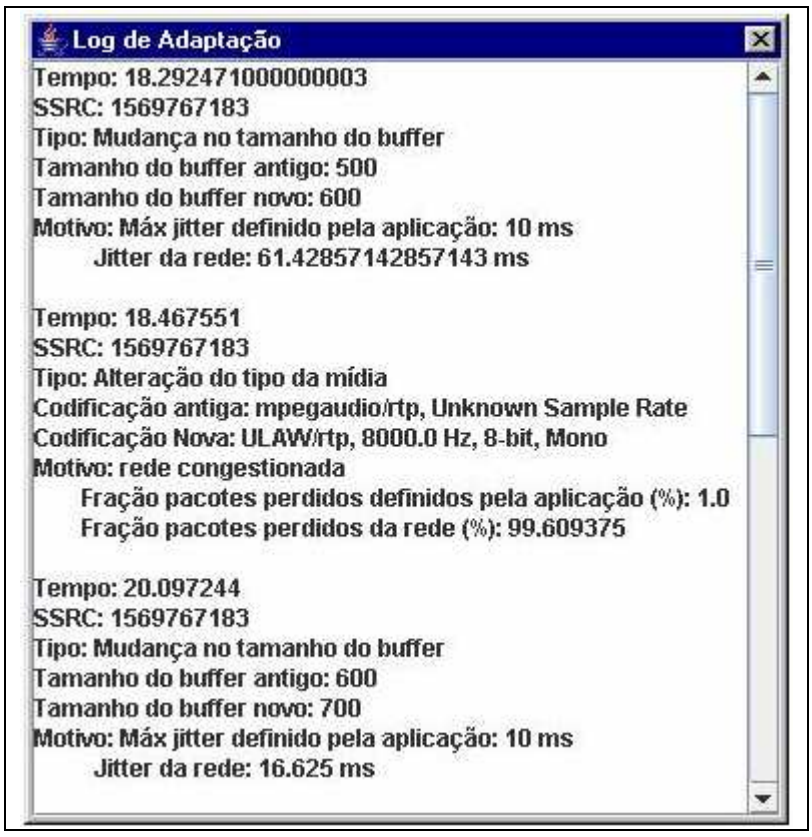


Figura 27 - Tela de *log* de adaptação

3.4 RESULTADOS E DISCUSSÃO

Para validar o protótipo foram realizados alguns testes simulando o estudo de caso realizado na seção anterior, com as mesmas definições dos parâmetros de QoS, apenas alterando o parâmetro variação do atraso (*jitter*) para 20 ms.

Inicialmente foi efetuada a transmissão do arquivo na sessão RTP com a rede normal, sem nenhuma carga adicional. A transmissão possuía a seguinte configuração inicial: formato RTP DVI/rtp 22.05 KHz (88.2 Kbps) com tamanho de *buffer* 400 ms no receptor.

Com este cenário houve apenas a necessidade de ajustar o tamanho do *buffer* devido a um pico na variação do atraso aos 2 segundos de recepção. Devido ao valor do *jitter* encontrado ser de 21.49 ms o tamanho do *buffer* foi aumentado para 500 ms, o que manteve a variação do atraso abaixo de 20ms, com valores entre 10 ms e 15 ms, ao longo da transmissão.

Posteriormente para simular a transmissão da mídia numa rede com tráfego, foram gerados pacotes UDP na estação servidora e enviados repetidamente à estação do cliente. Tendo em vista este cenário houve a necessidade de aplicar as políticas de adaptação referentes à mudança do formato RTP e alteração do tamanho do *buffer*. A Tabela 2 mostra os resultados obtidos pelo processo de alteração do formato RTP e a Tabela 3 ilustra o processo de alteração do tamanho do *buffer* no receptor. A transmissão iniciou com o formato DVI/RTP 22.05 KHz.

Tabela 2 – Resultados do processo de mudança do formato RTP

Tempo(seg.)	**Fração de pacotes perdidos anterior (%)**	**Formato RTP novo**	**Fração de pacotes perdidos atual (%)**	**Vazão de banda anterior (Kbps)**	**Vazão de banda atual (Kbps)**
1	73,82	ULAW/RTP	0,00	91	62
26	54,37	DVI/RTP 11.025 KHz	0,00	62	42

Tabela 3 – Processo de alteração do tamanho do *buffer* no receptor

Tempo (seg.)	***Jitter* da rede (ms)**	**Tamanho anterior do *buffer* (ms)**	**Tamanho novo do *buffer* (ms)**	**Novo valor do *jitter* (ms)**
1	78,63	500	600	15,44
6	28,75	600	700	12,14
25	149,00	700	800	18,23
53	26,40	800	900	13,79

Como o nível de carga da rede estava muito grande as políticas de adaptação foram disparadas várias vezes ao longo da transmissão. A mudança do formato RTP contribuiu na diminuição do volume de dados na rede, o que resultou na diminuição da fração de pacotes perdidos. Neste teste foi necessária apenas duas modificações no formato RTP, entretanto em casos extremos a taxa de transmissão pode ser reduzida até a vazão nominal do último formato RTP na hierarquia de formatos baseada na vazão de rede. Além dos formatos vistos anteriormente completam esta hierarquia os formatos DVI/RTP 8 KHz (32 Kbps), GSM (13Kbps) e G723 (6,4 Kbps).

Com relação ao processo de mudança do tamanho do *buffer* pode-se constatar que quando a rede é submetida a um grande tráfego, ocorre uma oscilação muito grande no *jitter* o que resulta na contínua execução da adaptação. O aumento do tamanho do *buffer* manteve temporariamente o *jitter* da rede dentro dos limites definido pela aplicação.

A definição de valores adequados para os parâmetros de QoS da aplicação garante êxito na execução das políticas de adaptação, entretanto a definição de valores inconsistentes pode inviabilizar o processo de adaptação da transmissão. Neste sentido os valores utilizados nos testes mostraram-se adequados.

Com relação aos trabalhos correlatos apresentados e o protótipo desenvolvido destaca-se as seguintes características:

a) Em Lunardi e Dotti (2001) a API especificada oferece qualidade de serviço para áudio e vídeo, baseada nas informações de *feedback* do protocolo RTCP. Este trabalho é destinado ao uso de desenvolvedores em aplicações multimídia que queiram fornecer qualidade de serviço, não sendo um produto final. Já o protótipo desenvolvido é uma aplicação multimídia destinada ao usuário final, que já embuti as funcionalidades relativas à qualidade de serviço de áudio. Uma vantagem do protótipo desenvolvido é a possibilidade de se definir a vazão de banda de rede

inicial da transmissão;

b) No trabalho de Gomes et al. (2000) a abordagem utilizada para efetuar as adaptações da mídia na sessão RTP é a permuta da mídia, ou seja, há a necessidade de manterem-se várias cópias do mesmo arquivo em diferentes níveis de qualidade. Esta permuta tem o objetivo de manter a semântica dos documentos utilizados no ensino a distância. Já no trabalho desenvolvido se objetivou a qualidade de serviço na transmissão e a abordagem de adaptação da mídia consiste em efetuar adaptações na transmissão e no *buffer*, mas sempre sob a mesma mídia;
c) Na ferramenta desenvolvida por Tschöke (2001), o *streaming* criado foi oriundo da placa de vídeo, ao contrário da ferramenta desenvolvida onde à mídia encontra-se previamente armazenada na máquina do aplicativo servidor. Outra diferença está no escopo do trabalho, pois a ferramenta citada não provê mecanismos de monitoração e adaptação de qualidade de serviço;
d) Com relação ao trabalho desenvolvido por Koliver (2001), a política de adaptação da taxa de transmissão é baseada em um controlador nebuloso que calcula a nova taxa de *bits* a ser enviada na rede. Já no presente trabalho a abordagem utilizada para a mudança da taxa de transmissão é a mudança do formato RTP, onde a nova taxa de transmissão será a vazão de banda de rede nominal do novo formato.

4 CONCLUSÕES

O protótipo de ferramenta desenvolvido é uma alternativa na transmissão de *streaming* de áudio, com o diferencial de proporcionar mecanismos dinâmicos de maximização da qualidade de serviço e de utilizar protocolos adequados para a transmissão multimídia, como é o caso do RTP e RTCP.

A eficiência da ferramenta está condicionada à definição de parâmetros de QoS adequados no aplicativo servidor, sendo assim o conhecimento de características da mídia e de noções de qualidade de serviço em redes de computadores é pré-requisito básico para a utilização do servidor desenvolvido. Já para a utilização do cliente RTP, basta o usuário ter o conhecimento sobre o conceito de sessão RTP.

Os resultados obtidos com a integração dos processos de monitoração e adaptação de QoS mostraram-se satisfatórios nos testes realizados com o estudo de caso. As ações adaptativas quando acionadas obtiveram êxito ao regularizar os parâmetros de QoS aos níveis definidos pela aplicação. Apenas em casos de extremo congestionamento na rede, o efeito da adaptação de ajuste do *buffer* não se mostrou durável, pois devido a grande oscilação do *jitter*, a política teve que ser aplicada várias vezes até o limite máximo do *buffer* (definido em 1000 ms), porém deve ser respeitado este limite do tamanho do *buffer* para que se evite a perda de pacotes por espera na liberação do *buffer*.

Os objetivos e requisitos previamente definidos foram alcançados, entretanto a intenção era fornecer mecanismos de qualidade de serviço para áudio e também para vídeo, sendo este último inicialmente implementado, porém não mantido ao final do trabalho. O processo de maximização da qualidade de serviço de vídeo não foi mantido, pois a abordagem de alterar o formato RTP para diminuir a taxa de transmissão fez com que a qualidade da imagem do vídeo diminuísse a ponto de tornar o vídeo incompreensível. Por exemplo, a

alteração do formato RTP de MPEG para H.263 proporcionou a diminuição na taxa de transmissão, entretanto acarretou uma perda considerável na qualidade da imagem do vídeo, tornando-a fosca e sem nitidez. Por este motivo apenas é realizado a transmissão de vídeo, sem nenhum mecanismo de QoS. Para contornar este problema, poderia ser feitos ajustes nas propriedades da mídia, como por exemplo, mudar a quantidade de *frames* por segundo ou alterar cores e luminosidade. Porém estas adaptações são classificadas como qualidade de serviço da aplicação, o que não era o objetivo deste trabalho que visa fornecer qualidade de serviço na transmissão, ou seja, na camada de rede.

A vantagem do protótipo está em se adaptar às reais condições da rede para efetuar a transmissão de áudio, fazendo com que o servidor RTP tenha mecanismos de controle na taxa de *bits* enviados à rede.

O protótipo desenvolvido apresenta algumas limitações, descritas nos itens abaixo:

a) não provê mecanismos de qualidade de serviço para vídeo;
b) devido a limitação da API JMF, não efetua a transmissão de arquivos de áudio MPEG *layer* 3 (MP3);
c) para a reprodução de vídeo são abertos dois *players*, um para a sessão de áudio e outro para a sessão de vídeo, ao invés de apenas um player fazendo a integração das mídias;
d) quando alterado o formato RTP da mídia, o *player* deve ser reiniciado;
e) a interface do protótipo de ferramenta desenvolvido não pode ser utilizado para transmissão de mídia na *web*, porém as técnicas utilizadas neste trabalho são totalmente aplicáveis para este tipo de trasmissão.

4.1 EXTENSÕES

Como extensões para este trabalho sugerem-se:

a) fornecer mecanismos de qualidade de serviço na transmissão de vídeo;

b) aplicar outras ações adaptativas da transmissão;

c) adequar o protótipo para utilização na *web*, onde os requisitos da qualidade de serviço são mais críticos.

d) transmitir mídia capturada de outras fontes como por exemplo microfones e placas de vídeo;

e) aplicar medidas de adaptação da qualidade de serviço da aplicação na mídia;

f) utilização de outras tecnologias ou APIs multimídia para a implementação.

REFERÊNCIAS BIBLIOGRÁFICAS

COSTA, Daniel G. **Protocolos de transporte IETF RTP e RTCP**. Natal, 2004. Disponível em: <http://www.pop-rn.rnp.br/videoconf/textos/RTP_RTCP.pdf>. Acesso em: 9 abr. 2005.

DEITEL, Harvey M.; DEITEL, Paul J. **Java:** como programar. Tradução Carlos A. L. Lisboa. 4. ed. Porto Alegre: Bookman, 2003.

FURLAN, Jose Davi. **Modelagem de objetos através da UML – The Unified Modeling Language.** São Paulo: Makron, 1998.

GOMES, Roberta L. et al. Avaliação de um sistema distribuído de criação e apresentação de documentos multimídia. In: SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES, 18., 2000, Belo Horizonte. **Anais...** Belo Horizonte: UFMG, 2000. p. 489-504.

GOULART, Rudinei. **Utilização de metadados no gerenciamento de acesso a servidores dew**. 1998. 102 f. Tese (Mestrado em Ciências) – Instituto de Ciências Matemáticas de São Carlos, Universidade de São Paulo, São Carlos. Disponível em: <http://www.teses.usp.br/teses/disponiveis/55/55134/tde-29012001-160200>. Acesso em: 24 ago. 2005.

HELD, Gilbert. **Voice and data internetworking**. New York: McGraw-Hill, 2000.

IETF - RFC 1889. **RTP:** A Transport Protocol for Real-Time Applications. Berkeley, 1996. Disponível em: <http://www.ietf.org/rfc/rfc1889.txt?number=1889>. Acesso em dez. 2005.

KOISTINEN, Tommi. **Protocol overview:** RTP and RTCP. Helsinki, 2005. Disponível em: < <http://tct.hut.fi/opetus/s38130/k99/presentations/4.pdf >. Acesso em: 9 nov. 2005.

KOLIVER, Cristian. **Uma abordagem para adaptação de QoS baseada em controle nebuloso**. 2001. 76 f. Tese (Doutorado em Engenharia Elétrica) – Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal de Santa Catarina, Florianópolis. Disponível em: <http://www.dein.ucs.br/profs/ckoliver/papers/allchapters.ps.gz>. Acesso em: 10 abr. 2005.

KUROSE, James F.; ROSS, Keith W. **Redes de computadores:** uma nova abordagem. Tradução Arlete S. Marques. São Paulo: Adisson Wesley, 2003.

LIESENBORGS, Jori. **Voice over IP networked virtual environments**. Maastricht, 2000. Disponível em: <http://research.edm.luc.ac.be/jori/thesis/onlinethesis/contents.html>. Acesso em: 9 nov. 2005.

LUNARDI, Sediane C.; DOTTI, Fernando L. Uma camada de adaptação à qualidade de serviço na internet para aplicações multimídia. In: SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES, 19., 2001, Florianópolis. **Anais...** Florianópolis: UFSC, 2001. p. 17-32.

MELO, Edison T. L. **Qualidade de serviço em redes IP com diffserv:** avaliação através de medições. 2001. 136 f. Dissertação (Mestrado em Ciência da Computação) – Programa de Pós-Graduação em Ciência da Computação, Universidade Federal de Santa Catarina, Florianópolis. Disponível em: <http://lrg.ufsc.br/teses/Melo.pdf>. Acesso: em 9 abr. 2005.

MICROSOFT, Corporation. **Windows Media Player**. [New York?], 2005. Disponível em: <http://www.microsoft.com>. Acesso: em 24 ago. 2005.

NAUGLE, Matthew. **Guia ilustrado do TCP/IP**. Tradução Ana B. Tavares. São Paulo: Berkeley, 2001.

PÉRICAS, Francisco A. **Redes de computadores:** conceitos e a arquitetura internet. Blumenau: Edifurb, 2003.

PETERSON, Larry L.; DAVE, Bruce S. **Redes de computadores:** uma abordagem sistêmica. 2. ed. Tradução José F. M. do Amaral. Rio de Janeiro: LTC, 2004.

REDE NACIONAL DE ENSINO E PESQUISA. **O que é qualidade de serviço (QoS)**. Rio de Janeiro, 2004. Disponível em: <http://www.rnp.br/noticias/2003/not-031017b-coord.html>. Acesso em: 9 abr. 2005.

SPARXS, Systems. **Enterprise Architect**. Victoria, 2005. Disponível em: <http://www.sparxsystems.com.au>. Acesso: em 9 nov. 2005.

TANENBAUM, Andrew S. **Redes de computadores**. 4. ed. Tradução Vandeberg D. de Souza. São Paulo: Campus, 2003.

TSCHÖKE, Clodoaldo. **Criação de streaming de vídeo para transmissão de sinais de vídeo em tempo real pela internet**. 2001. 73 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.