UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

DAYNA MARIA BORTOLUZZI

Utilização de Filtros de Escalamento de Mídia na Interconexão de Duas Redes Heterogêneas.

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação.

Prof. Dr. Roberto Willrich

Florianópolis, dezembro de 1999.

Utilização de Filtros de Escalamento de Mídia na Interconexão de Duas Redes Heterogêneas.

Dayna Maria Bortoluzzi

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação, Área de Concentração Sistemas de Computação, e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

Prof. Dr. Fernando Alvaro Ostuni Gauthier
Coordenador do Curso de Pós Graduação em Ciência da Computação

Banca Examinadora:

Prof. Dr. Roberto Willrich
Orientador

Prof. Dr. Rosvelter João Coelho da Costa

Prof. Dr. João Bosco Mangueira Sobral

Prof. Dr. Murilo Silva de Camargo

Este trabalho é em especial para Heloisa Maia Sobral,
pelo apoio e cuidados dedicados ao Luís.
Aos meus pais e irmãos pelo que sempre me motivaram a estudar.
Ao Marcelo pelos questionamentos e incentivo.
Obrigado Prof. Roberto, Verinha e Valdete
pelos trabalhos prestados aos alunos do CPGCC.

SUMÁRIO

LISTA DE QUADROS E FIGURAS VI
LISTA DE ABREVIATURAS E SIGLAS VII
RESUMO X
ABSTRACT XI
1. Introdução 1
2. Sistemas Multimídia 3
2.1 Definição de Multimídia 3
2.2 Sistemas Multimídia Standalone e Distribuídos 3
2.3 Características das Fontes de Tráfego Multimídia 4
2.4 Requisitos de Rede para Aplicações Multimídia 5
2.4.1 Largura de banda 5
2.4.2 Requisitos de atraso 6
2.4.3 Requisitos de variação de atraso 7
2.4.4 Requisitos de confiabilidade 8
2.4.5 Garantia de Desempenho 9
2.4.6 Uso adequado dos recursos de rede 10
2.4.7 Multicasting (Difusão Seletiva) 11
2.5 Conclusão 12
3. Aplicações Adaptativas 14
3.1 Protocolo RTP 14
3.1.1 Parte de dados do RTP 16
3.2 Parte de controle do RTP (RTCP) 18
3.3 Aplicações Adaptativas 24
3.3.1 Formas de adaptação 25
3.3.2 Elementos de adaptação 26
3.3.3 Tipos de aplicações adaptativas 27
3.4 Escalamento de Mídias 28
3.4.1 Qualidade de Vídeo 29
3.4.2 Elementos Responsáveis pelo Escalamento de Mídias 31
3.5 Conclusão 33
4. Multicast Backbone (MBone) 34
4.1 Como o IP Multicast Trabalha 35
4.1.1 Endereçamento Multicast IP 35
4.1.2 Grupos Multicast 36
4.1.3 Protocolo IGMP 37
4.2 Topologia do MBone 38
4.2.1 Túneis 39
4.3 Aplicações multimídia do MBone 40
4.3.1 Ferramentas de anúncio de sessão 40
4.3.2 Ferramentas de Áudio 41
4.3.3 Ferramentas de Vídeo 42
4.4 Conclusão 46
5. Redes ATM 47
5.1 Arquitetura ATM 48
5.2 Conexões Lógicas ATM 50
5.2.1 Gerenciamento de tráfego 51
5.2.2 Controle de Sinalização 52
5.3 Células ATM 52
5.3.1 Formato do cabeçalho 53

5.4 Categorias de Serviço ATM 55
5.4.1 Serviço em tempo real 55
5.4.2 Serviço não tempo real 56
5.5 Camada de Adaptação ATM 58
5.6 Integrando LANs às Redes ATM 59
5.6.1 LAN Emulation 59
5.6.2 Classical IP e ARP sobre ATM 61
5.7 Conclusão 63
6. Filtros Adaptativos no Transporte de Vídeo sobre Redes Heterogêneas 66
6.1 Integração da Tecnologia ATM na MBone 67
6.2 Gateway ATM para MBone 69
6.3 Fluxo de dados no Roteador 70
6.4 Arquitetura de Gateway Proposta 71
6.5 Filtragem Tempo-real 73
6.6 Modelagem do FiltroGW 73
6.6.1 Módulo de Captura 74
6.6.2 Módulo de Seleção 75
6.6.3 Módulo Filtro 77
6.6.4 Módulo de Devolução 78
6.6.5 Módulo de Descarte 78
6.7 Descrição do Protótipo 79
6.7.1 Módulo de Captura 79
6.7.2 Módulo de Seleção 82
6.7.3 Módulo Filtro 83
6.7.4 Módulo de Devolução 84
6.7.5 Módulo de Descarte 84
6.7.6 Avaliação do Protótipo 84
6.8 Testes realizados 84
6.8.1 Descrição do Ambiente de Teste 85
6.8.2 Testes do Gateway ATM/Ethernet Tradicional 86
6.8.3 Testes do Gateway com Filtro Adaptativo RTP 87
6.9 Conclusões 87
7. Conclusão 88
8. Referências Bibliográficas 90
ANEXO 1 94
ANEXO 2 99
ANEXO 3 106

LISTA DE QUADROS E FIGURAS

Figura 2.1 Técnica de bufferização [Lu, 96] *8*
Figura 3.1 Pilha de protocolo típica para mídias contínuas usando RTP *15*
Figura 3.2 Formato do cabeçalho RTP *16*
Figura 3.3 Formato do pacote SR *20*
Figura 3.4 Formato do pacote RR *22*
Figura 3.5 Formato do pacote SDES *23*
Figura 3.6 Tipos de itens do pacote SDES *23*
Figura 3.7 Formato do pacote BYE *23*
Figura 3.8 Formato do pacote APP *24*
Figura 3.9 Mecanismo de adaptação básico de um sistema multimídia distribuído [Gecsei 97] *26*
Figura 3.10 Espaço ocupado pelas resoluções padrão SQCIF,QCIF, CIF, 4CIF em telas de 800x600 ..... *31*
Tabela 3.11 Padrões de codificação de vídeo *32*
Figura 4.1 Classes de endereços IP *35*
Figura 4.2 Topologia MBone em 1994 ..... *38*
Figura 4.3 Janela principal do SDR *41*
Figura 4.4 Janela de informação de uma sessão da SDR *41*
Figura 4.5 Janela principal da Visual Audio Tool *42*
Figura 4.6 Janela principal do IVS *43*
Figura 4.7 Janela principal da VIC *44*
Figura 4.8 Janela Principal do NV *46*
Figura 5.1 Modelo de Referência para Protocolos - PRM *49*
Figura 5.2 Formato da célula ATM *53*
Figura 5.3 Cabeçalho da célula no NNI *54*
Figura 5.4 Cabeçalho da célula no UNI *54*
Figura 5.5 Emulação de LAN sobre ATM *60*
Figura 6.1 Gateway ATM para MBone *69*
Figura 6.2 Sobrecarga nos roteadores Mbone *70*
Figura 6.3 Arquitetura de Gateway proposta *72*
Figura 6.4 Redução do fluxo de vídeo *72*
Figura 6.5 Módulos do FiltroGW *74*
Figura 6.6 Formato do pacote UDP *74*
Figura 6.7 Funcionamento do módulo Seleção *77*
Figura 6.8 Estrutura utilizada pelo filtro IP *82*
Figura 6.9 Captura da estrutura utilizada pelo filtro IP *82*
Figura 6.11 Ambiente de teste *85*

LISTA DE ABREVITURAS E SIGLAS

AAL	- Adaptation Layer
ABR	- Available Bit Rate
ARP	- Address Resolution Protocol
ATM	- Asynchronous Transfer Mode
B-ISDN	- Broadband Integrated Services Digital Network
BUS	- Broadcast and Unknown Server
CBR	- Constant Bit Rate
CCITT	- Comité Consultatif International de Télégraphique et Téléphonique
CDV	- Cell Delay Variatio
CIF	- Comum Intermediate Format
CLIP	- Classical IP
CLR	- Cell Loss Ratio
CSMA/CD	- Carrier Sense Multiple Access with Collision Detection
CTD	- Cell Transfer Delay
DVMRP	- Distance Vector Multicast Routing Protocol
FEC	- Forward Correction Erro
FIFO	- First-In First-Out
IDNs	- Integrated Digital Networks
IEEE	- Institute of Electrical and Electronics Engineers
IETF	- Internet Engineering Task Force
IGMP	- Internet Group Management Protocol
IP	- Internet Protocol
ISDN	- Integrated Services Digital Network
ITU	- International Telecommunications Union
IVS	- INRIA Conferencing System
LAN	- Local Area Network
LANE	- LANEmulation
LEC	- LAN Emulation Client
LECS	- LAN Emulation Configuration Server
LES	- LAN Emulation Server
LIS	- Logical IP Subnet

LUNI	- User-Network Interface
MAN	- Metropolitan Area Network
MBone	- Multicast Backbone (Virtual Internet Backbone for Multicast IP)
MBS	- Maximum Burst Size
MMCC	- Multimedia Conference Control
MOSPF	- Multicast Open Shortest Path First
MOST	- Mobile Open Systems Technologies
MROUTER	- Multicast Router
MTU	- Maximum Transmit Unit
NNI	- Network-Node Interfaces
nrt-VBR	- no-real-time Variable Bit Rate
NV	- Network Video
PCR	- Peak Cell Rate
PDU	- Protocol Data Unit
PIM	- Protocol-Independent Multicast
QoS	- Qualidade de Serviço
RSVP	- ReSource ReserVation Protocol
RTCP	- RTP Control Protoco
RTP	- Real-Time Transport Protocol
rt-VBR	- real-time Variable Bit Rate
RV	- Rendez-Vous
SDR	- Session Directory
SDU	- Service Data Unit
SCR	- Sustainable Cell Rate
ST-II	- Stream Protocol Version II
SVCC	- Signalling Virtual Channel Connection
TCP	- Transmission Control Protocol
TTL	- Time To Live
UBR	- Unspecified Bit Rate
UDP	- User Datagram Protocol
VAT	- Visual Audio Tool
VBR	- Variable Bit Rate
VCC	- Virtual Channel Connection
VIC	- Video Conference

VLAN	- Virtual Local Area Network
VLSI	- Very Large-Scale Integration
VPC	- Virtual Path Connection
WAN	- Wide Area Network

RESUMO

Este trabalho apresenta uma visão geral sobre multimídia e propõe uma solução à problemática no transporte da mídia de vídeo entre duas redes heterogêneas. Os protocolos multimídia mais utilizados, as aplicações desenvolvidas, o *Multicast Backbone* (MBone) - um backbone responsável pela realização de conferências na Internet - e a rede ATM (*Asynchronous Transfer Mode*) são pontos de partida desse estudo para se avaliar: as soluções de interoperabilidade existentes entre duas redes heterogêneas (MBone e ATM), a problemática da transmissão da mídia vídeo entre essas redes e o comportamento das aplicações quando os recursos disponibilizados para esse tipo de transmissão são diferentes em cada rede. Por fim, o modelo de um filtro de escalamento de mídia é introduzido onde o tráfego da rede ATM é filtrado ao passar para a rede MBone – evitando sobrecarga nos roteadores - e as informações sobre a disponibilidade dos recursos relativos à rede MBone não são repassadas para a rede ATM. A justificativa do trabalho está baseada em testes realizados em um ambiente responsável pela interconexão entre as duas redes. A modelagem do "filtro de escalamento de mídia" também inclue um protótipo de implementação.

ABSTRACT

This study presents a general view on multimedia. It proposes a solution to the problematic of video media transport between heterogeneous networks. The commonest multimedia protocols, the developed applications, the Multicast Backbone (MBone – responsible for the realization of internet conferences) and the Asynchronous Transfer Mode Network (ATM) are the starting points in the evaluation of: the existential solutions of interoperability between heterogeneous networks (MBone and ATM), the problematic of the video media transmission through these two networks and the behavior of the applications when the available resources for this type of transmission are different in each network. Finally a scaling media filter network model is introduced where the ATM network traffic is filtered when passes through the MBone - avoiding the overload in the routers - and the information about the available resources related to the MBone network are not repassed over the ATM network. My study is based on tests that took place on an environment able to connect the two networks. The mould "media scaling filter" also includes an implementation prototype.

1. Introdução

O desenvolvimento tecnológico dos sistemas de informação e de comunicação de alta velocidade permitiram o surgimento de novas aplicações no domínio de sistemas distribuídos. Uma das principais tendências é a integração de diferentes tipos de mídia, como texto, imagem, voz e vídeo em um vasto domínio de aplicações informáticas distribuídas (por exemplo, tele-ensino e videoconferência). Esses sistemas são chamados de sistemas multimídia. Atualmente, as indústrias de informática, de telecomunicações e de dispositivos de áudio e vídeo têm contribuído muito para a grande evolução de sistemas multimídia.

Com a popularização da Internet e a evolução do hardware de componentes pessoais, é cada vez mais comum a utilização de multimídia. Qualquer usuário pode ter seu próprio equipamento e transmitir áudio e vídeo sobre a rede. As aplicações multimídia desenvolvidas são otimizadas procurando maior resolução de vídeo e menor distorção na transmissão de áudio. Mas quando o limite de otimização é atingido, torna-se necessário que a rede forneça meios que resultem na melhoria da transmissão desse tipo de tráfego.

Sobre a rede Internet foi implantado um *backbone multicast*, denominado MBone, que implementa sessões de videoconferência multiponto. O MBone está amplamente difundido, sendo utilizado por organizações e universidades na realização de videoconferências, e, sendo responsável pelo desenvolvimento, teste e otimização de grande parte dos protocolos e aplicações multimídia existentes. Usuários de todo o mundo possuem acesso à rede Internet e, por isso, além das ferramentas de videoconferência utilizadas pelo MBone, muitas outras aplicações multimídia foram desenvolvidas para funcionar sobre o protocolo IP.

O ATM (*Asynchronous Transfer Mode*) é uma tecnologia promissora na área de Redes de Computadores. Entre os objetivos do projeto da rede ATM estão: fornecer vídeo sob demanda para todas as residências, televisão ao vivo de muitas origens, correio eletrônico multimídia com vídeo, música com qualidade de CD, interconexão LAN e transporte de dados em alta velocidade para uso científico e industrial. A rede

ATM possui requisitos ideais para a comunicação multimídia, mas tal tecnologia ainda tem um custo alto, o que faz com que demore algum tempo para que *backbones Ethernet* sejam substituídos por *backbones* ATM.

Portanto, são necessárias soluções de interoperabilidade que possibilitem a transmissão do tráfego multimídia entre as duas redes: MBone e ATM, permitindo que aplicações IP sejam utilizadas na rede ATM, que um usuário da rede MBone possa ter acesso a uma transmissão de áudio/vídeo gerada por algum *host* do *backbone* ATM e que a rede ATM possa receber o tráfego multimídia existente na rede MBone.

O objetivo principal desse trabalho é analisar o comportamento da mídia de vídeo quando transportada entre redes heterogêneas e verificar a necessidade da utilização de filtros no nó que interliga as redes.

O presente trabalho é assim constituído:

- o capítulo 2 apresenta um estudo sobre sistemas multimídia, suas características, os requisitos de comunicação e os requisitos de rede necessários para que uma aplicação multimídia tenha um bom funcionamento;
- o capítulo 3 apresenta um estudo sobre as aplicações multimídia adaptativas, o protocolo e os padrões de codificação de vídeo utilizados;
- o capítulo 4 apresenta a rede virtual MBone, utilizada para realizar videoconferência sobre a Internet. Essa apresentação inclui o endereçamento dos pacotes *multicast*, o protocolo de comunicação em grupo IGMP (*Internet Group Management Protocol*) e as aplicações de conferência utilizadas na transmissão dos eventos.
- o capítulo 5 apresenta a arquitetura ATM: as conexões lógicas, o formato das células, as categorias de serviço, a camada de adaptação e as soluções de interoperabilidade existentes
- o capítulo 6 apresenta o ambiente de rede heterogênea onde foi detectado a necessidade de utilização de filtros de escalamento de mídia e descreve o modelo e um protótipo de implementação de um filtro de escalamento de vídeo.

2. Sistemas Multimídia

O grande desenvolvimento da tecnologia digital a nível de processamento, armazenamento e velocidade de transmissão, permitiu que diferentes tipos de informação, tais como áudio e vídeo, pudessem ser processados de forma integrada, originando os sistemas multimídia.

Este capítulo apresenta um estudo sobre as características das fontes de tráfego multimídia e os requisitos que um sistema de comunicação deveria satisfazer para possibilitar o transporte em tempo-real de informações multimídia. Mas antes, será apresentada uma definição de multimídia.

2.1 Definição de Multimídia

Para melhor definir o termo multimídia, deve-se inicialmente conhecer a classificação de tipos de mídias, que podem ser classificados de acordo com as características temporais das mídias:

- **Mídias discretas ou estáticas** são mídias com dimensões unicamente espaciais que não dependem do tempo. Como exemplo, podem-se citar as informações textuais, imagens e gráficos.
- **Mídias contínuas ou dinâmicas** são mídias que dependem do tempo, tais como áudio, vídeo e animações.

Baseada nessa classificação, a definição de sistemas multimídia mais adotada é a seguinte [Fluckiger, 95]: "Um sistema capaz de manipular ao menos um tipo de mídia discreta e um tipo de mídia contínua, as duas na forma digital".

2.2 Sistemas Multimídia Standalone e Distribuídos

Existem dois grupos de pesquisa em multimídia: um centra seus esforços em estações de trabalho *standalone* e o outro combina multimídia com sistemas distribuídos. Aplicações multimídia em sistemas *standalone* utilizam apenas os recursos presentes no sistema local para prover serviços multimídia. Esses sistemas são equipados com os dispositivos necessários, tais como microfones e câmeras, e não

suportam comunicações de dados multimídia. Como exemplo, pode-se citar um computador individual contendo um CD-ROM e ferramentas de autoria de documentos multimídia.

O propósito de muitas aplicações multimídia é fornecer serviços de comunicação a distância, como, por exemplo, videoconferência, redes de distribuição de pacotes de áudio e vídeo e *e-mail* multimídia. Nestes casos, estas aplicações devem ser suportadas por um sistema de comunicação. Estes sistemas são chamadas de sistemas multimídia distribuídos.

2.3 Características das Fontes de Tráfego Multimídia

O tráfego multimídia normalmente consiste de fluxos de uma grande quantidade de dados gerados por fontes de áudio e vídeo, que são quebrados em pacotes ou quadros para transporte na rede. Este tipo de tráfego pode ser caracterizado como:

- **Tráfego a taxa de bits constante** (CBR - *Constant Bit Rate*): muitas aplicações multimídia geram saída a taxa de bits constante. Um exemplo são os fluxos de áudio não compactados, que na qualidade telefone gera uma taxa constante de 64 Kbps (8000 amostras/s de 8 bits). Para aplicações em tempo-real, envolvendo fluxos de dados CBR, é importante que a rede também transporte estes fluxos de dados a uma taxa de bits constante. Em redes a comutação de pacotes, como as redes IP (*Internet Protocol*), este tipo de tráfego é difícil de ser sustentado.
- **Tráfego a taxa de bits variável** (VBR- *Variable Bit Rate*): possui uma taxa de bits que varia com o tempo. Este tipo de tráfego ocorre geralmente quando se utiliza mídias compactadas, como vídeo MPEG. Nestes casos, a taxa de bits gerada varia basicamente de acordo com o grau de variação entre as imagens do vídeo. Esse tráfego normalmente ocorre em rajadas, caracterizado por períodos aleatórios de relativa inatividade quebrados por rajadas de dados.

O tráfego gerado pelas fontes de áudio e vídeo também possui como características de dependência temporal e de continuidade temporal que devem satisfeitas pelo sistema. A característica de dependência temporal está relacionada com a sincronização entre as amostras de áudio e quadros de vídeo (sincronização intra-mídia) ou então entre as imagens e as amostras de áudio em uma sincronização labial

(sincronização inter-mídia). A continuidade temporal está relacionada a continuação da apresentação. Tanto a dependência e a continuidade temporal devem ser mantidas, do contrário, a qualidade obtida será inaceitavelmente baixa.

Quando pessoas estão envolvidas na comunicação, o sistema deve permitir um certo nível de interatividade, portanto, o atraso total deve estar abaixo do nível de tolerância perceptível pelo usuário. Por exemplo, testes mostram que o atraso entre a captura das cenas de uma pessoa falando e a sua apresentação deve estar na ordem de 300ms, pois, caso contrário, a interatividade se torna inviável.

2.4 Requisitos de Rede para Aplicações Multimídia

A natureza síncrona das mídias contínuas impõe duros requisitos em termos de largura de banda, atrasos, variação de atrasos e outros. Esta seção identifica os principais requisitos que devem ser satisfeitos pelo sistema de comunicação para que este seja considerado adequado para aplicações multimídia.

2.4.1 Largura de banda

A largura de banda da rede fornece a capacidade de transmissão da rede. Ela é determinada pelo meio de transmissão utilizado, protocolos, distância entre nós intermediários e velocidade de comutação nos nós intermediários. Entre par trançado, cabo coaxial e fibra ótica, esta última é a que oferece maior largura de banda.

Uma rede é considerada satisfatória para aplicações multimídia quando a sua largura de banda é suficientemente alta para suportar várias aplicações multimídia. Este requisito é de difícil quantificação, pois depende da qualidade de áudio e vídeo escolhida para a transmissão e das técnicas de compressão utilizadas. Por exemplo, para transmitir uma imagem utilizando o padrão de compressão H261 é necessário uma largura de banda de 64Kbytes a 2Mbps, enquanto que no padrão MPEG é necessário 1,2Mbytes a 80Mbps. Note que quanto mais alta a taxa de transmissão, melhor é a qualidade de restituição da mídia.

Em redes baseadas no melhor esforço, como no padrão Ethernet 10Mbps, os componentes de comutação e transmissão são compartilhados por muitas comunicações (através da multiplexação), cada uma obtendo uma fração da largura de banda. Esta

fração, na maioria das vezes, é insuficiente para satisfazer os requisitos das aplicações multimídia.

A rede Ethernet a 10 Mbps baseada em CSMA/CD (IEEE 802.3) é a tecnologia de rede locais mais utilizada. Assumindo que parte da largura de banda é necessária deixar para tráfego de dados e de controle e que Ethernets não poderiam ser mais carregadas que 70% a 80%, para manter as colisões a um nível aceitável, então apenas 5 a 6 Mbps são realmente disponíveis para fluxos multimídia. Além disso, esta largura de banda é compartilhada por todos os *hosts*. Assim, estas redes suportam não mais que poucos fluxos de vídeo em paralelo e de baixa qualidade de imagem.

Ao contrário da rede Ethernet a 10Mbps, as redes de alta velocidade, como as baseadas na tecnologia ATM (apresentada no capítulo 4), fornecem um acesso de alta velocidade para cada sistema final. Duas velocidades de acesso, 155 e 622 Mbps, foram padronizadas, embora velocidades mais altas e mais baixas sejam possíveis. Estas velocidades de acesso são suficiente para qualquer tipo de aplicação multimídia com a tecnologia de compressão atual.

2.4.2 Requisitos de atraso

Em qualquer sistema multimídia distribuído, sempre existe um atraso entre a captura/leitura de uma informação (p.e., a captura de um vídeo por uma câmera ou leitura de um arquivo de vídeo) de uma fonte e sua apresentação na tela dos destinos. Este atraso, chamado de atraso fim-a-fim, é gerado pelo processamento da informação na fonte, sistema de transmissão e processamento no destino.

Os principais subsistemas que afetam a taxa de bits, atraso e variação de atraso são os seguintes [Lu, 96]:

- **Codificadores**: devido a requisitos de tempo-real neste tipo de aplicação, os codificadores são geralmente implementados em hardware. Este subsistema, onde dados são compactados em tempo-real, gera atrasos na ordem de milissegundos.
- **Pilhas de Transporte no Transmissor e Receptor**: após codificados, os dados são enviados à pilha de transporte no transmissor, à pilha de protocolos no receptor e ao decodificador sob o controle de software. Muitas pilhas de

transporte são implementadas por software. A pilha de protocolos causa atrasos e variação de atrasos significativos.

- **Acesso à Rede**: diferentes redes tem diferentes protocolos de controle de acesso. Algumas redes garantem que um pacote seja capaz de acessar o meio dentro de um certo tempo, já em outras este tempo não pode ser garantido. Obviamente, este último protocolo de acesso à rede não é ideal para aplicações multimídia.
- **Rede de transmissão**: os dados na rede podem passar por vários nós intermediários antes de atingir o destino. Estes nós intermediários podem ter recursos insuficientes para retransmitir o dado. Assim, estes dados podem ser descartados ou atrasados. Isto depende dos protocolos de transporte e gerência de rede usados.
- **Decodificador**: decodificadores de dados podem ser implementados em hardware ou software. Decodificadores em hardware tem pequeno efeito na taxa de bits, atrasos e variação de atrasos. Mas um codificador em software pode ter problemas pelas mesmas razões das pilhas de protocolo.

Para atender os requisitos dos sistemas multimídia, o atraso fim-a-fim deve ser pequeno e limitado. Se o atraso fim-a-fim não for limitado, o tempo de resposta do sistema também não será. Isso é indesejável especialmente em aplicações multimídia interativas. Como já apresentado, testes mostram que o atraso fim-a-fim em aplicações interativas, do tipo videofonia e videoconferência, deve limitar-se a 300ms.

2.4.3 Requisitos de variação de atraso

Em redes de comutação de pacotes, os pacotes de dados não chegam ao destino em intervalos fixos como é necessário para transmissão de mídias contínuas. Por causa desta variação de atrasos, pacotes de áudio e vídeo que chegam não podem ser imediatamente apresentados. Caso contrário, tem-se uma apresentação de áudio e vídeo de má qualidade. A nível de percepção humana, a variação de atrasos na transmissão de pacotes de voz é o problema mais crítico, podendo tornar a fala incompreensível.

A abordagem mais utilizada para a remoção desta variação de atraso é o uso de *buffers* do tipo FIFO (First-In First-Out) no destino, antes da apresentação. Esta técnica é chamada de técnica de bufferização e pode ser visualizada na figura 2.1. Nela, a

medida que os pacotes chegam (em uma taxa variada) eles são colocados no *buffer*; o dispositivo de apresentação retira amostragens do *buffer* a uma taxa fixa [Lu, 96].

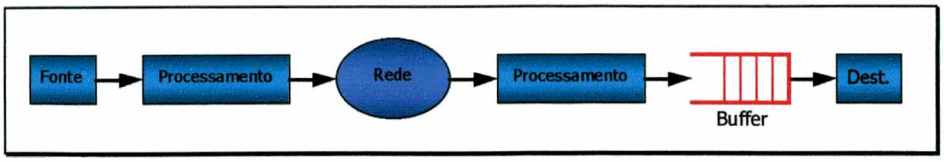

Figura 2.1 Técnica de bufferização [Lu, 96]

O princípio da técnica de bufferização é adicionar um valor de atraso variável a cada pacote de tal forma que o atraso total de cada pacote seja o mesmo. Por esta razão, este *buffer* é chamado de *buffer* de uniformização de atrasos. Para satisfazer os requisitos da comunicação multimídia o *buffer* não deve sofrer sobrecarga ou pouca utilização. Em contrapartida, o tamanho do *buffer* não deve ser muito grande: um *buffer* grande significa que o sistema é caro e o atraso fim-a-fim é muito grande.

2.4.4 Requisitos de confiabilidade

Redes podem perder dados. Normalmente, o que ocasiona perdas de dados é o congestionamento da rede, afetando nodos ou linhas de transmissão, e resultando no descarte dos pacotes nos nós intermediários.

A maneira que a rede respeita a integridade dos dados que ela transporta é avaliada através da taxa de erro. Taxa de erro é a medida do comportamento da rede com respeito à alteração, perda, duplicação ou envio dos dados fora de ordem. A alteração dos dados ocorre quando o que é transportado difere do que é emitido, ou seja, a rede não respeita a integridade dos dados. Em redes modernas, a alteração dos dados é a forma menos freqüente de ocorrência de erros. A duplicação de dados é um incidente raro que ocorre quando o mesmo dado é recebido mais de uma vez. Já o envio de dados fora de ordem é um incidente mais freqüente. Quando ocorre congestionamento, principalmente em redes TCP/IP, os pacotes são enviados por rotas diferentes e precisam ser ordenados quando chegam ao destino. Redes ATM não possuem essa característica, ou seja, não enviam pacotes fora de ordem.

É difícil precisar os requisitos de controle de erro para redes multimídia, pois as aplicações multimídia são, de certo modo, tolerantes a erros de transmissão. Parte da razão dessa tolerância é devida aos limites de percepção sensorial humana que permite

ou não escutar e entender uma transmissão de áudio e verificar se a visualização da imagem recebida é aceitável.

Requisitos de controle de erro são também difíceis de se quantificar, pois em muitos casos tais requisitos e aqueles referentes ao atraso fim-a-fim, são contraditórios. Esta contradição decorre do fato de que muitos esquemas de controle de erro envolvem a detecção e retransmissão do pacote com erros ou perdido. Algumas vezes, a transmissão deve ser realizada na base fim-a-fim, que significa um aumento no atraso. Para transmissão tempo-real de áudio e vídeo, o atraso é mais importante que a taxa de erros e, em muitos casos, é preferível ignorar o erro e trabalhar simplesmente com o fluxo de dado recebido.

2.4.5 Garantia de Desempenho

Um requisito importante para se obter apresentações de áudio e vídeo de boa qualidade é o fornecimento de garantias de desempenho. Isto tanto a nível de sistemas locais quanto a de sistema de comunicação. Quando a largura de banda, atraso, variação de atraso, taxa de erro são garantidos, a apresentação multimídia poderá ser feita com qualidade.

Para garantir o desempenho, a rede deve garantir que um pacote possa acessá-la em um tempo especificado e que o pacote deve ser liberado dentro de um tempo limitado. Existem duas possibilidades onde o desempenho não pode ser garantido: quando o protocolo de acesso ao meio não garante o tempo de acesso a rede (CSMA/CD), ou na ocorrência de falhas de garantia de desempenho nos nós intermediários e comutadores da rede (devido ao congestionamento, ao limite existente no tamanho dos *buffers* de armazenamento, aos atrasos na transmissão e as perda de pacotes).

As seguintes técnicas podem ser empregadas para permitir o uso eficiente de recursos de rede e garantir o desempenho da aplicação:

- Determinação das características dos diferentes tipos de tráfegos em termos de requisitos de pico da taxa de dados, intervalos de rajada, atrasos, variação de atrasos. Estas informações de tráfego devem ser enviadas à rede quando a conexão é solicitada e são usadas para a decisão de aceitação da conexão.

- Tempo de acesso à rede deve ser garantido.
- Recursos de rede (largura de banda e *buffers*) devem ser eficientemente gerenciados de maneira que tantas aplicações quanto possível sejam suportadas com garantia de desempenho.

Para fornecer uma única abordagem para que diferentes aplicações especifiquem as garantias de desempenho necessárias e para que sistemas forneçam as garantias requeridas, foi introduzido o conceito de Qualidade de Serviço (QoS). Não existe uma definição de QoS universalmente aceita atualmente. [Lu, 96] define QoS como "uma especificação qualitativa e quantitativa dos requisitos de uma aplicação que um sistema multimídia deveria satisfazer a fim de obter a qualidade desejada".

Baseada nesta definição, existem dois aspectos para QoS: aplicações especificam os requisitos de QoS e o sistema fornece as garantias de QoS. A QoS é normalmente especificada por um conjunto de parâmetros, principalmente a taxa de bits, taxa de erros, limites de atraso e de variação de atraso. Portanto, as aplicações podem indicar requerimentos específicos (como a largura de banda necessária) que a rede deve garantir antes do início da transmissão dos dados. A rede pode recusar a conexão, se ela não puder satisfazer os requisitos de QoS, ou aceitar a conexão reservando os recursos requisitados.

2.4.6 Uso adequado dos recursos de rede

Um aspecto importante para o desempenho na transmissão de dados multimídia é o uso adequado dos recursos que a rede oferece. Se cada aplicação que necessite de uma determinada largura de banda reservar largura de banda igual a taxa de pico, haverá um desperdício quando a taxa ocupada for inferior a que foi reservada.

A utilização ideal da largura de banda depende do tipo de comunicação que a rede utiliza: comutação de circuitos ou comutação de pacotes. Na comutação de circuitos é necessário estabelecer uma conexão ponto a ponto antes que os dados possam ser transmitidos. Ao contrário da comutação de circuitos, na comutação de pacotes não existe um caminho dedicado entre duas estações e um cabeçalho é adicionado à mensagem contendo o endereço destino. Essa mensagem é então transmitida de nó a nó pela rede até chegar ao destino.

Redes de comutação de circuitos são ideais para transmitir tráfego de dados contínuos, mas nem todos os tráfegos em uma rede são contínuos e com taxas constantes. Vídeo comprimido, texto e gráficos, podem ser tráfegos com taxa variável. A transmissão de tráfego com fluxo de dados variável em redes comutadas por circuitos causa um desperdício de largura de banda. Um fluxo de dados variável pode ser muito pequeno em um determinado instante e não ocupar toda a largura de banda alocada no canal de comutação.

Na comutação de pacotes, o tamanho dos pacotes também afeta a eficiência do uso da largura de banda. Quando o pacote é muito grande, muitos não são preenchidos, especialmente o último de uma mensagem, desperdiçando largura de banda. Quando eles são muito pequenos, há desperdício com os cabeçalhos.

Um outro fato que afeta o uso adequado dos recursos da rede é a retransmissão dos pacotes perdidos. Para transmissões de áudio e vídeo ao vivo, a retransmissão não é desejável porque pode causar um tempo de retardo extra, prejudicando a apresentação multimídia, com os pacotes chegando muito tarde ao destino.

2.4.7 Multicasting (Difusão Seletiva)

A maioria das aplicações tradicionais em redes envolve comunicação entre dois computadores. Com o surgimento das aplicações multimídia, tornou-se necessário distribuir o fluxo de áudio e vídeo para múltiplos destinos. Por exemplo, em uma videoconferência, a voz e imagem de um locutor precisam ser enviadas para todos os participantes da conferência localizados em lugares diferentes da rede.

Existem três maneiras de fazer essa distribuição de fluxo:

- **Difusão simples**: permite que as aplicações enviem uma cópia de cada pacote para cada membro de um grupo. Esta técnica é simples de implementar, mas possui uma restrição significativa em relação ao desempenho se o grupo for muito grande. Esta técnica também exige uma largura de banda extra porque a mesma informação tem que ser transmitida várias vezes sobre um mesmo enlace compartilhado.
- **Difusão (*broadcast*)**: em LANs onde as estações compartilham um mesmo meio de transmissão, esta técnica permite que as aplicações enviem uma cópia de cada

pacote (utilizando um endereço especial) para todas as estações conectadas à LAN. A utilização dessa técnica permite que a rede possa enviar pacotes para fora da LAN. Fazer difusão de pacotes para fora da LAN (alcançando uma MAN ou WAN) é inviável, pois resulta em mal aproveitamento dos recursos da rede, se esses pacotes estiverem direcionados apenas para um grupo pequeno de *hosts*.

- **Difusão Seletiva *(multicast)***: permite que aplicações enviem uma cópia de cada pacote apenas para um grupo de computadores que deseja recebê-los. Essa é a melhor solução para distribuição do fluxo de dados multimídia, pois a fonte envia os dados apenas uma vez e a rede se responsabiliza pela transmissão desses dados para os vários destinos.

Em redes de comutação de circuitos é difícil implementar difusão seletiva, mas em redes de comutação de pacotes é possível enviar um pacote para múltiplos destinos com relativa facilidade. Foram desenvolvidas técnicas de difusão seletiva para redes de comutação de pacotes, onde a estação interessada em enviar ou receber um pacote de dados faz parte de um grupo que possui um endereço único. O endereço do grupo é usado como endereço destino, que é conhecido e utilizado pelos roteadores *multicast*. Quando um roteador recebe um pacote com o endereço de grupo, ele envia os pacote nos enlaces de comunicação que conduzem às estações pertencentes a este grupo. O mesmo pacote não será enviado duas vezes sobre o mesmo enlace de comunicação.

2.5 Conclusão

Como apresentado neste capítulo, as fontes de áudio e vídeo impõe severos requisitos ao sistema de comunicação, em termos de largura de banda, atraso, variação de atraso, garantias de desempenho e suporte a *multicasting*, sendo a garantia de desempenho um dos requisitos mais importantes para se obter apresentações de áudio e vídeo de boa qualidade.

Afim de fornecer garantias de desempenho, algumas tecnologias (p.e., ATM) implementam o gerenciamento de Qualidade de Serviço. Alguns pesquisadores são contra a utilização da qualidade de serviço. Eles sugerem que a aplicação multimídia se

adapte aos recursos disponíveis da rede. Essas aplicações, denominadas aplicações adaptativas, serão vistas no próximo capítulo.

3. Aplicações Adaptativas

Como apresentado no capítulo anterior, alguns pesquisadores propõem uma abordagem muito radical à QoS (Qualidade de Serviço). Eles basicamente se opõem a qualquer espécie de reserva de recurso dentro da rede e argumentam que as aplicações deveriam ser capazes de se adaptar às mudanças arbitrárias das condições de rede [Bolot, 94] [Diot, 95]. Os principais argumentos são:

- Reserva de recurso é muito cara em termos de mensagens de protocolo.
- Reservas excessivas leva a um uso ineficiente de recursos.
- Reservas *hard* (a 100%) não podem se adaptar a fluxos com variação de QoS durante sua duração.
- Se a rede é realmente muito congestionada, pode-se sempre lançar largura de banda ao problema.

Assim, a idéia é transmitir fluxos multimídia sobre redes tradicionais do tipo melhor esforço, tal como a Internet. O emissor pode variar sua taxa de pacotes a qualquer momento. Se a rede sinaliza uma sobrecarga, as aplicações terão mecanismos específicos para reagir, tal como a modificação do parâmetro de quantização em MPEG ou a escolha de um outro algoritmo de compressão de áudio. Aplicações adaptativas usam este escalamento de mídia como sua única maneira de ajuste a QoS e não há protocolo de reserva ou contrato de QoS com a rede.

Um dos principais aspectos das aplicações adaptativas é a necessidade de um mecanismo para obtenção dos parâmetros de desempenho da rede, a fim de que sejam tomadas as medidas necessárias para adaptar a mídia a estes parâmetros de desempenho. Para isto, várias aplicações adaptativas utilizam o protocolo RTP (*Real-Time Transport Protocol*), que será apresentado na próxima seção.

3.1 Protocolo RTP

Vários grupos de trabalho Internet tem sido formados para estudar os requisitos da comunicação multimídia e tempo-real. O Grupo de Trabalho Transporte de Áudio e

Vídeo está desenvolvendo o protocolo RTP (*Real-Time Transport Protocol*) com a finalidade de fornecer um serviço de transporte fim-a-fim para dados com características tempo-real, como áudio e vídeo [Schulzrinne, 97]. O objetivo principal do RTP é satisfazer as necessidades da conferência multimídia multi-usuários, mas seu uso não é limitado a isto.

RTP não contém praticamente nenhuma suposição específica sobre as capacidades das camadas inferiores. Ele não contem nenhum endereço da camada de rede, de forma que não é afetado por mudanças de endereçamento. Além disso, qualquer capacidade adicional da camada inferior, como segurança ou garantias de qualidade de serviço, pode ser usada obviamente por aplicações que empregam RTP. Há suporte inclusive à transferência de dados para destinos múltiplos usando distribuição *multicast,* se provido pelas camadas inferiores.

Embora RTP seja um protocolo ainda muito novo, várias aplicações Internet já o implementam, em particular, as ferramentas MBone vic e vat. Os algoritmos RTP não são usualmente implementados como uma camada separada, mas sim como parte do código da aplicação.

A figura 3.1 mostra as relações entre vários protocolos. Nesta estrutura, o objetivo do RTP é fornecer uma comunicação fim-a-fim em tempo-real sobre a Internet. Notamos que ele fornece uma função de transporte de dados e é chamado de protocolo de transporte, apesar de ser realmente usado com freqüência sobre o UDP (*User Datagram Protocol)*, um protocolo de transporte sem conexão. Para fornecer garantias de qualidade de serviço, RTP deve operar no topo de um protocolo de reserva de recursos tal como o ST-II e RSVP.

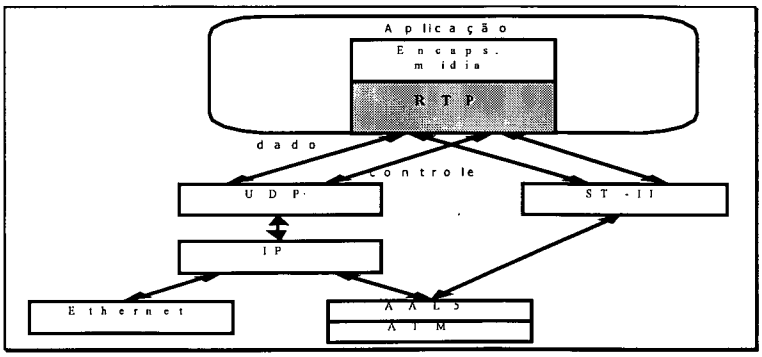

Figura 3.1 Pilha de protocolo típica para mídias contínuas usando RTP

RTP consiste de duas partes, uma parte de dados e outra de controle. O último é chamado de RTCP (*RTP Control Protocol*).

3.1.1 Parte de dados do RTP

Em redes de pacotes, como a Internet, ocorrem ocasionalmente perdas ou reordenação dos pacotes, acrescentando retardos no tempo da transmissão. Para solucionar esse problema, o cabeçalho RTP contém informação de temporização e um número de seqüência que permite que receptores reconstruam a temporização produzida pela origem. Esse número de seqüência também pode ser usado como estimativa da quantidade de pacotes perdidos e para determinar a alocação apropriada do pacote, por exemplo, na decodificação de vídeo.

O RTP faz uso dos serviços de multiplexação e *checksum* fornecidos por um protocolo da camada inferior que pode ser o UDP.

Formato do cabeçalho RTP

A figura 3.2 mostra o formato do cabeçalho RTP.

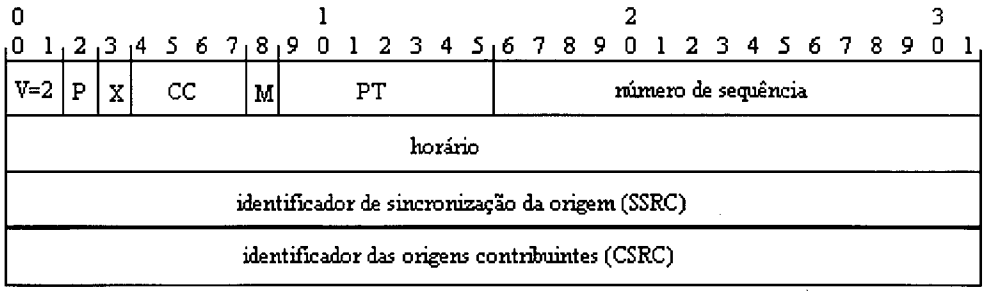

Figura 3.2 Formato do cabeçalho RTP

Os primeiros doze octetos do cabeçalho RTP estão presentes em todos os pacotes, enquanto que a lista de identificadores CSRC está presente apenas quando um misturador (*mixer*) tiver sido inserido. Misturadores são sistemas intermediários que recebem pacotes RTP de uma ou mais fontes, realizam mudanças em seu formato, combinam os pacotes em um mesmo formato e então os reenviam como se fosse um novo pacote RTP. A função dos misturadores é adaptar o áudio codificado nos enlaces de rede que possuem baixa velocidade de acesso e pouca largura de banda disponível.

Os campos do cabeçalho RTP tem o seguinte significado:

- **versão (V)**: ocupa dois bits. Identifica a versão do protocolo RTP. A versão atual é identificada por 2, o valor 1 é usado para a primeira versão do *draft* e o valor 0 é usado pelo protocolo inicialmente implementado pela ferramenta de áudio vat.
- **preenchimento (P)**: ocupa um bit. Quando P=1 significa que o pacote contém um ou mais octetos de preenchimento, adicionados no fim do pacote. Os octetos de preenchimento não fazem parte da informação de controle, e o último octeto contém o número de todos os octetos de preenchimento que devem ser ignorados (p.e. necessários, por exemplo, para criptografia).
- **extensão(X)**: ocupa 1 bit. Quando X=1, o cabeçalho fixo é seguido por uma extensão. Extensões de cabeçalho contêm informações dependentes de aplicação.
- **contador CSRC (CC)**: ocupa 4 bits. O contador CSRC (*Contributing source identifiers*) contém o número de identificadores de CSRC que acompanham o cabeçalho.
- **marca (M)**: ocupa 1 bit. Permite que um evento usado, por exemplo, para definir eventos como fronteiras de um quadro de vídeo.
- **tipo de carga (P - *payload*)**: ocupa 7 bits. Esse campo identifica o formato do tipo de carga RTP que será interpretado pela aplicação, por exemplo, vídeo MPEG, H.261 ou *Motion*-JPEG. O mapeamento do código ao formato do fluxo é normalizado.
- **número de seqüência**: ocupa 16 bits. O valor 1 é adicionado ao número de seqüência para cada pacote de dados RTP enviado e pode ser usado pelo receptor para detectar pacotes perdidos ou restaurar a seqüência de pacotes.
- **marca temporal (horário)**: ocupa 32 bits. esta marca temporal é incrementada com a freqüência de relógio e determinada pelo formato do dado transportado no pacote. Por exemplo, para um áudio à taxa fixa, a marca temporal poderia ser incrementada em um para cada amostra. Vários pacotes RTP consecutivos podem ter marcas temporais iguais se eles são gerados logicamente no mesmo instante (p.e., pertença ao mesmo quadro de vídeo).

- **SSRC**: ocupa 32 bits e identifica a fonte do fluxo de pacotes RTP (fonte de sincronização). Todos os pacotes de uma fonte de sincronização fazem parte um mesmo espaço de número de seqüência e de tempo, de maneira que o receptor pode montar o dado de uma mesma fonte para a apresentação. Este identificador é escolhido aleatoriamente (existem detecção de conflito a este nível). Exemplos de fontes de sincronização incluem o emissor de um fluxo de pacotes derivados de uma fonte de sinal tal como microfone ou câmera, ou um misturador.
- **lista CSRC**: varia de 0 a 15 itens, onde cada item possui 32 bits. A lista CSRC indica quais são as origens que contribuíram para a carga (*payload*) desse pacote. O número de identificadores é determinado através do campo CC. Uma aplicação exemplo é a audio-conferência, onde um misturador indica todos os interlocutores cuja fala foi combinada para produzir o pacote, permitindo que o receptor indique o interlocutor atual, mesmo que todos os pacotes de áudio contenham o mesmo identificar SSRC (que é do misturador).

3.2 Parte de controle do RTP (RTCP)

O protocolo de controle do RTP, chamado de RTCP, fornece suporte para conferência em tempo-real de grupos de qualquer tamanho dentro da Internet. O RTCP é responsável pela transmissão periódica de pacotes de controle para todos os participantes de uma sessão multimídia. O RTCP utiliza o mesmo mecanismo de distribuição de pacotes de dados do RTP (o protocolo UDP).

O RTCP possui as seguintes funções:

- Diagnósticos sobre a qualidade da distribuição dos dados a partir da monitorização de QoS e controle de congestionamento. Os membros da sessão emitem periodicamente relatórios (aproximadamente a cada 5s) para todos os emissores contendo o número de seqüência do último dado recebido, número de pacotes perdidos, e uma medida da variação temporal entre recepções e marcas temporais (necessários à estimação do atraso de transmissão entre emissores e receptores). O controle de congestionamento é feito da seguinte forma: cada participante envia pacotes de controle para todos os outros, portanto cada um

pode independentemente observar o número de participantes, que é utilizado para cálculo da taxa na qual os pacotes devem ser enviados.

- Envio do identificador CNAME (*Canonical NAME*) que é utilizado pelos receptores para manutenção do enlace de cada participante. Os receptores também utilizam o CNAME para associar múltiplos fluxos de dados de um determinado participante dentro de um conjunto de sessões RTP, por exemplo, para sincronizar áudio e vídeo.
- Sincronização intermídia: aplicações que enviaram dados recentemente emitem um relatório que contém informações úteis para a sincronização intermídia: o tempo real e uma correspondente marca temporal. Estes dois valores permite a sincronização de mídias diferentes, como a sincronização labial.
- Identificação da fonte: esta mensagem RTCP contém o *e-mail* do usuário emissor e opcionalmente outras informações (nome, endereço, telefone), pois pacotes de dado RTP não identificam a fonte. Esta função é opcional e é mais utilizada em sessões de controle fraco (*loosely controlled*), onde participantes podem entrar ou sair de uma sessão sem existir um controle dos membros integrantes ou parâmetros de negociação.

Tipos de pacotes RTCP

Existe vários tipos de pacotes de controle de informação RTCP [Xerox, 96]:

- **SR (*Sender Report*)**: são relatórios do emissor, para estatísticas de transmissão e recepção de participantes que são emissores ativos.
- **RR (*Receiver Report*)**: são relatórios do receptor, para estatísticas da recepção de participantes que não são emissores ativos.
- **SDES (*Source DEScription*)**: são relatórios sobre itens que descrevem a origem, incluindo CNAME.
- **BYE (*goodBYE*)**: indica o final da participação
- **APP (*APPlication*)**: são funções específicas da aplicação

O pacote RTCP possui uma parte fixa, similar ao pacote de dados RTP, mas a estrutura dos elementos varia de acordo com o tipo de pacote transmitido. Vários

pacotes RTCP podem ser anexados dentro de um, formando um pacote composto, que é enviado (como se fosse único) pelo protocolo da camada inferior.

Formato do pacote SR

O pacote SR é formado basicamente por três campos: cabeçalho, informação da origem e blocos de relatórios, conforme figura 3.3. Pode existir um quarto campo se a aplicação definir uma extensão do protocolo.

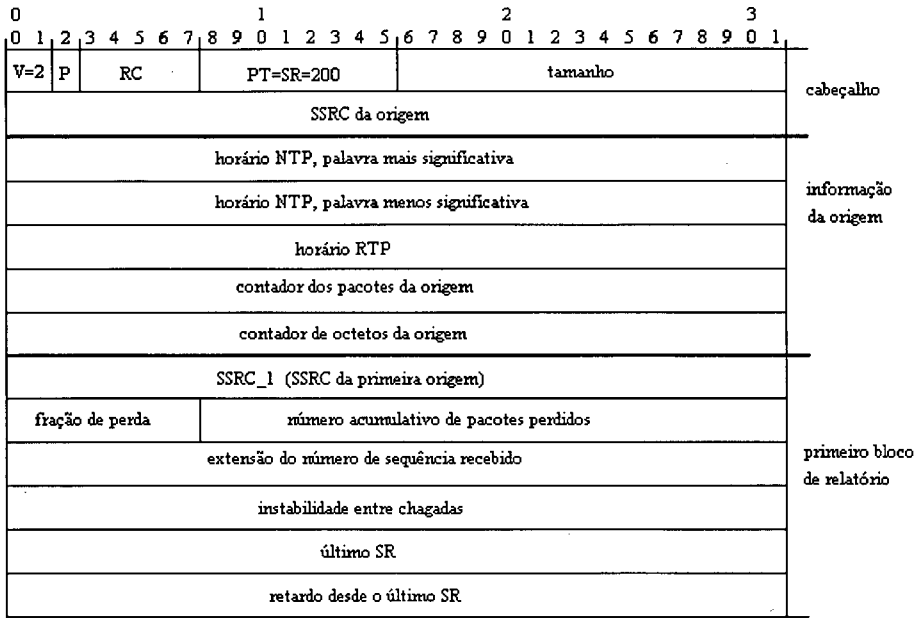

Figura 3.3 Formato do pacote SR

O cabeçalho do pacote SR possui os seguintes campos:

- **versão (V)**: é um campo que ocupa dois bits e identifica a versão do protocolo RTCP, que é a mesma versão dos pacotes de dados RTP. A versão utilizada é a 2.
- **preenchimento (P)**: ocupa um bit. Quando P=1 significa que o pacote RTCP contém alguns octetos de preenchimento, adicionados no fim do pacote, que não fazem parte da informação de controle. O último octeto de preenchimento contém o número de todos os octetos de preenchimento que serão ignorados.
- **contador de relatórios de recepção (RC)**: ocupa 5 bits. Contém o número de blocos de relatórios de recepção contidos nesse pacote. Zero é um valor válido.

- **tipo do pacote (PT)**: ocupa 8 bits. Contém a constante 200 que indica que é um pacote SR.
- **tamanho**: ocupa os 16 bits restantes e indica o tamanho do pacote RTCP (quantidade de palavras de 32 bits menos um) incluindo o cabeçalho e preenchimento.
- **SSRC**: ocupa 32 bits. É o identificador de sincronização da fonte que originou o pacote SR.

Os seguintes campos fazem parte da informação da origem:

- **horário NTP**: ocupa 64 bits. Indica a hora real em que o relatório foi enviado.
- **horário RTP**: ocupa 32 bits. Corresponde ao mesmo tempo do horário NTP, mas com a mesma unidade e o com o mesmo *offset* aleatório da marca temporal dos pacotes de dados RTP.
- **contador dos pacotes da origem**: ocupa 32 bits e indica o número total de octetos de dados (incluindo cabeçalho e preenchimento) transmitidos nos pacotes de dados RTP desde que a origem iniciou a transmissão até o tempo que o pacote SR foi gerado. Esse contador é zerado quando a origem modifica seu identificador SSRC.

Podem existir zero ou mais blocos de relatórios de recepção. Cada bloco de relatório de recepção contém estatísticas sobre os pacotes RTP recebidos de cada fonte de sincronização. Os blocos contém as seguintes informações:

- **SSRC_n (identificador da origem)**: ocupa 32 bits. O bloco de relatórios de recepção pertencente a uma origem identificada através do SSRC_n.
- **fração de perda**: ocupa 8 bits. A fração de pacotes de dados que a origem SSRC_n perdeu desde que o prévio pacote SR ou RR foi enviado. Definida como sendo o número de pacotes perdidos dividido pelo número de pacotes esperados.
- **número cumulativo de pacotes perdidos**: ocupa 24 bits. É o número total de pacotes de dados RTP que partiram da origem e foram perdidos desde o início da recepção.

- **extensão do número de seqüência recebido**: ocupa 32 bits. Diferentes receptores que participam da mesma sessão podem gerar diferentes extensões do número de seqüência se eles iniciarem a aplicação em tempos significativamente diferentes.
- **instabilidade entre chegadas**: ocupa 32 bits. É uma estimativa da variância estatística do tempo entre chegadas do pacotes de dados RTP, medidas em unidades de tempo e expressas como inteiro não sinalizados.

Formato do pacote RR

A figura 3.4 mostra o formato do pacote RR, que é idêntico ao formato do pacote SR, exceto no campo tipo do pacote que contém a constante 201, e nas cinco palavras da informação da origem que são omitidas.

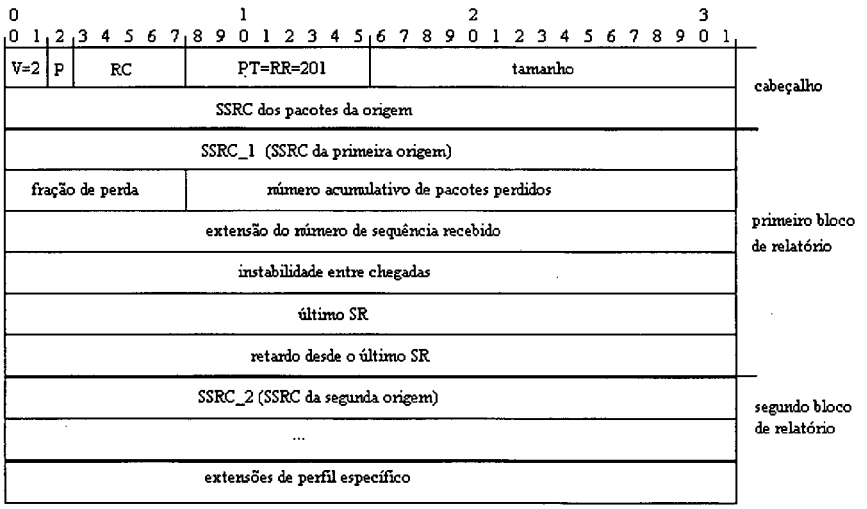

Figura 3.4 Formato do pacote RR

Formato do pacote SDES

O pacote SDES, visualizado na figura 3.5, é uma estrutura composta por um cabeçalho e zero ou mais trechos de mensagem. Esses trechos são compostos de itens que identificam a origem.

Além dos campos versão, preenchimento e tipo de pacote, o cabeçalho do pacote SDES possui o campo contador da origem (SC), que ocupa 5 bits e indica o número de pedaços SSRC/CSRC que o pacote SDES possui. O valor zero para o campo SC é aceito mas não é utilizado.

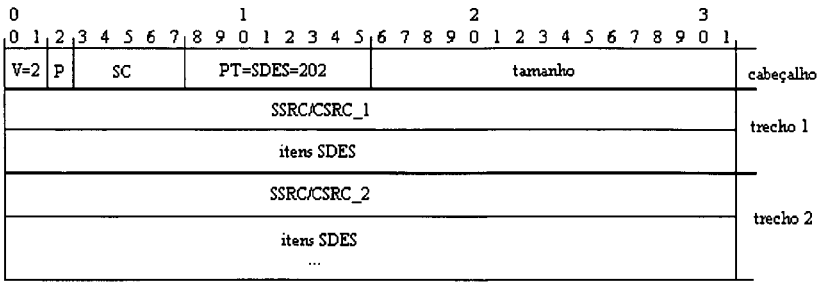

Figura 3.5 Formato do pacote SDES

Cada pedaço é composto de um identificador SSRC/CSRC seguido de uma lista de zero ou mais itens, que levam informação sobre o SSRC/CSRC.

Os tipos de itens atualmente definidos pelo pacote SDES podem ser visualizados na tabela da figura 3.6. Desses, apenas o CNAME é obrigatório. Alguns são utilizados apenas por implementações particulares, e ítens adicionais podem ser definidos, mas é necessário obter um registro com o IANA.

Abreviação	Nome	Valor
END	Finaliza uma lista SDES	0
CNAME	Nome canônico	1
NAME	Nome do usuário	2
EMAIL	Endereço eletrônico do usuário	3
PHONE	Número do telefone do usuário	4
LOC	Localização geográfica do usuário	5
TOOL	Nome da aplicação ou ferramenta	6
NOTE	Notificação sobre a origem	7
PRIV	Extensões privadas	8

Figura 3.6 Tipos de itens do pacote SDES

Formato do pacote BYE

O pacote BYE indica que uma ou mais origens não estão mais ativas. Como pode ser visualizado na figura 3.7, além dos campos versão, preenchimento e tipo de pacote, o pacote BYE possui o campo contador da origem (SC), que indica o número de identificadores SSRC/CSRC incluídos no pacote BYE.

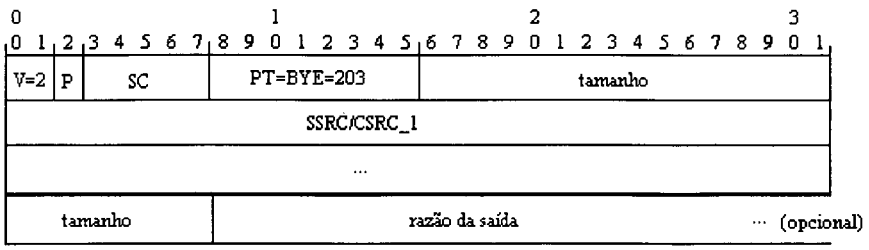

Figura 3.7 Formato do pacote BYE

Formato do pacote APP

O pacote APP, que pode ser visualizado na figura 3.8, serve para uso experimental de novas aplicações e desenvolvimento de novas características, sem precisar requerer um valor de registro para o tipo de pacote.

Além dos campos versão, preenchimento, tamanho e tipo do pacote, o pacote APP possui os seguintes campos:

- **subtipo**: ocupa 5 bits. Pode ser usado como um subtipo para permitir que um conjunto de pacotes APP sejam definidos sobre um único nome.
- **nome**: ocupa 4 octetos. É um nome escolhido pela pessoa que definiu o conjunto de pacotes APP, que deve ser único com relação aos outros pacotes APP que a aplicação irá receber.
- **data que depende da aplicação**: possui tamanho variável e pode ou não aparecer em um pacote APP. Esse campo é interpretado pela aplicação e não pelo RTP. Deve ter um tamanho múltiplo de 32 bits.

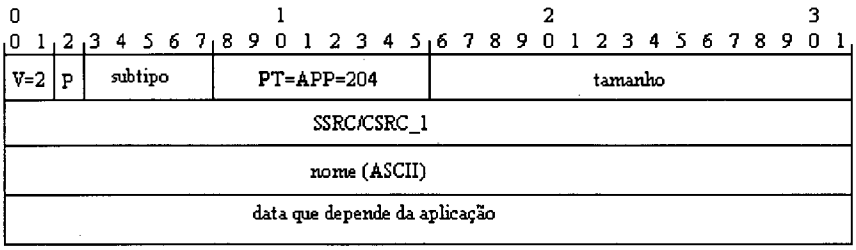

Figura 3.8 Formato do pacote APP

3.3 Aplicações Adaptativas

A maioria das aplicações multimídia utilizadas na Internet usam os protocolos de transporte RTP e UDP. Porém, nenhum desses dois protocolos conseguem garantir um nível de qualidade desejado. Existem duas maneiras de contornar esse problema: reserva de recurso e controle da aplicação.

A reserva de recurso aloca recursos da rede procurando garantir o nível de QoS necessário para a execução da aplicação. Isso nem sempre é possível porque os sistemas multimídia distribuídos são heterogêneos, envolvendo muitos tipos de comunicação e processamento. Por exemplo, quando uma aplicação de videoconferência é executada em um tipo de ambiente, usuários em estações de trabalho conectados diretamente a

LANs receberão vídeos com menor retardo do que aqueles conectados através de modems em microcomputadores. Portanto, nem todos os nodos da rede têm condições de suportar uma reserva de recursos pré-estabelecida.

A segunda maneira é construir aplicações mais tolerantes às inevitáveis flutuações de desempenho da rede. Nesse caso, o controle deve ser passado à aplicação que deve procurar se adaptar as condições da rede. Estas aplicações adaptativas podem utilizar relatórios emitidos pelo protocolo RTCP para obter informações sobre a variação da carga da rede e adaptar a taxa de bits gerada por uma mídia contínua em resposta à ocorrência de possíveis retardos.

3.3.1 Formas de adaptação

Desprezando o termo "aplicação adaptativa", alguns programas de computadores tem sido projetados para funcionarem em ambientes heterogêneos, podendo adaptar-se ao *hardware* e ao sistema operacional utilizado. Esses programas podem reconhecer automaticamente os recursos disponíveis ou isso pode ser feito manualmente durante o instante da instalação. "Um outro tipo de adaptatibilidade inclui a opção de negociação para o reconhecimento de parâmetros de componentes remotos (tal como terminais e modems) e a reconfiguração do programa em sistemas tolerantes a falhas" [Gecsei 97].

Uma adaptação pode ser estática ou dinâmica:

- A adaptação é estática quando ativada na instalação ou durante o instante da inicialização do sistema. Na instalação, quando o sistema verifica os recursos disponíveis e se autoconfigura para utilizá-los. Durante o tempo de inicialização do sistema, quando o usuário passa para a aplicação os parâmetros que correspondem à qualidade de transmissão desejada. Se o sistema não puder atender estes requisitos, a aplicação informa ao usuário quais são os recursos disponíveis que então pode utilizar esta informação para configurá-la.
- A adaptação dinâmica funciona continuamente durante a execução do programa, ou seja, a aplicação se adapta de acordo com as flutuações do sistema.

A figura 3.9 representa, de uma maneira simplificada, um mecanismo de adaptação dos sistemas multimídia distribuídos.

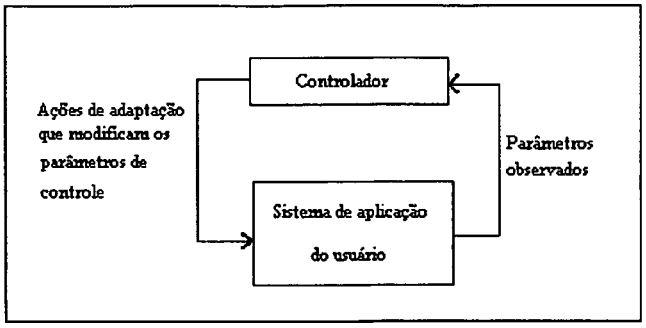

Figura 3.9 Mecanismo de adaptação básico de um sistema multimídia distribuído [Gecsei 97]

A adaptação em um sistema multimídia distribuído pode ser feita de duas formas:

- **Com alimentação (*feedback*)**: existe a modificação de alguns parâmetros durante o tempo de execução da aplicação. Por exemplo, quando os relatórios RTCP recebidos pela aplicação identificam um aumento do número de pacotes perdidos, a aplicação pode diminuir a taxa de transmissão (reduzindo o congestionamento) e, como conseqüência, diminuir também o número de pacotes perdidos.
- **Sem realimentação**: o controle da ação não tem impacto direto sobre o sistema. Nos sistemas MOST (*Mobile Open Systems Technologies*) [Gecsei 97], por exemplo, quando são observadas mudanças na conectividade da rede, a aplicação modifica sua funcionalidade de acordo com a interface do usuário sem a tentativa de influenciar a conectividade. Portanto, quaisquer que seja a modificação que a aplicação realize, ela não influenciará nos recursos disponíveis, aumentando largura de banda, ou reduzindo congestionamento.

3.3.2 Elementos de adaptação

Existem elementos que permitem identificar diferenças nos tipos de adaptatibilidade existentes. São eles [Gecsei 97]:

- a forma de adaptação utilizada: que pode ser estática ou dinâmica;

- pela posição do componente responsável por disparar e controlar a adaptação que pode estar localizado na interface do usuário, na aplicação ou em outra parte do sistema;
- com o algoritmo de adaptação utilizado;
- através das regras definidas pelo usuário quando o sistema possibilita que o usuário escolha o tipo de ação que deve ser realizada.

Além dos elementos acima, existe uma grande variedade de parâmetros que influenciam no funcionamento dos mecanismos de adaptação [Gecsei 97]:

- **parâmetros relativos ao sistema**: sistema operacional, características de hardware (quantidade de memória, resolução de vídeo, velocidade da CPU) e fatores de comunicação (taxa de pacotes, taxa de erro/perda, escolha do protocolo).
- **parâmetros relacionados com a aplicação**: *layout* da tela, taxa de quadros de vídeo, velocidade do banco de dados e latência.
- **parâmetros relacionados com o usuário**: grau de satisfação, tempo de resposta e custo.

3.3.3 Tipos de aplicações adaptativas

Um sistema multimídia distribuído é formado basicamente por três componentes: usuário, aplicação e sistema. Destes, o último inclui comunicação, banco de dados, hardware e sistema operacional. Em tese, o controle pode ser feito por qualquer um desses componentes [Gecsei 97].

Existe três tipos de aplicações adaptativas: centralizadas no usuário, no sistema ou em ambos.

Aplicações adaptativas centralizadas no usuário normalmente controlam a adaptação, são estáticas, não fornecem realimentação e dão ênfase à interface e à interação com o usuário. Um mecanismo básico de adaptação (que pode funcionar na aplicação e na interface usuária), monitora a conectividade da rede relatando para a aplicação os recursos disponíveis do sistema. Por exemplo, no período de baixa conectividade nenhum gráfico é transmitido entre *hosts* e cada *host* tem uma cópia do

mapa que é atualizada no período de alta conectividade. Nesses sistemas a aplicação modifica a interface mantendo o usuário informado sobre o estado atual da infraestrutura de comunicação. Um exemplo de aplicação multimídia centralizada no usuário é o MOST.

Aplicações centralizadas no sistema utilizam sistemas de controle de realimentação na tentativa de manter os parâmetros do sistema num nível constante. Não existe um interesse fundamental na interface com o usuário. Por exemplo, o *INRIA Conferencing System* (IVS) [Gecsei 97] foi projetado para funcionar sobre a Internet utilizando o protocolo UDP. A transmissão no estilo melhor-esforço não pode garantir níveis estáveis de rendimento. O IVS possui mecanismos que detectam o período de congestionamento da rede. Quando ocorre congestionamento, o processo de codificação de vídeo diminui a taxa de dados piorando a qualidade da imagem transmitida, mas mantendo a transmissão.

Aplicações mistas são aquelas que observam parâmetros tanto do usuário como do sistema. Normalmente, a entidade de adaptação é a aplicação. Esse esquema será apresentado usando como exemplo um sistema de notícia sob demanda. Quando o usuário seleciona um documento iniciam-se as negociações que levam em conta as preferências do usuário (formato de mídia, qualidade e limitações de custo), os elementos de mídia disponíveis, o custo entre diferentes servidores e conectividade entre o cliente e todos os servidores envolvidos. Esse processo gera um documento que é apresentado ao usuário. Procurando otimização, o usuário deve selecionar e configurar o conjunto mais apropriado de componentes do sistema (tal como *links* e servidores). Se ocorrerem mudanças de conectividade, dispara-se a renegociação, reconfigurando o sistema.

3.4 Escalamento de Mídias

Como visto anteriormente, de acordo com os parâmetros de desempenho do sistema de comunicação, as aplicações adaptativas devem alterar a taxa de bits gerada pelo áudio e vídeo. Isto é feito alterando-se os parâmetros de qualidade das mídias durante a conexão multimídia. Quando aplicado ao fluxo de dados multimídia, chama-se escalamento de mídia ou degradação suave da qualidade.

Os protocolos orientados a conexão tradicionais têm sempre suportado um método simples de escalamento de mídia, chamado de controle de fluxo, evitando que um emissor rápido sobrecarregue um receptor lento. Note-se que o único parâmetro controlado por estes algoritmos é a taxa de dados. Isto é razoável para conexão não tempo-real (por exemplo, *e-mail*, transferência de arquivo, etc.), mas ele obviamente dificulta a transferência isócrona de fluxos tempo-real.

Escalamento de mídia permite o controle de outros parâmetros além da taxa de dados. Em um fluxo de vídeo, a qualidade da imagem é um candidato óbvio.

Para implementar escalamento de mídia, a interface entre a aplicação e a rede deve ser estendida para passar informações de controle para cima e para baixo. Por exemplo, se a rede sinaliza sobrecarga, um codificador de vídeo MPEG poderia ajustar o tamanho de passo de quantificação: quanto maiores os passos de quantificação, menor será a taxa de dados e a qualidade da imagem.

Muitos projetos de pesquisa estudam o escalamento de mídia ([Käppner, 94], [McCanne, 96], [Wittig, 94]), mas não há padrões ainda para interface aplicação/rede ou para algoritmos e protocolos dentro da rede.

3.4.1 Qualidade de Vídeo

Para se adaptarem aos recursos de comunicação disponíveis, as aplicações adaptativas que representam imagens de vídeo podem aumentar a compressão dos dados, reduzindo a qualidade da imagem ou até mesmo diminuir a freqüência de exibição dos quadros.

Para poder entender as mudanças que as aplicações adaptativas podem realizar sobre o vídeo capturado, é necessário conhecer os parâmetros que influenciam na qualidade de reprodução do vídeo: os tipos de imagem, as resoluções disponíveis e a freqüência de exibição de quadros.

Tipos de Imagens

Existem três tipos de imagem que podem ser utilizadas de acordo com os recursos de comunicação disponíveis:

- **imagem estática**: não existe movimento. Utilizadas para prender a atenção do telespectador ou para mostrar informações com um certo grau de detalhamento. Podem ser usadas na demonstração de uma radiografia médica, um diagrama elétrico, uma planta baixa de um imóvel ou um desenho qualquer.
- **imagem com pouco movimento**: utilizadas em videoconferência, caracterizando-se por apresentar quase sempre um cenário estático por trás de um personagem cujos movimentos são corporais (praticamente os da cabeça, expressões faciais e membros).
- **imagem de ação**: são aquelas em que existe um grande número de objetos em cena, e esses objetos estão em constante movimento, podendo ocorrer mudanças abruptas da cena, composta de outros objetos diferentes.

Na representação de uma imagem estática, podem-se usar arquivos, que serão transmitidos aos interessados, nos formatos BMP, JPEG, GIF, etc. Estas imagens podem ser transmitidas por difusão, com grandes resoluções, ocupando bem menos largura de banda do que se fosse utilizado um vídeo em tempo real.

Na representação de imagens com pouco movimento pode-se experimentar a transmissão de média qualidade, com imagens que possuem resoluções mais baixas e taxa de exibição de quadros menores.

Como as imagens de ação apresentam uma maior quantidade de movimento, elas exigem boa qualidade, boa resolução e alta taxa de exibição de quadros para poder reproduzir melhor o que a origem está transmitindo.

Resoluções disponíveis

Por questão de interoperabilidade, foram especificadas cinco resoluções para a apresentação das mídias de vídeo. A resolução CIF *(Comum Intermediate Format)* com 352x288 *pixels*, mais dois submúltiplos, as resoluções QCIF (*Quarter* CIF) com 176x144 *pixels* e SQCIF (*Sub-Quarter* CIF) com 128x96 *pixels*, e mais dois múltiplos, as resoluções 4CIF (*four times* CIF) com 704x576 *pixels*, e 16 CIF (*sixteen times* CIF), com 1408x1152 *pixels*. A figura 3.9 demonstra o espaço ocupado pelas resoluções padrão SQCIF, QCIF, CIF e 4CIF em telas de 800x600 e 1024x768 *pixels*. A resolução 16CIF não está representada porque extrapola os limites da tela.

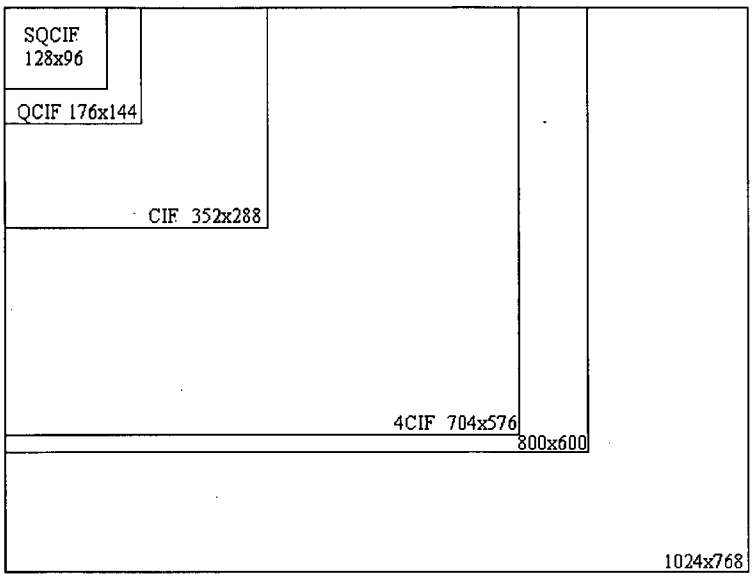

Figura 3.10 Espaço ocupado pelas resoluções padrão SQCIF,QCIF, CIF, 4CIF em telas de 800x600.

Freqüência de Exibição de Quadros

A freqüência ótima para exibição de quadros de vídeo é de 30 fps (*frames per second*) e é obtida nos vídeos reproduzidos em aparelhos de televisão. Valores abaixo de 15 fps causam desconforto para os espectadores em períodos prolongados de tempo, sendo uma fonte adicional de desestímulo.

3.4.2 Elementos Responsáveis pelo Escalamento de Mídias

Os seguintes elementos são responsáveis pelo escalamento de mídias:

- padrões de compressão de vídeo
- técnicas de compressão de vídeo
- protocolos de transporte
- tradutores e filtros

Padrões de compressão

Existem várias técnicas de codificação de mídias de vídeo digital e a maioria delas apresenta uma perda de fidelidade em relação ao vídeo original. Esta perda é representada através de um parâmetro de quantificação sendo que quanto menor for esse parâmetro, menor será a compressão sobre os quadros, melhor será a fidelidade da imagem e maior é a largura de banda utilizada na transmissão.

"Os codificadores projetados para a Internet requerem grande escalabilidade, baixa complexidade computacional, baixa latência de codificação e descodificação, e devem ser resilientes às perdas da rede. Eles também devem ser fortemente vinculados ao *software* que envia para a rede, alcançando a maior taxa de quadros e qualidade da imagem possível" [Hunter, 98].

Os codificadores mais importantes adotados pela indústria para envio do fluxo de vídeo são: H.261, H.263, MPEG-1, MPEG-2 e MPEG-4. Esses padrões reduzem o tamanho dos dados de vídeo através da compressão e facilitam a criação, transmissão e apresentação da informação multimídia, permitindo compatibilidade entre as várias tecnologias. A tabela 9.10 apresenta um comparativo contendo as principais características desses padrões.

Padrão	Formato	Taxa de bits	Aplicações
H.261	QCIF/CIF	P*64 Kbps, P=1,2,...,30	Videofone e videoconferência, etc.
H.263	SQCIF-16CIF	Flexível	Desde videofone a baixa taxa de bits até Videoconferência de alta qualidade, etc.
MPEG-1	SIF	1,5 Mbps	Armazenamento, entrega de vídeo com qualidade de de VCR em CD-ROM, etc.
MPEG-2	Flexível	> 2Mbps	Distribuição de TV a cabo, difusão de serviços via satélite, cinema eletrônico, etc.
MPEG-4	Flexível	Flexível	Multimídia interativa, multimídia móvel, distribuição de vídeo sobre a intranet e Internet, etc.

Tabela 3.11 Padrões de codificação de vídeo

Dentre os padrões de codificação de vídeo digital destacam-se o H261 e o H263, que serão definidos a seguir.

H.261

O padrão H.261 também é conhecido como P*64, onde P é um a média de um número inteiro que representa 64Kbits/sec. Ele foi projetado para ser utilizado em aplicações de teleconferência, com a futura intenção de enviar vídeo sobre ISDN - particularmente para aplicações de videofone face-a-face e videoconferência.

O algoritmo de codificação atual é similar, mas incompatível com o do MPEG. O algoritmo possui um mecanismo que otimiza o uso da largura de banda, quando não ocorre movimento na imagem; uma figura que sofre mudanças na imagem tem qualidade inferior à daquela que permanece com a imagem estática. H261 precisa de menos poder de CPU para codificação em tempo real que o MPEG.

H.263

É um padrão do ITU-T projetado para comunicação com baixa taxa de dados. Espera-se que ele seja usado em várias faixas de taxa de bits e não apenas em aplicações com taxa de bits baixa, na medida que o H.263 deva substituir o H.261 em muitas aplicações. Isto, pois apesar do algoritmo de codificação do H.263 ser similar ao algoritmo usado pelo H.261, ele apresenta algumas melhorias e mudanças para fornecer melhor desempenho e reconhecimento de erro.

Para aplicações de videoconferência em especial, o padrão H.263 apresenta um maior número de recursos destinados à transmissão sobre enlaces que oferecem quantidades limitadas de largura de banda, apresentando características de qualidade comparáveis ao padrão MPEG. O H.263 é melhor que o MPEG-1/MPEG-2 para baixas resoluções e baixas taxas de bit. O H.263 é menos flexível que o MPEG, mas requer muito menos sobrecarga (*overhead*).

Se o padrão for seguido, só é possível utilizar H.263 com certas resoluções: SQCIF, QCIF, CIF, 4CIF e 16CIF.

3.5 Conclusão

As aplicações adaptativas, através da adaptação da taxa de bits gerada pelas fontes de áudio e vídeo de acordo com os parâmetros de desempenho da rede, permitem a continuidade de apresentação dos fluxos multimídia em redes do tipo melhor esforço (como a Internet). Visivelmente, a qualidade da apresentação das mídias será piorada se a rede estiver muito congestionada. Em caso de congestionamento intenso, a apresentação será interrompida.

O próximo capítulo apresenta a rede MBone (*Multicast Backbone*), onde serão apresentadas algumas aplicações de áudio e vídeo adaptativas.

4. Multicast Backbone (MBone)

MBone é a abreviação de *Virtual Internet Backbone for Multicast IP* ou *Multicast Backbone.* Ela é uma rede virtual dentro da camada física da Internet construída como uma estrutura de teste do IP *multicast.* O MBone foi criado como uma rede experimental do protocolo IP *Multicast.*

A MBone iniciou em março de 1992, durante o encontro do *Internet Engineering Task Force* (IETF) em San Diego, USA, quando muitos eventos do encontro foram transmitidos em áudio, utilizando transmissões de pacotes de difusão seletiva do site IETF, para participantes de 20 sites em três continentes, abrangendo 16 zonas horárias [Casner, 92] [Casner, 95]. Foram transmitidas todas as sessões gerais do encontro, e também algumas sessões de grupos de trabalho. Os participantes remotos tinham a possibilidade de falar, participando das discussões e fazendo perguntas, e, embora a transmissão não fosse perfeita, foi razoável em ambas as direções. No site do IETF, e na maioria dos sites remotos, foi utilizada a ferramenta VAT, *Visual Audio Tool* (vista mais adiante), desenvolvida por Van Jacobson e Steve McCanne, do *Lawrence Berkeley Laboratory.* Foi usada a velocidade de 64 Kbps para áudio codificado com PCM.

A transmissão do evento foi possível graças a utilização de IP *multicast* implementado nos roteadores e *hosts*, onde uma cópia simples é enviada de cada pacote, sendo replicada apenas quando houver um braço na árvore lógica dos destinos. Se fosse necessário enviar uma cópia de cada pacote para cada destino, a largura de banda e processamento requerido seriam proibitivos.

Nos dias atuais, conferências científicas, propagandas, publicações de pesquisas e outras são transmitidas para o mundo todo através do MBone. Basta estar conectado a um nó de uma rede já ligada ao MBone para receber esse tipo de transmissão.

Limitações da Internet atual restringem a qualidade do áudio e vídeo recebido pelo MBone. Mesmo assim, tanto a qualidade de áudio como de vídeo são adequados para a finalidade de conferência. As taxas de vídeos geralmente não excedem de 8 quadros por segundo, e a qualidade auditiva é equivalente a uma conexão de telefone

boa. Futuramente, as teleconferências no MBone melhorarão muito com o desenvolvimento do ATM, que é uma tecnologia que permite a especificação do nível de desempenho de áudio/vídeo.

4.1 Como o IP Multicast Trabalha

Multicast possibilita a distribuição de pacotes IP para um ou mais lugares na Internet, sem precisar fazer *broadcast* para todos os *hosts Internet*. Portanto, *multicast* IP é a transmissão de um datagrama IP para um "grupo de *hosts*", constituído de um conjunto de zero ou mais *hosts* identificados por um simples endereço destino IP. Um datagrama de difusão seletiva IP é enviado para todos os membros do grupo destino da mesma forma que um simples datagrama IP é enviado, ou seja, não existe garantias contra perda, retardo ou entrega desordenada dos datagramas para nenhum membro do grupo destino.

4.1.1 Endereçamento Multicast IP

Os endereços IP são números com 32 bits, onde a primeira parte do endereço identifica uma rede e a segunda parte um *host* pertencente a essa rede. Os endereços IP possuem quatro classes diferentes, A, B, C e D, que podem ser visualizados na figura 4.1.

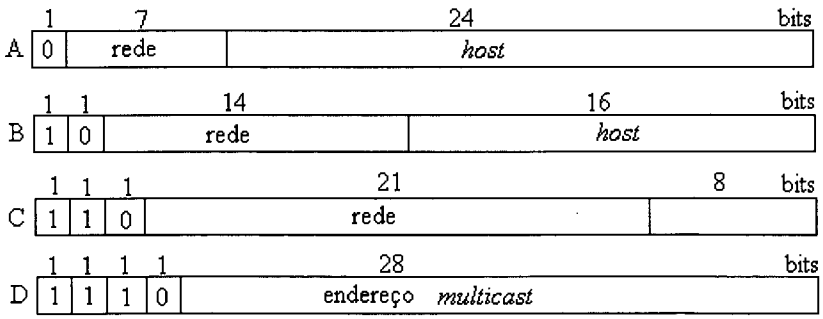

Figura 4.1 Classes de endereços IP

Um endereço classe A é utilizado para redes de grande porte e pode endereçar cerca de 16 milhões de *hosts*. O endereço classe B varia na faixa de 128.1 à 191.255, já que o número 127 do primeiro octeto e os números 0 e 255 do segundo octeto são usados para funções especiais. Endereços classe C atendem tipicamente a faixa das redes locais.

Endereçamento classe D é utilizado para difusão seletiva IP (*multicast IP*), utilizado pela rede MBone.

No padrão Internet, endereços *multicast* estão na faixa de 224.0.0.0 a 239.255.255.255. O endereço 224.0.0.0 é reservado e não pode ser usado por nenhum grupo *multicast*. O endereço 224.0.0.1 é designado para o grupo permanente de todos os *hosts* IP (incluindo *gateways*). Esta forma é usada para endereçar todos os *hosts* de difusão seletiva que estão conectados diretamente à rede. Os endereços a partir de 224.0.0.2 até 224.0.0.255 são reservados para protocolos de roteamento. Portanto, apenas os endereços a partir de 224.0.1.0 até 239.255.255.255 estão abertos para uso geral. Um subconjunto pequeno desses endereços classe D é alocado para conferência multimídia na Internet, constituindo o MBone.

4.1.2 Grupos Multicast

Um endereço *multicast* IP é associado a um conjunto de receptores que formam um grupo *multicast* IP. Cada grupo *multicast* individual pode ser identificado por um endereço particular IP Classe D. Cada *host* pode se registrar como membro de um grupo *multicast* através do uso do *Internet Group Management Protocol* (IGMP).

Somente os *hosts* interessados em um serviço *multicast* terão os dados roteados até si. Para isso, os *hosts* devem pertencer a um determinado grupo, formando uma lista dinâmica de membros, ou seja, *hosts* podem se juntar e abandonar os grupos a qualquer momento. Não há restrição de localização ou de número de membros em um grupo de *hosts*. Um *host* pode pertencer a mais de um grupo simultaneamente e não precisa pertencer a um grupo para enviar datagramas para ele.

Um grupo de *hosts* pode ser permanente ou transiente. Um grupo permanente está associado a um endereço IP conhecido. É o endereço, e não a lista de membros do grupo, que é permanente. Em um determinado tempo, um grupo permanente pode ter qualquer número de membros, até mesmo zero. Aqueles endereços de difusão IP não reservados para grupos permanentes estão disponíveis para designação dinâmica de grupos transientes. Grupos transientes existem apenas durante o tempo em que possuírem membros.

4.1.3 Protocolo IGMP

Para um *host* participar de difusão seletiva local, precisa de um software responsável por enviar e receber pacotes de difusão seletiva. Se este *hosts* quiser participar de difusão seletiva com outras redes, deve informar isto ao roteador de difusão seletiva local, também conhecido como *mrouter* (*multicast router*). O *mrouter* se encarregará de enviar/receber pacotes de difusão seletiva para os roteadores vizinhos. IGMP é mais utilizado entre roteadores *multicast* para trocar informações sobre os registro de associação em grupos.

O IGMP deve ser implementado em todos os *hosts*, seguindo o nível 2 da especificação de difusão seletiva IP [Stanford 89]. O IGMP estende o cabeçalho IP, adicionando campos para o envio de suas mensagens. O identificador de protocolo no cabeçalho IP recebe o valor 2, para indicar se tratar de uma PDU IGMP.

Quando um *host* deseja se juntar a um grupo, envia uma mensagem *IGMP_Join_Group,* informando ao *mrouter* que quer receber transmissão daquele endereço de grupo. Quando o *host* deseja sair de um grupo, envia uma mensagem *IGMP_Leave_Group,* informando ao *mrouter* que não quer mais receber as transmissões daquele grupo. Roteadores questionam periodicamente a LAN a fim de determinar se os membros conhecidos do grupo continuam ativos. Se houver mais que um roteador executando difusão seletiva IP sobre uma mesma LAN, um desses roteadores deve ser eleito como "questionador", e assumir a responsabilidade de perguntar à LAN sobre os estados dos membros do grupo.

Utilizando a informação de associação em grupos, obtida pelo IGMP, um roteador pode determinar (se houver) qual o tráfego necessário para enviar datagramas de difusão seletiva a cada membro do grupo pertencente à sub-rede. Os *mrouters* usam essa informação em conjunto com um protocolo de roteamento de difusão seletiva, a fim de dar suporte à difusão seletiva IP na Internet.

4.2 Topologia do MBone

O MBone é composto de sub-redes suportando *multicast* conectadas entre si através de caminhos de comunicação virtuais, conhecidos como túneis. Esses túneis são caminhos virtuais que estabelecem uma comunicação ponto a ponto entre roteadores. Roteadores que não suportam a forma de endereçamento *multicast* utilizam datagramas *multicast* encapsulados dentro de datagramas *unicast*. Sendo assim, a comunicação entre dois roteadores MBone é *unicast*, mas a comunicação entre um roteador MBone e a sub-rede em que ele está conectado é *multicast*. Para retransmitir dados do MBone para a sua sub-rede interna, o roteador utiliza um protocolo de roteamento *multicast*.

As rotas *multicast* são guardadas pelos roteadores, que constróem uma árvore de distribuição mapeando caminhos que partem de cada remetente, e alcançando todos os receptores. Atualmente, existem vários protocolos de roteamento que podem construir árvores de distribuição *multicast,* podendo-se citar: *Distance Vector Multicast Routing Protocol* (DVMRP), *Multicast Open Shortest Path First* (MOSPF), *Protocol-Independent Multicast* (PIM).

A topologia de MBone, que pode ser visualizada na figura 4.2, e o escoamento de sessões de difusão seletiva, devem ser administradas ativamente pela comunidade MBone, para minimizar o congestionamento da rede. Dessa forma, a topologia ideal é quando em cada sub-rede existe apenas um ponto conectado ao MBone. Um ponto minimiza o congestionamento e é suficiente para que a sub-rede configurada atinja várias outras sub-redes.

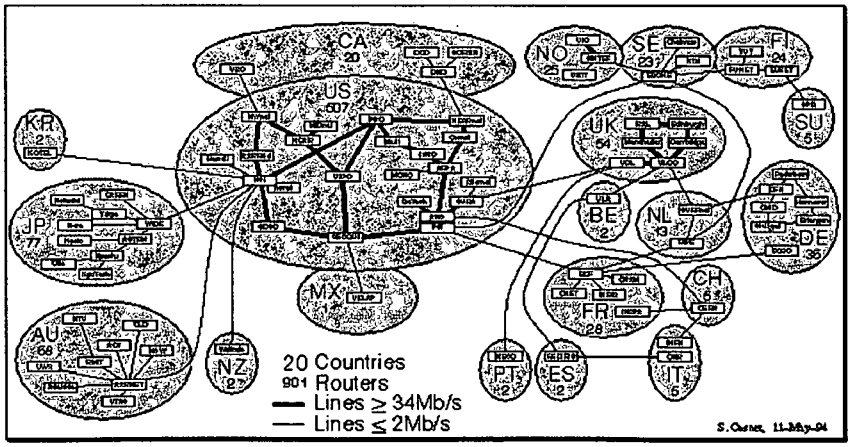

Figura 4.2 Topologia MBone em 1994

4.2.1 Túneis

O MBone está apoiado operacionalmente no mecanismo de túneis diretos, um esquema responsável pelo envio de pacotes *multicast* entre ilhas de sub-redes MBone. Neste esquema, os roteadores IP da Internet (roteadores *unicast*) que não suportam *multicast* não interrompem o tráfego *multicast*. Isso é feito através encapsulação de pacotes *multicast* em pacotes IP comuns. O encapsulamento dos pacotes IP *multicast* em pacotes IP *unicast* possibilita que os nodos físicos pertencentes a um túnel possam compreender esses pacotes.

Quando um *host* transmite um datagrama *multicast* dentro da sua rede local, este datagrama alcançará todos os membros do grupo pertencentes à esta rede local. O parâmetro TTL (*time-to-live*) do protocolo IP permite que o usuário especifique até onde o tráfego *multicast* pode chegar, ou seja, por quantos roteadores ele poderá passar.

Cada túnel possui uma métrica associada e um limiar (*threshold*) que permite truncar o tráfego *multicast*. A métrica especifica o custo do roteamento e o *threshold* é o TTL (*time-to-live*) mínimo que um datagrama precisa para ser transmitido por um dado túnel. Cada pacote enviado de um cliente para a rede possui um TTL específico, decrementado em uma unidade a cada *mrouter* que o pacote passa. Se o TTL de um pacote apresentar um valor menor que o *threshold* do túnel (construído pelo *mrouted*) por onde ele está passando, descarta-se o pacote. Por exemplo, se os datagramas *multicast* IP são enviados com um TTL de 1, ficarão apenas no limite da sub-rede local, não sendo repassados para outras sub-redes. O valor 16 para o campo TTL de um grupo limita o tráfego da difusão seletiva para redes dentro de organizações, e os valores de 127 ou 255 são usados para atingir a rede MBone inteira.

Nas fases iniciais do MBone, pacotes IP *multicast* foram roteados em cima do que são chamado túneis truncados. No esquema de *multicast* truncado, grupos de pacotes *multicast* foram remetidos para roteadores *multicast* vizinhos, contanto que o TTL do pacote fosse maior que o valor de *threshould* do *mrouter*. Como resultado, nodos de *mrouter* que não expressavam interesse em receber o tráfego do MBone associado a um certo grupo *multicast* ainda o recebiam, e por esta razão roteavam o tráfego, já que ele estava dentro do alcance da TTL do remetente. Apesar desse não ser

um esquema muito eficiente, forneceu um mecanismo de teste inicial muito útil para o MBone.

Hoje, a maioria dos túneis é podada. A poda em *multicasting* é conhecida como *multicasting* verdadeiro, onde pacotes de *multicast* não são remetidos aos roteadores *multicast* vizinhos a menos que eles expressem um interesse explícito em receber pacotes (através de mensagens de IGMP) pertencendo àquele *multicast* específico, e estejam dentro do alcance da TTL do remetente. Assim, o esquema de TTL/*threshould* juntamente com o verdadeiro *multicasting* provê um método bastante eficiente para distribuir e ter acesso aos fluxos de mídia de *multicast* na *Internet.*

4.3 Aplicações multimídia do MBone

Existe uma diversidade de aplicações multimídia consolidadas desenvolvidas para funcionar sobre a rede MBone que possibilitam a divulgação dos eventos disponíveis. Uma lista completa dessas aplicações existentes pode ser encontrada em *http://www.merit.net/net-research/mbone/index/titles.html.* Fazem parte desta lista as ferramentas de anúncio de sessão, de transmissão de áudio e de transmissão de vídeo.

4.3.1 Ferramentas de anúncio de sessão

Esse tipo de ferramenta, quando invocado, mostra todas os eventos do MBone que estão disponíveis. Os usuários utilizam essa ferramenta para participar dos eventos como membros integrantes de grupos específicos. A ferramenta indica através do nome os eventos disponíveis, e fornece informações sobre eles, apresentando uma descrição, o período de transmissão, os tipos de mídias utilizados e os endereços e portas reservados para cada evento.

As ferramentas de anúncio de sessão são utilizadas para criar e anunciar uma sessão *multicast*, reservando um endereço que fica disponível para o evento enquanto ele estiver ativo, sendo liberado somente depois que o evento é encerrado.

Existem diversas ferramentas de anúncio de sessão: como o *Session Directory* (SDR) e a *Multimedia Conference Control* (MMCC).

A ferramenta SDR, visualizada na figura 4.3, mostra quais as sessões que no momento estão ativas. Desta forma, o usuário pode escolher de qual sessão ele quer fazer parte.

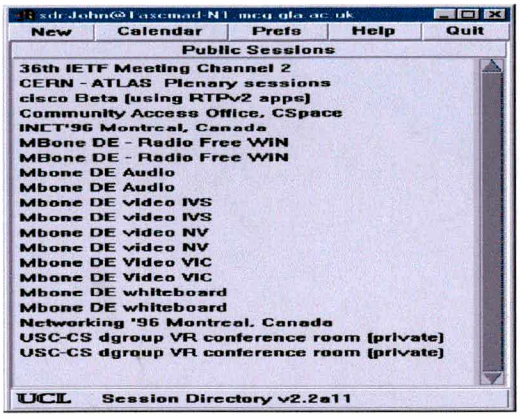

Figura 4.3 Janela principal do SDR

A figura 4.4, apresenta informações sobre as mídias que estão sendo utilizadas pela sessão. Cada mídia utiliza um endereço diferente, permitindo que o usuário possa escolhê-la. Para o escolher qualquer uma das mídias, basta selecionar os botões áudio, vídeo ou *Start All* que automaticamente aciona todas as mídias disponíveis.

O SDR também apresenta um calendário que mostra os eventos que já foram anunciados ou estão agendados para cada dia.

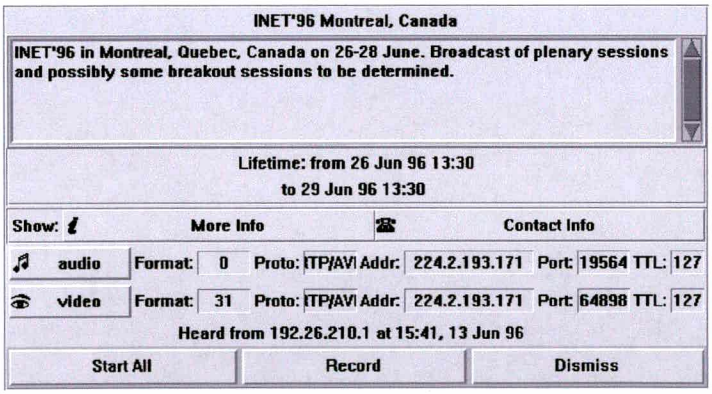

Figura 4.4 Janela de informação de uma sessão da SDR

4.3.2 Ferramentas de Áudio

Para utilizar as ferramentas de áudio do MBone é necessário apenas de um alto-falante nas estações receptoras e um microfone na estação transmissora.

O VAT (*Visual Audio Tool*) é a ferramenta de audioconferência mais usada no MBone, permitindo que os usuários realizem audioconferências sobre a Internet de um computador para outro (ponto a ponto), usando endereços *unicast,* ou entre vários computadores (*multicast*). O VAT utiliza o protocolo RTP para encapsulamento dos pacotes de áudio.

A figura 4.5 mostra a janela principal do VAT, de onde é possível identificar os membros participantes e o membro transmissor. Ela também controla a ativação do microfone e auto-falante, permitindo regular o volume para ambos. O VAT oferece informações como nome, endereço eletrônico e última transmissão enviada dos membros participantes, obtidas através de um clique sobre o membro. Informações de controle são trocadas por pacotes de controle RTCP do protocolo RTP.

VAT fornece estatísticas, permitindo ao usuário monitorar as estatísticas da rede tal como detecção de atraso e perda.

Figura 4.5 Janela principal da Visual Audio Tool

4.3.3 Ferramentas de Vídeo

As primeiras ferramentas de videoconferência da Internet foram a ferramenta NV (*Network Video*) desenvolvida pela Xerox PARC e a IVS (INRIA *video conferencing system*). Baseada na experiência obtida pelo NV e IVS, foi criada a ferramenta VIC (*VideoConference*). O INRIA desenvolveu o sucessor do IVS, chamado de *Rendez-Vous.*

Muitas dessas ferramentas são baseadas no protocolo RTP, que permite encapsular diferentes esquemas de codificação de áudio e vídeo e transmitindo-os sobre

o UDP e IP *multicast*. Como o protocolo é o mesmo, as ferramentas possuem interoperabilidade.

IVS

A figura 4.6 apresenta a ferramenta IVS (*INRIA Video Conferencing System*), uma aplicação adaptativa que usa a o padrão de codificação H.261, permitindo um bom funcionamento em enlaces com baixa largura de banda. As codificações de áudio e vídeo são realizadas via software.

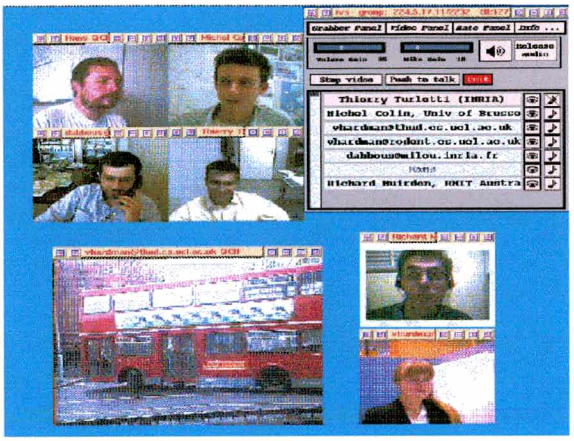

Figura 4.6 Janela principal do IVS

IVS implementa um algoritmo de controle de congestionamento que adapta sua taxa de saída de acordo com as condições de rede. O mecanismo de controle de congestionamento é baseado em dois componentes:

- Um **sensor de rede**, que mede a perda de pacotes através dos relatórios RTCP gerados pelos receptores após cada 100 pacotes.
- Um **controlador de vazão**, que ajusta a taxa de saída máxima do codificador fazendo com que a taxa de perda média para todos os receptores mantenha-se abaixo do valor tolerável. A taxa de saída do codificador pode ser ajustada alterando a taxa de quadros, os fatores de quantificação, ou o domínio de pesquisa do vetor de movimento.

IVS permite a seleção de dois modos diferentes: privilégio a qualidade (PQ) e privilégio a taxa de quadros (PFR). PQ é apropriado para aplicações exigindo alta precisão. PFR é apropriado quando a percepção do movimento é um importante fator de qualidade.

Rendez-Vous

O sucessor do IVS, chamado de *Rendez-Vous*, está atualmente sendo desenvolvido no INRIA. Ele já suporta JPEG e H.261, e o H.263 também está sendo implementado. Arquivos de vídeo MPEG-1 e MPEG-2 podem ser lidos pela ferramenta e traduzidos ou coodificados para um dos formatos mencionados. Procedimentos de escalonamento otimizados aumentam o uso dos recursos do sistema, resultando em um melhor comportamento em tempo de execução e melhor qualidade fornecida ao usuário.

VIC

O VIC (*Videoconference*) é hoje a ferramenta de videoconferência mais usada pelo MBone. Esta ferramenta foi desenvolvida com uma arquitetura flexível e extensível para suportar ambientes e configurações heterogêneas. Ela é baseada no protocolo RTP e está disponível para a maioria das plataformas Unix e Windows 95.

A figura 4.7 apresenta a janela principal do vic, onde são fornecidas informações sobre o evento, como a origem, a taxa de quadros que está sendo transmitida e a velocidade. Ela apresenta uma versão reduzida do vídeo, caso o usuário clique sobre este vídeo, será aberto uma janela com dimensões maiores.

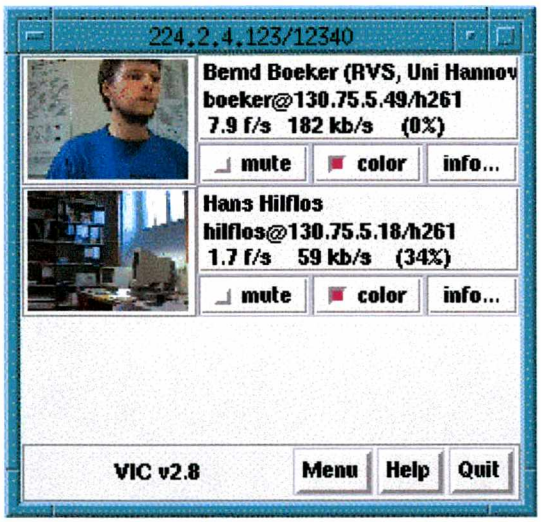

Figura 4.7 Janela principal da VIC

A ferramenta oferece diferentes esquemas de compressão, utilizando como configuração padrão (sem alterações) o esquema de compressão H.261 e o RTP como o mecanismo de transporte da aplicação. Oferece ainda controle da velocidade de transmissão e possibilidade de criptografia de acordo com a especificação RTP. Estas

alterações são feitas na janela Menu, que é acessada com um clique no botão Menu da janela principal.

VIC pode executar sobre diferentes camadas de redes tal como UDP/IP ou AAL5/ATM.

Enquanto que as ferramentas IVS e NV suportam a compressão/descompressão apenas em *software*, a VIC foi projetado suportar compressão/descompressão por software e *hardware.*

NV

A ferramenta de videoconferência *Network Video* (NV) permite a usuários transmitir e receber vídeo, via protocolo UDP, fazendo uso de uma baixa taxa de quadros.

NV utiliza um algoritmo de compressão de vídeo, desenvolvido especialmente para atingir baixa vazão de dados e alta vazão dos quadros, e o protocolo RTP versão 1 no nível de transporte da aplicação. Fluxos de vídeo podem ser enviados ponto a ponto, ou enviados para vários destinos simultaneamente, usando IP *multicast*.

O vídeo transmitido é uma imagem 320x240, onde a taxa de quadros varia de acordo com o movimento e a largura de banda disponível.

Assim como na VIC, a janela principal do NV visualizada na figura 4.8, apresenta uma imagem reduzida do vídeo e informações sobre a sessão, tais como nome do originador e TTL inicial dos datagramas. Com um clique sobre a imagem, é possível visualizar maior janela da aplicação.

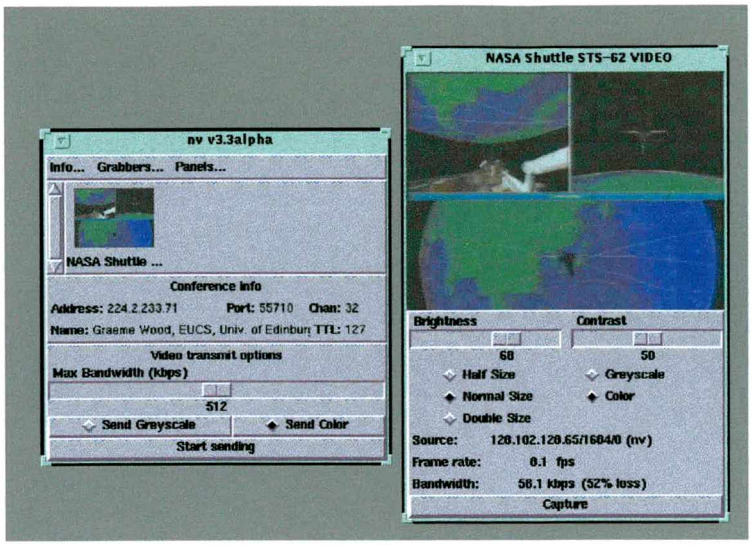

Figura 4.8 Janela Principal do NV

4.4 Conclusão

Este capítulo apresentou a rede MBone, um backbone fornecendo funcionalidades *multicast* para as aplicações sobre a Internet. Esta estrutura vem sendo atualmente muito utilizada para difusão de eventos, tipo seminários, reuniões, difusão de rádios e outros. Mas a tecnologia de rede em uso hoje na MBone permite apenas a transmissão de vídeo de baixa qualidade e baixa taxa de quadros. Além de não garantir a largura de banda necessária à transmissão de áudio e vídeo, ela não garante o atraso nem a variação de atraso.

No próximo capítulo será apresentada a tecnologia ATM, uma tecnologia de rede de alta velocidade satisfazendo quase todos os requisitos a nível de rede para o transporte de informações multimídia.

5. Redes ATM

ATM (*Asynchronous Transfer Mode*) é uma tecnologia de rede complexa, desenvolvida pelas companhias telefônicas e companhias de comunicação de dados entre computadores.

Durante muitas décadas as redes telefônicas públicas foram baseadas em comutação analógica. Com o surgimento da tecnologia de transmissão digital surgiu as redes digitais integradas (IDNs - *Integrated Digital Networks*), que refletiam a integração de comutação e transmissão usando técnicas digitais. A idéia do IDN foi proposta em meados de 1959, com o primeiro sistema de mensagem.

A evolução das redes telefônicas públicas de analógica para digital foi direcionada pela necessidade de fornecer um meio de comunicação econômico para voz. O IDN também foi útil em conhecimento, necessário para o surgimento de uma variedade de serviços digitais de dados. Desse modo, o que no passado era IDN, combinado com uma rede telefônica geograficamente extensa e com uma rede de dados digitais, deu origem a uma estrutura chamada ISDN (*Integrated Services Digital Network*), redes digitais de serviços integrados. O ISDN refere-se simultaneamente o envio de voz digitalizada e de uma variedade de tráfego de dados incluindo numa mesma transmissão digital sobre um canal de transmissão digital. O objetivo do ISDN é oferecer serviços de dados sobre uma rede digital telefônica sem nenhum custo ou penalidade para os serviços de transmissão de voz que já existiam no IDN.

Em 1988, como parte de sua primeira série de recomendações sobre ISDN, o antigo CCITT publicou as primeiras recomendações relatando as redes digitais de serviços integrados de banda larga, B-ISDN (*Broadband Integrated Services Digital Network)*. ATM foi escolhido como modo de transferência na implementação do B-ISDN. Portanto, o B-ISDN, iniciou a evolução da comutação de circuitos das redes telefônicas, envolvendo dentro da rede de comutação de pacotes seus serviços de banda larga.

Redes ATM são baseadas em conexões virtuais sobre um meio contendo alta largura de banda, sendo capaz de transferir voz, vídeo e dados através de redes privadas

cruzando redes públicas. O ATM é atraente e interessante pois além de oferecer um uso eficiente da largura de banda, fornece garantias de qualidade de serviço (QoS). Esses são os principais requisitos para se ter comunicação de alta qualidade.

Hoje o ATM continua evoluindo em vários grupos a fim de especificar padrões que procuram permitir a interoperabilidade entre equipamentos produzidos por vendedores nas redes públicas e redes industriais privadas. A definição dos padrões para essa tecnologia está centralizada em dois órgãos de padronização:

- **ITU-T**, ramo de padronização de telecomunicações da *International Telecommunications Union* (ITU);
- **ATM Forum**, um consórcio formado por várias empresas e usuários cujo maior objetivo é desenvolver a tecnologia e padronizar as redes ATM privadas.

5.1 Arquitetura ATM

O ATM envolve a transferência de dados em pedaços discretos, permitindo que múltiplas conexões lógicas sejam multiplexadas sobre uma simples interface física. No ATM, o fluxo de informação de cada conexão lógica é organizado dentro de pacotes de tamanho fixo, denominados células.

O ATM permite operar com altas taxas de dados porque reduz o número de *bits* suplementares necessários em cada célula e a quantidade de tempo gasto pelo computador no processamento dessas células. Sendo assim, utilizar um tamanho fixo de células simplifica o processamento necessário em cada nodo ATM.

A arquitetura do protocolo padronizada pelo ITU-T é mostrada na figura 5.1, está dividida em: camada física, camada de transporte, camada de adaptação e camadas superiores. A camada física envolve a especificação do meio de transmissão e de um esquema de codificação de sinais. A taxa de dados especificada na camada física foi originalmente 155,52Mbps e 622,08Mbps. Posteriormente, ela pôde variar para taxas de dados mais altas ou baixas

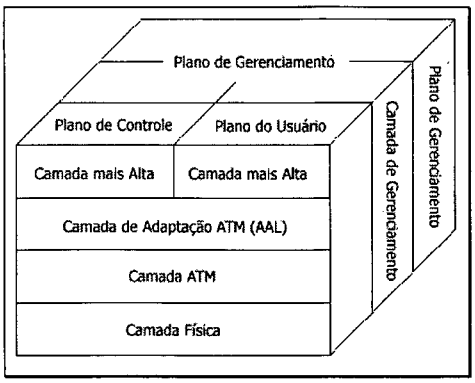

Figura 5.1 Modelo de Referência para Protocolos - PRM

Duas camadas da arquitetura do protocolo relatam as funções ATM. Existe uma camada ATM que é comum em todos os serviços que fornecem capacidade de transferência de pacotes, e uma camada de adaptação, também conhecida como AAL. A camada ATM define a transmissão de dados em células de tamanho fixo e também define o uso de conexões lógicas. A camada AAL é responsável pelo suporte à transferência de informação em protocolos não baseados no ATM, mapeando informações da camada superior dentro de células ATM que serão transportadas sob uma rede ATM, e coletando informação proveniente dessas células para enviar à camada superior.

O modelo de referência do protocolo mostra uma arquitetura baseada na interface entre usuário e rede que é formado de três planos separados:

- Plano de usuário: é utilizado na transferência de informações de usuários, junto com associações de controles (controle de fluxo e controle de erro);
- Plano de controle: realiza chamadas de controle e conexão das funções de controle;
- Plano de gerenciamento: Inclui gerenciamento de plano, que realiza funções de gerenciamento relacionadas com o sistema como um todo e fornece coordenação entre todos os planos; e gerenciamento de camadas, que realiza o relato das funções de gerenciamento de recursos e parâmetros associados a entidades de protocolos.

5.2 Conexões Lógicas ATM

Numa rede ATM células são transportadas através de conexões. Essas conexões não são conexões físicas ponto-a-ponto, e sim conexões lógicas que são configuradas antes da transferência de um dado, e finalizadas após o término da transferência.

Existe dois tipos de conexões lógicas ATM: circuitos virtuais permanentes e circuitos virtuais comutados. "Os circuitos virtuais permanentes são solicitados pelo cliente manualmente e em geral permanecem conectados durante meses ou anos. Os circuitos virtuais comutados se comportam como chamadas telefônicas: eles são estabelecidos de forma dinâmica, conforme a necessidade, e logo depois são desconectados." [Tanenbaum, 96].

Conexões lógicas ou circuitos virtuais em redes ATM são oficialmente chamadas de canais virtuais (*Virtual Channel Connection* - VCCs). Um VCC é a unidade básica de comutação das redes ATM. Um VCC é a concatenação de conexões virtuais entre dois usuários finais desde a origem até o destino. VCCs são também usados em transferências usuário-rede (controle de sinal) e transferências rede-rede (gerenciamento de rede e roteamento).

Os pontos finais de um VCC podem ser usuários finais, entidades de rede, ou um usuário final e uma entidade de rede. Em todos esses casos, a integridade da seqüência das células é preservada dentro do VCC; isto é, as células são entregues na mesma ordem que elas foram enviadas.

A associação de entradas de cada VCC, na tabela de rotas, implica em um alto volume de processamento, tanto no momento da conexão como no encaminhamento das células. Vários VCCs que possuem pontos finais idênticos são agrupados em uma conexão de caminho virtual *(Virtual Path Connection* – VPC), reduzindo o processamento gasto com a configuração das conexões.

Com a utilização de VPCs ocorre um aumento do desempenho da rede. A maior parte do trabalho é feito durante a configuração de um caminho virtual (VPC). Se o caminho virtual for mantido para conexões posteriores, um novo canal virtual poderá ser estabelecido executando uma simples função de controle entre os dois pontos finais

do VPC. Portanto, a adição de novos canais virtuais em um caminho virtual existente envolve o mínimo de processamento.

Os mecanismos de controle de caminhos virtuais incluem cálculo de rotas, capacidade de alocação e armazenamento de informações do estado das conexões.

5.2.1 Gerenciamento de tráfego

Conexões ATM possuem propriedades, tais como descritores de tráfego, parâmetros de QoS e categoria de serviço, que permitem especificar o tipo de tráfego na conexão e o modo como ele será tratado pela rede. Parâmetros de QoS, descritores de tráfego e outros parâmetros como endereço destino são passados para a rede como parte da configuração requisitada. Os parâmetros são levados em elementos de informação nas mensagens de sinalização. Esses elementos de informação são:

- A categoria de serviço: esta é uma propriedade do VCC que está sendo configurado. As categorias de serviços definidas são: CBR (*Constant Bit Rate*), rt-VBR (*real-time Variable Bit Rate*), nrt-VBR (*no-real-time Variable Bit Rate*), ABR (*Available Bit Rate*) e UBR (*Unspecified Bit Rate*). As categorias de serviço serão vistas em detalhes mais adiante.
- Parâmetros de QoS: a categoria de serviço define quais parâmetros de QoS serão especificados na conexão:
  - CLR (*Cell Loss Ratio*): taxa de células perdidas (número de células perdidas/ número de células transmitidas);
  - CTD (*Cell Transfer Delay*): tempo de retardo de transferência da célula
  - CDV (*Cell Delay Variation*): variação de retardo da célula.
  - Para conexões CBR, todos os três parâmetros anteriores são executados, mas para ABR apenas CLR é necessário.
- Descritor de tráfego: os descritores de tráfego estão baseados nas seguintes características de fluxo de células:
  - PCR (*Peak Cell Rate*): taxa de células pico(PCR);
  - SCR (*Sustainable Cell Rate*): taxa de células sustentável ;
  - MBS (*Maximum Burst Size*): tamanho máximo de explosão.

Após a conexão ter sido estabelecida com sucesso para uma certa categoria de serviço, o cliente e a rede ATM possuem um contrato de tráfego associado com a conexão. Se os usuários da conexão tentarem usar mais recursos da rede, isto é, se eles excederem os parâmetros do contrato de tráfego, a rede pode descartar células menos prioritárias ou usar uma solução dentro da capacidade da rede que possibilite manter o contrato. Os parâmetros de tráfego podem ser renegociados quando a conexão ainda estiver em uso.

A rede pode usar um número de estratégias para lidar com o congestionamento e gerenciar as conexões VCC existentes e as negociadas. Dentro de um esquema de prioridade, a rede pode simplesmente negar novos pedidos de conexões de VCCs previndo congestionamento. Células podem ser descartadas se os parâmetros de negociação foram violados ou se o congestionamento tornar-se sério. Em uma situação extrema conexões podem ser finalizadas.

5.2.2 Controle de Sinalização

Todo o processo de sinalização é da competência do plano de controle, que utiliza a camada ATM para o transporte de células com informações de sinalização [Soares 95].

O ATM precisa de um mecanismo para o estabelecimento e rompimento de VPCs e VCCs. A troca de informação envolvida neste processo é conhecida como controle de sinalização, e acontece em conexões separadas dessas que estão sendo gerenciadas. Essa conexão especial é denominada SVCC (*Signalling Virtual Channel Connection*). A alocação de SVCCs é feita numa fase de metassinalização, que pode ser ativada quando um equipamento é ligado.[Soares 95]

5.3 Células ATM

As células transitam em redes ATM passando através de dispositivos conhecidos como comutadores ATM. Os comutadores ATM são interconectados ponto a ponto por enlaces ATM ou interfaces. Eles analisam o cabeçalho da informação e enviam a célula ao destino através de uma interface de saída conectada ao próximo comutador. Comutadores ATM suportam dois tipos de interfaces: interfaces de usuários de rede

(*user-network interfaces* -UNI) e interfaces de nodos de rede (*network-node interfaces* - NNI), também conhecidas como interface rede a rede. A UNI conecta o ponto final do sistema ATM (*hosts*, roteadores) em um comutador ATM, enquanto que a NNI é mais precisamente definida para conectar dois comutadores ATM [Souza 97].

O ATM usa a tecnologia VLSI (*Very Large-Scale Integration*) para segmentar dados em alta velocidade para dentro de unidades chamadas células [Kyas 95]. Cada célula consiste de 5 octetos para o cabeçalho de informação e 48 octetos para dados, como mostra a figura 5.2.

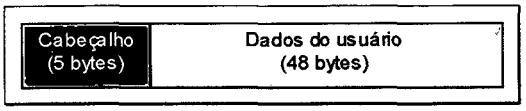

Figura 5.2 Formato da célula ATM

Existe várias vantagens em usar um tamanho pequeno e fixo de células. Primeiro, o uso de células pequenas pode reduzir o tempo de enfileiramento quando uma célula de prioridade mais alta chega um pouco atrás de uma célula de prioridade mais baixa que ganhou o acesso ao recurso. Segundo, células com um tamanho fixo podem ser comutadas mais eficientemente, o que é importante para as altas taxas de dados do ATM. Além disso, com células de tamanho fixo, é fácil de implementar um mecanismo de comutação por *hardware*.

5.3.1 Formato do cabeçalho .

A figura 5.3 mostra o formato do cabeçalho em uma interface UNI, e a figura 5.4 mostra o formato do cabeçalho de uma interface NNI. A interface UNI define o limite entre um *host* e uma rede ATM. A interface NNI diz respeito a linha entre dois comutadores ATM. A diferença entre as duas interfaces é que na interface UNI não existe o campo GFC que executa funções locais de controle de fluxo de células. Ao invés disso, o campo VPI é expandido de 8 para 12 bits. Isto permite que as células NNI possam suportar um grande espaço VPI, com maior número de VPCs internos à rede.

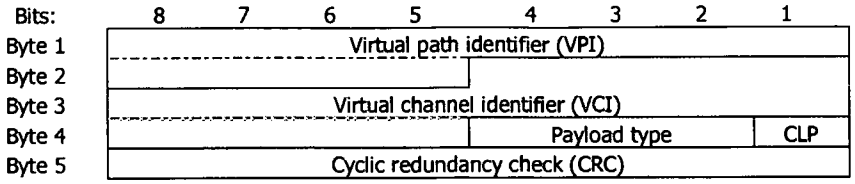

Figura 5.3 Cabeçalho da célula no NNI

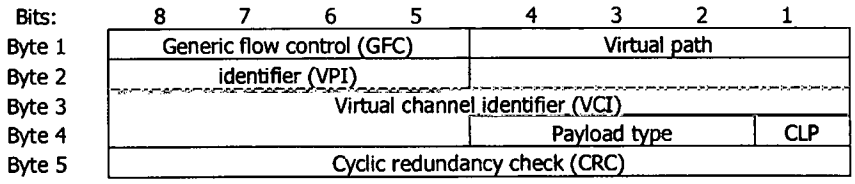

Figura 5.4 Cabeçalho da célula no UNI

O cabeçalho da células UNI e NNI tem em comum os seguintes campos:

- **VPI**: o campo VPI constitui um campo de roteamento para a rede, permitindo que mais canais virtuais sejam suportados dentro da rede;
- **VCI**: o campo é usado no roteamento a partir de um usuário final, funcionando como um ponto de acesso a serviço;
- **PTI**: indica o tipo de célula. O valor 0 no primeiro bit indica informação de usuário. O segundo bit indica se houve ou não congestionamento; um valor igual a 0 indica que não houve congestionamento no caminho e um valor igual a 1 significa que a célula passou por algum nó congestionado. O terceiro bit, conhecido como unidade de dados de serviço (*Service Data Unit* - SDU), é um campo de 1 bit que pode ser usado para discriminar dois tipos de SDUs ATM associados com a conexão e a interpretação depende do contexto. O valor 1 no primeiro bit do campo PT indica que esta célula leva informação de operação e manutenção da rede. Esta indicação permite a inserção de células de manutenção dentro de um VCC usuário sem danificar os dados do usuário, permitindo fornecer informação de controle dentro da banda;
- **CLP**: o campo indica prioridade em caso de necessidade de descarte de células. O valor 0 indica que a célula é de alta prioridade, que não pode ser descartada a menos que não haja outra alternativa disponível. O valor 1 indica que a célula pode ser descartada dentro da rede. O usuário pode ocupar esse campo se a rede não estiver congestionada, inserindo células extra dentro da rede, com um CLP em 1, e remetendo-as para o destino. A rede pode fixar esse campo em 1 em

alguma célula de dado que não estiver dentro dos parâmetros de tráfego negociados entre o usuário e a rede. Neste caso, é o comutador que faz a colocação sabendo que a célula excedeu o acordo dos parâmetros de tráfego. Se mais tarde houver congestionamento em algum ponto da rede, esta célula será marcada para descarte em preferência a células que ficaram dentro do limite do acordo de tráfego;

- **HEC**: o preenchimento correto do campo HEC é responsabilidade da camada física. As células ATM possuem 8 bits no campo de controle de erro do cabeçalho que correspondem a um código de redundância. Esse código permite que a camada física verifique se houve ou não erro no cabeçalho, podendo fazer a correção de 1 bit.

Depois do cabeçalho, há 48 *bytes* de carga útil (*payload*). No entanto, nem todos os 48 bytes estão disponíveis para o usuário, pois alguns protocolos AAL colocam seus cabeçalhos dentro da carga útil.

5.4 Categorias de Serviço ATM

A rede ATM foi projetada para possibilitar a transferência de vários tipos de tráfego simultaneamente, incluindo fluxo em tempo real como voz, vídeo, e fluxo em rajada. Apesar de cada fluxo de tráfego ser dirigido como um fluxo de células de 53 octetos que trafegam por um canal virtual, o modo no qual cada fluxo de dados é dirigido dentro da rede depende das características do fluxo e das exigências da aplicação. Por exemplo, tráfego de vídeo em tempo real deve ser transmitido dentro de uma variação mínima de tempo de retardo.

As seguintes categorias de serviço foram definidas pelo ATM Forum:

5.4.1 Serviço em tempo real

Nas aplicações típicas em tempo real um usuário espera que o fluxo de áudio e vídeo possa ser apresentado de maneira contínua e ajustada. A falta de continuidade resulta numa significante perda de qualidade. São serviços em tempo real: taxa de bits constante (CBR- *Constant Bit Rate*) e taxa de bits variável em tempo real (rt-VBR-*real-time Variable Bit Rate*).

Serviço CBR

CBR é usado por aplicações que exigem uma taxa de dados fixa que está continuamente disponível durante o tempo de vida da conexão, onde sua taxa média é igual a taxa de pico. CBR é comumente utilizado para áudio não comprimido e algumas técnicas de codificação de vídeo em tempo real.

Serviço rt-VBR

A categoria rt-VBR é definida para aplicações sensíveis ao tempo. A principal diferença entre aplicações adequadas para rt-VBR e aplicações adequadas para CBR é que aplicações rt-VBR transmitem com uma taxa que varia com o tempo. Uma origem rt-VBR pode ser caracterizada com alguma quantidade de tráfego em rajada, por exemplo, o padrão apropriado para compressão de vídeo resulta na seqüência de quadros de imagens de vários tamanhos.

Nesse serviço o retardo médio e a variação de retardo devem ser submetidos a um rígido controle.

O serviço rt-VBR permite mais flexibilidade que o CBR. A rede é habilitada para multiplexar um número estatístico de conexões sobre o mesmo meio dedicado e ainda fornecer o serviço requerido para cada conexão. Um exemplo de aplicação rt-VBR é o vídeo codificado MPEG.

5.4.2 Serviço não tempo real

Serviços que não são em tempo real são usados em aplicações que tem características de tráfego em rajadas e não tem constrangimento com variação de retardo. Consequentemente, a rede pode fazer melhor uso da multiplexação estatística aumentando a eficiência na rede. São exemplos: serviço nrt-VBR (*no real time – Variable Bit Rate*), serviço ABR (*Available Bit Rate*), serviço UBR (UBR - *Unspecified bit rate*)

Serviço nrt-VBR

Para algumas aplicações que não são em tempo real, é possível caracterizar o fluxo de tráfego esperado de maneira que a rede possa melhorar substancialmente a QoS nas áreas de perda e retardo. Com este serviço, o sistema final especifica uma taxa de

células pico, uma taxa de células média e uma medida de como pode ser as células em rajadas ou acumuladas. Com essas informações, a rede pode alocar recursos para fornecer um tempo de retardo relativamente baixo e o mínimo de células perdidas. O serviço nrt-VBR pode ser usado para transferência de dados que tem exigência crítica de tempo de resposta. Por exemplo: reserva de linhas aéreas, transações bancárias, e processo de monitoramento.

Serviço UBR

Esse serviço não é em tempo real. Num dado instante, uma certa quantia de capacidade de uma rede ATM é consumida por CBR e os dois tipos de tráfego VBR. Uma capacidade adicional é disponibilizada por ambas ou uma das seguintes razões: (1) Nem todos os recursos foram comprometidos para o tráfego CBR e VBR, e (2) a natureza em rajada do tráfego VBR precisa da capacidade comprometida que está sendo usada somente durante algum tempo. Toda esta capacidade em desuso pode ser disponibilizada para o serviço UBR Este serviço é adequado para aplicações que podem tolerar variação de tempo de retardo e perda de algumas células. Exemplos de aplicações UBR incluem: transferência de texto, dados e imagem, correio eletrônico e terminal remoto.

Serviço ABR

O serviço ABR foi definido para melhorar o serviço fornecido pela fonte do tráfego em rajadas. Se a aplicação que utiliza ABR alocar sua capacidade para trabalhar com uma taxa de pico, as linhas podem ficar ociosas aguardando um aumento no tráfego. Se a capacidade alocada for para trabalhar com uma taxa mínima, pode haver congestionamento. A capacidade não usada é compartilhada, de modo controlado, entre todas as fontes ABR. Seja qual for o valor da capacidade alocado, ele será garantido o tempo inteiro. Se houver necessidade de um valor maior existe a possibilidade, mas não a certeza, de que ele possa ser alcançado. Capacidades não utilizadas pelas fontes ABR permanecem disponíveis para o tráfego UBR.

5.5 Camada de Adaptação ATM

Enquanto no nível ATM o tráfego pode ser transportado em todas as categorias de serviço, camadas superiores precisam de serviços mais específicos, mais adequados ao tipo de tráfego produzido por estas camadas. Nem todos os usuários B-ISDN precisam de uma relação cronometrada entre entidades que enviam e recebem. Neste sentido, foi definida a camada de adaptação ATM.

A AAL é utilizada para prover segmentação e remontagem das células ATM provenientes das camadas superiores. A AAL também pode verificar células perdidas ou dados errados e executar multiplexação de canais de dados da camada superior dentro de um simples VCC.

Quando novos tipos de tráfego são reconhecidos, uma nova AAL pode ser definida para eles, enquanto que a camada ATM permanece inalterada. Para dar apoio às diferentes classes de serviço ATM, o ITU-T propôs três camadas de adaptação diferentes:

- **AAL1**: é usada para a transmissão do tráfego em tempo real, orientado a conexão e com taxa de bits constante, tal como o tráfego gerado por sinais de áudio e vídeo não compactados. Na AAL1 não são utilizados protocolos de detecção de erros porque não são aceitáveis retardos causados por *timeouts* e retransmissões. Nessa camada a aplicação é responsável por tomar providências para recuperar perdas de células, se houver;
- **AAL2**: é usada para transmissão do tráfego em tempo real, orientado a conexão e com taxa de bits variável, tal como os sinais de áudio e vídeo compactados. Esse protocolo permite detecção de erros e de células perdidas;
- **AAL3/4**: esse protocolo não tem restrições de tempo, é sensível a perdas e erros, com taxa de bits variável e pode ser ou não orientado a conexão. É o único protocolo que possui recurso de multiplexação, permitindo que várias sessões (ou seja, *logins* remotos) de um único *host* trafeguem pelo mesmo circuito virtual e sejam separadas no destino.

O ATM Forum propôs uma nova camada AAL5 e está em discussão a criação da AAL6:

- **AAL5**: é mais utilizada no transporte de tráfego produzido pela camada IP. A AAL5 oferece serviço confiável (entrega com garantia) e não confiável (entrega sem garantia). Tanto *unicast* quanto *multicast* são aceitos, mas *multicast* não oferece garantia de entrega. Ela fornece um transporte orientado a conexão com tamanho variável de quadros e com detecção de erro. Os quadros de dados de usuário podem ser superiores a 65.535 bytes, que são segmentados dentro de quadros de 48bytes adicionando o controle de informação e de da preenchimento AAL5. Esses quadros são enviados como *payload* (dados de aplicação) nas células ATM. Tanto *Classical* IP como LAN *Emulation* usam a AAL5 para transportar sua estrutura de protocolo sobre a rede ATM;
- **AAL6**: adequada para fluxo de dados multimídia compactado, em particular para vídeo codificado no formato MPEG e MPEG-II. A discussão da AAL6 inclui o uso de técnicas de correção de erros de reenvio (FEC - *Forward Correction Error*) aumentando a confiança em um nível onde não existirá a necessidade de reconhecer nenhum erro extra.

5.6 Integrando LANs às Redes ATM

Enquanto a rede pública telefônica não for substituída por uma rede ATM, existe a necessidade de contornar problemas de conectividade entre a tecnologia ATM e as redes locais existentes (padrão IEEE 802). Nesta abordagem, uma rede ATM pode funcionar como uma LAN, conectando *hosts* individuais, ou como pontes, conectando várias LANs. Existe duas propostas voltadas ao objetivo de contornar esses problemas de conectividade: LAN *Emulation*, desenvolvida pelo ATM Forum, e *Classical* IP, modelo da comunidade IETF. Ambas utilizam o campo de carga útil da AAL5 para transportar pacotes IP.

5.6.1 LAN Emulation

LAN *Emulation* permite que sistemas finais (como estações de trabalho, servidores, *gateways*) se conectem a uma rede ATM de forma que as suas aplicações interajam como se estivessem ligadas a LANs convencionais, interligadas através de pontes [Souza 97]. Este esquema pode ser visualizado na figura 5.5 a seguir. Esta

técnica possibilita a interoperabilidade entre aplicações residentes nas redes locais existentes ligadas a uma rede ATM.

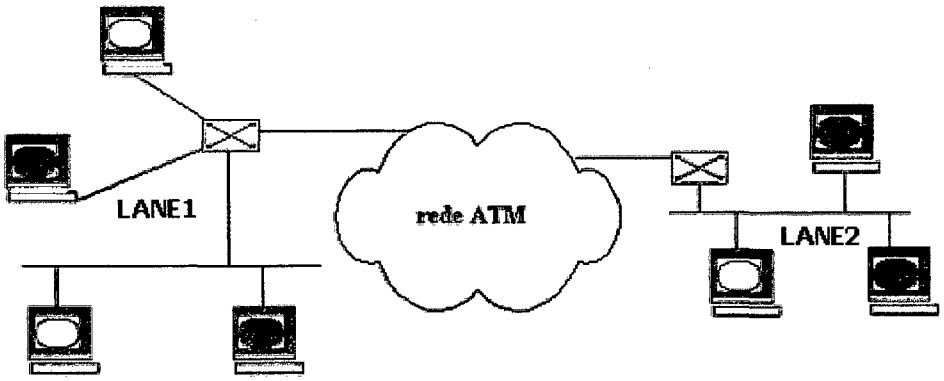

Figura 5.5 Emulação de LAN sobre ATM

Uma LAN emulada (LANE) permite a transmissão de quadros de dados entre todos os seus usuários como se eles estivessem fisicamente ligados a uma LAN, contudo, usuários de uma LANE não podem se comunicar com usuários de outra LANE. Em uma rede ATM podem ser definidas mais de uma LANE, como no exemplo acima LANE1 e LANE2, mas usuários da LANE1 não podem se comunicar diretamente com usuários da LANE2. Para solucionar esse problema é definido o que se chama de Rede Local Virtual (VLAN). O administrador de uma rede pode configurar várias VLANs sobre uma única rede ATM, permitindo a adaptação da sua rede sem que haja a necessidade de uma mudança física. Essas VLANs são comunicáveis apenas através de pontes e podem ser reconfiguradas de forma fácil e dinâmica.

Os componentes que implementam uma LANE são: LAN *Emulation Client* (LEC), LAN *Emulation Configuration Server* (LECS), *LAN Emulation Server* (LES) e o *Broadcast and Unknown Server* (BUS).

LAN Emulation Client

Os LECs estão tipicamente localizados em estações finais ATM e solicitam serviços do LECS, LES e BUS diretamente na LANE *User-Network Interface* (LUNI).

LAN Emulation Configuration Server

A função do LECS é fornecer os dados de configuração inicial para os LECs. A parte mais importante dos dados de configuração é o endereço ATM do LES. Quando

um LECS não está presente em uma ELAN, o LES de um LEC deve ser definido junto ao LEC pelo administrador do sistema.

LAN Emulation Server

O LES é o servidor de resolução de endereços numa rede local emulada. Apenas um LES pode estar presente em uma LANE e os membros da LANE são definidos pelo LES em uso. Quando os LECs se juntam a LANE, eles registram seus endereços MAC (o endereço emulado de controle da acesso ao meio) no LES, que pode ser utilizado mais tarde para resoluções de endereços em transmissões diretas. O LES estabelece uma conexão ATM ponto a multiponto que parte do LES em direção a todos LECs da LANE, a fim de distribuir informações de controle.

Broadcast and Unknown Server

Como ATM não tem mapeamento direto de endereços Ethernet *broadcast* e *multicast*, o BUS maneja todo o tráfego que os LECs enviam para esses endereços, estabelecendo uma conexão ponto a multiponto com todos os clientes da LANE. A segunda função do BUS é a de manejar o tráfego para clientes que tem que enviar para um endereço MAC *unicast* que até agora não foi resolvido, o pacote é enviado ao BUS para que ele seja difundido, e em seguida (paralelamente) o cliente procura o endereço desconhecido usando o LES.

A arquitetura LANE pode ser usada como um fornecedor completo de serviço LAN sobre ATM, incluindo *multicasting* (também referenciado como mIP/ATM). O BUS utiliza um VCC ponto a multiponto para enviar tráfego *multicast* para os LECs. Dentro da LANE, isso funciona como *broadcast* não permitindo o uso eficiente da rede, não tão ideal como se houvesse um VCC ponto a multiponto partindo diretamente do remetente para todos os membros de um grupo.

5.6.2 Classical IP e ARP sobre ATM

Os componentes e operações do *Classical* IP e ARP sobre ATM (CLIP) são especificados em um grupo de RFCs do IETF. Na RFC-1577 é dada uma especificação sobre como implementar um *Classical* IP (apenas *unicast*) e o serviço ARP sobre o topo da camada de adaptação AAL5. A RFC-1483 define como as unidades de dados do

protocolo (PDUs) são transportadas sobre a AAL5. A RFC-1755 mostra a sinalização ATM envolvida em uma operação IP sobre ATM. O valor padrão da unidade de transmissão máxima (MTU- *Maximum Transmit Unit*) de 9180 bytes é definido na RFC-1626.

O objetivo da RFC-1577 é usar ATM como uma possível subrede IP. Para isso ela deve resolver os seguintes problemas [Kno 98]:

- Mapeamento de endereços IP e ATM;
- Implementação de um mecanismo de resolução de endereços (ARP) sobre ATM;
- Estabelecimento e término de conexões diretas ATM (VCs) entre dois sistemas finais IP;
- Multiplexação de protocolos diferentes sobre a mesma conexão ATM (VC).

Os componentes que implementam o CLIP são: *Logical IP Subnet* (LIS), servidores ARP e clientes ARP.

Logical IP Subnet

No cenário de uma LIS, cada entidade administrativa configura separadamente seus *hosts* e roteadores como se fosse uma rede lógica IP separada. Cada LIS opera e se comunica independentemente com outras LISs sobre a mesma rede ATM. *Hosts* pertencentes a mesma LIS comunicam-se diretamente, mas *hosts* pertencentes a LISs diferentes comunicam-se através de um roteador IP.

Uma LIS possui todas as propriedades de uma subrede IP normal. Entretanto, como o ATM é uma rede de acesso múltiplo sem *broadcast* (NBMA- *Non Broadcast Multiple Access*), não é possível resolução de endereços utilizando *broadcast*, por isso, servidores e clientes ARP foram desenvolvidos para resolver esse problema.

Servidor ARP

O servidor ATMARP aceita chamadas/conexões de outro ponto final ATM. Existe apenas um servidor ARP por LIS. Ele solicita ao cliente (através de um InATMARP *request*) informações de mapeamento. O cliente responde (através de um ATMARP *reply*) o seu endereço IP e o seu endereço ATM. O servidor, então, adiciona

esses endereços em sua tabela ATMARP. A informação da tabela é temporizada, se o tempo expirar, a entrada na tabela é removida e a informação atualizada.

Cliente ARP

Quando um cliente é inicializado, ele tem a responsabilidade de contatar o servidor ATMARP para registrar sua própria informação ATMARP. Quando o cliente tem um tráfego para transmitir para outro cliente ele envia para o servidor um ARP *request* contendo o endereço IP destino. O servidor responde o endereço ATM do destino, ou um NAK se a informação não está disponível. O cliente utiliza o endereço ATM para criar um VC com o destino, por onde trafegarão os datagramas IP.

5.7 Conclusão

Este capítulo apresentou ATM, uma tecnologia de rede que satisfaz quase todos os requisitos e é potencialmente satisfatória para comunicações multimídia. A lista abaixo discute como ATM satisfaz alguns dos requisitos de comunicação multimídia:

- **Largura de Banda**: ATM fornece um acesso de alta velocidade para cada sistema final. Duas velocidades de acesso de 155 e 622 Mbps tem sido padronizadas, embora velocidades mais altas e mais baixas são possíveis. Estas velocidades de acesso são suficiente para qualquer tipo de aplicação multimídia com a tecnologia de compressão atual.
- **Flexibilidade e garantias de QoS**: redes ATM podem suportar aplicações com diferentes características de tráfego e requisitos de comunicação, que inclui largura de banda, restrição de atraso e sensibilidade a erro. ATM permite a alocação dinâmica de largura de banda para cada aplicação, enquanto o serviço orientado a conexão fornecido pela ATM pode reservar recursos para garantir a QoS de cada aplicação.
- **Integração**: o protocolo ATM pode simultaneamente suportar múltiplas aplicações com diferentes características e requisitos pela mesma UNI. Isto é crucial para suportar aplicações multimídia, pois múltiplos circuitos virtuais podem ser requeridos concorrentemente.

- **Arquitetura**: a topologia física em mecha ponto-a-ponto empregada pela B-ISDN fornece uma largura de banda dedicada muito maior para cada usuário comparado às redes usando meio compartilhado tal como FDDI e DQDB. Além disso, isto permite que a mesma arquitetura pode ser empregada tanto como solução de rede pública e privada, as duas executando essencialmente o mesmo protocolo.
- **Eficiência**: Desde que ATM emprega a multiplexação estatística para compartilhamento da largura de banda de transmissão, o ganho estatístico é significante quando aplicações com alto pico para a taxa de largura de banda média (i.e., tráfego em rajada) são multiplexadas. Desde que várias aplicações multimídia e colaborativas tendem a ser muito em rajadas, o ganho estatístico resultante implica em um compartilhamento muito eficiente do custo de transmissão.
- **Multicasting**: ATM permite a implementação de multicasting pela alteração de VCI e VPI em comutadores.
- **Versatibilidade**: ATM usa um mecanismo uniforme de multiplexação e comutação independente da taxa de bits, tamanho da rede e meio de transmissão. ATM é um padrão para serviços de rede locais, campus/backbone, pública e privada. Esta uniformidade simplifica o gerenciamento da rede usando a mesma tecnologia para todos os níveis de rede.

Aplicada a Internet/MBone, as redes ATM são interconectadas a outras tecnologias de rede via o protocolo IP (usando CLIP ou LANE). Neste caso, tem-se que parte da infra-estrutura de rede é capaz de suportar bem o tráfego multimídia, mas em partes tem-se o serviço de comunicação melhor-esforço. Suponha um cenário onde a aplicação adaptativa fonte da informação de áudio e vídeo esteja em uma rede ATM e os destinos destas informações estejam na mesma rede ATM ou em outras sub-redes de tecnologias diversas. Neste caso, a aplicação adaptativa terá que adaptar seu fluxo baseada nos relatórios das redes de mais baixa velocidade. Assim, os clientes na rede ATM terão seus fluxos de áudio e vídeo baixados de qualidade, mesmo que a rede entre o fonte e os destino seja capaz de transportar as mídias adequadamente. O próximo capítulo apresenta uma proposta de utilização de filtros de escalamento de mídia na

integração de redes ATM com redes de menor desempenho para solucionar este problema.

6. Filtros Adaptativos no Transporte de Vídeo sobre Redes Heterogêneas

A rede ATM é ideal para a comunicação multimídia, com alta velocidade e largura de banda disponível. Portanto, no ambiente ATM os problemas relacionados à existência de largura de banda suficiente para a transmissão de vídeo são mínimos ou inexistentes. Ao contrário, as redes padrão Ethernet 10Mbits/s, onde o meio físico é compartilhado entre todos, não satisfazem a maioria dos requisitos para a transmissão de áudio e vídeo. Por exemplo, estatísticas obtidas junto ao NPD (Núcleo de Processamento de Dados) da UFSC constataram durante alguns períodos do dia a ocupação da largura banda chega ao limite máximo, ocorrendo algumas colisões e perdas de pacotes.

A rede Internet, e portanto também a MBone, é composta de várias redes heterogêneas: redes Ethernet 10Mbps, 100Mbps, X.25, Frame Relay, PPP, redes ATM e até mesmo redes interligadas por rádio. Sendo o protocolo IP (*Internet Protocol*) o responsável por esta interconexão. Isto possibilita a interoperabilidade entre aplicações Internet independente da tecnologia da rede local onde se situam os clientes. No caso da tecnologia ATM, LAN *Emulation* ou *Classical* IP se encarregam de encapsular os pacotes de dados IP em células ATM. Então, não existe o problema de portabilidade.

Neste ambiente de redes heterogêneas, principalmente devido à diversidade de largura de banda disponibilizadas pelas tecnologias de rede, os participantes de aplicações multimídia de grupo, por exemplo videoconferência e distribuição de vídeo, receberão e poderão transmitir vídeos com qualidades diferentes dos demais participantes do grupo. Por exemplo, supondo que a fonte de uma videoconferência com alta qualidade de vídeo esteja em uma rede ATM e os destinos destas informações estejam na mesma rede ATM ou em outras sub-redes de tecnologias diversas (por exemplo, Ethernet a 10Mbps). Os destinos do vídeo situados na rede ATM normalmente receberão o fluxo de vídeo sem problemas. Mas os destinos situados em redes de baixa velocidade terão problemas na sua apresentação, baixando a qualidade de apresentação. Neste último caso, os roteadores entre as tecnologia de alta velocidade e baixa poderão

ter seus *buffers* sobrecarregados, ocasionando o descarte de informações. Neste caso, não se terá meramente uma baixa na qualidade em termos de resolução de imagem ou taxa de quadros. Isto pois os dados serão descartardos aleatoriamente.

Um problema em potencial no uso de aplicações adaptativas é a adaptação da fonte de vídeo baseado nos relatórios das redes de mais baixa velocidade. Assim, os clientes na rede ATM terão seus fluxos de áudio e vídeo baixados de qualidade, mesmo que a rede entre o fonte e os destino seja capaz de transportar as mídias adequadamente. Isso não é interessante para os demais espectadores da rede ATM, que possuem disponibilidade de recursos mas são "obrigados" a assistir um vídeo com qualidade inferior porque a aplicação se adaptou a uma realidade da qual eles não fazem parte. Este trabalho trata unicamente do problema do transporte de vídeo em redes heterogêneas, já que em aplicações de videoconferências, as fontes de áudio geram taxas iguais ou inferiores a 64Kbps, taxa normalmente aceitáveis mesmo em redes locais de baixa velocidade.

Esse capítulo propõe uma solução para o transporte de vídeo em redes heterogêneas, tratando-se da interconexão entre ATM e Ethernet 10Mbps. Mais especificamente, este trabalho propõe o uso de filtros adaptativos em *gateways* na interconexão de sub-redes heterogêneas na MBone, destacando o problema relacionado a execução de aplicações adaptativas nesse ambiente e apresentando uma solução para o problema.

6.1 Integração da Tecnologia ATM na MBone

A rede MBone possui protocolos e aplicações desenvolvidos e consolidados, além de ser uma infra-estrutura muito usada para divulgação de eventos. Com a expansão das sub-redes ATM e considerando que esta tecnologia satisfaz a maioria dos requisitos para o transporte de informações multimídia, é claro que esta tecnologia será usada em sub-redes na MBone.

Dois aspectos devem ser avaliados para a integração da tecnologia ATM na MBone: inclusão da camada IP na arquitetura ATM e a difusão seletiva. Como visto na seção 5.6, existem atualmente duas opções para a inclusão do protocolo IP na arquitetura ATM: LANE e CLIP.

Difusão Seletiva (*Multicast*)

O MBone realiza *multicast* utilizando o protocolo de comunicação em grupo IGMP e um protocolo de roteamento, que normalmente é o DVMRP. Juntos, esses protocolos permitem enviar pacotes somente para os membros do grupo que estão interessados em recebê-los. Isso é denominado de *multicast* "verdadeiro", que pode ser facilmente implementado em redes com meio físico compartilhado tal como Ethernet, Token Ring e FDDI.

A facilidade de realização de *multicast* "verdadeiro" sobre a rede MBone não se estende a rede ATM. LAN *Emulation* e *Classical* IP apresentam maneiras diferentes de implementar o *multicast*, mas elas possuem limitações. Estas limitações ocorrem devido à existência de apenas dois tipos de conexão no ATM: ponto a ponto e ponto a multiponto. Não existe comunicação multiponto a multiponto em ATM, que é o tipo de conexão ideal para a implementação do *multicast* verdadeiro.

LANE (LAN *Emulation*)

No LANE, quem realiza *multicast* é o BUS, um servidor especial que faz *broadcast* de todos os pacotes *multicast* para todos os nodos da sub-rede, através de uma conexão ponto a multiponto de cada vez. O maior inconveniente desse modelo é que todos os pacotes *multicast* são enviados para todos os nodos da sub-rede, portanto nodos recebem pacotes *multicast* de grupos que eles não fazem parte. Em conseqüência disso, todos os nodos receptores têm que filtrar os pacotes *multicast*, recebendo somente aqueles que lhe interessam e descartando os demais.

Classical IP

O protocolo *Classical* IP define um mecanismo de encapsulamento e um mecanismo de resolução de endereços. O mecanismo de encapsulamento pode ser aplicado a dos muitos protocolos existentes em redes locais, mas o mecanismo de resolução de endereços foi definido apenas para o protocolo IP. Diferente do LANE, no *Classical* IP uma PDU IP é encapsulada apenas uma vez, usando o padrão IEEE 802.2 LLC/SNAP e em seguida ela é enviada para a camada AAL5.

Sobre o protocolo *Classical* IP existem duas formas de realizar *multicast*: malhas-VC; ou servidores *multicast* ATM. O maior problema em utilizar malhas-VC é o uso extensivo de conexões VC. Para uma rede com "n" remetentes e "m" grupos

multicast é necessário estabelecer e atualizar n*m VCs ponto a multiponto. Além disso, a cada entrada/saída de um grupo são necessários modificar as conexões de n VCs ponto a multiponto. Isso dá um certo *overhead* de sinalização nos comutadores ATM. O modelo servidor *multicast*, quando comparado ao modelo malhas-VC, possui a desvantagem de concentrar todo tráfego *multicast* em um só ponto da rede, o servidor. Concluindo, utilizar malhas VC é melhor quando os grupos são pequenos e estáveis, enquanto que, utilizar servidores *multicast* é melhor quando os grupos são grandes e dinâmicos (com mudanças em grande escala).

6.2 *Gateway* ATM para MBone

A arquitetura mais utilizada atualmente para integrar sub redes ATM com a MBone é a arquitetura LANE. A figura 6.1, demonstra uma estação que possui duas interfaces, uma para a rede ATM e outra para a rede Ethernet que funciona como *gateway* das duas subredes, roteando pacotes *multicast* entre elas, através da execução do programa *mrouted.*

O funcionamento no nível 3 e 4 dessas redes permanece o mesmo, porém existe uma distinção a ser considerada. No protocolo Ethernet, a distribuição dos pacotes *multicast*, dentro da sub-rede, é feita através de difusão. Já no nível de enlace implementado pelo protocolo LAN *Emulation*, quem envia os demais datagramas dentro da sub rede ATM é o BUS. Portanto, no LANE os pacotes *multicast* saem da interface ATM, e são direcionados para o BUS que se encarrega de estabelecer uma comunicação ponto a ponto de cada vez, enviando os pacotes *multicast* para todos os membros da LANE independente ou não deles fazerem parte de grupos *multicast.*

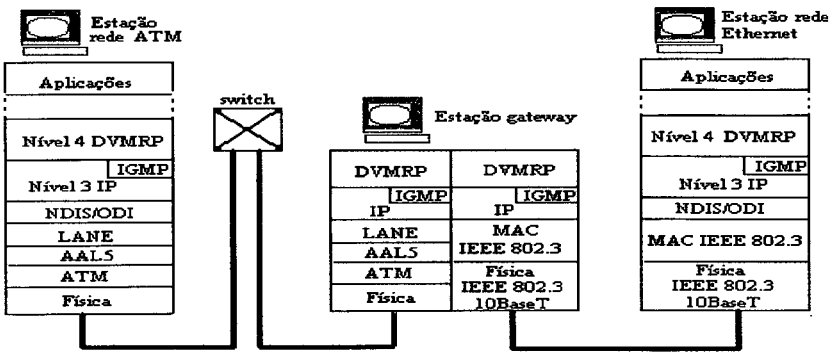

Figura 6.1 *Gateway* ATM para MBone

6.3 Fluxo de dados no Roteador

Um outro fator que deve ser considerado na integração entre redes ATM e outras tecnologias é a quantidade de fluxo de vídeo que pode ser enviada por uma fonte na sub-rede ATM e que deve ser difundida em outras redes de menor largura de banda.

O ATM pode gerar fluxo de vídeo da ordem de Mbits/s, por exemplo, se uma aplicação estiver transmitindo fluxo de vídeo padronizado do tipo MPEG a uma taxa de 8Mbits/s. Quando esse fluxo passar para a rede MBone, vai gerar sobrecarga no enlace da rede. Quando ocorre sobrecarga, o roteador da rede MBone descarta pacotes aleatoriamente. Esse descarte não programado de pacotes resulta em problemas na recepção de imagens do vídeo. A imagem observada no destino será incompleta e muitas vezes impossível de ser visualizada, pois terá varias falhas que correspondem aos quadros que não chegaram. A figura 6.2 mostra uma imagem que sofreu perda de pacotes.

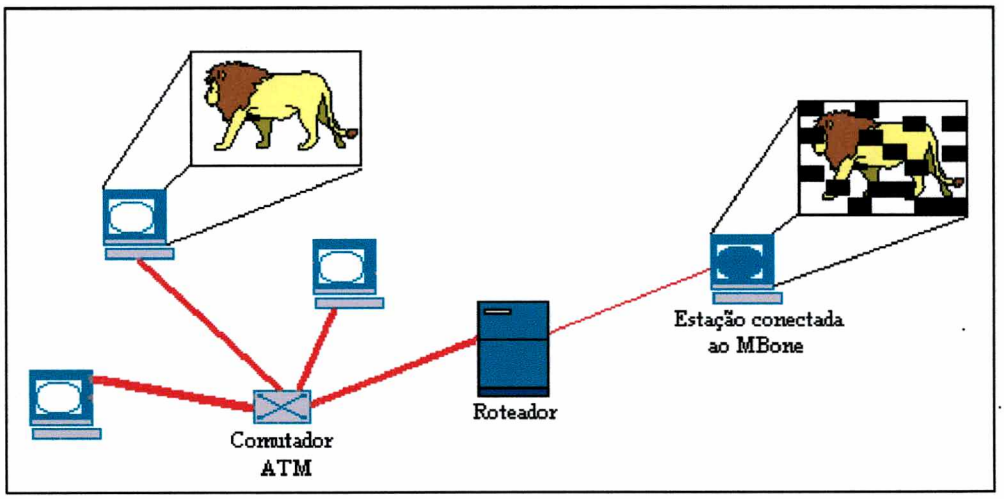

Figura 6.2 Sobrecarga nos roteadores Mbone

Outro problema ocorre quando a aplicação de videoconferência utilizada é adaptativa. Este tipo de aplicação adaptativa é útil justamente neste ambientes composto por redes heterogêneas, onde, independente da localização da fonte, os destinos possuem diferenciação nos recursos disponíveis.

No caso de aplicações adaptativas baseadas no protocolo RTP, e dependendo das políticas de escalamento de mídia, as fontes de vídeo receberão relatórios de QoS (via protocolo RTCP) dos destinos situados na rede ATM e na MBone como um todo. É

claro que os relatórios dos destinos fora da rede ATM comandarão o escalamento da mídia, fazendo que os receptores na rede ATM tenham sua qualidade de recepção deteriorada.

6.4 Arquitetura de Gateway Proposta

A solução encontrada para o problema das aplicações adaptativas, descrito anteriormente, é a utilização de um filtro de mídia adaptativo, chamado FiltroGW, no nó que interliga a rede ATM a outra tecnologia de menor largura de banda, ou seja, no *gateway*. Esta solução é ilustrada na figura 6.3. Esta proposta limita-se às aplicações adaptativas baseadas no protocolo RTP.

O FiltroGW tem basicamente duas funções:

- Escalamento do fluxo de vídeo enviado pela rede ATM às limitações existentes na rede de menor largura de banda, evitando sobrecarga nos roteadores. Os relatórios de QoS gerados pelos receptores, via RTCP, serão usados como base para o escalamento de mídia.
- Filtragem dos pacotes RTCP oriundos da sub-rede de menor largura de banda, responsáveis por baixar a qualidade da aplicação cuja fonte localiza-se na rede ATM. Neste caso, a fonte não escalonará o vídeo para se adaptar aos recursos disponíveis pelos receptores fora da rede ATM. A fonte receberá apenas os relatórios dos receptores dentro da rede ATM. A figura 6.3 apresenta a arquitetura proposta para o *gateway*. Note que a diferença está na presença do FiltroGW.

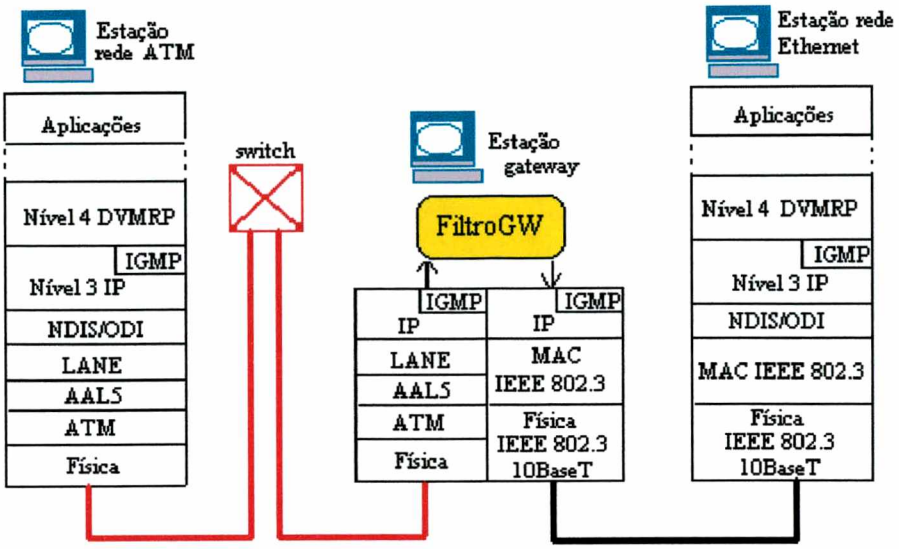

Figura 6.3 Arquitetura de *Gateway* proposta

Observa-se na figura 6.4 o exemplo de umas das funcionalidade do FiltroGW na interconexão ATMxEthernet, que é reduzir o fluxo de vídeo adaptando-o aos recursos disponíveis da rede Ethernet. Todas as estações receptoras localizadas na rede ATM recebem o mesmo fluxo enviado pela origem, e somente as estações localizadas na rede Ethernet sofrem as conseqüências da redução do fluxo de vídeo (menor fidelidade da imagem resultante do aumento na compressão dos dados).

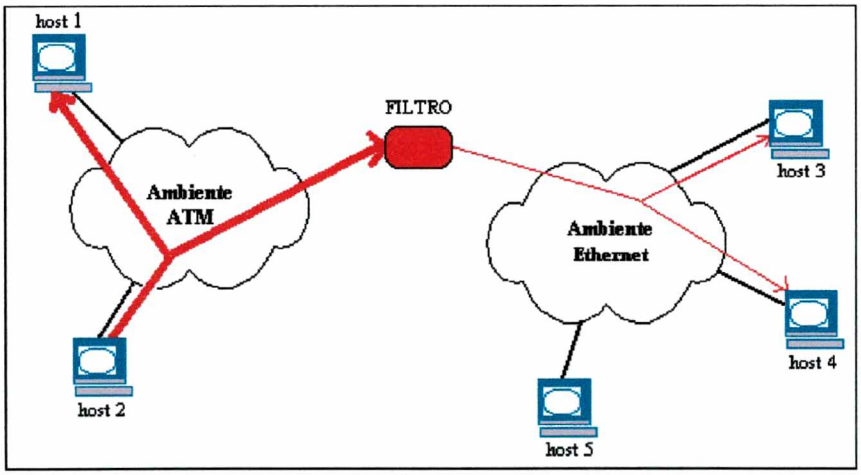

Figura 6.4 Redução do fluxo de vídeo

6.5 Filtragem Tempo-real

A implementação de filtros nos nós da rede não é simples. Primeiro existem várias formas de representação de mídias, assim muitos tipos de filtros são necessários. Existem sugestões onde os filtros devem ser fornecidos pelos emissores ou receptores durante a fase de reserva de recurso, mas o mecanismo exato não é claro. Segundo, para aplicações em tempo real, filtros devem executar suas funções em tempo real. Isto pode ser difícil, pois algumas funções de filtragem necessitam de um tratamento computacional muito grande e os computadores geralmente já funcionam sobrecarregados.

Filtragem de mídia pode trabalhar apenas se o fluxo de dados multimídia puder ser decomposto em sub-fluxos com diferentes parâmetros (resolução, velocidade, etc). Existem filtros de mídia que trabalham com vários padrões de codificação (H261, H263, MPEG), funcionando como tradutores de um padrão para outro, por exemplo de MPEG-1 para H261, ou tradutores que utilizam o mesmo padrão (de H261 para H261), porém reduzindo o número de quadros ou aumentando a compressão dos dados.

Na filtragem de mídia, são adicionados nós internos mais inteligentes à rede. Se eles entendem o conteúdo de um fluxo de dados, eles podem filtrar pacotes. A inconveniência é que o escalamento de mídia toma tempo de processamento, aumentando o atraso e a variação de atraso.

6.6 Modelagem do FiltroGW

Como já apresentado, o FiltroGW possui duas funções: escalar o vídeo de acordo com os relatórios de QoS do RTCP, e impedir que estes relatórios alcancem a fonte caso haja destinos na rede ATM. Para realização destas funções, o FiltroGW deve realizar uma série de atividades, que são divididas em quatro módulos (figura 6.5): captura, seleção, devolução, filtro e descarte.

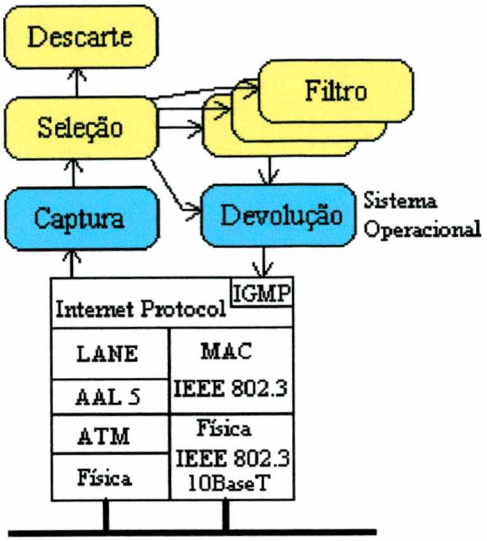

Figura 6.5 Módulos do FiltroGW

No FiltroGW, em cada fluxo de vídeo que será adaptado existe uma instância filtro responsável por fazê-lo. O módulo de seleção é responsável por acionar essa instância sempre que chega fluxo oriundo de uma nova fonte.

6.6.1 Módulo de Captura

O módulo de captura, como o nome indica, captura todos os pacotes UDP. Estes pacotes devem ser armazenados e analisados pelo módulo de seleção antes de serem descartados ou enviados para o roteamento.

A figura 6.6 mostra o formato do pacote capturado pelo módulo de captura.

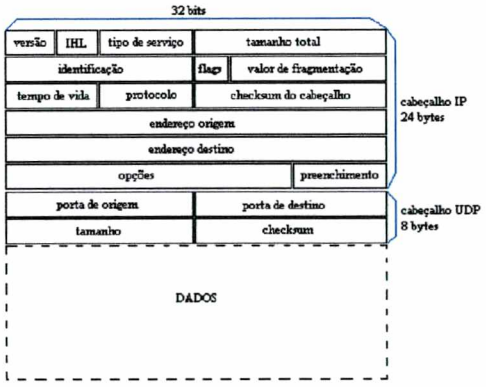

Figura 6.6 Formato do pacote UDP

6.6.2 *Módulo de Seleção*

O módulo de seleção recebe os pacotes do módulo de captura, utiliza as informações sobre a estrutura desse pacote (que indicam qual é a interface de entrada e saída do pacote), para selecionar, dentre os pacotes UDP, aqueles que possuem as seguintes características:

- São pacotes RTP de transmissão de vídeo, oriundos da interface ATM e cujo destino é a interface Ethernet. Esses pacotes são enviados à instância do módulo Filtro responsável pelo escalamento de mídia da sessão identificada nos pacotes RTP.
- São pacotes RTCP do tipo BYE, oriundos da interface ATM e cujo destino é a interface Ethernet. Esses pacotes identificam o término de uma sessão.
- São pacotes RTCP do tipo RR oriundos da interface Ethernet e cujo destino é a interface ATM. Estes pacotes contém os relatórios de QoS que serão armazenadas em uma lista (visto mais adiante). Após isto, o pacote é enviado para o módulo de descarte.

Pacotes RTCP

Os pacotes do tipo RTCP/RR capturados pelo módulo de seleção são pacotes que trazem informações sobre o fluxo de vídeo recebido pelos receptores localizados fora da sub-rede ATM, cuja fonte emissora localiza-se na rede ATM. Esses pacotes contém relatórios sobre os recursos disponíveis da rede Ethernet, que, quando interpretados por uma aplicação adaptativa, permitem que a aplicação adapte o fluxo de vídeo, aumentando ou diminuindo a qualidade da mídia. Esses pacotes são interpretados e em seguida descartados. Isso garante que nenhum pacote de controle RTCP seja enviado para a rede ATM, não permitindo que aplicações adaptativas localizadas na rede ATM adaptem-se à realidade encontrada apenas na rede Ethernet.

Como podem existir várias fontes de vídeo em uma videoconferência, o módulo Seletor receberá pacotes RTCP oriundo de todos os receptores de todas as fontes de vídeo. A identificação de qual fonte o pacote RTCP relata a QoS é identificada pelo campo SSRC (visto na sessão 3.1.1).

O módulo Seletor, além de selecionar os pacotes RTCP, é responsável por calcular os parâmetros de adaptação para uma determinada fonte de vídeo, que serão utilizados pelo módulo Filtro a fim de escalar a mídia. Estes parâmetros são armazenados em uma lista, denominada Lista_RR, que contém os seguintes tipos de dados:

- **ID_Fonte**: identificador da fonte do RTP a que se referem as informações contidas nos blocos de dados do pacote RTCP/RR;
- **Parâmetros de adaptação**: baseado nas estatísticas contidas nos pacotes RTCP/RR, o módulo Seletor deve determinar a taxa de quadro e parâmetros de resolução da imagem do vídeo.

Cada fonte de vídeo que passam pelo *gateway*, entrando pela interface ATM e saindo pela interface Ethernet, tem uma entrada na Lista_RR. Quando a sessão terminar, ela é retirada da lista. Isso pode ser identificado através do recebimento de um pacote RTCP do tipo BYE, que indica que a fonte não está mais ativa.

Temporariamente, a Lista_RR recebe parâmetros dos receptores que definem qual é o valor de escalamento de mídia ideal de cada receptor. O módulo filtro utiliza, para escalamento de mídia de uma determinada fonte, a média dos parâmetros de todos seus receptores. Para agilizar o processo de busca desse valor, ele é calculado e mantido sempre atualizado na primeira posição da Lista_RR.

Pacotes RTP

Os pacotes RTP recebidos devem ser encaminhados para a instância do módulo Filtro que escalam a mídia da fonte identificada no pacote RTP.

Sempre que uma nova fonte é identificada, é criada uma entrada na Lista_RR, onde ID_Fonte contém a fonte desse pacote RTP, e o primeiro elemento da lista de parâmetros corresponde a um valor padrão (configurável). Isto é feito para garantir o funcionamento do filtro, enquanto não chegam os pacotes RTCP/RR. Com o tempo este valor padrão é substituído pelo valor que corresponde a média dos parâmetros que irão compor a Lista_RR.

Além da entrada na Lista_RR, a cada detecção de uma nova fonte é criado um *buffer,* que receberá todos dados desta fonte, e disparada uma instância de execução do módulo filtro.

A funcionalidade do módulo de seleção pode ser representada pelo algoritmo descrito na figura 6.7

```
Recebe Pacote UDP do módulo de captura
limite = valor inicial para média dos parâmetros(default);

Se pacote for RTP de transmissão de vídeo e (interface de entrada = ATM) e (interface de saída = Ethernet) então
        Se não existe na Lista_RR uma entrada onde (Lista_RR_ID_Fonte = SSRC) então
                Cria entrada(Lista_RR_ID_Fonte=SSRC);
                Coloca primeiro elemento da Lista_RR_parâmetro[0]=limite;
                Cria buffer[SSRC];
                Dispara instância do módulo Filtro(SSRC, buffer);
        Senão
                Armazena pacote no buffer[SSRC];
        Fim Se
Fim Se
Se pacote for RTCF do tipo BYE e (interface de entrada=ATM) e (interface de saída=Ethernet) então
        Manda sinal de espera à instância do módulo Filtro para terminar a execução;
        Retira elemento da Lista_RR cujo ID_Fonte = SSRC;
        Destrói buffer[SSRC];
Fim Se
Se pacote for RTCP do tipo RR e (interface de entrada=Ethernet) e (interface de saída=ATM) então
        Para i=1 até rc (onde rc é um campo do RTCP que contém o número de blocos do relatório)
                Leia bloco de relatório RTCP/RR;
                Armazena os valores em Lista_RR_parâmetro[i];
        Fim Para
        Calcule a média dos valores de Lista_RR_parâmetro;
        Lista_RR_parâmetro[0]=valor obtido no cálculo da média;
Fim Se
```

Figura 6.7 Funcionamento do módulo Seleção

6.6.3 Módulo Filtro

Os pacotes RTP contendo vídeo capturados pela interface ATM que serão enviados através da interface Ethernet devem ser tratados, a fim de adaptar o fluxo de vídeo à realidade da rede de menor largura de banda (por exemplo, Ethernet), evitando a ocorrência de descarte dos pacotes pelo roteador que interliga as duas redes.

O módulo filtro deve escalar os dados de vídeo recebidos pela interface ATM, de forma a adaptá-los de acordo com os receptores localizados na rede Ethernet. Estes dados foram identificados pelo módulo seleção, que os armazenou, de acordo com a fonte RTP de onde eles originaram, em *buffers* separados. O filtro deve retirar os dados destes *buffers* para fazer o escalamento de mídia.

O funcionamento do módulo filtro é basicamente o seguinte. No momento em que o módulo seleção identifica a chegada de fluxo de vídeo oriundo de uma nova fonte RTP, através da interface ATM, ele dispara uma instância do processo filtro e armazena os dados deste fluxo em um *buffer* específico para ela. Desta forma, podem existir vários fluxos de vídeo filtrados em paralelo. Como os dados do *buffer* são acessados concorrentemente, para implementar colocação (módulo seleção) e retirada desses dados (módulo filtro) pode-se utilizar o conceito de monitores (definido em sistemas operacionais), que realizam exclusão mútua sobre regiões críticas (no caso os *buffers*) compartilhadas entre dois processos (o modo seleção que enche o buffer e o filtro que esvazia).

Antes de esvaziar o *buffer*, o filtro consulta a Lista_RR, obtendo o valor que identifica a média dos parâmetros de adaptação para escalar os dados do *buffer* de acordo com este valor. Depois de escalados, o filtro passa os pacotes para o módulo de devolução.

O pacote só será descartado pelo filtro se, mesmo aplicando a compressão máxima permitida pelos padrões, e reduzindo ao máximo a fidelidade da imagem, a taxa necessária para a transmissão de vídeo ainda for superior à suportada pela rede. Neste caso, já que o filtro está manipulando os pacotes, ele pode descartá-los antes que cheguem ao roteador, e armazenar mensagens de logs que podem ser úteis para a futuras análises e identificação do problema.

6.6.4 Módulo de Devolução

O módulo de devolução tem a função de devolver para o sistema todos os pacotes que foram enviados a ele. Estes pacotes são enviados de volta para o roteamento.

Esses pacotes podem ser os datagramas UDP não selecionados pelo módulo de seleção, ou datagramas que foram modificados pelo módulo filtro.

6.6.5 Módulo de Descarte

O módulo descarte descarta alguns datagramas específicos.

O módulo de descarte têm a função de descartar os pacotes RTCP do tipo RR depois que eles forem interpretados pelo módulo seleção. Estes pacotes não podem ser devolvidos ao sistema, pois do contrário todo o processamento e a funcionalidade do FiltroGW seria desperdiçado.

6.7 Descrição do Protótipo

Existe várias formas de implementar o modelo descrito na sessão 6.4. Uma delas é implementar cada módulo separadamente, e outra é utilizar algumas ferramentas adaptando-as aos objetivos do modelo. A seguir será apresentado um protótipo para implementação deste modelo, que envolve o uso de duas ferramentas: um filtro IP e uma ferramenta de aplicação que faz tanto escalamento de mídia como tradução de padrões.

Esse protótipo utiliza filtro IP para implementar os módulos de captura e devolução dos pacotes.

O módulo de seleção é implementado como um processo no espaço do usuário, que recebe e envia pacotes para o *kernel* através do filtro IP. Este processo fica em execução como "*daemon*" na estação *gateway*. Ele aciona várias instâncias do filtro, uma para cada fonte de fluxo de dados RTP.

O módulo filtro foi baseado no código fonte do programa *rtpgw*, resultado do projeto Daedalus (*Wireless Netwoking and Mobile Computing*) realizado na Universidade da Berkeley na Califórnia [Hunter 98].

As próximas sessões explicam melhor os detalhes deste protótipo. A implementação de todos esses detalhes ainda não foi terminada, mas será apresentada uma descrição de como ela deve ser.

6.7.1 Módulo de Captura

O captura não é feita diretamente na interface física e sim dentro do *kernel* do sistema operacional. Quando um pacote IP é recebido pelo *kernel,* este o coloca em uma fila para ser roteado. Estes pacotes podem ser analisados antes de entrarem na fila de roteamento, ou logo após serem roteados. Um módulo do *kernel* do sistema operacional

provê funcionalidades para a análise dos pacotes e tomada de ações, incluindo seus repasses a um processo no espaço do usuário para análise mais aprofundada.

A captura de datagramas que passam pelo *kernel* pode ser feita através de regras de filtragem IP que os repassam para processos no espaço do usuário. As informações sobre por qual interface o pacote entrou, e por qual ele irá sair, devem ser armazenadas em memória e ficar disponíveis até que o pacote seja devolvido a rede ou descartado. Desta forma, cabe a tais processos decidir se os datagramas continuam incólumes sua trajetória pelo *kernel*, se serão descartados, ou terão seus conteúdos modificados antes de retornarem ao *kernel*.

A implementação pode ser mais ou menos refinada, mas quanto maior for o refinamento melhor será o desempenho. Por exemplo, para tornar a seleção mais eficiente, a implementação deve escolher dentre pacotes IP aqueles que possuem em seu campo de dados um pacote UDP cujas portas de origem e destino possuem um valor na faixa de 49152 a 65535 (essas portas são utilizadas para transmissão de vídeo).

Várias implementações de Unix oferecem recursos que possibilitam esta abordagem. Como exemplo, os *kernels Linux* possuem um dispositivo *chamado IP Firewall Packet Netlink Device*, que é um dispositivo por onde podem passar pacotes selecionados via regra de filtragem IP para aplicações no espaço do usuário. No FreeBSD também se desviam pacotes através de regras de filtragem IP, assim como no SunOS 5 o módulo de *kernel IP Filter* permite a ação de processos do usuário na seleção de pacotes.

A implementação do módulo captura e devolução do FiltroGW utilizará um filtro IP de domínio público, denominado IP *Filter*, que funciona no SunOS 5, e onde será configurada a seguinte regra de seleção de pacotes:

- *auth in from any to* 224/4 *proto udp* – que permite registrar a entrada do datagrama na fila de roteamento do *kernel*.
- *auth out from any to* 224/4 *proto udp* – que permite registrar a saída do datagrama da fila de roteamento do *kernel*.

Ambos os comandos descritos acima aplicam a máscara 4 (240.0.0.0). Eles permitem registrar a entrada e saída de todos os pacotes *multicast* cujos endereços

variem de 224.0.0.0 a 239.255.255.255, e cujo campo de dados seja composto por datagrama UDP.

Como exemplo de uma seleção mais refinada, que seleciona também os pacotes cujo valor da porta é maior que 49152 e menor que 65535, podemos ter:

- *auth in from any to* 224/4 *proto udp* 49152 > *port* < 65535
- *auth out from any* to 224/4 *proto udp* 49152 > *port* < 65535

Após os dados terem sido filtrados, são utilizadas chamadas ao sistema, que recebem como parâmetros um descritor de arquivo (*fd*), uma constante definida pelo IP *Filter* para definir a operação a ser realizada, e um ponteiro para uma estrutura por onde serão retornadas informações que permitirão ao processo filtro manusear os datagramas. A chamada ao sistema que permite a captura dos datagramas selecionados pelas regras é [IPF 99]:

- ioctl (fd, SIOCAUTHW, struct fr_info *)

O descritor de arquivo *fd* deve ser obtido abrindo-se um dispositivo (*/dev/ipf*) utilizado para comunicação com o módulo IP *Filter* no *kernel*, *SIOCAUTHW* é uma constante definida pelo IP *Filter* que indica a operação a ser efetuada, e *struct fr_info* é a estrutura passada para o processo filtro. O formato da estrutura repassada para o usuário pode ser visto na figura 6.8.

A operação contrária pode ser realizada de forma semelhante. Apenas muda a constante indicadora da operação requerida [IPF 99]:

- ioctl (fd, SIOCAUTHR, struct fr_info *)

```
typedef struct fr_info {
    void      *fin_ipf;          /* interface de pacote ativa */
    struct    fr_ip   fin_fi;    /* sumário do pacote IP      */
    u_short   fin_data[2];       /*  portas TCP/UDP, código/tipo ICMP  */
    u_char    fin_out;           /* entrada ou saída ?  1== saída, 0==entrada  */
    u_char    fin_rev;
    u_short   fin_hlen;          /* tamanho em bytes do cabeçalho IP */
    u_char    fin_tcpf;          /* flags TCP  */
    u_char    fin_icode;
    u_short   fin_rule;
    u_short   fin_group;         /* membro do grupo -> se -1 nenhum  */
    struct    frentry  *fin_fr;
    char      *fin_dp;           /* início dos dados passados pelo cabaçalho IP */
    u_short   fin_dlen;          /* tamanho da porção de dados do pacote  */
    u_short   fin_id;            /* identificador do campo do pacote IP */
    void      *fin_mp;
#if SOLARIS && defined(_KERNEL)
    void      *fin_qfm;
    void      *fin_qif;
#endif
} fr_info_t;
```

Figura 6.8 Estrutura utilizada pelo filtro IP

6.7.2 Módulo de Seleção

A figura 6.9 demonstra um exemplo de implementação utilizado para obter a estrutura, que permite retornar os datagramas selecionados pelas regras de filtragem.

```
main ()
{
      struct  frauth fra;
      fr_info_t     *fin = &fra.fra_info;
      fr_ip_t       *fi = &fin ->fin_fi;
      char  yn[16];
      int fd;

     fd = open (IPL_NAME, O_RDWR);
     while  (ioctl (fd, SIOCAUTHW, &fra) == 0
     {
         .....

        if  (ioctl (fd, SIOCAUTHR, &fra) != 0)
            perror ("SIOCAUTHR");
     }
 }
```

Figura 6.9 Captura da estrutura utilizada pelo filtro IP

O módulo de seleção recebe a estrutura *fr_info*, contendo o pacote de dados e algumas informações sobre ele, e implementa as funções do algoritmo na figura 6.7.

6.7.3 Módulo Filtro

O módulo filtro do FiltroGW utiliza o código fonte da ferramenta *rtpgw*. A ferramenta *rtpgw* é uma aplicação que funciona no *gateway* na tentativa de amenizar a disparidade existente entre dois sistemas finais e as redes conectadas a ele, alterando a largura de banda mantendo um grau aceitável de qualidade visual percebida. A essência desse mecanismo está na tradução e controle da taxa de vídeo. O vídeo pode ser traduzido e convertido de um formato codificado para outro. O método mais simples é decodificar de um formato e codificar em outro, mas esse é o método que requer mais recursos computacionais. O projeto adotou o uso de métodos de filtragem baseados em DCT, que são computacionalmente mais eficientes.

O versão 1.0 do *rtpgw* contém dois componentes: as máquinas *vgw* (vídeo) e *agw* (áudio); e a interface usuária *rtpgw_ui*. Estes componentes rodam em processos separados e se comunicam através de um canal de comunicação compartilhado. Este modelo permite a outras aplicações controlar o *rtpgw,* possibilitando coordenação entre ferramentas de um ambiente de conferência.

A ferramenta *rtpgw* traduz os seguintes padrões de codificação de vídeo: de JPEG para H261, de NV para H261 e de H261 para H261.

O FiltroGW é uma aplicação dinâmica, executada no *gateway* de uma rede heterogênea, que possui várias instâncias que são acionadas pelo pelo módulo de seleção, de acordo com o número de sessões de vídeo recebidas.

A ferramenta *rtpgw* também é executada em um *gateway* , mas sua grande diferença em relação ao modelo proposto é ser uma entidade estática, ou seja, para filtrar cada fluxo de vídeo necessita-se de intervenção humana. Então, para utilizá-la seria necessária sempre a presença de uma pessoa em frente à máquina, controlando todo o fluxo de vídeo que passa pelo *gateway*. Isso é muito ineficiente.

Concluindo, este protótipo de implementação pretende utilizar o código do *rtpgw* que faz as traduções de padrões e escalabilidade de vídeo, adaptando-o no modelo de filtro proposto.

6.7.4 Módulo de Devolução

Como explicado no módulo de captura, o *kernel* do sistema operacional recebe o pacote de uma interface e sabe exatamente para qual ele deve enviar. Todo pacote que chega ao módulo de devolução deve ser colocado de volta à fila de roteamento, exatamente no mesmo ponto em que ele foi retirado.

Neste protótipo, a devolução é feita pelo IP *Filter*. O módulo de seleção deve retornar a estrutura da figura 6.8 para o IP *Filter*, que se encarrega de reinserir o datagrama por ela descrito à fila de roteamento.

6.7.5 Módulo de Descarte

Quando se capturam dados do *kernel* não basta descartá-los na rotina para que ele foi enviado, sendo preciso passar esta informação ao IP *Filter*. Este módulo tem a finalidade de informar ao *IP Filter* para que ele descarte o pacote.

6.7.6 Avaliação do Protótipo

A proposta do modelo aproveitando o uso de algumas ferramentas agiliza o processo de implementação, mas torna os módulos implementados "amarrados" ao sistema operacional escolhido. Nesse protótipo, as regras de filtragem e a estrutura de dados utilizada no IP *Filter* não são idênticas as regras de outros filtros IP desenvolvidos para outros sistemas operacionais. Portanto, se o *gateway* não for mais uma estação SUN com SunOS 5, será necessário estudar e implementar novamente os módulos de captura e devolução, fazendo alterações de acordo com o funcionamento das ferramentas e do sistema operacional escolhido. Por outro lado, utilizar rotinas que interagem diretamente com o *kernel* do sistema operacional resulta em ganho no desempenho de execução. Todo ganho em desempenho é de grande valia quando a aplicação trata de dados multimídia.

6.8 Testes realizados

A seguir serão apresentados alguns testes realizados.

6.8.1 Descrição do Ambiente de Teste

A figura 6.11 ilustra o ambiente de testes utilizado. Neste caso, uma interconexão da rede RMAV-FLN (Rede Metropolitana de Alta Velocidade de Florianópolis), baseada na tecnologia ATM, com a rede MBone. Esta interconexão é feita a partir da rede INF (rede local ao departamento de Informática e de Estatística da UFSC), baseada principalmente na tecnologia Ethernet a 10Mbps.

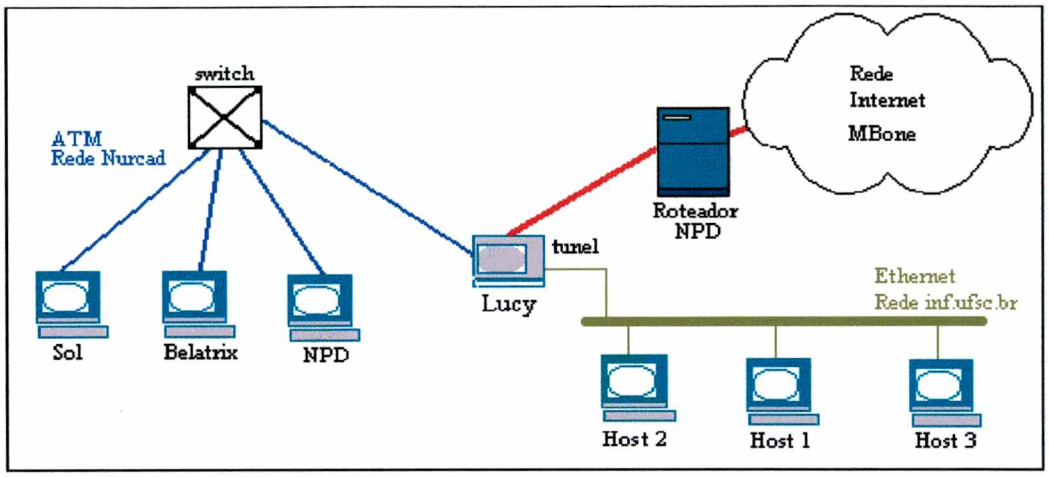

Figura 6.11 Ambiente de teste

A interconexão entre as redes Ethernet/ATM foi realizada através de LAN *Emulation*, que possibilitou a comunicação entre as duas arquiteturas. Note na figura 6.11 que a estação de trabalho Lucy pertence às duas sub-redes (RMAV-FLN e INF). Ela possui duas interfaces: uma ATM, ligada a RMAV-FLN e outra Ethernet ligada a rede INF. É esta estação que opera como *gateway*, transferindo dados entre a rede RMAV-FLN , que possui arquitetura ATM, e a rede INF, que possui arquitetura Ethernet. A estação de trabalho Lucy também está ligada ao MBone, através de um túnel *multicast* IP, que conecta a estação com um roteador no NPD da UFSC.

O programa responsável pelo roteamento dos dados *multicast* IP entre as duas redes é o *mrouted,* utilizado pela maioria dos roteadores do MBone. Esse programa também permite a configuração de um túnel o que faz com que a Lucy também esteja conectada ao MBone, podendo enviar e/ou receber seus eventos. Concluindo, a Lucy roteia dados em três redes: MBone, INF e RMAV-FLN. Conferências da rede RMAV-FLN são vistas pela rede INF e vice-versa. As redes INF e RMAV-FLN também recebem e enviam tráfego para o MBone.

6.8.2 Testes do Gateway ATM/Ethernet Tradicional

Para constatar a necessidade de um filtro adaptativo no *gateway* ATM/Ethernet, foram realizados alguns testes consistindo de realização de videoconferências utilizando ferramentas MBone, principalmente as ferramentas VIC e VAT.

O principal cenário de teste é o seguinte: realizar uma difusão para a rede MBone a partir de uma fonte de vídeo de alta qualidade situada na rede ATM e verificar a qualidade de recepção nos receptores situados na rede RMAV-FLN e na MBone como um todo.

Foi observado nas videoconferências realizadas sobre o ATM que as taxas máximas de bits geradas pelas aplicações de videoconferência utilizadas eram muito aquém daquelas esperadas. Portanto, não foi possível utilizar toda a capacidade de largura de banda da rede ATM. Estas limitações detectadas estão a nível poder de processamento a nível dos terminais ATM (as estações de trabalho). As máquinas utilizadas necessitavam de mais memória e poder de processamento afim de acelerarem os processos de codificação/decodificação realizados pelos aplicativos. Por exemplo, quando outras aplicações eram executadas na maquina fonte de vídeo, a taxa de quadros do vídeo nitidamente baixava. O mesmo foi observado nos pontos de recepção. Portanto, independente da rede utilizada: INF ou RMAF-FLN, a qualidade de vídeo recebida estava diretamente relacionada ao hardware receptor e não a falta de largura de banda, que sempre era suficiente.

Devido aos problemas descritos acima, não foi possível gerar um vídeo de alta qualidade. Neste caso, a necessidade do filtro não deveria ser sentida no caso dos destinatários situados na rede INF. Isto pois esta rede geralmente opera a menos de 5% de sua capacidade em termos de largura de banda. Assumindo que a rede Ethernet 10Mbps não pode estar ocupando mais que 80% da sua capacidade para manter as colisões a um nível aceitável, tem-se disponível 7,6 Mbps na rede INF.

Ao contrário, devido à quantidade de tráfego da linha de saída que liga a UFSC com a Telesc (uma linha de 2Mbits/s), e portanto com a MBone, mesmo com a menor taxa de bits gerada pelas fontes de vídeo ocorreria o descarte de pacotes no roteador do NPD. Neste caso, vê-se claramente a necessidade do FiltroGW. Mas outro problema ocorrido nos testes foi que o túnel ligando a UFSC ao MBone não estava operacional.

Para poder verificar a necessidade do filtro na conexão entre a rede ATM e a rede INF, foi necessário sobrecarregar a largura de banda da rede INF, tornando-a escassa. Isso pôde ser realizado através de um da utilização de um PC contendo Linux, ligado a rede INF. Nesse PC, foi executado um processo responsável por sobrecarregar a rede INF, tornando-a lenta e dispendiosa. Tendo a certeza de que o recurso largura de banda da rede INF estava escasso, bastou observar o comportamento das aplicações adaptativas, constatando a baixa na fidelidade da imagem nos demais pontos receptores, localizados na rede ATM.

6.8.3 Testes do Gateway com Filtro Adaptativo RTP

Devido a problemas de tempo, o protótipo do *gateway* com filtro adaptativo RTP não foi concluído. Restando para um trabalho futuro verificar vantagens e desvantagens desta proposta a nível de operação prática e não somente teórica.

6.9 Conclusões

Os experimentos realizados no ambiente de teste nos permitiu questionar porque as aplicações executadas em uma rede como ATM, que atende a maioria dos requisitos necessários para comunicação multimídia, eram tão lentas. Isso nos levou a pesquisar sobre as aplicações multimídia adaptativas e seu efeito sobre a rede. Embora não foi possível realizar todos os testes desejados, porque o túnel que interligava as duas redes com a rede MBone ficou a maior parte do tempo inoperante, e implementação do FiltroGW não foi concluída, foi possível chegar a um modelo que permite escalabilidade de mídia (vídeo) entre duas redes heterogêneas. A utilização desse modelo acrescenta mais atraso e variação de atraso na apresentação da mídia de vídeo, em compensação ele aproveita melhor o recurso de largura de banda, não gera congestionamento no roteador, evitando assim a perda de pacotes que deteriora a qualidade da imagem transmitida.

7. Conclusão

Atualmente existe um grupo de pessoas interessadas em montar um backbone multicast (MBone) interligando todas as universidades brasileiras. Estudos sobre o comportamento dos protocolos e os roteadores utilizados constataram que se existir um ponto da ilha MBone com um roteador mal configurado, ocorre descarte de pacotes e problemas na largura de banda que atingem toda a infraestrutura da rede MBone brasileira. Em compensação quando todos os roteadores são configurados corretamente, é possível participar de eventos com qualidade boa de áudio e vídeo. Mas essa qualidade não se compara a qualidade proporcionada pela arquitetura ATM, que é superior desde que as máquinas onde as aplicações estão executando possuam um hardware bom.

Nesse ambiente de transmissão de dados entre redes heterogêneas todo estudo que venha a acrescentar uma solução a um determinado problema existente, é de grande valia para o contexto geral das aplicações multimídia distribuídas.

Nesse sentido, o trabalho proposto teve alguns propósitos. Em primeiro lugar, elaborar um estudo sobre os requisitos necessários para a comunicação multimídia, e verificar a insuficiência ou não desses requisitos nos ambientes de rede estudados. Foi dado um destaque especial as possíveis implementações de *multicast* na rede ATM, identificando problemas em todas. Em segundo lugar, estudar o comportamento das aplicações multimídia distribuídas, bem como seu funcionamento a nível de protocolo de transporte e alocação de endereços para entrada e saída dos grupos de difusão seletiva. E, a contribuição final, o modelo de um mecanismo que barra a ação das aplicações multimídia adaptativas favorecendo o meio de transmissão que possui maior largura de banda, ao mesmo tempo que adapta o fluxo de vídeo para possibilitar a visualização de uma imagem com qualidade nos ambientes de rede que possuem menos recursos.

O modelo proposto nessa dissertação protege os usuários da rede ATM da ação das aplicações adaptativas quando existe usuários localizados na rede Ethernet, mas as aplicações adaptativas mantém seu comportamento normal junto aos usuários de rede

ATM. Portanto ele mantém funcionalidade e as vantagens desse tipo de aplicação eliminando problema que existe quando elas são executadas em ambientes de redes heterogêneos.

Dificuldades encontradas

O túnel responsável por interligar a rede UFSC com o MBone não ficou operacional durante tempo suficiente para que fossem realizados testes cujos resultados poderiam contribuir o estudo sobre o comportamento das aplicações multimídia distribuídas e para definição do modelo FiltroGW.

Não foi possível instalar o protocolo *Classical* IP na estação Lucy, devido a falta de *drivers* da SUN. A SUN possui suporte ao CLIP, mas sem a implementação de *multicast*. Sem *multicast* não é possível difundir as conferências na rede.

Algumas ferramentas utilizadas para transmissão de eventos do MBone funcionam bem no sistema operacional UNIX, mas no Windows elas apresentaram algumas deficiências operacionais.

Trabalhos Futuros

O mecanismo modelado satisfaz as necessidades dos dois ambientes de rede, não baixa a qualidade das conferências da rede ATM e adapta o fluxo de vídeo de acordo com a disponibilidade de recursos da rede Ethernet. Mas para isso ser possível ocorre um certo atraso e variação de atraso no fluxo de vídeo recebido pelos receptores localizados na rede Ethernet. Trabalhos futuros podem estudar o comportamento dos algoritmos existentes (utilizados nos processos de codificação e decodificação do módulo filtro) com a finalidade de identificar o algoritmo mais adequado para a funcionalidade do filtro, ou seja, aquele que implica em menor atraso e variação de atraso.

8. Referências Bibliográficas

[Bellcore 96] Bellcore. "Support for Multicast over UNI 3.0/3.1 based ATM Networks." Request for Comments 2022. Network Working Group, Novembro 1996.

[Bolot, 94] J.C. Bolot, T. Turletti, I. Wakeman. "Scalable Feedback Control for Multicast Video Distribution in the Internet." Proc. ACM SIGCOMM pp. 58-67, 1994.

[Casner, 92] Casner, S.; Deering, S. First IETF Internet Audiocast, ACM SIGCOMM Computer Communications Review. San Diego – Califórnia, jul. 1992, pp. 92-97. Documento também disponível em ftp://venera.isi.edu/pub/MBone/ietf-audiocast-article.ps.

[Casner, 95] Casner, S. Frequently Asked Question (FAQ) on *Multicast Backbone* (MBONE). Dez. 1995. Documento disponível em http://www.reserach.att.com/MBone-faq.html.

[Diot, 95] Ch. Diot. "Adaptative Applications and QoS Guarantees." Proc. IEEE International Conference on Multimedia and Networking. IEEE Computer Society Press pp 99-106, Aizu, Japan, 1995.

[Fluckiger 95] Francois Fluckiger."Understanding Networked Multimedia Aplications and Technology." New Jersey: Ed. Prentice Hall Interational (UK) Limited, 1995

[Gecsei 97] Jan Gecsei, "Adaptation in Distribuited Multimedia Systems." IEEE International Conference on Multimedia and Networking, Université de Moltréal, 1997

[HP 94] Hewlett-Packard Laboratories. "Classical IP and ARP over ATM". Request for Comments 2226. Network Working Group. Janeiro, 1994.

[Hunter, 98] Jane Hunter, Varuni Witana, Mark Antoniades, "A Review of Video Streaming over the Internet".

[IBM 97] IBM Corporation. "IP Broadcast over ATM Networks". Request for Comments 2226. Network Working Group. Outubro, 1997.

[IPF 99] IP Filter 1993 –1999. Versão 3.5. Darren Reed. http://coombs.anu.edu.au/~avalon/ip-filter.html

[ISI 95] ISI. "ATM Signaling Support for IP over ATM". Request for Comments 1755. Network Working Group. Fevereiro, 1995.

[Käppner, 94] T. Käppner, L.C. Wolf. "Media Scaling in Distributed Multimedia Object Services. In Multimedia: Advanced Teleservices and High-Speed Communication Architectures." R. Steinmetz .(Ed.). Springer LNCS 868, pp.34-43, 1994.

[Kieling 97] Alexandre Briani Kieling – "Organização da RedeUFSC para Ligação com o MBONE".
Florianópolis, 1997. Trabalho de Conclusão de Curso – INE, UFSC.

[Kumar 96] Vinay Kumar, "MBone - Interactive Multimedia on the Internet" Indianapolis. Ed: New Riders Publishing, 1996.

[Kno 98] Franklin Kno, Wolfgang Effelsberg, J.J. Garcia, Luna, Aceves. "Multimedia Communications Protocols and Applications" New Jersey: Ed. Prentice Hall PTR, 1998.

[Koegel 94] John F. Koegel Buford. "Multimedia Systems."
ACM Press, New York: Ed. University of Massachusetts Lowell Contributing Editor, 1994

[Kyas 95] Othmar Kyas, "ATM networks."
2 ed. London: International Thomson Computer Press, 1995.

[Lu 96] Guojun Lu, "Communication and Computing for Distributed Multimedia Systems"
1 Ed. Boston, London: Artech House Inc., 1996

[McCanne, 96] S. McCanne. "Scalable Compression and Transmission of Internet Multicast Video". Ph.D dissertation, University of California, Berkeley, 1996.

[McCanne 99] Steven McCanne. "Scalable Multimedia Communication Using IP Multicast and Lightweight Sessions". IEEE Internet Computing, pp 33-45, março/abril 1999.

[Medina 97] Nelkis de la Orden Medina. "Um Suporte para Aplicações Cooperativas sobre a Internet."
Florianópolis, março de 1997. Mestrado eng. Elétrica UFSC.

[Melchiors 97] Cristina Melchiors. "Sistemas Interpessoais de Videoconferência (MBone)".
Porto Alegre, janeiro de 1997. TI n 596 – UFRGS.
www.penta.ufrgs.br/~cristina/mbone/ti/indiceti.html

[Oosthoek 97] Simon Oosthoek. "Survey of Multicast Support for IP over ATM Implementation Purposes."
Julho de 1997. Thesis – University of Twente.
http://www.huygens.org/people/oosthoek/thesis1

[Pasquale 98] Joseph C. Pasquale, George C. Polyzos, George Xylomenos. "The Multimedia Multicasting Problem."
Multimedia Systems, pp 6:43-59, 1998.

[Proteon 94] Proteon Inc. "Multicast Extensions to OSPF". Request for Comments 1584. Network Working Group. Março, 1994.

[Raghavan 98] S.V. Raghavan, Satish K. Tripathi. "Networked Multimedia Systems Concepts, Architecture & Design."
New Jersey: Ed. Prentice Hall, 1998

[Soares 95] Luiz Fernando Gomes Soares, Guido Lemos, Sérgio Colcher. "Redes de computadores: das LANs MANs e WANs às redes ATM". 2.ed. Rio de Janeiro: Editora Campus, 1995.

[Soares 97] Luiz Fernando Gomes Soares, Marcelo Blois Ribeiro, Ricardo Choren Noya. "Emulação de LAN Sobre ATM".
Rio de Janeiro: PUC, 1997
http://www.telemidia.puc-rio.br

[Souza 97] Adriano de Souza, Richard Robert Reinert. "Implantação de uma Rede ATM Integrada com a RedeUFSC"
Florianópolis, 1997. Trabalho de Conclusão de Curso – INE, UFSC.

[Stallings 98] William Stallings. "TCP/IP And ATM Design Principles."
New Jersey: Ed. Prentice Hall, 1998.

[Schulzrinne, 97] H. Schulzrinne et al. RTP: A Transport Protocol for Real-Time Applications. Internet Draft draft-ietf-avt-rtp-02.new-01.ps, 1997.

[Stanford 88] Stanford University. "Distance Vector Multicast Routing Protocol". Request for Comments 1075. Network Working Group. Novembro 1988.

[Stanford 89] Stanford University. "Host Extensions for IP Multicasting" Request for Comments 1112. Network Working Group. Agosto, 1989.

[Tanenbaum 94] Andrew S. Tanenbaum. "Redes de Computadores."
2 ed. Rio de Janeiro: Editora Campus, 1994.

[Tanenbaum 96] Andrew S. Tanenbaum. "Redes de Computadores." 3 ed. New Jersey: Ed. Prentice Hall, 1996.

[Telecom 93] Telecom Finland." Multiprotocol Encapsulation over ATM Adaptation Layer 5". Request for Comments 1483. Network Working Group. Julho, 1993.

[Wittig, 94] H. Wittig, J. Winckler, J. Sandvoss. "Network Layer Scaling: Congestion Control in Multimedia Communication with Heterogeneous Networks and Receivers." Proc. Multimedia Transport and Teleservices, Springer LNCS 882, pp. 274-293, 1994.

[Xerox, 96] Xerox Palo Alto Research Center. "RTP A Transport Protocol for Real-Time Applications." Request for Comments 1889. Network Working Group. Janeiro, 1996.

ANEXO 1

Sintaxe do *IP Filter*, desenvolvido para a plataforma SUN. Formato dos arquivos.

```
File Formats                                                    IPF(4)


NAME
     ipf - packet filtering kernel interface

SYNOPSIS
     #include <netinet/ip_compat.h>
     #include <netinet/ip_fil.h>

IOCTLS
     To add and delete rules to the filter  list,  three  'basic'
     ioctls are provided for use.  The ioctl's are called as:

          ioctl(fd, SIOCADDFR, struct frentry *)
          ioctl(fd, SIOCDELFR, struct frentry *)
          ioctl(fd, SIOCIPFFL, int *)

     However, the full complement is as follows:

          ioctl(fd, SIOCADAFR, struct frentry *) (same as SUICADDFR)
          ioctl(fd, SIOCRMAFR, struct frentry *) (same as SUICDELFR)
          ioctl(fd, SIOCADIFR, struct frentry *)
          ioctl(fd, SIOCRMIFR, struct frentry *)
          ioctl(fd, SIOCINAFR, struct frentry *)
          ioctl(fd, SIOCINIFR, struct frentry *)
          ioctl(fd, SIOCSETFF, u_int *)
          ioctl(fd, SIOGGETFF, u_int *)
          ioctl(fd, SIOCGETFS, struct friostat *)
          ioctl(fd, SIOCIPFFL, int *)
          ioctl(fd, SIOCIPFFB, int *)
          ioctl(fd, SIOCSWAPA, u_int *)
          ioctl(fd, SIOCFRENB, u_int *)
          ioctl(fd, SIOCFRSYN, u_int *)
          ioctl(fd, SIOCFRZST, struct friostat *)
          ioctl(fd, SIOCZRLST, struct frentry *)
          ioctl(fd, SIOCAUTHW, struct fr_info *)
          ioctl(fd, SIOCAUTHR, struct fr_info *)
          ioctl(fd, SIOCATHST, struct fr_authstat *)

     The variations, SIOCADAFR vs. SIOCADIFR, allow operation  on
     the  two  lists,  active and inactive, respectively.  All of
     these ioctl's are implemented as being  routing  ioctls  and
     thus  the  same rules for the various routing ioctls and the
     file descriptor are employed, mainly being that the fd  must
     be  that  of  the  device  associated with the module (i.e.,
     /dev/ipl).
```

The three groups of ioctls above perform adding rules to the end of the list (SIOCAD*), deletion of rules from any place in the list (SIOCRM*) and insertion of a rule into the list (SIOCIN*). The rule place into which it is inserted is stored in the "fr_hits" field, below.

```
typedef struct  frentry {
        struct  frentry *fr_next;
        u_short fr_group;       /* group to which this rule
belongs */
        u_short fr_grhead;      /* group # which this rule starts
*/
        struct  frentry *fr_grp;
        int     fr_ref;         /* reference count - for grouping
*/
        void    *fr_ifa;
#if BSD >= 199306
        void    *fr_oifa;
#endif
        /*
         * These are only incremented when a packet  matches this
rule and
         * it is the last match
         */
        U_QUAD_T        fr_hits;
        U_QUAD_T        fr_bytes;
        /*
         * Fields after this may not change whilst in the kernel.
         */
        struct  fr_ip   fr_ip;
        struct  fr_ip   fr_mip; /* mask structure */

        u_char  fr_tcpfm;       /* tcp flags mask */
        u_char  fr_tcpf;        /* tcp flags */

        u_short fr_icmpm;       /* data for ICMP packets (mask)
*/
        u_short fr_icmp;

        u_char  fr_scmp;        /* data for port comparisons */
        u_char  fr_dcmp;
        u_short fr_dport;
        u_short fr_sport;
        u_short fr_stop;        /* top port for <> and >< */
        u_short fr_dtop;        /* top port for <> and >< */
        u_32_t  fr_flags;       /* per-rule flags && options (see
below) */
        u_short fr_skip;        /* # of rules to skip */
        u_short fr_loglevel;    /* syslog log facility + priority
*/
        int     (*fr_func) __P((int, ip_t *, fr_info_t *));
        char    fr_icode;       /* return ICMP code */
        char    fr_ifname[IFNAMSIZ];
#if BSD > 199306
        char    fr_oifname[IFNAMSIZ];
#endif
        struct  frdest  fr_tif; /* "to" interface */
```

```
        struct  frdest  fr_dif; /* duplicate packet interfaces */
} frentry_t;
```

When adding a new rule, all unused fields (in the filter rule) should be initialised to be zero. To insert a rule, at a particular position in the filter list, the number of the rule which it is to be inserted before must be put in the "fr_hits" field (the first rule is number 0).

Flags which are recognised in fr_pass:

```
        FR_BLOCK        0x000001    /* do not allow packet to pass */
        FR_PASS         0x000002    /* allow packet to pass */
        FR_OUTQUE       0x000004    /* outgoing packets */
        FR_INQUE        0x000008    /* ingoing packets */
        FR_LOG          0x000010    /* Log */
        FR_LOGB         0x000011    /* Log-fail */
        FR_LOGP         0x000012    /* Log-pass */
        FR_LOGBODY      0x000020    /* log the body of packets too */
        FR_LOGFIRST     0x000040    /* log only the first packet to
match */
        FR_RETRST       0x000080    /* return a TCP RST packet if
blocked */
        FR_RETICMP      0x000100    /* return an ICMP packet if
blocked */
        FR_FAKEICMP     0x00180     /* Return ICMP unreachable with
fake source */
        FR_NOMATCH      0x000200    /* no match occured */
        FR_ACCOUNT      0x000400    /* count packet bytes */
        FR_KEEPFRAG     0x000800    /* keep fragment information */
        FR_KEEPSTATE    0x001000    /* keep `connection' state
information */
        FR_INACTIVE     0x002000
        FR_QUICK        0x004000    /* match & stop processing list
*/
        FR_FASTROUTE    0x008000    /* bypass normal routing */
        FR_CALLNOW      0x010000    /* call another function
(fr_func) if matches */
        FR_DUP          0x020000    /* duplicate the packet */
        FR_LOGORBLOCK   0x040000    /* block the packet if it can't
be logged */
        FR_NOTSRCIP     0x080000    /* not the src IP# */
        FR_NOTDSTIP     0x100000    /* not the dst IP# */
        FR_AUTH         0x200000    /* use authentication */
        FR_PREAUTH      0x400000    /* require preauthentication */
```

Values for fr_scomp and fr_dcomp (source and destination port value comparisons) :

```
        FR_NONE         0
        FR_EQUAL        1
        FR_NEQUAL       2
        FR_LESST        3
        FR_GREATERT     4
        FR_LESSTE       5
        FR_GREATERTE    6
        FR_OUTRANGE     7
        FR_INRANGE      8
```

The third ioctl, SIOCIPFFL, flushes either the input filter list, the output filter list or both and it returns the number of filters removed from the list(s). The values which it will take and recognise are FR_INQUE and FR_OUTQUE (see above). This ioctl is also implemented for /dev/ipstate and will flush all state tables entries if passed 0 or just all those which are not established if passed 1.

General Logging Flags
There are two flags which can be set to log packets independantly of the rules used. These allow for packets which are either passed or blocked to be logged. To set (and clear)/get these flags, two ioctls are provided:

SIOCSETFF Takes an unsigned integer as the parameter. The flags are then set to those provided (clearing/setting all in one).

```
FF_LOGPASS      0x10000000
FF_LOGBLOCK     0x20000000
FF_LOGNOMATCH   0x40000000
FF_BLOCKNONIP   0x80000000    /* Solaris 2.x
only */
```

SIOCGETFF Takes a pointer to an unsigned integer as the parameter. A copy of the flags currently in used is copied to user space.

Filter statistics
Statistics on the various operations performed by this package on packets is kept inside the kernel. These statistics apply to packets traversing through the kernel. To retrieve this structure, use this ioctl:

```
ioctl(fd, SIOCGETFS, struct friostat *)

struct  friostat        {
        struct  filterstats     f_st[2];
        struct  frentry         *f_fin[2];
        struct  frentry         *f_fout[2];
        struct  frentry         *f_acctin[2];
        struct  frentry         *f_acctout[2];
        struct  frentry         *f_auth;
        u_long  f_froute[2];
        int     f_active;       /* 1 or 0 - active rule set */
        int     f_defpass;      /* default pass - from fr_pass */
        int     f_running;      /* 1 if running, else 0 */
        int     f_logging;      /* 1 if enabled, else 0 */
        char    f_version[32];  /* version string */
};

struct    filterstats {
        u_long  fr_pass;        /* packets allowed */
        u_long  fr_block;       /* packets denied */
        u_long  fr_nom;         /* packets which don't match any
rule */
        u_long  fr_ppkl;        /* packets allowed and logged */
```

```
            u_long  fr_bpkl;        /* packets denied and logged */
            u_long  fr_npkl;        /* packets unmatched and logged
*/
            u_long  fr_pkl;         /* packets logged */
            u_long  fr_skip;        /* packets to be logged but
buffer full */
            u_long  fr_ret;         /* packets for which a return is
sent */

            u_long  fr_acct;        /* packets for which counting was
performed */
            u_long  fr_bnfr;        /* bad attempts to allocate
fragment state */
            u_long  fr_nfr;         /* new fragment state kept */
            u_long  fr_cfr;         /* add new fragment state but
complete pkt */
            u_long  fr_bads;        /* bad attempts to allocate
packet state */
            u_long  fr_ads;         /* new packet state kept */
            u_long  fr_chit;        /* cached hit */
            u_long  fr_pull[2];     /* good and bad pullup attempts
*/
     #if SOLARIS
            u_long  fr_notdata;     /* PROTO/PCPROTO that have no
data */
            u_long  fr_nodata;      /* mblks that have no data */
            u_long  fr_bad;         /* bad IP packets to the filter
*/
            u_long  fr_notip;       /* packets passed through no on
ip queue */
            u_long  fr_drop;        /* packets dropped - no info for
them! */
     #endif
     };
```

If we wanted to retrieve all the statistics and reset the counters back to 0, then the ioctl() call would be made to SIOCFRZST rather than SIOCGETFS. In addition to the statistics above, each rule keeps a hit count, counting both number of packets and bytes. To reset these counters for a rule, load the various rule information into a frentry structure and call SIOCZRLST.

Swapping Active lists

IP Filter supports two lists of rules for filtering and accounting: an active list and an inactive list. This allows for large scale rule base changes to be put in place atomically with otherwise minimal interruption. Which of the two is active can be changed using the SIOCSWAPA ioctl. It is important to note that no passed argument is recognised and that the value returned is that of the list which is now inactive.

FILES

/dev/ipauth
/dev/ipl
/dev/ipnat
/dev/ipstate

ANEXO 2

Sintaxe do IP Filter, desenvolvido para a plataforma SUN. Cabeçalhos, tabelas e macros.

```
Headers, Environments, and Macros                          IPF(5)


NAME
     ipf, ipf.conf - IP packet filter rule syntax

DESCRIPTION
     A rule file for ipf may have any name or even be stdin.  As
     ipfstat  produces  parseable rules as output when displaying
     the internal kernel filter lists, it is quite  plausible  to
     use  its  output to feed back into ipf.  Thus, to remove all
     filters on input packets, the following could be done:

     # ipfstat -i | ipf -rf -

GRAMMAR
     The format used by ipf for construction of  filtering  rules
     can be described using the following grammar in BNF:

     filter-rule = [ insert ] action in-out [ options ] [ tos ] [ ttl
]
                   [ proto ] [ ip ] [ group ].

     insert    = "@" decnumber .
     action    = block | "pass" | log | "count" | skip | auth | call .
     in-out    = "in" | "out" .
     options   = [ log ] [ "quick" ] [ "on" interface-name [ dup ] [
froute ] ] .
     tos  = "tos" decnumber | "tos" hexnumber .
     ttl  = "ttl" decnumber .
     proto     = "proto" protocol .
     ip   = srcdst [ flags ] [ with withopt ] [ icmp ] [ keep ] .
     group     = [ "head" decnumber ] [ "group" decnumber ] .

     block     = "block" [ icmp[return-code] | "return-rst" ] .
     auth    = "auth" | "preauth" .
     log  = "log" [ "body" ] [ "first" ] [ "or-block" ] [ "level"
loglevel ] .
     call = "call" [ "now" ] function-name .
     skip = "skip" decnumber .
     dup  = "dup-to" interface-name[":"ipaddr] .
     froute    = "fastroute" | "to" interface-name .
     protocol = "tcp/udp" | "udp" | "tcp" | "icmp" | decnumber .
     srcdst    = "all" | fromto .
     fromto    = "from" [ "!" ] object "to" [ "!" ] object .

     icmp = "return-icmp" | "return-icmp-as-dest" .
     object    = addr [ port-comp | port-range ] .
```

```
     addr = "any" | nummask | host-name [ "mask" ipaddr | "mask"
hexnumber ] .
     port-comp = "port" compare port-num .
     port-range = "port" port-num range port-num .
     flags     = "flags" flag { flag } [ "/" flag { flag } ] .
     with = "with" | "and" .
     icmp = "icmp-type" icmp-type [ "code" decnumber ] .
     return-code = "("icmp-code")" .
     keep = "keep" "state" | "keep" "frags" .
     loglevel = facility"."priority | priority .
     nummask   = host-name [ "/" decnumber ] .
     host-name = ipaddr | hostname | "any" .
     ipaddr    = host-num "." host-num "." host-num "." host-num .
     host-num = digit [ digit [ digit ] ] .
     port-num = service-name | decnumber .

     withopt = [ "not" | "no" ] opttype [ withopt ] .
     opttype = "ipopts" | "short" | "frag" | "opt" ipopts  .
     optname   = ipopts [ "," optname ] .
     ipopts  = optlist | "sec-class" [ secname ] .
     secname   = seclvl [ "," secname ] .
     seclvl  = "unclass" | "confid" | "reserv-1" | "reserv-2" |
"reserv-3" |
            "reserv-4" | "secret" | "topsecret" .
     icmp-type = "unreach" | "echo" | "echorep" | "squench" | "redir"
|
              "timex" | "paramprob" | "timest" | "timestrep" |
"inforeq" |
              "inforep" | "maskreq" | "maskrep"  | decnumber .
     icmp-code = decumber | "net-unr" | "host-unr" | "proto-unr" |
"port-unr" |
              "needfrag" | "srcfail" | "net-unk" | "host-unk" |
"isolate" |
              "net-prohib" | "host-prohib" | "net-tos" | "host-tos" .
     optlist   = "nop" | "rr" | "zsu" | "mtup" | "mtur" | "encode" |
"ts" |
            "tr" | "sec" | "lsrr" | "e-sec" | "cipso" | "satid" |
"ssrr" |
            "addext" | "visa" | "imitd" | "eip" | "finn" .
     facility = "kern" | "user" | "mail" | "daemon" | "auth" |
"syslog" |
             "lpr" | "news" | "uucp" | "cron" | "ftp" | "authpriv" |
             "audit" | "logalert" | "local0" | "local1" | "local2" |
             "local3" | "local4" | "local5" | "local6" | "local7" .
     priority = "emerg" | "alert" | "crit" | "err" | "warn" | "notice"
|
             "info" | "debug" .

     hexnumber = "0" "x" hexstring .
     hexstring = hexdigit [ hexstring ] .
     decnumber = digit [ decnumber ] .

     compare = "=" | "!=" | "<" | ">" | "<=" | ">=" | "eq" | "ne" |
"lt" |
            "gt" | "le" | "ge" .
     range     = "<>" | "><" .
     hexdigit = digit | "a" | "b" | "c" | "d" | "e" | "f" .
     digit     = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" |
"9" .
```

flag = "F" | "S" | "R" | "P" | "A" | "U" .

This syntax is somewhat simplified for readability, some combinations that match this grammar are disallowed by the software because they do not make sense (such as tcp flags for non-TCP packets).

FILTER RULES

The "briefest" valid rules are (currently) no-ops and are of the form:

```
block in all
pass in all
log out all
count in all
```

Filter rules are checked in order, with the last matching rule determining the fate of the packet (but see the quick option, below).

Filters are installed by default at the end of the kernel's filter lists, prepending the rule with @n will cause it to be inserted as the n'th entry in the current list. This is especially useful when modifying and testing active filter rulesets. See ipf(1) for more information.

ACTIONS

The action indicates what to do with the packet if it matches the rest of the filter rule. Each rule MUST have an action. The following actions are recognised:

block
: indicates that the packet should be flagged to be dropped. In response to blocking a packet, the filter may be instructed to send a reply packet, either an ICMP packet (return-icmp), an ICMP packet masquerading as being from the original packet's destination (return-icmp-as-dest), or a TCP "reset" (return-rst). An ICMP packet may be generated in response to any IP packet, and its type may optionally be specified, but a TCP reset may only be used with a rule which is being applied to TCP packets. When using return-icmp or return-icmp-as-dest, it is possible to specify the actual unreachable `type'. That is, whether it is a network unreachable, port unreachable or even administratively prohibitied. This is done by enclosing the ICMP code associated with it in parenthesis directly following return-icmp or return-icmp-as-dest as follows:

```
block return-icmp(11) ...
```

Would return a Type-Of-Service (TOS) ICMP unreachable error.

pass will flag the packet to be let through the filter.

log causes the packet to be logged (as described in the LOGGING section below) and has no effect on whether the packet will be allowed through the filter.

count

causes the packet to be included in the accounting statistics kept by the filter, and has no effect on whether the packet will be allowed through the filter. These statistics are viewable with ipfstat(8).

call this action is used to invoke the named function in the kernel, which must conform to a specific calling interface. Customised actions and semantics can thus be implemented to supplement those available. This feature is for use by knowledgeable hackers, and is not currently documented.

skip <n>
causes the filter to skip over the next __n filter rules. If a rule is inserted or deleted inside the region being skipped over, then the value of __n is adjusted appropriately.

auth this allows authentication to be performed by a user-space program running and waiting for packet information to validate. The packet is held for a period of time in an internal buffer whilst it waits for the program to return to the kernel the __r__e__a__l flags for whether
it should be allowed through or not. Such a program might look at the source address and request some sort of authentication from the user (such as a password) before allowing the packet through or telling the kernel to drop it if from an unrecognised source.

preauth
tells the filter that for packets of this class, it should look in the pre-authenticated list for further clarification. If no further matching rule is found, the packet will be dropped (the FR_PREAUTH is not the same as FR_PASS). If a further matching rule is found, the result from that is used in its instead. This might be used in a situation where a person __l__o__g__s __i__n to
the firewall and it sets up some temporary rules defining the access for that person.

The next word must be either in or out. Each packet moving through the kernel is either inbound (just been received on an interface, and moving towards the kernel's protocol processing) or outbound (transmitted or forwarded by the stack, and on its way to an interface). There is a requirement that each filter rule explicitly state which side of the I/O it is to be used on.

OPTIONS
The list of options is brief, and all are indeed optional. Where options are used, they must be present in the order shown here. These are the currently supported options:

log indicates that, should this be the last matching rule, the packet header will be written to the ipl log (as described in the LOGGING section below).

quick
allows "short-cut" rules in order to speed up the filter or override later rules. If a packet matches a filter rule which is marked as quick, this rule will be the last rule checked, allowing a "short-circuit" path to avoid processing later rules for this packet. The current status of the packet (after any effects of the current rule) will determine whether it is passed or blocked.

If this option is missing, the rule is taken to be a "fall-through" rule, meaning that the result of the match (block/pass) is saved and that processing will continue to see if there are any more matches.

on allows an interface name to be incorporated into the matching procedure. Interface names are as printed by "netstat -i". If this option is used, the rule will only match if the packet is going through that interface in the specified direction (in/out). If this option is absent, the rule is taken to be applied to a packet regardless of the interface it is present on (i.e. on all interfaces). Filter rulesets are common to all interfaces, rather than having a filter list for each interface.

This option is especially useful for simple IP-spoofing protection: packets should only be allowed to pass inbound on the interface from which the specified source address would be expected, others may be logged and/or dropped.

dup-to
causes the packet to be copied, and the duplicate packet to be sent outbound on the specified interface, optionally with the destination IP address changed to that specified. This is useful for off-host logging, using a network sniffer.

to causes the packet to be moved to the outbound queue on the specified interface. This can be used to circumvent kernel routing decisions, and even to bypass the rest of the kernel processing of the packet (if applied to an inbound rule). It is thus possible to construct a firewall that behaves transparently, like a filtering hub or switch, rather than a router. The fastroute keyword is a synonym for this option.

MATCHING PARAMETERS
The keywords described in this section are used to describe attributes of the packet to be used when determining whether rules match or don't match. The following general-purpose attributes are provided for matching, and must be used in this order:

tos packets with different Type-Of-Service values can be filtered. Individual service levels or combinations can be filtered upon. The value for the TOS mask can either be represented as a hex number or a decimal

integer value.

ttl packets may also be selected by their Time-To-Live value. The value given in the filter rule must exactly match that in the packet for a match to occur. This value can only be given as a decimal integer value.

proto
allows a specific protocol to be matched against. All protocol names found in /etc/protocols are recognised and may be used. However, the protocol may also be given as a DECIMAL number, allowing for rules to match your own protocols, or new ones which would out-date any attempted listing.

The special protocol keyword tcp/udp may be used to match either a TCP or a UDP packet, and has been added as a convenience to save duplication of otherwise-identical rules.

The from and to keywords are used to match against IP addresses (and optionally port numbers). Rules must specify BOTH source and destination parameters.

IP addresses may be specified in one of two ways: as a numerical address/mask, or as a hostname mask netmask. The hostname may either be a valid hostname, from either the hosts file or DNS (depending on your configuration and library) or of the dotted numeric form. There is no special designation for networks but network names are recognised. Note that having your filter rules depend on DNS results can introduce an avenue of attack, and is discouraged.

There is a special case for the hostname any which is taken to be 0.0.0.0/0 (see below for mask syntax) and matches all IP addresses. Only the presence of "any" has an implied mask, in all other situations, a hostname MUST be accompanied by a mask. It is possible to give "any" a hostmask, but in the context of this language, it is non-sensical.

The numerical format "x/y" indicates that a mask of y consecutive 1 bits set is generated, starting with the MSB, so a y value of 16 would give 0xffff0000. The symbolic "x mask y" indicates that the mask y is in dotted IP notation or a hexadecimal number of the form 0x12345678. Note that all the bits of the IP address indicated by the bitmask must match the address on the packet exactly; there isn't currently a way to invert the sense of the match, or to match ranges of IP addresses which do not express themselves easily as bitmasks (anthropomorphization; it's not just for breakfast anymore).

If a port match is included, for either or both of source and destination, then it is only applied to TCP and UDP packets. If there is no proto match parameter, packets from both protocols are compared. This is equivalent to "proto tcp/udp". When composing port comparisons, either the service name or an integer port number may be used. Port comparisons may be done in a number of forms, with a number of

comparison operators, or port ranges may be specified. When the port appears as part of the from object, it matches the source port number, when it appears as part of the to object, it matches the destination port number. See the examples for more information.

The all keyword is essentially a synonym for "from any to any" with no other match parameters.

Following the source and destination matching parameters, the following additional parameters may be used:

with is used to match irregular attributes that some packets may have associated with them. To match the presence of IP options in general, use with ipopts. To match packets that are too short to contain a complete header, use with short. To match fragmented packets, use with frag. For more specific filtering on IP options, individual options can be listed.

Before any parameter used after the with keyword, the word not or no may be inserted to cause the filter rule to only match if the option(s) is not present.

Multiple consecutive with clauses are allowed. Alternatively, the keyword and may be used in place of with, this is provided purely to make the rules more readable ("with ... and ..."). When multiple clauses are listed, all those must match to cause a match of the rule.

flags
is only effective for TCP filtering. Each of the letters possible represents one of the possible flags that can be set in the TCP header. The association is as follows:

F - FIN
S - SYN
R - RST
P - PUSH
A - ACK
U - URG

The various flag symbols may be used in combination, so that "SA" would represent a SYN-ACK combination present in a packet. There is nothing preventing the specification of combinations, such as "SFR", that would not normally be generated by law-abiding TCP implementations. However, to guard against weird aberrations, it is necessary to state which flags you are filtering against. To allow this, it is possible to set a mask indicating which TCP flags you wish to compare (i.e., those you deem significant). This is done by appending "/<flags>" to the set of TCP flags you wish to match against, e.g.:

```
... flags S
          # becomes "flags S/AUPRFS" and will match
```

```
                # packets with ONLY the SYN flag set.

        ... flags SA
                # becomes "flags SA/AUPRFS" and will match any
                # packet with only the SYN and ACK flags set.

        ... flags S/SA
                # will match any packet with just the SYN flag set
                # out of the SYN-ACK pair; the common "establish"
                # keyword action.  "S/SA" will NOT match a packet
                # with BOTH SYN and ACK set, but WILL match "SFP".
```

icmp-type
: is only effective when used with proto icmp and must NOT be used in conjuction with flags. There are a number of types, which can be referred to by an abbreviation recognised by this language, or the numbers with which they are associated can be used. The most important from a security point of view is the ICMP redirect.

KEEP HISTORY

The second last parameter which can be set for a filter rule is whether or not to record historical information for that packet, and what sort to keep. The following information can be kept:

state
: keeps information about the flow of a communication session. State can be kept for TCP, UDP, and ICMP packets.

frags
: keeps information on fragmented packets, to be applied to later fragments.

allowing packets which match these to flow straight through, rather than going through the access control list.

GROUPS

The last pair of parameters control filter rule "grouping". By default, all filter rules are placed in group 0 if no other group is specified. To add a rule to a non-default group, the group must first be started by creating a group __h__e__a__d. If a packet matches a rule which is the __h__e__a__d of a group, the filter processing then switches to the group, using that rule as the default for the group. If quick is used with a head rule, rule processing isn't stopped until it has returned from processing the group.

A rule may be both the head for a new group and a member of a non-default group (head and group may be used together in a rule).

head <n>
: indicates that a new group (number n) should be created.

group <n>
indicates that the rule should be put in group (number n) rather than group 0.

LOGGING
When a packet is logged, with either the log action or option, the headers of the packet are written to the ipl packet logging psuedo-device. Immediately following the log keyword, the following qualifiers may be used (in order):

body indicates that the first 128 bytes of the packet contents will be logged after the headers.

first
If log is being used in conjunction with a "keep" option, it is recommended that this option is also applied so that only the triggering packet is logged and not every packet which thereafter matches state information.

or-block
indicates that, if for some reason the filter is unable to log the packet (such as the log reader being too slow) then the rule should be interpreted as if the action was block for this packet.

level <loglevel>
indicates what logging facility and priority, or just priority with the default facility being used, will be used to log information about this packet using ipmon's -s option.

See ipl(4) for the format of records written to this device. The ipmon(8) program can be used to read and format this log.

EXAMPLES
The quick option is good for rules such as:

block in quick from any to any with ipopts

which will match any packet with a non-standard header length (IP options present) and abort further processing of later rules, recording a match and also that the packet should be blocked.

The "fall-through" rule parsing allows for effects such as this:

```
block in from any to any port < 6000
pass in from any to any port >= 6000
block in from any to any port > 6003
```

which sets up the range 6000-6003 as being permitted and all others being denied. Note that the effect of the first rule is overridden by subsequent rules. Another (easier) way to do the same is:

```
block in from any to any port 6000 <> 6003
```

```
pass in from any to any port 5999 >< 6004
```

Note that both the "block" and "pass" are needed here to effect a result as a failed match on the "block" action does not imply a pass, only that the rule hasn't taken effect. To then allow ports < 1024, a rule such as:

```
pass in quick from any to any port < 1024
```

would be needed before the first block. To create a new group for processing all inbound packets on le0/le1/lo0, with the default being to block all inbound packets, we would do something like:

```
block in all
block in quick on le0 all head 100
block in quick on le1 all head 200
block in quick on lo0 all head 300
```

and to then allow ICMP packets in on le0, only, we would do:

```
pass in proto icmp all group 100
```

Note that because only inbound packets on le0 are used processed by group 100, there is no need to respecify the interface name. Likewise, we could further breakup processing of TCP, etc, as follows:

```
block in proto tcp all head 110 group 100
pass in from any to any port = 23 group 110
```

and so on. The last line, if written without the groups would be:

```
pass in on le0 proto tcp from any to any port = telnet
```

Note, that if we wanted to say "port = telnet", "proto tcp" would need to be specified as the parser interprets each rule on its own and qualifies all service/port names with the protocol specified.

FILES

```
/dev/ipauth
/dev/ipl
/dev/ipstate
/etc/hosts
/etc/services
```

ANEXO 3

Sintaxe do IP Filter, desenvolvido para plataforma SUN. Rotinas de manutenção.

```
Maintenance Procedures                                    IPF(8)

NAME
     ipf - alters packet filtering lists for IP packet input  and
     output

SYNOPSIS
     ipf [ -AdDEInoPrsUvVyzZ ] [ -l <block|pass|nomatch> ]  [  -F
     <i|o|a|s|S> ] -f <__f__i__l__e__n__a__m__e> [ -f
<__f__i__l__e__n__a__m__e> [...]]

DESCRIPTION
     ipf opens the filenames listed (treating "-" as  stdin)  and
     parses  the file for a set of rules which are to be added or
     removed from the packet filter rule set.

     Each rule processed by ipf is added to the kernel's internal
     lists  if there are no parsing problems.  Rules are added to
     the end of the internal lists, matching the order  in  which
     they appear when given to ipf.

OPTIONS
     -A   Set the  list  to  make  changes  to  the  active  list
          (default).

     -d   Turn debug mode on.  Causes a hexdump of  filter  rules
          to be generated as it processes each one.

     -D   Disable the filter (if  enabled).   Not  effective  for
          loadable kernel versions.

     -E   Enable the filter (if  disabled).   Not  effective  for
          loadable kernel versions.

     -F <i|o|a>
          This option specifies which filter list to flush.  The
          parameter should either be "i" (input), "o" (output) or
          "a" (remove all filter rules).  Either a single  letter
          or  an entire word starting with the appropriate letter
          maybe used.  This option maybe before,  or  after,  any
          other  with  the  order  on the command line being that
          used to execute options.

     -F <s|S>
          To flush entries from the state table, the -F option is
          used  in  conjuction  with  either  "s"  (removes state
          information about  any  non-fully  established  connec-
          tions)  or  "S" (deletes the entire state table).  Only
```

one of the two options may be given. A fully established connection will show up in ipfstat -s output as 4/4, with deviations either way indicating it is not fully established any more.

-f <filename>
This option specifies which files ipf should use to get input from for modifying the packet filter rule lists.

-I Set the list to make changes to the inactive list.

-l <pass|block|nomatch>
Use of the -l flag toggles default logging of packets. Valid arguments to this option are pass, block and nomatch. When an option is set, any packet which exits filtering and matches the set category is logged. This is most useful for causing all packets which don't match any of the loaded rules to be logged.

-n This flag (no-change) prevents ipf from actually making any ioctl calls or doing anything which would alter the currently running kernel.

-o Force rules by default to be added/deleted to/from the output list, rather than the (default) input list.

-P Add rules as temporary entries in the authentication rule table.

-r Remove matching filter rules rather than add them to the internal lists

-s Swap the active filter list in use to be the "other" one.

-U (SOLARIS 2 ONLY) Block packets travelling along the data stream which aren't recognised as IP packets. They will be printed out on the console.

-v Turn verbose mode on. Displays information relating to rule processing.

-V Show version information. This will display the version information compiled into the ipf binary and retrieve it from the kernel code (if running/present). If it is present in the kernel, information about its current state will be displayed (whether logging is active, default filtering, etc).

-y Manually resync the in-kernel interface list maintained by IP Filter with the current interface status list.

-z For each rule in the input file, reset the statistics for it to zero and display the statistics prior to them being zero'd.

-Z Zero global statistics held in the kernel for filtering only (this doesn't affect fragment or state statistics).

FILES
/dev/ipauth
/dev/ipl
/dev/ipstate

SEE ALSO
ipftest(1), mkfilters(1), ipf(4), ipl(4), ipf(5), ipfstat(8), ipmon(8), ipnat(8)

DIAGNOSTICS
Needs to be run as root for the packet filtering lists to actually be affected inside the kernel.

BUGS
If you find any, please send email to me at darrenr@pobox.com