SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP
Data de Depósito: 20.02.2001
Assinatura:

Transmissão de Vídeo Usando IPv6 e Multicasting em Redes de Alto Desempenho

Luciano Martins

Orientador: ***Prof. Dr. Edson dos Santos Moreira***

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências de Computação e Matemática Computacional.

USP – São Carlos
Fevereiro de 2001

A Comissão Julgadora:

Prof. Dr. Edson dos Santos Moreira _____

Profa. Dra. Regina Borges de Araujo _____

Prof. Dr. Carlos Antonio Ruggiero _____

Aos meus pais,
Maria do Carmo Conceição Martins
e em especial Gabrielino Martins (*in memorian*)

Aos meus irmãos,
Lilian Conceição Martins e
Marcelo Martins

À minha namorada e companheira,
Graciela Machado Leopoldino

Agradecimentos

Inicialmente, à Deus, que me permitiu estar na Terra entre meus familiares e amigos e passar por diversas experiências que me fazem crescer pessoalmente e profissionalmente...

À minha querida mãe Maria do Carmo, confidente e amiga de todas as horas, que me deu força e estímulo em todos os momentos destes 2 anos de mestrado e em toda a minha vida, que me dá conselhos, equilíbrio e esperança mesmo nos momentos mais difíceis e perturbadores. E em especial ao meu querido pai Gabrielino Martins (in memorian), que sempre me deu o exemplo de honestidade, caráter, garra, amizade, amor ao próximo e de conquista dos nossos sonhos, autor de uma frase que sempre trarei em minha lembrança "Quando uma porta se fecha, diversas se abrem". Dedico este trabalho especialmente a ele, que certamente continua nos aconselhando e nos amparando na esfera celestial.... Obrigado meu querido pai, por tudo, e por este momento que, sem os seus exemplos, certamente não estaria alcançando !!

Aos meus irmãos Lilian e Marcelo, que sempre compartilharam minhas horas difíceis e alegres, me aconselhando e me amparando, e pelos momentos que passamos juntos, e que certamente ainda passaremos...

À minha família, que sempre acreditou em minha capacidade e em meu sucesso e sempre me dão incentivo para seguir minha caminhada...

À minha querida namorada Graciela, companheira, confidente e conselheira,, por sua paciência, compreensão, apoio e pelo seu amor e carinho.

Ao meu orientador, Prof. Dr. Edson dos Santos Moreira, que sempre me auxiliou, dando idéias valiosas ao projeto e por todas as oportunidades proporcionadas.

Ao meu amigo Leonardo Antônio de Andrade, grande companheiro de guerra, que passou comigo horas a fio de trabalho no laboratório InterMídia, de testes e implementações. Valeu, Leo !!

Aos meus amigos Rudinei, Gustavo, João, Anja e a todos do laboratório InterMídia pelo compartilhamento de conhecimentos e idéias, pelas correções da qualificação e dissertação, pelos churrascos e companheirismo...

Ao grande colega Walter Soto, que me auxiliou tecnologicamente e que auxiliei em uma pequena parte de seu trabalho de doutorado.

Ao colega Marcelo Mendonça, que compartilhou despesas de nossa república, estudos e momentos de descontração .

A todos os docentes que me proporcionaram conhecimentos fundamentais no mestrado para o desenvolvimento do projeto em suas disciplinas.

Resumo

A competição pelo mercado de comunicações e o crescimento do número de aplicações e de máquinas conectadas à infra-estrutura de redes estão tomando dimensões imprevisíveis. Redes de Alto Desempenho, desenvolvimento de novos protocolos e de técnicas para transmissão de vídeo são extremamente necessários nesta nova configuração das redes. O presente projeto tem como foco o uso do protocolo IPv6 e a técnica de *multicasting* para o desenvolvimento de aplicações multimídia proporcionando economia de largura de banda e de recursos nos servidores. Apresenta uma avaliação da implementação em JAVA do IPv6 *multicasting* (*Java IP version Six readY* - JIPSY) através do desenvolvimento de uma aplicação de vídeo e sua comparação com o protocolos IPv4. Este trabalho foi dividido em quatro partes: revisão da literatura relacionada ao protocolo IPv6, *multicasting* e ATM; configuração de um ambiente de testes; desenvolvimento de uma aplicação usando JIPSY e testes. Os resultados obtidos mostram uma comparação do uso da rede e do tempo de transmissão do vídeo usando os protocolos IPv4 e IPv6 e as formas de endereçamento *multicast* e *unicast*.

Abstract

Competition has been the keyword in the communication industry and the number of applications and machines are increasing very quickly. The development of High Performance Networks, new protocols and new techniques for video transmission are extremely important in this ever growing changing scenario. This project aims at the use of IPv6 *multicasting* in developing effective multimedia applications providing economy of bandwidth and server resources. The JAVA-based JIPSY implementation of IPv6 is assessed and a comparison of the results with a IPv4 implementation is made. This paper has been divided into four parts: studies of IPv6, multicasting and ATM; configuration of an environment of tests, development of a JIPSY-based application and tests. The results show a comparison of bandwidth and transmission time of video between IPv6 and IPv4 protocols, as well as multicast and unicast addressing.

Índice

Índice de Figuras

Índice de tabelas

Capítulo 1 Introdução

As comunicações, atualmente, estão gerando uma grande corrida por novas tecnologias e serviços, ocasionada por diversos fatores, sendo um dos mais importantes a demanda dos usuários por uma infra-estrutura mais poderosa, que possa atender às exigências de usuários e de novas aplicações. Exemplos de aplicações que necessitam de redes com qualidade e com alta largura de banda são as aplicações multimídia, que estão se tornando cada vez mais comuns e importantes, tanto em ambiente acadêmico quanto empresarial.

É fato que objetos de dados multimídia, como imagens estáticas e animadas, vídeo e áudio, ao serem digitalizados, geram uma quantidade muito grande de dados (Thakrar, 1996). Essa quantidade varia em cada aplicação, dependendo da qualidade requerida, como por exemplo, o número de bits para representação dos dados (quantização), que afeta a resolução do objeto. Devido ao fato de objetos multimídia produzirem um grande volume de dados, vários problemas surgem quando esses devem ser armazenados, recuperados, transmitidos e exibidos, sendo os mais críticos o armazenamento e a transmissão.

Sendo assim, tendo como uma nova realidade para os usuários a execução de tais aplicações sobre a atual infra-estrutura de comunicações, onde a Internet teve seu sucesso até o momento, percebe-se que há uma grande necessidade de mudança nesse cenário, ou modificando as tecnologias da infra-estrutura das redes, ou utilizando mecanismos que permitam às aplicações serem executadas com maior desempenho e maior qualidade.

O uso de técnicas como *multicasting* (Tanembaum, 1996)(Stevens, 1998) e compressão de dados (Tanembaum, 1996) permite que a largura de banda utilizada pelas aplicações, principalmente as multimídia, seja economizada. Além disso, *multicasting* permite que

servidores multimídia economizem recursos do sistema, fazendo com que aumente o número de clientes que possam estar em uma sessão, simultaneamente.

Porém, apenas o uso dessas técnicas não seria suficiente para as redes comportarem o tráfego multimídia. Sendo assim, um aumento na largura de banda é importante e diversas tecnologias de rede de alta velocidade estão disponíveis atualmente. Algumas destas tecnologias são *Fast Ethernet* de 100 Mbps, *GigaBit Ethernet* de 1000 Mbps, ATM (*Asynchronous Transfer Mode*) de 155 Mbps e 622 Mbps, FDDI (*Fiber Distributed Data Interface*) de 100 Mbps e futuramente (padronização prevista para 2003) o *10 Gigabit Ethernet* com 10 Gbps.

A grande maioria das aplicações atuais utiliza o protocolo IP (*Internet Protocol*) e este protocolo é extremamente difundido mundialmente. A tendência é que se tenham endereços IP não apenas em *hosts* e dispositivos de rede, mas também em aparelhos eletrônicos como geladeiras, fornos micro-ondas, aparelhos celulares, *palmtops*, carros e outros. Porém, o número de endereços IP possui um limite, cujo esgotamento está previsto para no máximo 10 anos. Soluções devem ser encontradas para evitar tal esgotamento, e uma das soluções mais prováveis de ser utilizada no futuro é a substituição do protocolo IP atual (versão 4 - IPv4) para sua nova versão, o IPv6.

O protocolo de transporte IPv6 (Bradner, 1996)(Loshin, 1999) pode fornecer um maior desempenho nas redes, possibilidade de escalabilidade nos sistemas e segurança de uma forma muito melhor que o protocolo IPv4. Uma provável aplicação que poderá popularizar o uso do IPv6 é a telefonia móvel, que está ocasionando um enorme crescimento no uso de telefones celulares. Com tal protocolo, cada celular poderá ter um endereço unicamente identificado.

1.1 Objetivos

O projeto ***"Transmissão de Vídeo Usando IPv6 e Multicasting em Redes de Alto Desempenho"*** tem como objetivos um estudo do protocolo IPv6 e da técnica de *multicasting*

em redes *Fast Ethernet* e ATM e sua experimentação em uma aplicação de vídeo em redes *Fast Ethernet*.

Realizou-se a configuração de um sistema com suporte a IPv6 e *Multicasting* e desenvolveu-se uma aplicação servidora de vídeo, com o intuito de avaliar os desafios da implementação de aplicações que transmitam vídeo usando *UDP* (*User Datagram Protocol*) *multicasting*, além de avaliar o impacto do uso de IPv6 em sistemas Linux e em redes *Fast Ethernet*.

São utilizadas tecnologias atuais abertas (gratuitas), como JAVA, da *Sun Microsystems*, e Linux. Uma comparação do desempenho do sistema utilizando-se *multicasting* e *unicasting* nos protocolos IPv4 e IPv6 é feita, com o objetivo de se verificar a viabilidade de tais técnicas para aplicações de vídeo.

Um estudo de redes ATM e o uso de *multicasting* nessas redes também é feito, para que trabalhos futuros possam realizar testes e comparações em relação a este trabalho. ATM também é escolhido como foco de estudo devido ao projeto "Rede Metropolitana de Alta Velocidade", ReMAV (projeto Fapesp, 1996/10864-5), que interliga vários laboratórios de pesquisa do ICMC (Instituto de Ciências Matemáticas e de Computação) e IFSC (Instituto de Física de São Carlos) da USP (Universidade de São Paulo) - São Carlos, laboratórios do DC (Departamento de Computação) da UFSCAR (Universidade Federal de São Carlos) com conexões ATM de 155 e 622 Mbps. Futuramente, interligará auditórios, salas de aula das duas universidades e o laboratório de Imagens Tomográficas da Santa Casa de Misericórdia de São Carlos. Um cabo de fibra óptica monomodo de 622 Mbps já está chegando à Santa Casa. A **Figura 1** ilustra o cenário da ReMAV (Pissioli, 1999).

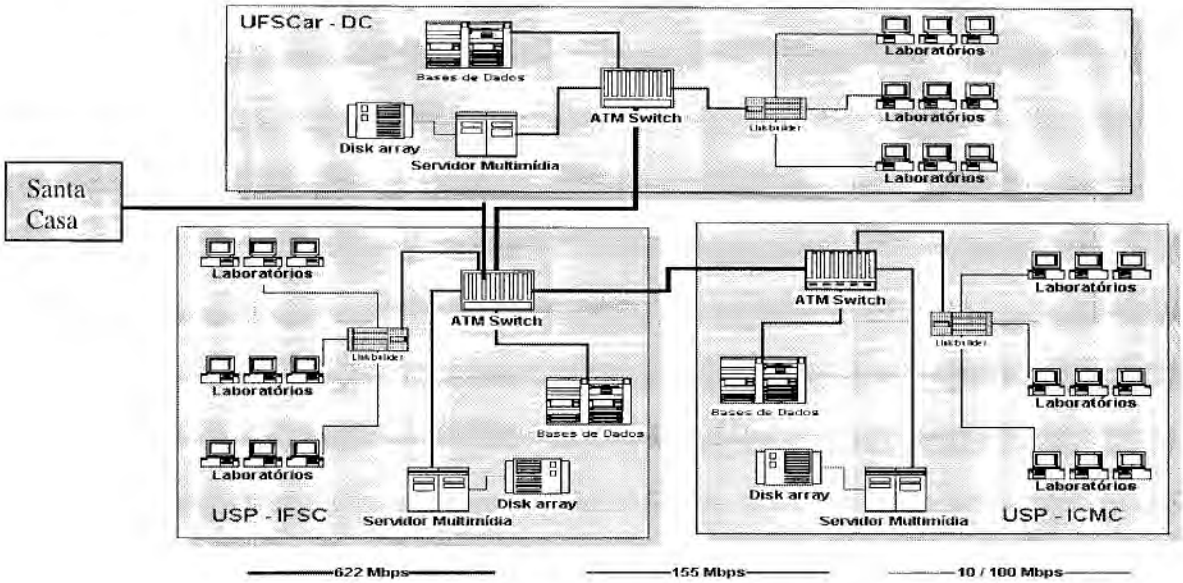


Figura 1: Cenário da ReMAV

O protocolo IPv6, o uso do *multicasting* nos protocolos IPv4 e IPv6 e aspectos das redes ATM e seu suporte a *multicasting* são abordados nos capítulos seguintes, bem como a descrição das tecnologias utilizadas, do cenário e da configuração realizada para o desenvolvimento do projeto, a implementação do servidor de vídeo e os resultados alcançados.

1.2 Organização

No capítulo 2 dessa dissertação serão abordadas as características técnicas do novo protocolo IPv6, tais como a hierarquia de endereçamento, além de ser abordada a integração deste protocolo nos sistemas Linux e Windows e na linguagem de programação JAVA.

No capítulo 3 serão mostradas as características da transmissão de pacotes através de *multicasting* e seu uso nos protocolos IPv4 e IPv6.

No capítulo 4 serão detalhados aspectos tecnológicos das redes ATM, além de um maior destaque à integração dos protocolos IPv4 e IPv6 com ATM, através das técnicas de LANE (*Lan Emulation*) e CLIP (*Classical IP*). O uso da técnica de *multicasting* nas redes ATM também será abordado nesse capítulo.

No capítulo 5 serão abordadas as ferramentas e APIs (*Application Programming Interface*) utilizadas e alguns projetos relacionados.

No capítulo 6 serão detalhados o cenário de testes, a implementação da aplicação de vídeo e os resultados obtidos.

Por fim, no capítulo 7, serão dadas as conclusões do projeto e propostas para trabalhos futuros.

Capítulo 2 Protocolo IPv6

2.1 Considerações Iniciais

IP versão 6 (IPv6) é a nova geração do protocolo IP (cuja versão atual é o IPv4), que tem como principal objetivo aumentar o número de endereços de rede no mundo e prover um melhor desempenho e segurança na transmissão de pacotes nas redes de computadores.

Neste capítulo será apresentada uma visão geral das mudanças do IPv6 em relação ao protocolo IPv4 e a arquitetura de endereçamento do novo protocolo. Informações do suporte ao IPv6 na linguagem JAVA e nos sistemas Linux e Windows serão apresentadas no final desse capítulo.

2.2 Visão Geral das Mudanças

O protocolo IPv6 provê mudanças nos seguintes aspectos: endereçamento expandido, formato de *header* simplificado, *headers* de extensão, rotulamento de fluxo, autenticação e privacidade (Loshin, 1999)(Bradner, 1996).

O endereçamento expandido significa que o IP pode continuar a crescer sem problemas de falta de endereços; o *header* simplificado e de tamanho fixo provê a melhora de eficiência no roteamento devido aos roteadores não necessitarem de grande processamento; os *headers* de extensão, inseridos após o *header* principal, permitem que necessidades especiais possam ser acomodadas sem afetar significativamente o desempenho de roteamento; o rotulamento de fluxo provê um mecanismo para tratamento de *streams* de pacotes, útil para aplicações de

tempo real; autenticação e privacidade proporcionam segurança de informação e recursos, desejáveis para aplicações comerciais, por exemplo (Loshin, 1999).

O endereçamento *unicast* e *multicast* do protocolo IPv4 basicamente continuam imutáveis na versão IPv6, e os protocolos de roteamento *multicast* estão sendo modificado para suportar o endereçamento IPv6, tal como o PIMv6 (*Protocol Independent Multicast version 6*). O espaço de endereço de 32 bits passou para 128 bits e o endereçamento *broadcast* foi excluído e um novo endereço foi inserido: o *anycast*. O protocolo IGMP (*Internet Group Management Protocol*) do IPv4 deu origem a um novo protocolo de descobrimento de elementos de grupo, cujo IPv6 utilizará nos roteadores da rede, chamado MLD (*Multicast Listener Discovery*), baseado no IGMP versão 2 (Fenner, 1999).

O *header* simplificado consiste de oito campos e possui um tamanho de 40 bytes fixos, ao contrário do cabeçalho do IPv4, que varia de 20 a 40 bytes, o que dificulta o processamento dos pacotes nos roteadores. O roteamento torna-se mais eficiente com pacotes de tamanho uniforme. Com isso, por exemplo, o campo *LENGTH FIELD* pode ser eliminado.

As opções são adicionadas no IPv6 em *headers* de extensão separados, diferentemente do IPv4, no qual as opções são adicionadas no fim do *header* IP original. Como exemplo, pode-se citar o *header* de fragmentação, que é inserido pelo nó de origem, após a inserção do cabeçalho principal e antes da inserção dos cabeçalhos de camadas superiores, ou seja, como uma parte do *payload* do pacote. Os nós intermediários não realizam processamento sobre esse cabeçalho e sim, apenas o nó destino, que utiliza este cabeçalho para a remontagem de pacotes.

Além disso, um fato importante é que os roteadores configurados para funcionar com o protocolo IPv6 não realizarão fragmentação, já que esta é realizada apenas pelo nó de origem, que utiliza o protocolo *Path MTU Discovery* para descobrir o menor MTU (*Minimum Transfer Unit)* no caminho da rede até o destino. Assim, a máquina de origem fragmenta o pacote em vários pacotes com o tamanho descoberto, e o *header de fragmentação* é adicionado a cada pacote, permitindo sua remontagem no nó destino. Caso haja necessidade de processamento de pacotes em nós intermediários, o *header "hop-by-hop"* pode ser usado.

A **Figura 2** ilustra os campos do *header* IPv6, os cabeçalhos de extensão e a área de dados do datagrama IPv6 (Bradner, 1996).

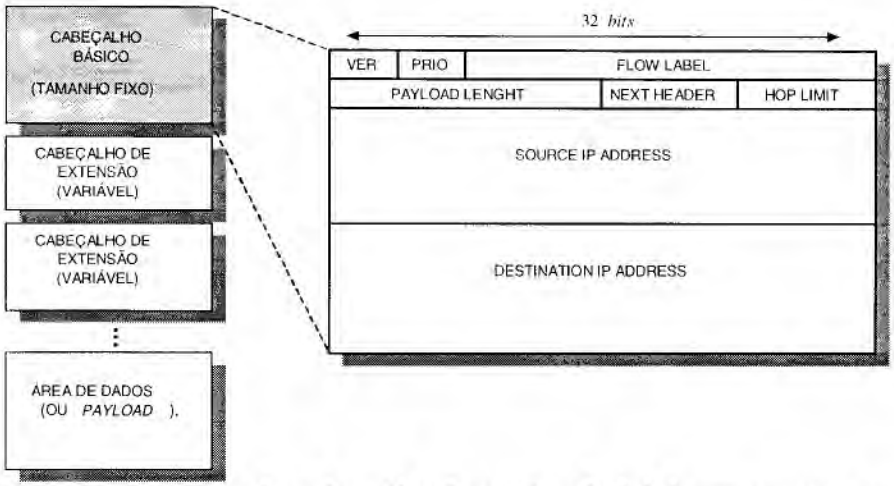


Figura 2: Formato do cabeçalho do protocolo IPv6 e seus campos

O campo *VERSION* deve ser 6 para o IPv6. O campo *PRIO* determina o nível de importância dos pacotes IPv6. Eles são divididos em duas faixas: os valores de 0 a 7 são usados para especificar a prioridade do tráfego de pacotes cuja origem esteja provendo controle de tráfego. A faixa de 8 a 15 é usada para o tráfego de pacotes que não geram respostas ao nó de origem e que necessitam de tráfego prioritário, como pacotes *real-time*.

Um fluxo é uma seqüência de pacotes enviados de uma origem particular para um destino particular (*unicast* ou *multicast*) que o nó de origem deseja manipular por meio de roteadores intermediários. O IPv6 implementa o conceito de fluxo, mediante o qual os pacotes podem ser tratados de maneira diferenciada e mais rapidamente pelos roteadores, que utilizam filas e outros métodos para prover um serviço diferenciado para os fluxos. O campo *FLOW LABEL* é usado para identificar pacotes que pertencem a um mesmo fluxo, já que um nó fonte pode ser gerador de mais de um fluxo de pacotes simultaneamente.

O campo *PAYLOAD LENGHT* contém um valor inteiro igual ao tamanho da área de dados (*payload*) do pacote, em bytes. O campo *NEXT HEADER* indica que existe um *header* de extensão fazendo parte do *payload* do pacote e é representado por um número que indica o tipo deste *header*.

O campo *HOP LIMIT* contém o número máximo de saltos que um pacote pode alcançar, sendo decrementado a cada salto, e descartado se chegar a zero. Por fim, os campos *DESTINATION ADDRESS* e *SOURCE ADDRESS* são endereços de 128 bits de destino e origem, respectivamente.

2.3 Arquitetura do Endereçamento IPv6

A especificação do RFC (*Request For Comments*) 2373 "IPv6 *Addressing Architecture*" define a arquitetura de endereçamento do protocolo IPv6 (Hinden, 1998a).

A maior diferença entre o endereço IPv4 e IPv6 é o seu tamanho. Os endereços IPv6 possuem 128 bits, enquanto que o IPv4 possui apenas 32 bits. Com isto, o IPv6 é capaz de endereçar 340.282.366.920.938.463.463.374.607.431.768.211.456 interfaces (Bradner, 1996).

Existem três tipos de endereços IPv6: *unicast*, *multicast* e *anycast*. O endereço *broadcast* foi descartado neste novo protocolo, já que, na prática, causa uma queda de desempenho na rede, pois os nós precisam processar os pacotes que, muitas vezes, não são a eles destinados. A funcionalidade deste tipo de endereço é provido pelo endereçamento *multicast*, por meio da união de todos os nós da rede a um grupo (Loshin, 1999).

O endereçamento *unicast* identifica uma única interface IPv6. Caso uma máquina contenha várias interfaces, cada uma delas conterá seu próprio endereço de 128 bits. Múltiplos endereços *unicast* podem ser associados a uma interface e múltiplos prefixos de sub-rede podem ser associados ao mesmo *link*.

No endereçamento *multicast*, cada nó pertencente a um grupo de interfaces deve verificar se o pacote deve ser por ele processado, conferindo se pertence a este grupo. Esse grupo é formado por um conjunto de máquinas, que podem estar espalhadas ao longo de redes físicamente separadas (Comer, 1995).

Um endereço *anycast* é associado a mais de uma interface (pertencentes a nós diferentes). É semelhante ao endereço *multicast*, porém endereça somente a interface mais próxima do remetente (menor custo, de acordo com o protocolo de roteamento).

Todas as interfaces requerem pelo menos um endereço *unicast* de *link* local *(local-link)* e podem ter outros tipos de endereços associados (*anycast*, *multicast* e *unicast*) ou endereços com diferentes escopos.

2.3.1 Representação de endereços

Há três formas convencionais para representação de endereços IPv6 como *string* textual (Loshin, 1999):

- A forma mais usual é: X:X:X:X:X:X:X:X, sendo que cada 'X' possui 4 dígitos hexadecimais, cada um com 4 bits. A seguir ilustra-se um exemplo dos endereços IPv4 e IPv6, por fator comparativo:

  IPv4 → 143.107.231.195
  IPv6 → fe80:910A:2222:5765:8475:1111:3900:2020

- É muito comum endereços IPv6 conter muitos zeros. Quando isto acontecer, "::" pode substituir múltiplos grupos de 16 bits. Exemplo (Elz, 1996):

  unicast → 1080:0:0:0:8:800:200C:417A *unicast* → 1080::8:800:200C:417A
  multicast→ FF01:0:0:0:0:0:0:101 *multicast*→ FF01::101
  loopback→ 0:0:0:0:0:0:0:1 *loopback*→ ::1
  não especificado → 0:0:0:0:0:0:0:0 *não especificado* → ::

- Quando os endereços IPv4 e IPv6 forem unidos, a forma será: X:X:X:X:X:X:d.d.d.d, sendo d um número decimal relativo ao IPv4, e X valores hexadecimais.

  0:0:0:0:0:0:13.1.68.3 = ::13.1.68.3
  0:0:0:0:0:FFFF:129.144.52.38 = ::FFFF:129.144.52.38

2.3.2 Representação de prefixos de endereços

A representação dos prefixos de endereços IPv6 é similar a de endereços IPv4 em notação CIDR (*Classless Inter-Domain Routing*) (Fuller, 1993) (Rekhter, 1993):

endereço IPv6/*tamanho do prefixo*

O tamanho do prefixo é um valor decimal que especifica quantos bits contíguos mais significativos representarão o prefixo. Por exemplo, as possíveis representações para o prefixo de 60 bits 12AB00000000CD3 seriam:

12AB:0000:0000:CD30:0000:0000:0000:0000/60
12AB::CD30:0:0:0:0/60
12AB:0:0:CD30::/60

Quando se escreve o endereço do nó e o prefixo juntos, os dois são combinados como segue:

12AB:0000:0000:CD30:123:4567:89AB:CDEF (nó)
12AB:0:0:CD30::/60 (sub-rede)
12AB:0000:0000:CD30:123:4567:89AB:CDEF/60 (união)

2.4 Tipos de Endereços IPv6

Como já foi dito, existem três tipos de endereços IPv6: *unicast*, *multicast* e *anycast*, que serão explicados mais detalhadamente na próxima seção.

2.4.1 Endereços *Unicast*

Os endereços IPv6 *unicast* foram desenvolvidos assumindo-se que o sistema de roteamento da Internet faz decisões de direcionamento (*forwarding*) baseadas em um algoritmo de combinação de prefixos mais longos, com fronteiras de bits arbitrárias, e não tem conhecimento algum da estrutura interna dos endereços IPv6 (Hinden, 1998b).

Existem diferentes tipos de endereços *unicast*, relacionados abaixo (Hinden, 1998a):

- Endereços *unicast* globais agregáveis (*Aggregatable Global Unicast Address*);
- Endereços de uso local baseados em um *Site* (*Site-local-use addresses*);

- Endereços de uso local baseados em um *Link* (*Link-local-use addresses*);
- Endereços IPv6 com endereços IPv4 embutidos;
- Endereços NSAP (*Network Service Access Point*);
- Endereços IPX (*Internetwork Packet Exchange*);

Um nó pode ter pouco ou considerável conhecimento da estrutura interna do endereço IPv6. No mínimo, o nó pode considerar endereços IPv6 sem nenhuma estrutura interna, como ilustra a **Figura 3** (Hinden, 1998b):

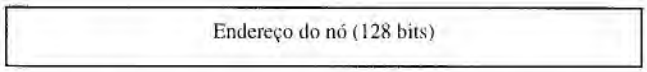


Figura 3: Estrutura mínima de um endereço IPv6

Um nó um pouco mais sofisticado, mas ainda simples, pode ser consciente de prefixos de rede para o *link* em que ele esteja unido, onde diferentes endereços podem ter diferentes valores para N, como ilustrado na **Figura 4** (Hinden, 1998b) (Bradner, 1996):

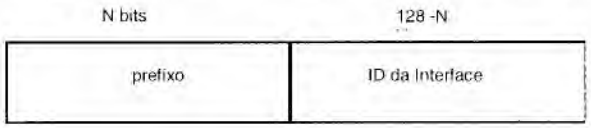


Figura 4: Estrutura de endereço IPv6 com um prefixo de rede

Nós ainda mais sofisticados podem ser conscientes de outros limites hierárquicos no endereço *unicast*. Roteadores podem ter consciência diversificada da estrutura do endereçamento, dependendo de sua posição na hierarquia de roteamento.

Os endereços *unicast* globais agregáveis são organizados em três níveis hierárquicos:

- Topologia Pública;
- Topologia de Site;
- Identificador de Interface.

O formato deste endereço é ilustrado na **Figura 5** (Hinden, 1998b) (Bradner, 1996)(Loshin, 1999):

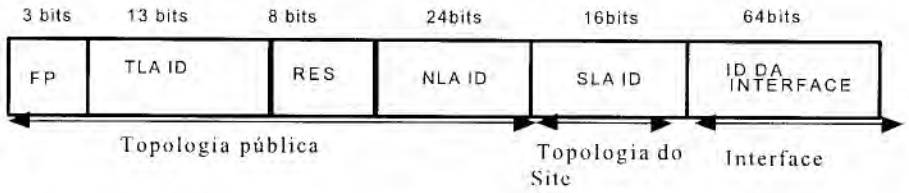


Figura 5: Estrutura do endereço *unicast* global agregável

Onde:

- FP: *Format Prefix* (3 bits) - prefixo utilizado para identificar endereços *unicast* globais, com FP = 001 em binário;
- TLA ID: *Top-Level Aggregation Identifier* (13 bits): identifica provedores de grande trânsito;
- RES: Reservado para uso futuro e deve ser zero. Serve para dar lugar à expansões dos campos NLA ID ou TLA ID;
- NLA ID: *Next-Level Aggregation Identifier* (32 bits), designado para identificar redes de trânsito em um nível hierarquicamente inferior a TLA, de acordo com a arquitetura do 6Bone;
- SLA ID: *Site-Level Aggregation Identifier* (16 bits), identifica a rede do usuário final;
- Interface ID (64 bits): identifica a interface de cada sistema (roteador, servidor, estação, e outros)

2.4.2 Endereços *Unicast* de uso Local em Site e em Link

Endereços de uso local em um Site (*Site-local-use addresses*) são endereços *unicast* com escopo apenas local e podem ser associados a uma Intranet. Seu formato é ilustrado na **Figura 6** (Bradner, 1996)(Loshin, 1999):

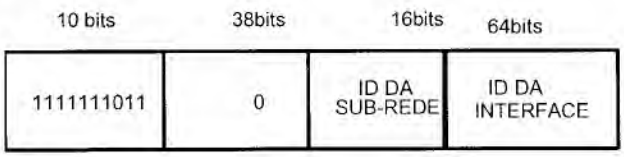


Figura 6: Estrutura do endereço de uso local em um site

Endereços de uso local em um *Link* (*Link-local-use addresses*) serão usados por indivíduos que utilizam um *link* simples, quando nenhum roteador estiver presente. Seu formato é ilustrado na **Figura 7** (Bradner, 1996)(Loshin, 1999):

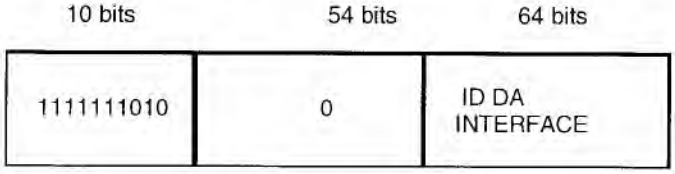


Figura 7: Estrutura do endereço de uso local para um Link

2.4.3 Endereço com IPv4 embutido

Os mecanismos de transição do protocolo IPv4 para o IPv6 incluem uma técnica para nós e roteadores encapsularem pacotes IPv6 sobre uma infra-estrutura de roteamento IPv4. Esta técnica é chamada de tunelamento (*tunneling*). Nós IPv6 que utilizem essa técnica possuem um endereço IPv6 *unicast* especial que carrega um endereço IPv4 nos seus 32 bits de baixa ordem. Esse endereço é chamado de "endereço IPv6 IPv4-compatível" e tem o formato ilustrado na **Figura 8** (Bradner, 1996)(Loshin, 1999):

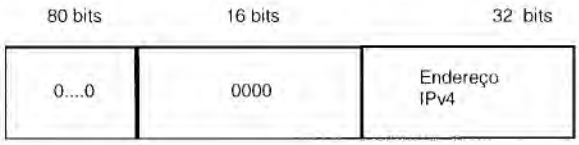


Figura 8: Estrutura do endereço IPv6-IPv4 compatível

O outro endereço IPv6 com IPv4 embutido é usado para representar os endereços de nós apenas IPv4, que não suportam IPv6. Este endereço é chamado "endereço IPv6 IPv4-mapeado" e tem o formato ilustrado na **Figura 9** (Bradner, 1996)(Loshin, 1999):

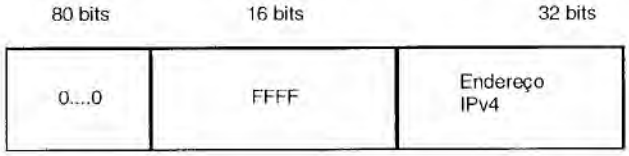


Figura 9: Estrutura do endereço IPv6- IPv4 mapeado

2.4.4 Endereços *Anycast*

Diversos nós (grupo) são associados a um único endereço *anycast*. Apenas a interface com o menor custo, segundo o protocolo de roteamento, receberá o pacote enviado a este endereço. Os nós que estarão associados a este endereço devem ser explicitamente configurados.

Um uso para esse tipo de endereçamento é identificar um conjunto de roteadores pertencentes a uma organização provedora de serviço Internet. Seu formato é ilustrado na **Figura 10** (Bradner, 1996)(Loshin, 1999):

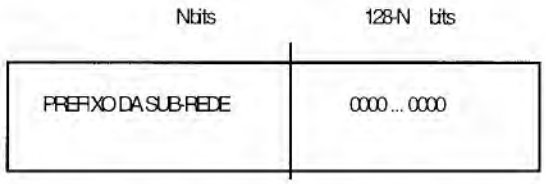


Figura 10: Estrutura do endereço *Anycast*

2.4.5 Endereços *Multicast*

Em um endereçamento *multicast*, um nó deve unir-se a um grupo de interfaces. Esse grupo pode ser formado por um conjunto de máquinas, que podem estar espalhadas ao longo de redes físicamente separadas (Comer, 1995). Devido ao *multicast* ser um tópico muito importante e utilizado neste projeto, o capítulo 3 abordará o assunto de uma forma mais detalhada.

2.5 Implementação de IPv6 nos sistemas Linux e Windows

Os sistemas operacionais Linux, WindowsNT e Windows2000 possuem suporte para o protocolo IPv6 (além de outros como FreeBSD). As distribuições Linux *Red Hat 6.2* e *Slackware 7.0*, com *kernel 2.2.x* ou superior, possuem aplicações de rede que suportam o novo protocolo, como por exemplo, o *ifconfig*, *telnet*, e outros. Para versões do Linux em que as aplicações não suportam o IPv6 nativamente, encontra-se na Internet para *download* uma

ferramenta chamada "*net-kit*" (um pacote com aplicativos de rede com suporte a IPv6), além de outros aplicativos adicionais, desenvolvidos por Peter Bieringer, precursor do IPv6 no Linux (Bieringer, 2000).

Para que as interfaces de rede possam ser configuradas com endereços IPv6, uma recompilação do *kernel* é necessária, habilitando-se o protocolo nas opções de rede. Uma vez habilitado o protocolo, as configurações das interfaces e das rotas podem ser feitas usando-se comandos específicos de configuração de interface e de rotas. Um exemplo, usado na configuração das máquinas utilizadas no ambiente de desenvolvimento do projeto (detalhadas no capítulo 5), foi o comando "*ifconfig eth0 fec0:0:0:1::1/64*", que criou uma entrada no arquivo de roteamento para o prefixo de rede "fec0:0:0:1::" através da interface *ethernet* (eth0) e configurou o endereço da interface para "fec0:0:0:1::1". O roteamento para os endereços *multicasting* (iniciados com ff) e a configuração de um endereço com escopo de *link* nas interfaces de rede são feitos automaticamente na inicialização do sistema, após o *kernel* ter sido recompilado.

O WindowsNT e o Windows2000 possuem um pacote especial desenvolvido pela MSRIPv6 (*Microsoft Research* IPv6,- uma equipe de pesquisa da *Microsoft*), disponível gratuitamente no site desta equipe (MSR, 2000). Para habilitar o protocolo nesses sistemas, o protocolo, incluído no *kit*, deve ser adicionado nas propriedades de rede, em "protocolos".

2.6 IPv6 e a linguagem de programação Java

A linguagem JAVA, utilizada para o desenvolvimento da aplicação deste projeto, possui uma biblioteca de classes responsáveis pela programação em rede, contendo *sockets unicast* (TCP e UDP) e também *multicast* (UDP). Esta biblioteca é a java.net (Sun, 2000a). Porém, tal biblioteca não possui suporte para endereços IPv6, atualmente. Sendo assim, um recurso adicional é necessário, caso o programador queira utilizar endereços IPv6 em sua aplicação.

Para o sistema operacional Linux, uma ferramenta chamada JIPSY (*Java IP Six readY*), que será explanada em detalhes no capítulo 5, foi desenvolvido na Universidade de Tecnologia de Sidney (Flanagan, 2000a) (Flanagan, 2000b). Esta ferramenta, que contém o código para a utilização de *sockets* com suporte a endereços IPv6, além de um *makefile* para a compilação dos códigos e geração de uma biblioteca de sistema compartilhada (arquivo com extensão ".so"), foi utilizada neste projeto.

2.7 Considerações Finais

Neste capítulo foram discutidos os aspectos de mudanças do protocolo IPv6 em relação ao protocolo IPv4, além de um detalhamento da arquitetura e tipos de endereços deste protocolo. Além disso, foram apresentadas informações sobre o suporte do IPv6 na linguagem JAVA e nos sistemas Linux e Windows.

Diante da importância desse novo protocolo e de *multicasting* para transmissão de dados, no capítulo 3 serão explanadas as diferenças dessa técnica em relação ao *unicasting* e ao *broadcasting*. Também serão abordadas as diferenças estruturais dos endereços *multicast* nos protocolos IPv4 e IPv6, bem como uma visão geral do *backbone* experimental para *multicasting* chamado MBONE (*Multicast backBONE*).

Capítulo 3 Multicasting

3.1 Considerações Iniciais

O modo mais simples de transmissão de dados para múltiplos receptores é emitir uma cópia de dados para cada um, individualmente (Savetz, 1999)(Kosieur, 2000). Essa técnica é chamada de *unicasting*, e quanto maior o número de receptores, maior é a largura de banda utilizada. Além disso, *unicasting* limita o número de sessões paralelas no servidor, tendo este que possuir uma grande capacidade de processamento e bufferização para cada um dos fluxos *unicast*. Enquanto o *unicasting* manipula múltiplas transferências de dados ponto-a-ponto, o *broadcasting* envia uma mensagem para cada nó na rede e deixa cada nó determinar se ele deve aceitar a mensagem e processá-la (fazer uma filtragem).

Multicasting possui como abordagem a emissão de dados para múltiplos receptores, com uma origem transmitindo somente uma cópia de dados e replicando-os nos roteadores ou *switches* finais. Os receptores, ao contrário do *broadcasting*, devem indicar aos roteadores finais que desejam receber dados de uma sessão *multicast* (Savetz, 1999). Se não existir nenhum membro da LAN (*Local Area Network*) a qual este roteador pertença na sessão, o emissor não emitirá tráfego algum na rede. Na **Figura 11** são ilustradas as diferenças entre os processos de *unicasting*, *broadcasting* e *multicasting* (Kosieur, 2000).

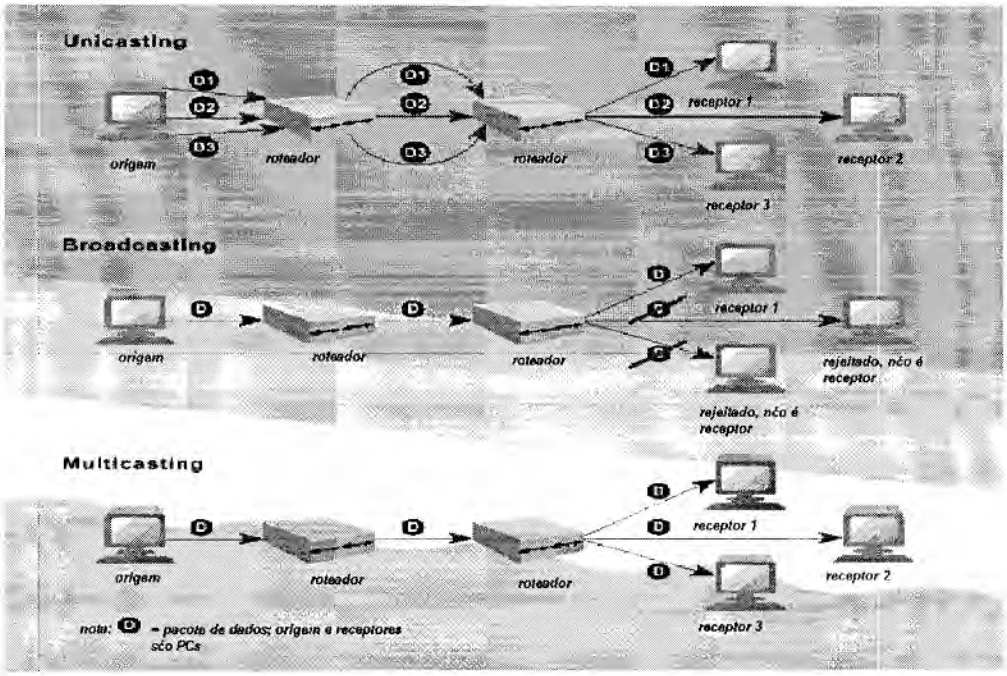


Figura 11: Diferenças entre *unicasting*, *broadcasting* e *multicasting*

Multicasting permite que um datagrama IP seja transmitido para um conjunto de máquinas que formam um grupo de difusão seletiva identificado por um endereço IP único. Esses grupos são formados por um conjunto de máquinas, que podem estar espalhadas ao longo de redes físicas separadas. A entrega de um datagrama *multicast* é realizada com as mesmas características de confiabilidade dos datagramas regulares IP, o que significa que não há garantia contra perda, retardo, duplicação ou entrega fora de ordem para nenhum dos membros do grupo (Comer, 1995).

Um grupo pode ser permanente (possui um endereço conhecido, fixo) ou transiente (criado quando necessário, e descartado quando o número de membros atinge zero ou seu tempo de vida termina). *Multicasting* pode ser utilizado em uma rede física simples ou através da Internet.

Nos tópicos seguintes será explicado o funcionamento da técnica de *multicasting* em uma LAN, comparando-a com a técnica de *broadcasting*, e também será abordado seu funcionamento em uma WAN. Serão abordados alguns protocolos utilizados nesta técnica.

3.2 *Multicasting* em LAN (*Local Area Network*)

A técnica de *multicasting* pode ser utilizada em redes locais (LANs) ou através da Internet (WAN). Para um entendimento do funcionamento do processo de *multicasting*, é importante que seja detalhado o endereçamento *multicasting*, nos protocolos IPv4 e IPv6, e seu mapeamento para endereços MAC *Ethernet*, já que este é utilizado tanto nas redes *Ethernet* quanto nas redes ATM com o uso de LANE.

3.3.1. Endereços *Multicast*

Quando o endereçamento *multicast* é abordado, uma distinção entre o protocolo IPv4 e IPv6 deve ser feita. A **Figura 12** ilustra o formato do endereço *multicast* no protocolo IPv4 (Comer, 1995) (Stevens, 1998).

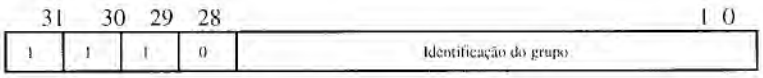


Figura 12: Endereço IP de classe D

Os endereços *multicast*, representados em números decimais através da classe D, vão de 224.0.0.0 até 239.255.255.255, sendo o endereço 224.0.0.0 reservado e o endereço 224.0.0.1 é permanentemente associado a todos os nós do grupo de uma rede local, incluído também roteadores participantes.

Os 28 bits de ordem mais baixa formam o identificador (ID) de grupo *multicast*. Quando este endereço é mapeado para endereços *Ethernet*, apenas os 23 bits de ordem mais baixa são utilizados.

O mapeamento de endereços IPv4 *multicast* em *Ethernet* é descrito no RFC 1112 "Nó *Extensions for IP Multicasting*" (Deering, 1989). O RFC 1390 "*Transmission of IP and ARP over FDDI Networks*" (Katz, 1993) mostra como deve ser feito o mapeamento em redes FDDI e o RFC 1469 "*IP Multicast over Token-Ring Local Area Networks*" (Pusateri, 1993) para redes *Token-Ring*. A **Figura 13** ilustra este mapeamento (Stevens, 1998):

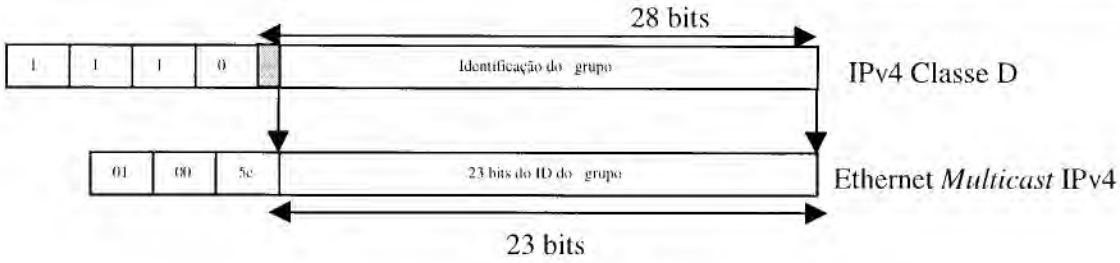


Figura 13: Mapeamento do IPv4 *Multicast* em *Ethernet Multicast*

Como apenas os 23 bits de ordem mais baixa são mapeados para o endereço *Ethernet*, percebe-se que tal mapeamento não é um-para-um.. Alguns endereços IP *multicast* são utilizados pelas autoridades da Internet e correspondem a grupos que sempre existem mesmo que eles não possuam membro algum. Esses endereços são chamados de "bem-conhecidos". Outros endereços *multicast* são disponíveis para uso temporário, correspondendo a grupos *multicast* transientes que são criados quando necessário e descartados quando o número de membros chega a zero.

O padrão TCP/IP para *multicast* define o endereçamento IP *multicast*, especifica como enviar e receber datagramas *multicast* e descreve o protocolo que os roteadores utilizam para determinar os membros de um grupo *multicast* em uma rede. Um nó pode pertencer a um dos três níveis mostrados na **Tabela 1** (Deering, 1989) (Comer, 1995).

Nível	Significado
0	Nó não pode enviar nem receber pacotes IP *multicast*
1	Nó pode enviar mas não pode receber pacotes IP *multicast*
2	Nó pode enviar e receber pacotes IP *multicast*

Tabela 1: Níveis de IP multicast na versão 4

Nós de nível 0, em geral, não são afetados pela atividade de *multicasting*. Os datagramas identificados por um endereço de classe D são descartados por estes nós. O Nível 1 permite que um nó participe de alguns serviços baseados em *multicast*, tais como recursos de localização e reportagem de estado, mas não permite que um nó se una a um grupo. O Nível 2 permite que um nó se una ou deixe um grupo de nós, bem como o envio de datagramas IP para grupos de nós.

O formato de endereçamento do IPv6 é ilustrado na **Figura 14** (Bradner, 1996). As funcionalidades de *multicasting* foram formalmente incorporadas ao IPv4 em 1988, com a definição dos endereços classe D e do IGMP, e ganhou força com o advento do MBONE (*Multicast Backbone*), mas seu uso ainda não é universal. Estas funcionalidades foram automaticamente incorporadas ao IPv6. Isto significa que não será mais necessário implementar túneis MBONE, pois todos os nós e roteadores IPv6 deverão suportar *multicasting* (Bradner, 1996)(Stevens, 1998).

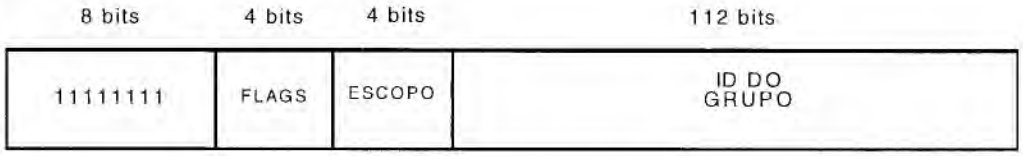


Figura 14: Endereço *Multicast* IPv6

O campo *FLAGS* possui informação sobre o grupo, que pode ser transiente (1) ou fixo (0). *ESCOPO* é um campo de 4 bits usado para limitar o escopo do grupo *multicast*. Esse campo pode adquirir os valores 1 (*node-local scope*), 2 (*link-local scope*), 5 (*site-local scope*) e 8 (*organization-local scope*). O restante dos valores não é associado. O campo *ID* identifica o grupo *multicast*, ou permanente ou transiente, dentro do escopo

O mapeamento do endereços IPv6 *multicast* em endereços *Ethernet multicast* se dá de acordo com o RFC 1972 "*A Method for the Transmission of IPv6 Packets over Ethernet Networks*" (Crawford, 1996), onde se especifica que apenas os 32 bits de ordem mais baixa são utilizados. A **Figura 15** ilustra o mapeamento do endereço IPv6 *Multicast* em *Ethernet Multicast* (Stevens, 1998):

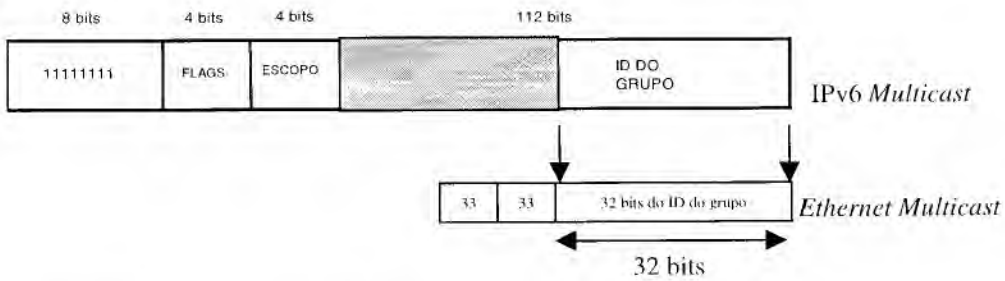


Figura 15: Mapeamento do IPv6 *Multicast* em *Ethernet Multicast*

3.3.2. *Broadcasting* versus *Multicasting*

Uma visão em camadas do processo de *broadcasting* e de *multicasting* será apresentada. A **Figura 16** ilustra um exemplo do processo de *broadcasting* (Stevens, 1998).

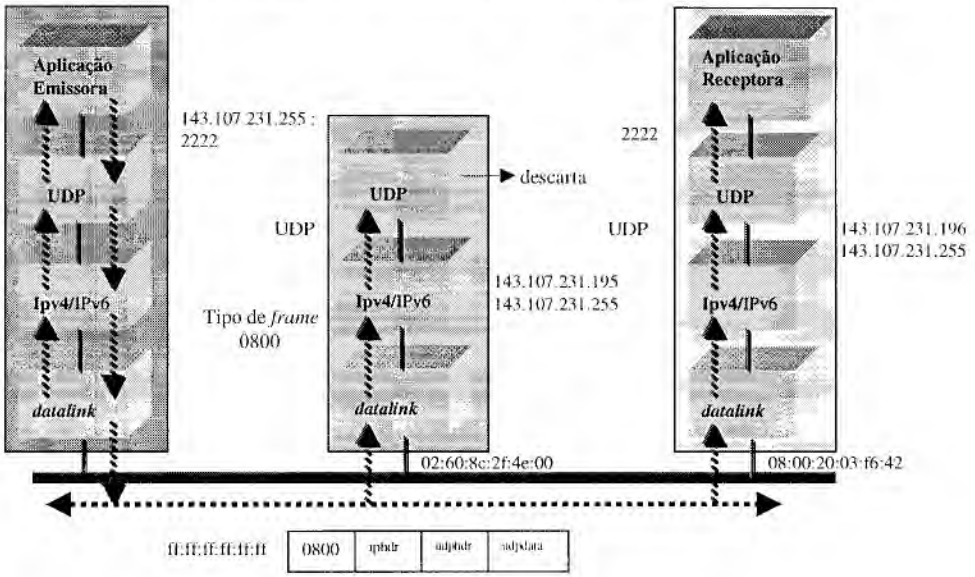


Figura 16: Visão em camadas do processo de *broadcasting*

Quando o nó da esquerda envia um datagrama, ele nota que o endereço de destino é um endereço *broadcast* (terminado em 255) e mapeia isto para um endereço de 48 bits: ff:ff:ff:ff:ff:ff. Todas as interfaces da rede entendem esse endereço, o que faz todos os nós receberem o *frame*. Como o tipo de *frame* é 0800, todos os nós passam o pacote para a camada IP. Como o endereço IP de destino se associa ao endereço *broadcast* (com final 255), e como o campo de protocolo é 17 (UDP), os nós passam o pacote para a camada UDP. No caso do nó do meio, nenhuma aplicação se ligou com um *socket* UDP na porta 2222, sendo o pacote descartado na camada UDP.

No nó à direita, uma aplicação se ligou em um *socket* na porta 2222. Desta forma, o processo responsável pelas funcionalidades da camada UDP reinicia o processo da aplicação ligada a esta porta para que ela possa receber o pacote e o processá-lo na camada de aplicação. Uma desvantagem do *broadcasting*, como se pode perceber, é o grande processamento em todos os nós da sub-rede (até a camada UDP) antes de descartar o pacote.

O *multicasting* tem a vantagem desse fato não acontecer, devido à camada *datalink* dos nós que não se uniram a um grupo descartar os pacotes que chegam até ela. É por isto também que no protocolo IPv6 não se utiliza o endereçamento *broadcast*, e apenas o *multicast* (Stevens, 1998). Na **Figura 17**, a visão em camadas do processo de *multicasting* é apresentada, e seu funcionamento será explanado a seguir (Stevens, 1998).

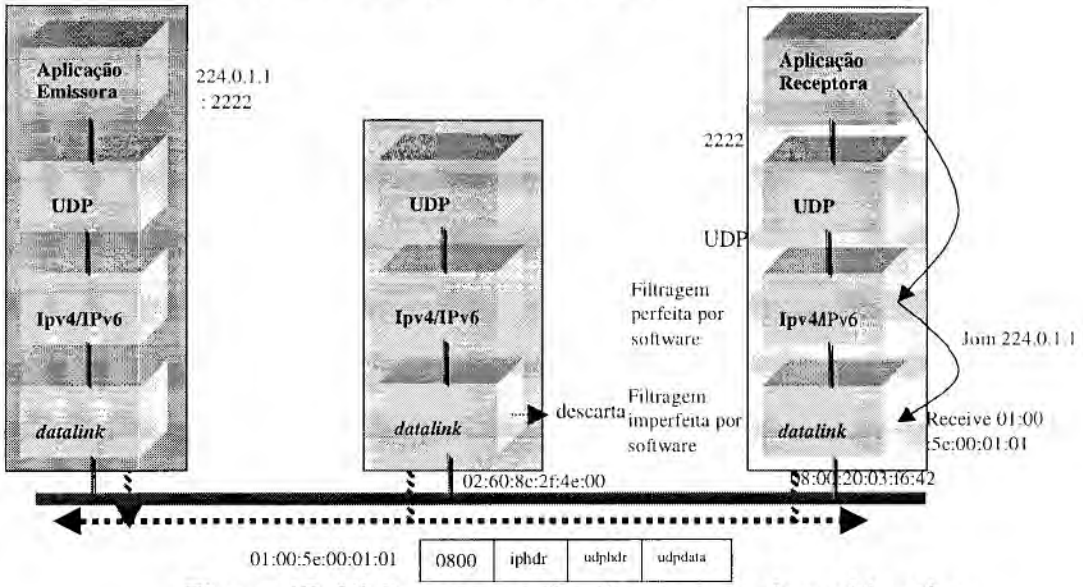


Figura 17: Visão em camadas do processo de *multicasting*

A aplicação receptora no nó do lado direito inicia e cria um *socket* UDP, liga-se à porta 2222 e então une-se (*join*) ao grupo *multicast* com endereço 224.0.1.1. Quando isso acontece, a camada IP (IPv4 e IPv6) salva a informação internamente e então informa à camada *datalink* para receber *frames Ethernet* com destino a 01:00:5e:00:01:01 (ou 33:33:00:00:01:01 no caso de IPv6). Esse endereço é o correspondente ao mapeamento explanado anteriormente (Stevens, 1998)

A aplicação emissora no nó à esquerda cria um *socket* UDP e envia um datagrama para 224.0.1.1 na porta 2222. Para uma aplicação enviar um pacote com endereços *multicast* não é necessário preparar a interface como acontece no nó receptor.

Assumindo que o nó do meio da figura não tenha se unido ao grupo ou não tenha capacidades para *multicasting*, ele ignora o *frame* depois que este passa pela camada *datalink*, devido ao

endereço *Ethernet* de destino não concordar com o endereço *Ethernet* da interface e pelo fato do endereço de destino *Ethernet* não ser um endereço *broadcast*.

O *frame* é recebido pelo nó da direita, baseado em uma filtragem imperfeita feita pela interface usando o endereço de destino *Ethernet*. A filtragem é feita por hardware, e é chamada de imperfeita devido à interface poder receber pacotes de mais de um grupo *multicast*. Por exemplo, os endereços de grupo 224.0.1.1 e 230.0.1.1 serão aceitos de forma equivalente por este nó. Porém, no exemplo, a aplicação se uniu ao grupo de endereço 224.0.1.1. É por isto que na camada IP uma filtragem perfeita por software é realizada, para que apenas pacotes realmente endereçados corretamente, baseados no campo de endereço destino do *header* IP, sejam enviados a camada UDP (Stevens, 1998).

3.3 Multicasting em WAN (Wide Area Network)

De uma forma geral, quando uma instituição ou corporação utiliza-se dos benefícios da técnica de *multicasting* em uma rede local (LAN), onde não há a necessidade de roteadores interligando mais de uma rede, o processo é exatamente como explanado anteriormente. Porém, muitas aplicações utilizam-se da Internet para executar aplicações *multicast*, e para isso é necessário a utilização de roteadores e dispositivos de rede que possam operar com essa técnica (Stevens, 1998) (Tanembaum, 1996).

Roteadores que executam protocolos próprios para manipulação de *multicasting* são chamados de *mrouters*, e tais dispositivos podem estar ligados a outros que não possuem esses protocolos, trabalhando apenas com *unicasting*. Para que uma aplicação *multicast* funcione em uma infra-estrutura de rede em que se tenha dispositivos *multicast*-habilitados e *multicast*-não habilitados, é necessária uma técnica de encapsulamento de pacotes chamada tunelamento (*tunneling*), em que pacotes *multicast* são encapsulados em pacotes *unicast* até o próximo roteador *multicast*-habilitado. Essa técnica é muito utilizada no *backbone multicast* chamado MBONE. Nas seções a seguir, uma descrição do MBONE e do funcionamento da técnica de *multicasting* em uma WAN serão abordadas.

3.3.3. MBONE – Uma Rede Virtual na Internet

O MBONE é uma rede virtual construída sobre a Internet, surgida em 1992, composta de sub-redes, denominadas ilhas, que suportam *multicast*, conectadas umas às outras através de enlaces virtuais ponto-a-ponto. Cada uma das ilhas é composta por uma ou mais redes locais conectando um número de nós clientes e por um nó *mrouter*, que são os roteadores *multicast*.

A comunicação entre estes roteadores é realizada utilizando túneis, que são enlaces virtuais ponto-a-ponto entre os roteadores, possibilitando a transmissão de pacotes *multicasting* entre os roteadores que não suportam esta forma de endereçamento, encapsulando tais pacotes dentro de pacotes *unicast* IP. A conexão entre esses roteadores é denominada túnel (Eriksson, 1994) (Stevens, 1998).

Os *mrouters* replicam e distribuem os dados *multicast* para os túneis que conduzem a nós participantes do grupo e para a rede local, caso exista um nó membro do grupo nela. A **Figura 18** ilustra um tunelamento entre LANs (Tanembaum, 1996).

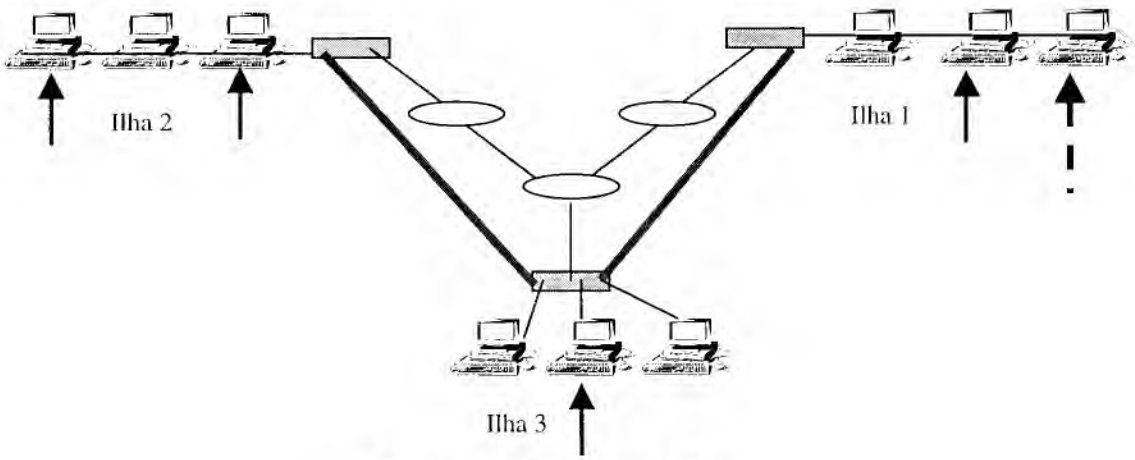


Figura 18: Túneis no MBone

Nessa figura, tem-se que os nós indicados pela seta contínua dentro das ilhas fazem parte de um grupo *multicast*. Os retângulos representam os *mrouters* e as elipses são roteadores sem suporte a *multicasting*. O nó indicado com a seta tracejada representa o emissor para o grupo de receptores *multicast*. Quando o emissor envia pacotes com um endereço *multicast* na LAN, ocorre o processo de filtragens perfeitas e imperfeitas, já explicado anteriormente. Porém,

quando o pacote chega ao roteador da ilha 1, ele é encapsulado dentro de uma pacote IPv4 endereçado ao roteador da ilha 3, e este, quando recebe o pacote, desencapsula e envia às máquinas pertencentes ao grupo.

Diversos protocolos são utilizados para *multicasting*, responsáveis pela união dos nós ao grupo e pelo anúncio de sua particapação ao roteador de borda, e pela manutenção da informação de grupos entre os roteadores. Na seção seguinte será explanado o protocolo IGMP envolvido no funcionamento do *multicasting*.

3.3.4. O Protocolo IGMP (*Internet Group Management Protocol*)

Esse protocolo é utilizado entre nós e *mrouters* e entre *mrouters*. Três funcões básicas são providas por esse protocolo (Johnson, 1997a):

- IGMP_JOIN: permite que um nó una-se a um grupo *multicast;*
- IGMP_QUERY: permite que um roteador receba, periodicamente, respostas dos nós, para que ele mantenha a entrada do grupo na tabela de roteamento;
- IGMP_REPORT: permite que os nós enviem uma resposta para o roteador, dizendo que ainda continuam unidos ao grupo. Essa resposta é emitida apenas por um elemento pertencente ao grupo, para evitar redundância e sobrecarga de respostas na rede. Os outros elementos, que "ouvem" a resposta do primeiro, não emitem a resposta. Esse processo utiliza um *timer* randômico nos nós.

3.3.4.1. Busca Periódica do IGMP

A **Figura 19** ilustra, como exemplo, três nós e um *mrouter* fazendo parte de uma LAN, suas camadas IP e superiores e as interfaces entre as camadas. Essa figura servirá como apoio para a explicação da busca periódica feita no protocolo IGMP. Nesse caso, o nó 1 pertence aos grupos *multicast* A e B, o nó 2 ao grupo A e o nó 3 ao grupo B e C (Johnson, 1997a).

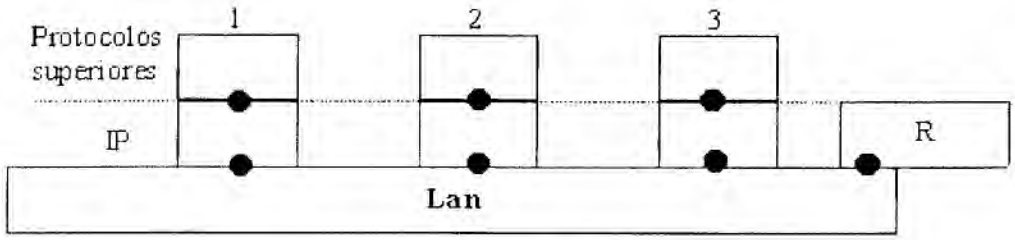


Figura 19: LAN com nós e *mrouters*

Baseando-se na **Figura 19**, a **Figura 20** ilustra o processo de busca de grupos pelos *mrouters*, e permite observar o *start* e *stop* dos *timers* que são usados pelos receptores no momento que recebem o IGMP_QUERY do roteador (Johnson, 1997a).

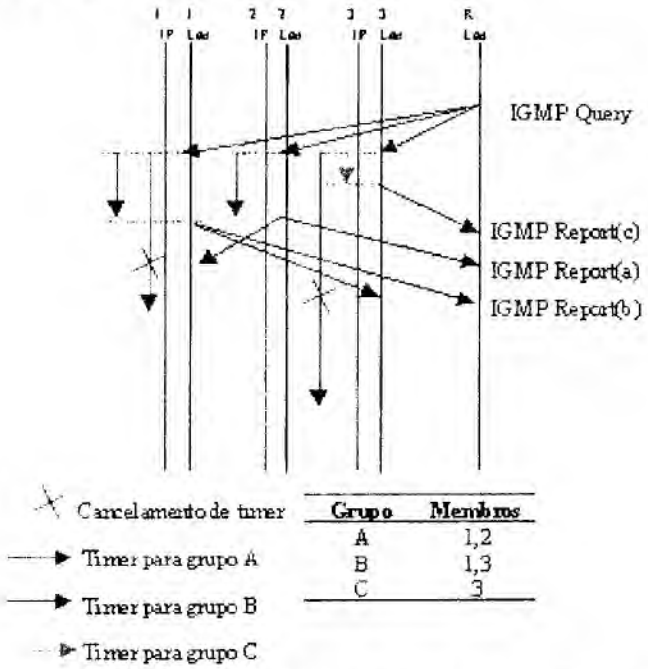
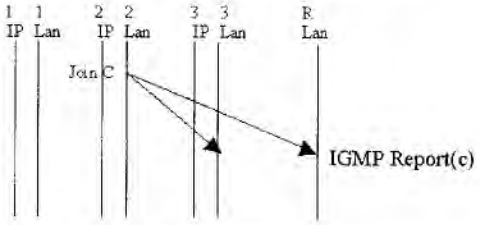


Figura 20: Busca IGMP realizada pelos *mrouters*

Através da **Figura 20**, percebe-se que o roteador envia temporariamente uma mensagem IGMP_QUERY para manter a tabela de grupos. Cada nó que recebe essa mensagem aguarda um tempo aleatório para o envio de uma resposta a essa busca, para que não haja uma inundação de respostas repetidas pelos nós pertencentes a um mesmo grupo. Quando o *timer* de um nó chegar a 0, este enviará a resposta a todos do grupo, ocasionando a atualização da tabela do roteador e a finalização da contagem dos *timers* nos outros nós do mesmo grupo.

3.3.4.2.União a um Grupo

A **Figura 21** ilustra o processo de JOIN de um nó a um grupo (Johnson, 1997a).

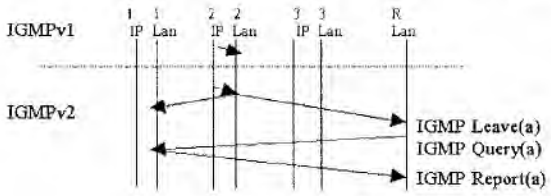


Grupo	Membros
A	1,2
B	1,3
C	3 >2 ,3

Percebe-se que o roteador tinha em sua tabela que apenas o nó 3 estava unido ao grupo C. No momento em que o nó 2 fez um *join*, o roteador atualizou sua tabela e o nó 3 tomou conhecimento da nova união

3.3.4.3.Deixando um grupo

A **Figura 22** ilustra o processo de saída de um nó (Johnson, 1997a).



Grupo	Membros
A	1,2 → 1
B	1,3
C	2, 3

Figura 22: Saída de um grupo no IGMP

Existe uma diferença entre o *leave* do IGMPv1 e IGMPv2. Na primeira, o processo é local, no qual a interface de rede simplesmente pára de receber mensagens *multicast* de certo grupo. No IGMPv2, o nó envia a todos do grupo uma mensagem LEAVE e o roteador imediatamente envia uma QUERY aos elementos da rede, para verificar se é o último do grupo.

3.3.5. Os Protocolos de Roteamento *Multicasting*

Quando um processo em um nó se une a um grupo *multicast*, ele envia uma mensagem IGMP dizendo a eles que se uniu ao grupo. Os roteadores *multicast* (MR) então troca estas informações usando um protocolo de roteamento *multicast*, tal que cada MR sabe o que fazer se ele recebe um pacote destinado ao endereço *multicast*. A **Figura 23** ilustra, sem particularizar um protocolo de roteamento (MRP) em específico, a comunicação entre os roteadores após nós unirem-se a um grupo (Stevens, 1998).

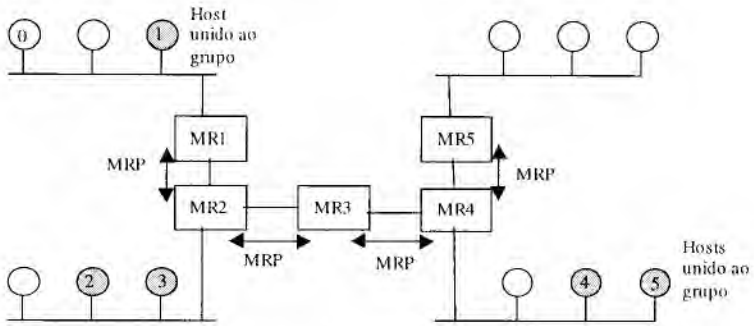


Figura 23: Comunicação de *mrouters* em uma WAN

Os passos para o transmissor (nó 0) emitir pacotes *multicast* para os receptores (nós 1, 2, 3, 4 e 5) são os seguintes (Stevens, 1998):

- Os pacotes emitidos para o grupo são recebidos pelo nó 1 e pelo roteador MR1;
- MR1 envia o pacote *multicast* para o MR2, porque o protocolo de roteamento *multicast* informou ao MR1 que MR2 recebe pacotes detinados a este grupo;
- MR2 envia o pacote aos nós 2 e 3 na LAN, já que estes pertencem ao grupo, e também ao MR3;
- MR3 envia o pacote ao MR4;
- MR4 envia para os nós 4 e 5, mas não para MR5, já que não existe nós unidos ao grupo em sua LAN;

A **Figura 24** ilustra esses passos (Stevens, 1998):

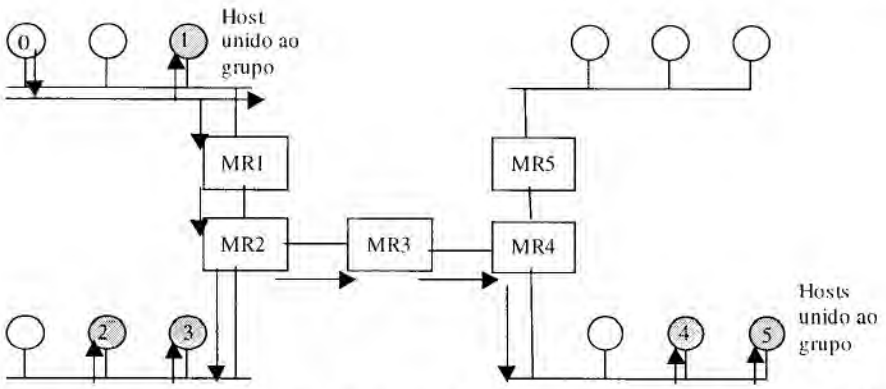


Figura 24: Transmissão de pacote *multicast* para um grupo em WAN

Existem diversos protocolos de roteamento *multicast* que não serão abordados nesta dissertação e diversas empresas produzem dispositivos com tais protocolos. Dentre eles estão: MOSPF (*Multicast Open Shortest Path First*), PIM-SM (*Protocol Independent Multicast - Sparse Mode*) e PIM-DM (*Protocol Independent Multicast - Dense Mode*) (Fenner, 2000) (Johnson, 1997b).

3.4 Considerações Finais

O uso de *multicasting* é interessante tanto para LANs quanto para WANs e é uma técnica que economiza largura de banda das redes e também recursos de nós que estejam transmitindo dados. *Multicasting* é uma técnica usada nos protocolos IPv4 e IPv6 e é objeto de estudo e de testes neste projeto.

A fim de estudar-se alguns aspectos das redes ATM, o capítulo 4 explanará alguns detalhes desta tecnologia, além de enfatizar sua integração com os protocolos IPv4 e IPv6. Por fim, abordar-se-á o uso da técnica de *multicasting* em redes ATM.

Capítulo 4 Redes ATM

4.1 Considerações Iniciais

O Modo de Transferência Assíncrono (*Asynchronous Transfer Mode* - ATM) é um padrão da ITU-T (*International Telecommunication Union Telecommunication Standardization sector*) para transmissão de células, em que informações de múltiplos tipos, tais como áudio, vídeo ou dados são transportadas através de pequenos pacotes de tamanho fixo (células), de forma assíncrona e orientada à conexão (Cisco, 1999).

O ATM utiliza uma tecnologia de chaveamento e multiplexação de pacotes que combina os benefícios de um chaveamento de circuito, como atraso de transmissão constante, com o chaveamento de pacotes, que proporciona flexibilidade e eficiência para o tráfego. Provê uma grande largura de banda e é mais eficiente que as tecnologias síncronas como *TDM (Time-Division Multiplexing)*, pela sua natureza assíncrona. Devido ao fato do ATM ser assíncrono, slots de tempo são disponíveis sob demanda com informações identificando a origem da transmissão contida no cabeçalho de cada célula.

4.2 Algumas Características do Padrão ATM

Serão abordadas nas seções seguintes algumas características do padrão ATM, tais como o formato da célula, dispositivos e interfaces ATM, serviços e conexões ATM. Em seguida, serão mostradas as técnicas *Classical IP* (CLIP) e *LAN Emulation* (LANE) para a utilização dos protocolos IP e IPv6 em redes ATM. Por fim, será citado o projeto desenvolvido inicialmente no ICA (*Institute for Computer Communication and Applications*) (Werner, 2000), que permite que o sistema Linux suporte o protocolo e serviços ATM.

4.2.1 Formato Básico da Célula ATM

Cada célula contém 53 octetos ou bytes. Os primeiros 5 bytes contêm a informação do cabeçalho da célula, e os 48 restantes contêm o *payload* (informações de usuário). Pequenas células de tamanho fixo são apropriadas para transferência de voz e tráfego de vídeo porque tal tráfego é intolerante a atrasos resultantes do *download* de uma grande quantidade de pacotes. A **Figura 25** possui o formato básico de uma célula ATM (Cisco, 1999).

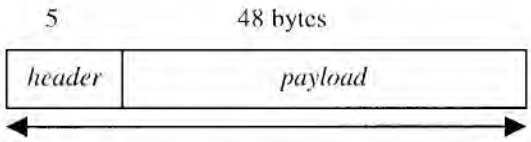


Figura 25: Formato Básico de uma célula ATM

4.2.2 Dispositivos e interfaces ATM

Uma rede ATM é constituída de *switches* ATM e de nós ATM. O nó contém um adaptador de interface de rede ATM, e como exemplos de nós tem-se roteadores, PCs, LAN *switches* e CODECs (Codificadores/Decodificadores).

A *switch* é responsável pelo trânsito de células através da rede ATM. O *job* de uma *switch* ATM aceita uma célula vinda de um nó ou de um outra *switch*. A *switch*, então, lê e atualiza a informação do cabeçalho e rapidamente chaveia a célula para uma interface de saída para o destino.

Switches ATM suportam dois tipos de interfaces: *User-Network Interface* (UNI) e *Network-Network Interface* (NNI). A interface UNI conecta um nó a uma *switch* ou duas *switches* (neste caso uma será usuária da outra) e a NNI conecta duas *switches*.

4.2.3 Serviços ATM

Há três tipos de serviços ATM: serviços com circuitos virtuais permanentes (*Permanent Virtual Circuit* - PVC) e serviços com circuitos virtuais chaveados (*Switched Virtual Circuit* -

SVC), orientados à conexão, e serviços sem conexão. Um PVC permite conectividade direta entre *sites*. Ele garante disponibilidade de conexão e não requer configurações de chamadas de procedimentos entre *switches*. Entre as desvantagens estão a conectividade estática e a configuração manual.

Um SVC é criado e liberado dinamicamente e permanece em uso somente enquanto os dados estão sendo transferidos. É similar a uma chamada de telefone. O controle da chamada dinâmica requer um protocolo de sinalização entre o nó e a *switch*, e possui a vantagem de flexibilidade de conexão e a configuração de chamada pode ser manipulada automaticamente por um dispositivo de rede. Porém, um tempo extra e um *overhead* são requeridos para configurar uma conexão.

4.2.4 Conexões Virtuais ATM

Quando uma conexão é estabelecida entre uma origem e um destino, um canal virtual deve ser estabelecido através da rede antes da transferência de qualquer dado, de forma semelhante a um circuito virtual.

Existem dois tipos de conexões ATM:

- caminhos virtuais (*Virtual Path* – VP): são identificados por identificadores de caminho virtual (*VP Identifier*);
- canais virtuais (*Virtual Channel* – VC): são identificados pela combinação de um VPI e um identificador de canal virtual (*VC Identifier* - VCI). Um caminho virtual é um conjunto de canais virtuais, que são chaveados transparentemente através da rede ATM na base de um VPI. Todos os VCIs e VPIs, todavia, têm somente um *link* particular e são remapeados em cada *switch*.

Um caminho de transmissão é um conjunto de VPs. A **Figura 26** ilustra a concatenação de VCs para criar VPs, que por sua vez, são concatenados para criar um caminho de transmissão (Cisco, 1999)

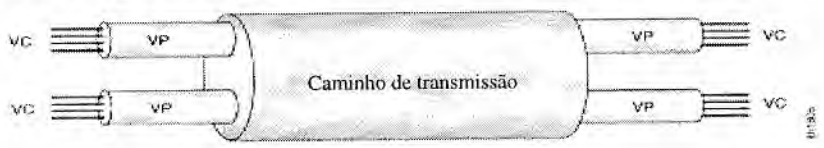


Figura 26: VPs e VCs de um caminho ATM

4.3 Protocolos IP e IPv6 sobre ATM

A tecnologia IP, ao contrário da tecnologia ATM, permite a transmissão dos dados na rede sem o estabelecimento prévio de uma conexão fim-a-fim com algum tipo de contrato. Pacotes são emitidos seguindo a filosofia do *best-effort*, não havendo garantia nenhuma de chegada ou perda destes até seu destino.

Com novas aplicações envolvendo diversas mídias, como áudio e vídeo, surgindo em diversas áreas, tanto acadêmica quanto empresarial, percebe-se que existe uma grande necessidade de qualidade de serviço (QoS) nos protocolos e nas redes de computadores. A tecnologia ATM provê garantia de QoS, através do estabelecimento de contratos e da configuração de parâmetros de conexão que serão utilizados por protocolos de sinalização durante o processo de transmissão de dados. Os parâmetros utilizados no estabelecimento de contrato podem ser: taxa de bits máxima, atraso entre células mínimo e máximo, taxa de pico de células, tempo máximo de rajadas (*burst*) de células, e outros. Tais contratos são estabelecidos pelas aplicações através de sinalização ou manualmente pelo administrador, através de softwares de configuração distribuídos pelas empresas fabricantes.

Porém, quando se deseja utilizar o protocolo IP ou IPv6 sobre as redes ATM, suas características de QoS são, em princípio, anuladas, devido a esses protocolos utilizarem a camada ATM de adaptação 5 (AAL5) para aplicações sem conexão (esta camada é responsável pelo tráfego de dados sem conexão e sem restrições de tempo) (Tanembaum, 1996). Nesse caso, para que qualidade de serviço seja utilizada para as aplicações que não utilizam ATM nativo com suas características de QoS e sim protocolos como IPv4 ou IPv6, técnicas de serviços integrados ou diferenciados devem ser utilizados, tais como DiffServ (*Differentiated Services*), IntServ (*Integrated Services*) e RSVP (*Resource reSerVation Protocol*). Tais técnicas utilizam enfileiramento com precedência de pacotes baseadas em perfis de tráfego, que caracterizam métricas temporais para um fluxo de pacotes de determinada aplicação. Tais métricas envolvem atraso de pacotes, variação de atraso entre pacotes (*jitter*), descarte de pacotes e outros (IETF, 2000a) (IETF, 2000b) (Braden, 1997) (Stardust, 1999).

Sendo assim, pacotes IP ou IPv6 são marcados de acordo com determinado perfil, e nos equipamentos da rede habilitados para tratar de serviços diferenciados, tais pacotes são analisados e uma ação é tomada, de acordo com o estado do pacote (se estiver em conformidade ou não com o perfil estabelecido). Uma ação pode ser um descarte de pacote, o seu atraso ou uma remarcação para outro perfil (IETF, 2000a)(IETF, 2000b)(Braden, 1997).

Este projeto não utiliza nenhuma característica de QoS, mas permite uma abertura conceitual para que projetos futuros preocupem-se com este fato. Os protocolos IPv4 e IPv6 podem ser experimentados sobre ATM, através da técnica de LANE (*Lan Emulation*). Tal técnica permite que as redes ATM comportem-se como redes *Ethernet*, porém com um desempenho de 155 Mbps por porta. Testes básicos usando LANE são tópicos da seção 6.5.1 dessa dissertação.

Nas próximas seções, uma explanação do funcionamento das técnicas LANE e CLIP (*Classical IP*) será apresentada, para que o entendimento do uso dos protocolos IPv4 e IPv6 integrados à tecnologia ATM seja concretizado.

4.3.1 LAN *Emulation* (LANE)

LANE é um padrão especificado pelo ATM FORUM e permite que nós se conectem a redes ATM sem modificação das camadas mais altas para interfaceamento direto ao ATM. Provê uma interface igual às tecnologias de LANs *Ethernet* ou *Token Ring* (Marko, 1996) (Giordano, 1997).

A **Figura 27** ilustra a camada LANE substituindo a camada *Ethernet*. As camadas IP (IPv4/IPv6) e superiores não são modificadas. LANE provê todos os serviços oferecidos pela *Ethernet*, como *broadcast* e *multicast*, usando uma interface similar à *Ethernet*. Comportamentos como colisão e o protocolo CSMA-CD (*Carrier Sense Multiple Access – Colision Detection*) (Tanembaum, 1996) (Comer, 1995) não são emulados (Marko, 1996).

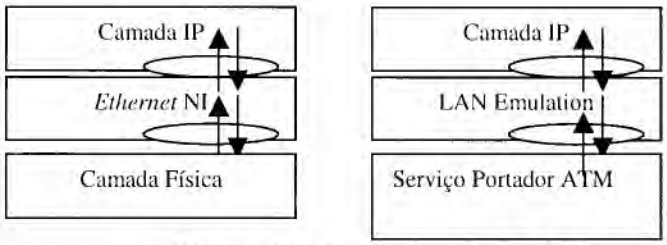


Figura 27: Camadas da LANE

Uma LAN constituída de máquinas com placas ATM, conectadas por uma ou mais *switches* através de uma infra-estrutura ATM e utilizando a técnica de LANE é chamada de ELAN (*Emulated LAN*). A **Figura 28** ilustra a diferença entre uma LAN física e uma LAN emulada (Marko, 1996).

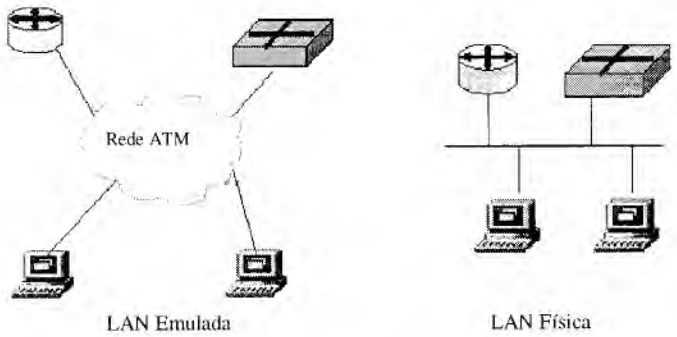


Figura 28: Diferença entre LAN Física e LAN Emulada (ELAN)

4.3.1.1 Arquitetura de uma LANE

Uma LANE consiste de LECs (*Lan Emulation Clients*) e LEService, constituído de LECS (*Lan Emulation Configuration Server*), LES (*Lan Emulation Server*) e BUS (*Broadcast Unknown Server*). LECs se comunicam com LEService através da interface LUNI (*LANE User-Network Interface*) (Marko, 1996).

Os dados são carregados sobre a AAL5 e as conexões podem ser feitas por PVC e SVC. Em ambiente SVC, sinalização é utilizada para estabelecimento de conexão. A **Figura 29** ilustra os componentes de uma LANE e as conexões estabelecidas em uma ELAN (Marko, 1996).

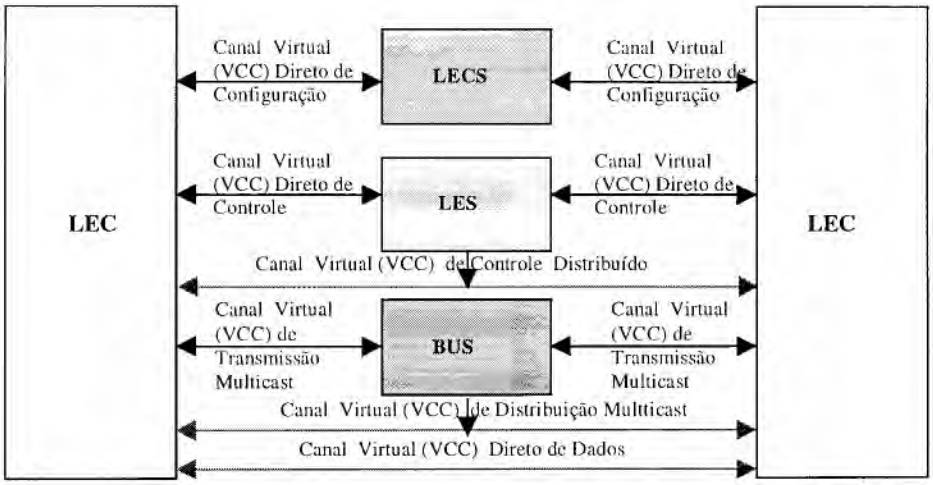


Figura 29: Componentes LANE e conexões

A **Figura 30** ilustra as conexões LANE que emitem e recebem dados, estabelecidas na criação de uma ELAN (Marko, 1996).

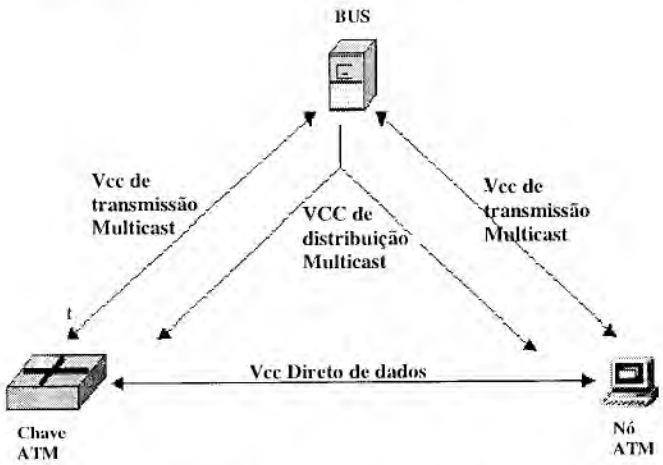


Figura 30: Conexões de dados LANE

A **Figura 31** ilustra as conexões de controle estabelecidas na criação de uma ELAN (Marko, 1996):

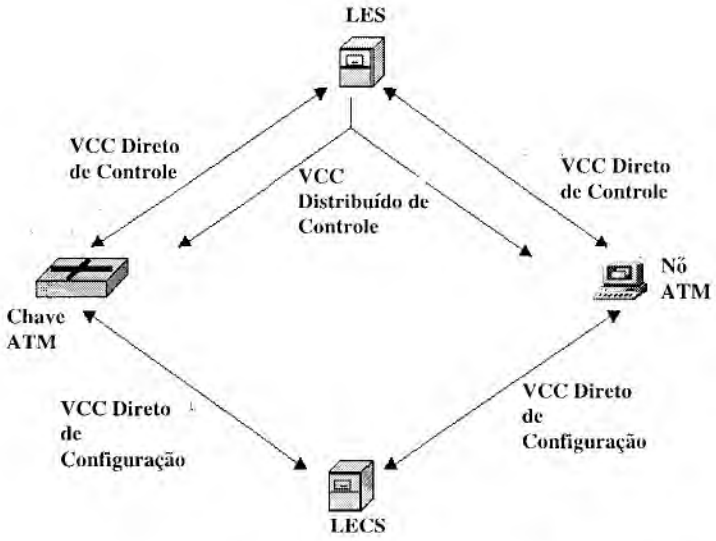


Figura 31: Conexões de controle LANE

O LEC é uma entidade que realiza emissão de dados, resolução de endereço e outras funções de controle. Ele provê às camadas superiores uma interface similar à *Ethernet* e *Token-Ring*.

O LECS implementa a distribuição dos LECs a diferentes ELANs, atribuindo diferentes endereços ATM de LES para os LECs. Essa distribuição é feita pela base de dados do servidor ou por informações enviadas pelos LECs. Se o endereço do LES for dado ao LEC antes da conexão (manualmente), o LECS não é necessário.

O LES desempenha a coordenação de controle para uma ELAN. Recebe registros de endereços MAC e descritores de rotas dos LECs e oferece respostas de buscas por estes. O LES envia essas mensagens usando uma conexão de canal virtual (*Virtual Channel Connection* – VCC) de controle distribuído configurado através de uma conexão ponto-multiponto para cada cliente na ELAN.

O BUS manipula dados de clientes para endereços MAC (*Multiple Access Control*) *broadcast* e *multicast* e alguns dados direcionados a *unicast*.

4.3.2 CLASSICAL IP e ARP sobre ATM

Da mesma forma que o LANE utiliza o termo ELAN para nomear uma LAN Emulada, *Classical IP* (CLIP) utiliza o termo LIS (*Logical IP Subnet*).

Uma LIS é definida por um servidor ATMARP *Server* configurado. Todos os membros de uma LIS usam um mesmo ATMARP *Server* e devem ser capazes de estabelecer uma conexão direta uns com os outros. Múltiplas LISs podem ser configuradas em uma rede ATM e cada LIS deve possuir seu próprio ARP *Server*.

O RFC 1577 "*Classical IP and ARP over ATM*" (Laubach, 1994) especifica como a camada IP deve ser adaptada para funcionar no topo da AAL5, usando os métodos de encapsulamento definidos no RFC 1483 "*Multiprotocol Encapsulation over ATM Adaptation Layer 5*" (Heinanen, 1993) e especifica a funcionalidade do ATMARP. Os pacotes IP encapsulados são carregados no campo de *payload* do PDU (*Payload Data Unit*) da CPCS (*Common Part Convergence Sublayer*) da AAL5. Pelo fato da camada IP necessitar de ajustes para o suporte à camada ATM, não é possível utilizar outros protocolos (IPv6, IPX ou outros) com o CLIP, sem que haja modificação em suas implementações.

O protocolo ARP em *Ethernet* usa um meio *broadcast* para resolução de endereços, o que não acontece em ATM, já que não suporta *broadcast*. Novos métodos são usados para resolver endereços na tecnologia ATM, e o RFC 1577 define protocolos como ATMARP e InATMARP.

4.3.2.1 ATMARP e InATMARP

Para a resolução de endereços IP para endereços ATM, um servidor ATMARP *Server* é necessário. Todos os nós registram-se neste servidor na inicialização da conexão, podendo-se utilizar canais virtuais permanentes ou chaveados. O servidor responde com um InATMARP_REQUEST no mesmo VCC. O nó responde, enviando um InATMARP reply com seu endereço IP e ATM.

No caso de uma máquina que conheça apenas o endereço IP de uma outra máquina e queira enviar dados a ela, a requisição de resolução é feita ao ATMARP *Server* que, baseado nos registros feitos pelos clientes na inicialização, envia a resposta (endereço ATM) ao requisitante. O requisitante, tendo o endereço ATM, pode, assim, conectar-se diretamente ao destino. A **Figura 32** ilustra a operação do ATMARP e InATMARP (Marko, 1996) (Oosthoek, 1997).

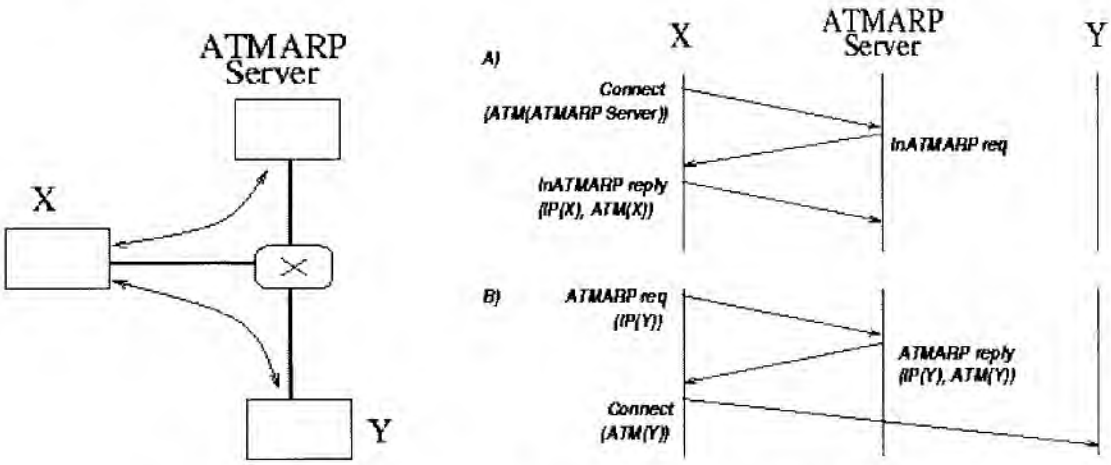


Figura 32: Funcionamento do ATMARP e InATMARP

Nessa figura, no item A, a máquina de origem (X) tenta se conectar ao servidor, e este envia uma mensagem InATMARP para descobrir o par IP/ATM de X. Como resposta, a máquina requisitante envia um InATMARP Reply com estas informações.

No item B, a máquina X possui o IP da máquina de destino, mas não possui o endereço ATM desta máquina. Assim, um ATMARP Request é enviado para o servidor e este, após uma pesquisa em sua tabela de endereços, envia um ATMARP Reply com os endereços IP e ATM do destino. Descoberto o endereço ATM, uma conexão direta é realizada com o destino.

4.4 *Multicasting* em redes ATM

A primeira possibilidade de se executar *multicasting* em redes ATM é através de LANE, em que as particularidades da transmissão *multicast* é similar à transmissão em meios *broadcast*.

Como foi visto anteriormente, a LANE possui um elemento chamado BUS (*Broadcast Unknown Server*), responsável pelo tráfego de pacotes com endereços MAC *Ethernet broadcast* e *multicast*, e também para aqueles com endereços MAC ATM ainda desconhecidos. Então, quando um processo de aplicação envia um pacote com endereço *multicast*, ao chegar na camada *datalink*, que neste caso será uma camada do LANE, um mapeamento será feito para endereços *Ethernet*, como explanado na seção 3.3.1. Todo o tráfego com endereços de *broadcast* e *multicast* será enviado ao BUS através de canais virtuais previamente estabelecidos entre o nó e o BUS e este transmite a todos os nós que se uniram a ele (todos os nós folha da conexão ponto-multiponto) (Oosthoek, 1997).

A **Figura 33** ilustra, em camadas, o processo de *multicasting* em LANE.

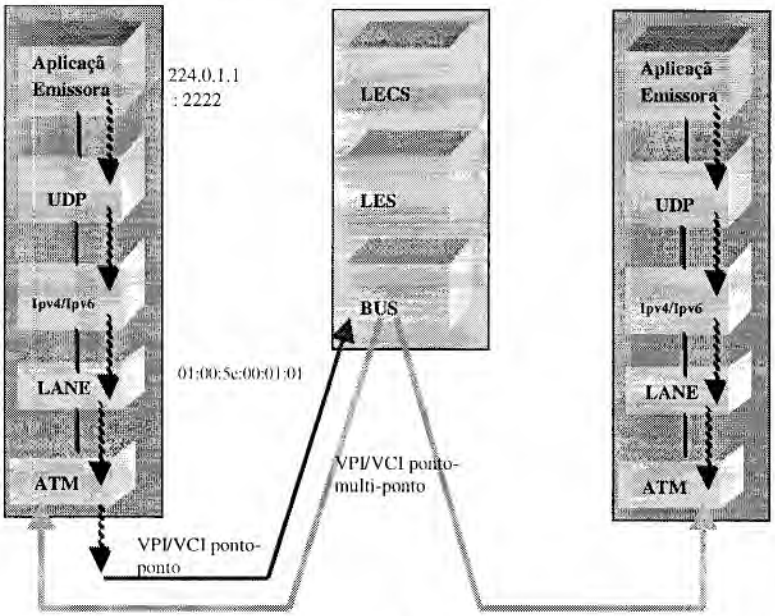


Figura 33: Visão em camadas da transmissão *multicast* usando LANE

Uma outra opção é estender o modelo *Classical* IP e ARP sobre ATM para suportar *multicast*. No caso de se utilizar o IPv6 *multicast*, modificações para suportar o endereçamento IPv6 devem ser feitas neste modelo também.

Esta última abordagem é interessante por permitir-se utilizar as capacidades das redes ATM (MTU maior, eliminação da camada LANE para transportar pacotes IP ou IPv6, uso de QoS). Porém, há uma dificuldade muito grande em transformar endereços IP *multicast* em canais

virtuais ponto-multiponto ATM. Uma conexão ponto-multiponto deve ser explicitamente configurada antes da transmissão de dados, iniciada de um único nó que funciona como o nó raiz da árvore *multicast* e pode ser modificado somente por ele.

O RFC 2022 "*Support for Multicast over UNI 3.0/3.1 based ATM Networks*". (Armitage, 1996) é a especificação para implementar *multicast* diretamente sobre ATM.

4.4.1 *Multicasting* em ATM nativo

O RFC 2022 é uma especificação de *multicasting* de camada 3 geral e usa a camada IP como exemplo. No caso, o IPv6 também seria suportado nesse modelo. Ele é uma especificação que visa implementação do RFC 1112 "*Host Extensions for IP Multicasting*" (Deering, 1989) sobre ATM. Três itens são abordados para o *multicast* em ATM:

- resolução de endereços *multicast* para endereços ATM nativo;
- onde e por qual entidade a cópia de dados deve ser feita;
- administração de sociedade de grupo.

A resolução de endereços do IPv4 ou IPv6 *multicast* deve ser realizada por um servidor chamado MARS (*Multicast Address Resolution Server*), que mantém um histórico de informações de sociedade de grupo. Esse servidor é responsável pela resolução de endereços e atualização de informações de grupo (Micheli, 2000).

Para que a transmissão de dados *multicast* possa ser feita em ATM, existem duas opções de implementação de duplicação de pacotes:

- VC-MESH: cada nó que envia dados *multicast* configura um canal virtual ponto-multiponto para todos os elementos do grupo;
- MCS (*Multicast Server*): uso de um servidor centralizador para replicação de pacotes.

O domínio do servidor MARS é especificado como *MARS-Cluster*, que é um conjunto de nós que escolhem um MARS para registrar sociedade e receber atualizações do grupo. O MARS *Cluster* sobrepor-se-á à LIS.

4.4.2 VC Mesh

O VC-Mesh cria uma malha de VCs e permite conexões multiponto-multiponto, onde cada nó transmissor cria conexões ponto-multiponto para todos os elementos de um grupo. Outra abordagem é a utilização de um servidor *multicast* MCS. As vantagens e desvantagens serão abordadas em seguida. A **Figura 34** ilustra as conexões do VC-Mesh (Oosthoek, 1997).

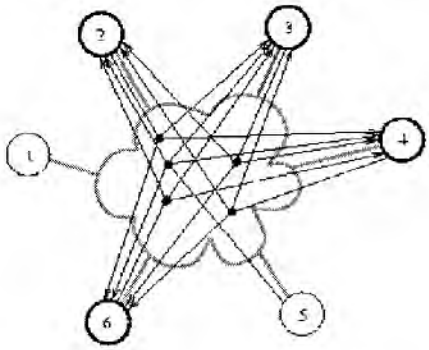


Figura 34: VC-Mesh

VANTAGENS:

- não há ponto de falha e congestionamento;
- VCC Direto ponto-multiponto do emissor aos receptores;
- nenhum pacote é refletido;
- emissão simultânea de dados não é problema, cada nó tem sua fila de VCCs ponto-multiponto para as folhas.

DESVANTAGENS:

- uso pesado das entradas da tabela de conexão nos nós e nas chaves ATM;
- quando o grupo de nós muda, todos os membros devem iniciar o envio de mensagens de adição e remoção de nós, o que pode gastar muito tempo para estabilização, principalmente se o grupo for muito grande;
- número de conexões na chave ATM é muito alto, o que pode ser caro se o provedor cobrar por VC, e os VCCs ponto-multiponto requerem muito recurso da chave;
- sobrecarga de sinalização, pois quando o emissor pára de emitir, o VCC ponto-multiponto é desfeito.

4.4.3 MCS (*MultiCast Server*)

MCS é um servidor que centraliza as conexões com os nós e possui uma conexão ponto-multiponto com os elementos do grupo. A interface do MCS com o MARS é especificada no RFC 2022 e os detalhes de como deve ser sua implementação estão detalhados no RFC 2149 "*Multicast Server Architectures for MARS-based ATM Multicasting*" (Talpade, 1997). A **Figura 35** ilustra as conexões ponto-a-ponto e ponto-multiponto na estrutura do MCS (Oosthoek, 1997).

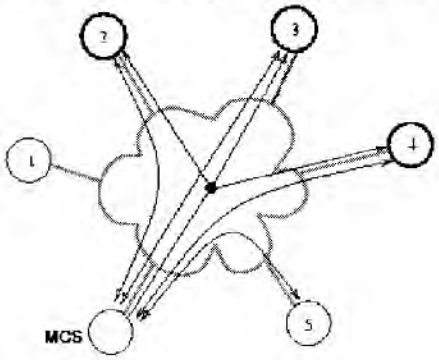


Figura 35: Estrutura do Multicast Server

Diversos MCSs podem ser utilizados com um MARS, o que facilita a redundância de servidores.

VANTAGENS:

- Cada nó precisa apenas de uma ou duas (quando este for emissor e receptor) conexões;
- Muito fácil mudar sociedade do grupo. Somente o MCS precisa ser informado para adicionar ou remover da árvore *multicast*;
- Baixo consumo de recurso nas chaves em relação ao VC-Mesh;

DESVANTAGENS:

- Formação de um gargalo e ponto de falha no MCS;
- Roteamento menos eficiente da fonte para todos os nós;
- Dificuldade de manusear múltiplas emissões simultâneas, pois AAL5 não possui método de ordenação de pacotes;

- MCS envia pacotes para todos os membros, incluindo o emissor. Assim, o emissor deve conter um filtro de pacotes, para evitar o processamento de um pacote refletido.

A escolha entre VC-Mesh ou MCS depende da situação. Quando grupos mudam muito e são muito grandes, o uso de MCS é preferível. Se um grupo é pequeno e estável, e o atraso com desmontagem e retransmissão de AAL5 for inaceitável, o VC-Mesh é a melhor solução.

Para que o MCS consiga os endereços de todos os membros de um grupo para poder criar uma árvore ponto-multiponto e assim poder realizar a transmissão *multicast*, um MARS_REQUEST deve ser enviado para o MARS com o endereço IP classe D (ou IPv6 *multicast*). O MARS responderá com uma lista de endereços ATM dos nós que fazem parte daquele grupo no MARS *Cluster*.

Se não existe mapeamento para os endereços *multicast* no MARS, um MARS_NAK é retornado. Se existe, uma *string* de mensagens MARS_MULTI será enviada ao cliente, e uma *flag* indicará o fim das mensagens. Uma mensagem MARS-MULTI conterá uma lista com os endereços ATM requisitados. O MARS é utilizado nas abordagens VC-Mesh e MCS.

4.5 ATM no Linux

Werner Almerberger, pesquisador do LRC (*Laboratoire de Réseaux de Communication*), implementou programas e APIs para o estabelecimento de conexões e configuração do ATM em sistema Linux (Almesberger, 1996).

Esses programas são instalados e testados no ambiente de testes deste projeto, após o *download* dos programas, que são distribuídos gratuitamente na Internet (Almesberger, 2000). O pacote disponível contém os códigos fonte dos programas, manuais e *makefile* para a geração dos binários.

Tal implementação vem tendo contribuições de implementadores e pesquisadores de todo o mundo, o que vem acrescentando diversas funcionalidades e melhoramentos a esse recurso tão importante para sistemas abertos como Linux.

4.6 Considerações Finais

As redes ATM proporcionam uma grande largura de banda e qualidade de serviço para aplicações multimídia, além de permitir estabelecimento de contratos particulares a cada aplicação.

O uso dos protocolos IPv4 e IPv6 em redes ATM não é uma tarefa simples, o que tem atraído um grande número de pesquisas, principalmente no que diz respeito à utilização de *multicasting*. Este capítulo abordou as características gerais de ATM, enfatizando a integração dos protocolos IPv4 e IPv6 com ATM, além de *multicasting* sobre ATM.

No capítulo 5 serão apresentadas as ferramentas e APIs utilizadas no projeto e alguns projetos relacionados. No capítulo 6 serão abordados o cenário de testes, a implementação e projeto da aplicação servidora de vídeo e os resultados obtidos e o capítulo 7 mostrará conclusões e propostas de trabalhos futuros.

Capítulo 5 Ferramentas e APIs Utilizadas e Projetos Relacionados

5.1 Considerações Iniciais

Diversas tecnologias e ferramentas foram pesquisadas e avaliadas para utilização neste projeto, dando-se preferência para iniciativas gratuitas e abertas. Nas seções seguintes serão apresentadas as ferramentas e APIs, além de alguns projetos relacionados.

5.2 Linguagem, Ferramentas e APIs

Estudaram-se e experimentaram-se neste projeto:

- linguagem JAVA, que é a linguagem utilizada para o desenvolvimento da aplicação de vídeo;
- API JMF (*JAVA Media Framework*), que é uma interface para manipulação de vídeo em aplicativos JAVA;
- ferramenta JIPSY (*JAVA IP Six readY*), que é um *kit* com códigos fontes e *makefile* que habilita às aplicações JAVA fazerem uso de funções do *kernel*, escritas em C, para manipulação de *sockets* com endereços IPv6 em Linux. JIPSY foi desenvolvido utilizando a API JNI (*JAVA Native Interface*), descrita a seguir;
- API JNI, que é uma interface que permite a utilização de código escrito em C/C++ em JAVA ou vice-versa.
- ferramenta de ATM para Linux, que possui códigos fontes e *makefile* para a geração de aplicativos que possibilitam a configuração de canais virtuais e de IP sobre ATM (*Classical IP* e *LANE*). Juntamente com a configuração dos nós para uso desse pacote, foi necessária a configuração da *switch* ASX200BX da ForeSystems (atual Marconi), por

meio do IOS (*Internetworking Operating System*) *ForeThought* 6.1.1. É importante ressaltar que apenas os testes de *throughput* básicos foram realizados sobre LANE.

Nas seções seguintes serão descritas a API JMF, a ferramenta JIPSY e a API JNI com maiores detalhes.

5.3 JMF (*JAVA Media Framework*)

JAVA Media Framework (JMF) é uma API para incorporação de áudio, vídeo e outras mídias baseadas em tempo dentro de aplicações JAVA e Applets, desenvolvida pela Sun Microsystems e pela IBM. Consiste de interfaces que definem o comportamento e a interação de objetos usados para capturar e apresentar as mídias. JMF provê suporte a grande quantidade de padrões multimídia, tais como AIFF, AU, AVI, GSM, MIDI, MPEG, QuickTime, RMF e WAV (Sun, 1999). As implementações do JMF possuem os softwares JMStudio, que permite aos clientes se conectarem a um servidor e receberem um vídeo via transmissão RTP (*Real Time Protocol*), e o JMF *Registry*, utilizado para registrar codificadores, decodificadores, entre outros.

Dispositivos tais como toca-fitas e videos-cassetes provêem um modelo familiar para gravação, processamento e apresentação de mídias baseadas em tempo. A **Figura 36** ilustra este modelo (Sun, 1999):

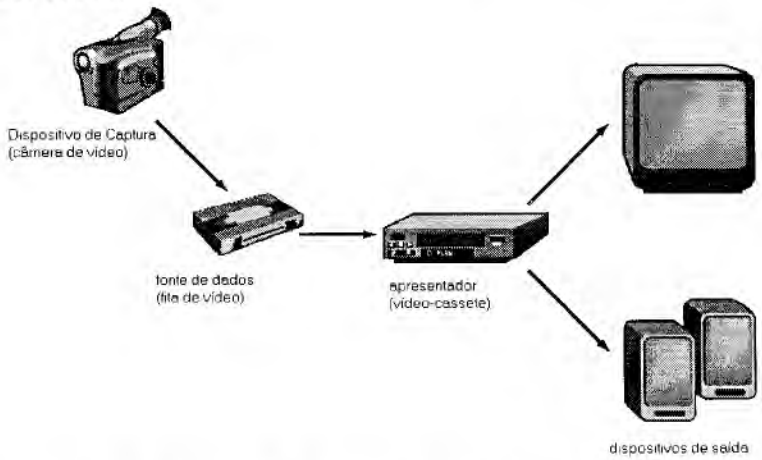


Figura 36: Modelo para gravação, processamento e apresentação de mídias baseadas em tempo

O JMF usa o mesmo modelo básico. Uma fonte de dados (*DataSource*) encapsula um fluxo de determinada mídia (como uma câmera de vídeo) e um apresentador (*Player*) provê o processamento e mecanismos de controle similares aos de um vídeo-cassete. Apresentar e capturar áudio e vídeo com JMF requer dispositivos apropriados de entrada e saída, tais como microfones, câmeras, caixas de som e monitores. Fontes de dados e apresentadores fazem parte da API de alto nível do JMF.

O JMF também provê uma API de mais baixo nível que suporta a integração de componentes de processamento e extensões personalizados. Esta API possui três camadas, ilustradas na **Figura 37**. A camada inferior permite que códigos personalizados pelo desenvolvedor, tais como codificadores, renderizadores e outros, sejam inseridos à arquitetura do JMF e interajam com as funções já existentes, o que torna essa API bastante flexível (Sun, 1999).

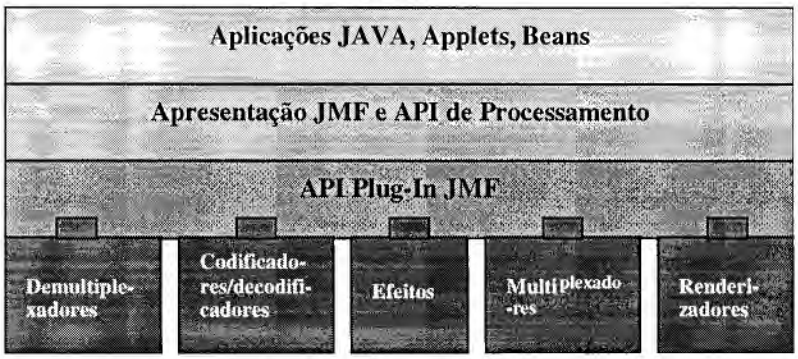


Figura 37: Arquitetura do *JAVA Media Framework (JMF)*

5.3.1 Aquisição de *Frames* Individuais

Com o JMF é possível capturar-se cada frame individual de um vídeo e transformá-lo em determinado tipo personalizado.

A partir de uma fonte de dados (*DataSource*), que pode ser um arquivo armazenado em disco ou uma imagem real originada em uma câmera, cria-se um processador (*Processor*), que pode ser configurado para realizar processamentos diversos. Um processador é um objeto que realiza alguma ação específica sobre os *frames* de um vídeo. Os estados de um processador são mostrados na **Figura 38** (Sun, 1999):

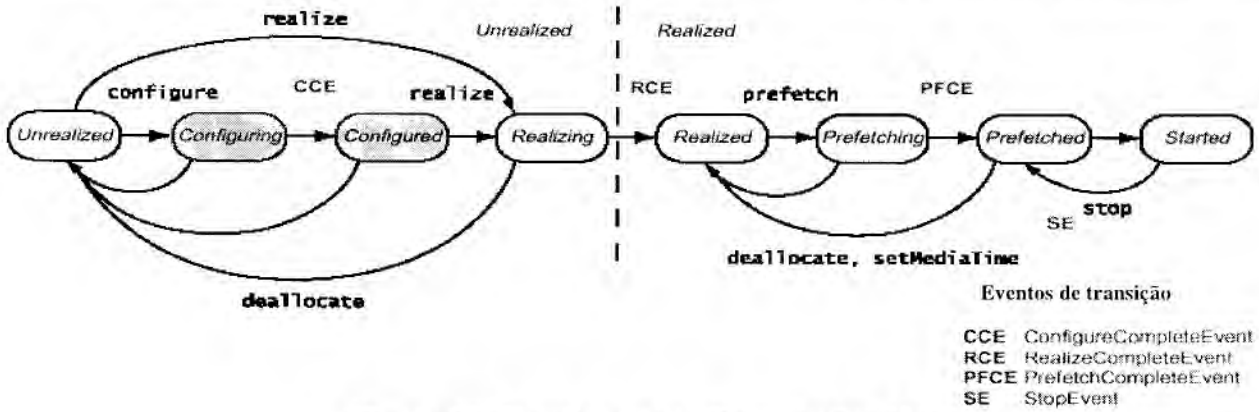


Figura 38: Estados de um Processador JMF

Os passos para realizar a aquisição de *frames* individuais de um vídeo são mostrados a seguir, segundo os estados ilustrados na **Figura 38**:

- define-se um *DataSource* (o arquivo do vídeo a ser processado);
- cria-se um *Processor* (baseado no *DataSource*);
- coloca-se o *Processor* no estado *configured*;
- obtêm-se o controle das trilhas do vídeo;
- determina-se qual tipo de vídeo será utilizado;
- coloca-se o *Processor* no estado *realized*;
- obtêm-se o controle de *frame grabber*;
- obtêm-se o controle sobre *frames* individuais;
- coloca-se o *Processor* no estado *prefetched*;
- realiza-se qualquer operação desejada com os *frames* do vídeo.

5.4 JIPSY (*JAVA IP Six readY*)

JIPSY é uma ferramenta que contém um conjunto de funções que habilitam uma aplicação escrita em JAVA a se comunicar mediante endereços IPv6, substituindo uma porção das classes da API java.net para Linux.

Uma aplicação não necessita ser recompilada para a utilização dessas funções, nem mesmo o código precisa ser modificado, tornando fácil sua utilização. A única modificação a ser feita é a instalação da ferramenta JIPSY e a configuração do sistema para utilizar essa ferramenta.
A ferramenta JIPSY foi desenvolvida por Matthew Flanagan, da Universidade de Tecnologia de Sidney (Flanagan, 2000a), sendo um projeto iniciado no *SourceForge*[1] em 1999 e atualmente vem adquirindo contribuições de desenvolvedores e pesquisadores. Em outubro de 2000 foi lançada a versão 0.2.1 que suporta JAVA 2 e que foi utilizada neste projeto (Flanagan, 2000b). A JIPSY foi desenvolvida utilizando para sua implementação a API JNI (*JAVA Native Interface*) e será abordada na seção a seguir.

5.4.1 JNI (*JAVA Native Interface*)

JAVA Native Interface é uma interface de programação nativa para JAVA que faz parte do JDK (*JAVA Development Kit*), o *kit* de desenvolvimento da *Sun Microsystems*. A JNI permite que programas JAVA que executam em uma JVM (*JAVA Virtual Machine*) interoperem com aplicações e bibliotecas escritas em outras linguagens, tais como C, C++ e Assembly. Além disso, uma API de invocação permite que um código escrito em uma dessas linguagens possa utilizar código escrito em JAVA (Sun, 2000b).

Programadores usam a JNI em situações em que métodos escritos em outra linguagem (nativa) não podem ser escritos completamente em JAVA. Algumas dessas situações podem ser citadas:

- a biblioteca de classes padrão de JAVA pode não suportar recursos necessários às necessidades do programador. Esse é o caso do JIPSY, que utiliza a JNI para aumentar as funcionalidades do java.net, que não possibilita a utilização de endereços IPv6;
- o programador pode ter códigos escritos em outra linguagem, e querer aproveitar tais códigos na linguagem JAVA. Esse também é o caso do JIPSY, que utilizou funções já

[1] É um serviço gratuito para desenvolvedores de código aberto que oferece listas de e-mail, fóruns, e outros

escritas em C, embutidas nos *kernels* 2.2.x ou superior do Linux, que utilizam a nova família de endereços INET6;

- o programador pode querer implementar uma pequena parte de código tempo-crítico em uma linguagem de baixo nível, como Assembly e utilizá-la em JAVA;

Basicamente, a JNI gera uma interface entre um programa JAVA e um programa, por exemplo, em C. A **Figura 39** ilustra essa idéia (Sun, 2000b):

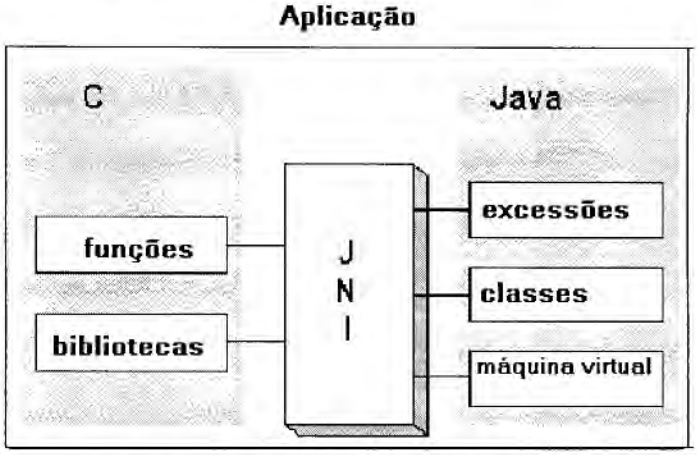


Figura 39: Interface JNI entre JAVA e C

Os passos para a escrita de métodos nativos para programas JAVA são os seguintes:

1) escrever um programa JAVA e criar uma classe que declare um método nativo. Essa classe contém a declaração ou a assinatura para o método nativo;
2) compilar a classe JAVA que declara o método nativo e o método principal (*main*);
3) gerar um arquivo de cabeçalho para o método nativo usando o programa "javah" com a *flag* "–jni". Uma vez gerado esse arquivo, tem-se uma assinatura formal para o método nativo;
4) escrever a implementação do método nativo em uma linguagem, como C ou C++;
5) compilar o cabeçalho e os arquivos de implementação em um arquivo de biblioteca compartilhado;
6) executar o programa "java";

A ferramenta JIPSY cria, quando compilada, uma biblioteca compartilhada chamada *libnet6.so*, e esta é chamada no programa JAVA para utilizar as funções que configuram *sockets* IPv6, escritos em C.

Na seção 5.5 serão abordados alguns projetos relacionados e no capítulo 6 serão mostrados os detalhes da configuração das máquinas utilizadas no ambiente de teste, os cenários configurados, a aplicação desenvolvida e os testes realizados.

5.5 Projetos Relacionados

Um módulo que utiliza a API JMF, desenvolvida pelo trabalho *"Transmissão Multicasting de Vídeo Comprimido"* (Andrade, 2001), foi utilizado no presente projeto. Este módulo realiza a transformação de *frames* de um vídeo armazenado de qualquer tipo para um formato especial de dados a serem transmitidas na rede, que será melhor explicado na seção 6.4, referente aos detalhes da implementação da aplicação.

Outros projetos relacionados indiretamente foram, no decorrer do desenvolvimento deste trabalho, acompanhados e estudados. O trabalho *"Projeto de Sistema de Sincronização de Streams de Vídeo para Serviço Multicast"* (Silva, 2000) propõe uma modelagem em UML (*Unified Modelling Language*) de um sistema de transmissão de vídeo sincronizado usando *multicasting* e uma das tecnologias apresentadas para a implementação é a API JMF, dentre outras.

O trabalho *"Sobre a implementação e avaliação de técnicas de distribuição de vídeo interativo baseadas em conteúdo"* (Goularte, 2000) estuda o protocolo MPEG-4, MPEG-7 e suas funcionalidades, inclusive no que diz respeito à transmissão de vídeo usando *multicasting* e *unicasting*.

O projeto *"Sistema Portátil para Transmissão de Video-On-Demand"* (Pissioli, 1999) ilustra resultados de transmissão de vídeo usando contratos em redes ATM e apresenta conceitos relativos a IP sobre ATM usando LANE e CLIP.

Por fim, um projeto que pôde gerar perspectivas de trabalhos futuros para testes com ATM foi o *"IP Multicast Sobre ATM"* (Micheli, 2000), foco de testes em ambientes ATM configurados para transmissão de dados usando *multicasting* através de LANE e MARS.

5.6 Considerações Finais

Diversas tecnologias de implementação e de redes estão disponíveis atualmente, tais como linguagens, APIs, ferramentas e dispositivos (*switches*, roteadores, *hubs* e outros). A escolha por algumas delas é dependente das necessidades tecnológicas e possibilidades financeiras de cada entidade que delas se utilizem.

Neste trabalho preferiu-se optar por tecnologias que ofereçam grande possibilidades de desenvolvimento, que sejam gratuitas e possuam códigos fontes abertos. Sendo assim, diante das necessidades, JAVA, JMF, JNI e JIPSY (todas gratuitas e abertas) são tecnologias que podem solucionar os problemas e geram uma grande possibilidade de novos projetos e estudos.

No capítulo 6 serão abordados o ambiente de trabalho configurado, a aplicação desenvolvida e os resultados alcançados.

Capítulo 6 Ambiente de Testes e Resultados

6.1 Considerações Iniciais

O projeto ***"Transmissão de Vídeo Usando IPv6 e Multicasting em Redes de Alto Desempenho"*** tem como objetivos o estudo do protocolo IPv6 e da técnica de *multicasting* em redes *Ethernet* e ATM, além da experimentação e a avaliação de IPv6 e *multicasting* usando JAVA e JIPSY em redes *Fast Ethernet*.

Neste capítulo serão abordados:

- configuração do ambiente de testes, no qual foram instaladas quatro máquinas para comunicação *multicasting* de vídeo. Os detalhes serão abordados na seção 6.2;
- desenvolvimento de um servidor de vídeo para transmissão *multicasting* com endereçamento IPv6 e IPv4. Tal servidor foi desenvolvido em JAVA (Cornell, 1998) (Sun, 2000a), podendo ser utilizado nos sistemas operacionais Linux e Windows, quando o protocolo IPv4 é usado, ou somente no sistema Linux, no uso do protocolo IPv6. Os detalhes da implementação e os problemas serão abordados na seção 6.3;
- medidas básicas de *throughput* da rede *Fast Ethernet* e da rede ATM usando LANE e medidas de *throughput* na rede *Fast Ethernet* com a utilização da aplicação desenvolvida, sobre a qual é feita uma comparação com o uso de *unicasting* e *multicasting* nos protocolos IPv4 e IPv6. A seção 6.4 mostrará gráficos com os resultados obtidos das medidas.

6.2 Ambiente de Testes

O ambiente de testes para o projeto, ilustrado na **Figura 40,** possui quatro PCs (*Personal Computers*): S (servidor), C1 (cliente), C2 (cliente) e M (medidor de tráfego). Os detalhes de configuração serão abordados a seguir.

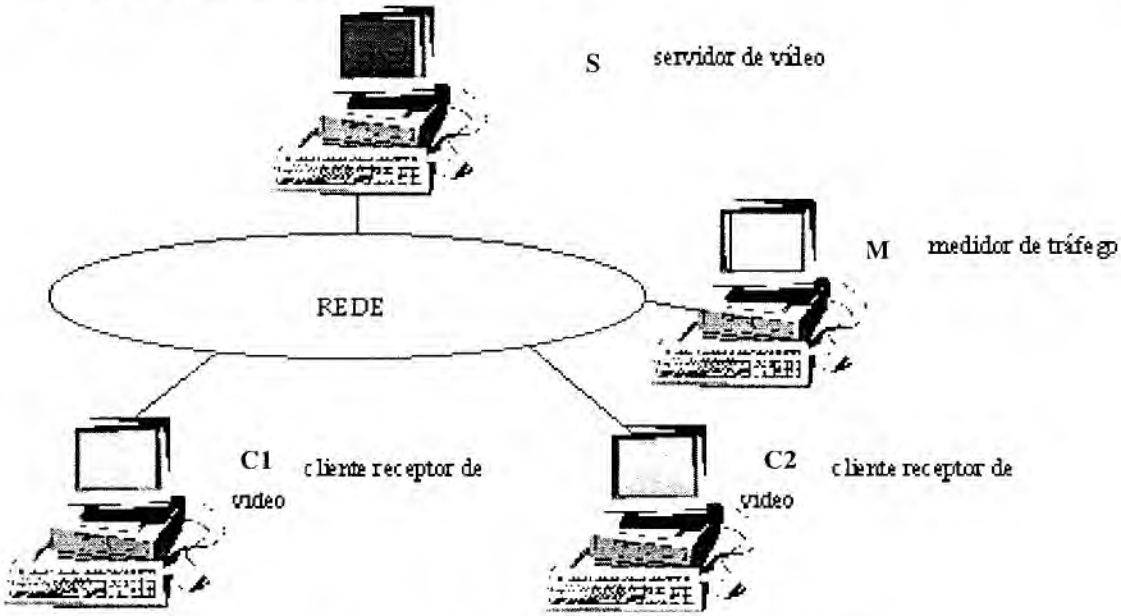


Figura 40: Ambiente de rede do projeto

6.2.1 Servidor

A configuração da máquina servidora é a seguinte:

S:

➔ Sistemas Operacionais:

- *Windows NT 4.0*
- *Linux Red Hat 6.2 (com kernel 2.4.0-teste11)*

➔ Hardware:

- *2 microprocessadores PII 350 MHZ*
- *256 MB de memória RAM*
- *2 discos SCSI QUANTUM de 8 GB, com 2 partições de 4 GB cada um*
- *Placa Fast Ethernet Genius GF100TXRII 10/100Mbps, chipset Realtek 8139*
- *Placa ATM Fore PCA200E 155Mbps*

➔ Configurações de Rede

- Interface *Fast Ethernet*:

IPv4: *143.107.231.195*

Mask: *255.255.255.224*

IPv6: *fec0:0:0:1::10/64*

- Interface LAN *Emulation* (LANE):

IPv4: *172.30.2.10*

Mask: *255.255.255.0*

IPv6: *fec0:0:0:2::10/64*

6.2.2 Clientes e Medidor de Tráfego

Foram configuradas duas máquinas clientes com configurações diferentes e uma máquina para medir tráfego da rede, com as seguintes características:

C1:

➔ Sistema Operacional:

- *Linux Red Hat 6.2(com kernel 2.4.0-teste11)*

➔ Hardware:

- *1 microprocessador Pentium PRO-S 200 MHz*
- *96 MB de memória RAM*
- *1 disco IDE QUANTUM SIROCCO1700A de 1.6 GB*
- *1 disco IDE NEC DSE1700A de 1.4 GB*
- *Placa Fast Ethernet Genius GF100TXRII 10/100Mbps*
- *Placa ATM Fore PCA200E 155Mbps*

➔ Configurações de Rede

- Interface *Ethernet*:

IPv4: *143.107.231.196*

Mask: *255.255.255.224*

IPv6: *fec0:0:0:1::11/64*

- Interface LAN *Emulation* (LANE):

IPv4: *172.30.2.11*

Mask: *255.255.255.0*

IPv6: fec0:0:0:2::11/64

C2:

➔ Sistema Operacional:

- *Windows 98*
- *Linux Red Hat 6.2 (com kernel 2.4.0-teste11)*

➔ Hardware:

- *1 microprocessador Pentium III 500 MHz*
- *128 MB de memória RAM*
- *1 disco IDE QUANTUM Fireball de 10 GB*
- *Placa Fast Ethernet Genius GF100TXRII 10/100Mbps*
- *Placa ATM Fore PCA200E 155Mbps*

➔ Configurações de Rede

- Interface *Ethernet*:

IPv4: 143.107.231.201

Mask: 255.255.255.224

IPv6: fec0:0:0:1::13/64

- Interface LAN *Emulation* (LANE):

IPv4: 172.30.2.13

IPv6: fec0:0:0:2::13/64

MASK: 255.255.255.0

M:

➔ Sistema Operacional:

- *Linux Red Hat 6.2 (com kernel 2.4.0-teste11)*

➔ Hardware:

- *1 microprocessador Pentium 166 MHz*
- *64 MB de memória RAM*
- *1 disco IDE QUANTUM Fireball de 4 GB*
- *Placa Fast Ethernet Genius GF100TXRII 10/100Mbps*
- *Placa ATM Fore PCA200E 155Mbps*

➔ Configurações de Rede

- Interface *Ethernet*:

IPv4: 143.107.231.197

Mask: 255.255.255.224

IPv6: fec0:0:0:1::12/64

- Interface LAN *Emulation* (LANE):

IPv4: 172.30.2.12

IPv6: fec0:0:0:2::12/64

MASK: 255.255.255.0

6.3 Cenários

Utilizando as máquinas do ambiente de testes, um *hub* 10/100 Mbps da 3Com e uma *switch* ATM ASX200BX da *ForeSystems,* configuraram-se dois cenários para testes e avaliações. Tais cenários são detalhados a seguir.

6.3.1 Cenário 1: Servidor e clientes em único segmento ATM (LANE)

Esse cenário foi configurado com o objetivo da familiarização com a tecnologia ATM em *switches* ATM e também em sistemas Linux. Nesse cenário apenas testes de *throughput* de rede básicos foram feitos. Medidas utilizando-se *multicasting* e IPv6 sobre LANE não foram realizados, apesar de terem sido testados.

Nessa configuração, ilustrada pela **Figura 41**, a largura de banda total da rede é de 155Mbps (19.325 Mbytes por segundo) em cada porta da chave ATM. Os clientes estão conectados à chave através de cabos UTP (*Unshielded Twisted Pair*), usando uma placa ATM. Um total de 8 máquinas pode ser conectado à chave simultaneamente, devido a esta possuir 8 portas ATM de 155Mbps cada uma.

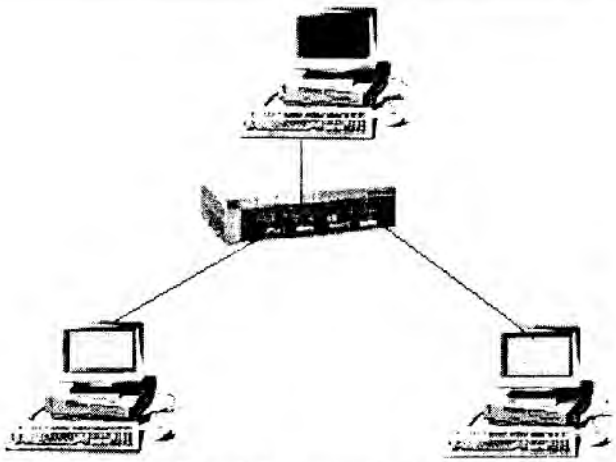


Figura 41: Cenário usando a chave ATM

6.3.2 Cenário 2: Servidor e clientes em único segmento *Fast Ethernet*

Nessa configuração, ilustrada pela **Figura 42**, a largura de banda total da rede é de 100Mbps (12.5 Mbytes por segundo) compartilhadas entre as máquinas.

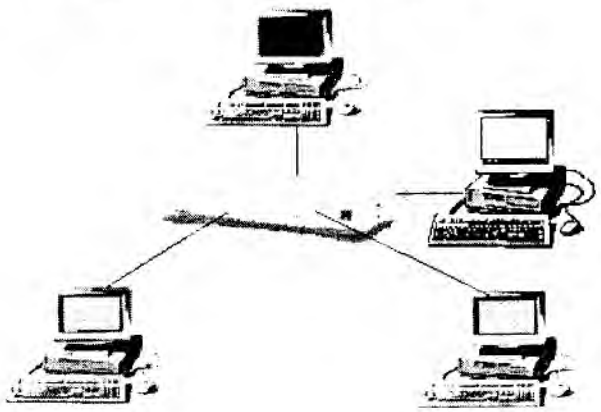


Figura 42: Cenário usando hub *Ethernet* 10/100 Mbps

Os clientes estão conectados à chave através de cabos UTP, usando uma placa *Fast Ethernet* de 100Mbps. Um total de 8 máquinas pode ser conectado ao *hub* simultaneamente, devido a este possuir 8 portas *Fast Ethernet* de 10/100 Mbps.

Nesse cenário foram feitos testes de *throughput* de rede básicos e também testes de medidas com a aplicação desenvolvida, utilizando-se endereçamento IPv4, IPv6 e *multicasting*.

6.4 Aplicação Desenvolvida

A experimentação da problemática envolvida no desenvolvimento de uma aplicação de transmissão de vídeo, da técnica de *multicasting* com endereços IPv4 e IPv6, e a realização de medidas comparando *unicasting* e *multicasting* nesses protocolos, foram os motivadores para o desenvolvimento de uma aplicação de transmissão de vídeo que utiliza os protocolos IPv4 e IPv6. Além da aplicação que transmite vídeo usando *multicasting*, foi desenvolvida uma versão que utiliza *unicasting*, para que uma comparação pudesse ser feita no momento da realização das medidas. Ambas aplicações (*multicasting* e *unicasting)* consistem de um programa servidor que transmite *frames* de vídeo e de um cliente que recebe os *frames* de vídeo, sem realizar processamento algum sobre os pacotes e *frames* recebidos.

As medidas realizadas sobre essa aplicação utilizaram clientes "*dumb*", que não mostram os *frames* de vídeo na tela, mas apenas recebem os pacotes relativos aos *frames* de um vídeo. Devido à capacidade de transmissão de pacotes pelo servidor ser muito maior que a capacidade de recepção pelos clientes, não foi utilizado para os testes um cliente que processasse os *frames*. Além disso, o projeto concentrou-se na avaliação da capacidade de transmissão do servidor.

Um cliente que receba os *frames* e os apresente na tela do cliente foi projetado e também implementado, mas não é utilizado para testes de medidas devido à necessidade de atraso sobre cada pacote emitido pelo servidor e de um controle rigoroso de tempo. Neste capítulo, o projeto do cliente completo será abordado. Uma explicação detalhada das aplicações *multicast* e *unicast* será abordada nas seções seguintes.

6.4.1 Aplicação *Multicast*

A primeira versão da aplicação foi o desenvolvimento do cliente "*dumb*" e do servidor de vídeo que envia *frames* usando *multicasting*. Suas funcionalidades são as seguintes:

Servidor:

a) Possibilidade de receber conexão de mais de um cliente;

b) Emissão de *frames* de vídeo usando *multicasting*;

c) Possibilidade de escolha entre endereços IPv4 e IPv6;

d) Tratamento de datagramas UDP para os casos de desordenação e perda;

Cliente:

a) União a grupos *multicast*;

b) Recepção de *frames* individuais de vídeo por *sockets multicast*;

c) Possibilidade de escolha entre endereços IPv4 e IPv6;

A linguagem JAVA foi utilizada para o desenvolvimento da aplicação, que teve como alvo o sistema operacional Linux, por ser um sistema aberto que está em ascensão, tanto no meio acadêmico, quanto no meio empresarial. Apesar disto, quando se utiliza endereçamento IPv4, a aplicação fica interoperável em outros sistemas operacionais, tais como WindowsNT e Windows9x.

Porém, quando se utilizam os endereços IPv6, apenas o Linux pode ser utilizado, devido à utilização da ferramenta JIPSY explanada anteriormente. Como este utiliza uma bilblioteca compartilhada compilada para a plataforma Linux e o sistema de cada cliente, não é possível executar a aplicação na plataforma Windows.Nas seções seguintes, detalhar-se-á uma descrição das características do vídeo utilizado e do funcionamento da aplicação.

6.4.1.1 O vídeo utilizado e sua transformação

Para os testes desta aplicação foi utilizado um vídeo capturado em laboratório, com as seguintes características:

- 320x240 *pixels*;
- 24 bits (3 bytes) por *pixel*;
- 999 *frames*;
- 15 *frames* por segundo (fps);
- duração de 60 segundos.
- Tipo AVI, sem compressão

Como já exposto no capítulo anterior, foi utilizado um módulo (Andrade, 2001) para transformação de *frames* de qualquer tipo de vídeo para sua representação RGB (*Red, Green,*

Blue), utilizando-se um *array* de tipo "inteiro". No caso deste projeto, apesar da possibilidade de se utilizar diversos tipos de vídeo, apenas o vídeo AVI sem compressão capturado no laboratório foi utilizado nos testes. Esse módulo utiliza o processo de captura de *frames* individuais explanado na seção 5.3.1 e transforma *frames* de vídeo em um *array* de "inteiros" em que cada posição possui um valor (A, R, G ou B) de um *pixel* do *frame*. Isto é ilustrado na **Figura 43**:

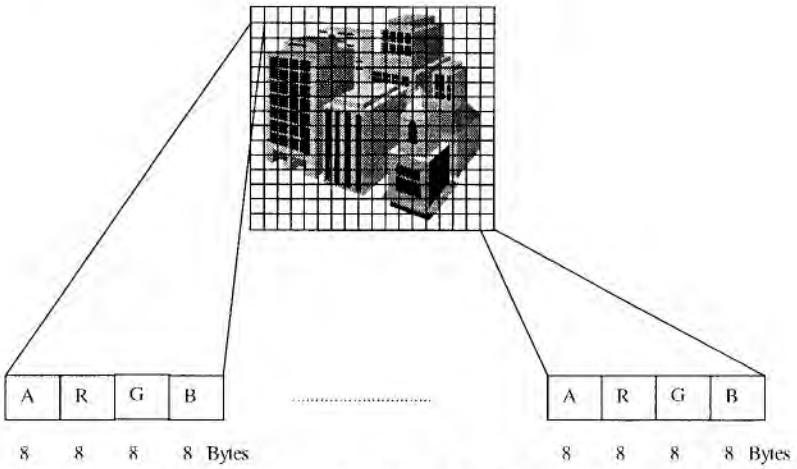


Figura 43: Um *frame* de vídeo e sua representação em RGB

Sendo assim, após ser capturado do disco e passar pelo processador que utiliza JMF, cada *frame* terá o formato **ARGB**ARGB**ARGB** ..., sendo que A é sempre 255 (não sendo necessária sua transmissão pela rede), relativo à transparência do *frame*, R é o valor relativo à crominância vermelha, G à verde e B à azul. Cada um desses valores são representados por 1 byte, sendo cada pixel representado por 4 bytes.

Com a configuração do vídeo utilizado no projeto, tem-se que cada *frame* possuirá 320x240x3 (230.400, ou 225K) bytes, que deverão ser transmitidos na rede (a transparência, por sempre possuir valor 255, não será transmitida). Porém, no *payload* de um datagrama cabem somente 65535 bytes (em sua carga total), o que ocasiona a necessidade de divisão de um *frame* em 4 partes (3 partes de 196605 bytes e 1 parte de 33795 bytes).

Escolheu-se, neste projeto, utilizar-se datagramas com tamanhos iguais e de, no máximo, 16000 bytes, por ocasionar um desempenho melhor e uma fragmentação em *frames Ethernet* menor em relação ao *payload* de 64KB.

Sendo assim, os seguintes cálculos foram usados para a divisão de um *frame* para gerar datagramas com *payload* de tamanhos iguais:

$$\left\lceil \frac{(largura \times altura \times bytes\_por\_pixel)}{(tamanho\ máximo\ do\ payload)} \right\rceil = número\ de\ datagramas$$

$$\frac{(largura \times altura \times bytes\_por\_pixel)}{(número\ de\ datagramas)} = tamanho\ do\ payload\ de\ todos\ os\ datagramas$$

No caso de se utilizar pacotes com, no máximo, 16000 bytes, os cálculos resultam em:

$$\left\lceil \frac{(320x240\ x3)}{(16000)} \right\rceil = 14.4\ (ou\ 15)\ pacotes$$

$$\frac{(320x240\ x3)}{(15)} = 15360\ bytes,\ cada\ datagrama$$

Dessa forma, para que um *frame* do vídeo seja transmitido na rede, são necessários 15 pacotes com *payload* de tamanho 15360 bytes. Como os *sockets multicast* trabalham com o protocolo UDP, existe um risco de que os pacotes se percam e cheguem fora de ordem, sem que o servidor tenha conhecimento do ocorrido.

Assim, juntamente com os valores RGB do *frame* do vídeo, algumas informações são transmitidas a fim de habilitar um certo controle sobre os pacotes UDP transmitidos. Tais informações são:

- número do *frame* (fr): 4 bytes;
- número do datagrama transmitido (pc): 4 bytes;
- *flag* (fl): 1 byte. Indicadora do acontecimento ou não de algum comando pelo cliente. Tal *flag* foi projetada para que um cliente pudesse ser implementado para realizar controles sobre o vídeo, como REWIND, FORWARD, PAUSE e PLAY.

A **Figura 44** ilustra um *frame* (*array* com representações RGB) sendo dividido, hipoteticamente segundo os cálculos ilustrados anteriormente, e em cada divisão as informação adicionais que serão transmitidas na rede. Tal figura não ilustra exatamente a realidade, já que cada pacote conterá um número maior de posições do array:

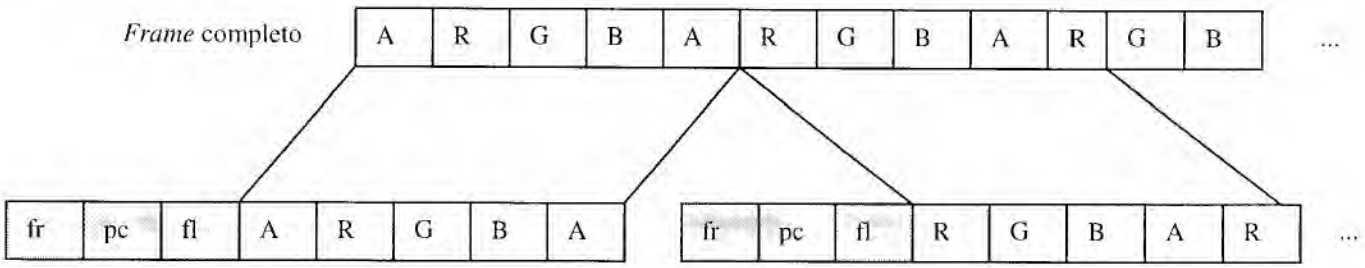


Figura 44: *Frame* dividido em pacotes com informações inseridas

Tais informações adicionam 9 bytes em cada datagrama. Assim, cada datagrama possuirá 15369 bytes.

Com o intuito de tornar mais eficiente o processo de transmissão de *frames* de vídeo na rede, devido ao desempenho do JMF para a manipulação de vídeo no Linux ser relativamente baixo, um arquivo pré-formatado foi utilizado. Com esse arquivo com extensão ".ll", o *loop* principal de emissão de *frames* consegue enviar um vídeo a uma taxa de *frames* muito maior quando comparado à utilização do módulo JMF capturando e transmitindo cada *frame* individualmente.

O módulo que captura *frames* individuais usando JMF foi utilizado para a criação do arquivo ".ll". Sendo assim, quando um cliente pede um vídeo armazenado com determinado tipo (MPEG, AVI, etc), uma *thread* que desempenha as funções do módulo citado (explanada na próxima seção), cria um arquivo ".ll". Essa *thread* captura cada um dos *frames* de vídeo e os transforma em um *array* RGB, dividindo este *array* com o cálculo explanado anteriormente e inserindo as informações adicionais a cada divisão deste *array*. A **Figura 45** ilustra este processo:

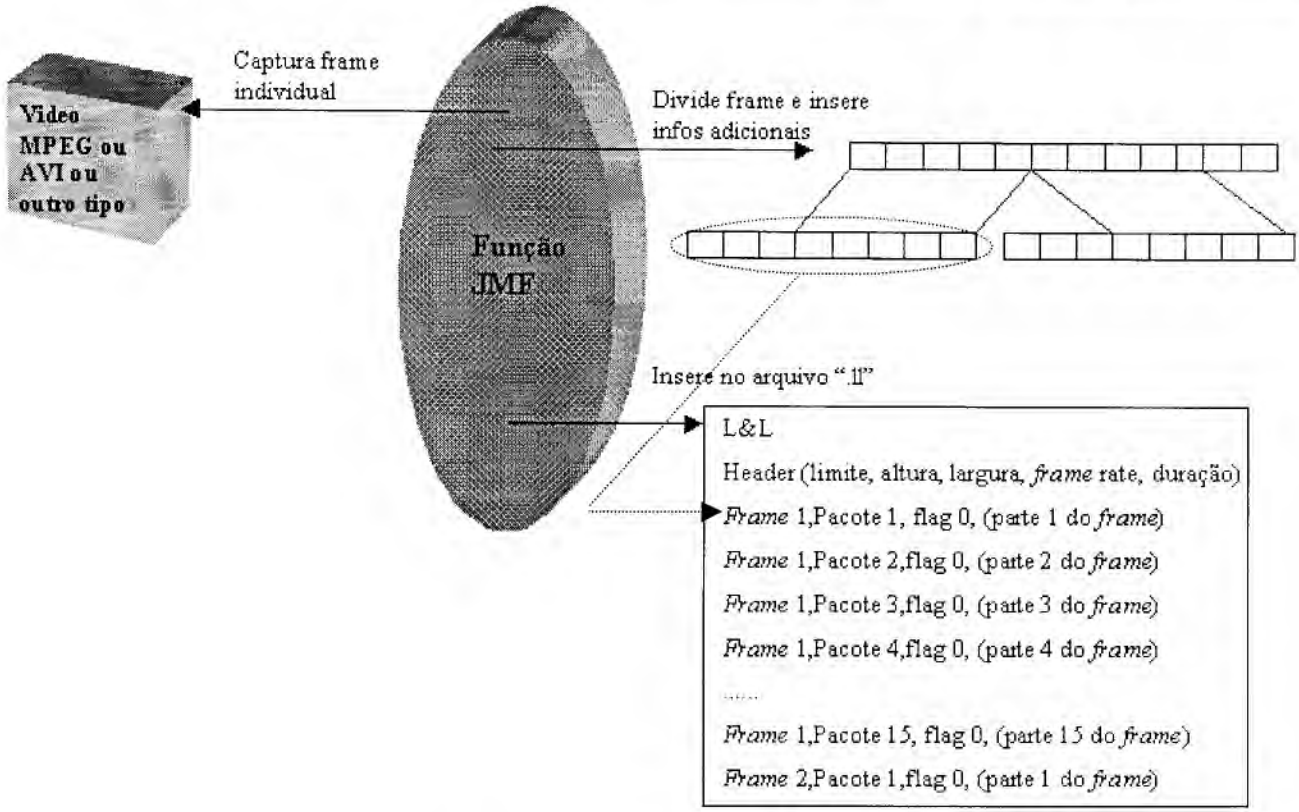


Figura 45: Processo de criação do arquivo do tipo ".ll"

A seguir, uma descrição do funcionamento da aplicação será detalhada.

6.4.1.2 Detalhes do funcionamento da Aplicação

A aplicação consiste de um cliente e de um servidor que utilizam as funcionalidades *multithreading* da linguagem JAVA. Como a máquina servidora possui dois processadores e o sistema operacional Linux foi recompilado para suportar processamento simétrico entre múltiplos processadores, o uso de múltiplas *threads* permite um melhor desempenho nas tarefas do servidor.

A aplicação é inicializada através da seguinte linha de comando:

{java | jipsy} porta {IPv4 | IPv6} diretório_vídeo

Esse comando permite uma escolha entre o programa "java" do JDK e o "jipsy" (comando criado com o *makefile* incluído na ferramenta JIPSY para Linux). Se o primeiro é escolhido, apenas endereços IPv4 podem ser utilizados, tanto no cliente quanto no servidor. Caso o segundo programa seja escolhido, apenas endereços IPv6 podem ser usados. A opção entre os protocolos IPv4 e IPv6 na linha de comando acima dependerá do programa escolhido.

O cliente é inicializado através da seguinte linha de comando:

{java | jipsy} endereço_servidor porta {IPv4 | IPv6}

Após servidor e cliente terem sido inicializados, uma troca de mensagens é feita no momento em que o cliente se conecta ao servidor. A **Figura 46** ilustra o processo de troca de mensagens inicial, que ocorre quando o servidor tiver sido inicializado e estiver escutando às conexões dos clientes em uma determinada porta específica. Essa comunicação inicial é realizada inicialmente por conexões TCP, e a transmissão de vídeo por UDP *multicasting*.

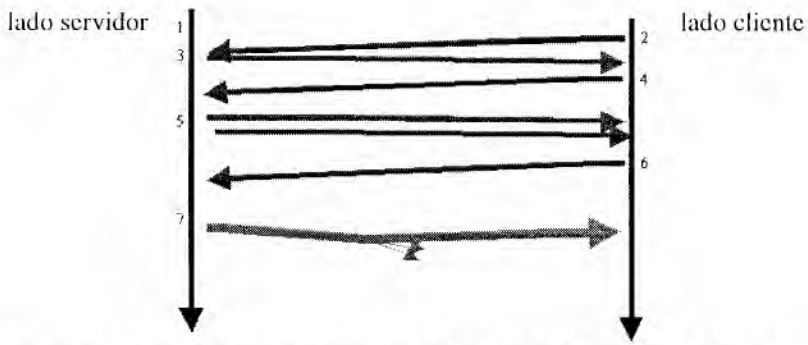


Figura 46: Troca de mensagens Iniciais entre servidor e cliente

1. servidor escuta em uma porta específica;
2. cliente conecta-se ao servidor na porta especificada;
3. servidor envia lista e *status* (Executando e Não Executando) dos vídeos existentes. A aplicação servidora já está preparada para manipular diversos vídeos simultâneamente. Porém, nos testes realizados apenas um vídeo foi utilizado;
4. cliente envia o número do filme desejado. No caso, apenas um poderá ser escolhido;
5. servidor envia dados de configuração do vídeo, endereço *multicast* do vídeo, e outros;
6. cliente envia "Ok" para o servidor iniciar a transmissão *multicast*;
7. servidor inicia transmissão de vídeo usando uma conexão UDP *multicasting*;

O vídeo, assim, é transmitido até o final. O endereço IPv4 *multicast* 224.10.10.10 é utilizado para o vídeo, no caso de se optar por endereços IPv4. Caso endereços IPv6 sejam escolhidos, utiliza-se o endereço IPv6 *multicast* ff02::10. É importante ressaltar que, para o uso de *multicasting* no sistema Linux, uma configuração é necessária, além da recompilação do *kernel*. Uma rota deve ser inserida no sistema, caso contrário as interfaces de rede das máquinas não serão capazes de mapear endereços *multicasting*. Sendo assim, para endereços IPv4, foi inserida uma rota para a rede 224.0.0.0 com *netmask* 240.0.0.0, e para endereços IPv6 foi inserida uma rota para a rede ff00::0/16.

Com essas configurações, a aplicação, no momento de se unir ao grupo por meio do endereço *multicasting* e através de uma porta, é capaz de realizar o mapeamento *IP Multicasting/ Ethernet Multicasting* como explicado na seção 3.2.1.

6.4.1.3 Múltiplas *threads* da aplicação

Para que pudesse ser explorada a vantagem da máquina servidora possuir dois processadores, foram utilizadas várias *threads* no desenvolvimento da aplicação. O servidor, em sua execução, gera uma grande quantidade de *threads,* proporcional ao número de clientes conectados ao servidor.

As funcionalidades das *threads* do servidor e do cliente são ilustradas nas **Figura 47** e **Figura 48:**

Servidor

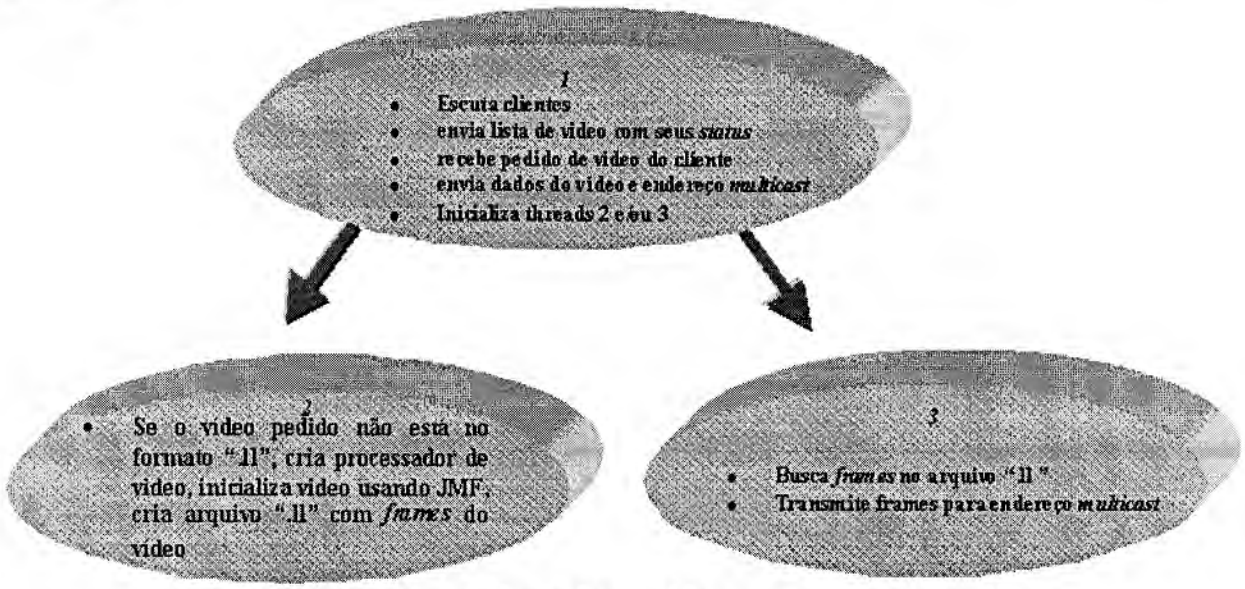


Figura 47: Threads geradas pelo servidor para atender um cliente

Cliente

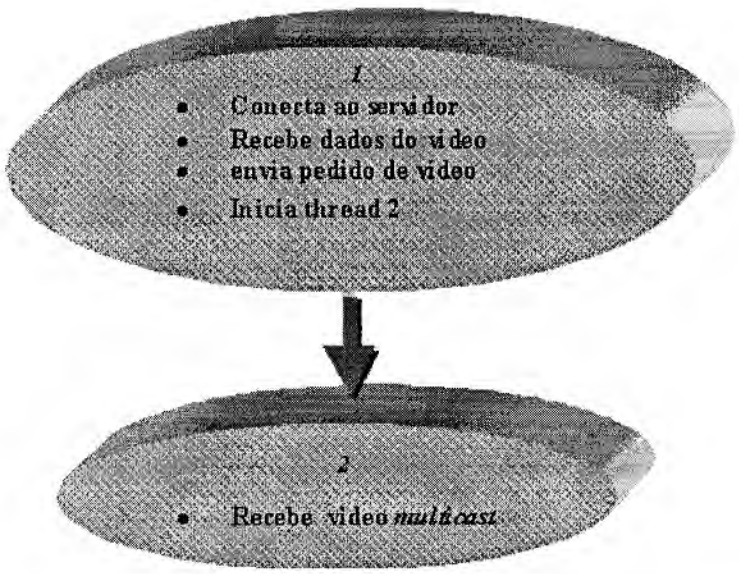


Figura 48: Threads geradas pelo cliente para receber um vídeo

A *thread* principal do cliente inicializa uma tela gráfica com botões que permitem sua conexão ao servidor e a escolha de um vídeo. Implementações futuras podem inserir a apresentação do vídeo juntamente com controles sobre este.

A **Figura 49** ilustra a tela inicial do cliente:

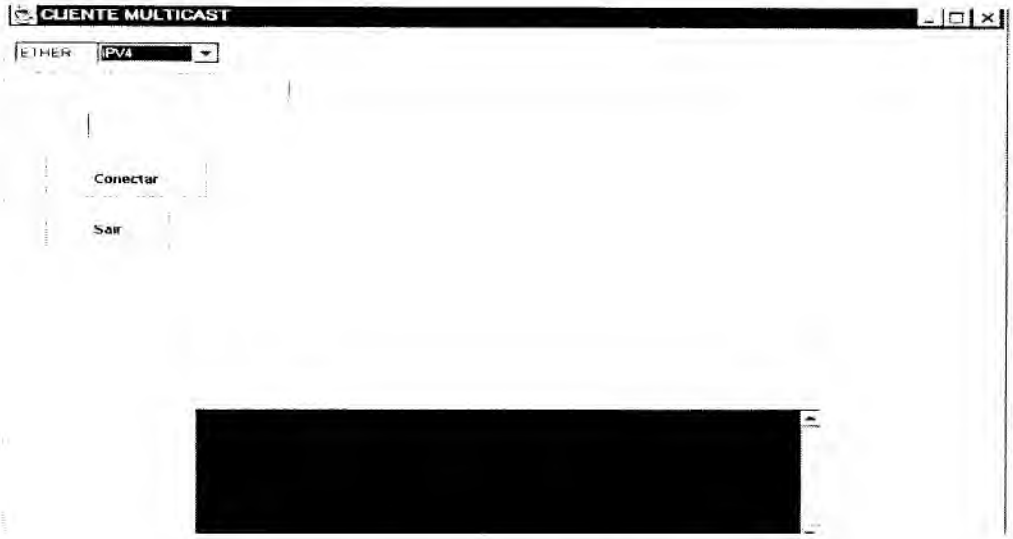


Figura 49: Tela Inicia do Cliente

Após a incialização do cliente, o protocolo a ser utilizado (IPv4 ou IPv6) é selecionado em uma "caixa de escolha" e o endereço e porta do servidor são inseridos nas "caixas de texto", para que o cliente possa criar *sockets* nos endereços escolhidos.

Durante o processo de transmissão do vídeo, uma *thread* de controle com uma conexão TCP é mantida, possibilitando assim uma ativação de controle (REWIND, por exemplo) sobre o vídeo pelo cliente. Apenas uma *thread* e uma conexão UDP é criada e mantida na transmissão de vídeo usando *multicasting*. A **Figura 50** ilustra as *threads* das conexões realizadas, representadas por círculos. A *thread* azul diz respeito à conexão TCP e a alaranjada à conexão UDP *multicast*:

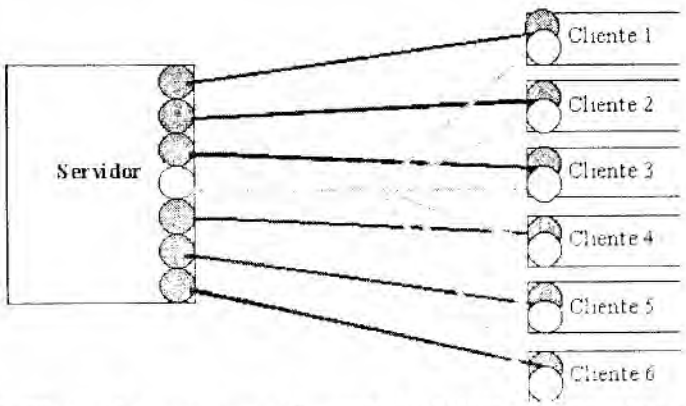


Figura 50: Threads com conexões TCP e UDP dos clientes e do servidor

Como a *thread* com conexão UDP é a que consome maiores recursos do servidor, por realizar acesso à disco a cada *frame* a ser enviado na rede, o uso de *multicasting* torna o servidor mais eficiente, por apenas utilizar uma *thread* para uma grande quantidade de clientes.

A **Figura 51** ilustra todo o processo envolvido na transmissão de vídeo através de *multicasting*.

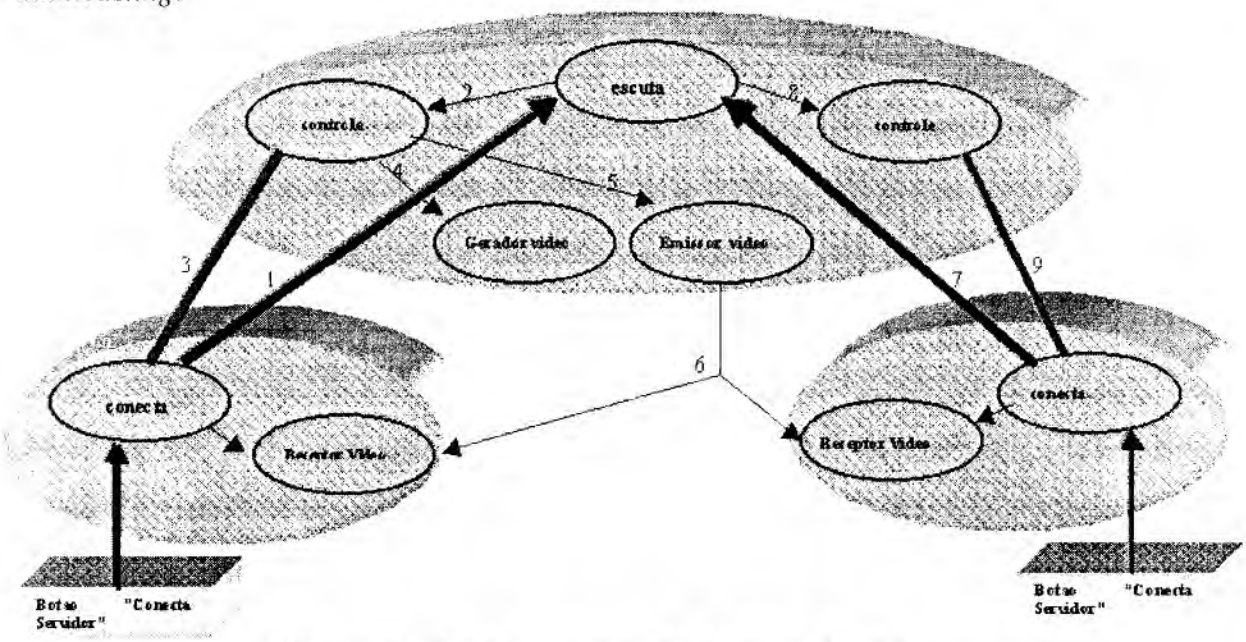


Figura 51: Visão completa de Threads do sistema

1. cliente 1 conecta-se ao servidor;
2. servidor cria uma *thread* de controle, que poderá receber comandos do cliente durante a execução do vídeo;
3. a *thread* de controle cria uma conexão TCP com o cliente. A linha vermelha representa o *socket* TCP criado. É através desse *socket* que: o servidor enviará os filmes e seus respectivos *status*; que o cliente emite o pedido do filme; que o servidor envia os dados do vídeo (altura, largura, taxa de *frame*, duração, quantidade de *frames*, e outros) e dados de configurações (endereço *multicast*, porta *multicast*, e outros);
4. se o vídeo com formato ".ll" não houver sido ainda criado, o servidor cria uma *thread* que, usando a API JMF, iniciará um processador JMF e executará os estados do processador, como explicado na seção 5.3.1. Essa *thread* criará o arquivo ".ll" como explanado na seção 6.4.1.1;

5. a *thread* de controle inicializa a *thread* que transmitirá o vídeo. Essa *thread* abrirá um *socket multicast* segundo o endereço de grupo relativo ao vídeo que foi pedido e comecará a enviar os datagramas UDP *multicast* na rede;
6. essa *thread* envia os pacotes separadamente de cada *frame*, buscados no arquivo ".ll" criado. Os datagramas dos *frames* do vídeo são transmitidos na rede por *multicasting*, e todos os clientes que estiverem unidos ao grupo pertencente a este endereço os receberão;
7. cliente 2 se conecta ao servidor;
8. servidor cria uma *thread* de controle, que poderá receber comandos do cliente 2 durante a execução do vídeo;
9. Essa *thread* cria uma conexão TCP com o cliente, semelhante ao cliente 1. No caso, é suposto que este cliente queira receber o mesmo vídeo. Assim, ele envia o mesmo número do filme pedido pelo cliente 1 e, inicializando a *thread* receptora, assim como acontece com o cliente 1, começa a receber o vídeo. A *thread* receptora do vídeo une-se ao grupo pelo endereço *multicast* relativo ao filme pedido. Percebe-se que o servidor não cria outra *thread* geradora de vídeo nem outra *thread* emissora de vídeo.

6.4.1.4 Implementação de um cliente completo

Nos testes realizados neste projeto, o cliente não realiza processamento algum sobre os pacotes recebidos. Porém, um cliente um pouco mais elaborado que o utilizado nos testes também foi implementado. Esse cliente captura os pacotes, analisa-os, remonta o *frame* e o apresenta. Todo este processo é bastante demorado em relação à capacidade de transmissão do servidor. Por isso, o cliente "*dumb*" foi escolhido para os testes, a fim de analizar-se apenas a capacidade de transmissão do servidor, e não da recepção do cliente.

Na implementação do cliente que apresenta o vídeo, a *thread* receptora de pacotes receberá os datagramas relativos aos *frames* do vídeo, e tentará remontar o *frame*, segundo os valores do número do *frame* e do número do pacote (informações adicionais transmitidas juntamente com as informações RGB do *frame* do vídeo). Uma vez remontado, o método de apresentação do *frame* é chamado, e o vídeo começará a ser mostrado no cliente. Supondo que o cliente tenha recebido os datagramas do *frame* 10 na ordem 3, 2, 0, 1 (nesse caso um *frame* está dividido em apenas 4 datagramas), a remontagem se dá como ilustra a **Figura 52**:

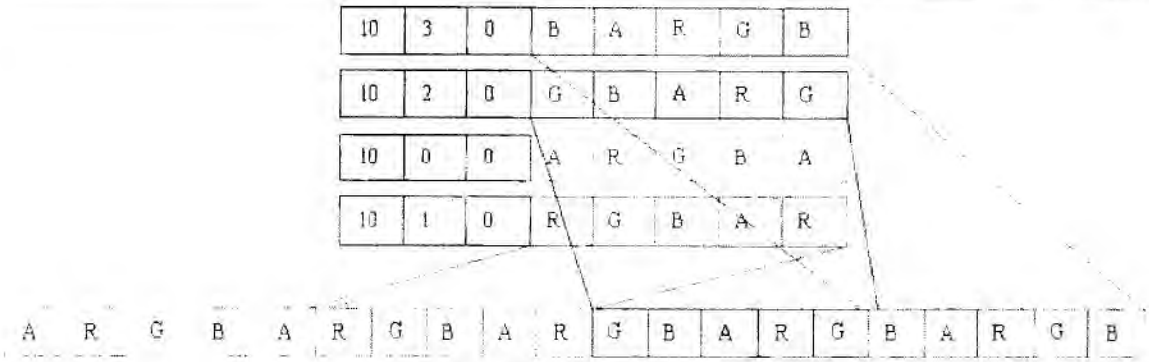


Figura 52: Reconstrução de um *frame*

O campo de *flag*, neste caso, é 0. O valor 0 significa que o cliente não realizou nenhum comando, como REWIND, FORWARD, STOP ou RESUME. Em uma possível implementação de controle sobre o vídeo, tal *flag* pode ser modificada para outro valor conforme a ação realizada sobre o vídeo (essa funcionalidade não foi implementada). Caso um desses comandos acontecesse, a *thread* de controle do servidor receberia o comando e avisaria a *thread* transmissora de vídeo o comando a ser executado.

Uma vez remontado o *array* RGB do *frame*, o cliente pode desenhá-lo na tela, como ilustra a **Figura 53**:

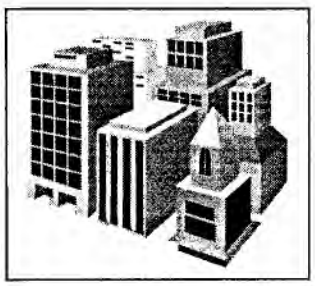


Figura 53: *Frame* reconstruído

6.4.2 Aplicação *Unicast*

Essa versão é muito semelhantre à anterior. A única diferença se dá pelo fato de que agora, para cada cliente conectado, uma conexão UDP *multicast* para a transmissão de vídeo é criada e mantida. Suas funcionalidades são:

Servidor:

a) possibilidade de receber conexão de mais de um cliente;
b) provimento de apenas 1 vídeo;
c) emissão de *frames* de vídeo usando *unicast*;
d) possibilidade de escolha entre endereços IPv4 e IPv6;

Cliente:

a) recepção de datagramas contendo partes dos *frames* do vídeo;
b) conexão com o servidor por *unicast*;
c) possibilidade de escolha entre endereços IPv4 e IPv6;

Nessa versão, portanto, para cada cliente conectado ao servidor, uma nova *thread* transmissora de vídeo é criada. Portanto, não haverá apenas um fluxo de vídeo na rede, como acontece na versão *multicasting*, mas vários fluxos. A **Figura 54** ilustra as conexões na aplicação *unicasting*.

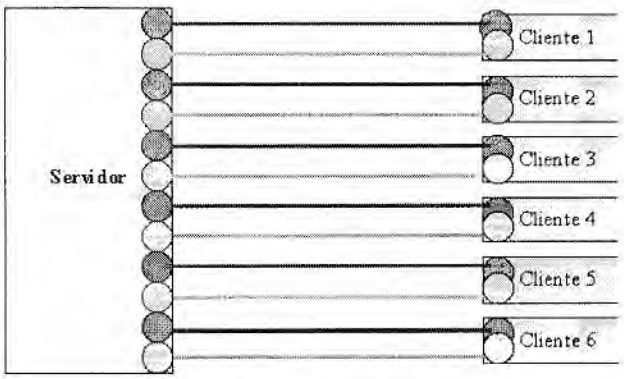


Figura 54: Conexões TCP e UDP *unicasting*

O desempenho do servidor, nesse caso, diminui devido à execução das várias *threads* transmissoras. O tempo de transmissão do vídeo deverá, agora, ser dividido entre os clientes conectados. Uma avaliação desse fato será mostrada com gráficos na seção 6.5.2, e uma comparação com a transmissão utilizando-se *multicasting* nos protocolos IPv4 e IPv6 será discutida.

6.5 Testes realizados

A fim de analisar-se a capacidade da infra-estrutura de redes do ambiente de testes, alguns testes de referência foram realizados sobre a *Ethernet* e sobre a LANE. Após tais testes de referência, testes sobre a aplicação servidora foram feitos, a fim de se verificarem as diferenças na utilização de *unicasting* e de *multicasting* e dos protocolos IPv4 e IPv6 com o uso do JIPSY.

A **Tabela 2** mostra os testes realizados sobre a aplicação servidora:

	MULTICAST	
JAVA (IPv4)	**1 cliente**	**A1**
	2 clientes	**A2**
JIPSY (IPv6)	**1 cliente**	**B1**
	2 clientes	**B2**
	UNICAST	
JAVA (IPv4)	**1 cliente**	**C1**
	2 clientes	**C2**
JIPSY (IPv6)	**1 cliente**	**D1**
	2 clientes	**D2**

Tabela 2: Testes realizados sobre as aplicações *unicast* e *multicast*

As seções seguintes terão como objetivo a discussão das medidas obtidas com esses testes.

6.5.1 Medidas de referência sobre a infra-estrutura *Ethernet* e LANE

Antes de discutir os resultados adquiridos com a aplicação e o uso de *unicasting* e *multicasting*, serão mostradas algumas medidas conseguidas através do uso do aplicativo NETPERF (*NETwork PERFormance*). Esse aplicativo insere na rede uma enorme quantidade de pacotes durante 10 segundos a fim de medir o *throughput* da rede. Para esses testes foram utilizados pacotes UDP de diversos tamanhos entre cada uma das máquinas clientes e o servidor. Os testes foram feitos sobre os cenários que utilizam *Ethernet* e LANE e as medidas adquiridas foram as seguintes:

a) ***Throughput* da rede *Ethernet* usando UDP entre a máquina cliente C1 e a máquina servidora S**

	2k	4k	8k	16k	32k	64k
UDP send	89,2786	85,0438	82,37	70,0152	86,2562	86,1166

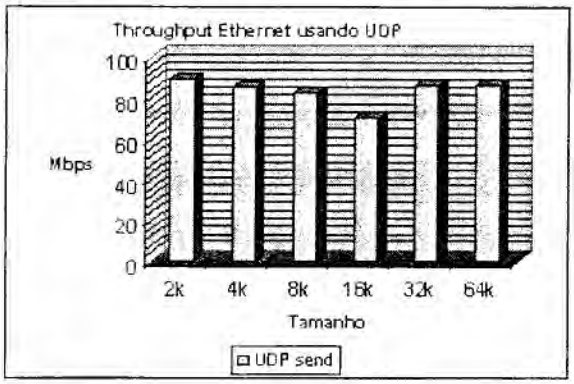


Figura 55: *Throughput Ethernet* entre as máquinas C1 e S

Nesse caso, o maior *throughput* da rede *Ethernet* foi 89,2786 Mbps com pacotes de 2K, e o menor foi 70,0152 Mbps com pacotes de 16K.

b) ***Throughput* da rede *Ethernet* usando UDP entre a máquina C2 e S**

	2k	4k	8k	16k	32k	64k
UDP send	89,4446	90,882	90,065	90,2704	89,8866	89,8624

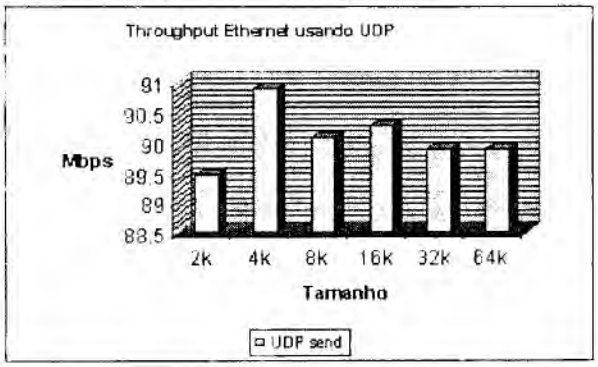


Figura 56: *Throughput Ethernet* entre a máquina C2 e S

Nesse teste, o maior valor foi de 90,882 Mbps com pacotes de 4K e o menor valor foi 89,446 Mbps, com pacotes de 2K.

c) ***Throughput*** **da rede ATM (LANE) usando UDP entre a máquina C1 e S**

	2k	4k	8k	16k	32k	64k
UDP send	86,827	101,7432	118,2394	121,8774	121,913	121,9532

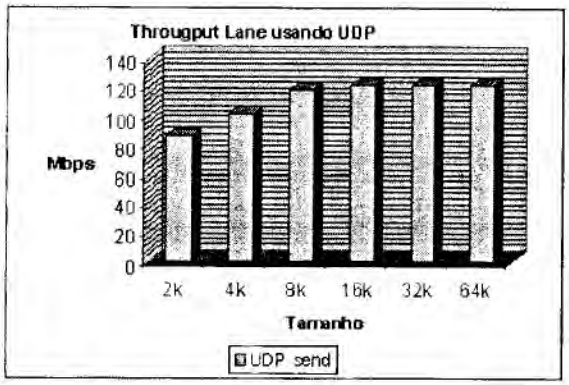


Figura 57: ***Throughput*** **LANE entre a máquina C1 e S**

Nesse caso, o maior *throughput* da rede ATM usando LANE foi 121,9532 Mbps com pacotes de 64K, e o menor foi 86,827 Mbps com pacotes de 2K.

d) ***Throughput*** **da rede ATM (LANE) UDP entre a máquina C2 e S**

	2k	4k	8k	16k	32k	64k
UDP send	119	119,3694	119,9853	120,5506	121,9868	121,534

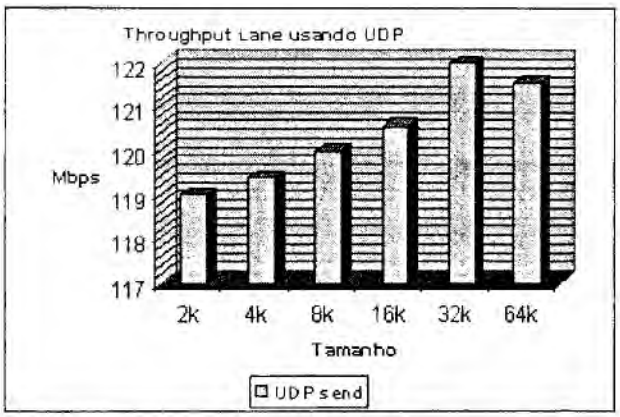


Figura 58: ***Throughput*** **LANE entre a máquina C2 e S**

Nesse teste, o menor valor foi de 119,0 Mbps com pacotes de 2K e o maior valor foi 121,9868 Mbps, com pacotes de 32K.

Tais medidas serviram para verificar o *throughput* das redes *Ethernet* e ATM usando LANE com pacotes UDP de tamanhos diversificados. Através da análise dos valores e dos gráficos,

os valores máximos de *throughput* da rede variam de acordo com o tamanho do pacote, sendo difícil definir qual é o tamanho ótimo dos datagramas para se utilizar na aplicação transmissora de vídeo.

Percebeu-se que não há grandes diferenças entre os valores de *throughput* usando pacotes de 8K, 16K e 32K. Sendo assim, resolveu-se utilizar pacotes de no máximo 16K para o *payload* dos datagramas da aplicação, por ser um tamanho intermediário e não causar tantas divisões em nível de aplicação nos pacotes. Uma fragmentação será necessária no nível de *datalink*, já que o MTU do *Ethernet* e LANE é de 1500 bytes. Tal fragmentação é realizada em hardware, possuindo um desempenho superior quando comparado à divisão feita na aplicação.

6.5.2 Medidas sobre a aplicação servidora usando *Unicasting* e *Multicasting* e os protocolos IPv4 e IPv6

Para os testes com a aplicação sobre o cenário *Ethernet*, foi utilizado o aplicativo IPTRAF (*IP Traffic*), cuja função é capturar pacotes da rede usando a biblioteca de captura de pacotes do sistema (*libpcap*, no caso do Linux) e realizar uma estatística no número de *frames Ethernet* que estão ocupando a rede, além de medir o *throughput* da rede e o número de pacotes IP e não IP.

A transmissão do vídeo foi feita 10 vezes, ou seja, foram transmitidos na rede 9990 *frames* em um tempo que varia de acordo com o protocolo utilizado, com a técnica (*unicasting* ou *multicasting*) e até mesmo com o número de clientes conectados. Após adquiridos os valores, fez-se uma média e gerou-se um gráfico. Os tópicos seguintes ilustrarão as medidas tomadas, de acordo com a **Tabela 2**.

6.5.2.1 *Throughput* da rede e Tempo de Transmissão do Vídeo usando a Aplicação Multicast (A1, B1, A2, B2)

	IPv4 Multicast com 1 cliente	IPv6 Multicast com 1 cliente	IPv4 Multicast com 2 clientes	IPv6 Multicast com 2 clientes
Throughput Médio na Rede (Mbps)	44,667605	52,963043	46,819241	51,109277
Tempo de Transmissão do Vídeo (segundos)	30,5283	16,817	30,8203	17,0221

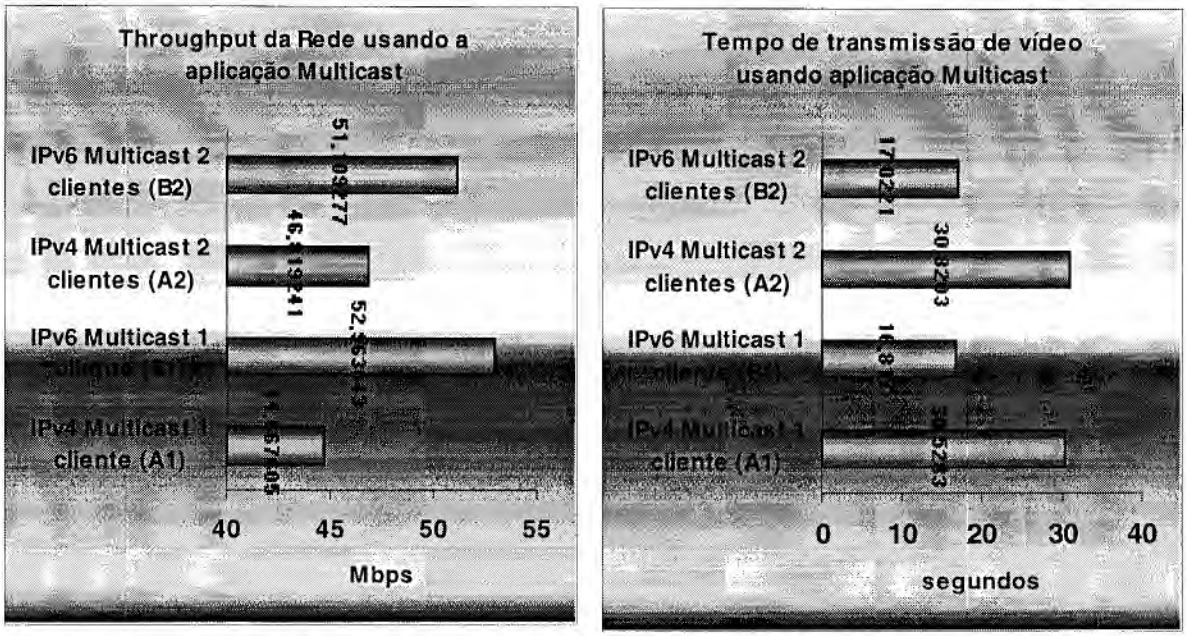


Figura 59: *Throughput* da rede e Tempo de Transmissão do Vídeo usando a Aplicação Multicast

Esses gráficos foram traçados após o cálculo de uma média dos valores gerados pela aplicativo IPTRAF. Percebe-se, com esses gráficos, que a capacidade de transmissão do servidor utilizando a aplicação *multicast* é de aproximadamente 47 Mbps para o protocolo IPv4 e de 54 Mbps para o protocolo IPv6. A capacidade de transmissão *multicast* do servidor não depende do número de clientes conectados a ele e apenas de sua arquitetura. Desta forma, o *throughput* da rede fica limitado à capacidade de transmissão da máquina servidora e de seu sistema como um todo (sistema operacional, velocidade de HD, memória, barramento, e outros). O fato da aplicação utilizar uma largura de banda maior com o protocolo IPv6 faz concluir que sua implementação e utilização pelo sistema Linux é mais eficiente em relação ao IPv4.

O tempo de transmissão do vídeo é proporcional ao *throughput* calculado (quanto maior o *throughput*, menor o tempo de transmissão). O tempo de transmissão com o protocolo IPv6 é, em média, 17 segundos, o que equivale à transmissão de aproximadamente 59 frames por segundo (999/17). Esse valor é muito maior que a taxa do vídeo original, que é de 15 frames por segundo. Sendo assim, para que o vídeo seja apresentado a essa taxa no cliente, ou pode-se atrasar o vídeo no servidor ou pode-se bufferizar o vídeo no cliente. Já o tempo de

transmissão com o protocolo IPv4 é, em média, de 30 segundos, o que equivale a aproximadamente 33 frames por segundo (999/30).

Um fato muito importante a se observar com esses testes é que o tempo de transmissão do vídeo não mudou com 2 clientes conectados. Isto se deve ao fato de que a aplicação *multicast* utiliza apenas uma *thread* de transmissão de vídeo e, independente do número de clientes conectados, o tempo sempre será o mesmo (em média). Percebe-se, dessa forma, que para aplicações em que o mesmo vídeo é transmitido ao mesmo tempo para uma série de clientes, o uso de *multicasting* é importante.

6.5.2.2 *Throughput* da rede e Tempo de Transmissão do Vídeo usando a Aplicação Unicast (C1, D1, C2, D2)

	IPv4 Unicast com 1 cliente	IPv6 Unicast com 1 cliente	IPv4 Unicast com 2 clientes	IPv6 Unicast com 2 clientes
Throughput Médio na Rede (Mbps)	49,94947917	60,04417969	51,04978013	58,89220313
Tempo de Transmissão do Vídeo (segundos)	29,421	16,5426	68,1963	33,93825

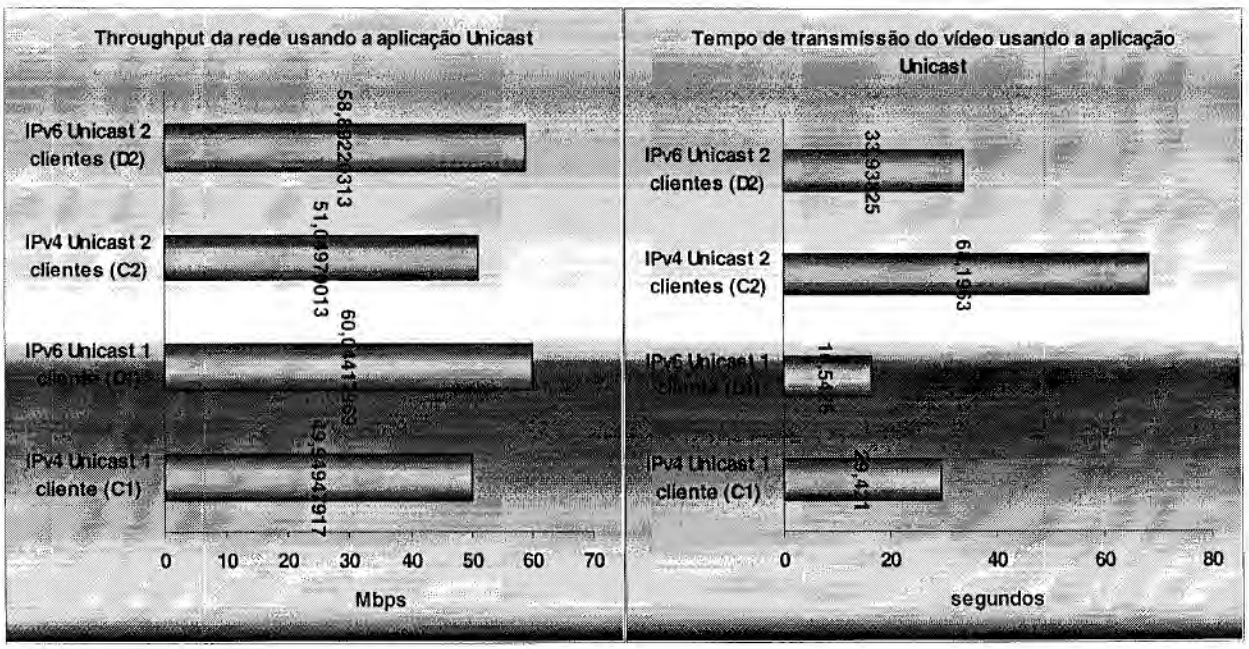


Figura 60: *Throughput* da rede e Tempo de Transmissão do Vídeo usando a Aplicação Unicast

Nos testes com a aplicação *unicast* obtiveram-se os dados ilustrados na **Figura 60**, em que o *throughput* da rede ficou em torno de 50 Mbps com o protocolo IPv4 e 59 Mbps com o

protocolo IPv6. Novamente percebe-se uma melhor eficiência no uso do protocolo IPv6 e os valores, quando comparados aos da aplicação *multicast*, são maiores, concluindo-se que o uso de *sockets unicast* permite o envio de um número maior de pacotes por segundo.

Porém, o uso de *unicasting* para aplicações de vídeo com mais de 1 cliente conectado não é interessante. Isto pode ser percebido com os valores do gráfico de tempo de transmissão do vídeo, em que com apenas 1 cliente, o tempo médio foi de 29, 421 segundos (33 fps, aproximadamente) e com 2 clientes o tempo foi de 68, 1963 segundos (15 fps, aproximadamente), usando o protocolo IPv4. Desconsiderando-se que o *kernel* do sistema pode variar o tempo de transmissão, pode-se dizer que o tempo de transmissão para cada cliente praticamente dobrou e, conseqüentemente a taxa de *frames* por segundo caiu para a metade por cliente. Caso um terceiro cliente se conectasse, o tempo supostamente triplicaria, em média, e a taxa de frames por segundo seria 1/3 da original.

Mais uma vez percebe-se que o tempo usando-se o protocolo IPv6 é menor, sendo de aproximadamente 16,5s (59 fps, aproximadamente) para um cliente e 33s (30 fps, aproximadamente) para 2 clientes.

6.5.3 Considerações Finais

Os principais objetivos do desenvolvimento desse projeto foram a experimentação do novo protocolo IPv6 e da técnica de *multicasting* e a comparação de medidas entre este protocolo com o protocolo IPv4, além da comparação das técnicas *multicasting* e *unicasting*.

Uma problemática muito grande surge no desenvolvimento de aplicações de transmissão de vídeo, tais como a largura de banda da rede e a eficiência do servidor, e o uso de *unicasting* e *multicasting* proporcionam diferenças quantitativas que podem afetar o desempenho de aplicações multimídia, como visto nos resultados obtidos. *Multicasting*, neste caso, é mais propício a este tipo de aplicações.

No capítulo 7 serão abordadas as conclusões sobre o trabalho desenvolvido, sobre os resultados obtidos, as contribuições do projeto e possíveis trabalhos futuros.

Capítulo 7 Conclusões e Trabalhos Futuros

Aplicações multimídia (e em particular as de vídeo) estão se tornando bastante comuns atualmente, e protocolos e técnicas que proporcionam maior qualidade, escalabilidade, economia de largura de banda e de recursos são muito importantes.

O protocolo IPv6 foi foco de estudo e experimentos desse projeto, já que proporciona um maior desempenho e benefícios quando comparados ao protocolo IPv4, além de possibilitar uma grande escalabilidade, devido ao número de bits utilizados para o endereçamento de interfaces. A utilização desse protocolo proporcionará o endereçamento de um número muito grande de interfaces, e este projeto possibilita o conhecimento de algumas tecnologias, como o JIPSY, que poderão ser utilizadas para o desenvolvimento de aplicações com suporte a este novo protocolo.

O *multicasting* beneficia tanto LANs quanto WANs, quando um número relativamente significativo de clientes se conecta simultâneamente ao servidor requisitando a mesma porção do vídeo. Caso vários servidores estejam transmitindo ao mesmo tempo, existe um ganho ainda maior, tanto em recursos e desempenho do servidor, quanto em largura de banda utilizada na rede.

Alguns benefícios que o IPv6 e *multicasting* proporcionam foram comprovados neste projeto por meio de medidas comparativas realizadas com os protocolos IPv4 e IPv6 e endereçamentos *multicast* e *unicast* usando JAVA e JIPSY, com um e dois clientes conectados a um servidor. Com os resultados obtidos, ilustrados nas **Figuras 55** e **56**, pôde-se verificar uma melhora de 18, 57% na eficiência (*throughput* da rede) de transmissão IPv6 *multicasting* em relação ao IPv4 *multicasting* para apenas 1 cliente, e uma melhora de 9,16% para 2 clientes. Em relação à eficiência da transmissão *unicast*, houve uma melhora de 20,21% do IPv6 em relação ao IPv4 com 1 cliente, e de 15, 37% com 2 clientes.

Além dos dados de *throughput* da rede, observou-se uma diminuição no tempo de transmissão do vídeo no uso do protocolo IPv6. Quando considera-se apenas 1 cliente conectado ao servidor e transmissão *multicast*, o tempo de transmissão do IPv6 é 44, 92% inferior ao IPv4, e com 2 clientes 44, 78%. Considerando-se transmissão *unicast*, IPv6 é 43,77% inferior com 1 cliente e 50, 23% com 2 clientes conectados.

Por fim, um dado muito importante obtido foi a diferença de tempo de transmissão do vídeo e *throughput* da rede usando-se *unicasting* e *multicasting*. Considerando 2 clientes conectados, percebeu-se que o *throughput* da rede foi 9,03% maior usando-se IPv4 *unicasting* em relação ao IPv4 *multicasting*, e 15, 22% maior usando-se IPv6 *unicasting* em relação ao IPv6 *multicasting*. O tempo de transmissão com 2 clientes conectados foi 54, 8% maior com IPv4 *unicasting* em relação ao IPv4 *multicasting* , e 49, 84% maior com o IPv4 *unicasting* em relação ao IPv4 *multicasting*.

Estes resultados confirmam a melhoria de eficiência na transmissão de dados quando *multicasting* e IPv6 são utilizados, o que torna seus usos em aplicações multimídia em LANs e WANs uma solução bastante interessante.

7.1 Contribuições do projeto

As principais contribuições deste trabalho foram as medidas comparativas abordadas anteriormente. Além disso, o desenvolvimento deste projeto proporcionou diversos benefícios tanto para o grupo InterMídia e para o ICMC - USP, quanto para pesquisadores e implementadores de tecnologias de rede. Como diversas pesquisas do grupo InterMídia têm como foco tecnologias de vídeo, que vão desde implementações de servidores de vídeo até modelagem e estudos de aplicações que utilizam padrões como MPEG-2, 4 e 7, a implementação deste servidor de vídeo poderá ser um início para implementações reais e testes usando tais padrões.

Além disto, como este projeto se baseou no IPv6, e devido a este prever técnicas de segurança mais bem elaboradas do que as existentes, tal protocolo poderá ser foco de estudos, de

implementações e testes para o grupo de segurança do InterMídia e outros pesquisadores de outras instituições.

Para implementadores e pesquisadores de tecnologias de rede, este projeto proporciona um benefício no que diz respeito aos estudos e testes realizados com o LINUX, JAVA, JIPSY e, conseqüentemente, JNI. Tais tecnologias são distribuídas gratuitamente na Internet, o que facilita ainda mais o desenvolvimento e pesquisa por diversas instituições. Além disto, tecnologias como LINUX e JAVA já são focos de diversas implementações no mundo da Internet, e o uso do JIPSY e JNI são tecnologias muito poderosas que podem gerar implementações e pesquisas em áreas diversas.

Este trabalho infere três focos diferentes de pesquisa: cliente, servidor e infra-estrutura de rede. Sendo assim, medidas e melhorias de eficiência nestes três focos são interessantes, visto que o desempenho de um sistema multimídia não depende apenas de um destes focos individualmente, e sim da interação entre cada um.

A revisão bibliográfica desta dissertação provê referências importantes para todos que se interessem por esta área. Além dessas contribuições, propostas interessantes para continuidade deste projeto surgiram no decorrer de seu desenvolvimento. Tais propostas serão abordadas na seção seguinte.

7.2 Propostas de trabalhos futuros

No decorrer dos estudos e pesquisas das tecnologias emergentes e trabalhos nesta área, e no desenvolvimento deste projeto, muitas idéias surgiram para trabalhos futuros. Essas propostas são enumeradas em diversos itens, como infra-estrutura e implementação.

7.2.1 Infra-estrutura

O laboratório InterMídia utiliza como infra-estrutura de rede um *hub Ethernet* 10/100 Mbps que foi utilizado neste projeto, além de uma *switch Ethernet* 10/100 Mbps que está

interconectada com todo o instituto ICMC e uma *switch* ATM, que está interconectada com outra *switch* no instituto de física (IFSC), fazendo parte do projeto ReMAV explanado na introdução desta dissertação. Assim, os seguintes trabalhos são propostos:

- testes sobre LANE com a aplicação *unicasting* e *multicasting* desenvolvida e uma comparação com os resultados obtidos;
- utilizar a infra-estrutura ATM que interliga o ICMC com o IFSC e realizar testes de *throughput* de rede dentro de uma LANE criada entre os dois departamentos;
- configurar máquinas com IPv6 sobre ATM (LANE) no IFSC e testar o servidor implementado, transmitindo-se vídeos nessa infra-estrutura;
- configurar e testar alguma implementação de MARS para Linux (ou outro sistema) com os protocolos IPv4 e IPv6 e realizar medidas com a aplicação desenvolvida, comparando os resultados com o LANE;

Testes similares poderão ser feitos por quaisquer instituições com uma infra-estrutura adequada. Além de testes com equipamentos convencionais, diversos trabalhos surgem dentro do contexto desse projeto quando equipamentos atuais de rede (com suporte à QoS) são considerados:

- configurar equipamentos que suportem QoS e comparar a trasmissão de vídeo com pacotes prioritários com a transmissão de vídeo sem prioridade nos pacotes. Tais equipamentos estão sendo atualmente bastante utilizados no mercado, e geralmente suportam padrões como 802.1p/Q (padrões para priorização de pacotes e criação de redes virtuais para isolamento de tráfego), protocolo RSVP (*ReSerVation Protocol* – um protocolo que provê reserva de recursos de rede e sinalização fim-a-fim para qualidade de serviço);
- configurar o sistema Linux para a utilização de QoS, através da marcação de pacotes;
- estudos da utilização do IPv6 e da integração das tecnologias de QoS, incluindo DiffServ (*Differentiated Services*) e IntServ (*Integrated Services*), que são serviços de QoS que também estão incluídos em novos equipamentos de rede;

7.2.2 Implementação

Em relação à implementação da aplicação, algumas modificações e melhoramentos poderão ser feitos em trabalhos futuros, tais como:

- Implementação de um cliente completo, que possa realizar comandos sobre o vídeo, tais como REWIND, FORWARD, PAUSE e RESUME, utilizando bufferização de pacotes. A bufferização será muito importante, já que a capacidade de envio de pacotes do servidor é maior do que a capacidade de recebimento e processamento de pacotes (remontagem de frames e exibição) pelo clientes. A **Figura 68** ilustra esta idéia:

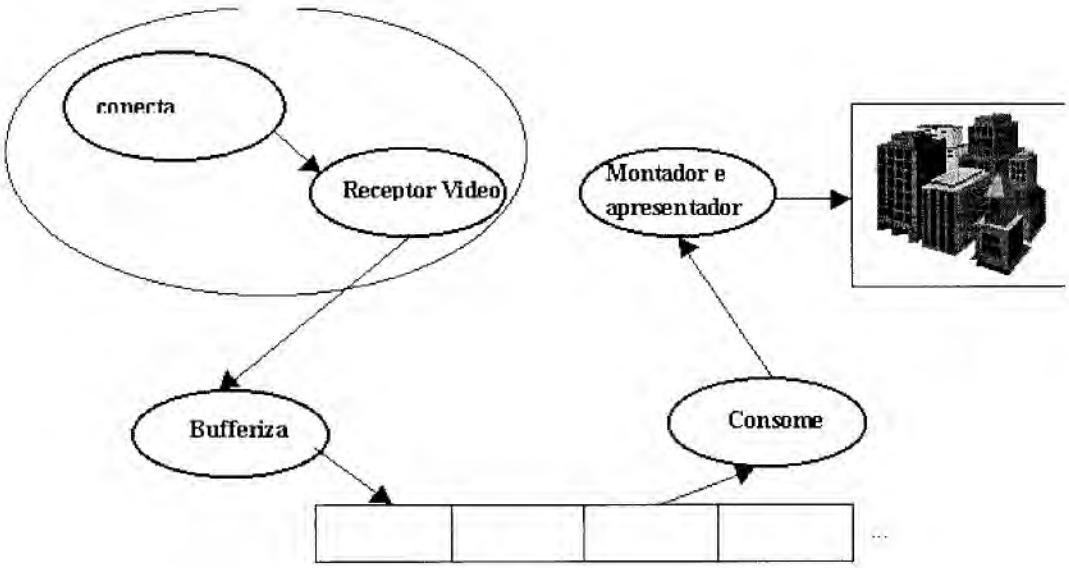


Figura 36: Bufferização no cliente

Neste modelo, além das *threads* de conexão e de recepção de pacotes, haveria uma *thread* responsável pela bufferização dos pacotes recebidos (produtor), uma *thread* para consumo destes pacotes e uma outra para a montagem e apresentação dos *frames* do vídeo;

- usar o JNI para melhorar a eficiência do cliente. Existem partes do código do cliente que demoram muito tempo a serem processadas pelo JAVA. Assim, a implementação em C destas porções de código e sua integração no código JAVA com o JNI poderão melhorar consideravelmente o desempenho do cliente;
- modificar o JIPSY para suportar ATM. O pacote de ATM para Linux possui códigos escritos em C para a criação de conexões ATM e para encapsulamento de IP sobre ATM.

Como o JIPSY utiliza o JNI para interfacear os códigos C do *kernel* para a manipulação de *sockets* que utilizam endereçamento IPv6, uma extensão ao JIPSY utilizando a API ATM seria extremamente interessante. Assim, a aplicação poderia utilizar IPv6 sobre ATM e ATM nativo, e uma comparação de desempenho poderia ser feita.

Referências Bibliográficas

(Almesberger, 1996) ALMESBERGER, W., *ATM on LINUX*, 11 de março de 1996, LRC (*laboratoire de Réseaux de Communication*), Lausane, Switzerland

(Almesberger, 2000) ALMESBERGER, W., *ATM on Linux*, janeiro de 2001, site http://lrcwww.epfl.ch/linux-atm/,

(Andrade, 2001) ANDRADE, L.A., MOREIRA, E.S., *Transmissão Multicasting de Vídeo Comprimido*, artigo submetido para o Simpósio Brasileiro de Redes de Computadores (SBRC), 07 de fevereiro de 2001

(Armitage, 1996) *ARMITAGE, G., Support for Multicast over UNI 3.0/3.1 based ATM Networks*, RFC 2022, novembro de 1996, disponível em http://www.graphcomp.com/info/rfc/rfc2022.html

(Bieringer, 2000) BIERINGER, P., *Linux:*IPv6, disponível em http://www.bieringer.de/linux/ IPv6/, 11 de novembro de 2000;

(Braden, 1997) BRADEN, B., ESTRIN, D. et al, *Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification*, RFC 2205; Setembro de 1997, disponível em http://www.ietf.org/rfc/rfc2205.txt

(Bradner, 1996) BRADNER, S.O.; MANKIN, A. *Ipng, Internet Protocol Next Generation.* Massachusetts:Addison-Wesley, 1996

(Cisco, 1999) CISCO SYSTEM, *Asynchronous Transfer Mode (ATM) Switching*, disponível em http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/atm.htm, 9 de dezembro de 1999

(Comer, 1995) COMER, D.E. *Internetworking with TCP/IP* – Principles, Protocols and Architecture, New Jersey:Prentice Hall, v.I, 3ª edição, 1995

(Cornell, 1998) CORNELL, G.,HORSTMANN, C.S., *Core JAVA – guia autorizado Sunsoft Press*, Makron Books, 1998;

(Crawford, 1996) CRAWFORD, M., *A Method for the Transmission of IPv6 Packets over Ethernet Networks*, RFC 1972, agosto de 1996, disponível em http://www.graphcomp.com/info/rfc/rfc1972.html

(Deering, 1989) DEERING, S., *Host Extensions for IP Multicasting*, RFC 1112, disponível em http://sunsite.hr/cgi-bin/nfc/rfc1112.txt, 9 de dezembro de 1999

(Elz, 1996) ELZ, R., *A Compact Representation of IPv6 Addresses*, 1 de abril de 1996, RFC 1924

(Eriksson, 1994) ERIKSSON, H. *MBONE:*The *Multicast Backbone*, *Communications of the ACM*, V. 37, n.8, p.54-60, 1994

(Fenner, 1999) FENNER, W., DEERING, S. and HABERMAN, B. , *Multicast Listener Discovery (MLD) for IPv6*, RFC 2710, novembro de 1999, disponível em http://www.kblabs.com/lab/lib/rfcs/2700/rfc2710.txt.html

(Fenner, 2000) FENNER, B. HANDLEY, M.,HOLBROOK, H. et al., *Protocol Independent Multicast – Sparse Mode (PIM-SM): Protocol Specification (Revised)*, Internet Draft , 24 de novembro de 2000, disponível em http://www.ietf.org/internet-drafts/draft-ietf-pim-sm-v2-new-01.txt

(Flanagan, 2000a) FLANAGAN, M., *JIPSY:IPv6 for JAVA*, disponível em http://www.progsoc.uts.edu.au/~mpf/JIPSY/, 11 de novembro de 2000;

(Flanagan, 2000b) FLANAGAN, M, *Jipsy: IPv6 For JAVA – Summary* , disponível em http://sourceforge.net/projects/jipsy/

(Fuller, 1993) FULLER,V., LI,T. et al, *"Classless Inter-Domain Routing (CIDR): An Address Assignment and Aggregation Strategy"*, RFC 1519, setembro de 1993

(Giordano, 1997) GIORDANO, S., SDHMID, R.M., et al, *IP and ATM – a position paper*, outubro de 1997, Purdue University, USA

(Goularte, 2000) GOULARTE, R., *Sobre a implementação e avaliação de técnicas de distribuição de vídeo interativo baseadas em conteúdo*, proposta de projeto de mestrado, 2000, ICMC-USP

(Heinanen, 1993) HEINANEN, J., *Multiprotocol Encapsulation over ATM Adaptation Layer 5*, RFC 1483, julho de 1993, disponível em http://www.graphcomp.com/info/rfc/rfc1483.html

(Hinden, 1998a) HINDEN, R., DEERING, S., *Ipv6 Addressing Architecture*, Julho de 1998, RFC 2373, disponível em http://www.ietf.rnp.br/ftp/rfc/rfc2373.txt

(Hinden, 1998b) HINDEN, R., DEERING, S., *An IPv6 Aggregatable Global Unicast Address Format"*, Julho de 1998, RFC 2374

(IETF, 2000a) INTERNET ENGINNERING TASK FORCE, *Differenciated Services - DiffServ*, disponível em http://www.ietf.org/html.charters/diffserv-charter.html, 11 de novembro de 2000;

(IETF, 2000b) INTERNET ENGINNERING TASK FORCE, *Integrated Services – IntServ*,

disponível em http://www.ietf.org/html.charters/intserv-charter.html, 11 de novembro de 2000;

(Johnson, 1997a) JOHNSON, V., JOHNSON, M., *How IP Multicast Works - A technical overview of IP Multicast concepts, addressing, group management and approaches to routing*, outubro de 1997, disponível em http://www.ipmulticast.com/community/whitepapers/howipmcworks.html

(Johnson, 1997b) JOHNSON, V., JOHNSON, M., *Introduction to IP Multicast Routing - A technical overview of IP Multicast routing protocols and their features*, outubro de 1997, disponível em http://www.ipmulticast.com/community/whitepapers/introrouting.html

(Katz, 1993) KATZ, D., *Transmission of IP and ARP over FDDI Networks*, RFC 1390, 1993 disponível em http://www.graphcomp.com/info/rfc/rfc1390.html

(Kosieur, 2000) KOSIEUR, D., *Network Stratergy Overview – IP Multicasting*, disponível em http://www.cisco.com/warp/public/cc/techno/protocol/ipmu/tech/ipmc_wp.htm, 23 de novembro de 2000

(Laubach, 1994) LAUBACH, M., *Classical IP and ARP over ATM*, RFC 1577, janeiro de 1994, disponível em http://www.graphcomp.com/info/rfc/rfc1577.html

(Loshin, 1999) LOSHIN, P, IPv6, Clearly Explained. São Francisco: Morgan Kaufmann, 1999

(Marko, 1996) MARKO, K., *Implementation of LAN Emulation over ATM in Linux*, 21 de agosto de 1996, tese de mestrado, Tempere University of Technology, Finlandia;

(Micheli, 2000) MICHELI, M.P., COSTA, H.D. et al., *IP Multicast sobre ATM, 18° Simpósio Brasileiro de Redes de Computadores (SBRC), II Workshop RNP2*, Belo Horizonte, 23 e 24 de maio de 2000, p.139 a 150

(MSR, 2000) MICROSOFT RESEARCH IPV6, *Internet Protocol Version 6*, disponível em http:// www.research.microsoft.com/msrIPv6/, 11 de novembro de 2000;

(Oosthoek, 1997) ***OOSTHOEK, S,*** *Survey of Multicast Support for IP over ATM for Implementation Purposes*, 4 de Julho de 1997, Tese de mestrado, University of Twente, disponível em http://margo.student.utwente.nl/simon/finished/thesis/thesis1/thesis1.htm

(Pissioli, 1999) PISSIOLI, M. *Sistema Portátil para transmissão de vídeo-on-demand*, São Carlos: ICMC - USP, 1999. Dissertação de Mestrado

(Pusateri, 1993) PUSATERI, T. *IP Multicast over Token-Ring Local Area Networks*, RFC 1469, junho de 1993,disponível em http://www.graphcomp.com/info/rfc/rfc1469.html

(Rekhter, 1993) REKHTER, Y., LI,T. et al, *"An architecture for IP Address Allocation with CIDR,* RFC 1518, setembro de 1993

(Savetz, 1999) SAVETZ, K., RANDALL, N., LEPAGE, Y. *MBONE: Multicasting Tomorrow's Internet*, disponível em <http://www.savetz.com/mbone/>, 9 de dezembro de 1999

(Silva, 2000) SILVA, D.C., *Projeto de Sistema de Sincronização de Streams de Vídeo para Serviço Multicast,*São Carlos: ICMC - USP, 2000. Dissertação

(Stardust, 1999) STARDUST.COM, *QoS protocols and Architectures,* White Paper, julho de 1999, disponível em http://www.qosforum.com

(Stevens, 1998) STEVENS, W., *Unix Network Programming - Networking APIs: Sockets and XTI*, vol 1, segunda edição, Prentice Hall, 1998

(Sun, 1999) SUN MICROSYSTEMS, *Java Media Framework API Guide*, 1998-1999 disponível em http://java.sun.com/products/JAVA-media/jmf/index.html

(Sun, 2000a) SUN MICROSYSTEMS, *The JAVA Tutorial – A pratical guide for programmers*, disponível em http://JAVA.sun.com/tutorial, 11 de novembro de 2000

(Sun, 2000b) SUN MICROSYSTEMS, Lesson: Overview of the JNI, 2001, disponível em http://JAVA.sun.com/docs/books/tutorial/native1.1/concepts/index.html

(Talpade, 1997) TALPADE, R., *Multicast Server Architectures for MARS-based ATM multicasting*, RFC 2149, maio de 1997, disponível em http://www.graphcomp.com/info/rfc/rfc2149.html

(Tanembaum, 1996) TANEMBAUM, A.S. *Computer Networks*, New Jersey:Prentice-Hall, 3ª edição, 1996

(Thakrar, 1996) THAKRAR, K. , ANDLEIGH, P.K. *Multimedia Systems Design*, Prentice-Hall, New Jersey, 1996

(Werner, 2000) WERNER, A., *ATM on Linux,* disponível em http://lrcwww.epfl.ch/linux-atm/, 11 de novembro de 2000

Apêndice A – Glossário de Siglas

AAL	*ATM Adaptation Layer*
AIFF	*Audio Interchange File Format*
API	*Application Programming Interface*
ARP	*Address Resolution Protocol*
ATM	*Asynchronous Transfer Mode*
ATMARP	*ATM Address Resolution Protocol*
AU	*AUdio para Unix*
BUS	*Broadcast Unknown Server*
CIDR	*Classless InterDomain Routing*
CLIP	*Classical Internet Protocol*
CPCS	*Common Part Convergence Sublayer*
CSMA-CD	*Carrier Sense Multiple Access – Collision Detection*
DC	*Departamento de Computação*
DiffServ	*Differentiated Services*
ELAN	*Emulated LAN*
FDDI	*Fiber Distributed Data Interface*
FPS	*Frames Por Segundo*
Gbps	*Gigabit por segundo*
GSM	*Global System for Mobile communications*
ICA	*Institute for Computer Communication and Applications*
ICMC	*Instituto de Ciências Matemáticas e de Computação*
IEEE	*Institute of Electrical and Electronics Engineers*
IFSC	*Instituto de Física de São Carlos*
IGMP	*Internet Group Management Protocol*

IGMPv1	*Internet Group Management Protocol versão 1*
IGMPv2	*Internet Group Management Protocol versão 2*
IntServ	*Integrated Services*
IOS	*Internetworking Operating System*
IP ou IPv4	*Internet Protocol versão 4*
IPTraff	*Internet Protocol Traffic*
IPv6	*Internet Protocol versão 6*
IPX	*Internetwork Packet Exchange*
ITU-T	*International Telecommunication Union Telecommunication Standardization sector*
JDK	*JAVA Development Kit*
JIPSY	*JAVA IP version Six readY*
JMF	*JAVA Media Framework*
JNI	*JAVA Native Interface*
JVM	*JAVA Virtual Machine*
LAN	*Local Area Network*
LANE	*LAN Emulation*
LEC	*LAN Emulation Client*
LECS	*LAN Emulation Configuration Server*
LES	*LAN Emulation Server*
LEService	*LAN Emulation Service*
LIS	*Logical IP Subnet*
LRC	*Laboratoire de Réseaux de Communication*
LUNI	*LANE User-Network Interface*
MAC	*Medium Access Control*
MARS	*Multicast Adress Resolution Server*
MBONE	*Multicast backBONE*
Mbps	*Megabits por segundo*

MCS	*MultiCast Server*
MIDI	*Music Instrument Digital Interface*
MLD	*Multicast Listener Discovery*
MOSPF	*Multicast Open Shortest Path First*
MPEG	*Motion Picture Expert Group*
MPEG-4	*Motion Picture Expert Group versão 4*
MPEG-7	*Motion Picture Expert Group versão 7*
MROUTER	*Multicast Router*
MSRIPv6	*MicroSoft Research in IPv6*
MTU	*Minimum Transfer Unit*
NetPerf	*Network Performance*
NLA	*Next-Level Aggregation*
NNI	*Network-Network Interface*
NSAP	*Network Service Acess Point*
OSI	*Open Systems Interconnection*
PC	*Personal Computer*
PDU	*Payload Data Unit*
PIM-DM	*Protocol Independent Multicast – Dense Mode*
PIM-SM	*Protocol Independent Multicast – Sparse Mode*
PIMv6	*Protocol Independent Multicast versão 6*
PMD	*Physical Medium-Dependent*
PVC	*Permanent Virtual Circuit*
QoS	*Quality of Service*
RAM	*Ramdom Access Memory*
ReMAV	*Rede Metropolitana de Alta Velocidade*
RFC	*Request For Comments*
RGB	*Red, Green, Blue*
RMF	*Rich Music Format*

RSVP	*ReserVation Protocol*
RTP	*Real Time Protocol*
SLA	*Site Level Aggregation*
SVC	*Switched Virtual Circuit*
TC	*Transmission Convergence*
TCP	*Transmission Control Protocol*
TDM	*Time Division Multiplexing*
TLA	*Top-Level Aggregation*
UDP	*User Datagram Interface*
UFSCAR	*Universidade Federal de São Carlos*
UNI	*User-Network Interface*
UTP	*Unshielded Twisted Pair*
VCC	*Virtual Channel Connection*
VCI	*Virtual Channel Identifier*
VPI	*Virtual Path Identifier*
WAN	*Wide Area Network*
WAV	*WAVe*

Apêndice B – Artigos Submetidos

• 19º Simpósio Brasileiro de Redes de Computadores (SBRC)

Título: **Provimento de Vídeo em Ambiente IPv6 com Multicasting**
Autores: **Luciano Martins, Edson dos Santos Moreira**
Data de submissão**: 07 de março de 2001**
Instituição: **Universidade Federal de Santa Catarina (UFSC)**
Local: **Florianópolis – S.C.**

• VI Simpósio de Teses e Dissertações

VI Workshop de Teses e Dissertações em Andamento

Título: **Transmissão de Vídeo usando IPv6 e Multicasting em Redes de Alto Desempenho**
Autores: **Luciano Martins, Edson dos Santos Moreira**
Data de submissão: **15 de fevereiro de 2001**
Instituição: **Universidade de São Paulo (USP)**
Local: **São Carlos – S.P.**

• V Simpósio de Teses e Dissertações

V Workshop de Teses e Dissertações em Andamento

Título: **Sincronização de Vídeo usando IPv6 e Multicasting em Redes ATM**
Autores: **Luciano Martins, Edson dos Santos Moreira**
Data de apresentação: **12 de maio de 2000**
Instituição: **Universidade de São Paulo (USP)**
Local: **São Carlos – S.P.**