UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉLTRICA

Um Ambiente Integrado para Manipulação de Tráfego Multicast

WELDSON QUEIROZ DE LIMA

ORIENTADOR:
Prof. D.Sc. Sergio Vianna Fialho

Dissertação submetida ao corpo docente do Programa de Pós-graduação em Engenharia Elétrica da Universidade Federal do Rio Grande do Norte como parte dos requisitos necessários para a obtenção do Grau de Mestre em Ciências (M.Sc.) em Engenharia Elétrica.

Natal-RN
Dezembro de 2004

UFRN/Programa de Pós-Graduação em Eng° Elétrica
Biblioteca Setorial do PPgEE

Catalogação da Publicação na Fonte.UFRN / Biblioteca Central
Zila Mamede. Divisão de Serviços Técnicos

Lima, Weldson Queiroz de.
Um ambiente integrado para manipulação de tráfego multicast / Weldson Queiroz de Lima. – Natal, RN, 2004.
xiii, 67 p. : il.

Orientador: Sergio Vianna Fialho.
Dissertação (Mestrado) – Universidade Federal do Rio Grande do Norte. Programa de Pós-Graduação em Engenharia Elétrica.

1. Redes de computação. 2. Multicast IP. 3. Sistemas multimídia. 4. Videoconferências. 5. Java (Linguagem de programação de computador) I. Fialho, Sergio Vianna. II. Universidade Federal do Rio Grande do Norte. III. Título.

RN/UF/BCZM CDU 004.7

Um Ambiente Integrado para Manipulação de Tráfego Multicast

WELDSON QUEIROZ DE LIMA

Aprovada, em 10 de dezembro de 2004, pela Comissão Examinadora formada pelos seguintes membros:

Prof. D.Sc. Sergio Vianna Fialho
Orientador - **UFRN**

Prof. Dr. Lisandro Zambenedetti Granville
UFRGS

Prof. Dr. João Batista Bezerra
UFRN

Prof. Dr. Luiz Affonso H. Guedes de Oliveira
UFRN

Natal, RN - Brasil
Dezembro de 2004

A Edson e Sara, meus pais,
cujo amor, sabedoria e honra
alicerçam minha vida.

"... e, assim, habite Cristo no vosso coração, pela fé,
estando vós arraigados e alicerçados em amor,
a fim de poderdes compreender, com todos os santos,
qual é a largura, e o comprimento, e a altura, e a profundidade,
e conhecer o amor de Cristo, que excede todo entendimento,
para que sejais tomados de toda a plenitude de Deus."
Efésios 3.17-19

Dedico meus sinceros agradecimentos para:

- o professor D.Sc. Sergio Vianna Fialho pela forma sem medida com que me ajudou. Agradeço-o intensamente pela oportunidade, investimento, incentivo e instrução. Jamais poderei retribuir o bem que fez para comigo, salvo em propagar a integridade, honra e atenção de sua docência aos que cruzarem meu caminho.

- a toda minha família, pelo incentivo e sustento emocional. Em especial a minha tia Raquel, que não apenas me recebeu em sua casa, mas me apoiou silenciosamente nos momentos mais difíceis. Agradeço ao meu irmão Sanderson, cuja proteção e referência foram o norte dos meus caminhos. Ao meu irmão Jorge, por sua fidelidade e lealdade.

- a Larissa, delícia dos meus olhos, por a cada manhã desenhar um sorriso em meu rosto, adoçar minha alma e aquecer meu coração.

- ao Ponto de Presença da Rede Nacional de Ensino e Pesquisa no Rio Grande do Norte. Agradeço intensamente a cada um que o representa, sem eles nenhuma destas linhas haveria.

- a PPgEE e todo seu corpo docente.

- todos os colegas do Mestrado em Engenharia Elétrica da UFRN.

Resumo da tese apresentada ao PPgEE/UFRN como parte dos requisitos necessários para a obtenção do Grau de Mestre em Ciências (M.Sc.) em Engenharia Elétrica.

Um Ambiente Integrado para Manipulação de Tráfego Multicast

WELDSON QUEIROZ DE LIMA

Dezembro de 2004

Orientador: Prof. D.Sc. Sergio Vianna Fialho
Programa: PPgEE/UFRN

Nas duas últimas décadas do século passado, com a consolidação da Internet como rede mundial de computadores, aplicações de fluxos mais robustos começaram a surgir. O crescente uso de videoconferências impulsionou a criação de uma forma de transmissão ponto-multiponto chamada Multicast IP.

Todas as empresas que desenvolviam *software* e *hardware* para videoconferência adequaram seus produtos e criaram novas soluções para o uso do fluxo multicast. Entretanto, a configuração das diversas soluções não é trivial e, normalmente, alterações no sistema operacional precisam ser realizadas. Além disso, ferramentas gratuitas apresentam funcionalidades limitadas, e as soluções proprietárias encontradas na atualidade são muito dependentes de plataformas específicas.

Com o amadurecimento da tecnologia Multicast IP e com sua inclusão em todos os sistemas operacionais atuais, as linguagens de programação desenvolveram classes capazes de manipular tráfego multicast. Com as APIs Java para redes, banco de dados e páginas Web, tornou-se possível a criação de um Ambiente Integrado capaz de manipular tráfego multicast, que se constitui na proposta central deste trabalho.

Esse documento descreve então a implementação deste ambiente que agrega diversas funcionalidades para utilização e gerência de tráfego multicast, funcionalidades até então presentes de forma limitada em poucas e distintas ferramentas comummente proprietárias. O ambiente se adequa a diferentes perfis de usuário, no sentido de que pode ser usado por leigos em Engenharia de Redes, que desejem apenas participar de sessões de multimídia na Internet, como também por especialistas e administradores de rede que desejem monitorar e manipular o tráfego multicast.

Palavras-chave: Multicast IP, Videoconferência, Java

Abstract of thesis presented in the PPgEE as partial fulfillment of the requirements for the degree of Master in Sciences (M.Sc.) in Eletrical Engineering

An Integrated Environment to Handle Multicast Traffic

WELDSON QUEIROZ DE LIMA

December 2004

Advisor: Prof. D.Sc. Sergio Vianna Fialho
Program: PPgEE/UFRN

In the two last decades of the past century, following the consolidation of the Internet as the world-wide computer network, applications generating more robust data flows started to appear. The increasing use of videoconferencing stimulated the creation of a new form of point-to-multipoint transmission called IP Multicast.

All companies working in the area of software and the hardware development for network videoconferencing have adjusted their products as well as developed new solutionsfor the use of multicast. However the configuration of such different solutions is not easy done, moreover when changes in the operational system are also requirede. Besides, the existing free tools have limited functions, and the current comercial solutions are heavily dependent on specific platforms.

Along with the maturity of IP Multicast technology and with its inclusion in all the current operational systems, the object-oriented programming languages had developed classes able to handle multicast traffic. So, with the help of Java APIs for network, data bases and hipertext, it became possible to the develop an Integrated Environment able to handle multicast traffic, which is the major objective of this work.

This document describes the implementation of the above mentioned environment, which provides many functions to use and manage multicast traffic, functions which existed only in a limited way and just in few tools, normally the comercial ones. This environment is useful to different kinds of users, so that it can be used by common users, who want to join multimedia Internet sessions, as well as more advenced users such engineers and network administrators who may need to monitor and handle multicast traffic.

Key-words: IP Multicast, Videoconference, Java

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Acrônimos

API - Applications Programming Interfaces

CPU - Central Processing Unit

DoS - Denial of Service

DVMRP - Distance Vector Multicast Routing Protocol

HTML - HyperText Markup Language

HTTP - HyperText Transfer Protocol

IETF - Internet Engineering Task Force

IGMP - Internet Group Management Protocol

IOS - Internetwork Operation System

IP - Internet Protocol

JDBC - Java Database Connectivity

JDK - Java Development Kit

JSP - Java Server Pages

JVM - Java Virtual Machine

LAN - Local Area Network

MIB - Management Information Base

MOSPF - Multicast Open Shortest Path First

MRTG - Multi Router Traffic Grapher

PIM - Protocol Independent Multicast

PIMwg - PIM Working Group

RFC - Request For Comments

RIP - Routing Information Protocol

RP - Rendezvous Point

SAP - Session Announcement Protocol

SDP - Session Description Protocol

SGBD - Sistema Gerenciador de Bancos de Dados

SQL - Structured Query Language

TTL - Time to Live

UML - Unified Modeling Language

URI - Uniform Resource Identifier

1 *Introdução*

Em meados da década de oitenta, quando a Internet começou a consolidar-se como rede mundial de computadores, surgiu a necessidade de se criar soluções de Engenharia de Redes para prover fluxos de dados mais robustos. Essa necessidade tornou-se mais patente quando se iniciaram transmissões de vídeo e áudio em tempo real na Internet, principalmente quando essas transmissões se destinavam a um grupo de usuários.

Diversas soluções foram apresentadas para as camadas superiores da pilha de protocolos dos sistemas de comunicação. Entretanto, o crescimento de fluxos multimídia presentes na rede começava a onerar muito a qualidade do tráfego. Integrando as propostas de solução para camadas superiores, naquela época utilizavam-se somente formas muito rudimentares de transmissão de dados para as aplicações multimídia, a saber: transmissão *unicast* e *broadcast*. A primeira promove a transmissão ponto-a-ponto entre as máquinas de origem e destino, enquanto a segunda trata de uma forma de transmissão em que pacotes são enviados a todos os *hosts* de uma rede local.

Para que a Internet pudesse atender às exigências das novas aplicações, foi preciso criar uma forma alternativa de transmissão chamada Multicast IP, que consiste em um modelo de transmissão de grupo ponto-multiponto. Multicast é uma solução possível de ser implementada nas camadas de enlace e rede, e rapidamente surgiram diversos aperfeiçoamentos do modelo original, culminando em se tornar a forma de transmissão padrão para a nova versão do protocolo IP, o IPv6. A comunidade científica da Internet desenvolveu então protocolos e padrões que dessem suporte à transmissão multicast e, assim, todas as grandes empresas de telecomunicações e de dados começaram a adequar seus produtos ao modelo emergente.

Embora a transmissão multicast possa ser usada para qualquer natureza de fluxo de dados, ela tem sido basicamente usada para transmissão de tráfego multimídia. Adequando-se os sistemas operacionais, foram desenvolvidos muitos programas de multimídia e videoconferência baseados no uso da transmissão multicast. No entanto, para que esses pro-

gramas funcionem adequadamente sua configuração não é trivial. Alguns dos programas de videoconferência tradicionais da Internet foram atualizados para usar Multicast IP, no entanto, estes apresentam limitações em suas funções quanto ao uso da potencialidade do tráfego. Portanto, para usar potencialmente o tráfego multicast do Mbone[1] [Almeroth 2004], é preciso dispensar grande tempo na configuração dos equipamentos de rede e inter-rede, além de ser necessário o uso de diversos programas concomitantes.

Este trabalho tem por objetivo minimizar o esforço do usuário no uso do tráfego multicast, isentando-o da instalação de diversos programas no sistema operacional de sua máquina. Para isso é proposto um ambiente que concentre todos os agentes manipuladores de tráfego no navegador, fazendo com que tanto um usuário leigo quanto um experiente possam usufruir de todo o potencial que a transmissão multicast pode oferecer. Essa ferramenta, desenvolvida em Java, também é capaz de recuperar, gerar, mensurar e testar o tráfego multicast. Assim, a ferramenta também poderá ser usada por administradores de rede que desejem gerenciar o tráfego multicast de sua rede. A fácil acessibilidade da ferramenta torna-a útil também a usuários que apenas desejam participar de conferências, em vídeo, áudio ou texto, sem preocupar-se em gerenciar o tráfego multicast. E ainda, o ambiente é codificado de forma livre e aberta, servindo como aparato para usuários desenvolvedores de ferramentas multimídia para a Internet.

Este documento está organizado da seguinte forma: no Capítulo 2 são apresentadas definições teóricas sobre Multicast IP, explicitando sua história, características técnicas e o estado da arte. No Capítulo 3 são apresentadas algumas ferramentas capazes de isoladamente manipular o tráfego multicast, explicitando-se suas características e deficiências. A seguir, nos Capítulos 4 e 5, o documento trata da proposta de desenvolvimento de um Ambiente Integrado para Manipulação de Tráfego Multicast, onde se utiliza diagramas UML (*Unified Modeling Language*) para sua especificação e implementação. Finalmente, no Capítulo 6 apresentam-se os testes realizados e os resultados obtidos com o uso do ambiente desenvolvido e, no Capítulo 7, as conclusões finais deste trabalho.

[1] A Mbone, Multicast Backbone, é uma rede virtual que funciona em camadas superiores da Internet. Isso porque nem todos os roteadores do mundo, nodos do *backbone* mundial, suportam transmissão multicast. Para que o tráfego multicast possa fluir na Internet, foram criadas ilhas multicast ligadas por enlaces virtuais ponto-a-ponto chamados túneis.

2 *Multicast IP*

2.1 Breve histórico

Na Universidade de Stanford, no início dos anos 80, o aluno de doutorado Steve Deering começou a esboçar uma nova forma de transmissão que não mais restringia um emissor a mandar mensagens para apenas um receptor (transmissão *unicast*), ou para todos possíveis receptores (transmissão em *broadcast*).

Em 1985 com a RFC 966, em 1986 com a RFC 988, aperfeiçoada em 1998 pela RFC 1054 e em 1999 pela RFC 1112 [Deering 1989], Deering descreveu toda a base para uma nova tecnologia, a transmissão multicast IP. Deering descreve o multicasting como sendo uma forma de transmissão de um único datagrama IP para um grupo de *hosts*, identificados por um único endereço IP.

A tecnologia multicast rapidamente começou a se difundir no mundo, e diversas empresas e grupos de pesquisa se dedicaram ao estudo desta tecnologia. Dentre alguns dos benefícios em seu uso é possível citar a grande economia de banda passante, suporte a aplicações distribuídas e facilidade de crescimento em escala.

Com o amadurecimento da tecnologia multicast, grandes empresas de equipamentos de rede passaram a disponibilizar produtos para o uso de fluxos multicast. Algumas das empresas que vêm concentrando seus projetos em multicast são: *Cisco Systems, Juniper Networks, Microsoft Corporation, RedHat, Apple, Dell, 3Com, Sun Microsystems, IBM Corporation, AT&T Labs e HP,* dentre outras. Algumas tornaram o multicast nativo em seus sistemas operacionais e desenvolveram ferramentas para seu uso, integrando outras ferramentas aptas a, de algum modo, também mensurar esse tipo de tráfego. Entretanto, essas são ferramentas dedicadas a sistemas operacionais específicos, normalmente de limitada ou difícil configuração, além de possuirem funcionalidades restritas.

O Multicast IP não é orientado a conexão: um datagrama multicast é entregue aos membros de um grupo destino através do mesmo esquema de melhor esforço (*best effort*)

que datagramas IP *unicast* empregam. O único parâmetro que distingue um datagrama *unicast* de um *multicast* é o endereço de destino utilizado: enquanto endereços *unicast* de destino, no IPv4, utilizam as classes A, B e C, um endereço de grupo multicast pertence à classe D dos endereços IPv4, que estende-se de 224.0.0.0 à 239.255.255.255.

No IPv6, também foi reservada uma fatia específica de espaço de endereçamento para endereços de grupo multicast, caracterizada pelo prefixo ff::0/8.

2.2 Gerência local de grupos

Os *hosts* individuais devem poder se associar a um novo grupo ou deixar um grupo multicast a qualquer momento. Não existem restrições quanto à localização geográfica ou quanto ao número de membros em um grupo multicast. Um *host* pode ser membro de um ou mais grupos multicast, e mesmo que não pertença a um grupo pode enviar mensagens aos seus membros.

Como descreveu Deering, *hosts* que desejam receber tráfego multicast devem associar-se a sessões[1] multicast. Em uma rede local, todo o processo de inscrição e abandono de membros de um grupo multicast é feito através do protocolo IGMP (*Internet Group Management Protocol*) [Deering 1989]. Se um *host* deseja receber tráfego multicast de um grupo específico, ele deve informar seu desejo aos roteadores vizinhos diretamente conectados a sua rede local, para que os mesmos possam gerenciar estes grupos e repassar o tráfego multicast até o *host* que o solicitou [Williamson 2000]. No IPv4, essa comunicação entre *hosts* e roteadores para gerência dos grupos multicast é estabelecida unicamente pelo IGMP.

2.3 Protocolos de roteamento multicast

Para que as informações registradas nos roteadores das redes locais, a respeito da intenção de um *host* participar de um grupo, sejam difundidas através da Internet, se faz necessário o uso de protocolos de roteamento específicos para esse fim. Esses protocolos definem as regras para troca, entre roteadores multicast, das mensagens utilizadas na construção de suas tabelas de roteamento para grupos.

Existem atualmente diversas soluções para implementar um protocolo de roteamento multicast dentro de um sistema autônomo, a saber: o DVMRP (*Distance Vector Multicast*

[1] Semanticamente análogo a Grupos Multicast

Routing Protocol), o MOSPF (*Multicast Open Shortest Path First*) e o PIM (*Protocol Independent Multicast*).

O DVMRP [Waitzman, Partridge e Deering 1988], que baseia seu funcionamento num algoritmo do tipo vetor-distância, foi o primeiro protocolo de roteamento multicast desenvolvido, e também é largamente utilizado na criação de túneis multicast. Seu funcionamento é derivado do protocolo de roteamento *unicast* RIP (*Routing Information Protocol*)[Malkin 1998], se caracterizando pelas mesmas vantagens e desvantagens de uso. O DVMRP é simples de instalar e de configurar, mas é pouco escalável, uma vez que suas mensagens tendem a crescer em número e em tamanho em função do número de roteadores presentes no sistema autônomo.

O MOSPF [Moy 1994] é uma extensão do OSPF[Moy 1998] para a transmissão de pacotes multicast dentro de um mesmo sistema autônomo. Do mesmo modo que o OSPF, a versão multicast usa para sua operação um algoritmo, baseado no estado dos enlaces dos roteadores do sistema autônomo (algoritmo *link-state*). O MOSPF depende do uso do OSPF, além de ser aconselhável principalmente para ambientes com poucos pares (origem, grupo destino) ativos, devido a restrições de processamento.

O protocolo de roteamento multicast PIM [Pusateri e McBride 2004] foi desenvolvido pelo grupo de trabalho PIMwg da IETF (*Internet Engineering Task Force*). O objetivo deste grupo de trabalho foi de desenvolver um protocolo de roteamento multicast padrão, capaz de fornecer um roteamento multicast inter-domínio escalável através da Internet.

O nome PIM foi advindo de sua característica de não ser dependente dos mecanismos fornecidos por qualquer protocolo de roteamento *unicast*. Porém, qualquer implementação que suporte o PIM requer a presença de rotas *unicast* (estáticas ou dinâmicas), para fornecer informações da tabela de roteamento e para adaptar-se às mudanças na topologia [Osterloh 2002].

O PIM faz uma clara distinção entre um protocolo de roteamento multicast projetado para ambientes densos e outro projetado para ambientes esparsos [Parkhurst 1999]. O PIM Modo Denso, PIM-DM, refere-se ao protocolo projetado para operar num ambiente onde os membros dos grupos estão densamente distribuídos e a banda passante é abundante. Já o de modo esparso, descrito na RFC 2362 [Estrin et al. 1998] e chamado PIM-SM, refere-se ao protocolo otimizado para ambientes nos quais os membros do grupo estão distribuídos através de muitas regiões da Internet e a banda passante não é, necessariamente, largamente disponível. É importante notar que o modo esparso não significa que o grupo tem poucos membros, e sim que os mesmos se encontram muito dispersos

pela rede.

2.4 Anúncio de grupos multicast

Grupos multicast são anunciados no Mbone usando-se o protocolo padrão SDP (*Session Description Protocol*). Os anúncios de sessão usam o endereço reservado de grupo 224.2.127.254. Descrito na RFC 2327, o SDP define um conjunto de especificações para os anúncios de sessão [Handley e Jacobson 1998]. Algumas dessas especificações são: descrição da sessão, autor, tipo de codificação da mídia, endereço de origem, endereço do grupo, fuso horário, uso da banda, entre outras.

Os pacotes SDP para anúncio de sessões multicast são difundidos automaticamente no Mbone por todos os roteadores habilitados com multicast. Regidas pelas implicações da transmissão de pacotes através do paradigma de melhor esforço e das limitações do TTL (*Time to Live*), é pouco provável que qualquer ferramenta de captação de anúncios de sessões receba absolutamente todos os pacotes SDP transeuntes no Mbone.

A figura 2.1, retirada da RFC 2327 [Handley e Jacobson 1998], apresenta a descrição dos campos que compõem um pacote SDP. Alguns campos do SDP são obrigatórios. Os demais, sinalizados com " * ", são opcionais. No entanto, todos os campos devem respeitar a ordem de seqüência apresentada. O pacote SDP é dividido em três segmentos, sendo o primeiro reservado para descrições concernentes à sessão, o segundo traz informações de tempo da sessão e a última parte descreve informações das mídias usadas na sessão. Os campos que pertencem à terceira parte são os que podem aparecer diversas vezes em um pacote SDP.

Os campos de um pacote SDP tem o seguinte significado:

`v=` Este campo marca o início do pacote SDP, é obrigatório, e representa a versão do protocolo. Até os dias atuais a versão `0` ainda está em uso.

`o=` Também obrigatório, este campo é construído no formato `o=<autor> <id da sessão> <versão da sessão> <tipo da rede> <versão do IP> <endereço IP do autor>`. Embora a RFC 2327 não padronize um formato para o *id* e a versão da sessão, normalmente é usado o *Unix Time Stamp*[1] do instante em que o anúncio da sessão foi criado ou modificado.

[1] O Unix Time Stamp, também conhecido como Unix Epoch, é o número de segundos transcorridos desde as 0:00:00 de 1 de janeiro de 1970 GMT.

```
Session description
        v=  (protocol version)
        o=  (owner/creator and session identifier).
        s=  (session name)
        i=* (session information)
        u=* (URI of description)
        e=* (email address)
        p=* (phone number)
        c=* (connection information - not required if included in all media)
        b=* (bandwidth information)
        One or more time descriptions (see below)
        z=* (time zone adjustments)
        k=* (encryption key)
        a=* (zero or more session attribute lines)
        Zero or more media descriptions (see below)

Time description
        t=  (time the session is active)
        r=* (zero or more repeat times)

Media description
        m=  (media name and transport address)
        i=* (media title)
        c=* (connection information - optional if included at session-level)
        b=* (bandwidth information)
        k=* (encryption key)
        a=* (zero or more media attribute lines)
```

Figura 2.1: Descrição dos campos do pacote SDP

`s=` Neste campo encontra-se o nome da sessão. Normalmente é este campo que é exibido por ferramentas de captura de anúncios de sessão.

`i=` Este campo é usado para apresentar um pequeno texto explicativo sobre a sessão a que o pacote SDP se refere. O campo **i=** também pode ser usado em cada mídia descrita no pacote SDP.

`u=` É usado para fornecer o endereço de uma página que contenha mais informações sobre a sessão que está sendo anunciada.

`e=` Contém o *e-mail* do autor do anúncio.

`p=` Contém um telefone para contato, com " + " precedendo o código do país e separando os prefixos regionais com um espaço ou hífen.

`c=` Pacotes SDP devem conter este campo em cada descrição de mídia ou uma única vez na primeira parte do pacote (descrição da sessão). Este campo é construído no formato `<tipo da rede> <versão do IP> <endereço do grupo multicast>/ <TTL>/ <número de endereços consecutivos>`

`b=` Contém informações quanto ao uso da banda passante consumida por todas as mídias utilizadas na sessão(CT:) ou, se presente na descrição de uma mídia, o uso apenas daquela mídia (AS:).

`t=` Apresenta no formato *Unix Time Stamp* a hora de início e fim da transmissão da sessão.

`r=` Caso haja retransmissões da sessão, este campo apresenta o intervalo de tempo entre elas, a partir da hora de início do campo **t=**.

`z=` É usado para informar o fuso horário da hora de início de transmissão da sessão.

`k=` A existência deste campo revela que a sessão está encriptada. A chave de decriptação é exposta em texto plano neste campo.

`a=` Este campo pode surgir inúmeras vezes no pacote SDP, tanto na parte de descrição da sessão, quanto em cada mídia descrita. O campo **a=** pode ser usado para estender as funcionalidades do SDP, uma vez que seu conteúdo é livremente preenchido pelo autor do anúncio, no formato `a=<atributo>:<valor>`.

A figura 2.2 apresenta, como exemplo, o conteúdo de um pacote SDP. A partir dele pode-se obter não apenas informações obrigatórias como o endereço e porta do grupo

multicast (mostrados nos parâmetros `c=IN IP4 227.0.0.7/32/1` e `m=audio 27006/1 RTP/AVP 0`), como também obter informações adicionais relevantes à melhor adequação da ferramenta que irá receber o *stream* de dados. Algumas destas informações adicionais são o nome do anunciante do grupo e seu e-mail (parâmetro `e=Weldson Q. Lima <well@pop-rn.rnp.br>`), o *site* (`u=http://www.pop-rn.rnp.br`), a taxa de transmissão do *stream* (`b=CT:192`), entre outras.

```
v=0
o=well 3299494553 3299494553 IN IP4 200.137.0.60
s=IMPA Feedback from PoP-RN/RNP
i=Retorno da sala do PoP-RN para o curso de Inverno do IMPA
u=http://www.pop-rn.rnp.br
e=Weldson Q. Lima <well@pop-rn.rnp.br>
p=+55 84 2153170
c=IN IP4 227.0.0.7/32/1
b=CT:192
z=1185421713 120
a=tool:Polycom FX
a=type:meeting
t=1185421713 1185436113
r=1d -28800
m=audio 27006/1 RTP/AVP 0
i=Som Ambiente
b=AS:64
a=quality:good
m=video 27008/1 RTP/AVP 31
i=Tuma do IMPA no RN
b=AS:128
a=quality:good)
```

Figura 2.2: Pacote SDP

O pacote SDP da figura 2.2 respeita rigorosamente os padrões definidos na RFC 2327 [Handley e Jacobson 1998]. No entanto, pouquíssimas ferramentas obedecem completamente aos padrões estipulados. Mesmo as ferramentas que respeitam os padrões, o fazem de forma limitada, normalmente não oferecendo ao usuário a opção de preenchimento de campos opcionais.

3 *Ferramentas multicast*

3.1 Ferramentas livres e abertas

Servindo de referência para ferramentas mais atuais, as primeiras ferramentas multicast utilizadas no Mbone foram as desenvolvidas pelo Departamento de Ciência da Computação da universidade inglesa UCL [University College London 2001]. Essas ferramentas, de código livre, inicialmente foram desenvolvidas para plataformas Unix, embora algumas tenham ganho versões para Windows. As ferramentas da UCL ainda hoje são largamente utilizadas no Mbone, e possuem como diferencial a capacidade de configuração e estabilidade. São as clássicas ferramentas da UCL:

Session Directory Rendezvous (SDR) - O SDR é um programa que lê pacotes SDP e lista em sua tela principal a descrição das sessões de videoconferência multicast anunciadas. Essas descrições são apresentadas na forma de *links*, que levam a uma tela de descrições mais detalhadas sobre a sessão. Como mostra a figura 3.1, as sessões são classificadas, utilizando-se determinados ícones, quanto à sua natureza, podendo ser: de conferência ( ), apenas recepção ( ) ou em fase de teste ( ); e ainda quanto ao horário sendo corrente ou agendada. Embora também seja capaz de anunciar sessões geradas pelo usuário, o SDR não é capaz de participar de sessões, para tal é preciso que se instale um programa apto a manipular as mídias que a sessão usa. Caso o usuário deseje anunciar sessões, o SDR disponibiliza ao usuário o preenchimento de um formulário com os dados que irão compor o pacote SDP. Mesmo respeitando os padrões descritos na RFC 2327, apenas uma minoria dos campos é oferecida ao preenchimento pelo usuário, ignorando os demais campos. Além disto, com informações colhidas dos pacotes SDP, o SDR exibe um calendário com os dias em que haverá sessões multicast sendo transmitidas.

Video Conference Tool (VIC) - O VIC, ilustrado na figura 3.2, é uma ferramenta de vídeo de uso bastante intuitivo: em sua tela principal há um pequeno resumo descritivo de cada usuário da sessão e, ao lado de cada descrição abre-se uma pequena

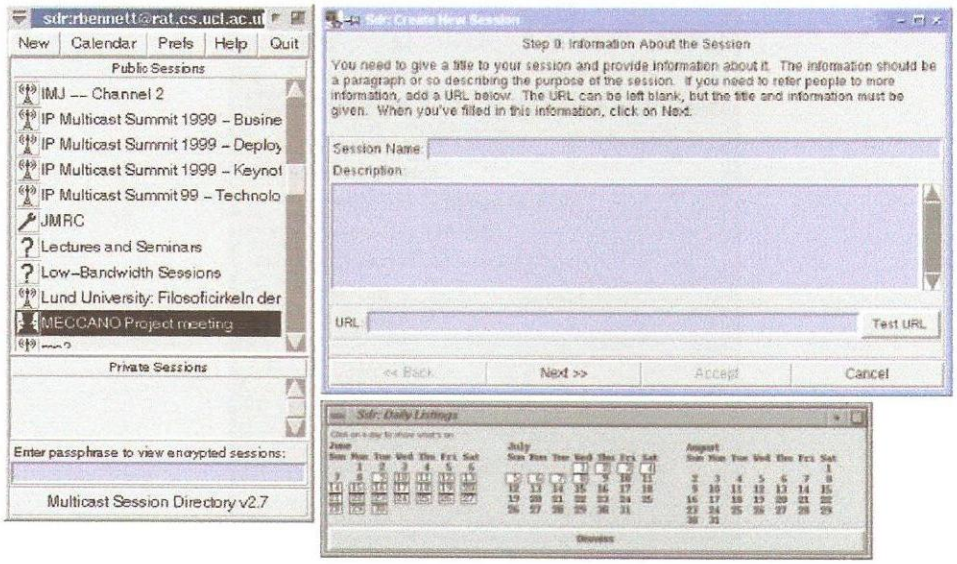

Figura 3.1: Multicast Session Directory Rendezvous v2.7

janela de vídeo do usuário. Como as demais ferramentas de mídia da UCL, o VIC pode ser acionado automaticamente pelo SDR ou manualmente através de linha de comando.

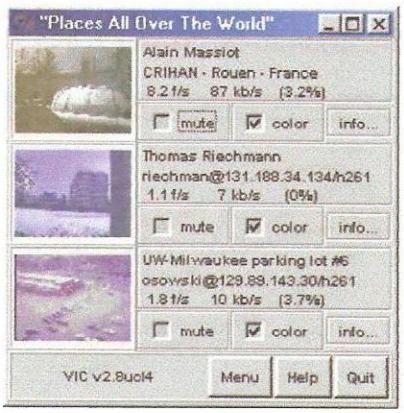

Figura 3.2: Video Conference Tool v2.8

Robust Audio Tool (RAT) - Talvez o RAT seja a ferramenta mais usada dentre as de seu time, isso porque manipula áudio tanto em multicast como em *unicast*. O RAT possui diversas diretivas de configuração, que podem ser configuradas manualmente ou armazenadas em um arquivo de configuração compartilhado pelo VIC. Como mostra a figura 3.3, a tela principal do RAT exibe o nome dos participantes da sessão.

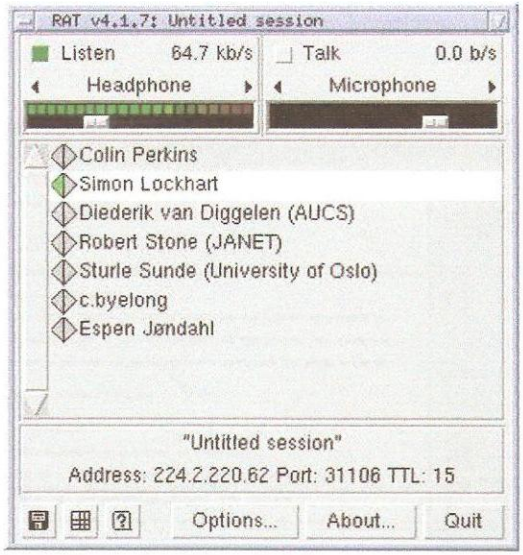

Figura 3.3: Robust Audio Tool v4.1.7

White Board Tool (WB) - Como mostra a figura 3.4, o WB disponibiliza uma área de exibição compartilhada, na qual os participantes podem escrever e/ou desenhar. O WB oferece ferramentas de desenho básicas como escolha de cor, fonte e tamanho de pincel. Há controles para mudar e criar páginas. Figuras e texto podem ser inseridos, usando o teclado e o *mouse*, ou importando-se um arquivo.

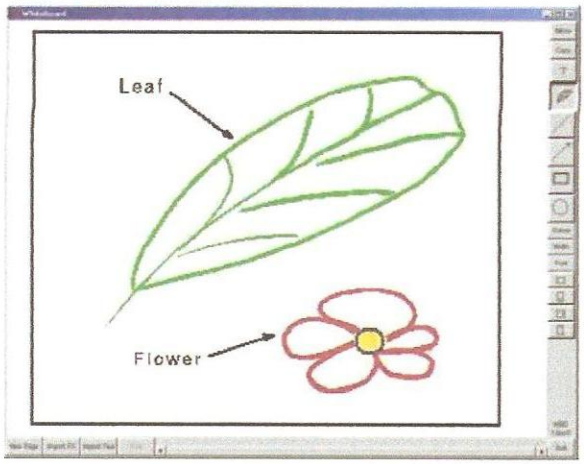

Figura 3.4: White Board Tool v1.0

Network Text Editor (NTE) - O NTE é uma ferramenta de texto semelhante aos *programas de bate-papo convencionais. Como o RAT, o NTE pode ser usado tanto* em *unicast* como em multicast. Em uma videoconferência, o NTE pode ser usado

como um fórum de debate ou para carregar arquivos de texto, conforme mostra a figura 3.5.

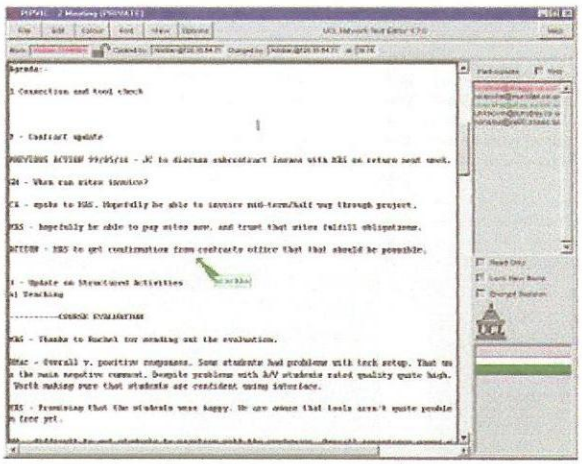

Figura 3.5: Network Text Editor v2.4

Como se pode observar, as ferramentas gratuitas mais amplamente utilizadas tratam de aspectos isolados da comunicação multimídia, uma vez que cada ferramenta é dedicada a uma mídia ou funcionalidade específica. Deste modo, torna-se necessário instalar e configurar várias ferramentas para implementar uma aplicação mais abrangente, a exemplo de uma ferramenta de videoconferência.

Existem ainda algumas outras ferramentas de videoconferência gratuitas, que utilizam tráfego multicast, a exemplo da ferramenta *Shrimp* da própria UCL. Essas, em sua maioria, tratam apenas de aglutinar o código das ferramentas anteriormente apresentadas.

3.2 Soluções comerciais

Além destas ferramentas clássicas, descritas na sessão 3.1, existem muitas outras ferramentas comerciais, de diferentes fornecedores, que se propõem a usar tráfego multicast. Uma das mais elaboradas ferramentas para visualização de tráfego multicast é o Cisco IP/TV, ilustrado na figura 3.6. Desenvolvida apenas para plataforma Windows, suporta diversos tipos de mídia, inclusive mp3, e é gratuita. Entretanto, a versão servidor, para que usuários possam ser geradores de tráfego multicast, não é gratuita. Ambas as versões, cliente e servidor, exigem um alto desempenho de hardware. Segundo o manual do Cisco IP/TV [Team 2004], a versão cliente exige um mínimo de 400MHz de freqüência de processamento e, pelo menos, 24 Mb de memória RAM apenas para o Cisco IP/TV.

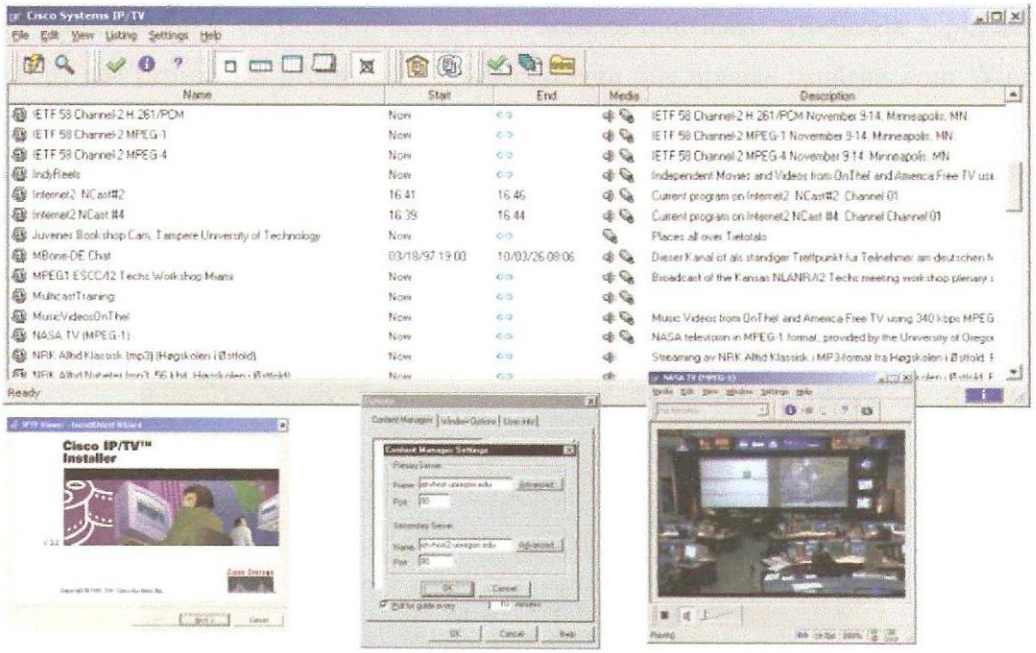

Figura 3.6: Cliente Cisco IP/TV v3.4

Embora não seja pequeno o número de ferramentas clientes, ferramentas servidor ainda são escassas e caras. As três mais utilizadas talvez sejam a Windows Media Service, Helix Universal Server e QuickTime Streaming Server. Todas estas ferramentas basicamente implementam as mesmas funcionalidades das ferramentas da UCL, entretanto de forma dedicada às suas ferramentas clientes.

O Helix Universal Server é uma plataforma que oferece suporte para transmissão em tempo real ou sob demanda. O Helix suporta diversos formatos de vídeo, e pode promover transmissões em *unicast* ou multicast. Não apenas no cliente Real Player, mas também no Helix Universal Server, foram encontradas diversas falhas de segurança, a exemplo de vulnerabilidades a ataques de DoS (*Denial of Service*) e acesso privilegiado não autorizado [RealNetworks 2004].

Embora seja uma ferramenta comercial, o QuickTime Streaming Server possui uma versão com código aberto, chamada Darwin Streaming Server. Originalmente feito para Mac OS X Server, também possui versões para outras plataformas. Além da capacidade de manipular tráfego em tempo real com baixos requerimentos de *hardware*, outro grande diferencial do QuickTime é seguir os padrões definidos pelas RFCs.

O Microsoft Windows Media Service, ilustrado na da figura 3.7, vem incorporado ao Windows 2000/2003 Server e, talvez, seja a mais fácil e intuitiva ferramenta capaz

de agir como servidor de mídia e anúncio de sessões multicast. Dedicada à plataforma Microsoft, usa protocolos proprietários, fazendo com que apenas usuários com Windows Media Player possam receber o *stream* de dados e os anúncios por ela servidos. Além desta demasiada dependência à plataforma, o Windows Media Service apresenta falhas de segurança que deixam o servidor vulnerável à ataques de DoS [TechNet 2004].

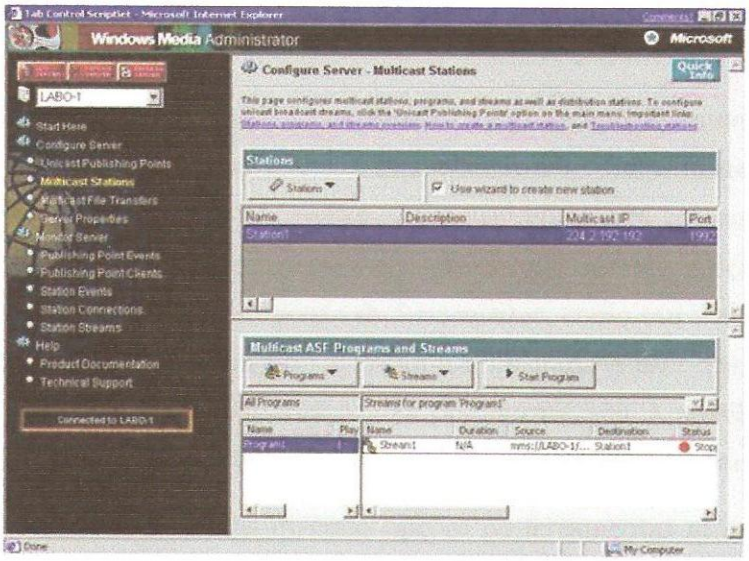

Figura 3.7: Microsoft Windows Media Service

O cliente QuickTime e o VideoLAN, ilustrados na figura 3.8, a direita e esquerda respectivamente, são duas das mais versáteis ferramentas cliente para assistir a transmissões multicast, sendo o QuickTime a ferramenta mais fiel aos padrões. O VideoLAN também é capaz de funcionar como servidor de mídia multicast, no entanto a única participação do usuário na criação dos campos do pacote SDP é a escolha do endereço e porta de grupo, ficando a cargo da ferramenta preencher os demais campos.

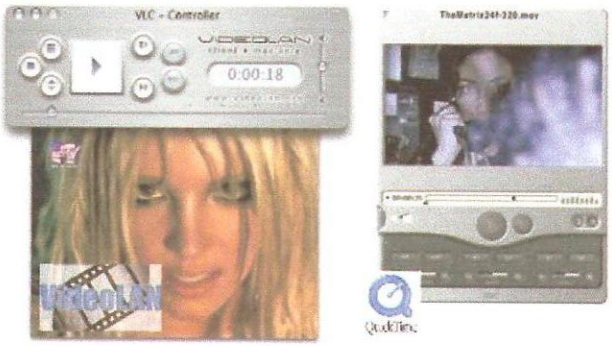

Figura 3.8: VideoLAN e cliente QuickTime

3.3 Comentários

Para fazer uso de tráfego multicast, através de ferramentas que baseiam sua transmissão neste modelo, é preciso que um usuário dispense demasiado tempo em instalação e configuração de diversas ferramentas. Além disto, o uso concomitante das diferentes ferramentas, dedicadas a cada tipo de mídia, exige maior quantidade de recursos computacionas da estação do usuário.

A dependência de plataforma é uma característica própria de ferramentas comerciais. Deste modo, o usuário fica dependente de uma plataforma, comprometendo a portabilidade. Como descrito na seção 3.1, muitas ferramentas deixam o usuário a mercê de falhas de segurança e, por usarem protocolos e código não abertos, forçam ao usuário a instalar, não somente um grande conjunto de ferramentas, como também atualizações de segurança. O uso de protocolos proprietários limita o usuário a usar apenas ferramentas de uma mesmo desenvolvedor, dedicados a uma plataforma, sem a possibilidade de criar suas próprias aplicações, que interajam com outras ferramentas existentes.

Agregar as funcionalidades das diversas opções de ferramentas atuais em um único ambiente livra o usuário do esforço instalar, configurar e preocupar-se com atualizações de segurança. Soluções baseadas no navegador garantem a portabilidade, visto que funcionam independente da plataforma em que o navegador está instalado. Portanto, uma solução desta natureza, que manipula tráfego multicast a partir de aplicações no navegador, sugere uma promissora quebra de paradigmas.

4 *Especificação do ambiente*

4.1 Análise de requisitos

Após análise das características das ferramentas existentes, realizou-se um estudo dos requisitos que um ambiente de *software* deveria apresentar, para ser de utilidade aos usuários da tecnologia de transmissão multicast.

Definiu-se, então, que a principal característica deste ambiente seria sua capacidade de manipular tráfego multicast, garantindo que o usuário não precise instalar, em sua estação, nenhuma ferramenta de videoconferência multicast. Identificou-se, ainda, que o usuário deste ambiente poderia ser de três tipos:

- Um usuário comum, que não conheça os aspectos técnicos envolvidos na transmissão multicast, mas que gostaria de ter acesso aos serviços disponibilizados no MBone, a exemplo de: listar as videoconferências agendadas na rede, participar de uma sessão com transmissão multimídia e outras;
- Um administrador de rede, que tenha interesse em monitorar o tráfego multicast em sua rede, através de funções de consulta de estatísticas, monitorar parâmetros de configuração de equipamentos da rede, a exemplo do acesso à MIB (*Management Information Base*) dos roteadores de sua rede, realizar testes de conectividade multicast e ainda gerenciar anúncios e tráfego multicast;
- Um desenvolvedor de aplicações multicast, que possa fazer uso das classes do ambiente, de modo que estas interajam com classes criadas pelo próprio usuário, promovendo, desta forma, uma expansão das funcionalidades de suas aplicações agregando o uso de tráfego multicast a elas.

Como a operação do ambiente deve atender a usuários não especializados na área, se torna importante a facilidade de interação com os recursos disponibilizados pelo ambiente. Dessa forma, é recomendável que a interface do usuário seja amigável e familiar a ele, de

fácil operação, fazendo uso de recursos gráficos, tais como menus, botões, barras, dentre outros.

Para que o ambiente livre o usuário da necessidade de instalar diferentes ferramentas de propósitos específicos, é preciso que ele agregue as funcionalidades das ferramentas existentes de uma forma independente da plataforma, e, portanto, portável ao cliente. Para que a agregação das ferramentas seja eficaz, é preciso que o ambiente faça uso apenas dos padrões definidos na Internet. Dentre as funções identificadas, as de possível interesse a um usuário comum podem ser listadas a seguir:

- Listar as sessões anunciadas;
- Gerar novos anúncios de sessão;
- Selecionar uma sessão da lista e participar da sessão;
- Poder especificar uma sessão previamente conhecida mas não listada;
- Participar de um bate-papo.

Além das funções para usuários comuns, administradores de rede podem fazer uso de outros recursos que exijam algum conhecimento mais aprofundado da tecnologia multicast, e que são listados a seguir:

- Testar a conectividade multicast;
- Gerar pacotes multicast na rede;
- Acessar a MIB (PIM e IGMP) de roteadores CISCO de sua rede;
- Consultar estatísticas de tráfego.

Como requisito de implementação, no servidor do ambiente devem ser usadas soluções de *software* livre para a plataforma, base de dados e linguagem de programação. Para garantir a portabilidade, o servidor deve ser codificado em um linguagem baseada em Web e, portanto, acessado pelos clientes por meio do navegador. O ambiente deve ser responsável apenas por manipular o tráfego multicast, e não por decodificar os fluxos de áudio e/ou vídeo em que as sessões são transmitidas. Para isto, deve interagir com alguma ferramenta dedicada apenas a este fim.

A exemplo da geração de tráfego e acesso a MIB de roteadores, algumas funções do ambiente precisam ser executadas na estação do cliente. Portanto, o ambiente deve prover meios de fazer com que o usuário invoque estas funções para serem executadas em sua estação, independente da plataforma de que faça uso e, ainda, com alguma forma de certificado que garanta a segurança da aplicação. Diante disto, como ilustrado na figura 4.1, o ambiente deve ser criado sob a arquitetura *Cliente-Servidor*, em que o servidor disponibilizará funções aos clientes e, ainda, captará informações da rede para compor sua base de dados.

O ambiente proposto não assume um comportamento de refletor, capturando tráfego multicast e o retransmitindo aos clientes. No entanto, o ambiente deve apenas capturar os anúncios de sessão multicast e, quando requisitado por um usuário, deve prover meios de que o tráfego flua diretamente da fonte ao usuário que o solicitou. Este comportamento é ilustrado na figura 4.1.

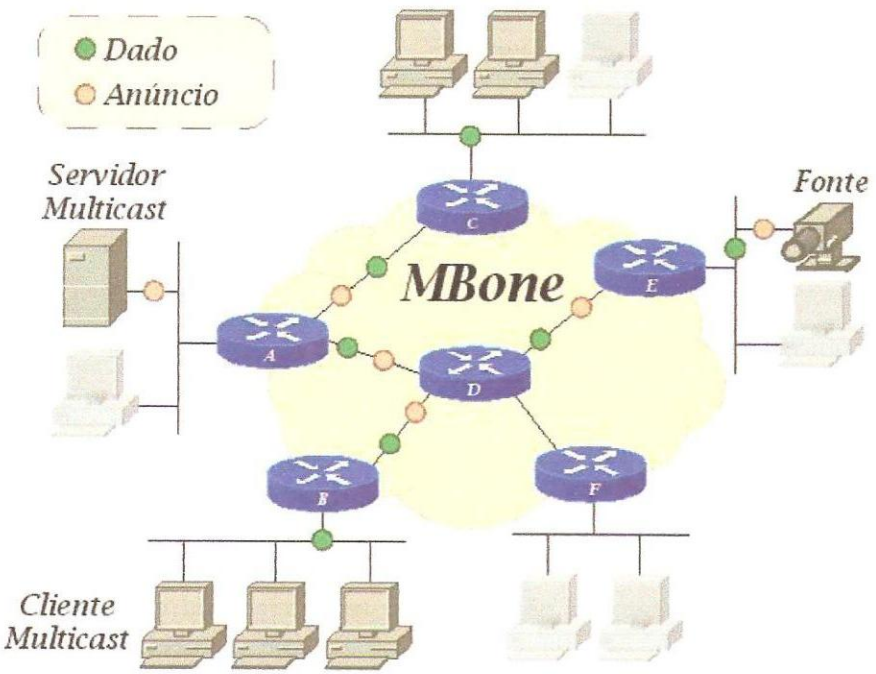

Figura 4.1: Contexto de execução do ambiente

4.2 Especificação formal

Com o auxilio da UML, apresenta-se nesta seção uma especificação formal da ferramenta proposta. O diagrama de Casos de Uso, mostrado na figura 4.2, descreve o comportamento geral do sistema e seus sub-sistemas, como também os agentes externos

que interagem com o mesmo [Rumbaugh, Jacobson e Boock 1999]. Os casos de uso incluídos nesse diagrama refletem as funcionalidades identificadas para a ferramenta durante a análise de requisitos realizada.

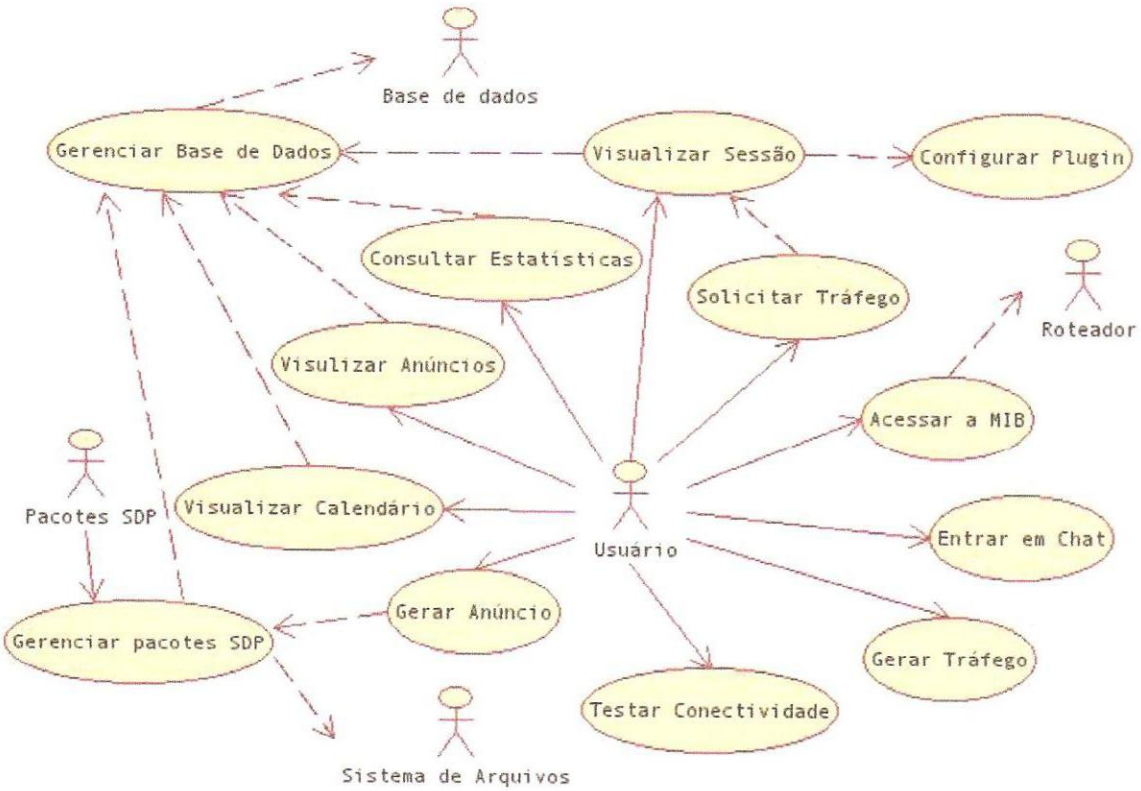

Figura 4.2: Diagrama Casos de Uso do Ambiente Integrado

As ações que cada Ator, agente externo, pode executar são melhor detalhadas nos Diagramas de Seqüência, onde é possível se perceber a execução cronológica de cada tarefa. O ambiente tem apenas o Usuário como um agente externo ativo. Outros agentes externos que também interagem com o sistema são Pacotes SDP, Roteadores, o Sistema de Arquivos do servidor e a Base de Dados.

Para as funções descritas no diagrama de Casos de Uso, o servidor deve incluir minimamente as classes mostradas no Diagrama de Classes da figura 4.3.

A classe *Gerenciador de Pacotes SDP* deve ser responsável pela captura dos pacotes SDP transmitidos no MBone e por solicitar o registro das informações desses pacotes na base de dados. Cada pacote SDP precisa ter seus campos interpretados para impedir que pacotes mal formados sejam assimilados pela base de dados.

A classe *Gerenciador de Base de Dados* deve ser a responsável pelo acesso à base de

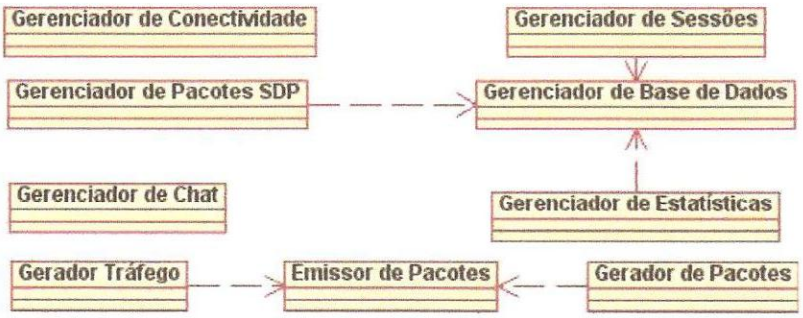

Figura 4.3: Diagrama de Classes simplificado

dados e, portanto, deve ser usada por todas as classes que precisam interagir com o banco.

A classe *Emissor de Pacotes* deve ser responsável por enviar pacotes na rede. Fazem uso dessa classe as classes *Gerador de Tráfego*, para gerar tráfego multicast, e *Gerenciador de Pacotes*, para criar pacotes SDP.

A classe *Gerenciador de Estatísticas* deve criar dados estatísticos com base nas informações contidas na base de dados.

A classe *Gerenciador de Conectividade* deve fornecer ao usuário uma função que teste se o seu segmento de rede está ou não com multicast habilitado.

A classe *Gerenciador de Sessões* deve fornecer ao usuário uma lista das sessões anunciadas no Mbone e registradas na base de dados. A partir desta lista de sessões o usuário deve poder selecionar uma sessão para ver seu conteúdo.

A classe *Gerenciador de Chat* deve ser a responsável em promover um bate papo entre os usuários do sistema. Todas as mensagens trocadas entre os participantes do bate papo devem ser transmitidas em multicast.

A seguir, um conjunto de diagramas de seqüência é apresentado, de forma a ilustrar, ainda que de modo abstrato, a seqüência de execução de cada função do ambiente.

Conforme ilustrado na figura 4.4, quando se dá a chegada de um pacote de anúncio de sessão multicast, é verificado, por meio do identificador, se as informações deste pacote já constam ou não na base de dados. Caso não constem o pacote é armazenado, do contrário é descartado. Não apenas o identificador da sessão é verificado, mas também o campo que explicita a versão deve ser verificado antes que um pacote seja armazenado na base de dados.

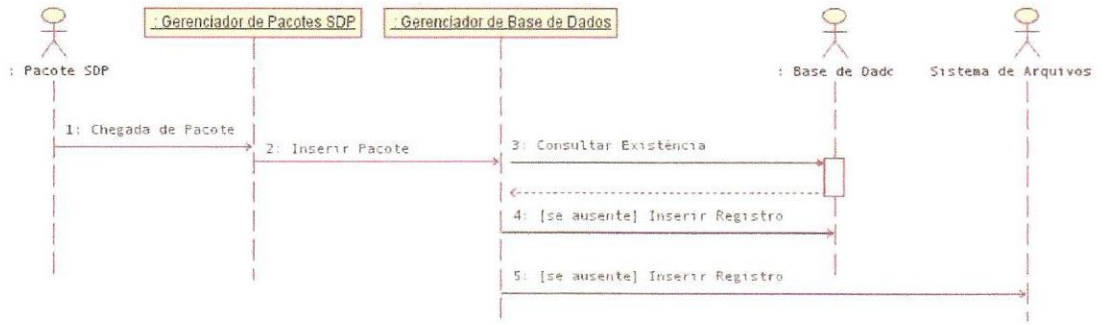

Figura 4.4: Gerenciar pacotes SDP

Quando um usuário solicita ver a lista das sessões anunciadas no Mbone, o ambiente consulta a base de dados e fornece ao cliente informações sobre cada sessão anunciada, conforme ilustrado na figura 4.5.

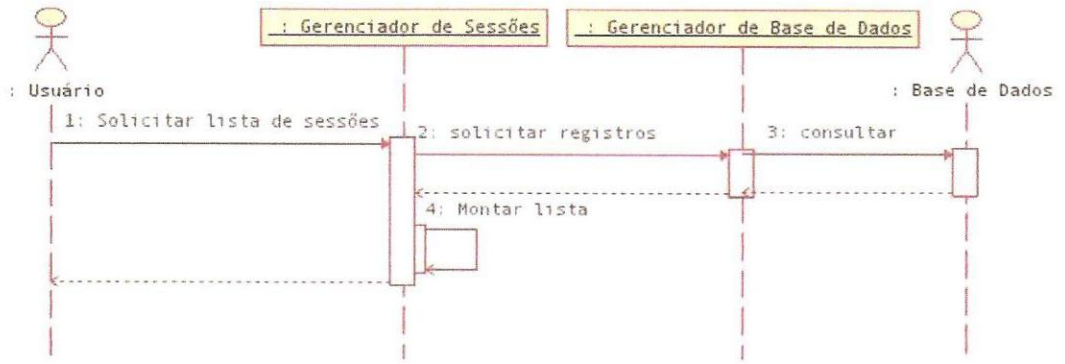

Figura 4.5: Visualizar Anúncios

Quando o usuário deseja ver o conteúdo de uma sessão anunciada e, portanto, participar do grupo multicast da sessão, o ambiente fornece ao usuário uma interface pela qual o mesmo pode receber o conteúdo transmitido, conforme ilustrado na figura 4.6. O ambiente deve fazer com que o tráfego multicast seja transmitido diretamente da fonte ao cliente.

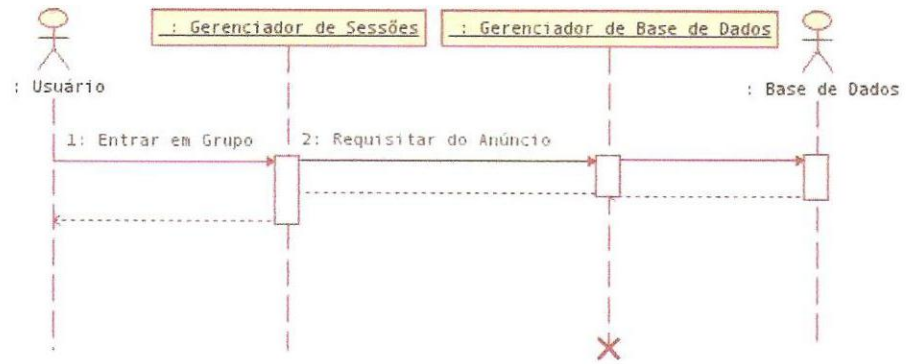

Figura 4.6: Visualizar Sessão

Mesmo que o anúncio de uma sessão não conste no ambiente, conforme ilustrado no diagrama da figura 4.7, o usuário pode visualizar o conteúdo da mesma informando ao ambiente os parâmetros da sessão. Com esses parâmetros fornecidos pelo usuário, o ambiente deve preparar uma interface para exibição do conteúdo transmitido e invocar o tráfego multicast para que este chegue até a estação do cliente.

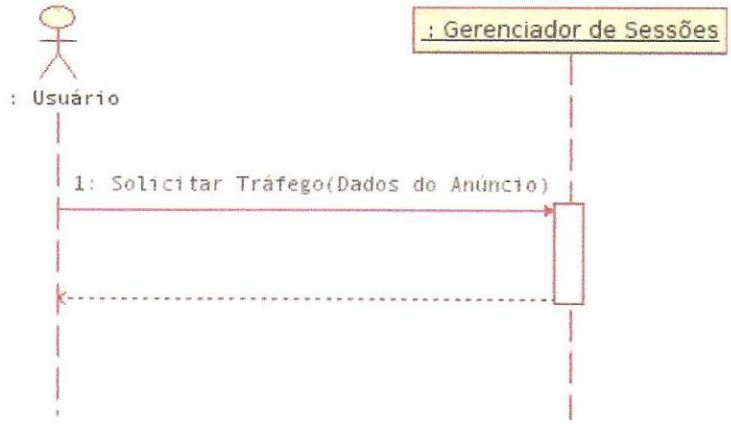

Figura 4.7: Solicitar Tráfego

Usuários mais experientes podem criar anúncios de sessão através do preenchimento de um formulário contendo os campos de um pacote SDP. Conforme ilustrado na figura 4.8, antes de enviar pacotes SDP com os dados fornecidos pelo usuário, o ambiente verifica se há inconsistência nas informações que irão compor o pacote SDP. Alguns dos campos do pacote SDP o próprio ambiente se encarrega de preencher, a exemplo do identificador da sessão, o endereço IP do anunciante, dentre outros.

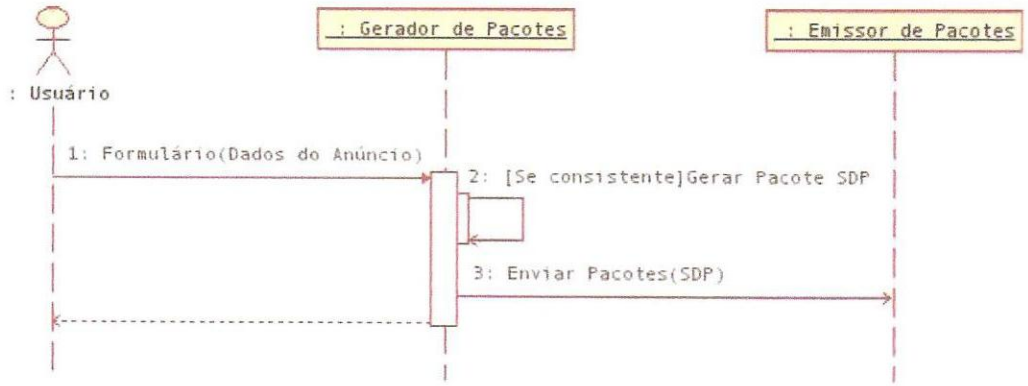

Figura 4.8: Gerar Anúncio

Uma vez criado o pacote SDP, por meio da classe *Gerador de Pacotes*, a classe *Emissor de Pacotes* deve receber o pacote recém formado e enviá-lo ao MBone. Este en-

vio de pacotes SDP deve seguir estritamente os padrões descritos na RFC 2327, o que inclui a construção do protocolo de transporte SAP (*Session Announcement Protocol*) [Handley, Perkins e Whelan 2000].

Usuários com perfil de administrador de rede podem acessar MIBs de roteadores, para colherem informações quanto a configuração de multicast dos mesmos, conforme ilustrado na figura 4.9.

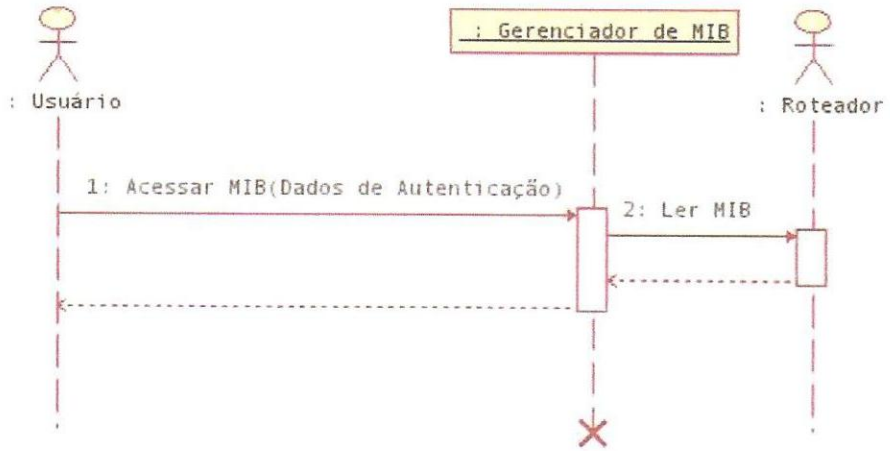

Figura 4.9: Acessar a MIB

Pelo fornecimento de algumas informações (a exemplo do endereço de grupo e porta de destino), conforme ilustra a figura 4.10, o usuário pode gerar tráfego de dados de tamanhos desprezíveis ou com carga útil.

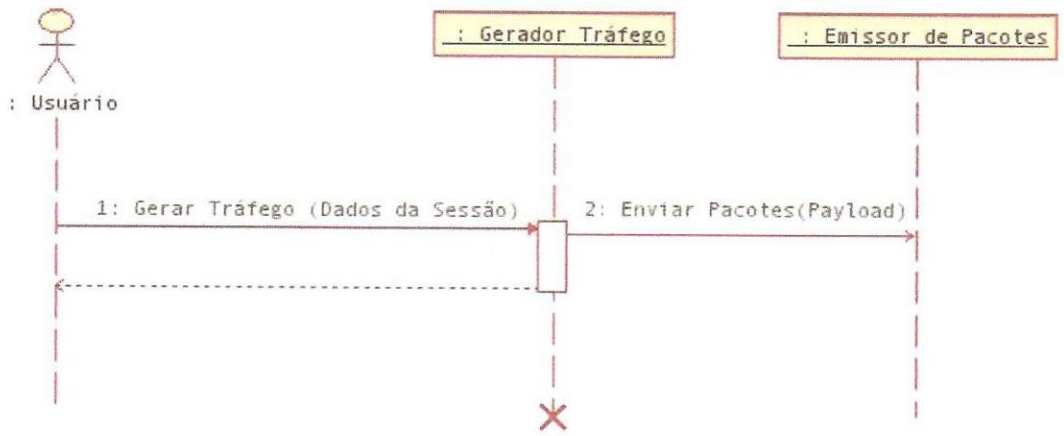

Figura 4.10: Gerar Tráfego

A partir de informações colhidas em pacotes SDP, o ambiente monta relatórios estatísticos passíveis de serem apresentados a um usuário que os solicitar, conforme explicitado no diagrama da figura 4.11.

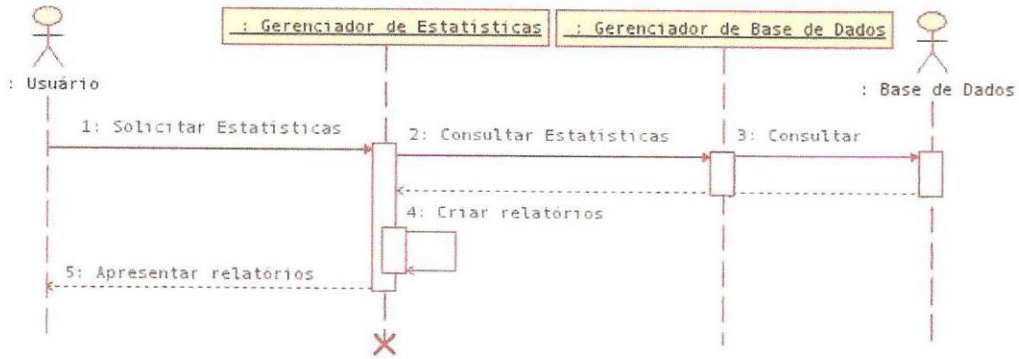

Figura 4.11: Consultar Estatísticas

O usuário pode participar de sessões de bate papo, sem a ciência de que todos os dados trafegam em multicast, conforme ilustrado na figura 4.12.

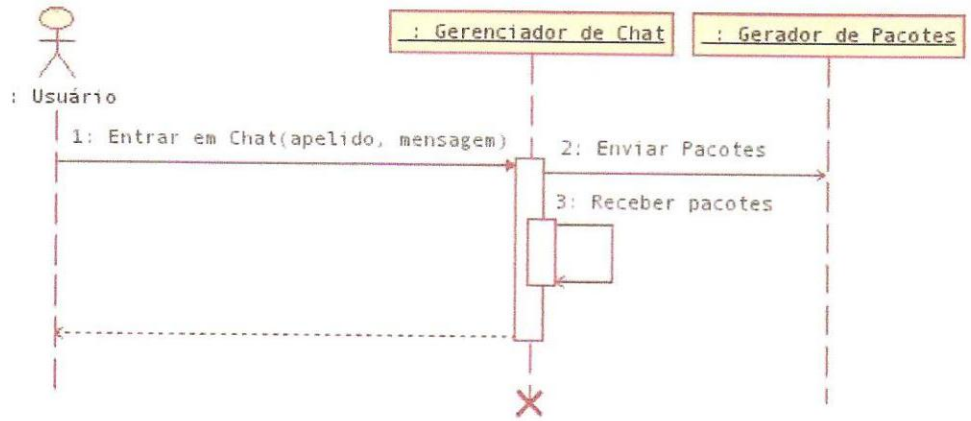

Figura 4.12: Entrar em (*chat*)

Para fazer uso integral das funcionalidades do ambiente o usuário precisa estar inserido em uma rede com capacidade de transmissão baseada em multicast. Para certificar-se de tal prerrogativa, conforme ilustrado na figura 4.13, o usuário pode testar a conectividade multicast.

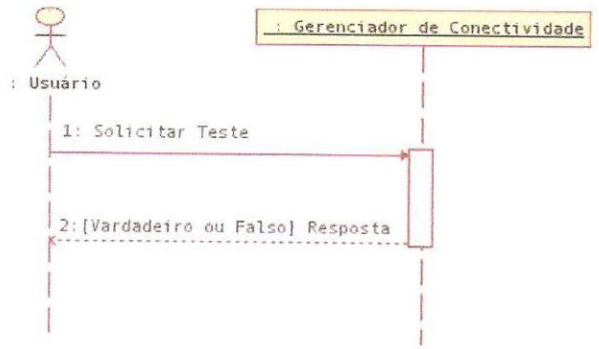

Figura 4.13: Testar conectividade

A partir dessa especificação formal do ambiente integrado foi possível proceder à implementação do mesmo, conforme apresentado no próximo capítulo.

5 *Implementação*

Atendendo aos requisitos listados na seção 4.1, adotou-se tecnologias aplicadas ao ambiente Web como solução central para o desenvolvimento do ambiente, uma vez que seu uso permite que o ambiente seja composto de uma aplicação servidor, responsável pelo recebimento de pacotes multicast da rede, e aplicações cliente, executadas no navegador, com a finalidade de permitir ao usuário interagir com as sessões anunciadas no MBone. Essa solução, ao isolar as funções de manipulação do tráfego multicast da interface com usuário, garante que todos os aspectos de configuração das ferramentas multicast sejam concentradas na aplicação servidor, proporcionando total transparência dessas atividades ao usuário final do ambiente. Além disso, por usar tecnologias baseadas em Web, faz com que o usuário sinta-se familiarizado em acessar o ambiente a partir de seu navegador.

5.1 Visão global do ambiente

A proposta desse trabalho consiste no desenvolvimento de um ambiente capaz de manipular o tráfego multicast, de modo que o usuário não precise instalar, em sua estação, nenhuma ferramenta de videoconferência baseada na transmissão multicast. Todas as funcionalidades das ferramentas atuais são agregadas em um único ambiente, acessado através de um navegador e disponível a qualquer usuário na Web. O usuário pode participar de sessões multicast do Mbone e, caso seja um administrador da rede, pode gerenciar o tráfego multicast através de funções de consulta de estatísticas, acesso a MIB dos roteadores, teste de conectividade multicast e pode ainda gerar anúncios e tráfego multicast.

O ambiente, escrito em Java, é composto por um objeto servidor, responsável por ouvir os pacotes multicast da rede, enquanto objetos clientes tem a finalidade de permitir ao usuário interagir com as sessões. Conforme ilustrado na figura 5.1, um servidor Web, hospedado na mesma máquina do ambiente, recebe as requisições do cliente e fornece as páginas concernentes às funções do ambiente requisitadas pelo usuário.

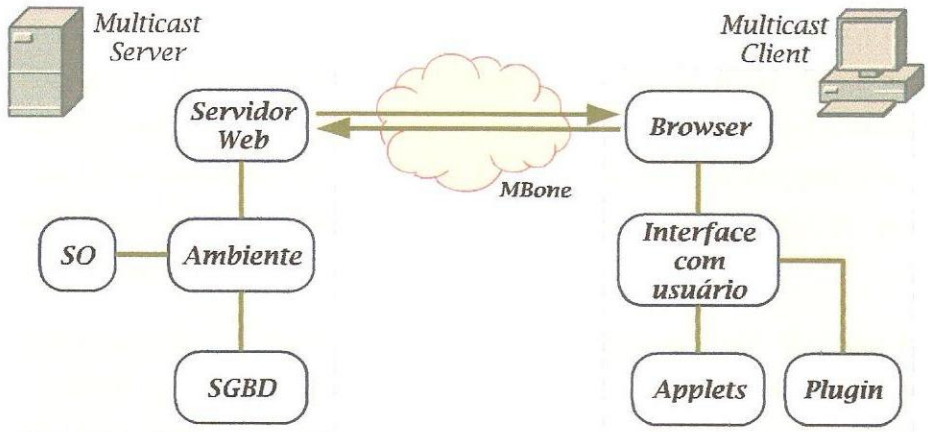

Figura 5.1: Topologia do ambiente

O servidor ouve as sessões multicast e, por meio de um SGBD (*Sistema Gerenciador de Bancos de Dados*), alimenta a base de dados com as informações contidas nos respectivos pacotes SDP. Quando um usuário entra no ambiente e solicita uma lista das sessões anunciadas, o ambiente apresenta as informações contidas nesse banco. O usuário, através de formulário provido pelo ambiente, também pode criar um anúncio de sessão multicast. Para tal, o servidor colherá as informações do formulário do cliente e gerará pacotes SDP.

Ao selecionar uma sessão anunciada no ambiente, o usuário pode assistir à transmissão em execução, seja esta de áudio e/ou vídeo. Ao receber uma solicitação deste tipo, o servidor do ambiente monta uma resposta HTML (*HyperText Markup Language*) contendo um *plugin*[1] de videoconferência e a envia ao usuário. Este *plugin* é configurado automaticamente pelo servidor: conforme o grupo que tenha sido selecionado, o servidor fica encarregado de gerenciar, de forma transparente ao cliente, as etapas necessárias do estabelecimento de sua participação no grupo multicast. A existência deste *plugin* se faz necessária, visto que não pertence ao escopo desta proposta de ambiente interpretar as diversas codificações de vídeo e áudio usadas atualmente.

Para que o ambiente funcione adequadamente, o equipamento do usuário precisa estar inserido em uma rede com Multicast IP habilitado. Caso o usuário não esteja certo desta prerrogativa, o ambiente disponibiliza um *link* para testar sua conectividade.

Caso deseje participar de uma sessão não listada pelo ambiente, o usuário pode requisitar, através de formulário, o fluxo de um determinado grupo multicast, se já conhecer de antemão os parâmetros deste fluxo.

[1]Programa que é acoplado a um aplicativo para ampliar suas funções

O ambiente, a partir das informações em seu banco de dados, fornece algumas estatísticas consultadas abertamente por usuários do ambiente. Estas estatísticas podem ser de interesse a um administrador de rede e consistem em informações colhidas da rede e dos usuários. O ambiente também é capaz de acessar a MIB de um roteador, conforme suas políticas de segurança, e colher informações sobre a configuração do protocolo de roteamento multicast PIM.

O ambiente também disponibiliza ao usuário a funcionalidade de gerar uma carga mínima de tráfego multicast. Essa funcionalidade pode ter utilidade para o usuário desenvolvedor/administrador que desejar consultar a tabela de rotas multicast do roteador que serve à sua LAN (*Local Area Network*). Por exemplo, ao gerar esta carga mínima, os roteadores ligados à LAN que estiverem corretamente configurados, atualizarão suas tabelas de rota multicast evidenciando a presença do novo grupo.

Além destas funcionalidades, o usuário também pode interagir em um *chat* com outros usuários do ambiente com transmissão em multicast. O ambiente se comporta de tal forma, que um usuário leigo pode usufruir de suas funções de videoconferência, sem a percepção que todas as informações trafegam em Multicast IP.

Algumas aplicações precisam ser executadas na estação do cliente, portanto, conforme ilustra a figura 5.1, é preciso que o mesmo possua o *plugin* de videoconferência e a JVM (*Java Virtual Machine*), responsável por executar as classes Java.

5.2 Tecnologias usadas

Nesta seção são descritos os principais recursos usados na implementação do ambiente proposto. O ambiente é totalmente desenvolvido em soluções livres, em que o fator desempenho é sempre mensurado em cada escolha de tecnologia usada para a criação do mesmo. A figura 5.2 ilustra não somente as tecnologias escolhidas para o servidor que hospedará o ambiente proposto, como também ilustra o sentido dos fluxos de informação do ambiente.

5.2.1 Sistema Operacional

O núcleo do Linux garante grande estabilidade e autonomia de funcionamento aos servidores em que é instalado. As três principais características que tornam o sistema operacional Linux extremamente adequado para hospedar aplicações servidores são: ex-

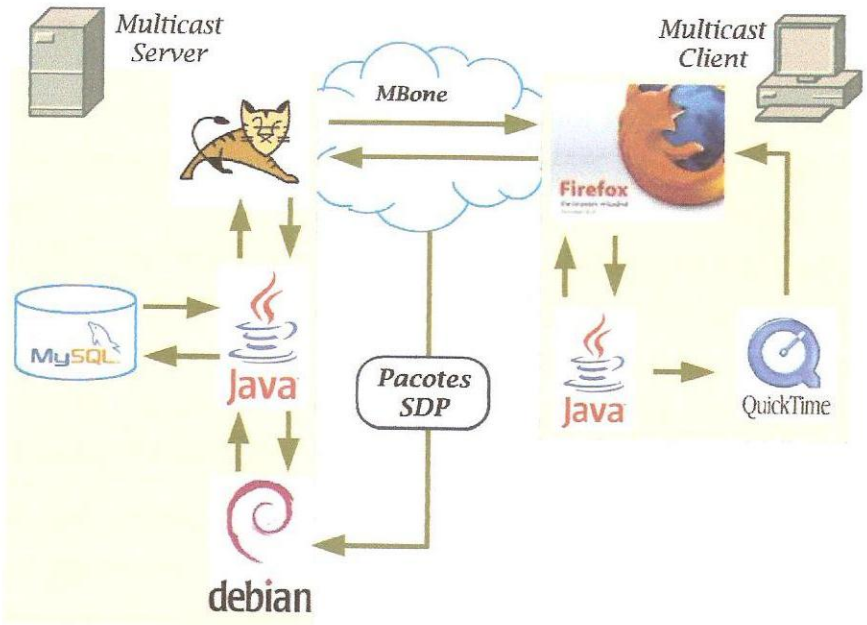

Figura 5.2: Tecnologias utilizadas

trema preocupação de seus desenvolvedores quanto à segurança, sua alta capacidade de configuração e o fato de possuir código livre e aberto. Multicast é habilitado por padrão no *kernel* [2] do Linux, o que significa que a capacidade de manipular tráfego multicast é nativa ao sistema operacional. Apenas quando se deseja habilitar roteamento multicast se faz necessário recompilar o *kernel*, adicionando módulos avançados de roteamento [Goyeneche 1998].

A distribuição Linux Debian, além de ser de código livre e aberto, é totalmente sem fins lucrativos, o que não ocorre com outras distribuições que possuem grandes empresas associadas a elas. Embora não hajam versões separadas para estações de trabalho e servidores, cada versão só é lançada quando são feitos exaustivos testes de segurança e estabilidade, o que faz com que o Debian seja perfeitamente adequado para o uso em servidores.

A escolha da distribuição foi feita ao gosto do autor, embora distribuições do Linux também possam ser usadas para hospedar o ambiente proposto. Devido a portabilidade inerente à linguagem Java, o ambiente poderia também ser instalado em plataformas Windows, no entanto, isto fere o propósito de usar tecnologias de código livre.

[2] Núcleo de um sistema operacional. É a parte do sistema operacional que primeiro carrega na memória principal, responsável por gerenciar e controlar o acesso ao sistema de arquivos, à memória, à tabela de processos e o acesso aos dispositivos e periféricos, entre outras atribuições.

5.2.2 Linguagem de programação

Totalmente orientada a objeto, Java é uma versátil linguagem de programação que permite ao programador usar seus objetos de aplicações locais para páginas Web, aplicações em rede ou em banco de dados [Horton 1997]. Essa flexibilidade de escrever um objeto para diferentes contextos dá ao programador o poder de manipular dados entre estas diferentes situações. A capacidade de interagir simultaneamente com protocolos de rede, banco de dados e requisições HTML serve como aparato para a criação de um ambiente capaz de, a partir de requisições de um usuário Web, manipular fluxos de rede e guardar os dados em um banco.

Uma das mais poderosas funcionalidades de Java é sua capacidade de manipular e interagir com protocolos de rede. Os recursos de Java para rede estão agrupados em diversos pacotes, porém, os fundamentais estão definidos em classes e interfaces do pacote *java.net* [Deitel e Deitel 2003]. Através do *java.net* é possível estabelecer comunicações baseadas em *sockets*, que possibilita a transmissão em fluxo de dados orientados à conexão, ou em pacotes não orientados à conexão [Ingram 2002].

A partir da JDK 1.1 (*Java Development Kit*), a classe *MulticastSocket*, decendente da classe *DatagramSocket*, passou a integrar o pacote *java.net*. Esta classe implementa os métodos *joinGroup()* e *leaveGroup()* para gerência das sessões multicast [Oser 2001]. Estes métodos assumem a responsabilidade pelo uso adequado do IGMP. No início do desenvolvimento da tecnologia multicast, a única forma de limitar o alcance dos pacotes era pelo correto uso do TTL, por isso, a classe *MulticastSocket* também implementa um método para o usuário manipular o campo de TTL dos pacotes [Wakeman 2004].

Com o uso da API (*Applications Programming Interfaces*) JDBC (*Java Database Connectivity*), criada em 1996 pela *Sun Microsystems*, é possível escrever aplicações que envolvam banco de dados em linguagem puramente Java [Anselmo 2001]. O amplo acesso às informações contidas em bancos de dados relacionais é feito pela escrita de códigos com consultas SQL (*Structured Query Language*).

A base do desenvolvimento de aplicações para Web usando a linguagem Java é a tecnologia *Servlet* [Kurniawan 2002]. Um *servlet* é uma classe Java, que pode ser automaticamente carregada e executada por um servidor Web especial, chamado de "contentor de servlet". *Servlets* interagem com clientes por meio de um modelo de solicitação-resposta baseado em HTTP. O JSP (*Java Server Pages*) na verdade é uma extensão da tecnologia *servlet*.

Algumas aplicações de Java para Web requerem que instruções não sejam executadas no servidor (contentor de servlet), e sim na estação cliente. Pequenos programas escritos em Java, chamados *Applets*, são embutidos em páginas HTML, que são executados no lado do cliente [Hoff, Shaio e Starbuck 1996]. Applets Java já foram a grande solução para dar dinamismo e efeitos especiais a páginas Web, no entanto, isso tem um considerável custo de processamento na estação. Outras soluções já existem com muito mais recursos e mais leves para dar dinamismo estético às páginas, tais como: Macromedia Flash e Action Script. Porém, a capacidade de executar códigos avançados no cliente, a exemplo de emitir e capturar pacotes de rede, ainda é um privilégio dos *Applets*.

Outras linguagens de programação também possuem recursos similares aos oferecidos pela tecnologia Java. Entretanto, apenas agregando recursos de diferentes linguagens poderia se construir um ambiente com código equivalente ao proposto.

5.2.3 Servidor Web

O Tomcat, mais popular contentor *servlet*, originalmente desenvolvido pela *Sun Microsystems*, em outubro de 1999 teve seu código fonte entregue à *Apache Software Fundation*. Quando pertencente ao grupo Apache, o Tomcat foi incluído ao projeto Jakarta, que recebeu inúmeras contribuições que culminaram na versão 3.0, logo seguida pela 3.3. A atual versão 4.0 foi rebatizada com o nome "Catalina" e possui uma arquitetura totalmente nova e dedicada ao alto desempenho e flexibilidade. O Tomcat também é um servidor Web, o que o torna apto a não somente responder solicitações de páginas JSP, mas também páginas puramente HTML. Somente na primeira vez em que é requisitada, toda página JSP é convertida em um *servlet*, tornando-se, portanto, uma classe Java [Kurniawan 2002].

Associado ao ambiente proposto, o servidor Web Tomcat será responsável por receber as requisições HTTP (*Hypertext Transfer Protocol*), executar o código Java/JSP e fornecer as respostas ao cliente. Portanto, o servidor Web é a interface entre ambiente e usuário. Embora existam outros servidores alternativos ao Tomcat, este torna-se mais adequado ao uso nesse ambiente: sendo o servidor Web Java oficial da Sun, vários esforços são feitos pelo fornecedor no sentido de garantir a estabilidade do produto.

5.2.4 Banco de Dados

Um dos mais populares bancos de dados relacionais é o banco gratuito MySQL, desenvolvido pela T.c.X DataKonsultAB. Além de ter versões para diversas plataformas, o

MySQL apresenta muitas outras características que o faz um dos banco de dados mais usados no mundo. São algumas destas características: lidar com um número teoricamente ilimitado de usuários, manipular mais de 50 milhões de registros, execução extremamente rápida dos comandos, sistema de segurança simples e funcional, entre outras [Suehring 2002]. A principal característica do MySQL é sem dúvida a grande velocidade de resposta aliada ao alto nível de estabilidade. O MySQL é um banco de dados de poucos recursos, se comparado a soluções proprietárias, no entanto é um *software* livre e toda sua documentação está disponível em seu *site*.

O ambiente proposto não exige demasiados recursos da base de dados: nenhuma operação complexa, a exemplo de sub-consultas e gatilhos, é usada pelo ambiente. No entanto, por tratar-se de uma aplicação de tempo real, a velocidade do banco é um fator de extrema importância. Diante disto, o banco de dados MySQL mostra-se como solução adequada ao ambiente, uma vez que não possui recursos avançados, é de código livre e responde a consultas em altíssima velocidade.

5.3 Arquitetura do ambiente

5.3.1 Diagrama de classes

O Diagrama de Classes mostrado na figura 4.3, obtido durante a fase de modelagem, apresenta uma visão simplificada do Diagrama de Classes apresentado na figura 5.3. Embora o Diagrama de Classes não explicite as mensagens trocadas entre as classes, através da figura 5.3 é possível perceber a interação entre as mesmas, conhecendo seus métodos e tipos de dados.

Algumas das classes criadas para o ambiente possuem funções específicas, a exemplo de enviar pacotes de rede e criar conexões com o banco de dados. Estas classes com funções especializadas possuem objetos instanciados por outras classes que desejam fazer uso de suas funções, ilustrados com setas pontilhadas na figura 5.3.

Algumas das classes que compõem o ambiente foram construídas em arquivos JSP, a exemplo das classes *GetSessions* e *SdpSender*. Estas classes foram construídas de forma estrutural, não orientada a objeto, no entanto foram adicionadas ao Diagrama de Classes para que seus métodos sejam explicitados.

Ainda através da figura 5.3, é possível perceber que a classe *AccessMIB* herda recursos da classe *AdventNet SNMP API 4*.

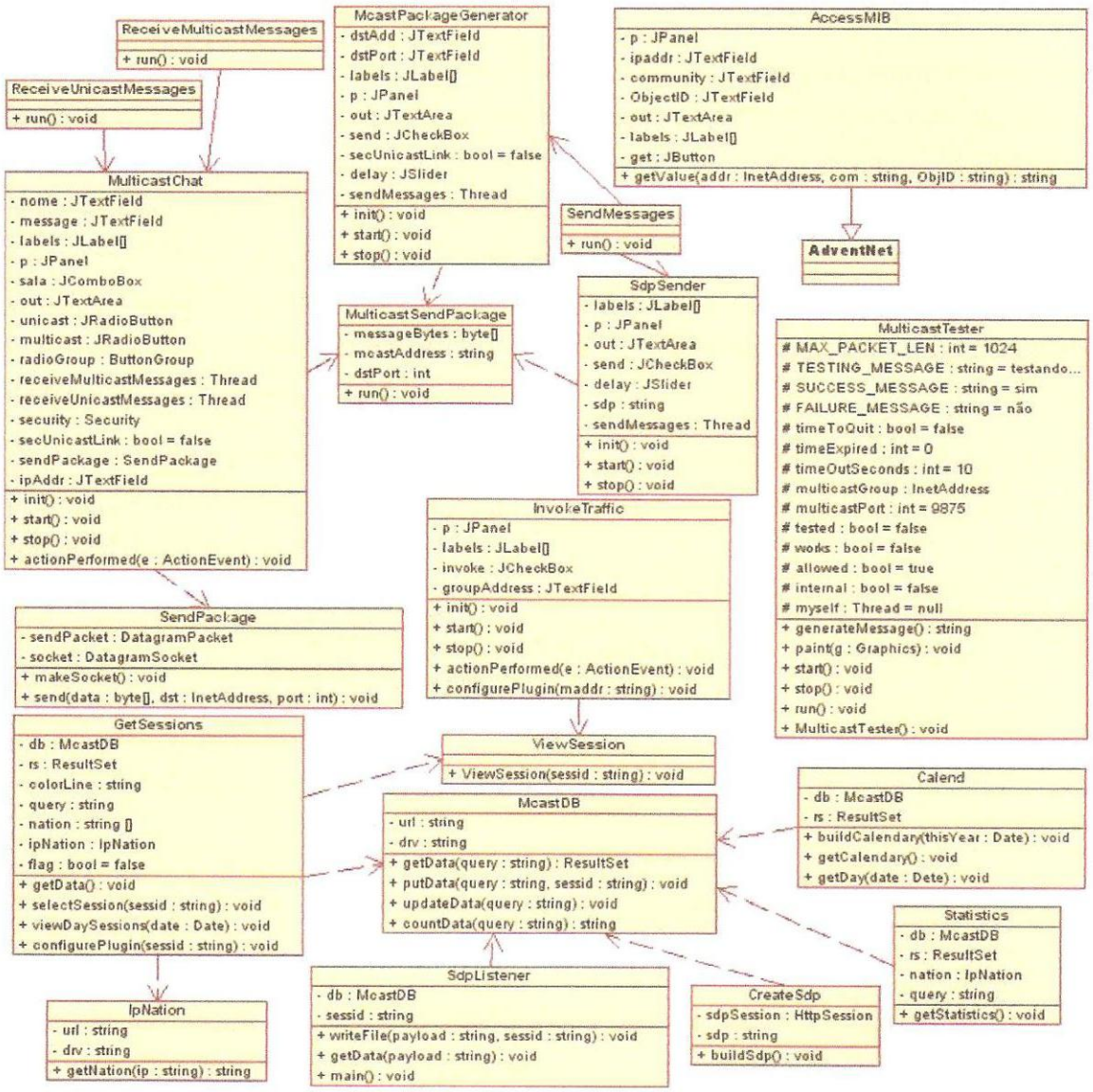

Figura 5.3: Diagrama de Classe do Ambiente Integrado

5.3.2 Dicionário de classes

O ambiente integrado é formado por um conjunto de classes que se relacionam umas com as outras. Nesta seção são relacionadas as classes desenvolvidas para a construção do ambiente, descrevendo-as sucintamente e fazendo menção aos principais métodos e atributos utilizados por cada uma.

Construída em JSP, a classe *GetSessions*, ilustrada na tabela 5.1, tem por finalidade listar as principais informações dos anúncios de sessão que constam na base de dados. Caso haja uma data passada como parâmetro na URI (*Uniform Resource Identifier*), a classe aciona o método *viewDaySessions* e lista apenas as sessões anunciadas naquela data. Caso o usuário selecione um anúncio de sessão, o ambiente prepara uma página e configura um *plugin* para que a sessão selecionada possa ser exibida. Nesta classe há instâncias das classes IpNation e McastDB que são as duas classes responsáveis pelo acesso à base de dados.

Com intervalos de 1 minuto, a classe *GetSessions* aplica um auto-*reload* para que a página que está sendo exibida ao usuário possa estar o mais atualizada possível com o conteúdo da base de dados.

Classe	**GetSessions**
Atributos	- db : McastDB - rs : ResultSet - ipNation : IpNation
Métodos	+ getData() : Void + selectSession(sessid : string):void + viewDaySessions(data : Date):void + configurePlugin(sessid:string):void

Tabela 5.1: Classe **GetSessions**

Todas as classes que de algum modo necessitam ter acesso ao banco fazem uso de instâncias da classe *McastDB*, ilustrada na tabela 5.2. A classe *McastDB* é responsável pela execução do *driver* e pela gerência das conexões com o banco de dados. Os métodos *getData()*, *putData()*, *updateData()* e *countData()* servem para realizar as operações básicas de acesso ao banco, respectivamente: consultar, inserir, atualizar e contar.

A classe *IpNation* tem por finalidade recuperar o país de origem dos anúncios das sessões multicast, fazendo isto através de seu único método, conforme ilustrado na tabela 5.3. Esta classe usa um banco de dados separado que armazena todas as faixas de endereço IP pertencentes a cada país.

Classe	**McastDB**
Atributos	- url : string - drv : string
Métodos	+ getData(query : string):ResultSet + putData(query : string, sessid : string):void + updateData(query : string):void + countData(query : string):string

Tabela 5.2: Classe **McastDB**

Classe	**IpNation**
Atributos	- url : string - drv : string
Métodos	+ getNation(ip : string):string

Tabela 5.3: Classe **IpNation**

A classe *Calend* é responsável por construir uma agenda anual com os dias marcados para haver transmissões multicast, conforme agendadas nos anúncios. Ilustrada na tabela 5.4, a classe *Calend* também é construída em JSP e alguns métodos triviais de *JavaScript*, usados para a gerar do calendário anual. Os dias em que há sessões agendadas são *links* criados pelo método *getDay()*, que uma vez acionados, invocam a classe *GetSessions* para listar as sessões anunciadas para aquele dia.

Classe	**Calend**
Atributos	- db : McastDB - rs : ResultSet
Métodos	+ buildCalendary(thisYear : Date):void + getCalendary():void + getDay(date : Date):void

Tabela 5.4: Classe **Calend**

A classe *SdpListener* é o coração do servidor multicast, ela assume o comportamento de servidor "ouvindo" a porta apropriada a espera de novos pacotes SDP. Quando o método *main()* percebe a chegada de um pacote SDP, ele retira o protocolo de transporte SAP (*Session Announcement Protocol*) e envia somente a área de dados do SDP para o método *getData()* tratar estes dados e inserir-los no banco de dados. Este tratamento se faz necessário pelos diversos campos opcionais existentes no SDP, definidos pela RFC 2327 [Handley e Jacobson 1998].

A classe *SdpListener*, ilustrada na tabela 5.5, através do método *writeFile()* guarda no sistema de arquivos do servidor uma cópia dos dados de todos os pacotes SDP. Isto se

faz necessário por dois motivos: em primeiro lugar para pacotes SDP mal formados não causarem inconsistências no banco de dados e, o segundo motivo, para satisfazer certas exigências de alguns *plugins* de vídeo. O *plugin* do Quicktime, usado neste trabalho, se faz mais adequado por condizer estritamente com os padrões definidos, não necessitando que este recurso, de salvar cópias dos pacotes, seja usado. No entanto, a exemplo do Kaffeine e Videolan, alguns *plugins* carecem deste recurso por características limitantes de seu funcionamento.

Classe	**SdpListener**
Atributos	- db : McastDB - sessid : string
Métodos	+ writeFile(payload : string, sessid : string):void + getData(payload : string):void + main():void

Tabela 5.5: Classe **SdpListener**

O processo de criação de um anúncio de sessão multicast se dá em duas etapas: a primeira, com as respostas do usuário aos formulários que irão colher informações para a composição do pacote SDP, e a segunda etapa é o envio do pacote SDP por um *applet* na máquina do usuário. Da primeira etapa de criação de pacotes SDP trata a classe *CreateSdp*. Ilustrada na tabela 5.6, esta classe é responsável em apresentar formulários ao usuário e tratar os dados por eles colhidos com JSP. A medida em que o usuário responde as perguntas dos formulários, a classe monta o pacote SDP e, ao fim, invoca o *applet* para que o pacote seja enviado.

Classe	**CreateSdp**
Atributos	- sdpSession : HtpSession - sdp : string
Métodos	+ buildSdp():void

Tabela 5.6: Classe **CreateSdp**

A classe *SdpSender*, ilustrada na tabela 5.7, é o *applet* responsável pelo envio dos pacotes SDP montados pela classe *CreateSdp*. O atributo *out* é uma área de texto onde se encontra a área de dados do pacote SDP a ser enviado. Caso o usuário deseje fazer alguma alteração, ainda pode fazê-la diretamente neste atributo, como ilustrado na figura 5.18. O método *init()* é executado apenas uma vez, e é responsável apenas por declarações de variáveis e inicialização da interface gráfica. Já no método *start()* se encontra todo o funcionamento da classe: o método *start()* colhe o conteúdo da área de dados do pacote SDP, e constrói o protocolo de transporte SAP.

Classe	**SdpSender**
Atributos	- out : JTextarea - sdp : string - delay : JSlider
Métodos	+ init():void + start():void + stop():void

Tabela 5.7: Classe **SdpSender**

No momento que o método *start()* da classe *SdpSender* precisa enviar um pacote SDP, a classe interna privada *SendMessages* é instanciada. Ilustrada na tabela 5.8, esta classe tem a finalidade de gerenciar o tempo de envio de forma ininterrupta, com intervalos em segundos, determinados pelo usuário, através do atributo *delay*.

Classe	**SendMessages**
Atributos	
Métodos	+ run():void

Tabela 5.8: Classe **SendMessages**

Enquanto que a classe *SdpSender* gerencia o ambiente gráfico e a classe *SendMessages* administra o tempo de envio, a classe *MulticastSendPackage* é efetivamente a responsável por enviar pacotes multicast. Ilustrada na tabela 5.9, a classe recebe o conteúdo a ser enviado da classe *SdpSender*, SAP/SDP, e envia para o endereço e porta de grupo determinados 224.2.127.254:9875 [Handley e Jacobson 1998].

Classe	**MulticastSendPackage**
Atributos	- messageBytes : byte[ ] - mcastAddress : string - dstPort : int
Métodos	+ run():void

Tabela 5.9: Classe **MulticastSendPackage**

A tabela 5.10 apresenta a classe *McastPackageGenerator*, que dá ao usuário a funcionalidade de gerar pacotes multicast em seu próprio segmento. Os dados que o usuário deseja enviar podem ser colocados na área de texto do atributo *out* e, após preenchidos os campos *dstAdd* de endereço de destino e *dstPort* de porta de destino, os dados são enviados repetidamente, a cada intervalo de tempo *delay* selecionado.

Da mesma forma que a classe *SdpSender*, a classe *McastPackageGenerator* também

faz uso da classe *MulticastSendPackage* para enviar pacotes multicast, e também possui uma classe interna privada *SendMessages* para administrar o tempo de envio. A classe *SendMessages* é um *Thread*[1], possibilitando que ao mesmo tempo em que a interface gráfica esteja em execução, e mesmo que o usuário ainda esteja editando um texto a ser enviado, se a chave de envio *send* estiver acionada, pacotes SDP serão enviados em *background*.

Classe	**McastPackageGenerator**
Atributos	- dstAdd : JTextField - dstPort : JTextField - out : JTextarea - sendMessages : string - delay : JSlider - send : JCheckBox
Métodos	+ init():void + start():void + stop():void

Tabela 5.10: Classe **McastPackageGenerator**

Ilustrada na tabela 5.11, a classe *InvokeTraffic* é capaz de invocar um determinado tráfico multicast para o segmento do usuário que o requisitou, através do protocolo IGMP. Para que um fluxo multicast seja requisitado através do IGMP, a única exigência é o endereço de grupo multicast. No entanto, se o usuário desejar ver a sessão do tráfego invocado é preciso que mais informações sejam fornecidas, tais como porta de destino, codificação de vídeo, dentre outras, conforme a necessidade do *plugin* usado. Apenas invocar o tráfego, sem ver o que está sendo transmitido, pode ser útil para testar a conectividade multicast do segmento, ou ainda apenas para comprovar a transmissão do fluxo.

Classe	**InvokeTraffic**
Atributos	- invoke : JCheckBox - groupAddress : JTextField
Métodos	+ init():void + start():void + stop():void + actionPerormed(e : ActionEvent):void + configurePlugin(maddr : string):void

Tabela 5.11: Classe **InvokeTraffic**

[1]Threads são fluxos de execução que funcionam em paralelo com aplicação que os originou.

O método *start()* da classe *InvokeTraffic* usa o método *joinGroup()* da classe *MulticastSocket* para enviar mensagens *IGMP.Report*, solicitando aos roteadores do segmento o tráfego multicast indicado.

A classe *ViewSession*, construída em JSP, tem como única finalidade servir como interface com o usuário, para exibição do conteúdo de uma sessão selecionada por ele. É nesta classe, interface, que estará o *plugin* de videoconferência previamente configurado. O desenvolvimento desta classe sugere especializações, por meio de herança, para comportar diferentes tipos de *plugins*.

Conforme ilustra a tabela 5.12, a classe *ViewSession* não necessita de atributos por tratar apenas de uma interface com o usuário.

Classe	**ViewSession**
Atributos	
Métodos	+ ViewSession(sessid : string):void

Tabela 5.12: Classe **ViewSession**

Para testar a conectividade multicast o usuário pode fazer uso da classe *MulticastTester*, ilustrada na tabela 5.13. Esta classe é uma adaptação do *applet* desenvolvido por Marty Schoch [Schoch 2002] e tem como funcionamento invocar um tráfego multicast e aguardar um determinado tempo a sua chegada: caso isso ocorra significa que o segmento testado está com multicast habilitado. Caso o tempo de espera do fluxo expire, entende-se que o segmento não está com multicast habilitado.

Classe	**MulticastTester**
Atributos	- multicastGroup : InetAddress - multicastPort : int - timeExpired : int = 0
Métodos	+ run():void + start():void + stop():void + generateMessage():string

Tabela 5.13: Classe **MulticastTester**

Para que a classe funcione corretamente é preciso invocar um tráfego multicast que tenha transmissão ininterrupta, portanto, o ambiente integrado invoca o tráfego de anúncios de sessão para testar a conectividade. No entanto, para que não ocorra um "falso negativo" é preciso que o roteador que serve o segmento não esteja bloqueando apenas o tráfego de anúncio de sessões multicast e repassando os demais. Caso seja um roteador

CISCO, é preciso que o comando "`ip sap listen`" tenha sido executado [Parkhurst 1999]. Caso contrário, apenas o tráfego multicast para o grupo 224.2.127.254, anúncio de sessões, será barrado. Em roteadores IBM o grupo de anúncio de sessão é tratado como os demais, portanto não há necessidade de comandos prévios.

A classe *MulticastChat*, representada na tabela 5.14, é responsável por inicializar os componentes gráficos do *chat* multicast, e ainda tratar as mensagens oriundas dos outros usuários. Esta classe faz uso de duas outras classes, *ReceiveUnicastMessages* e *ReceiveMulticastMessages*, que são *threads* que funcionam como servidores à espera de novas mensagens, em *unicast* ou multicast, respectivamente. Quando a classe *ReceiveMulticastMessages* recebe alguma mensagem, ela entrega ao método *start()* da classe *MulticastChat*, que trata a mensagem colocando-a no formato "`Apelido@IP>Mensagem`". Quando um usuário escreve e envia uma mensagem, este mesmo método a coloca no formato "`Apelido>Mensagem`". Quando um usuário escreve uma mensagem e pressiona *Enter* para o seu envio, a classe *MulticastSendPackage* é instanciada para enviar a mensagem. Logo em seguida, a classe *ReceiveMulticastMessages* recebe a mensagem enviada pelo próprio usuário e a entrega para a classe *MulticastChat*, para que a mensagem seja exibida através do atributo *out : JTextArea*.

Classe	**MulticastChat**
Atributos	- sala : JComboBox - nome : JTextField - message : JTextField - out : JTextArea - unicast : JRadioButton - multicast : JRadioButton - receiveMulticastMessages : Thread - receiveUnicastMessages : Thread - receiveMulticastMessages : Thread - ipAddr : JTextField
Métodos	+ init():void + start():void + stop():void + actionPerormed(e : ActionEvent):void

Tabela 5.14: Classe **MulticastChat**

Para a conversação em multicast, o usuário pode selecionar uma sala pré-definida através do atributo *sala : JComboBox*, ou ainda escolher um endereço multicast aleatório, em prévio acordo com os outros participastes, e escrevê-lo no atributo *IpAddr : JTextField*. Caso o usuário deseje uma conversação privada e segura, ao selecionar o atributo *unicast :*

JRadioButton, a classe *MulticastChat* cria um par de chaves, pública e privada, e as negocia com a máquina do outro usuário, cujo IP deve ser informado no campo de endereço. Essa negociação de chaves é transparente ao usuário.

A classe ilustrada na tabela 5.15 tem por finalidade acessar a MIB de um roteador e fornecer objetos concernentes à configuração do PIM e IGMP. A classe *AccessMIB* usa os recursos da *AdventNet SNMP API 4* para ter acesso às MIBs *IGMP-MIB.my* e *PIM-MIB.my* implementadas nos roteadores CISCO desde o IOS 12.0(15)S [Cisco Systems 2003]. O pacote de APIs da *AdventNet* fornece suporte a diversos recursos que possibilitam ao usuário informar apenas a comunidade, o endereço IP do nó gerenciado e o identificador do objeto.

Classe	**AccessMIB**
Atributos	- community : JTextField - objectID : JTextField - out : JTextArea - ipAddr : JTextField
Métodos	+ init():void + start():void + stop():void + getvalue(addr : InetAddress, com : string, objID : string):string

Tabela 5.15: Classe **AccessMIB**

O ambiente ainda faz uso de algumas outras classes que não foram citadas nesta seção, por não pertencerem ao núcleo de funções do ambiente. Portanto, estas classes possuem funções coadjuvantes, criadas ao gosto do codificador e não são visíveis ou perceptíveis ao usuário. Algumas destas funções definem a estrutura de troca de mensagens, formatação de páginas HTML, armazenamento temporário de informação, dentre outras.

5.3.3 Diagramas de seqüência

Nessa seção são apresentados os diagramas de seqüência para cada caso de uso do ambiente, explicitando a comunicação entre as classes, descritas anteriormente.

Quando o servidor do ambiente constatar a chegada de um anúncio de sessão em seu segmento, ele capta as informações do pacote, trata-as e alimenta o banco de dados. Como especificado no diagrama da figura 5.4, o método *getData()* da classe *SdpListener* interpreta o conteúdo dos pacotes SDP, para constatar se o pacote SDP respeita os padrões definidos pela RFC. Caso o recém chegado pacote respeite os padrões, através do método

getData() da classe *McastDB*, o ambiente verifica se já consta no banco uma versão idêntica ou mais recente deste pacote: em caso afirmativo o ambiente o descarta, caso contrário invoca o método *putData()* e grava na base de dados todos os campos do pacote SDP. No mesmo instante em que a classe *SdpListener* invoca a inserção do pacote SDP na base de dados, o método *writeFile()* é invocado para gravar uma cópia do pacote SDP no sistema de arquivos local do servidor. Esta cópia redundante é gravada no sistema de arquivos para facilitar a adequação do ambiente a diferentes tipos de *plugins* de vídeo.

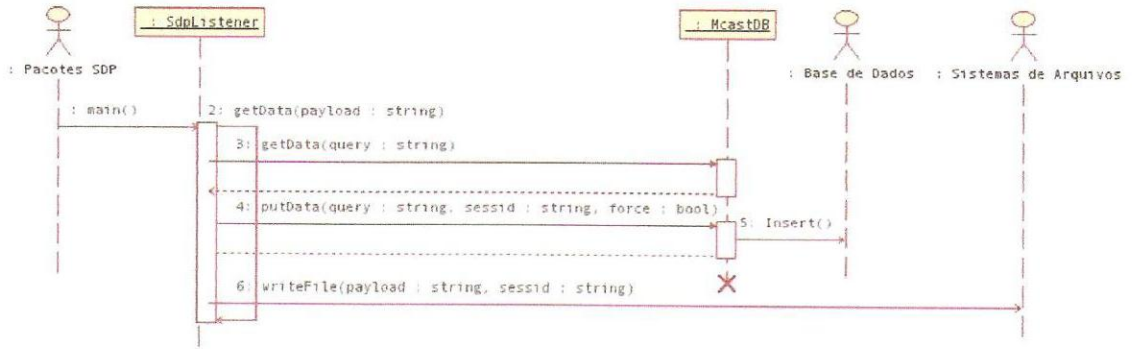

Figura 5.4: Gerenciar pacotes SDP

Quando um usuário do ambiente desejar consultar as sessões atualmente anunciadas, o ambiente consultará a base de dados onde constam todas as informações dos pacotes SDP propagados na rede, conforme mostra a figura 5.5. São também listadas as sessões que o próprio usuário criou, uma vez que os pacotes SDP são criados e enviados pelo usuário são recebidos pelo ambiente, da mesma forma que os demais anúncios do Mbone.

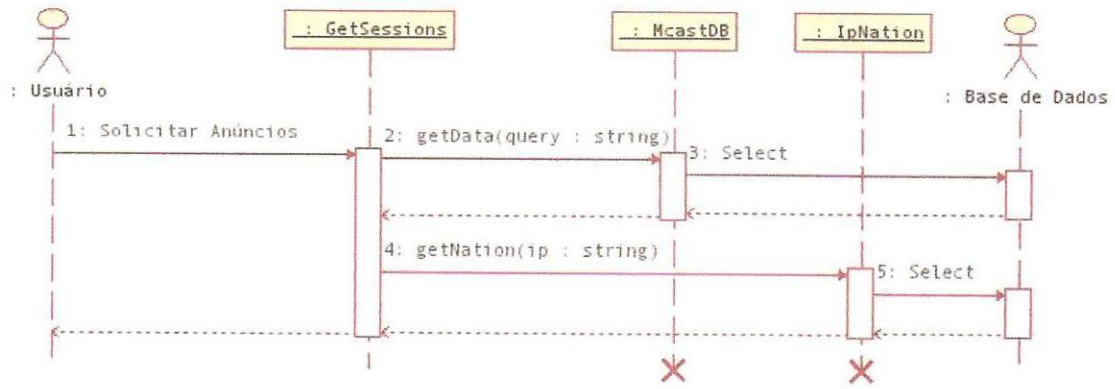

Figura 5.5: Visualizando lista de anúncios

A figura 5.6 mostra o processo de visualização do conteúdo de um grupo. Caso o usuário selecione uma sessão, através do método *selectSession()* da classe *GetSessions*, o ambiente acessará o banco dados para recuperar todos os detalhes do grupo selecionado.

Em seguida, o método *configurePlugin()* é invocado, para configurar o *plugin* de videoconferência para que este capte o fluxo da sessão. Finalmente, o construtor da classe *ViewSession* encarrega-se de apresentar ao usuário o conteúdo da sessão selecionada.

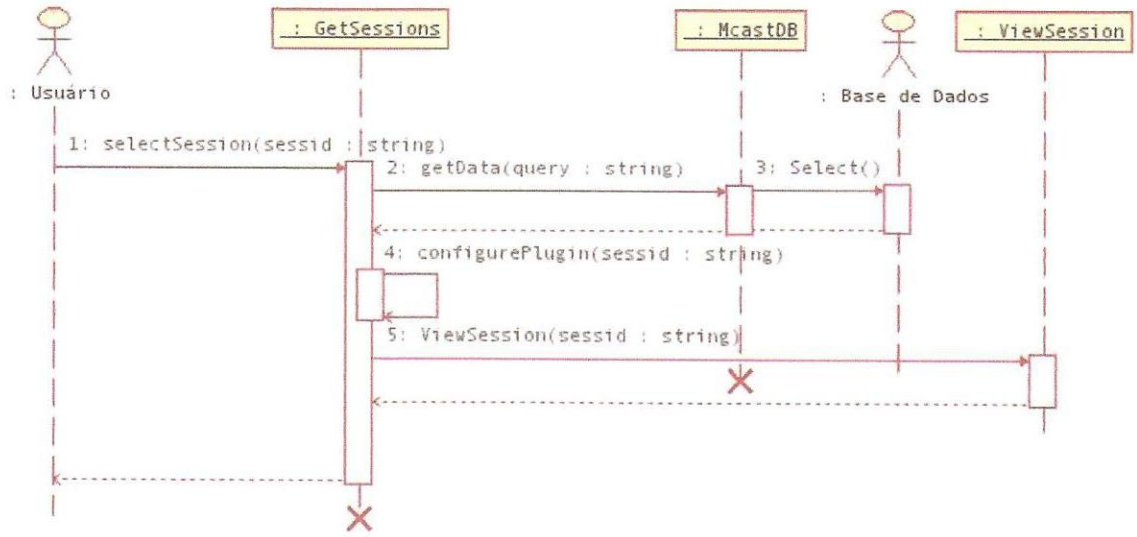

Figura 5.6: Visualizando o conteúdo de um grupo

Como mencionado na seção 2.4, é altamente provável que nem todas as sessões existentes estejam registradas na base de dados. Portanto, o ambiente permite que o usuário selecione um fluxo de um grupo multicast não registrado, mas que ele previamente conheça a existência. Para tal, como descrito no diagrama da figura 5.7, o usuário precisa preencher um formulário com os dados requeridos para a configuração do *plugin*. Este será enviado ao usuário para que ele possa iniciar a recepção do fluxo.

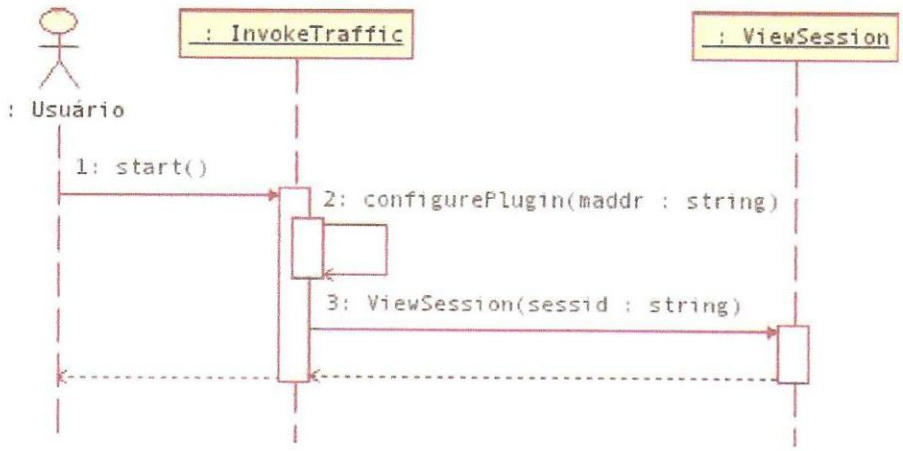

Figura 5.7: Solicitação de Tráfego

Para criar um anúncio de grupo multicast, o usuário deve preencher um formulário com pelo menos os ítens obrigatórios para a criação de um pacote SDP descritos na RFC

[Handley e Jacobson 1998]. Com estas informações o ambiente monta um pacote SDP, ilustrado no diagrama da figura 5.8, e envia ao usuário a classe *SdpSender* (construída em *applet*), para que a própria estação do usuário se encarregue de enviar o pacote SDP recém formado.

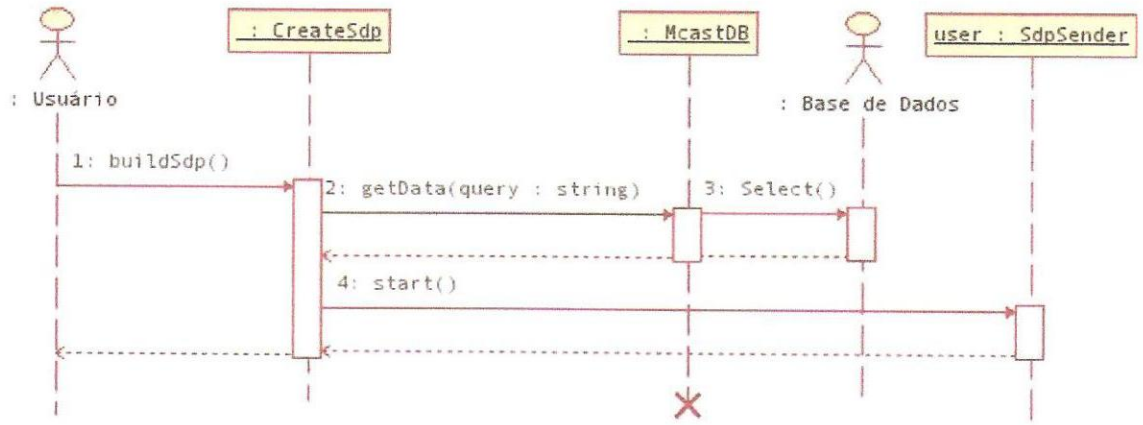

Figura 5.8: Geração anúncio de grupo

Sujeita aos padrões da RFC2934 [McCloghrie et al. 2000], a partir do IOS 12.0(15)S os roteadores Cisco já implementam a MIB para PIM [Cisco Systems 2003]. Através dela, consultando o RP [1] (*Rendezvous Point*), o usuário pode identificar mudanças na topologia multicast de sua rede. Além disso, pode-se monitorar o comportamento das interfaces quanto ao repasse de fluxo, existência de vizinhanças e tabelas de rota multicast, dentre outros. O roteador sob consulta, como apresentado no diagrama da figura 5.9, fornece passivamente os dados ao ambiente, que por sua vez os repassa ao cliente. Este caso de uso do ambiente tem caráter apenas de leitura, não permitindo ao usuário alterar dados nos objetos da MIB.

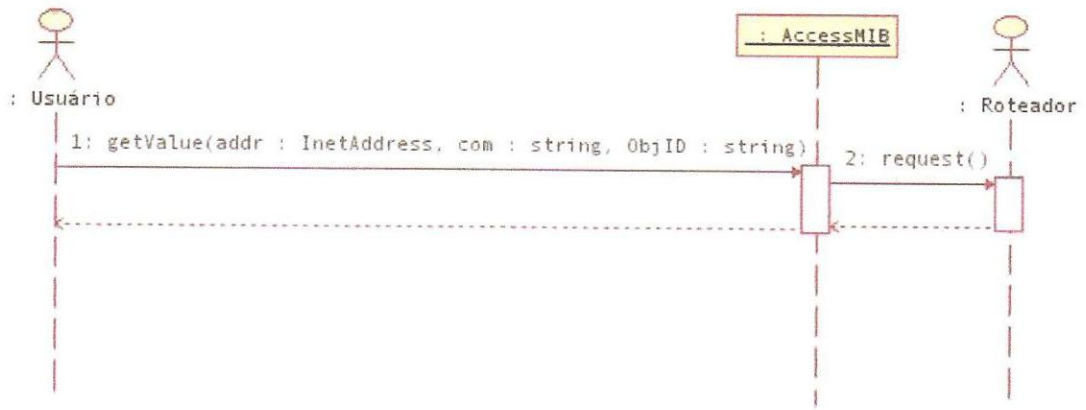

Figura 5.9: Acesso à MIB PIM de roteadores Cisco

[1] Roteador central da árvore de distribuição PIM-SM.

Mesmo que o usuário do sistema não esteja em uma rede habilitada para Multicast IP, não podendo, portanto, usar plenamente os recursos do ambiente, ele pode gerar tráfego multicast em seu segmento para realizar eventuais testes, conforme mostra o diagrama da figura 5.10. Uma boa utilidade para este serviço é verificar o comportamento do roteador que lhe serve, frente ao tráfego multicast. A carga útil destes pacotes gerados no segmento do usuário é mínima e desprezível, capturada de um campo de texto a ser preenchido pelo usuário.

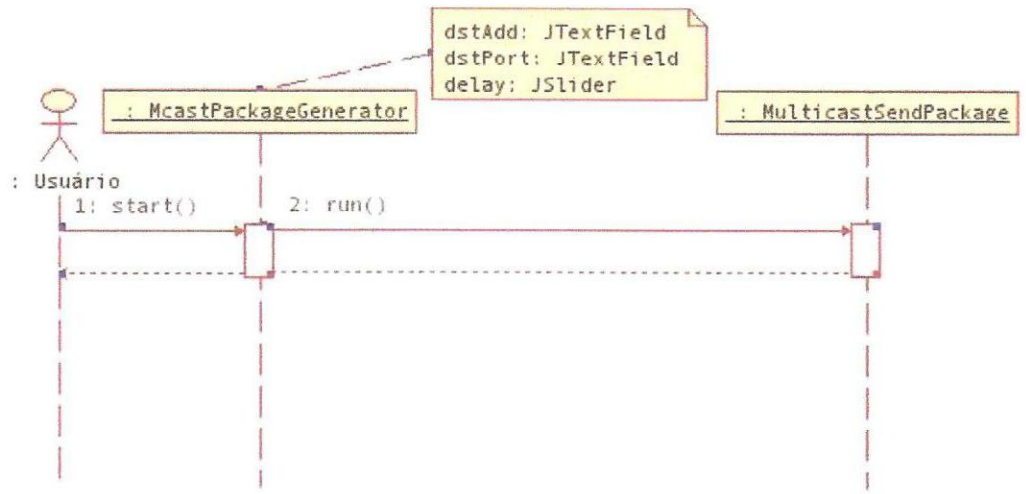

Figura 5.10: Gerando tráfego no segmento do cliente

Como mostrado na figura 5.11, a partir das informações dos pacotes SDP armazenadas na base de dados, o ambiente sob consulta pode calcular dados estatísticos e apresentá-los ao usuário. Alguns destes dados são: número de sessões anunciadas, sessões em transmissão, número de sessões anunciadas por país, número de sessões anunciadas por mídia, ferramentas mais usadas nas sessões, dentre outros.

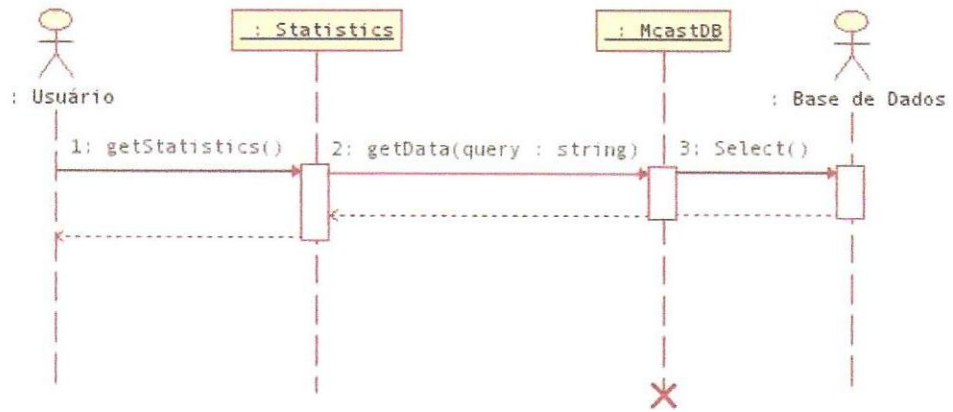

Figura 5.11: Consultar estatísticas do ambiente

Caso o usuário deseje entrar em uma sessão de *chat*, basta que ele selecione uma sala e insira um apelido. A partir daí, de forma transparente ao usuário, suas mensagens serão enviadas em multicast, por meio da classe *MulticastSendPackage*, a todos os usuários que estiverem na mesma sala, conforme mostra a figura 5.12. O ambiente oferece a opção do usuário conversar em privado com outro usuário de sua sala, para isto basta que ele insira o endereço IP do *host* do usuário com quem deseja conversar em particular. Em conversas privadas, as mensagens trocadas entre os usuários são transmitidas em *unicast* pelo uso das classes *SendPackage* e *ReceiveUnicastMessages*.

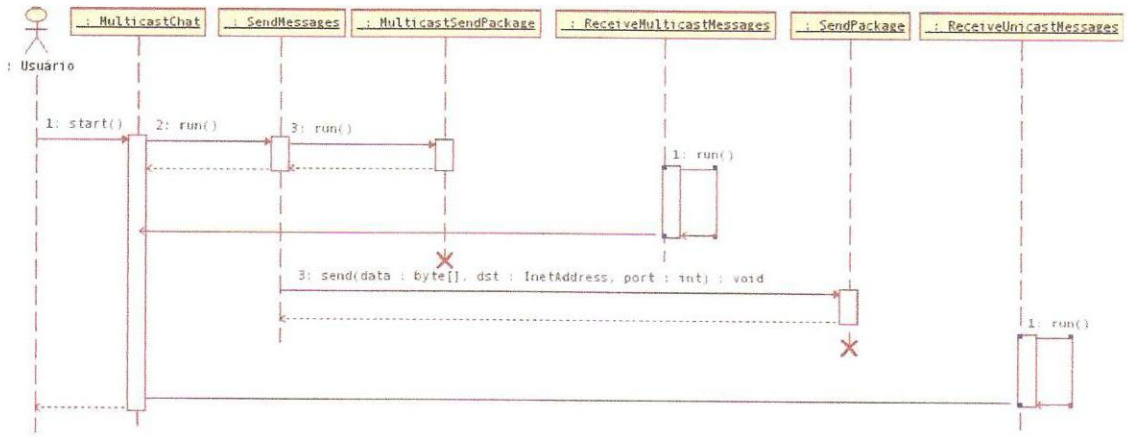

Figura 5.12: Entrando em uma sessão de *chat*

Finalmente, a figura 5.13 apresenta o diagrama de seqüência da opção de verificar se o ambiente de rede no qual a estação do usuário se encontra, disponibiliza ou não recursos para tráfego multicast. Esse método consiste de um *applet*, enviado ao cliente, que tenta invocar tráfego de um grupo que nunca se extingue no Mbone. Caso o *applet* perceba a chegada do fluxo do grupo no segmento, significa que o mesmo está ligado à MBone.

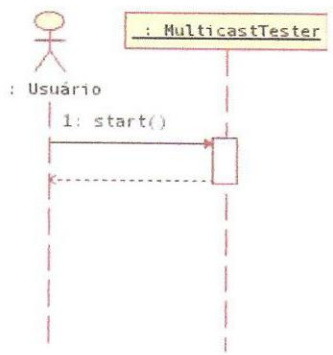

Figura 5.13: Testando a conectividade multicast do usuário

5.4 Estrutura do banco de dados

O ambiente faz uso do banco de dados apenas para guardar as informações concernentes aos anúncios de sessão oriundos de pacotes SDP. Conforme descrito na seção 2.4, pacotes SDP possuem diversos campos opcionais, sendo alguns deles possíveis de aparecer em mais de um momento no pacote. Esta flexibilidade estrutural do protocolo, faz com que sua especificação e codificação torne-se complexa.

Devido a maioria dos campos do pacote SDP serem de existência facultativa e, ainda, sem número determinado de ocorrências, foi preciso criar tabelas específicas para estes campos opcionais. A figura 5.14 ilustra as tabelas do banco de dados usado pelo ambiente para armazenar todas as informações contidas nos pacotes SDP captados pelo ambiente. O grande número de tabelas se fez necessário para suprir com rigorosidade as especificações do protocolo SDP [Handley e Jacobson 1998]. Os nomes das tabelas e seus campos, e até mesmo a estrutura do banco, condizem com os nomes sugeridos pela RFC 2327.

Para que não existam inconsistências no banco, a classe *SdpListener*, responsável em captar pacotes SDP, trata cada pacote para verificar sua conformidade com os padrões definidos na RFC 2327. Os pacotes que não respeitam os padrões definidos são descartados, não armazenados, portanto, no banco de dados. No entanto, um pequeno grupo de pacotes que não respeita os padrões pode ser aceito. A seção 6.2.3 aborda os critérios observados pelo ambiente para que pacotes SDP sejam ou não descartados.

Além deste banco de dados, ainda há um outro, com a finalidade apenas de revelar o país ao qual pertence uma faixa de endereços IP.

5.5 Descrição de operação

Todo o ambiente é desenvolvido em Java e, embora agregue tecnologias avançadas da linguagem, dá ao usuário um aspecto amigável e intuitivo. Isso faz com que o ambiente possa ser útil tanto a usuários leigos quanto a administradores de rede. Um bom exemplo é quando o usuário solicita ver as sessões anunciadas, o ambiente colhe da base de dados o título e o tipo (*broadcast, meeting* ou *test*) de cada sessão, coloca a bandeira do país de origem do anúncio e apresenta a lista completa dos anúncios ao usuário. A resposta que o usuário terá do ambiente é mostrada na figura 5.15. Cada nome de sessão é um *link*, que uma vez acionado, faz com que o conteúdo da sessão selecionada seja apresentado ao usuário.

announcement

Field	Type
protoversion	int(2)
username	varchar(64)
sessid	varchar(16)
sessversion	varchar(16)
nettype	char(2)
addrtype	varchar(4)
addr	varchar(16)
uri	varchar(128)
sessionname	varchar(128)

media

Field	Type
sessid	varchar(16)
medid	int(8)
mediatype	varchar(64)
port	tinyint(4)
number	tinyint(4)
transport	varchar(16)
fmt	tinyint(4)

connection

Field	Type
sessid	varchar(16)
connid	int(8)
nettype	char(2)
addrtype	varchar(4)
address	varchar(16)
ttl	tinyint(4)
number	tinyint(4)

repeat

Field	Type
sessid	varchar(16)
repeid	int(8)
interval	varchar(16)
duration	varchar(16)
offset1	varchar(16)
offset2	varchar(16)
offset3	varchar(16)

zone

Field	Type
sessid	varchar(16)
zoneid	int(8)
ajustment	varchar(16)
offset	varchar(16)

phone

Field	Type
sessid	varchar(16)
pid	int(8)
phone	varchar(64)

time

Field	Type
sessid	varchar(16)
timeid	int(8)
start	varchar(16)
stop	varchar(16)

information

Field	Type
sessid	varchar(16)
infoid	int(8)
information	longtext

encryption

Field	Type
sessid	varchar(16)
encrid	int(8)
method	varchar(64)
key	varchar(128)

attributes

Field	Type
sessid	varchar(16)
attrid	int(8)
attribute	varchar(64)
value	varchar(64)

bandwidth

Field	Type
sessid	varchar(16)
bwid	int(8)
bwtype	char(2)
bandwidth	varchar(16)

email

Field	Type
sessid	varchar(16)
eid	int(8)
email	varchar(64)

Figura 5.14: Tabelas do Banco de Dados do Ambiente

Figura 5.15: Anúncios listados pelo ambiente

A principal função do ambiente integrado é livrar o usuário da instalação e configuração de diversas ferramentas para o uso do tráfego multicast, portanto, o ambiente enquadra-se na categoria de uma ferramenta para engenharia de redes. Para um usuário comum, a função do ambiente é identificar sessões do MBbone e, caso solicitado, fazer com que fluxos multicast cheguem até ele, independente da plataforma que esteja sendo usada. O ambiente ainda proporciona a este usuário condições de participar de *chats*, sem que o mesmo perceba que eles funcionam baseados em transmissão multicast. Para administradores de redes o ambiente agrega diversas ferramentas de controle, teste e manipulação do tráfego multicast. Entretanto, não pertence ao escopo do ambiente interpretar as codificações de vídeo ou áudio usadas nas transmissões multicast. A função básica do ambiente é fazer com que o usuário veja o anúncio de uma sessão e possa invocar seu fluxo. A capacidade de interpretar este fluxo fica a cargo de um *plugin* de videoconferência, que o próprio ambiente configura e envia ao usuário, para que este apresente o conteúdo das sessões no terminal do cliente. Portanto, como ilustrado na figura 5.2, os únicos elementos que o usuário precisa instalar em seu computador, para que possa usufruir das funcionalidades completas do ambiente, são: um navegador, a máquina virtual Java e um *plugin* de videoconferência.

O ambiente é preparado para funcionar independente da escolha do *plugin*; basta ape-

nas que o *plugin* escolhido seja apto a receber fluxo multicast e, principalmente, respeite os padrões definidos pelas RFCs. Para usuários que usam plataformas Windows, os dois *plugins* mais indicados são o do QuickTime e o do Realplayer, enquanto que para plataformas Linux, as melhores opções são o Kaffeine, QuickTime4Linux, MPEG4IP e Videolan. Obviamente, estes *plugins* possuem características diferentes e, possivelmente, nem todos sejam aptos a ver todos os tipos de mídias existentes. O *plugin* do QuickTime é o mais adequado para uso em conformidade com o sistema integrado, isto porque é o mais fiel ao padrões. Já o *plugin* do Videolan possui limitações e não respeita fielmente os padrões. A figura 5.16 apresenta a visualização do conteúdo de um grupo solicitado pelo usuário.

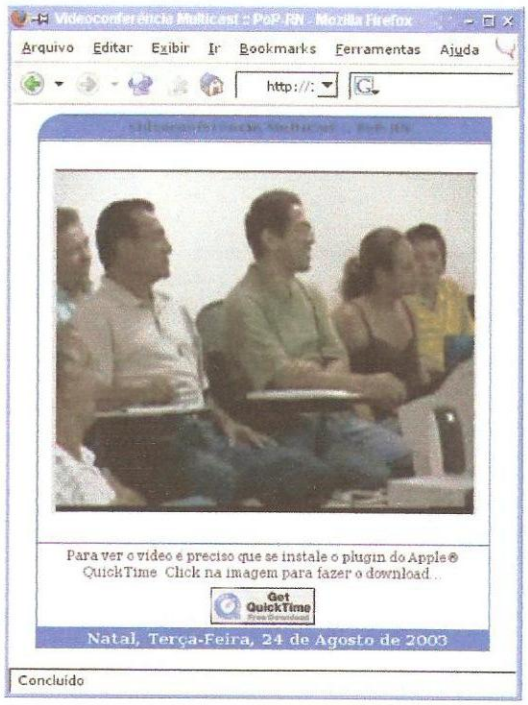

Figura 5.16: Conteúdo do grupo selecionado

O ambiente respeita fielmente os padrões definidos pelas RFC relacionadas à multicast, quanto a sua funcionalidade, nome dos atributos e métodos e, ainda, estrutura do banco. Isto se faz necessário, uma vez que o ambiente se propõe a agregar diversas funcionalidades, até então existentes em ferramentas distintas e de funcionamento *standalone*, ou seja, não executadas no navegador. Uma das funções que mais exige fidelidade aos padrões é a criação de anúncios de sessões multicast. O SDP possui diversas opções de configuração, muitos campos opcionais e, até mesmo, campos criados pelo próprio usuário. Isso dificulta em muito a codificação de ferramentas para gerar e interpretar pacotes SDP.

Para que o pacote SDP criado pelo usuário do ambiente não fira os padrões definidos

pela RFC 2327, o ambiente faz com que o usuário que deseja criar um anúncio de sessão, passe por uma seqüência de formulários que receberão os campos que irão compor o pacote SDP. Como mostrado na figura 5.17, os formulários possuem regras de validação, para evitar que o usuário leigo desobedeça os padrões, como por exemplo, informar um endereço de grupo que não pertença à classe D do espaço de endereçamento IPv4.

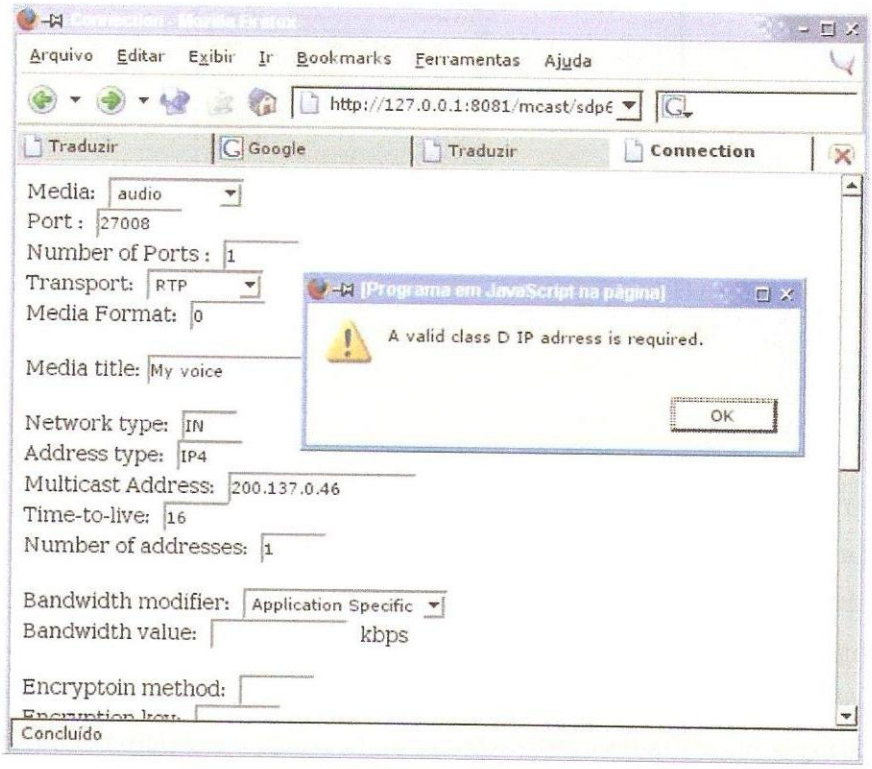

Figura 5.17: Formulário com campos do SDP

Os formulários são amigáveis: assim basta que o usuário responda a perguntas simples e o ambiente se encarrega de interpretá-las e colocá-las nos padrões da RFC. Como mostra a figura 5.18, quando o usuário responde a toda seqüência de formulários, o ambiente envia um *applet* ao *host* do usuário, para que o anúncio SDP seja feito a partir de sua máquina.

Outra função do ambiente é proporcionar ao usuário a capacidade de manipular tráfego multicast em sua rede. Portanto, a maior parte das funções do ambiente precisa ser executada na estação do usuário. *Applets* são programas escritos em Java que podem ser incluídos em páginas HTML. Quando um servidor recebe a requisição a uma página contendo uma *applet*, ele envia as classes Java relativas ao *applet* aos clientes, para serem executados pela máquina virtual Java nos navegadores.

Por serem executados na máquina do cliente, *applets* são sujeitos a rígidas regras de

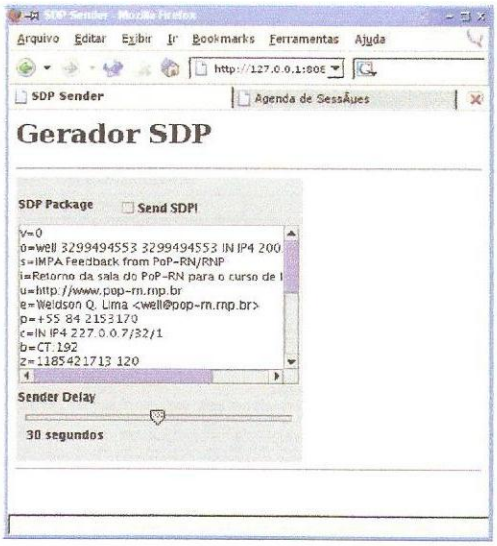

Figura 5.18: *Applet* emissor de pacotes SDP

segurança, não podendo ler ou escrever arquivos no cliente ou fazer conexões de rede, exceto com o servidor de onde são oriundos [Community 2004]. Por estas restrições de segurança, a documentação da classe *Multicastsocket* alerta que esta classe não pode ser usada com *applets* [Documentation 2003], sem que estes tenham algum certificado de segurança. Para que o ambiente forneça *applets* capazes de manipular tráfego multicast, é preciso que o usuário responda a uma assinatura de segurança, afirmando que confia no servidor emissor daquele *applet*. *Applets* com esta assinatura de segurança, *Signed Applets*, adquirem as mesmas permissões de segurança de arquivos locais do cliente. O pedido de confirmação oferecido ao usuário é mostrado na figura 5.19.

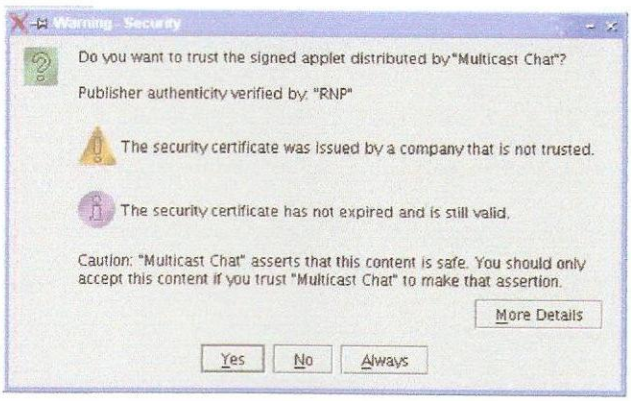

Figura 5.19: Certificado de confiabilidade de *Applets* assinados

Nem todas as funções do ambiente precisam ser executadas na estação do cliente.

Como mostra a figura 5.20, um calendário anual de transmissão de sessões pode ser formado no servidor e enviado ao usuário através de páginas HTML.

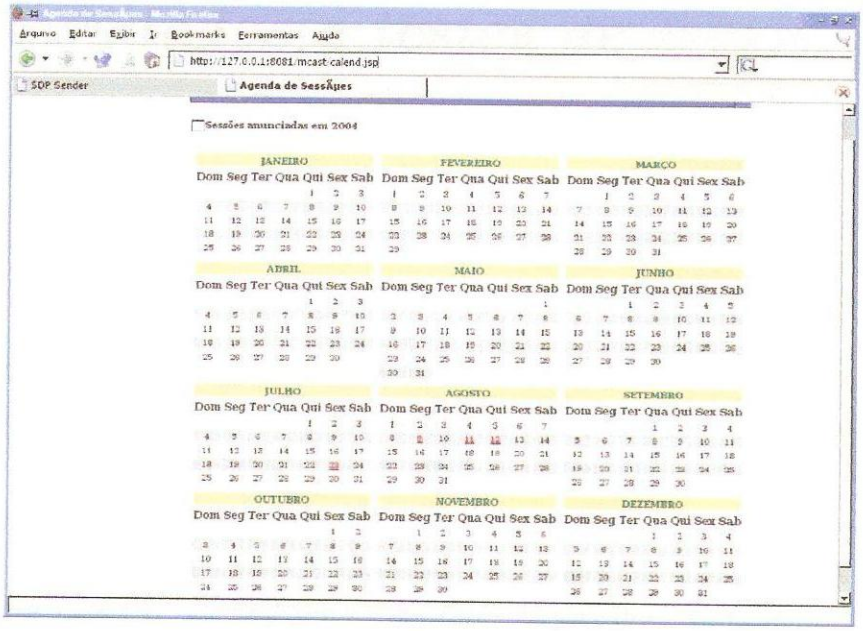

Figura 5.20: Calendário do anúncios agendados

6 *Resultados e testes de desempenho*

Como mencionado na seção 5.2, todas as tecnologias usadas no ambiente são de código livre, eliminando, portanto, custos com tecnologias proprietárias. A escolha destas tecnologias traz diversos benefícios como a alta capacidade de configuração e adequação às necessidades do ambiente no servidor. No entanto, o fator desempenho precisa ser mensurado para que se comprove a qualidade da ferramenta quanto ao baixo custo de processamento da CPU (*Central Processing Unit*) do servidor. Nesta seção são descritos sucintamente alguns testes e resultados obtidos com a execução do ambiente em um contexto real, portanto, não simulado.

6.1 Cenário de testes

Grande parte das funções do ambiente é implementada em *applets*, o que significa que essas funções são executadas na máquina do cliente. No entanto, algumas funcionalidades como o servidor de captura de pacotes SDP e os acessos ao banco de dados são aplicações executadas na estação que hospeda o ambiente integrado.

A tabela 6.1 apresenta as características de *hardware* e *software* de uma máquina servidor onde foi hospedado o ambiente integrado para fins de teste.

Embora não ilustre os requisitos mínimos de *hardware* para o funcionamento do servidor do ambiente, as características listadas na tabela 6.1 são úteis para coletar dados quanto ao uso da CPU, da placa de rede e da memória. A partir destes dados coletados pode-se inferir quanto à viabilidade do uso do ambiente integrado.

A figura 6.1 ilustra o cenário e tecnologias usadas para testes de funcionamento em situação de operação real do ambiente integrado. Os fornecedores das tecnologias escolhidas para o servidor são opcionais. Embora se tenha justificado, no capítulo anterior,

Hardware	Software
AMD Duron (tm) processor, 1.4 MHz, cache size 64 KB	Linux 2.4.22-xfs i686 Debian GNU/Linux
ASUS A7V8X-MX motherboard, VIA KM400 Chipset	Tomcat Web Server v3.3a Final
512 MB DDR SDRAM PC2100 266 MHz	J2SDK 1.4.2_04-b05
Western Digital 20 GB 5200 RPM	MySQL Ver 12.22 Distrib 4.0.18
VIA Technologies T6103 10/100 Mbps Ethernet PHY	MRTG 2.10.13

Tabela 6.1: Características de Hardware e Software de um servidor de teste

as escolhas de *software* listadas na tabela 6.1 para desenvolvimento desse ambiente, esses programas podem ser substituídos por soluções equivalentes ou versões similares.

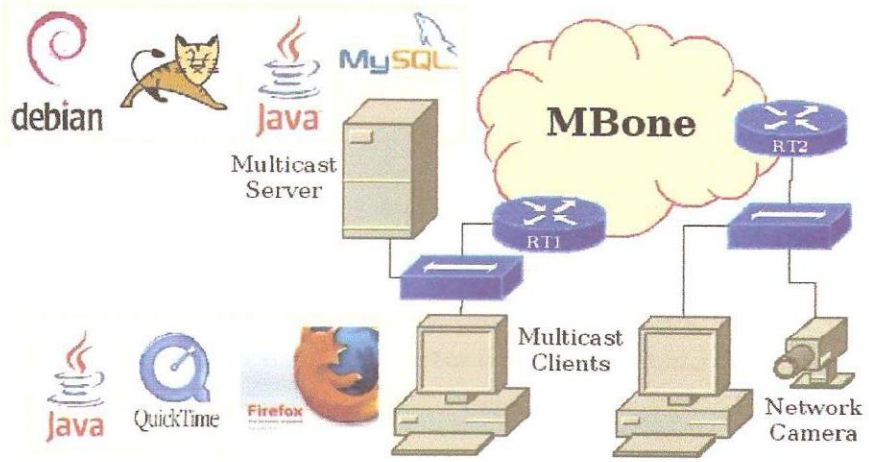

Figura 6.1: Cenário e tecnologias usadas para testes do ambiente integrado

6.2 Resultados obtidos

6.2.1 Desempenho do Servidor

Com o auxilio do MRTG (*Multi Router Traffic Grapher*) [Oetiker e Rand 2004], através do uso do *script* apresentado no Anexo A, foi obtido um gráfico que indica medidas de desempenho do servidor que hospeda o ambiente. Esse gráfico é apresentado na figura 6.2, onde se mostra em cor verde o uso da CPU (linha preenchida) e em cor azul o uso da memória (linha vazada). Embora o gráfico tenha sido contaminado com o processa-

mento do ambiente gráfico KDE 3.2.2[1], os resultados mostram que o servidor teve um pico máximo de uso de 36%, mas que a média é de apenas 10% de uso dos recursos da CPU. O gráfico da figura 6.2 ainda mostra que a memória do servidor também possui baixas medidas de uso, com pico máximo no período de amostragem de 45%, com média de 22% de uso.

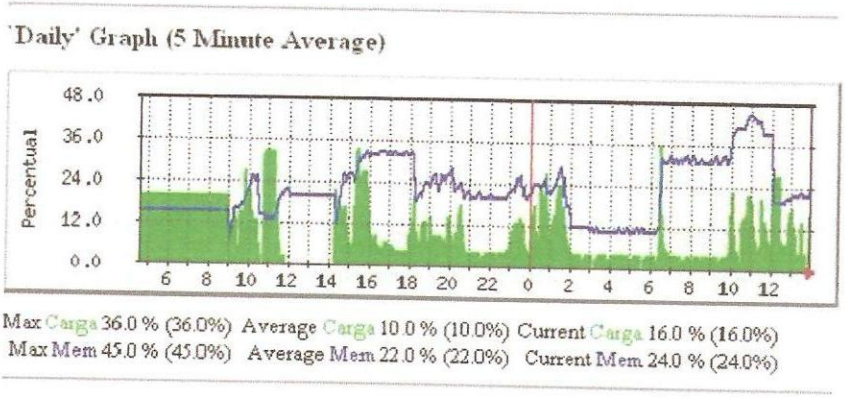

Figura 6.2: Gráfico de uso da CPU e memória

Para uma máquina que não possui demasiados recursos de *hardware*, uma média de 10% de uso da CPU e média de 22% de uso da memória mostram que o ambiente integrado provoca um custo perfeitamente aceitável de processamento. Dentro destes 10% de CPU e 22% de memória, ainda estão diluídos os custos de processamento dos servidores MySQL e Tomcat, que residem na mesma estação.

A figura 6.3 mostra algumas linhas da resposta do sistema ao comando "`$ps -aux`" e, através delas, podemos comprovar que o coração do ambiente, a classe *SdpListener*, é responsável por um custo médio de processamento de 0.0%, ou seja, menor que 0,04%. Como descrito na seção 5.3.2, esta classe é o núcleo funcional do ambiente, responsável por capturar os pacotes SDP, tratá-los e alimentar o banco com seus dados. A classe *SdpListener* age como um servidor a espera de pacotes SDP e, para cada pacote SDP recebido por ela, são feitas em média 20 acessos (consultas e inserções) ao banco de dados, variando conforme o número de campos opcionais existentes no pacote. Mesmo com tantas consultas por pacote, a figura 6.3 revela que seu processamento provoca praticamente um custo nulo de uso da CPU do servidor.

A figura 6.4 ilustra a resposta do servidor ao comando "`$mysqladmin -u root -p status`", que exibe algumas informações concernentes ao estado do servidor de banco de

[1]Durante o período de amostragem o ambiente gráfico KDE 3.2.2 estava em execução apenas para exibir os gráficos de desempenho gerados pelo MRTG.

```
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
mysql      433  0.0  1.0  64820  5484 ?        S    19:33   20:13 /usr/sbin/mysqld
www-data  4710  0.5  3.4 234056 17280 ttyp1    S    22:51   20:41 /usr/local/j2sdk1.4.2_04/bin/java
well      4721  0.0  1.6 210600  8296 ttyp1    S    22:51   19:03 java SdpListener
```

Figura 6.3: Parte da resposta ao comando "`$ps -aux`"

dados. Percebe-se que, no momento da execução do comando, havia apenas um cliente conectado ao banco, uma instância da classe *McastDB*, que estava sendo usada para consultar e armazenar os dados dos pacotes SDP capturados pelo servidor. A figura 6.4 revela que a média de consultas por segundo, apenas relativas aos pacotes SDP, é de 28.317. A esta velocidade média, no momento da amostragem, o servidor já havia respondido a 32933 consultas e não considerou nenhuma delas lenta, conforme explicitado no campo *Slow queries*.

```
Uptime: 1163  Threads: 9  Questions: 32933  Slow queries: 0  Opens: 24
Flush tables: 1  Open tables: 18  Queries per second avg: 28.317
```

Figura 6.4: Resposta do servidor ao comando "`$mysqladmin -u root -p status`"

A alta velocidade de resposta do servidor MySQL, ilustrada na figura 6.4, e o baixo custo de processamento, ilustrado na figura 6.3 comprovam a viabilide do uso desta tecnologia de banco de dados para o ambiente integrado. Por usar aplicações em "tempo real", o fator velocidade é mais relevante do que a capacidade de recursos do banco, o que comprova a escolha do MySQL a opção mais adequada ao ambiente.

A figura 6.3 revela ainda que a classe *SdpListener* ocupa em média 1.6% da memória, o que, em termos absolutos para o servidor em questão, significa algo em torno de 7.8 Mb. Dentre os elementos que envolvem o funcionamento do ambiente, o que exige maior desempenho é o contentor de *servlet* Tomcat, com custo somado ao da JVM de 3.6% de uso da memória e 0.5% de uso da CPU.

A RFC 2327, que descreve o protocolo SDP, não sugere um tempo de retransmissão do pacote, ficando portanto a cargo do emissor a escolha do intervalo de tempo decorrido entre dois anúncios de sessão. Pacotes SDP transeuntes no Mbone normalmente tem em torno de 4 Kb de tamanho e o montante possui um custo médio de 16 Kbps na banda passante. A figura 6.5 mostra um gráfico de uso da interface de rede do servidor, usado para testes de desempenho do ambiente integrado. O gráfico revela que a média de entrada na interface é de 18 kbps, e que o fluxo de saída possui pico máximo de 858.4

kbps. Estes valores de picos discrepantes são conseqüência do servidor também fornecer páginas HTML e JSP, assumindo uma característica de transmissão em rajada.

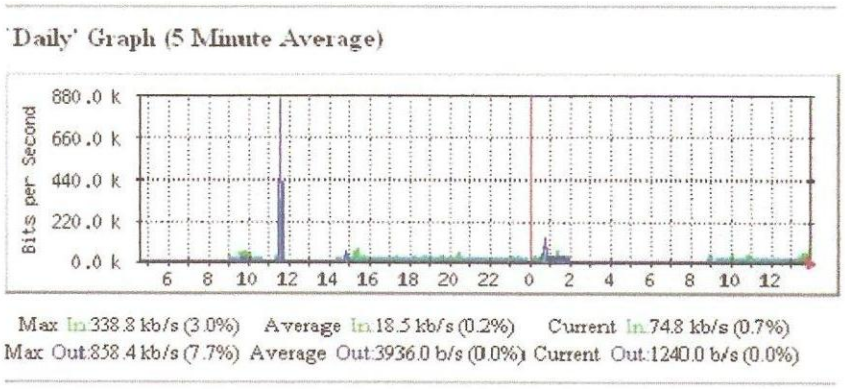

Figura 6.5: Gráfico de uso da interface de rede

A maior preocupação do uso de *applets* em aplicações para a Web é que estes são enviados, com todas as classes que o compõem, para que sejam executados na estação do usuário. As funcionalidades construídas com *applets* no contexto do ambiente integrado estão localizadas em classes que possuem tamanhos pequenos, evitando provocar demasiadamente grande tempo de transferência do servidor ao usuário que solicitou. A figura 6.6 mostra a resposta do sistema ao comando "`$du -h a*.jar`", listando o tamanho em Kb dos *applets*, já assinados com certificado de segurança, usados pelo servidor do ambiente.

```
root@tenda:/var/lib/tomcat/webapps/mcast# du -h a*.jar
16K     aMChat.jar
8,0K    aPkgGen.jar
8,0K    aSdp.jar
4,0K    aMtest.jar
4,0K    aInvTra.jar
```

Figura 6.6: Tamanho dos *applets* do ambiente

Portanto, amparado pelos resultados dos testes de desempenho obtidos durante a execução do ambiente integrado, instalado em uma máquina com recursos de *hardware* medianos, conforme descrito na tabela 6.1, comprova-se a viabilidade do uso desta ferramenta. O servidor do ambiente integrado exige valores ínfimos em termos de desempenho da CPU do servidor para sua operação e, mesmo as funções que são executadas na estação do usuário, estão construídas em arquivos de tamanho pequeno, que exigem pouco tempo para sua transferência.

O ambiente integrado, além de livrar o usuário da instalação e configuração de diversas ferramentas, também provoca uma grande otimização de desempenho de sua estação, visto que ao invez de usar muitas ferramentas multicast, faz uso apenas do navegador e do *plugin* de videoconferência.

6.2.2 Dados coletados

A primeira versão da classe *SdpListener*, responsável por capturar os pacotes SDP da rede, obedecia fielmente aos padrões definidos pela RFC 2327. Portanto, antes de repassar os dados colhidos dos pacotes SDP para que a classe *McastDB* alimentasse o banco, cada pacote era submetido a rigorosos testes de conformidade com os padrões definidos. A finalidade destes testes era impedir que pacotes SDP mal formados causassem inconsistências no banco de dados. Alguns testes realizados incluíam: presença do campo de versão do protocolo, nome e identificador válidos da sessão, horário de início e término da sessão não nulos, dentre outros; que conforme citado pela RFC [Handley e Jacobson 1998]. No entanto, esta primeira versão da classe *SdpListener*, ao aplicar os filtros descritos na RFC, capturava em média apenas 16% dos pacotes SDP advindo do MBone. Este baixo aproveitamento pode ser explicado pela existência de uma grande quantidade de pacotes SDP mal formados.

```
o=VideoLAN 3247692199 3247895918 IN IP4 VideoLAN
s=Diffusion RAP (8Mbits/s,MPEG2) - Le Réseau Académique Parisien
u=VideoLAN
t=0 0
m=video 1234 udp 33
c=IN IP4 233.9.118.1/15
a=type:test
```

Figura 6.7: Pacote SDP mal formado

A figura 6.7 ilustra um anúncio criado pelo VideoLAN que se mostra bastante fora do padrão, fazendo com que apenas outros usuários, que também usem o VideoLAN, possam interpretá-los corretamente. Nessa figura percebe-se que: o pacote não contém a linha obrigatória da versão "v="; ao fim da primeira linha encontra-se a *string* "VideoLAN" como endereço IP do *host* do anunciante; o pacote contém informações relativas à taxa de transferência no campo "s=", quando a mesma deveria pertencer ao campo "b="; e ainda

traz os horários de início e fim de sessão zerados.

Embora a classe *SdpListener* aceite pacotes mal formados, quando o usuário deseja criar um anúncio de sessão multicast, usando o ambiente integrado, este cria pacotes SDP perfeitamente em conformidade com o padrão definido na RFC 2327. A figura 2.2, apresenta o conteúdo de um pacote SDP, criado pelo ambiente integrado, que respeita integralmente as exigências da RFC 2327. A prova do rigor com que a ferramenta cria pacotes SDP, em total conformidade com a norma, pode ser vista na figura 6.8, que apresenta a captura de um desses pacotes pelo *sniffer* Ethereal, onde os campos foram interpretados corretamentes pelo *sniffer*.

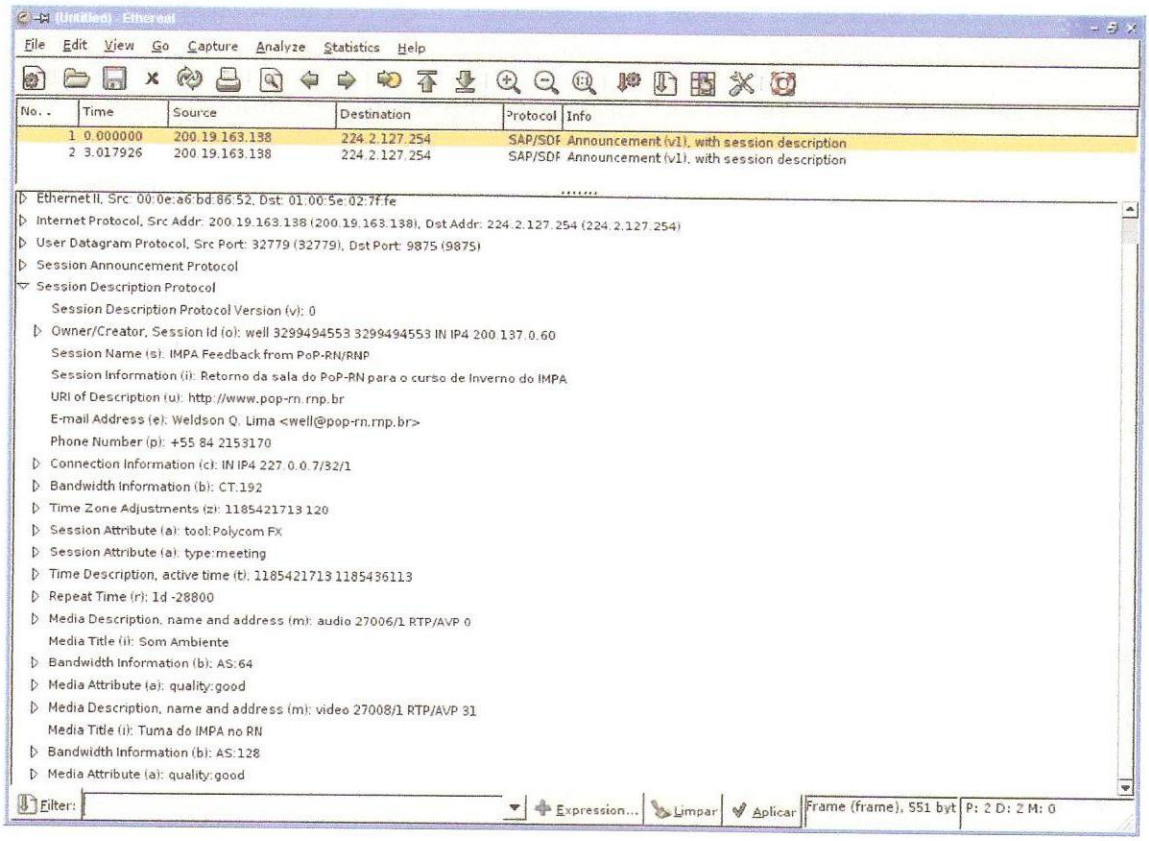

Figura 6.8: Pacote SDP capturado pelo *sniffer* Ethereal

6.2.3 Critérios de tolerância

Para que o ambiente integrado fosse capaz de capturar o máximo de pacotes SDP possíveis, a segunda versão da classe *SdpListener* foi construída, de forma a incluir alguns relaxamentos nos testes de conformidade com a RFC 2327, aceitando pacotes SDP mal formados, mas que não comprometem o bom funcionamento do ambiente. Com a

diminuição da rigorosidade dessa classe, a mesma passou a capturar em média 83% dos pacotes SDP, descartando apenas os que ferem completamente os padrões definidos. Os critérios adotados como tolerância a falta de conformidade dos pacotes SDP com as normas são:

`v=` A norma estabelece que os dois primeiros *bytes* do pacote SDP devem ser dos caracteres "*v*=". Caso o ambiente encontre outra informação nestes dois primeiros *bytes*, o mesmo aceita o pacote SDP mal formado e aplica uma reconstrução do mesmo, adicionando "`v=0`" ao seu início.

`o=` A ausência do nome do criador da sessão e, até mesmo, a ausência de seu endereço IP, caracteriza mal formação do pacote. No entanto, o ambiente aceita pacotes nesta condição porque considera que esta sessão possui autoria anônima. Este anonimato não compromete o funcionamento do ambiente, apenas provoca a ausência da bandeira do país de origem do criador da sessão.

`s=` De acordo com a norma, o nome da sessão é um campo obrigatório para pacotes SDP. No entanto, a ambiente aceita pacotes que firam as normas relativas a este campo por este não ser um campo que afete a operação do ambiente. Com dano ao ambiente isto provoca nome de sessão ausente na interface de lista de sessões anunciadas.

`t=` Em uma amostra de 1133 pacotes SDP capturados, menos de 1% dos pacotes não possuía estes campos nulos. Embora esta não seja uma infração prevista na norma descrita na RFC 2327, isto caracteriza uma inconformidade funcional. Diante disto, o ambiente aceita pacotes que possuam valores nulos como tempo de início e fim da sessão.

Os demais campos do pacote SDP são opcionais, exceto o campo "`m=`", que descreve uma mídia utilizada na transmissão da sessão. No entanto, mesmo estes campos opcionais podem criar inconsistências nos pacotes SDP, tornando-os mal formados. São algumas das situações que caracterizam pacotes SDP mal formados:

- Ausência do campo obrigatório "o="
- Campos fora da ordem se seqüência
- Ausência do campo "c=" ou "m="

- Segundo *byte* de uma linha diferente de "="

Ainda são muitos os fatores que inviabilizam a leitura de um pacote SDP, por estar este em muitos fatores diferente da norma. Exceto pelas tolerâncias descritas anteriormente, qualquer pacote SDP que fira com a norma descrita na RFC 2327 [Handley e Jacobson 1998] é rejeitado pelo ambiente.

A figura 6.9 apresenta um pacote SDP mal formado capturado pelo *sniffer* Ethereal. Na parte inferior dessa figura o Ethereal sinaliza que o pacote possui mal formação, isso devido a irregularidades nos bytes de terminação do pacote.

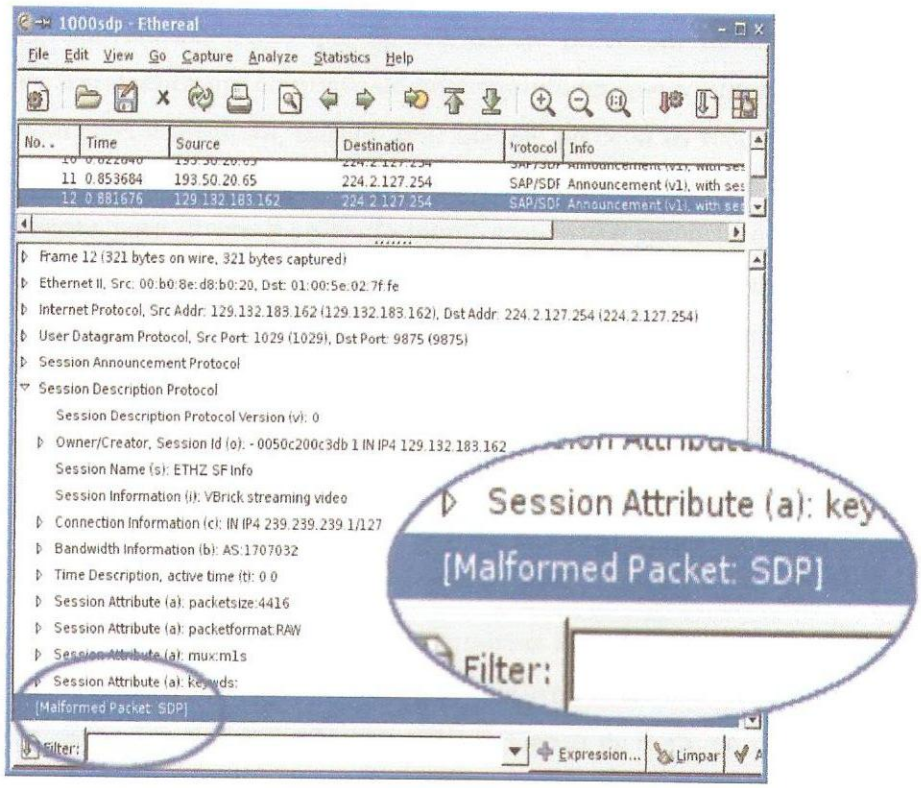

Figura 6.9: Pacote SDP mal formado capturado pelo *sniffer* Ethereal

7 Conclusão

Foi proposto o desenvolvimento de um ambiente, que possa ser útil como ferramenta para usuários de diferentes perfis e que desejam fazer uso dos recursos do tráfego multicast no MBone, um usuário leigo, que apenas deseje participar de sessões de multimídia, encontra na ferramenta um conjunto de funcionalidades que o auxiliam em seu propósito, de forma amigável e intuitiva, mesmo sem que ele perceba os recursos de Engenharia de Rede que o ambiente disponibiliza. Um usuário administrador de rede, por sua vez, pode usar ferramenta para diversas ações de monitoramento e gerência de tráfego multicast, a exemplo do teste de conectividade, acesso a MIB de roteadores, dentre outros. Ainda um desenvolvedor de aplicações multimídia poderá se valer da ferramenta, para fazer uso de suas APIs em aplicações em que esteja desenvolvendo.

Para selecionar as tecnologias usadas na construção do ambiente, foram estipulados três critérios básicos: ser tecnologia de código livre, fazer uso dos padrões definidos nas RFCs e proporcionar bons índices de desempenho. Diante destes fatores, escolheu-se a linguagem de programação Java, com o uso do MySQL como base de dados e o Linux Debian como sistema operacional. Embora outras soluções pudessem ser adotadas, este conjunto de tecnologias se mostrou altamente estável, consumindo pouquíssimos recursos do servidor. Em operação, o ambiente utilizou menos de 0,5% da capacidade de processamento do servidor, com baixo nível de uso da memória instalada.

O desenvolvimento desta ferramenta preenche uma lacuna, no que se refere a ferramentas para tráfego multicast. Isto porque o ambiente agrega diversas funcionalidades que existiam até então isoladas em outras ferramentas, normalmente proprietárias e de capacidade de recursos limitada. Por seu funcionamento no navegador, a acessibilidade também é um diferencial do ambiente, livrando o usuário da necessidade de instalação de diversas ferramentas em sua estação.

Portanto, respaldado pelos resultados obtidos durante a fase de testes do ambiente em situação real de funcionamento, e aliado aos altos índices de desempenho medidos,

pode-se constatar a viabilidade do uso desta ferramenta como suporte à manipulação do tráfego multicast.

O desenvolvimento desse ambiente foi divulgado através de publicação na revista internacional *WSEAS Transactions on Information Science and Applications*, como também no congresso 4th WSEAS Multimedia, Internet and Video Technologies (*ICOMIV 2004 Izmir - Turquia*), em setembro do corrente ano [Lima e Fialho 2004].

Ainda é preciso uma investigação mais criteriosa quanto ao desempenho do servidor do ambiente integrado em uma situação de uso real, com muitos usuários fazendo uso de seus recursos simultaneamente, para se determinar a escalabilidade da solução implementada. Além disso, propor uma representação dos campos de um pacote SDP em um formato padrão XML, objetivando facilitar a integração do ambiente com outras aplicações de semelhante natureza, são temas para trabalhos futuros.

ANEXO A

Script utilizado para fornecer ao MRTG medidas de uso da CPU e da memória do servidor em que o ambiente está hospedado.

```
#!/bin/sh
unset LANG
mem=$(/usr/bin/free|grep ^-)
load=$(cat /proc/loadavg)
/usr/bin/awk -v load="$load" -v mem="$mem" '
BEGIN{
        split(load,loadstats)
        print int(100*loadstats[2])
        split(mem,memstats);
        print int(100*memstats[3]/(memstats[3]+\memstats[4]));
}'
```

Referências bibliográficas

[AdventNet 2004]ADVENTNET. *AdventNet SNMP API 4 Documentation.* 2004. Disponível em: <http://www.multicasttech.com/>. Acesso em: 05 Set, 2004.

[Almeroth 2004]ALMEROTH, K. C. *Getting Started with Multicast.* 2004. Disponível em: <http://multicast.internet2.edu/wg-multicast-started.shtml>. Acesso em: 20 Dez, 2004.

[Anselmo 2001]ANSELMO, F. *Tudo que você queria saber sobre JDBC.* 1ª. ed. Santa Catarina: Visual Books, 2001. 200 p.

[Black 1999]BLACK, D. P. *Building Switched Networks.* 1st. ed. USA: Addison-Wesley, 1999. 298 p.

[Cisco Systems 2003]CISCO SYSTEMS. *PIM MIB Extension for IP Multicast.* [S.l.] , jan. 2003.

[Commer 1997]COMMER, D. E. *Interligação em redes TCP/IP, Volume 1.* 2ª. ed. Rio de Janeiro: Editora Campus, 1997. 672 p.

[Community 2004]COMMUNITY, S. D. N. *Code Samples and Apps Applets.* 2004. Disponível em: <http://java.sun.com/applets/>. Acesso em: 25 Ago, 2004.

[Deering 1989]DEERING, S. E. *RFC 1112: Host extensions for IP multicasting.* ago. 1989. Obsoletes RFC0988, RFC1054. See also STD0005. Updated by RFC2236. Status: STANDARD. Disponível em: <http://www.ietf.org/rfc/rfc1112.txt>.

[Deitel e Deitel 2003]DEITEL, H. M., DEITEL, P. J. *Java, Como Programar.* 4ª. ed. Rio Grande do Sul: Editora Bookman, 2003. 1386 p.

[Documentation 2003]DOCUMENTATION, J. . S. S. D. *MulticastSocket (Java 2 Platform SE v1.4.2).* 2003. Disponível em: <http://java.sun.com/j2se/1.4.2/docs/api/java/net/MulticastSocket.html>. Acesso em: 25 Ago, 2004.

[Estrin et al. 1998]ESTRIN, D. et al. *RFC 2362: Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification.* jun. 1998. Status: EXPERIMENTAL, Obsoletes: 2117. Disponível em: <http://www.ietf.org/rfc/rfc2362.txt>.

[Eyler 2001]EYLER, P. *Networking Linux - A Pratical Guide to TCP/IP.* 1st. ed. USA: New Riders, 2001. 404 p.

[Goyeneche 1998]GOYENECHE, J. M. *Multicast over TCP/IP HOWTO.* mar. 1998. Disponível em: <http://jungla.dit.upm.es/~jmseyas/linux/mcast.howto/multic.html>. Acesso em: 22 Jan, 2004.

[Handley e Jacobson 1998]HANDLEY, M., JACOBSON, V. *RFC 2327: SDP: Session Description Protocol.* abr. 1998. Status: PROPOSED STANDARD. Disponível em: <http://www.ietf.org/rfc/rfc2327.txt>.

[Handley, Perkins e Whelan 2000]HANDLEY, M., PERKINS, C., WHELAN, E. *RFC 2974: Session Announcement Protocol.* out. 2000. Disponível em: <http://www.ietf.org/rfc/rfc2974>.

[Hoff, Shaio e Starbuck 1996]HOFF, A., SHAIO, S., STARBUCK, O. *Ligado em Java - Criando Sites para Web com Applets Java.* 1ª. ed. São Paulo: Makron Books, 1996. 207 p.

[Horton 1997]HORTON, I. (Ed.). *Beginning Java.* 4th. ed. Canada: Wrox Press, 1997. 1039 p.

[Ingram 2002]INGRAM, S. *Java Socket Programming.* 2002. Disponível em: <http://hplasim2.univ-lyon1.fr/c.ray/bks/java/htm/ch26.htm>. Acesso em: 21 Jan, 2004.

[Kurniawan 2002]KURNIAWAN, B. *Java para Web com Servlets, JSP e EJB.* 4ª. ed. Rio de Janeiro: Editora Ciência Moderna, 2002. 807 p.

[Lima e Fialho 2004]LIMA, W. Q., FIALHO, S. V. An integrated environment to handle multicast traffic. *WSEAS Transactions on Information Science and Applications*, v. 1, Set 2004. ISBN 1790-0832.

[Malkin 1998]MALKIN, G. S. *RFC 2453: Routing Information Protoco Version 2.* nov. 1998. Disponível em: <http://www.ietf.org/rfc/rfc2453>.

[McCloghrie et al. 2000]MCCLOGHRIE, K. et al. *RFC 2934: Protocol Independent Multicast bib for IPv4.* out. 2000. Status: EXPERIMENTAL. Disponível em: <http://www.ietf.org/rfc/rfc2934.txt>.

[Miller 1999]MILLER, C. K. *Multicast Networking and Applications.* 1st. ed. USA: Addison-Wesley, 1999. 282 p.

[Moy 1994]MOY, J. *RFC 1584: Multicast Extensions to OSPF.* mar. 1994. Disponível em: <http://www.ietf.org/rfc/rfc1584>.

[Moy 1998]MOY, J. *RFC 2328: Open Shortest Path First Version 2.* abr. 1998. Disponível em: <http://www.ietf.org/rfc/rfc2328>.

[Nemeth et al. 2002]NEMETH, E. et al. *Manual do Administrador do Sistema Unix.* 3ª. ed. Rio Grande do Sul: Bookman, 2002. 895 p.

[Object Management Group 2003]OBJECT MANAGEMENT GROUP. *UML 2.0 Infrastucture Spetification.* [S.l.] , nov. 2003.

[Oetiker e Rand 2004]OETIKER, T., RAND, D. *MRTG Documentation Pack.* jul. 2004. Disponível em: <http://people.ee.ethz.ch/ oetiker/webtools/mrtg/>. Acesso em: 03 Set, 2004.

[Oser 2001]OSER, H. *JAVA multicast sockets.* jun. 2001. Disponível em: <http://www.oser.org/~oser/ds/node27.html>. Acesso em: 21 Jan, 2004.

[Osterloh 2002]OSTERLOH, H. *IP Routing Primer Plus.* 2nd. ed. USA: Sams Publishing, 2002. 490 p.

[Parkhurst 1999]PARKHURST, W. R. *Cisco Multicast Routing and Switching.* 1st. ed. USA: McGraw-Hill, 1999. 368 p.

[Pusateri e McBride 2004]PUSATERI, T., MCBRIDE, M. *Protocol Independent Multicast Working Group.* set. 2004. Disponível em: <http://www.ietf.org/html.charters/pim-charter.html>. Acesso em: 20 Dez, 2004.

[RealNetworks 2004]REALNETWORKS. *RealNetworks Security Updates and Incident Reports.* 2004. Disponível em: <http://service.real.com/help/faq/security/>. Acesso em: 28 Set, 2004.

[Rumbaugh, Jacobson e Boock 1999]RUMBAUGH, J., JACOBSON, I., BOOCK, G. *The Unified Modeling Language Reference Manual.* 2nd. ed. USA: Addison-Wesley, 1999. 550 p.

[Schoch 2002]SCHOCH, M. *Streaming Solutions for Affordable Internet Broadcasting.* 2002. Disponível em: <http://www.multicasttech.com/>. Acesso em: 05 Set, 2004.

[Suehring 2002]SUEHRING, S. *MySQL Bible.* 1st. ed. USA: Wiley Publishing, 2002. 687 p.

[Team 2004]TEAM, C. C. O. *Cisco IP/TV Software User Guides.* 2004. Disponível em: <http://www.cisco.com/en/US/products/sw/conntsw/ps1869/products_user_guide_list.html>. Acesso em: 25 Ago, 2004.

[TechNet 2004]TECHNET, M. *Vulnerability in Windows Media Services Could Allow a Denial of Service (832359).* mar. 2004. Disponível em: <http://www.microsoft.com/technet/security/bulletin/ms04-008.mspx>. Acesso em: 15 Ago, 2004.

[University College London 2001]UNIVERSITY COLLEGE LONDON. *Mbone Conferencing Applications.* jul. 2001. Disponível em: <http://www-mice.cs.ucl.ac.uk/multimedia/software/>. Acesso em: 21 Jan, 2004.

[Waitzman, Partridge e Deering 1988]WAITZMAN, D., PARTRIDGE, C., DEERING, S. *RFC 1075: Distance Vector Multicast Routing Protocol.* nov. 1988. Disponível em: <http://www.ietf.org/rfc/rfc1075>.

[Wakeman 2004]WAKEMAN, I. *Multicast Sockets.* jan. 2004. Disponível em: <http://www.cogs.susx.ac.uk/users/ianw/teach/dist-sys/multicast.html>. Acesso em: 21 Jan, 2004.

[Williamson 2000]WILLIAMSON, B. *Developing IP Multicast Networks, Volume I.* 1st. ed. USA: Cisco Press, 2000. 568 p.