UM SISTEMA CONFIÁVEL DE DISTRIBUIÇÃO DE MÍDIA

Marcio Elkind

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Aprovada por:

_____

Prof. Jorge Lopes de Souza Leão, Dr.Ing.

_____

Prof. Luis Felipe Magalhães de Moraes, Ph.D.

_____

Prof. Luiz Pereira Caloba, Dr.Ing.

RIO DE JANEIRO, RJ - BRASIL
OUTUBRO DE 2006

ELKIND, MARCIO

Um Sistema Confiável de Distribuição de Mídia [Rio de Janeiro] 2006

XIV, 121 p. 29,7 cm, (COPPE/UFRJ, M.Sc., Engenharia Elétrica, 2006)

Dissertação - Universidade Federal do Rio de Janeiro, COPPE

1. Distribuição de Mídia

2. Avaliação Subjetiva da Qualidade de um Vídeo

3. Qualidade de Serviço

4. Tolerância a Falhas

I. COPPE/UFRJ II. Título (série)

Agradecimentos

Esta dissertação foi possível graças à ajuda de diversos fatores e pessoas, algumas das quais não mencionadas aqui.

Agradeço ao Grupo de Teleinformática e Automação da COPPE pelas condições de trabalho fornecidas, fruto da dedicação de seus professores e funcionários.

Agradeço ao aluno Danilo Michalczuk Taveira, de graduação em Engenharia Elétrica do GTA, por inestimáveis dicas sobre Linux, MatLab e infra-estrutura computacional associada.

Agradeço ao aluno André Sion Fernandes Muniz Corrêa, aluno de mestrado em Engenharia Elétrica do GTA, por suas valiosas contribuições na concepção geral do SCDM, pela sua ajuda em Linux e infra-estrutura computacional e, principalmente, por sua persistência e paciência em continuar ajudando-me, mesmo nos muitos momentos difíceis.

Agradeço ao aluno Airon Fonteles da Silva, mestrando pelo RAVEL/COPPE, e ao professor Luis Felipe Magalhães de Moraes, pela ajuda com o Emulador Paramétrico de Redes IP, desse laboratório, o qual serviu de sólida base para o posterior desenvolvimento da minha versão de um Emulador Paramétrico.

Agradeço professor Edmundo de Souza e Silva e aos mestrandos Edson Hiroshi Watanabe e Fernando Jorge Silveira Filho, todos do LAND/COPPE, por suas valiosíssimas contribuições no meu entendimento das Cadeias de Markov e demais variáveis aleatórias, essenciais aos modelos matemáticos usados nesta dissertação.

Agradeço aos professores Luis Felipe Magalhães de Moraes e Luiz Pereira Caloba pela participação na banca examinadora.

Agradeço à Marinha do Brasil, e em especial ao Capitão-de-Fragata João Alberto Vianna Tavares, pela oportunidade de realização deste trabalho, confiança e orientações recebidas durante o longo percurso até aqui.

Agradeço ao meu orientador e amigo, professor Jorge Lopes de Souza Leão, pelos valiosos ensinamentos, sem os quais muito pouco teria sido realizado.

Agradeço aos meus pais, Raul e Hilda Elkind, pela vida que me deram, pela educação, valores morais e apoio recebidos.

Agradeço, por fim, a Deus. Sem sua Divina graça, nenhum dos agradecimentos anteriores poderia ter sido feito.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

UM SISTEMA CONFIÁVEL DE DISTRIBUIÇÃO DE MÍDIA

Marcio Elkind

Outubro/2006

Orientador: Jorge Lopes de Souza Leão

Programa: Engenharia Elétrica

Um sistema confiável de distribuição de mídia necessita ser tolerante a falhas e tratar de forma convergente a distribuição de: vídeo de tempo real, mensagens imediatas, dados comuns e voz. Nesta dissertação apresenta-se a concepção de um sistema de gerência distribuída e pró-ativa, que controla a admissão de novos fluxos e a QoS fim-a-fim de aplicações, segundo uma regra preemptiva de prioridades. Ocorrendo falhas na distribuição, a gerência do sistema procura degradar graciosamente a qualidade dos fluxos de mídia sendo distribuídos. Os sistemas de distribuição de vídeo e de gerência são construídos sobre uma infra-estrutura que utiliza componentes disponíveis comercialmente.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

A REALIBLE MEDIA DISTRIBUTION SYSTEM

Marcio Elkind

October/2006

Advisor: Jorge Lopes de Souza Leão

Department: Electrical Engineering

A reliable media distribution system must be fault tolerant and manages the integration of streams of real time media, immediate messages, common data and voice. A distributed and proactive management system that controls the admission and the end-to-end QoS of applications, based on a preemptive priority scheme is presented. As failures happen in the distribution system, the management system tries to gracefully degrade the quality of the videos being distributed. The distribution and management systems are supported by an infrastructure based on commercially available components.

Sumário

Índice de Figuras

Índice de Tabelas

Tabela de Siglas e Abreviaturas

ANFIS	*Adaptative Neuro-Fuzzy Inference System*
CIF	*Common Intermediate Format*
CMIP	*Common Management Information Protocol*
CONSIST	Consoles de Sistema
DC	Classe para os Dados Comuns
DiffServ	*Differentiated Services*
DSCP	*Differentiated Services Code Point*
DVMRP	*Distance Vector Multicast Routing Protocol*
GD	Gerente de Desempenho
GOB	*Group of Block*
HSRP	*Hot Standby Router Protocol*
ID	Inspetor de Desempenho
IGMP	*Internet Group Management Protocol*
IntServ	*Integrated Services*
IPSTB	*IP Standby Protocol*
JMF	*Java Media Framework*
LAN	*Local Area Network*
MB	*Macroblock*
MI	Classe para as Mensagens Imediatas
MJPEG	*Motion JPEG vídeo*
MOSPF	*Multicast Open Shortest Path First*
OSPF	*Open Shortest Path First*
PEL	*Picture Element*
PIM-DM	*Protocol Independent Multicast - Dense Mode*
PIM-SSM	*Protocol Independent Multicast - Source Specific Multicast*
QCIF	*Quarter CIF*
QoS	Qualidade de Serviço
RIP	*Routing Information Protocol*
RTCP	*Real Time Control Protocol*
RTP	*Real Time Protocol*
SNMP	*Simple Network Management Protocol*
UDP	*User Datagram Protocol*
VPN	*Virtual Private Network*
VoIP	Voz sobre IP
VRRP	*Virtual Router Redundancy Protocol*
VTR	Classe para os Vídeos de Tempo Real
WAN	*Wide Area Network*
XRN	*eXpandable Resilient Networking*

Capítulo 1 Introdução

Os sistemas multimídia são sistemas computacionais que manipulam de forma integrada vários tipos de mídias. Estas se dividem em mídia discreta, como imagens e textos, e contínua, como vídeos e áudios. Esses sistemas, freqüentemente são distribuídos, isto é, seus componentes estão localizados em diferentes nós em uma rede local ou de longa distância.

Como pode ser visto em (GOMES, 2001), as aplicações multimídia apresentam três características:

(i) exigem maior banda de transmissão, comparadas às aplicações com textos;

(ii) tráfegos em tempo real (áudio e vídeo) que devem estar disponíveis nos receptores para apresentação na mesma taxa com que foram gerados;

(iii) a geração dos fluxos multimídia apresenta inerentemente rajadas, devido à compressão utilizada; e

(iv) devido aos três itens anteriores e às características probabilísticas da rede, os parâmetros de QoS podem ser violados, tais como o atraso entre o transmissor e o receptor, a variação randômica deste atraso, a perda de pacotes e a chegada dos pacotes em ordem diferente da qual foram transmitidos. Como conseqüência, poderá ocorrer escassez de recursos durante a apresentação destas mídias, ocasionando a danificação ou até mesmo a interrupção dessas.

Portanto, uma questão muito importante a ser considerada para a transmissão de multimídia é a qualidade de serviço (QoS), tornando-se uma exigência cada vez mais freqüente nas redes de computadores que suportam este tipo de tráfego. A garantia de níveis de qualidade de serviço específicos depende fundamentalmente da presença de um mecanismo de gerenciamento preciso, eficiente e rápido, não só dos recursos da rede como também dos serviços de comunicação (fim a fim), das estações e das aplicações.

Esse mecanismo deve ser de fácil atualização e flexível o suficiente para a implantação de novas funcionalidades com a devida rapidez além de, preferencialmente, ser independente de plataforma. CECÍLIO *et al.* (2003) menciona que outra característica desejável é que seja pró-ativo, de forma a permitir que sejam executadas ações com

o objetivo de se tentar reverter quedas nos níveis de QoS antes da ocorrência de falhas ou, caso isso seja impossível, minimizar os prejuízos para os usuários. O gerenciamento de desempenho, em particular, é de vital importância, pois é quem deve monitorar a situação da rede no que diz respeito à qualidade dos serviços que estão sendo prestados.

1.1 Motivação

A integração de voz, vídeo, imagem e dados, cada um com seus próprios requisitos de qualidade de serviço, exige o desenvolvimento de aplicações cada vez mais complexas e que precisam funcionar satisfatoriamente sobre as atuais redes de dados. Estas, por sua vez, estão organizadas sobre uma arquitetura que não garante QoS (Qualidade de Serviço). Aplicações tais como VoIP (Voz sobre IP), videoconferência e comércio eletrônico, precisam prover serviços em tempo real sobre esta arquitetura.

A dualidade acima ocorre porque o protocolo IP foi idealizado com o objetivo de ser robusto, não sendo, portanto, capaz de atender às demandas advindas desse novo cenário de forma simples. O paradigma de entrega apenas por melhor esforço (*best-effort delivery*) não consegue atender às severas restrições de QoS porque, no protocolo IP, nada é garantido quanto ao tempo ou à certeza de entrega. Outra questão é a falta de garantia na ordem de chegada de pacotes de um mesmo fluxo, uma vez que caminhos distintos podem ser usados entre origem e destino.

A infra-estrutura de rede, que fornecerá níveis garantidos de serviços às novas aplicações, deverá não somente dispor de tecnologias de hardware mais complexas mas também de arquiteturas de *software* com mecanismos que deverão atuar desde o nível de aplicação (gerenciamento de rede) até a camada de enlace (mecanismos de escalonamento de quadros com prioridade).

1.2 Objetivos

Este trabalho faz parte de um projeto de pesquisa, da Marinha do Brasil, na área de redes de computadores, que se insere no contexto da concepção de um sistema que trate de forma convergente a distribuição de vídeo de tempo real (e.g. radar), mensagens imediatas (e.g. designação de alvos) e a transmissão de dados comuns.

O Sistema Confiável de Distribuição de Mídia (SCDM) tem como objetivo a distribuição de fluxos de vídeo e de dados segundo o paradigma cliente-servidor. Ele procura manter a QoS dos fluxos de vídeo, atendendo a uma política de prioridades definida para estes e para os clientes. Além disto, ele deve monitorar e reagir às ações dos mecanismos de tolerância a falhas existentes na infra-estrutura. Esses dois objetivos são inter-relacionados e complementares.

O controle da infra-estrutura, a atribuição de prioridades a fluxos e usuários, a admissão de novos fluxos de forma preemptiva, segundo uma política bem definida, a (re)alocação de recursos para os fluxos e a tomada de ações em resposta às degradações na rede de infra-estrutura, são de responsabilidade de uma gerência automática, calcada no paradigma peer-to-peer, que se localiza distribuída nos hosts do sistema.

O fator crucial do SCDM é a sua determinação em permanecer sustentando a prestação de serviços aos diferentes usuários, mesmo que degradações diversas venham a ocorrer no navio. Esta sustentação dos serviços será também de forma preemptiva, isto é, fluxos e/ou clientes com mais altas importâncias, terão maiores prioridades. O SCDM sacrificará recursos, fluxos e usuários de menor prioridade em prol de, eventualmente, poucos recursos que serão alocados aos clientes e fluxos prioritários.

Neste trabalho, é implementada uma pequena parte do SCDM: o Inspetor Neuro-Fuzzy de Desempenho. Utilizamos a linguagem JAVA e o pacote JMF (Java Media Framework) como ferramentas de trabalho. Após a avaliação de alguns esquemas de compressão de vídeo, decidimos usar o *codec* H-263, implementado pelo JMF. Adotamos modelos matemáticos para modelar o comportamento de redes IP e os implementamos em um Emulador Paramétrico. Com o Emulador e o codec H263 do JMF, conseguimos definir os limites para *jitter* e perdas, para esta aplicação. Finalmente, com o codec, o Emulador e valores de jitter e perda, programamos uma rede neuro-fuzzy, que acoplada ao código do Inspetor de Desempenho, permite a avaliação on-line da qualidade do vídeo no cliente, dadas as condições presentes da rede.

1.3 Organização

Esta dissertação está organizada em cinco capítulos.

O Capítulo 2 provê uma visão geral dos principais mecanismos necessários à compreensão do Sistema Confiável de Distribuição de Mídia (SCDM). Os conceitos de

Qualidade de Serviço (QoS) e seus principais parâmetros são relatados. A classificação dos tipos de aplicações multimídia, a Qualidade de Percepção (QoP) e a avaliação pelo usuário final são abordadas a seguir. Outros tópicos importantes são: os protocolos RTP/RTCP, a arquitetura DiffServ e o gerenciamento de redes.

O Capítulo 3 apresenta a concepção do SCDM: um sistema de gerência distribuída e pró-ativa para a distribuição de mídia de tempo real, que controla a admissão de novos fluxos e a QoS fim-a-fim de aplicações, segundo uma política preemptiva de prioridades. Ocorrendo falhas na distribuição, a gerência procura degradar graciosamente a qualidade dos fluxos de mídia sendo distribuídos. Os sistemas de distribuição de mídia e de gerência são construídos sobre uma infra-estrutura que utiliza componentes disponíveis comercialmente.

O Capítulo 4 apresenta diversas decisões de implementação do SCDM. A plataforma de hardware utilizada será o computador pessoal tipo PC, o protocolo para transmissão de vídeo adotado será o RTP/RTCP, enquanto que a plataforma de software será a Java/JMF e é descrita em 4.1. Em função destas decisões escolheu-se o *codec* como H.263 (vide 4.2). A topologia de implementação está descrita em 4.3, o Inspetor de Desempenho em 4.4, os modelos matemáticos adotados são descritos em 4.5, enquanto que o Emulador Paramétrico que utiliza estes modelos está descrito em 4.6.

O Capítulo 5 apresenta os resultados relativos ao escopo de implementação definido no Capítulo 4. O Emulador paramétrico necessitou ser validado como ferramenta e o Classificador Neuro-Fuzzy, presente no Inspetor de Desempenho, foi treinado segundo as condições de rede definidas para os modelos matemáticos adotados. A presente dissertação é concluída, no Capítulo 6, indicando-se melhorias e continuidade do trabalho para a completa implementação do SCDM.

Capítulo 2 Aspectos Preliminares

2.1 Qualidade de Serviço (QoS)

O conjunto de exigências (requisitos) para o atendimento de serviços a serem fornecidos pela rede durante o encaminhamento de um fluxo de pacotes pode ser definido com sendo a Qualidade de Serviço (QoS). Esta pode também ser considerada como o conjunto de requisitos que garanta o desempenho mínimo da rede para as aplicações dos usuários.

O entendimento dos fatores que afetam a QoS de uma aplicação é, então, de fundamental importância para o SCDM.

2.1.1 Parâmetros de QoS

Do ponto de vista do sistema computacional, FERGUSON (1998) descreve os seguintes parâmetros de QoS:

"*Atraso* é o tempo decorrido para que um pacote seja passado do emissor, através da rede, até o receptor". O atraso pode ser causado por diversos fatores tais como:

- enfileiramento (propagação nas filas de transmissão nos nós intermediários da rede);
- propagação (tempo de trânsito entre origem e destino); e
- transmissão (tempo necessário à serialização do pacotes).

"*Jitter* é a variação no atraso fim-a-fim. Altos níveis de *jitter* são inaceitáveis em situações onde a aplicação é baseada em tempo real (áudio e vídeo). Nestes casos, o *jitter* torna o sinal distorcido, podendo ser recuperado apenas com um aumento no buffer do receptor, o que afeta o atraso, tornando as sessões interativas muito difíceis de serem mantidas". Para considerações adicionais vide seção 4.5.2.

"*Largura de banda* é a taxa de transferência máxima que pode ser sustentada entre dois nós da rede. Constata-se que a largura de banda é limitada não apenas pela infra-

estrutura física da rota dentro das redes de trânsito (o que fornece um limite superior para a banda), mas também pelo número de outros fluxos que compartilham os componentes comuns da rota selecionada".

"*Confiabilidade* é comumente conceituada como uma propriedade do sistema de transmissão, e neste contexto, pode ser vista como a taxa média de erro do meio de transmissão". Usaremos o termo taxa de perdas para referir-nos à confiabilidade.

2.2 Aplicações Multimídia

As aplicações multimídia (também referenciadas como aplicações de mídia contínua) – fluxo contínuo de vídeo, telefonia IP, rádio pela Internet, teleconferência, jogos interativos, mundos virtuais e outras – são anunciadas diariamente. Os requisitos de serviço destas diferem significativamente dos requisitos das aplicações "elásticas" – e-mail, Web, login remoto, compartilhamento de arquivos - onde o mais importante é a confiabilidade dos dados e não a temporização destes.

2.2.1 Classes de Aplicações Multimídia

KUROSE (2003) considera três grandes classes de aplicações multimídia: vídeo/áudio pré-armazenado, vídeo/áudio ao vivo e vídeo/áudio interativo.

a) Vídeo/áudio pré-armazenado – nesta classe de aplicações, os clientes solicitam arquivos de áudio e/ou vídeo que estão pré-armazenados em servidores. Esta classe de aplicações tem três características chave:

- Mídia armazenada – como o conteúdo multimídia foi pré-gravado e está armazenado no servidor, o usuário pode pausar, voltar, acelerar a mídia ou até mesmo usar um índice dentro do conteúdo da mídia. O tempo de resposta entre uma solicitação do cliente até a sua manifestação é da ordem de um a dez segundos para uma reação aceitável.

- Continuidade de fluxo – a mídia começa a ser exibida em um cliente alguns segundos após o arquivo ter sido recebido no receptor. Isto significa que a mídia é tocada no cliente enquanto este recebe demais partes dela, provenientes do servidor.
- Exibição contínua – uma vez iniciada a exibição, esta deve manter a temporização da mídia originalmente gravada no servidor, impondo requisitos críticos de tempo na recepção dos dados. Estes devem chegar a tempo de serem exibidos. Apesar destes requisitos serem críticos, para as outras duas classes têm-se limites mais justos ainda.

b) Vídeo/áudio ao vivo – esta classe é similar às transmissões públicas de TV e rádio, exceto que o veículo é uma rede. Como a mídia é ao vivo, não sendo pré-armazenada, um cliente não pode avançá-la. Contudo, algumas aplicações permitem pausar e retroceder a mídia. Retardos na exibição da mídia desde a solicitação do cliente, da ordem de dezenas de segundos, podem ser tolerados.

c) Vídeo/áudio interativo – esta classe permite às pessoas se intercomunicarem em tempo real. Uma aplicação tipo Telefonia IP, do ponto de vista do usuário, tem as mesmas características de um PBX ou telefonia convencional. Para voz, retardos menores que 150 milisegundos não são percebidos pelo ouvido humano, retardos entre 150 e 400 ms podem ser aceitáveis enquanto que os maiores que 400ms resultam em conversações frustrantes ou completamente inteligíveis.

A tabela 2.1 exemplifica algumas aplicações multimídia com a sensibilidade aos parâmetros de Qualidade de Serviço (CISCO, 2005b).

Aplicações		Sensibilidade às métricas de QoS		
Tipo	Exemplos	Retardo	Jitter	Perda de Pacotes
Vídeo e voz interativos	Vídeo conferência, VoIP	S	S	S
Vídeo Contínuo	Vídeo sob demanda, filmes	N	S	S
Transações / Interativo	Processamento de ordens, pagamentos, inventórios, relatórios.	S	N	N
Dados em massa	Email, dados de backups, arquivos de impressão	N	N	N

Tabela 2-1 - Aplicações Multimídia e sensibilidades às métricas de QoS

2.3 Qualidade de Percepção

Embora os parâmetros técnicos de QoS sejam úteis, eles possuem pouco significado para o usuário final. Considerando a perspectiva do usuário, a qualidade de serviço oferecida durante uma apresentação está associada à possibilidade de assimilar os conteúdos informacionais da mesma. Muitas vezes, a quantidade de recursos alocada a uma aplicação multimídia pode não interferir na percepção por parte do usuário (STEINMETZ, 1996).

GHINEA (1999) define Qualidade de Percepção (*Quality of Perception* – QoP) para representar a QoS do ponto de vista do usuário. Segundo Ghinea, QoP é um termo que abrange tanto a satisfação com relação à qualidade da apresentação, quanto à habilidade de analisar, sintetizar e assimilar o conteúdo informacional das mídias exibidas.

O conceito de QoP permite a associação entre medidas do estado atual da rede – a QoS e a avaliação, pelos usuários, da qualidade da mídia. Esta avaliação, pelos usuários, ocorre de forma subjetiva. O método mais comum é o *Mean Opinion Score* (MOS).

2.4 Avaliação Subjetiva da Qualidade e MOS

Como visto no item 2.2, para cada tipo de aplicação multimídia que o SCDM deve tratar, diferentes requisitos de QoS devem ser satisfeitos. Somando-se a isto, o fator importante para o usuário é a QoP. Esta deve ser avaliada, em última análise, pelo usuário final. Um método de avaliação será abordado neste item.

Existem diversos parâmetros que afetam a qualidade de transmissão de um vídeo através de uma rede de pacotes. MOHAMED (2002) classifica-os da seguinte forma:

- parâmetros de codificação e compressão: controlam as de perdas de qualidade que ocorrem durante o processo de codificação da mídia, portanto dependem do algoritmo de CODEC utilizado (MPEG, H26x, etc.), da taxa de bits de saída, da taxa de *frames* (o número de *frames* por segundo), da relação temporal entre os tipos de *frames*, etc;

- parâmetros de rede: são o resultado do processo de empacotamento do fluxo de mídia e a transmissão destes através da rede – taxa de perdas de pacotes, a distribuição (estatística) das perdas, variação do retardo (*jitter*), etc;
- outros parâmetros, como por exemplo, a natureza da cena (i.e. quantidade de movimento, cor, contraste, tamanho da imagem, etc) também podem afetar a percepção humana do vídeo.

A análise da qualidade de um vídeo pode ser feita usando testes objetivos ou subjetivos. Os testes objetivos são sempre funções diretas dos parâmetros mensuráveis relacionados ao codificador ou à rede (OLSSON, 1997).

Os testes subjetivos baseiam-se nas avaliações feitas por seres humanos, sob condições bem definidas e controladas. Obviamente, a referência é a percepção do usuário final, que é capturada diretamente pelos testes subjetivos. Sabe-se que os testes objetivos disponíveis, não se correlacionam bem com a percepção humana (MOHAMED, 2002).

Na literatura existem diversos métodos para a avaliação da qualidade de um vídeo, porém o mais geralmente usado é a *Mean Opinion Score* (MOS), recomendado pelo ITU (ITU-R BT.500-10, 2000) e (ITU-T P.910, 1999). Neste método, cada uma das pessoas em um grupo de avaliadores, assiste a várias apresentações de um mesmo vídeo. Cada apresentação consiste de uma situação (p.ex. das condições de rede - *jitter* e perda de pacotes) que degrada o vídeo original. Logo, cada avaliador assiste a diversas amostras distorcidas do mesmo vídeo, emitindo seu julgamento para cada amostra. Ao final, computa-se a média das opiniões para cada situação que gerou a degradação sendo avaliada.

2.5 Mecanismos para comunicação multimídia na Internet

A Internet transporta todo tipo de tráfego, cada tipo tendo suas características e requisitos específicos. A solução genérica para a multimídia sobre IP envolve a classificação dos tráfegos, a alocação de prioridades para diferentes aplicações e a implementação da reserva de recursos.

Dois modelos de serviços avançados para a Internet, criados pelo IETF, e denominados Serviços Integrados (BRADEN, R et al, 1994) e Serviços Diferenciados

(BLAKE, S. et al., 1998) buscam habilitar as redes baseadas em IP a prover qualidade de serviço para as aplicações multimídia. O protocolo de reserva de recursos (RSVP) (BRADEN, R. et al, 1997) em conjunto com o RTP (*Real-time Transport Protocol*) (SCHULZRINNE, H. et al, 2003), o RTCP (*RTP Control Protocol*) (idem RTP) e o RTSP (*Real-Time Streaming Protocol*) (SCHULZRINNE, H. et al, 1998), fornecem a base fundamental para os serviços de tempo real solicitados pela maioria das aplicações multimídia. Os modelos de serviços do IETF permitem às aplicações configurar e gerenciar uma infra-estrutura comum tanto às aplicações multimídia quanto às aplicações tradicionais.

Dentre os mecanismos acima mencionados, serão usados no SCDM os seguintes: *DiffServ*, RTP, RTCP e RTSP.

2.5.1 Protocolos de Transporte de Mídias de tempo real

O IETF definiu através da RFC1889, atualizada pela RFC3550 (SCHULZRINNE, H. et al., 2003), como padrão para o transporte de áudio e vídeo em redes IP o protocolo RTP (Real-time Transport Protocol) consistindo de duas partes inter-relacionadas:

- o RTP para transportar dados com propriedades de tempo real; e.
- o protocolo do controle do RTP (RTCP), para monitorar a qualidade de serviço e para coletar informações sobre os participantes de uma sessão em curso.

A maioria das implementações do RTP faz parte de uma aplicação ou biblioteca, localizando-o acima dos soquetes do UDP fornecidos pelo sistema operacional. Esta não é a única implementação possível: algumas situam o RTP acima do TCP e outras usam RTP em redes não-IP, tais como o Asynchronous Transfer Mode (ATM), conforme a figura 2.1.

O RTP fornece serviços úteis para o transporte de mídias em tempo real, tais como o áudio e o vídeo que trafegam sobre redes IP. Estes serviços incluem a recuperação de temporização, a detecção e a correção de perdas, a identificação da carga e da fonte, informações sobre a qualidade da recepção, a sincronização de mídias, e a gerência de participantes da sessão.

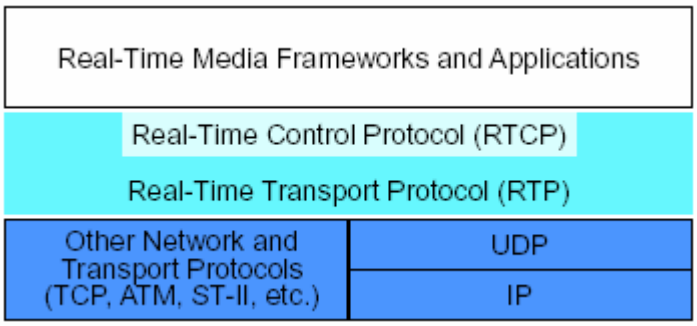


Figura 2-1 - Contexto do RTP/RTCP

O RTP foi projetado originalmente para o uso em conferências multicast. Desde esse tempo, provou ser útil para outras aplicações: em conferencias de vídeo H.323, em distribuição de TV e em telefonia cabeada e celular. O protocolo foi demonstrado ser escalável desde uso ponto-a-ponto a sessões multicast com milhares dos usuários, e das aplicações celulares com baixa largura de banda à entrega de sinais de alta definição de TV (HDTV) em taxas gigabit.

O anexo A detalha funcionalidades dos protocolos UDP, RTP e RTCP.

2.5.2 A Arquitetura DiffServ

O estudo da arquitetura DiffServ (BLAKE et al, 1998) é importante para o SCDM porque o interior de sua infra-estrutura de suporte (a ser detalhada em capítulo posterior) é calcado no *DiffServ*. Este se baseia no tratamento diferenciado a cada classe de tráfego, para garantir a QoS fim-a-fim desta (e não de um fluxo individual). Uma porção do tráfego é tratada de forma privilegiada sobre o restante. É oferecida manipulação mais rápida, mais largura de banda na média e menor taxa de perda. Esta é uma preferência estatística, não uma garantia rígida. Com métodos adequados, incluindo a utilização de determinadas políticas nas extremidades da rede (domínio *DiffServ*), esta arquitetura pode prover um tratamento adequado para uma boa gama de aplicações, incluindo aquelas de missão crítica, as que necessitam de baixo atraso, aplicações de voz sobre pacotes, e outras.

A figura 2.2 mostra onde os componente-chave de uma rede *DiffServ* são empregados (MELO, 2001). *DiffServ* define o conceito de domínio DS: um conjunto contíguo de nós DS que aplicam um conjunto comum de políticas sobre o tráfego que atravessa o domínio. Um domínio DS tem nós de borda e nós de interior. Os nós DS de

borda são responsáveis pela classificação e condicionamento do tráfego que entra no domínio DS.

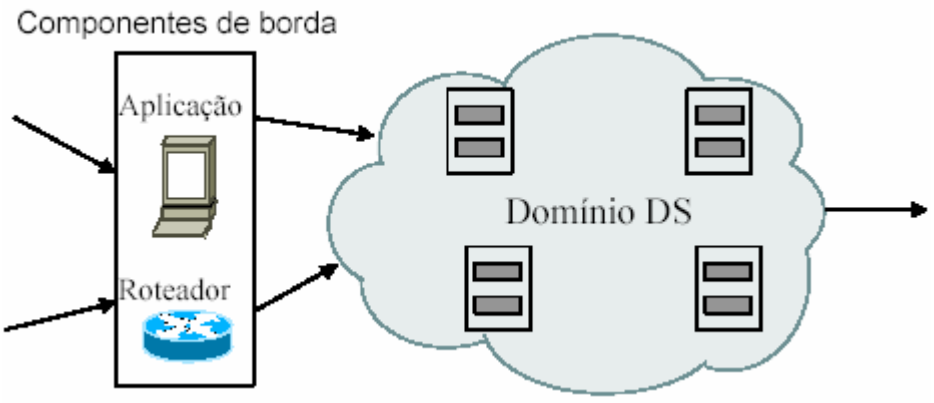


Figura 2-2 - Serviços diferenciados - visão geral

Este tratamento é repetido nó a nó, ou seja, os pacotes de uma aplicação prioritária quando chegam a um nó (roteador) são separados e recebem um tratamento diferenciado. Este comportamento chama-se PHB (Per-hop Behavior). O *DiffServ* não necessita controle de admissão, sinalização de QoS e nem reserva de banda fim-a-fim, o que aumentaria a carga da rede com protocolos de controle.

Nas bordas da rede (domínio *DiffServ*), os bits de um campo do cabeçalho IP, renomeado pela RFC2475 como campo DS e mantendo compatibilidade ascendente com o campo ToS do IPv4, são marcados para que possuam um valor específico. No interior da rede, esses mesmos bits servirão de parâmetro para determinar o tipo de tratamento que será dado a cada pacote. A estrutura do campo DS é mostrada na figura 2.3 abaixo.

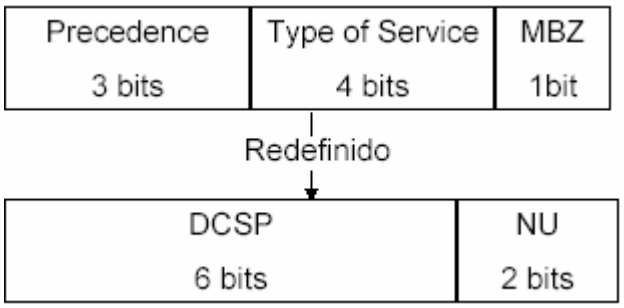


Figura 2-3 - O campo IP TOS versus DCSP

2.5.2.1 Classes DiffServ

Os seis bits do campo DS são usados para selecionar o PHB que o pacote terá em cada nó. Estes comportamentos nó-a-nó definem as classes DiffServ. Note-se que os PHBs de tráfegos com DSCP mais alto devem ter encaminhamento preferencial sobre aqueles com valor mais baixo. Por exemplo, o PHB 111000 deve ter prioridade mais alta que 000000.

PHB Padrão (PHB-BE)

O PHB Padrão é um comportamento correspondente ao melhor esforço. Esse PHB é utilizado quando um pacote sem nenhuma marcação especial no seu codepoint ingressa em um domínio DS. Assim, usuários que não contrataram, nenhum serviço especial podem continuar se utilizando da infra-estrutura da rede. O codepoint padrão recomendado para esta situação é 000000.

Encaminhamento Expresso (PHB-EF)

O PHB EF, também referido como serviço premium ou de canal dedicado, pode ser usado para tráfego com requisitos de baixa perda, baixo atraso, baixo *jitter* (variação de atraso) e garantia de largura de banda. Estes requisitos são alcançados assegurando-se que os agregados de tráfego encontram nenhum ou pouco enfileiramento. Para isso, ele deve garantir que a taxa de serviço contratada seja maior do que a taxa de chegada em todo o instante. Essa garantia deve ser tal que independa de outros fluxos que cheguem ao nó. Para isso, os roteadores de entrada em um domínio DS devem policiar o ingresso de pacotes para que atenda ao contratado, evitando assim o uso abusivo da rede. A implementação de um PHB-EF pode ser realizada usando um mecanismo de escalonamento de filas, como o mecanismo de fila prioritária. Em testes de simuladores chegou-se a conclusão que para o sistema funcionar de forma estável, é necessário que se aloque uma banda aproximadamente 6% superior a banda efetivamente contratada.

Encaminhamento Assegurado (PHB-AF)

O PHB AF, ao invés de fornecer uma garantia estrita como no modelo anterior, fornece apenas uma expectativa de serviço quando a rede passar por momentos de congestionamento.

O PHB AF tem por objetivo fornecer entrega de pacotes IP, com largura de banda assegurada, em quatro classes de transmissão, mas não oferece garantias quanto ao atraso. A garantia dada é que os pacotes marcados como de alta prioridade, têm uma grande probabilidade de serem entregues com êxito, desde que o sistema esteja sendo usado dentro do valor combinado. O usuário pode exceder a taxa de transmissão, mas fica ciente que tais pacotes não terão essa mesma alta probabilidade de sucesso na entrega.

Cada classe tem três precedências de descartes (Drop Precedence), as quais são utilizadas para determinar a importância do pacote. Assim, um nó congestionado dá preferência para serem descartados, entre os pacotes de uma mesma classe, àqueles com valores maiores de precedência de descarte. Os três primeiros bits do DSCP identificam a classe de transmissão, 001, 010, 011 e 100 e os três últimos bits definem a precedência de descarte 010 para a precedência mais baixa (ou seja, primeiro a ser descartado), 100 para precedência média e 110 para a mais alta precedência de descarte (ou seja, último a ser descartado).

Nas Tabela 2-2, Tabela 2-3 e Tabela 2-4 abaixo, são apresentados os DSCP recomendados para as classes de tráfego BE, EF, e EF, respectivamente (MELO,2001).

Precedência de descarte	Classe BE	
	Binário	Decimal
N/A	000000	0

Tabela 2-2 - PHB- Melhor

Precedência de descarte	Classe EF	
	Binário	Decimal
N/A	101110	46

Tabela 2-3 - PHB – Encaminhamento Expresso

Precedência de descarte	Classe AF1		Classe AF2		Classe AF3		Classe AF4	
	Binário	Decimal	Binário	Decimal	Binário	Decimal	Binário	Decimal
Baixa	001010	10	010010	18	011010	26	100010	34
Média	001100	12	010100	20	011100	28	100100	36
Alta	001110	14	010110	22	011110	30	100110	38

Tabela 2-4 - PHB AF, com quatro classes de tráfego independentes

2.6 Gerência de Redes

O estudo de técnicas de gerenciamento de redes é importante para o SCDM porque o controle de sua infra-estrutura de suporte, a detecção de falhas e o controle da taxa de compressão do(s) vídeo(s) no(s) servidor(es) será implementado na Gerência.

A gerência de redes, conforme padronizada em (YEMINI, 1993), é divida em cinco áreas funcionais: configuração, segurança, desempenho, falha e contabilidade.

Particularmente importante para o SCDM, o gerenciamento de desempenho é responsável por monitorar o estado da rede e os parâmetros de QoS dos fluxos das aplicações que possuem níveis de serviços assegurados. Em função dos resultados desse monitoramento, o gerenciamento deve desencadear ações nos diversos mecanismos existentes nos nós e estações da rede. Essas ações visam evitar ou minimizar os efeitos das falhas de QoS nos serviços que estão sendo prestados.

Porém, a Gerência (que no caso do SCDM será distribuída) possui outras atribuições: RAJKUMAR et al (1997), por exemplo, apresentam um modelo analítico para a gerência de QoS cujo objetivo é alocar recursos às várias aplicações, de forma tal que a utilidade global do sistema é maximizada, sob a condição de que cada aplicação possa ser atendida em suas mínimas necessidades.

2.6.1 Gerenciamento de redes centralizado versus distribuído

A abordagem clássica de gerenciamento – centralizada – que é utilizada, por exemplo, nas implementações iniciais das arquiteturas SNMP e CMIP (SLUMAN, 1989) é baseada em um paradigma cliente-servidor, onde este último, instalado em um elemento da rede, na forma de um agente, sofre pollings freqüentes para que envie dados a uma estação de gerenciamento, que nesse caso faz o papel de cliente. Esta estação central constrói então uma visão do estado da rede e gera alarmes quando os problemas são detectados.

Segundo (CECÍLIO, 2002), essa abordagem gera um grande tráfego na rede, que não só pode prejudicar o desempenho da rede como também pode fazer com que o tempo decorrido entre a detecção de uma situação anormal e a intervenção apropriada para resolvê-la seja muito grande, podendo torná-la até mesmo inútil. À medida que o número de elementos gerenciados cresce, a complexidade, os requisitos de poder de com-

putação da estação de gerenciamento e de consumo de banda da rede gerenciada também aumentam.

Com o objetivo de superar essas limitações da abordagem centralizada, vários esquemas de gerenciamento distribuídos vêm sendo propostos. A idéia é, ao invés de realizar a coleta das informações para posterior tratamento em um ponto central, levar o processamento para o mais próximo possível dos elementos a serem gerenciados. A hierarquização do gerenciamento, dividindo a rede em domínios com cada domínio possuindo uma estação de gerenciamento foi uma das primeiras tentativas.

Em (WADDINGTON, 1998), o gerenciamento de QoS é apresentado como um mecanismo adaptativo para a configuração e o controle dos recursos fim-a-fim. Esta adaptação é implementada através de um balanço e uma redistribuição dos recursos locais e de rede, a partir de notificações emitidas por mecanismos de monitoramento de QoS.

2.6.2 Gerenciamento Pró-Ativo

O gerenciamento de desempenho, ao constatar o desrespeito aos limites de QoS acordados, deve localizar os dispositivos responsáveis e desencadear ações com o intuito de reverter essa situação (CORREIA, 2003). Esta forma de atuação é dita reativa porque as ações são realizadas após a ocorrência da falha de QoS. Este tipo de abordagem tem a desvantagem de não ser transparente, ou seja, a qualidade da apresentação sofre degradações que, em geral, são perceptíveis aos usuários. Logo, é recomendável que o gerenciamento de desempenho desencadeie ações de forma pró-ativa. Nesse caso, ao serem detectadas tendências de ocorrência de falhas de QoS, o gerenciamento de desempenho deve atuar antecipadamente, tentando minimizar ou até mesmo eliminar os efeitos das falhas nas aplicações ou, devido ao desencadeamento dessas ações, evitar que as falhas ocorram. Exemplos dessas ações são: o disparo de adaptações em aplicações multimídia adaptativas, conforme previsto em (GOMES, 2001) e (CUNHA, 2000), a mudança de políticas de escalonamento de pacotes e as solicitações de rerroteamento de fluxos aos mecanismos da camada de rede.

Capítulo 3 Proposta para o SCDM

Neste capítulo é apresentada a concepção do SCDM: um sistema de gerência distribuída e pró-ativa para a distribuição de mídia de tempo real, que controla a admissão de novos fluxos e a QoS fim-a-fim de aplicações, segundo uma política preemptiva de prioridades. Ocorrendo falhas na distribuição, a gerência procura degradar graciosamente a qualidade dos fluxos de mídia sendo distribuídos. Os sistemas de distribuição de mídia e de gerência são construídos sobre uma infra-estrutura que utiliza componentes disponíveis comercialmente.

3.1 Descrição do SCDM

O SCDM (Sistema Confiável de Distribuição de Mídia), disposto na figura 3.1, tem como missão a distribuição de fluxos de vídeo e de dados (com previsão para VoIP e VPN), segundo o paradigma cliente-servidor, de forma confiável e controlada.

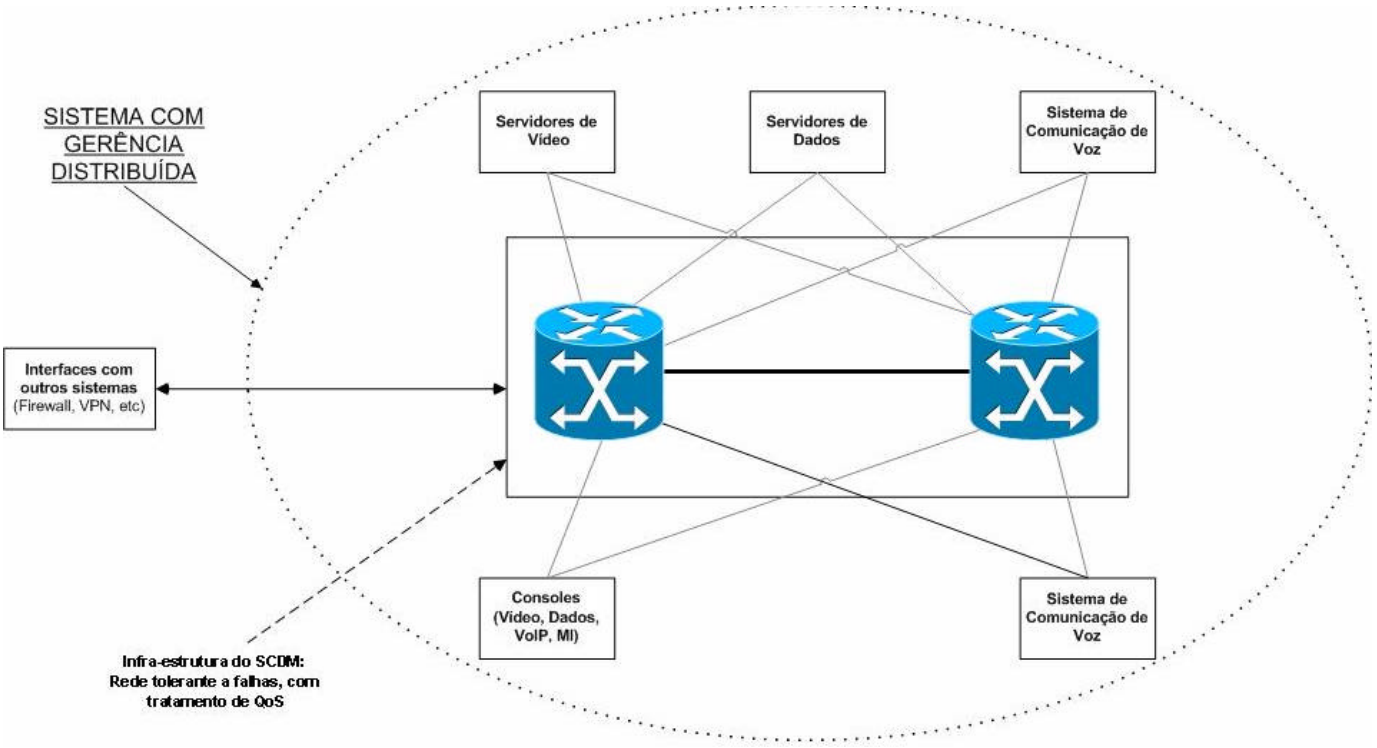


Figura 3-1 - O SCDM - Sistema Confiável de Distribuição de Mídia

Sob uma perspectiva macroscópica, o SCDM constitui-se de duas partes:

- uma infra-estrutura, constituindo o núcleo físico do sistema, que possui mecanismos os quais oferecem diversos serviços e permitem receber controles externos;
- uma entidade distribuída que gerencia todo o SCDM, colhendo informações e atuando pró-ativamente na infra-estrutura, sempre com uma visão panorâmica do que ocorre nos seus diversos componentes.

A infra-estrutura do SCDM é constituída por uma rede IP capaz de suportar os seus requisitos (descritos abaixo em 3.1.2). O sistema fará solicitações para a infra-estrutura e aproveitar-se-á dos serviços oferecidos por esta para cumprir sua missão. Esta arquitetura está descrita no item 3.2.

A Gerência Distribuída do SCDM, calcada no paradigma "peer-to-peer", deverá implementar dois objetivos, inter-relacionados e complementares:

- manter a QoS fim-a-fim dos fluxos de vídeo, atendendo a uma política de prioridades definida para os fluxos e para os clientes, e
- monitorar e reagir às ações dos mecanismos de tolerância a falhas existentes na infra-estrutura, aliada a uma política de segurança.

A descrição dos critérios da gerência será feita no item 3.3.

3.1.1 Descrição dos Componentes do SCDM

O sistema possuirá os seguintes componentes principais:

a. "N" (um certo número de) servidores independentes de vídeo

b. Os servidores de dados serão compostos por:

- sensores espalhados pelo navio
- "M" consoles de sistema (CONSIST), para apresentação de vídeos radar, recepção de dados de outros consoles e sensores, através da rede, assim como envio de dados a outros CONSIST s.

c. Um sistema de distribuição de voz, usando a tecnologia VoIP, o que implicará em aumento de carga na rede, com modelo de tráfego já conhecido.

d Com a finalidade de possibilitar a troca de informações táticas do SCDM com outros sistemas, de forma segura, será necessário a configuração de um ambiente que contemple VPNs e firewalls

3.1.2 Requisitos do SCDM

1) O sistema deverá diferenciar cinco tipos de informações segundo prioridades (da maior para a menor):

- Mensagens Imediatas (MI) de dados (alta prioridade e pequeno volume de tráfego);
- Vídeos de Tempo Real (VTR);
- Dados Comuns (DC);
- VoIP (futuramente); e.
- Gerenciamento/ Controle da rede.

Obs: O fluxo de controle da própria infra-estrutura deverá ter alguma garantia de latência e de vazão para que o sistema atenda os seus requisitos, devendo, para esse fim, ser considerado como MI.

2) O sistema deverá diferenciar, dentro das classes de fluxo mencionadas no item anterior, clientes segundo prioridades. Os clientes são executados nos consoles, em nome de usuários comuns ou privilegiados.

3) O sistema deverá possuir controles que garantam a QoS das informações levando em conta as prioridades dos fluxos e dos clientes como, por exemplo, a admissão e a suspensão de fluxos.

4) O sistema deverá possuir um serviço de gerência pró-ativa da QoS. Esta gerência leva em conta o conjunto dos fluxos e dos clientes, suas prioridades e a conectividade do sistema (ou de suas seções). A manutenção dos limites de QoS aceitáveis deve ser preocupação constante no sistema.

5) O sistema deve ser tolerante a falhas quanto à distribuição dos fluxos de informações, garantindo a entrega aos clientes. Esta tolerância refere-se a:

- perdas de enlaces por defeitos físicos na infra-estrutura ou seção da rede entre origem e destino;

- falha de um dispositivo de interconexão (switch ou roteador).

6) O sistema deverá ser capaz de prover a qualquer console, o recebimento de até duas das fontes de vídeo radar, distribuídas através de multicast, a partir de servidores de vídeo selecionados, bem como a variação de escala de distâncias (no caso de vídeo radar) em qualquer uma dessas fontes.

7) Cada console poderá, a qualquer tempo, decidir pela comutação entre servidores de vídeo radar, passando a associar-se a outro(s) grupo(s) multicast de vídeo radar.

3.2 Infra-Estrutura de Suporte ao SCDM

A rede deverá ser constituída de componentes, protocolos e padrões abertos já disponíveis no mercado. Sua arquitetura está baseada no modelo TCP/IP, provendo roteamento unicast e multicast, reorganização de enlaces, resiliência dos comutadores e tratamento da QoS.

3.2.1 Roteamento unicast

A escolha da melhor rota para os pacotes será feita pelo algoritmo de roteamento, implementado nos comutadores nível três. Existem basicamente dois tipos: vetor de distância e estado dos enlaces (HUITEMA, 2000). Os protocolos do tipo vetor de distância (ex. RIP) possuem duas características proibitivas no SCDM, pois caso ocorra o rompimento de um enlace ou defeito em um comutador:

- propagam esta informação de salto em salto
- a verificação do enlace só ocorre de tempos em tempos

Por outro lado, os protocolos tipo estado dos enlaces propagam informações somente quando há uma alteração nos enlaces e a difundem para todos os enlaces, de forma que cada nó tem uma visão completa do sistema. Esta característica é essencial para a tolerância a falhas no SCDM.

Uma escolha natural seria uma extensão do OSPF para roteamento com QoS, o Q-OSFP. Seus objetivos são (APOSTOOPOULOS, G. et al, 1999):

1. Obter as informações necessárias para computar a QoS dos enlaces e selecionar o caminho capaz de atender aos requisitos de QoS de um dado pedido de conexão;
2. Estabelecer o caminho selecionado para acomodar o pedido de conexão; e.
3. Manter o caminho estabelecido.

Contudo, apesar do Q-OSPF ser um padrão e existirem trabalhos relacionados (MIEGHEM, V.P. et al, 2001), uma pesquisa nos principais fabricantes (3Com, Cisco, Juniper, NortelNetworks) de equipamentos de redes, revelou que esse protocolo ainda não é implementado. Sendo assim, esta opção está descartada.

Em conseqüência, nossa escolha será o OSPF, por: ser comercialmente difundido, facilitar o balanceamento de carga e possuir boa convergência (CISCO, 2004).

3.2.2 Roteamento multicast

O encaminhamento multicast é mais eficiente que a difusão *broadcast* por permitir a recepção dos dados apenas por membros de um grupo previamente definido, em vez de inundar os dados para toda a rede. É, também, mais eficiente que várias comunicações unicast (ponto a ponto) simultâneas porque faz melhor uso do meio de comunicação, evitando redundância de informação trafegada (KOEHLER, E., 2003).

Um cliente indica sua intenção de participar de um grupo através do IGMP (Internet Group Management Protocol). O IGMP permite também que os roteadores controlem e limitem o fluxo de tráfego multicast através da rede. Este controle é feito por mensagens IGMP (CISCO, 2005a).

O gráfico da figura 3.2 faz uma comparação entre diversos protocolos de multicast. As métricas de avaliação são o tamanho da rede e a carga gerada pelo protocolo na própria rede (KOEHLER, E., 2003).

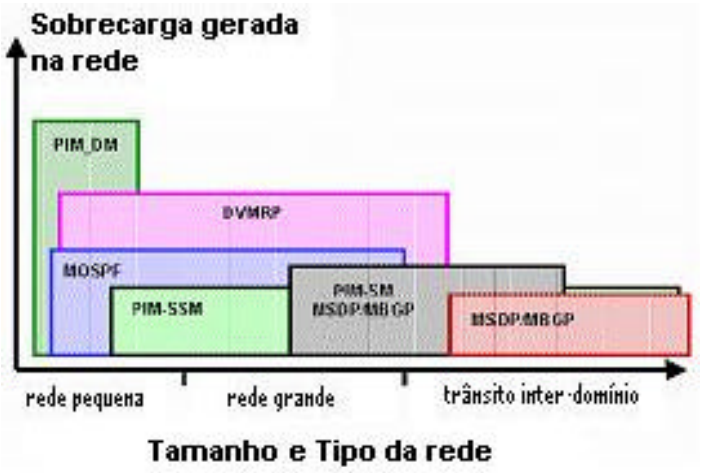


Figura 3-2 - Comparação de Protocolos de Roteamento Multicast

Pelo critério de dimensões da rede, os protocolos elegíveis seriam:

- PIM-DM e SSM (Protocol Independent Multicast/Dense Mode e Source Specific Multicast);
- DVMRP (Distance Vector Multicast Routing Protocol); e
- MOSPF (Multicast Open Shortest Path First).

O protocolo MOSPF requer que um nó calcule o menor caminho para todas as árvores multicast, para cada combinação de fonte e grupo. O número de computações que se seguem à qualquer atualização de rota cresce com o quadrado do tamanho da área, tendo cada computação a complexidade de O(N.logN) (HUITEMA, C., 2000).

Em (MORAES I. M., 2004) é feita uma comparação entre os protocolos DVMRP, PIM-DM, PIMSM e o PIM-SSM. Para avaliar o comportamento de cada protocolo, MORAES realizou experimentos que simulam a transmissão de vídeo. As métricas usadas para comparar o desempenho dos protocolos foram: a capacidade de reconfiguração da árvore de distribuição, a taxa de utilização dos enlaces da rede e o número e a carga de pacotes de controle de roteamento.

Segundo os autores, "os resultados mostram que protocolo DVMRP possui a maior carga de controle, por implementar o seu próprio protocolo de roteamento unicast. Já o PIM-DM, que usa um mecanismo de inundação-e-poda para construir sua árvore de distribuição, apresenta uma taxa de utilização da rede 25% superior à dos protocolos de mensagens de inscrição e poda explícitas, o PIM-SM e o PIM-SSM. Estes dois protocolos possuem a menor taxa de utilização da rede. Entretanto, o PIM-SSM gera a

metade da carga de controle de roteamento do PIM-SM. Por isso, o PIM-SSM é o protocolo mais indicado para ser usado no serviço de distribuição vídeo".

A escolha para o SCDM será, portanto, o protocolo PIM-SSM, por ser adequado a redes de pequeno e médio porte, ser compatível com o protocolo de roteamento unicast escolhido (OSPF), gerar baixa carga na rede e ter boa capacidade de reconfiguração da árvore de distribuição.

3.2.3 Reorganização de enlaces partidos

Caso ocorram falhas em algum enlace ou comutador, a rede deverá ser capaz de rapidamente ajustar-se à nova configuração. As situações que podem ocorrer são:

- rompimento de ligações entre comutadores: neste caso um protocolo tipo *Spanning Tree* (STP) irá habilitar uma ligação redundante, que fora previamente bloqueada;

- rompimento de uma das ligações entre o servidor ou cliente e um comutador: este caso pode ser visualizado na infra-estrutura da rede ilustrada na figura 3.1, onde um servidor está ligado por enlaces principal e alternativo com os dois comutadores. O sistema operacional presente nos *hosts* deverá detectar que uma das interfaces de rede encontra-se inoperante por falha, avisando à aplicação. Esta redirecionará o fluxo para a outra interface de rede. Obs: Esta aplicação também poderá fazer balanceamento de carga entre os caminhos principal e alternativo.

3.2.4 Resiliência dos comutadores

O protocolo VRRP (Virtual Router Redundancy Protocol – RFC 3768, HINDEN, R, 2004) foi projetado para eliminar pontos de falha únicos em ambientes roteados estaticamente, isto é, onde as rotas de saída da LAN para a WAN passam sempre pelo mesmo roteador. Com a arquitetura VRRP, tem-se redundância de conexões de saída entre a LAN e a WAN, como pode ser visto na figura 3.3.

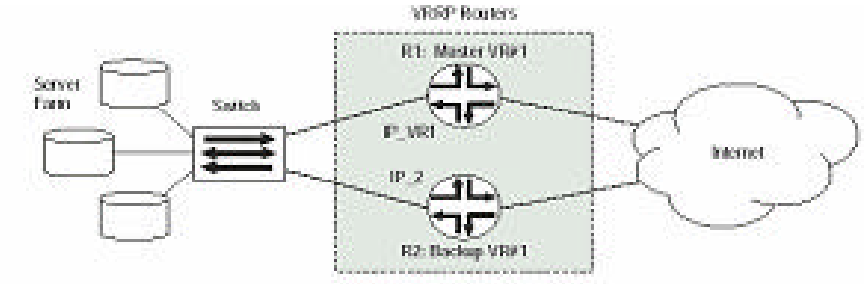


Figura 3-3 - Contexto de utilização do VRRP

Resumidamente, esse protocolo executa as seguintes funções:

- os roteadores redundantes são identificados por um grupo
- um único mestre é eleito para o grupo
- um ou mais roteadores podem ser reservas do mestre do grupo
- o mestre comunica seu estado aos dispositivos reserva
- se o mestre falhar, o VRRP tenta comunicação com cada reserva, segundo uma ordem de precedência;
- o reserva que responder primeiro assumirá como mestre do grupo.

Outros protocolos proprietários semelhantes são: HSRP (Hot Standby Router Protocol, Cisco) e IPSTB (IP Standby Protocol, DEC).

CROZIE, K. R. (1997), usando uma rede de teste, propõe três situações (falha nos enlaces entre um hub e um comutador, entre um roteador e um comutador e entre dois comutadores) e avalia seis situações de falha, indicado que o tempo mínimo de reconfiguração da rede varia de 4 a 12 segundos, com o HSRP. Já que o HSRP é um protocolo semelhante ao VRRP, concluímos que a solução de redundância usando camada três poderá ser demasiadamente lenta para o uso no SCDM.

Diferentemente das implementações de redundância em camada três, como VRRP ou HSRP, a 3Com Corp. desenvolveu a tecnologia XRN (eXpandable Resilient Networking), que permite às *switch fabric* dos comutadores serem distribuídas e trabalharem em conjunto, conforme a figura 3.4 (3Com Corp, 2004).

Por este esquema, os dois comutadores também são vistos como uma única entidade, possuindo um endereço IP e *gateway*, com uma máscara de sub-rede e sendo con-

trolável via SNMP. Em caso de falha de um comutador, o outro assume imediatamente as funções de comutação nos níveis dois e 3.

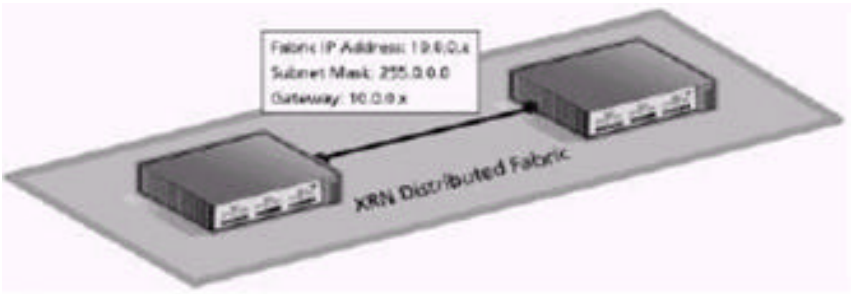


Figura 3-4 - Arquitetura XRN

O esquema XRN distribui informações de roteamento, como interfaces, processamento dos protocolos de roteamento e tabelas de roteamento por todos os comutadores pertencentes à *Distributed Fabric.* Neste caso, o hardware de comutação é distribuído.

O XRN habilita cada comutador membro a rotear pacotes localmente, à medida que estes surgirem em suas portas. Isto permite maximizar a performance de roteamento do conjunto, fazendo uso de toda a capacidade de largura de banda. Um esquema de ligações de servidores de vídeo com comutadores XRN pode ser visto na figura 3.5.

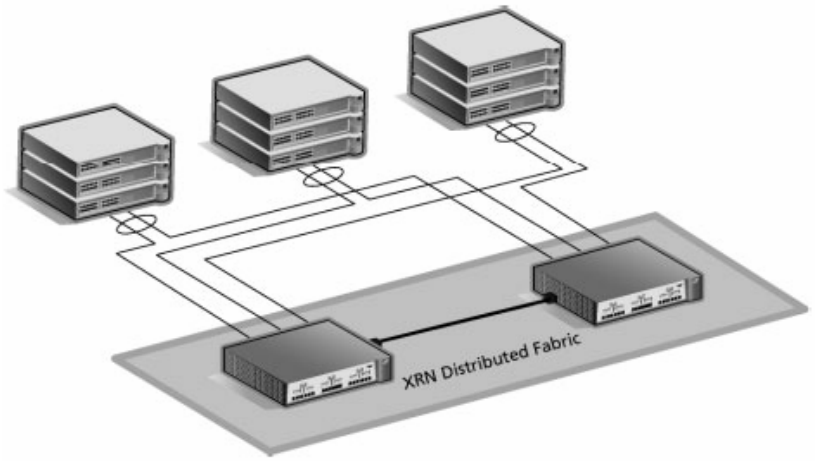


Figura 3-5 - Interligação de servidores de vídeo com comutadores XRN

Em (NEWMAN, D., 2003) são medidos os tempos de recuperação quando um dos enlaces ligados ao conjunto XRN é partido (ver figuras 3.1 e 3.5) e quando a alimentação de um dos comutadores XRN é desconectada. No primeiro teste, em 822 milisegundos todo o tráfego foi redirecionado ao enlace restante, enquanto que no segundo, em apenas 438 milisegundos, o comutador restante assumiu todo o tráfego.

O esquema de resiliência de comutadores a ser escolhido para o SCDM será a tecnologia XRN da 3Com.

3.2.5 Tratamento de QoS

Devido ao overhead gerado pela granularidade da arquitetura IntServ e às limitações do esquema Precedência no IP, a arquitetura escolhida para controlar a QoS no interior do SCDM será a arquitetura DiffServ, descrita no item 2.5.3.

A marcação do DSCP (RFC 2474) e a distribuição de classes no SCDM serão definidas conforme a tabela 3.1.

Classe do SCDM	classe Diffserv	DSCP
Mensagens Imediatas (MI) Gerenciamento/Controle da rede	EF	101110
Cliente prioritário	AF4_Alta	100110
vídeos alta prioridade (VAP)	AF3_Alta	011110
Voz alta prioridade (VoAP)	AF3_Média	011100
vídeos média prioridade (VMP)	AF2_Alta	010110
Voz média prioridade (VoMP)	AF2_Média	010100
vídeos baixa prioridade (VBP)	AF1_Alta	001110
dados comuns (DC)	BE	000000

Tabela 3-1 - Definição das classes de tráfego no SCDM

Pelo ilustrado nas tabelas 2.2, 2.3, 2.4 e 3.1 podemos notar que:

- quanto maior o valor do código DSCP, maior a prioridade do tráfego; e.
- será possível a definição de novas classes no futuro.

Como os pacotes serão classificados com base na 5-tupla (endereço IP fonte e destino, número da porta fonte e destino e o Protocolo de Transporte), o SCDM atribui-

rá uma prioridade a cada usuário quando este for autenticado por sua identificação e senha. O sistema utilizar-se-á da tabela 3.2 para a atribuição de uma classe a um usuário.

Classe	IP Fonte	IP Destino	Porta Fonte	Porta Destino	Protocolo
MI	X	X	Define	Define	TCP
Cli_Pri	X	X	Define	X	X
VAP	Define	X	X	X	UDP
VMP	Define	X	X	X	UDP
VBP	Define	X	X	X	UDP
DC	X	X	Define	Define	TCP

Tabela 3-2 - Critérios de atribuição de classes a usuários

Nesta tabela, o símbolo (X) significa que o item de classificação na coluna não influi na definição da classe. Observa-se que, por exemplo, o Cliente_Prioritário será identificado pela sua porta, independente do tipo de protocolo que vá usar, enquanto que os servidores de vídeo serão identificados pelo seu IP.

3.2.6 Recuperação de Perdas de Pacotes no SCDM

A rede pode descartar pacotes de forma aleatória (no item 4.2 é feita uma modelagem do processo de perdas na Internet), principalmente por congestão nos comutadores. Define-se episódio de perda como o evento de perda de pacotes consecutivos. Logo, um episódio de perda com tamanho dois significa que dois pacotes seguidos foram perdidos. Esta definição será aplicada no modelo matemático empregado no item 4.2. Além disso, o retardo nos pacotes pode ser grande o suficiente para considerá-los perdidos, o que os inviabiliza na aplicação.

Algumas técnicas têm sido propostas para contornar o problema e discutidas em (SILVA, E.S. et al, 2002). A proposta de (FIGUEIREDO, D.R., 1999), ilustrada na figura 3.6, pode ser incorporada aos codecs a serem usados no SCDM. Com este esquema é possível a recuperação parcial de um episódio de perda com tamanho quatro. No exemplo, se os pacotes 2,3,4, e 5 forem perdidos, poderão ser recuperados parcialmente, pois utiliza-se de esquemas de compressão com qualidade reduzida.

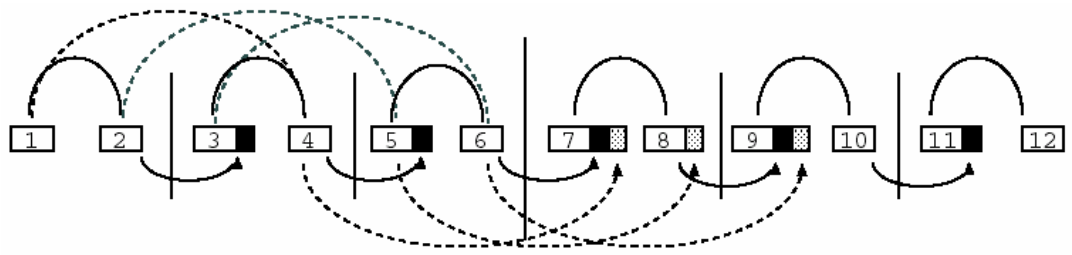


Figura 3-6 - Esquema de recuperação de perdas de pacotes do SCDM

3.3 Arquitetura do SCDM

O SCDM possui duas funções básicas: distribuição de fluxos multimídia e dados de gerência para garantir a operacionalidade com QoS, sempre com enfoque tolerante a falhas. Esta tolerância reflete-se na duplicação dos comutadores, nas ligações entre os clientes e servidores com os comutadores e aos dois gerentes ligados cada um ao seu comutador correspondente.

O panorama global do sistema também deverá ser replicado nos gerentes, de maneira que, caso um falhe, outro assuma o controle imediatamente.

No próximo capítulo, está descrita a implementação de uma pequena parte da Gerência do SCDM: o Inspetor de Desempenho.

3.3.1 Diretrizes do SCDM

As características principais do SCDM são o controle de admissão, segundo uma política preemptiva de prioridades, o monitoramento pró-ativo da QoS e a detecção de conectividade do sistema. Estas características estão aqui descritas por meio de diretrizes, elencadas nas sub-seções 3.3.1.1 a 3.3.1.5, enquanto que os critérios e linhas de ação a serem empreendidas pela Gerência serão tratadas na sub-seção 3.3.2.

3.3.1.1 Diferenciação dos Fluxos

O objetivo da diferenciação entre os fluxos no SCDM é orientar o controle de admissão e a distribuição dos recursos do sistema segundo uma regra comum, ou seja, por um valor numérico de prioridade. Define-se uma classe de clientes prioritários, in-

dependente do tipo de mídia e classes inferiores onde é o tipo de mídia que define a prioridade (os clientes têm a mesma prioridade). A diferenciação dos fluxos baseia-se na tabela 3.1 e está descrita na tabela 3.3.

3.3.1.2 Controle de Admissão

O controle de admissão será requisitado apenas na solicitação de estabelecimento de uma conexão. Este controle permitirá a conexão de acordo com a prioridade da classe solicitante. Uma conexão será aceita se houverem recursos disponíveis na rede (caso sua prioridade seja igual) ou se essa for mais prioritária que outras já existentes. Isto significa que, estando a rede em sobrecarga e dependendo da situação, algum fluxo/cliente poderá ser preterido em função de outro. Como exemplo, se o Comandante de um navio desejar ver qualquer vídeo, algum fluxo ou cliente menos prioritário poderá ter sua conexão degradada e até mesmo suspensa. Mas se dois clientes de escalão inferior, porém de mesma prioridade, escolherem vídeos com importâncias diferentes, o vídeo prioritário terá mais recursos da rede enquanto que o outro poderá ser degradado ou suspenso, dependendo da prioridade relativa destes fluxos em relação aos já aceitos e do estado da rede no momento.

Prioridade	Classe do SCDM	Sigla da Classe
0	Mensagens Imediatas Gerenciamento/Controle da rede	MI
1	Cliente prioritário	Cli_Pri
2	vídeos alta prioridade	VAP
3	Voz alta prioridade	VoAP
4	vídeos média prioridade	VMP
5	Voz média prioridade	VoMP
6	vídeos baixa prioridade	VBP
7	dados comuns	DC

Tabela 3-3 - Diferenciação entre os fluxos de mídia no SCDM

3.3.1.3 Distribuição dos fluxos

Os fluxos de vídeo não serão armazenados em lugar algum ao longo do sistema, visto que se trata de geração de informações e respectivo consumo em tempo real. Logo, o cliente receberá os quadros que estiverem sendo produzidos a partir do momento da sua solicitação. Esta poderá ser feita, por exemplo, com o cliente solicitando adesão a um grupo multicast de um dos servidores de vídeo.

3.3.1.4 Monitoramento da QoS

O monitoramento da QoS no cliente inicia-se quando este tentar estabelecer uma conexão com um servidor. O processo de estabelecimento da conexão instancia um Inspetor de Desempenho (ID), que permanecerá relacionado apenas a este fluxo específico.

Logo, em um cliente, será possível a co-existência de diversos Inspetores, cada qual enviando alarmes de QoS à um Gerente. Como os Gerentes comunicam-se permanentemente, qualquer informação relevante enviada para um será replicada nos outros.

3.3.1.5 Conectividade do Sistema

Uma das diretrizes mais importantes é que o SCDM deverá manter-se operacional mesmo com um mínimo de conectividade. Este possuirá adaptabilidade quanto ao estado operacional dos comutadores ou ligações de rede, significando que mesmo que a rede seja totalmente secionada por falhas, cada seção sobrevivente (A e B) deverá envidar todos os esforços para continuar funcionando, isto é, os seus clientes continuarão a receber os vídeos dos servidores alcançáveis.

O secionamento do sistema poderá ser total, no pior caso, ou parcial. Este tipo também terá gradações, dependendo de quantos enlaces restantes continuem a interligar as partes da rede.

O primeiro caso, de secionamento total, está ilustrado na figura 3.7 se desconsiderarmos inicialmente a ligação em pontilhado. Tudo transcorre como se existissem dois sistemas independentes: SCDM_A e SCDM_B. Cada seção terá a sua própria ge-

rência (aqui simplificada para uma representação concentrada e monitorando seu respectivo comutador). Neste caso, os clientes de 1 a m vão poder solicitar vídeos dos servidores 1 e 2. Observa-se que todos os requisitos da seção 3.1.2 continuam sendo aplicáveis, mas a cada uma das seções isoladamente.

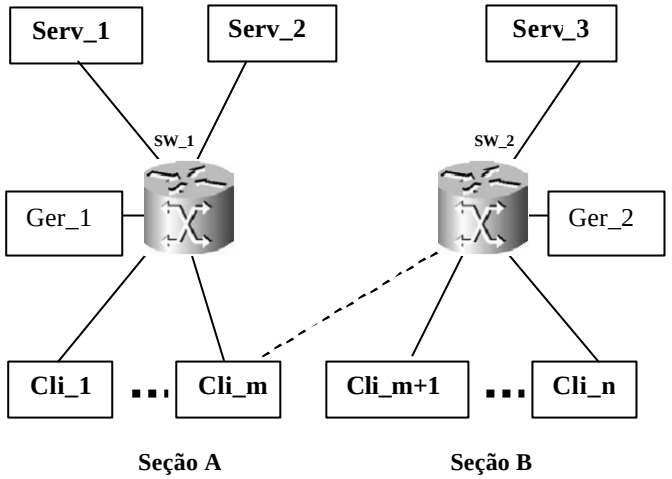


Figura 3-7 - Secionamento do SCDV devido à falhas

Outra situação é o secionamento parcial, onde resta(m) ligação(ões) entre as seções A e B. Neste caso, a pior hipótese é ilustrada na figura 3.7, mas agora considerando-se a ligação em tracejado. Nesta situação ainda existe uma ligação entre o Cliente_m e o Comutador_2 (em tracejado). Observa-se que, a princípio, a função do Cliente_m não é comutar pacotes e sim consumi-los. Se o Gerente_1 tiver conhecimento deste secionamento, poderá solicitar ao Cliente_m que re-encaminhe informações (Mensagens Imediatas) ao Comutador_2. Outra linha de ação é distribuir a gerência por todas as estações, capacitando-as com inteligência de modo que o próprio Cliente_m passe a encaminhar pacotes adiante, por intermédio de seu software aplicativo. O envio de fluxos de vídeo por este enlace poderá ser muito restrito, visto que se trata de um gargalo entre as duas seções.

3.3.2 Gerenciamento do SCDM

Como mencionado no item 3.1, uma das duas partes que constituem o SCDM é uma entidade distribuída que o controla, colhendo informações e atuando pró-ativamente na rede de infra-estrutura, sempre com uma visão panorâmica do que ocorre nos diversos componentes do sistema.

A Gerência do SCDM deverá monitorar e reagir às ações dos mecanismos existentes de tolerância a falhas e QoS, aliada a uma política de segurança. Esta política tem as diretrizes mencionadas na seção 3.3.1.

Além disto, como a Gerência será calcada no paradigma "peer-to-peer", o panorama global do sistema é replicado nos gerentes (em número de dois, no caso ilustrado na figura 3.7), de maneira que, caso um falhe o outro assuma o controle imediatamente.

3.3.2.1 Objetivos da Gerência

Os objetivos da gerência pró-ativa são:

- efetuar o controle de admissão de novos fluxos, segundo a política preemptiva, descrita em 3.3.1.2;
- monitorar, através de algoritmos inteligentes e preditivos, qualquer piora na QoS fim-a-fim das aplicações;
- detectar a conectividade do sistema (ou de uma seção deste)
- efetuar o diagnóstico panorâmico do sistema
- manter a QoS fim-a-fim das aplicações, degradando-a o mínimo possível, caso ocorram falhas em comutadores ou ligações.

Um diagnóstico é uma função diferente de uma monitoração. Enquanto esta apenas detecta a ocorrência de um fenômeno, de forma isolada e sem vistas a descobrir suas causas, aquele procura investigar as causas, que freqüentemente estão relacionadas a outros componentes do sistema.

Desta forma, o monitoramento é uma função adequada ao Inspetor de Desempenho(ID) e o diagnóstico, ao Gerente de Desempenho(GD), de forma semelhante à descrita em (CECÍLIO, E.L, 2002).

Para executar essas tarefas, a Gerência conta com os IDs, os GDs e a inteligência dos algoritmos de manutenção e operação da rede (prevenção de loops de comutação, roteamento, resiliência dos comutadores, etc). Os Inspetores de Desempenho são responsáveis pelo monitoramento da QoS fim-a-fim entre um servidor e um cliente. Cada ID reporta alarmes para o seu GD. Os GDs trocam informações entre si, mantendo em cada GD um panorama do sistema. Uma decisão é tomada pela Gerência sempre fundamentada no panorama do sistema.

3.3.2.2 Critérios da Gerência

A fim de atender aos requisitos do SCDM, elencados em 3.1.2, a Gerência executa os objetivos mencionados em 3.3.2.1. Estes devem ser norteados por critérios, descritos nesta seção, que envolvem a localização de um problema e a seleção da mais adequada linha de ação para o momento.

A avaliação do Estado do sistema e a escolha da melhor linha de ação estão inter-relacionadas. Dependendo do tipo de problema que está ocorrendo no sistema, a solução será diferente. Se, por exemplo, a QoS num determinado cliente começar a degradar, a gerência da rede deve avaliar se a causa desta degradação é local no cliente, na rede ou no servidor. Por outro lado, se a causa estiver na rede, por exemplo, poderá ser adotado o rerroteamento de algum tráfego, a ser definido para o momento em questão.

Avaliação do Estado do Sistema

A Gerência deve tentar localizar se um problema é:

- apenas em um fluxo deste cliente ou em outros fluxos
- somente em um cliente ou em outros também
- contido na mesma área da rede, apenas
- a rede está congestionada (provavelmente por queda de algum link)
- o servidor está congestionado (aumenta o retardo, perda de pacotes, etc);
- o cliente está congestionado (com muitos processos rodando)

Escolha de Linhas de Ação:

Após a gerência ter avaliado o estado global do sistema, algumas medidas podem ser tomadas, tais como:

- rerroteamento de tráfego pelos comutadores do sistema
- redistribuição (com fins a balanceamento) de tráfego na saída dos servidores
- redução de recursos de rede alocados a um fluxo
- redução da quantidade (com conseqüente degradação da qualidade) de tráfego de saída dos servidores
- suspensão de um fluxo:
  - total nos servidores ou
  - parcial a partir de certo ponto na rede

3.3.2.3 Gerência Distribuída

A Gerência, além dos critérios definidos anteriormente, envolve dois aspectos da computação distribuída: a detecção de falhas e o consenso.

A detecção de falhas ocorre pelo não recebimento de uma mensagem ou sinal tipo "heart-beat" enviada periodicamente por um processo *p* a outro processo *q*, passando por uma rede que introduz retardos aleatórios. A questão é: quando *q* irá considerar que *p* falhou?

No problema do consenso, além das questões relacionadas à detecção de falhas, deve-se garantir que um grupo de processos (p.ex. gerentes) decida por um mesmo valor V, como resultado de uma mesma computação, dado que cada processo p possa ter proposto um valor diferente.

Os autores de (CHANDRA, T.D.,1996) introduzem o conceito de detectores de falhas não confiáveis e como estes podem resolver o problema do consenso em sistemas assíncronos.

Em (CHEN, W., 2000) são propostas métricas de QoS e mostra-se como elas podem qualificar completamente o serviço de um detector de falhas: *tempo de detecção* (velocidade do detector), *tempo entre falhas consecutivas* e *duração das falhas* (define o tempo para o detector corrigir o erro). Em (MACÊDO, R.J.A., 2004) é proposta uma melhoria das métricas do detector, em (CHANDRA, T.D.,1996), usando-se Redes Neurais.

Em (GOERENDER, S., 2002) é proposto um modelo para tolerância a falhas em Sistemas Distribuídos, mas considerando QoS e as arquiteturas IntServ e DiffServ. Este modelo é adaptável às condições de QoS, utilizando um Detector de Falhas que provê a detecção de falhas dependendo da QoS do ambiente de execução.

A linha de pesquisa indicada nos parágrafos acima permite à gerência distribuída (localizada em dois computadores gerentes, conforme a figura 3.7, ou em cada estação, conforme a arquitetura peer-to-peer) lidar com os aspectos da detecção de falhas e com o consenso.

Capítulo 4 Escopo de Implementação do SCDM

Neste capítulo são apresentadas diversas decisões de implementação de uma parte do SCDM. A plataforma de hardware utilizada é o computador pessoal tipo PC, o protocolo para transmissão de vídeo adotado é o RTP/RTCP, a plataforma de software é a Java/JMF, que é descrita em 4.1. Além disto, adotou-se o *codec* como H.263 (vide 4.2). A topologia de implementação está descrita em 4.3, o Inspetor de Desempenho em 4.4, os modelos matemáticos usados estão descritos em 4.5 e por fim, o Emulador Paramétrico, utilizando-se desses modelos, está descrito em 4.6.

4.1 Definição da Plataforma de Software

Uma das primeiras definições de implementação desta dissertação foi a adoção do Java Media Framework (JMF) como infra-estrutura, visto que este implementa o protocolo RTP e já possui diversos codecs integrados. O JMF 2.0 suporta a captura, o processamento, o envio, a recepção e a apresentação de mídia em diversos formatos, além de permitir aos desenvolvedores controle adicional sobre estas tarefas. Ele provê uma arquitetura com plug-ins, que permitem acesso direto aos dados e customizações de processos. Para maiores detalhes veja o Anexo D.

Após a definição de uso do JMF, verifica-se que esta ferramenta é fornecida pela SUN com alguns codecs já implementados, apesar de permitir a implementação de qualquer algoritmo de compressão/apresentação de mídia.

Conforme (SUN MICROSYSTEMS, 1999), os formatos de vídeo implementados e atualmente suportados pelo JMF quando se utiliza o empacotamento RTP são: H.261, H.263 e MJPEG. Esta condição foi decisiva para a definição do codec utilizado nesta dissertação, conforme descrito no item 4.2, a seguir.

4.2 Codificação a ser empregada

Como visto em 4.1, a fim de permitir compatibilidade entre o protocolo de transporte de mídia de tempo real (RTP) e a infra-estrutura de implementação (JMF) adotados, três codificações de vídeo podem ser utilizadas - H.261, H.263 e MJPEG. Já que o H.263 é uma codificação posterior à H.261, a escolha será entre esse e o MJPEG.

Para cada uma destas duas codificações foram efetuadas medidas, cujos arquivos contendo os traços para a H.263 estão descritos no item C.3.2.

A partir dos traços para a codificação MJPEG, gerou-se os gráficos das figuras 4.1 a 4.3.

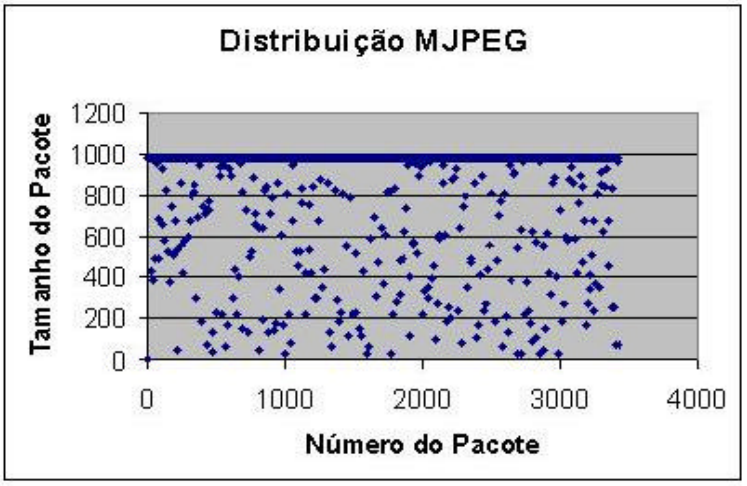


Figura 4-1 - Tamanho de pacotes (bytes) para a codificação MJPEG

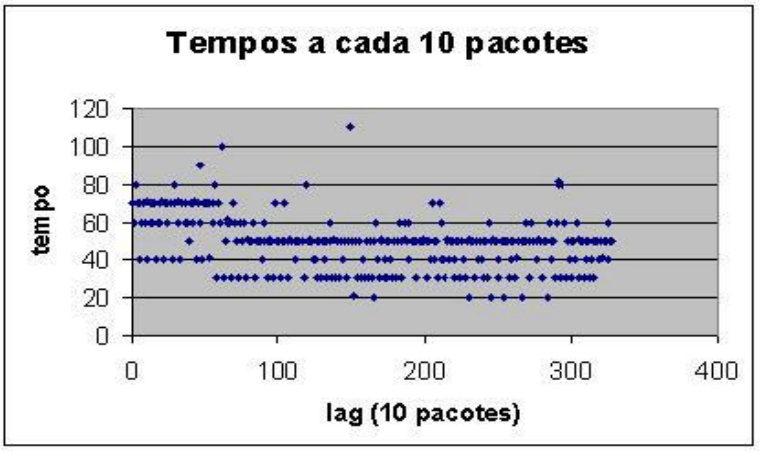


Figura 4-2 – Tempos (ms) a cada 10 pacotes para a codificação MJPEG

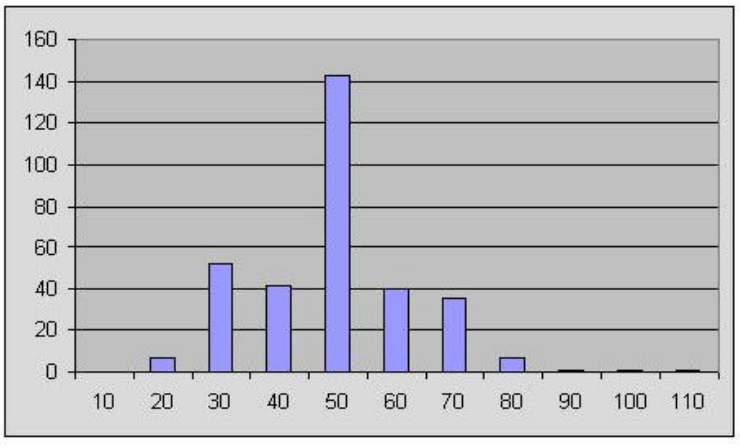


Figura 4-3 – Distribuição de tempos(ms) a cada 10 pacotes para MJPEG

Verifica-se que o tamanho médio dos pacotes é da ordem de 1000 bytes e o tempo entre cada um deles é, em sua grande maioria, da ordem de 5ms, o que equivale a uma taxa média de 1,6 Mbps (note-se que este valor, originado do vídeo comprimido, é muito inferior aos 36,5 Mbps do vídeo YUV bruto, como descrito no anexo C1.).

A partir dos *traces* para a codificação H.263, gerou-se os gráficos das figuras 4.4 a 4.7.

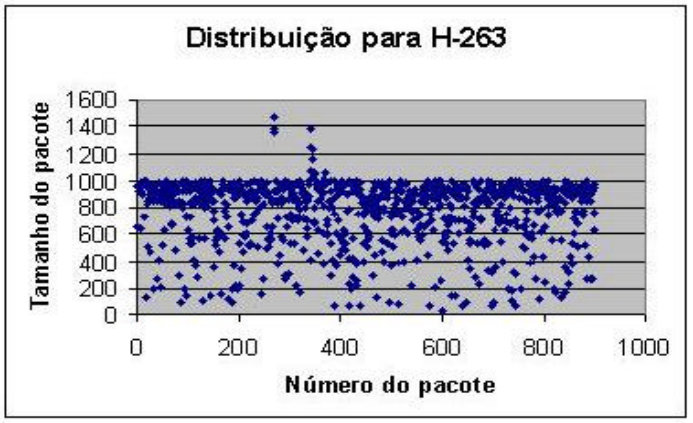


Figura 4-4 - Tamanho de pacotes (bytes) para a codificação H.263

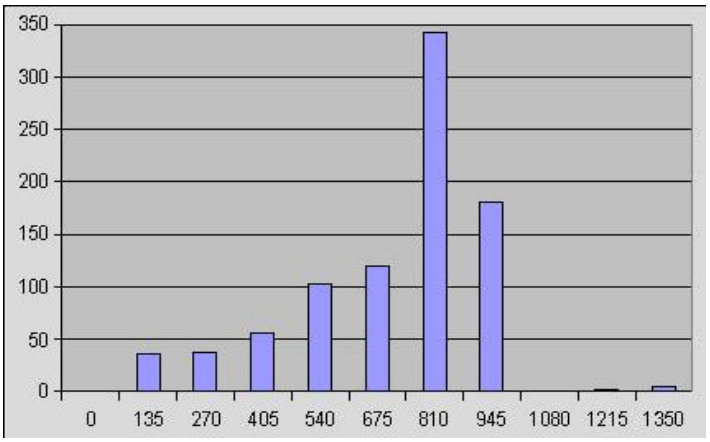


Figura 4-5 – Distribuição de tamanhos dos pacotes para H.263

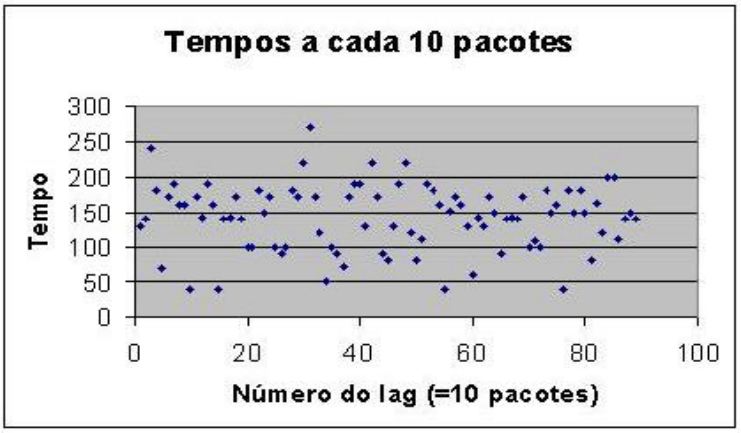


Figura 4-6 – Distribuição de tempos(ms) a cada 10 pacotes para H.263

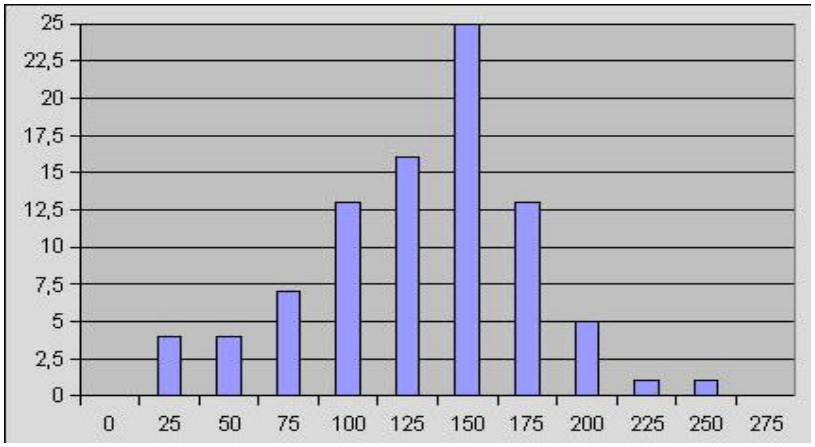


Figura 4-7 – Distribuição dos tempos(ms) entre 10 pacotes para H.263

Verifica-se que o tamanho médio dos pacotes é da ordem de 800 bytes e o tempo entre cada um deles é da ordem de 15ms, o que equivale a uma taxa média de 427 Kbps. Deve-se lembrar que o codificador foi configurado para comprimir em VBR, não sendo adotada nenhuma taxa alvo de bits, como descrito no Anexo C.

Em função da taxa média do H.263 ser aproximadamente 25% da taxa ofertada pelo codificador MJPEG, essa foi escolhida como a compressão a ser utilizada nesta dissertação.

4.3 Topologia Adotada para a Implementação

Por ser o SCDM um sistema complexo, decidiu-se limitar o escopo de implementação neste trabalho ao Inspetor de Desempenho (ID), responsável por monitorar uma conexão, entre um cliente e um servidor e com a possibilidade de gerar alarmes de tendência de piora da QoS. Estes alarmes serão informados à Gerência, que poderá escolher dentre as linhas de ação descritas em 3.3.2.2 para normalizar preventivamente o sistema.

Foi adotado o Java Media Framework (JMF) para a implementação dos códigos do Transmissor e do Receptor, segundo a topologia ilustrada na figura 4.8.

No transmissor, foi adotada a codificação H.263 para enviar, através do protocolo RTP, um vídeo de aproximadamente 10s para o receptor.

O Emulador Paramétrico (descrito em 4.6), permite a variação dos parâmetros relevantes para a rede do SCDM (jitter e perdas), ocorrendo segundo os modelos matemáticos descritos em 4.5.

A fim de viabilizar a implementação do ID, adotou-se o seguinte esquema de implementação:

Figura 4-8 - Topologia de Implementação do SCDM

4.4 O Inspetor de Desempenho Neuro-Fuzzy

Um cliente (programa de aplicação sendo executado em um console) pode solicitar a recepção de um vídeo proveniente de algum servidor do sistema. Para o estabelecimento desta conexão é instanciado um Inspetor de Desempenho (ID).

4.4.1 Objetivos do Inspetor de Desempenho

Os IDs monitoram a variação de retardo e a taxa de pacotes perdidos (parâmetros de rede relevantes para o SCDM) da conexão e podem gerar alarmes de tendência. Estes alarmes serão informados aos seus Gerentes, que poderão tomar medidas (segundo os critérios descritos em 3.3.2.2) a fim de normalizar preventivamente o sistema.

Com o objetivo de gerar o mínimo possível de alarmes falsos, o Inspetor de Desempenho possui um algoritmo inteligente que pode ser alimentado com os seguintes dados:

- o *jitter* médio entre pacotes (implementado na versão atual);
- a taxa de perdas média (implementado na versão atual); e
- o número de pacotes atualmente no *playout buffer* deste Cliente (a ser implementado em versão futura).

Note-se que o *jitter* e as perdas têm uma relação direta com o estado presente da rede, enquanto que número de pacotes no *buffer* do receptor está relacionado com o estado futuro da aplicação. Este estado futuro relaciona-se à (não) violação na Qualidade de Percepção (QoP - descrita no item 2.3), pois, mesmo ocorrendo degradação do estado (*jitter*) presente da rede, o receptor vai continuar consumindo os pacotes que já estão em seu *buffer*, não interrompendo a apresentação do vídeo. Este fato não com-

promete a QoP, mesmo com a piora da QoS de rede. A QoP permanecerá razoável se o estado da rede não ficar degradado por muito tempo.

Dado este contexto de manutenção da QoP frente à piora da QoS, a geração de alarmes falsos ocorrerá se o ID anunciar uma degradação "passageira" da QoS de rede, isto é, ainda existiriam pacotes no *buffer* do receptor, suficientes para manter a apresentação do vídeo com qualidade (QoP) por mais algum tempo.

Nesta dissertação, usa-se um classificador neuro-fuzzy para avaliar se, apesar da piora do estado da rede (QoS) é realmente necessário o envio de um alarme ou se o Inspetor de Desempenho pode aguardar um pouco mais.

No capítulo 5, são descritos o método de treinamento do classificador e os resultados obtidos.

4.4.2 Dinâmica de Monitoramento do ID

O Inspetor de Desempenho (ID) monitora dois tipos de conexões: ponto-a-ponto e ponto-multiponto (multicast).

No caso de ponto-a-ponto, a monitoração ocorrerá entre um transmissor (servidor) e um receptor (cliente), enquanto que para o caso ponto-multiponto a monitoração ocorrerá no transmissor e em cada receptor, sendo que cada nó tem conhecimento da qualidade da conexão com os demais. Em ambos os casos, cada *peer* da conexão (um nó da sessão RTP) executará uma instância de monitoração, seja como transmissor ou como receptor.

O conhecimento da qualidade de outras conexões, envolvidas na sessão RTP, é importante, por exemplo, em uma videoconferência, onde um "participante" necessita saber se sua informação está sendo recebida com qualidade pelos demais e, em caso negativo, avaliar quais conexões apresentam problemas e tomar medidas compatíveis. Note-se que estes mecanismos são provenientes da concepção do próprio protocolo RTP, que é em essência, *peer-to-peer*. Este protocolo também foi projetado para distribuição (multicast) de vídeo (vide Anexo A).

Nos nós (*peers*) receptores é executado o programa AVReceiver.Java, enquanto no transmissor, executa-se o AVTransmitter.Java. Estes programas foram desenvolvidos utilizando a API do JMF.

Uma sessão RTP funciona recebendo/transmitindo pacotes de dados (mídia) empacotados por RTP e, de acordo com o intervalo de reportagem para a sessão, pacotes compostos RTCP são trocados entre todos os participantes da sessão.

O transmissor envia aos receptores *Sender Reports* (SR) enquanto que os receptores enviam aos outros participantes (o transmissor e os outros receptores) seus *Receiver Reports* (RR). É através deste mecanismo que todos os participantes conhecem o estado global da sessão, incluindo a qualidade (pacotes perdidos, *jitter*) das outras conexões da sessão.

4.4.3 Estatísticas da Conexão RTP

O protocolo RTCP provê mecanismos próprios para esta monitoração. Estes são o Sender Report (SR) e o Receiver Report (RR). O JMF, além de implementar o protocolo RTP/RTCP disponibiliza uma série de estatísticas que são mantidas pela classe RTPManager, para cada conexão (sessão RTP). Para outros detalhes, vide seção D.4.1.

Para a sessão como um todo tem-se:

- GlobalTransmissionStats: mantém estatísticas cumulativas de transmissão para todos os transmissores locais (note-se que uma sessão RTP pode ter mais de um fluxo de mídia, por exemplo, áudio e vídeo)
- GlobalReceptionStats: mantém estatísticas globais de recepção. Estas estatísticas podem ser obtidas com o método getGlobalReceptionStats(). Algumas estão ilustradas a seguir:
    - getBytesRecd() e getPacketsRecd() - número de bytes e pacotes recebidos
    - getRTCPRecd() - número de pacotes RTCP totais recebidos
    - getSRRecd() - número de pacotes SR totais recebidos
    - getBadRTPkts() - número de pacotes RTP recebidos com algum problema
    - getLocalColls() e getRemoteColls() - colisões, na tentativa de estabelecimento de um SSRC (um identificador único para o *peer*)

As estatísticas para um peer particular ou fluxo de saída são:

- ReceptionStats: mantém estatísticas de recepção para um participante individual
- TransmissionStats: mantém estatísticas de transmissão para fluxo individual

4.5 Modelos Matemáticos usados no Emulador

Nos últimos anos, muitas abordagens foram sugeridas para garantir a QoS das aplicações multimídia. Os componentes principais utilizados para este propósito são: os sistemas localizados nos extremos da rede e no interior da mesma (formando um domínio, por exemplo).

A compreensão do comportamento dos parâmetros de QoS - tais como as perdas, o atraso e o *jitter* - é um elemento crucial para muitos dos esforços de pesquisa para tornar a Internet melhor do que o serviço de "melhor-esforço", já ofertado pelo protocolo IP.

De posse de modelos matemáticos adequados, é possível o uso de um Emulador Paramétrico (descrito a seguir em 4.6) para controlar os parâmetros que determinam o comportamento da rede e responder a perguntas tais como:

- qual a capacidade mínima que o canal de comunicação deve ter ?
- qual o retardo máximo admitido pela aplicação ?
- qual a variação máxima admitida para o retardo ?
- qual a taxa máxima de perda de pacotes admitida pela aplicação ?

4.5.1 Modelo de Perdas

A congestão na rede implica em taxas de perda e em atrasos maiores, e conseqüentemente, o desempenho resultante da aplicação multimídia será degradado. Diversos fatores geram perdas de pacotes nas redes de comunicação. Entre eles podemos citar o transbordo (*overflow*) de filas nos roteadores, descarte de pacotes (se, por exemplo, ele chegar atrasado) pelos protocolos envolvidos na comunicação e a troca de bits no meio físico, que em geral é ocasionada por distorção dos sinais tais como ruído, atenuação ou eco.

4.5.1.1 Fontes de perdas nos roteadores

Se a carga de tráfego entrante em um roteador for maior que a sua velocidade de encaminhamento, os pacotes serão armazenados temporariamente em seus *buffers*, e

aguardarão seus instantes de serem designados às respectivas interfaces de saída. O tamanho dos *buffers* é finito, logo, se estes já estiverem cheios, qualquer novo pacote entrante provocará um descarte. Ou seja, os pacotes são perdidos devido ao transbordamento do *buffer*.

Note-se que o pacote a ser descartado não necessariamente será o último pacote a chegar, visto que existem diversas políticas de tratamento das filas nos *buffers*. É possível que outro pacote seja selecionado para descarte, permitindo a acomodação deste último pacote recebido. Políticas de descarte preemptivas são essenciais ao SCDM porque viabilizam a prioridade entre fluxos de mídia, isto é, os fluxos pertencentes a uma classe DiffServ de mais alta prioridade terão maiores garantias de recursos e menor probabilidade de descarte ao longo do sistema.

O transbordamento dos *buffers* é a causa principal de perda nas redes cabeadas (como a rede do SCDM), que contribuem para mais de 99% de todos os pacotes perdidos (JACOBSON, V., 1988). Deve-se notar que as perdas de pacotes devido ao transbordamento de um *buffer* é uma conseqüência de seu tamanho finito e da congestão na rede (MARKOVSKI, V., 2000).

A fim de reduzir a probabilidade de perdas na rede, deve-se projetá-la cautelosamente, procurando garantir que os pacotes a atravessem com uma alta taxa de probabilidade (p.ex. 99,99% do tempo não ocorrerão perdas de pacotes). O dimensionamento da rede exige necessariamente a completa compreensão do tráfego que esta deverá suportar. Ou seja, a caracterização estatística do tráfego é uma condição primordial para a garantia da baixa probabilidade de perdas na rede.

Após o dimensionamento da rede para as condições ideais - sem falhas (nos conectores, cabos, comutadores, interfaces de rede, etc) ou sobrecargas de processamento (nos servidores ou clientes) é a Gerência Pró-Ativa do SCDM que envidará todos os esforços para manter a alta disponibilidade de mídias para todos os clientes.

4.5.1.2 Caracterização das perdas

Medidas feitas com tráfegos na Internet, tanto para encaminhamento unicast quanto para multicast indicaram que a probabilidade de perda de k pacotes consecutivos (episódio de perda de tamanho k, ilustrados na figura 4.8) diminui aproximadamente de forma geométrica com k (YAJNIK, M., et al, 1999) e (BOLOT, J.C., et al, 1999).

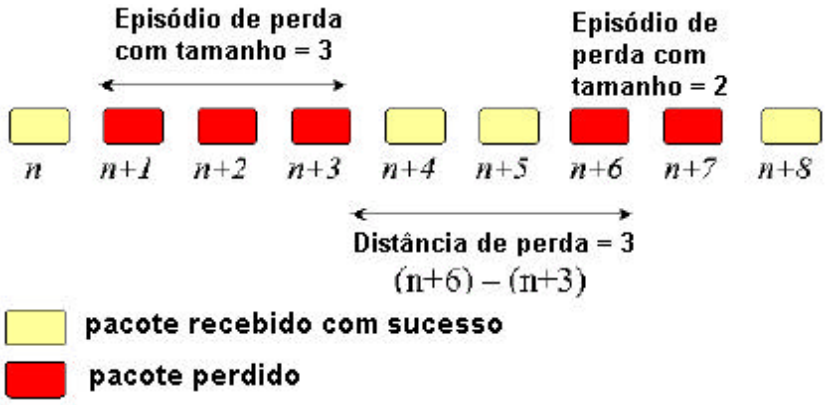


Figura 4-9 – Episódios de Perda de Pacotes

Ou seja, a contribuição dos episódios para a perda agregada em um *buffer* de um roteador e, mesmo para as perdas em um único fluxo UDP, mostrou possuir diminuição aproximadamente geométrica com aumento do tamanho do episódio de perda (MARKOVSKI, V., 2000).

Este comportamento é bem representado com o Modelo de Gilbert simplificado. Para os episódios de perda mais longos, o modelo de Gilbert simplificado desvia dos dados simulados e subestima a contribuição de episódios mais longos. O modelo estendido de Gilbert descreve a contribuição de episódios de perda de forma mais precisa. Entretanto, ambos os modelos não contabilizam as durações de pacotes recebidos consecutivamente com sucesso.

Nesta dissertação, será adotado o modelo simplificado de Gilbert visto que:

- os *codecs* a serem adotados no SCDM poderão tratar de episódios de perda com tamanhos de até 4;
- trata-se de uma rede local e cabeada. Logo o SCDM estará num ambiente com carga de tráfego "previsível", portanto, para uma razoável faixa de condições de operação, os episódios de perda não serão muito grandes. O caso de interrupção dos serviços (falhas) significa episódio de perda com tamanho muito grande, sendo tratado pelos mecanismos de tolerância a falhas da infra-estrutura do SCDM; e
- as perdas também poderão ocorrer por sobrecargas de processamento, sendo possível a atuação pró-ativa da Gerência

4.5.1.3 Modelo Simplificado de Gilbert

O diagrama de estados para o modelo simplificado de Gilbert (uma cadeia homogênea de Markov com dois estados) é ilustrado na figura 4.9.

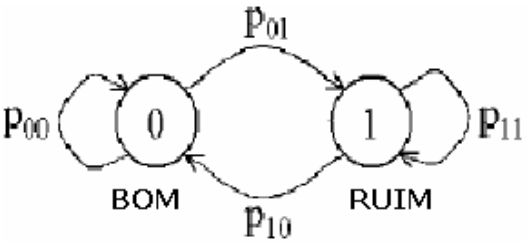


Figura 4-10– O modelo simplificado de Gilbert

O estado X = 0 representa um estado da rede no qual os pacotes chegam com sucesso enquanto que X = 1 representa um estado onde, durante o tempo em que a rede estiver nele, todos os pacotes que chegarem serão perdidos. Note-se que essa cadeia de Markov é discreta, onde p_{00} a p_{11} representam probabilidades de transição de estado.

O estado da rede, sendo então função de eventos discretos (e não do tempo, como no caso contínuo), está sujeito a transições a cada pacote que entra na rede.

p_{01} e p_{10} representam as probabilidades de transição de estado, dadas pelas equações 4.1:

p01 = P(X_n = 1| X_n = 0) = P (o pacote n é perdido dado que o pacote n-1 é recebido)

p10 = P(X_n = 0 | X_n = 1) = P (o pacote n é recebido dado que o pacote n-1 é perdido).

Equações (4.1)

Para o regime estacionário, as probabilidades de estado podem ser expressas no seguinte formato de matriz:

$$\begin{bmatrix} 1-p_{01} & p_{10} \\ p_{01} & 1-p_{10} \end{bmatrix} \begin{bmatrix} P(X=0) \\ P(X=1) \end{bmatrix} = \begin{bmatrix} P(X=0) \\ P(X=1) \end{bmatrix}$$

Equação (4.2)

Da equação 4.2 e da equação P (0) + P (1) = 1, obtém-se a probabilidade incondicional P de perda:

$$P(X=1) = \frac{p_{01}}{p_{01}+p_{10}}$$ **Equação 4.3**

Visto que esta probabilidade foi obtida com a premissa da rede ter alcançado o regime estacionário, a partir equação 4.2, P(X=1) também relaciona-se ao tempo médio em que a rede estará no estado RUIM. Este fato será utilizado, no capítulo 5, na escolha dos parâmetros deste modelo, com o objetivo de levantar gráficos de desempenho do ID.

A probabilidade condicional de perda é igual a probabilidade de ocorrer uma perda, dado que o pacote precedente foi perdido, que é $P(X_n = 1 \mid X_n = 1) = 1 - p10 = p11$, ou seja, a probabilidade de permanecer no estado X=1.

A equação 4.4 indica a probabilidade de ocorrência de um episódio de perda com comprimento k (k pacotes perdidos consecutivamente), dado que a rede entrou no estado X = 1 (durante o qual todos os pacotes serão perdidos):

$$p_k = (1 - p_{10})^{k-1} \cdot p_{10}. \qquad \textbf{Equação 4.4}$$

Conseqüentemente, os comprimentos dos episódios de perda (no modelo de simplificado Gilbert) são geometricamente distribuídos. O modelo simplificado de Gilbert tem uma memória de somente um evento passado. Ou seja, a probabilidade de que o evento seguinte seja "pacote recebido com sucesso" ou "pacote perdido" depende somente do estado precedente, visto que só existem dois estados.

4.5.2 Modelo de *Jitter*

O retardo fim-a-fim consiste nos atrasos de transmissão e de propagação mais um atraso variável de enfileiramento ao longo da rede. A fim de simplificar os termos desta soma, suponhamos que, caso todos os pacotes utilizem a mesma rota do transmissor ao receptor, eles terão o mesmo atraso de propagação, e que caso os pacotes tenham o mesmo tamanho, o atraso de transmissão também será o mesmo.

Porém, mesmo que todos os pacotes utilizem a mesma rota até o receptor e tenham o mesmo tamanho, eles sofrerão diferentes níveis de enfileiramento dentro da rede. Este fato é o que causa a variação no atraso fim-a-fim, ou jitter (MOON, S.B., 1999).

O jitter pode ser causado por diversos fatores tais como a variação do tamanho das filas de transmissão nos nós intermediários da rede (enfileiramento), alteração das

rotas entre origem e destino (propagação) ou uso de pacotes com tamanhos variáveis (transmissão).

A variação do tamanho dos pacotes pode ser minimizada usando-se a técnica de fragmentação dos pacotes. Neste caso, os pacotes IP são divididos em datagramas com um tamanho máximo especificado, minimizando a variação do tempo necessário para serialização dos mesmos.

Quanto ao tempo de enfileiramento, este pode ser minimizado priorizando-se o tráfego de mídias contínuas em relação aos outros tipos de tráfego (KROPOTOFF, A.B., 2002), como por exemplo, ocorre com a classe EF de um domínio DiffServ.

Em (TRUMP, T., 2000) a variação no atraso é modelada como um processo aleatório. O autor usa processos aleatórios identicamente distribuídos, com distribuição exponencial. Note-se que o fato de termos processos identicamente distribuídos significa que, para cada pacote a atravessar a rede teremos uma nova "rodada" ou "execução" do mesmo processo com distribuição exponencial. Isto é, o evento jitter que cada pacote sofre ao atravessar a rede é totalmente independente do evento jitter para o próximo pacote. Os autores de (MOON, S.B., 1999) também assumem em suas simulações uma distribuição exponencial para a variação de atraso fim-a-fim.

Nesta dissertação, iremos monitorar a variação de retardo com a plataforma PC / Windows XP, que nos testes apresentou precisão de medidas em torno de 10 ms (o que depende do relógio do sistema). Como a variável aleatória exponencial descreve eventos que ocorrem em tempo contínuo e só conseguimos medi-los aqui em tempo discreto, adotamos seu equivalente discreto: a variável aleatória geométrica.

Seja, então, X uma V.A. que denota o número de falhas até que ocorra o primeiro sucesso. Se cada falha estiver associada a um intervalo de 10ms (a precisão do sistema operacional), então X pode representar o número de intervalos em que o pacote ficará retido na rede até que seja por esta liberado (sucesso ao atravessá-la).

O retardo que a rede introduzirá, segundo esse modelo, depende dos valores que a V.A. X pode assumir e da precisão de medidas do sistema operacional (10ms), ou seja:

retardo da rede = 0, 1, 2, ... x 10 ms. **Equação 4.5**

A probabilidade com que a V.A. X assume estes valores é dada por:

$$\mathrm{P}(X = k) = (1 - p)^k p \quad \text{, k = 0, 1, 2, ...} \quad \textbf{Equação 4.6}$$

O valor médio da V.A. X é dado por:

$E[X] = \frac{1-p}{p}$, onde p é o parâmetro da geométrica. **Equação 4.7**

Note-se que, o valor a ser programado no Emulador Paramétrico será justamente o valor médio de X. No item 5.1 adiante, mais detalhes são fornecidos sobre esta decisão.

4.6 O Emulador Paramétrico

O Emulador Paramétrico utilizado nesta dissertação foi originado a partir de (KROPOTOFF, A.B., 2002) e foi implementado em código Java.

Algumas aplicações tais como distribuição de vídeo digital, VoIP (Voz sobre IP), videoconferência e comércio eletrônico, precisam prover serviços em tempo real sobre uma arquitetura de rede que não garante QoS.

A avaliação do impacto das perturbações introduzidas pelas redes LAN/WAN (jitter, latência, perda de pacotes, etc) na qualidade final dessas aplicações é uma tarefa ca, dependem, inclusive, do tipo da aplicação, da carga de processamento exigida dos complexa. Isto ocorre devido a diversos fatores: as perturbações têm natureza randômi computadores que executam os aplicativos, etc.

Usando-se um emulador, é possível criar diversos perfis de comportamento de uma rede. Sendo possível repetir um mesmo experimento diversas vezes, mantendo o comportamento da rede inalterado, o que é muito difícil de se conseguir em uma rede real e bastante útil para identificar e isolar problemas. Além disso, usando um emulador, é possível controlar individualmente os diversos parâmetros que compõem o comportamento da rede, permitindo assim avaliar o impacto de cada um deles separadamente.

4.6.1 Implementação do Emulador Paramétrico

O esquema do Emulador Paramétrico implementado nesta dissertação está ilustrado na figura 4.11. De forma diversa do trabalho de (KROPOTOFF, A.B., 2002), aqui pode-se usar apenas uma CPU, visto que os endereços de transporte diferenciam-se pelas portas.

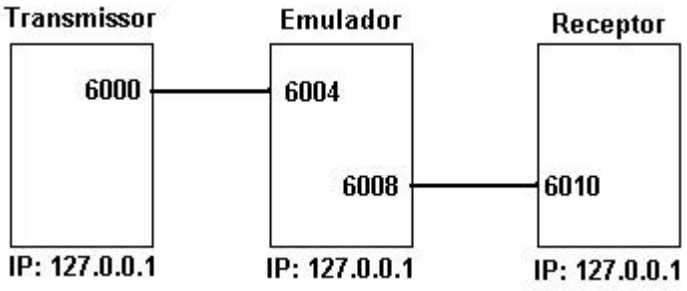


Figura 4-11 – Esquema do Emulador Paramétrico implementado

O elemento central do Emulador é uma fila_temporal, consistindo de uma tabela hash indexada pelo tempo de saída do pacote. Resumidamente, temos os seguintes pseudocódigos mais relevantes:

```
thread main():
-estabelece os soquetes UDP
-ativa as threads:  RTP_IDA
                    RTCP_IDA
                    Despachante
                    RTCP_VOLTA
                    RTP_VOLTA
- while(true) {
                 verifica_estado_rede ()
               }
```

```
thread RTP_IDA:
-recebe pacote proveniente do transmissor
-Se (estado_rede = BOM)
{
        obtém tempo_chegada_pkt
        tempo_saída = tempo_chegada_pkt
                        + jitter_geométrico()
        insere na fila_temporal
}
```

```
thread Despachante:
- obtém tempo_atual
- obtém tempo de saída do 10 pacote na fila
- Se (tempo_atual >= tempo_saída)
{
        remove da fila_temporal
        envia pacote ao Receptor
}
```

As threads RTCP_IDA e RTP_IDA encarregam-se, respectivamente de fluxos RTCP e RTP, no sentido transmissor-para-receptor, ao passo que as threads

RTCP_VOLTA e RTP_VOLTA encaminham estes fluxos no sentido receptor-para-transmissor.

O método *verifica_estado_rede ()* implementa a cadeia de Markov homogênea com os dois estados: BOM e RUIM (modelo simplificado de Gilbert) para o modelo de perdas. O método *jitter_geométrico()* retorna um valor randômico, com distribuição geométrica, para o modelo adotado de *jitter*. Os valores da média da geométrica e as probabilidades de transição para o modelo de Gilbert serão descritos no capítulo 5.

Capítulo 5 Resultados

Neste capítulo, apresenta-se os resultados relativos ao escopo de implementação definido no capítulo anterior. O Emulador paramétrico necessitou ser validado e o Classificador Neuro-Fuzzy, presente no Inspetor de Desempenho, foi treinado segundo as condições de rede definidas para os modelos matemáticos adotados.

5.1 Considerações sobre *jitter*

Duas questões centrais nesta dissertação são: o jitter e a sua medida. Tornam-se, pois, necessários alguns comentários sobre o método de medidas adotado e as simplificações no conceito de jitter.

5.1.1 Simplificações adotadas

Na seção 2.1.1, foi visto que o retardo fim-a-fim sofrido pelos pacotes pode ser causado por diversos fatores tais como:

- enfileiramento (propagação nas filas de transmissão nos nós intermediários da rede);
- propagação (tempo de trânsito entre origem e destino); e
- transmissão (tempo necessário à serialização do pacotes).

Contudo, podemos considerar que:

- todos os pacotes utilizam a mesma rota do transmissor ao receptor. Note-se que, pela topologia adotada para a implementação (item 4.3), este é o caso. Além disto, o SCDM é concebido para uma rede local, onde não existem tantas alternativas de rotas como no caso da internet. Portanto, esta simplificação é bem razoável. Como a rota é a mesma para todos os pacotes, eles terão o mesmo atraso de propagação.

- os pacotes têm tamanhos próximos. A codificação empregada para o vídeo de testes foi o H.263 (vide item 4.2) e, conforme o gráfico da figura 4.4, esse vídeo foi constituído de aproximadamente 900 pacotes. Observando-se a figura 4.5, percebe-se que

71% dos pacotes têm tamanhos entre 675 e 945 bytes, contrastando com 13,8% dos pacotes tendo tamanhos menores que 405 bytes ou com 0,77% dos pacotes com tamanho de 1350 bytes. Caso os pacotes tenham tamanhos próximos, seus atrasos de transmissão também o serão.

Devido às duas considerações acima, a parte variável do retardo fim-a-fim reduz-se ao atraso de enfileiramento. Como o *jitter* é a variação no atraso fim-a-fim, somente esta parcela será considerada no jitter.

5.1.2 Cálculo do *jitter*

Neste item, são tecidas considerações que visam complementar o exposto no anexo A, item A.3.1.1, quando o *interrarrival jitter* é descrito em mais detalhes.

O *interrarrival jitter* é definido pela RFC 3550 (SCHULZRINNE, H. et al., 2003) como uma estimativa da variância do tempo de trânsito na rede dos pacotes emitidos de uma fonte para um dado receptor.

O *jitter* entre pacotes J é definido como sendo o desvio médio da diferença D temporal entre a chegada de dois pacotes no receptor em relação ao tempo quando foram enviados pelo emissor. Se Si for o tempo de envio do pacote i, e Ri for o instante de chegada no receptor, então para os pacotes i e j, D pode ser expresso como:

$$D(i,j) = (Rj - Ri) - (Sj - Si) = (Rj - Sj) - (Ri - Si) \qquad \textbf{Equação 5.1}$$

O *jitter* entre pacotes deve ser continuamente calculado à medida que cada pacote i é recebido, computando-se a diferença D entre este pacote i e seu precedente i-1, na ordem em que chegaram (não necessariamente na mesma seqüência em que foram enviados), de acordo com a fórmula:

$$J(i) = J(i-1) + (|D(i-1,i)| - J(i-1))/16. \qquad \textbf{Equação 5.2}$$

A RFC 3550 declara que este algoritmo é um estimador ótimo de primeira ordem da média do jitter e que o parâmetro de amortecimento 1/16 provê boa redução de ruído, mantendo uma razoável taxa de convergência (CADZOW, J., 1987).

A equação 5.2 pode ser reescrita como:

$$J(i) = 15/16.J(i-1) + 1/16.|D(i-1,i)| , \qquad \textbf{Equação 5.3}$$

que é uma média ponderada recorrente, com pesos 15 e 1, entre o jitter anterior e a diferença de trânsito relativo entre os pacotes anterior e atual. O comportamento desta média, para um D constante (e igual a um) pode ser visto na figura 5.1.

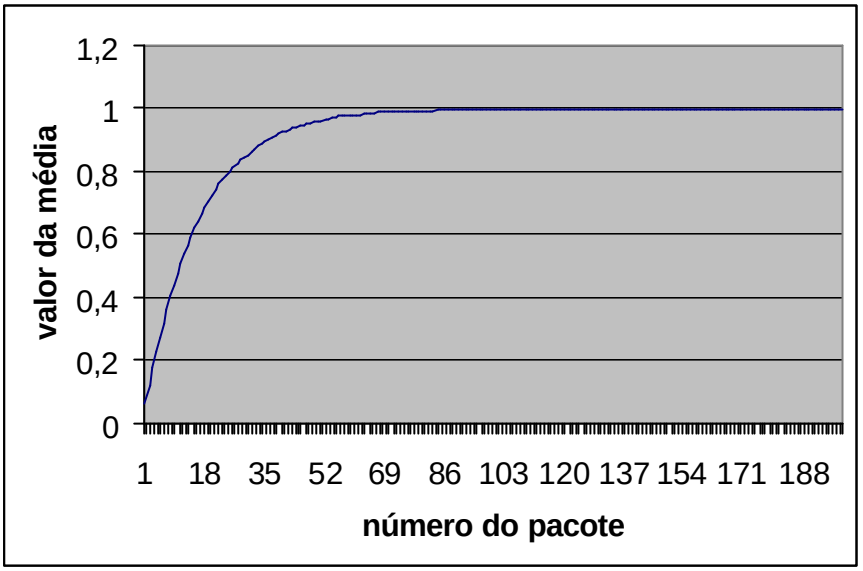


Figura 5-1 – Comportamento da fórmula do *jitter* definida pela RFC 3550

Apesar da validação da fórmula do jitter ou de seus pesos não estarem no escopo desta dissertação, podemos notar que, com exatamente 72 pacotes, a média alcança o valor de 0,99040747, ou seja uma diferença de 1% do objetivo (notar que a entrada é constante e igual a 1). Este comportamento assemelha-se à carga exponencial de um capacitor, com constante de tempo igual a 16, o mesmo fator de amortecimento da fórmula. Outro dado é que, para um vídeo com 24 frames/seg (filme de 16 mm), codificado, por exemplo em H.263 (que, em média, adota 3 pacotes por frame), em um segundo tem-se aproximadamente 72 pacotes.

Pode-se inferir que a equação 5.2 foi constituída para suportar padrões comerciais, podendo por exemplo, rastrear o jitter (com uma diferença de 1%) com um tempo de convergência de 1 segundo. Estes valores podem constituir uma solução de compromisso entre a convergência do algoritmo, o processamento exigido e a não percepção por um observador humano (o conceito de QoP), não sendo crítico em aplicações como distribuição de vídeo pela internet.

5.2 Medidas de validação do Emulador Paramétrico

A fim de permitir o uso do Emulador Paramétrico, este precisa antes ser validado. Para tal tarefa, a topologia de testes adotada na figura 5.2 foi utilizada. O transmissor emite pacotes de dados, este tráfego é alterado pelo Emulador, segundo os modelos matemáticos descritos no item 4.5 e finalmente o receptor perfaz estatísticas sobre os pacotes coletados. Os pseudocódigos para o transmissor e o receptor sintéticos são descritos nos itens E1 e E2, respectivamente.

Figura 5-2- Topologia de Validação do Emulador Paramétrico

A função do Transmissor Sintético é enviar pacotes UDP de forma controlada: tanto o conteúdo do pacote quanto o intervalo de tempo entre os pacotes são conhecidos, *a priori*, nos testes. O conteúdo, ou carga útil, dos pacotes constitui-se de um contador seqüencial (4000 pacotes, para os testes conduzidos) e de uma estampa de tempo em milisegundos, obtida a partir do relógio de tempo real do sistema. O intervalo de tempo para a emissão dos pacotes é sempre uma constante, isto é, não há *jitter* na geração do tráfego sintético.

O Receptor Sintético, ao receber os pacotes que sofreram *jitter* e/ou perdas por parte do Emulador Paramétrico, pode então validar a sua correta atuação, através da verificação estatística desse tráfego. Como exemplo, se o Emulador é programado para introduzir *jitter* com distribuição geométrica com média de 40 ms, o receptor tem condições de verificar se os pacotes são recebidos efetivamente com esta distribuição.

Para todos os experimentos de validação do Emulador Paramétrico, mencionados neste item, o Transmissor Sintético foi programado para gerar pacotes UDP com comprimento de 1000 bytes.

O primeiro experimento teve como objetivo validar toda a topologia de testes. Para tanto foram conduzidos três testes (onde o Emulador não introduziu *jitter* nem perdas): a emissão dos pacotes ocorrendo com intervalos de 10, 15 e 20 milisegundos. Ou seja, no primeiro teste, todos os pacotes foram gerados pelo transmissor com intervalos fixos a cada 10 ms. Em seguida outro experimento, mas com intervalos fixos de 15 ms e

finalmente, com 20 ms. O objetivo aqui era verificar se existe alguma variação de comportamento da topologia e, principalmente, do Emulador. Este teste foi bem sucedido.

Neste ponto pode-se compreender porque o modelo adotado nesta dissertação não foi o exponencial, mas o geométrico. Note-se que na tabela E1 (na seção E3 – Jitter entre pacotes) as diferenças de tempo entre os pacotes recebidos estão, na esmagadora maioria, expressos em múltiplos discretos de 10 ms, mesmo que sejam gerados a partir de valores variáveis. Ou seja, as diferenças normalmente são: 10, 20 (e não os 15 gerados, por exemplo) e 30 milisegundos, ao invés de serem espalhadas por um espectro contínuo. Isto ocorre devido à precisão de medidas do sistema operacional (Windows XP ou até mesmo o Linux) ficar em torno de 10 ms, o que depende da temporização (relógio) do sistema. Como a variável aleatória exponencial descreve eventos que ocorrem em tempo contínuo e só conseguimos medi-los aqui em tempo discreto, adotamos seu equivalente discreto: a variável aleatória geométrica.

5.2.1 Verificação do *Jitter* introduzido pelo Emulador

Neste experimento, o Emulador não introduz perdas de pacotes, ou seja, estes chegam em seqüência, sem interrupção de seu contador (que fora previamente inserido como carga útil). O transmissor mantém uma variação no intervalo de geração dos pacotes: 15 e 20 ms. Agora, porém, o Emulador introduz *jitter* com distribuição geométrica, de acordo com as médias indicadas nos testes abaixo relacionados.

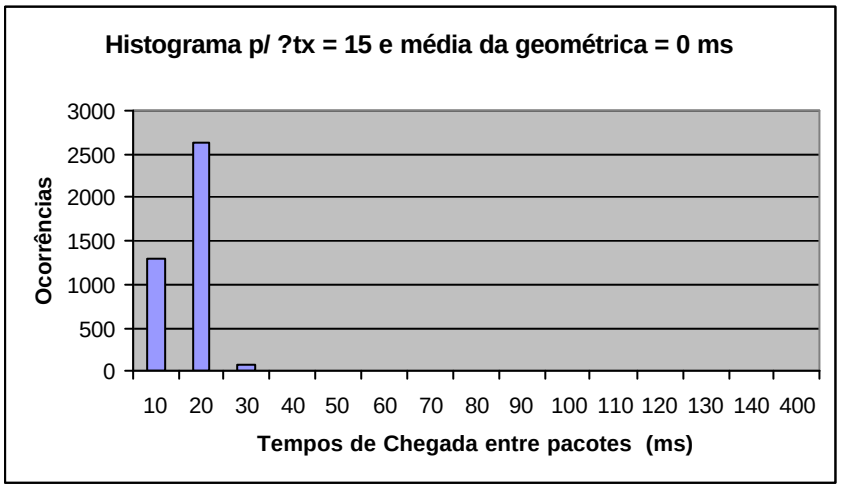
O receptor, ao coletá-los, efetua cálculos de diferença nos tempos medidos de recepção. Um extrato deste tipo de medida está ilustrado na tabela E1. À medida que os pacotes vão chegando e as diferenças de tempo sendo computadas, um histograma vai sendo construído para cada teste realizado. Um exemplo de histograma é ilustrado na tabela E2 (seção E3), para o caso do transmissor emitir cada pacote com intervalo fixo de 15 ms e o Emulador introduzir jitter com distribuição geométrica com média de 20 ms.

Podemos ver nas figuras 5.3 e 5.4 como a precisão de medidas de tempo do sistema operacional afeta o resultado final. Temos, para estes dois casos, o transmissor programado com os intervalos de emissão de pacotes de 15 e 20ms respectivamente e o Emulador programado para a média do jitter (modelado por uma variável geométrica) próxima de zero. Podemos ver que, em ambos os casos, apesar da média teórica (pro-

gramada no Emulador) ser próxima a zero, para a maioria dos pacotes ocorre uma variação no intervalo entre estes de 10 < *Jitter* = 20. Estes valores coincidem com os intervalos de emissão de pacotes, gerados pelo transmissor sintético.

Sendo a resolução de medidas do sistema operacional igual a 10 ms (dependente do relógio do sistema), ocorrem arredondamentos nas medidas. Para o caso da figura 5.3 alguns pacotes são recebidos com 0 < *Jitter* = 10 (quando o intervalo de geração de 15ms mais o *jitter* introduzido pelo Emulador for arredondado para baixo) enquanto que a maioria dos pacotes são recebidos com 20 < *Jitter* = 30 (lembrar que a média deste caso foi zero, o que não exclui as probabilidades da variável aleatória assumir valores mais altos). Para o caso da figura 5.4 (média = 20 ms) a proporção entre os pacotes nos intervalo 20 < *Jitter* = 30 e 0 < *Jitter* = 10 e maior do que no caso da figura 5.3, como esperado pois a geração ocorreu a intervalos maiores.

Nestes dois testes, não percebemos ainda o caráter geométrico do jitter, pois as médias teóricas (programadas) são muito menores que a média medida (gerada pelo Emulador Paramétrico).



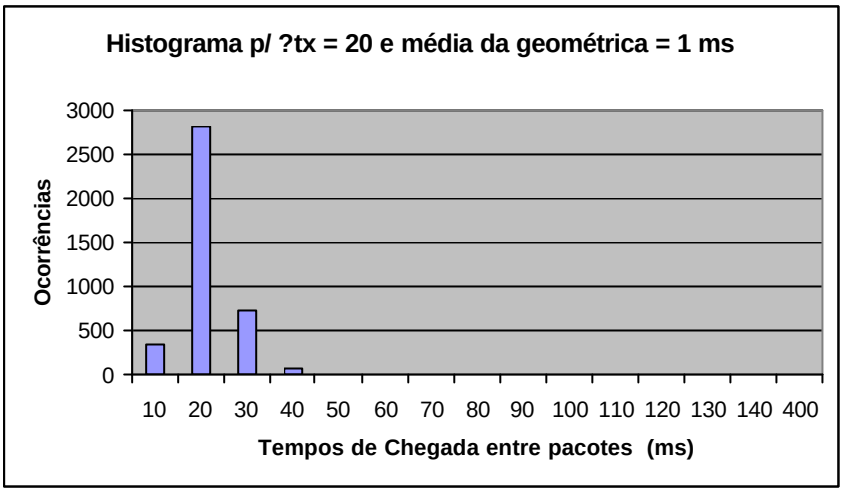
Média medida = 16,78

Média teórica = 0

Figura 5-3 – Histograma para ?tx = 15 ms e média da geométrica = 0 ms

Nas figuras 5.5 e 5.6 já é possível a observação do comportamento geométrico do *jitter*. Observa-se também que as médias teórica e medida coincidem. Usaremos o caso da figura 5.5 para explicar este fato: mesmo que o transmissor sintético envie pacotes com intervalo fixo de 15 ms, o Emulador introduz um jitter que é, em média, superior (= 20).

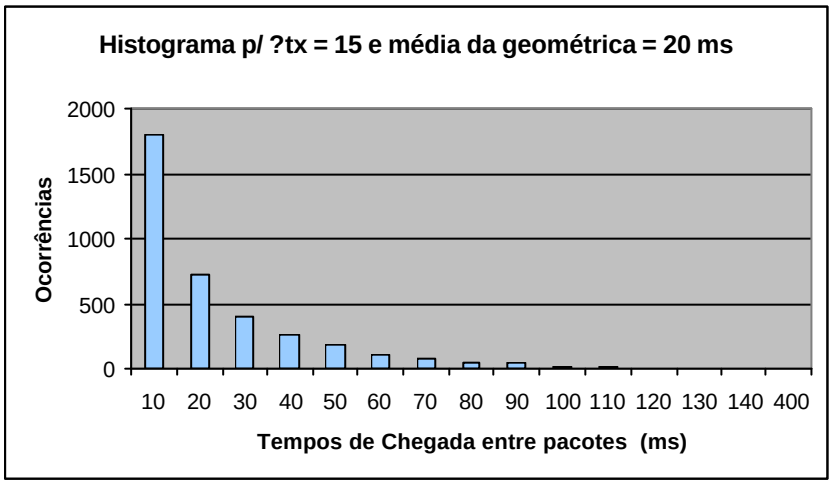
Observa-se também que mesmo o transmissor usando o intervalo de 15 e de 20 ms entre cada pacote, o sistema de medidas adotado continua funcionando corretamente.



Média medida = 21,06

Média teórica = 1

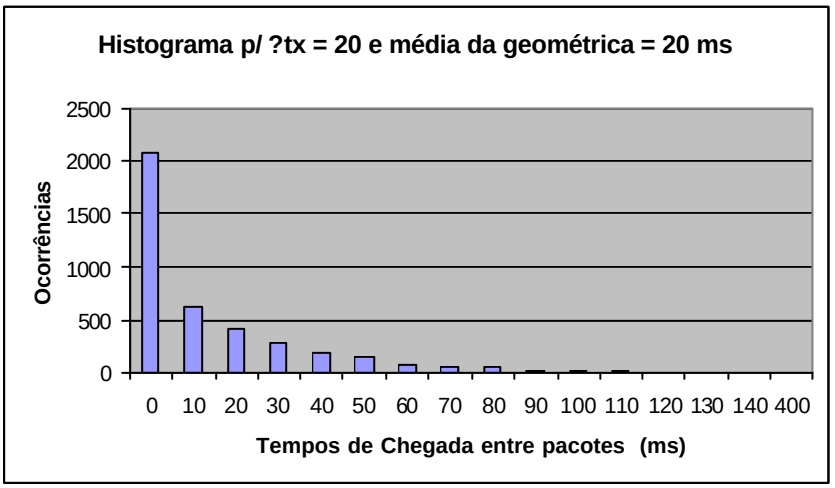
Figura 5-4 - Histograma para ?tx = 20 ms e média da geométrica = 1 ms



Média medida = 21,89

Média teórica = 20

Figura 5-5 – Histograma para ?tx = 15 ms e média da geométrica = 20 ms

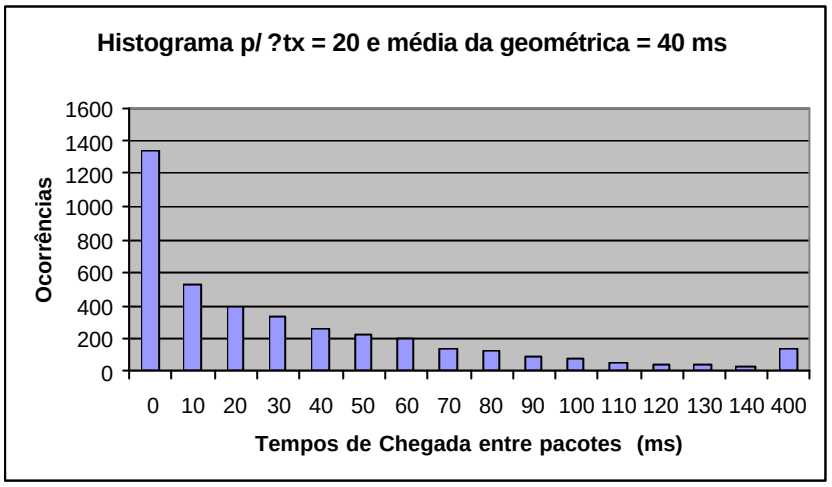


Média medida = 21,51

Média teórica = 20

Figura 5-6 - Histograma para ?tx = 20 ms e média da geométrica = 20 ms

O comportamento da média da variável geométrica também pode ser observado de compararmos os grupos de figuras 5.3 e 5.4 com 5.5 e 5.6 e com a figura 5.7. Note-se que quanto maior a média programada para a variável geométrica, mais o histograma será espalhado. Inclusive, na figura 5.7, notamos que dos 4000 pacotes gerados, 131 deles apresentaram variação na faixa 150 < *Jitter* = 400, demonstrando maior espalhamento que nos demais casos.



Média medida = 42,28

Média teórica = 40

Figura 5-7- Histograma para ?tx = 20 ms e média da geométrica = 40 ms

Com isto, temos a certeza de que o Emulador Paramétrico está introduzindo *jitter* com distribuição geométrica de forma correta. Deve-se, no entanto, ter em mente que se este for programado com um valor, para a média da geométrica, inferior ao período de envio entre os pacotes, não será possível a verificação no receptor desta característica geométrica. Este fato ocorre por limitações de precisão de medidas do relógio de tempo real do sistema operacional.

5.2.2 Verificação das perdas introduzidas pelo Emulador

Neste experimento, o Emulador não introduz *jitter* entre os pacotes e o transmissor mantém o intervalo de geração dos pacotes para cada um dos dois testes: 10 e 40 ms. Agora, porém, o Emulador introduz perdas com distribuição markoviana, segundo o modelo de Gilbert indicado no item 4.5.1.3.

O modelo de Gilbert também está associado a eventos discretos - a chegada de cada um dos pacotes, ou seja, para cada pacote que tenta atravessar a rede IP, é feito um "sorteio" para decidir se este será ou não perdido.

Enquanto a rede IP estiver no estado RUIM, todos os pacotes que tentarem atravessá-la serão perdidos, o que caracteriza os episódios de perda, ilustrados na figura 4.9 no item 4.5.1.2. O número médio de eventos em que o sistema estará no estado RUIM é dado pela equação 4.3.

Logo, para estes testes, os parâmetros do modelo de Gilbert são programados no Emulador segundo a equação 4.3 enquanto que, no receptor, verifica-se os comprimentos dos episódios de perda pelo cômputo de lacunas no contador seqüencial que fora inserido originalmente na carga de cada pacote. Estes episódios deverão decrescer geometricamente com seu comprimento, conforme (YAJNIK, M., et al, 1999) e (BOLOT, J.C., et al, 1999).

Foram conduzidos dois testes. Em ambos, exatamente 1600 pacotes foram emitidos pelo transmissor sintético, o tamanho dos pacotes foi de 1000 bytes e o período de amostragem da cadeia de Markov foi de 10 ms. Este é o período com que a *thread* implementada no Emulador vai consultar o estado da rede e decidir pelo descarte ou não do pacote. Quanto menor este intervalo, mais processamento é exigido da CPU, porém mais preciso torna-se o funcionamento do Emulador. O valor de 10 ms foi adotado por três motivos:

- o tempo entre dois pacotes H.263 (a codificação adotada) é maior do que 10 ms, para a maioria dos casos, vide a figura 4.7 na seção 4.2. Logo, a rede tem boa precisão para avaliar cada pacote, o que deve ser feito pois o modelo de Gilbert é discreto e acionado a cada evento (a chegada de um pacote);
- a precisão de medidas do sistema operacional é de 10 ms, logo não adiantaria um período de amostragem menor; e
- é uma solução de compromisso entre precisão do modelo e carga de processamento exigida da CPU.

No receptor, pelo cômputo dos episódios de perda e pela freqüência com que ocorreram, calculam-se quantos pacotes foram perdidos no experimento. A taxa média medida de perdas é a razão entre os pacotes perdidos e os transmitidos, que deve ser idêntica ao que fora programado no Emulador, através da média teórica, dada pela equação 4.3.

Configurações para o primeiro teste:

TRANSMISSOR:

a) período entre pacotes: fixo de 40ms

EMULADOR:

a) taxa média de perdas = 30%

Medidas no RECEPTOR:

perdas[1]: 205

perdas[2]: 72

perdas[3]: 23

perdas[4]: 5

perdas[5]: 1

perdas[6]: 0

perdas[7]: 0

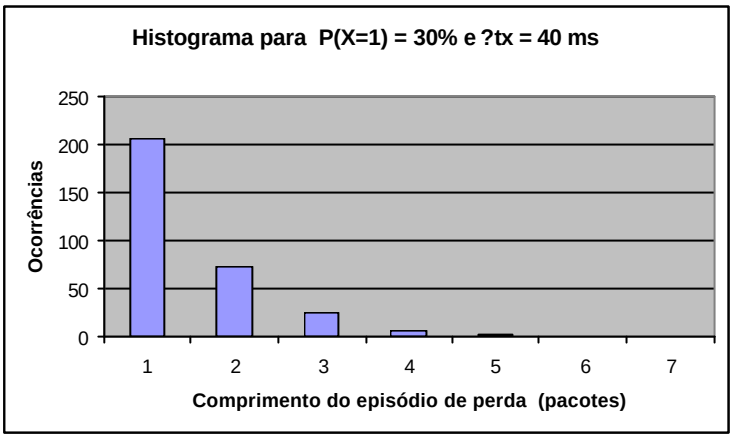


Figura 5-8 – Histograma para P(X=1) = 30% e ?tx = 40 ms

Configurações para o segundo teste:

TRANSMISSOR:

a) período entre pacotes: fixo de 10ms

EMULADOR:

a) taxa perdas = 30%

Medidas no RECEPTOR:

perdas[1]: 207

perdas[2]: 77

perdas[3]: 21

perdas[4]: 1

perdas[5]: 1

perdas[6]: 1

perdas[7]: 0

perdas[8]: 0

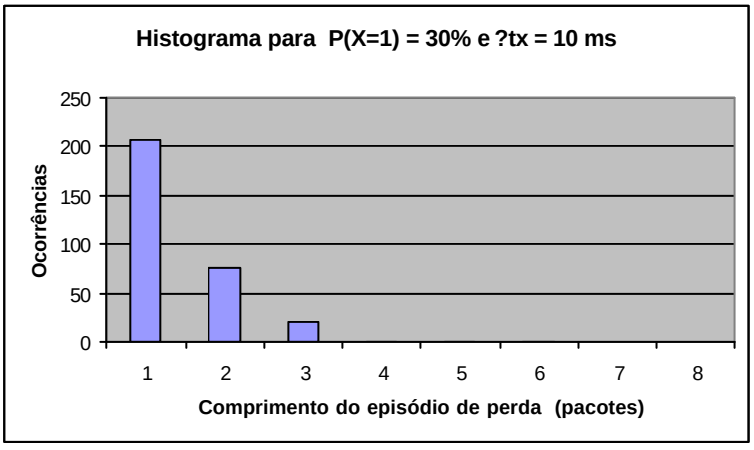


Figura 5-9 - Histograma para P(X=1) = 30% e ?tx = 10 ms

Com isto, temos a certeza de que o Emulador Paramétrico está introduzindo perdas de forma correta e com distribuição segundo o modelo de Gilbert. Deve-se, no entanto, ter em mente que para alguns dos pacotes de vídeo H.263 (os consecutivos que têm período menor que 10 ms) o emulador não terá a resolução necessária (a cada pacote) exigida pelo modelo matemático. Este fato ocorre por limitações de precisão de medidas do relógio de tempo real do sistema operacional.

5.3 Treinamento do Classificador Neuro-Fuzzy

Conforme definido pelos itens 4.3 e 4.4.1, as variáveis de QoS de rede a serem implementadas nesta versão do SCDM são: *jitter* e perdas. O transmissor irá transferir um vídeo, de aproximadamente 10 segundos e codificado em H.263, que atravessará o Emulador Paramétrico e será coletado pelo receptor.

Conforme o método de avaliação subjetiva da qualidade descrito no item 2.4, cada observador (de um grupo de 15) assiste e avalia um vídeo, relacionado a uma condição de rede que degrada o vídeo original. Ou seja, cada avaliador assiste às diversas amostras, com diferentes graus de degradação do mesmo vídeo original, emitindo seu julgamento. Ao final, computa-se a média as opiniões (MOS) dos 15 avaliadores, para cada situação que gerou a degradação sendo avaliada. Estes resultados estão computados na tabela F1, conforme o método em (ITU-T P.910, 1999). Vide também item 5.3.

A tabela 5.1 abaixo, uma versão simplificada da tabela F1, possui os dados a serem programados no Classificador Neuro-Fuzzy (estrutura ANFIS), e representa o comportamento que este deve apresentar, pois foi produzido a partir da média das notas dadas pelos avaliadores quando observaram cada um dos 25 vídeos que lhes foram apresentados (relativos a uma condição de *jitter* e perdas). O comportamento esperado também pode ser visualizado na superfície da figura 5.10. O classificador, ao ser programado, "aprende" esta superfície e repetirá as saídas, dadas as mesmas entradas. Para o caso de uma entrada (condição da rede) não ser exatamente igual aos dados originalmente programados, a estrutura fará uma interpolação e fornecerá uma saída (representando a MOS) correspondente.

	Jitter (ms)				
Perda (% tempo no estado RUIM)	0	20	40	60	80
0	4,466667	4,733333	4,066667	4,133333	4,2
20	3,333333	2,2	2,666667	3	3,4
40	2,2	2,666667	2,333333	1,933333	2
60	1,866667	1,266667	1,533333	1,2	1,333333
80	1,266667	1,6	1,266667	1,133333	1,266667

Tabela 5-1 – MOS *versus* condições da rede (*jitter* e perdas)

O Inspetor de Desempenho então, ao medir o estado atual da rede, poderá através do Classificador Neuro-Fuzzy (JANG, J.-S., 1995), avaliar a qualidade do vídeo que trafega na rede, substituindo a observação humana.

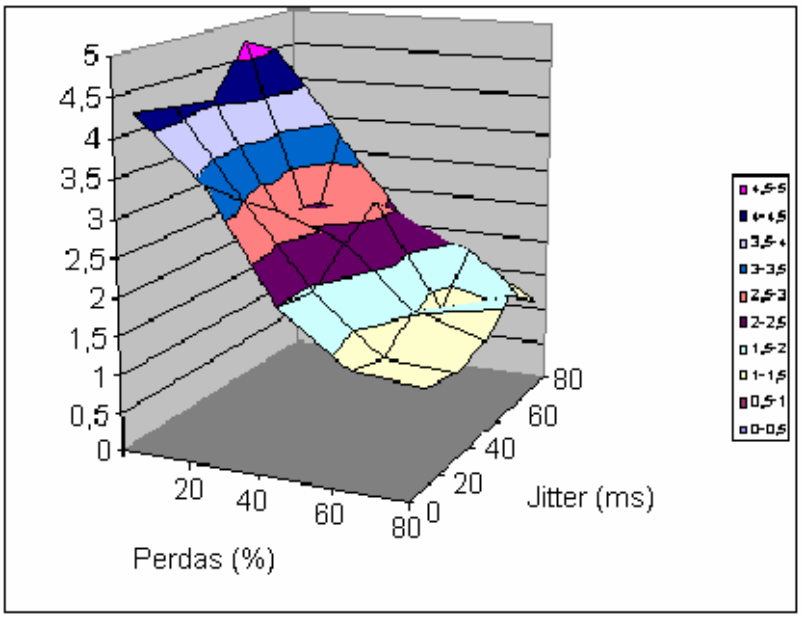


Figura 5-10 – Comportamento subjetivo MOS *versus* estado da rede

5.3.1 Definição das Funções de Pertinência do ANFIS

Para implementar o Classificador Neuro-Fuzzy foi escolhida a estrutura ANFIS, segundo o modelo de Sugeno descrito em B.4.1 (JANG, J.-S., 1995). A ferramenta utilizada foi o Matlab (MATLAB, 2004). A figura 5.11 ilustra um esquema da implementação ANFIS adotada. A estrutura neuro-fuzzy chama-se "Teste-1", possuindo duas variáveis de entrada - "Jitter" e "Perdas – e uma saída – "MOS".

A ferramenta Matlab possui um conjunto integrado de ferramentas para lógica Fuzzy (MATLAB, 2004). Conforme ilustrado nas figuras 5.12 e 5.13, escolhemos três funções triangulares para as variáveis de entrada (*jitter* e perdas) a fim de fuzzificá-las

como *jitter* alto, médio e baixo. A própria ferramenta escolheu os parâmetros de configuração para as funções de pertinência, durante o processo de treinamento. Como as variáveis de entrada variam linearmente por partes de 0 a 80, podemos observar nas figuras 5.12 e 5.13 o valor de 40 como sendo um ponto de inflexão.

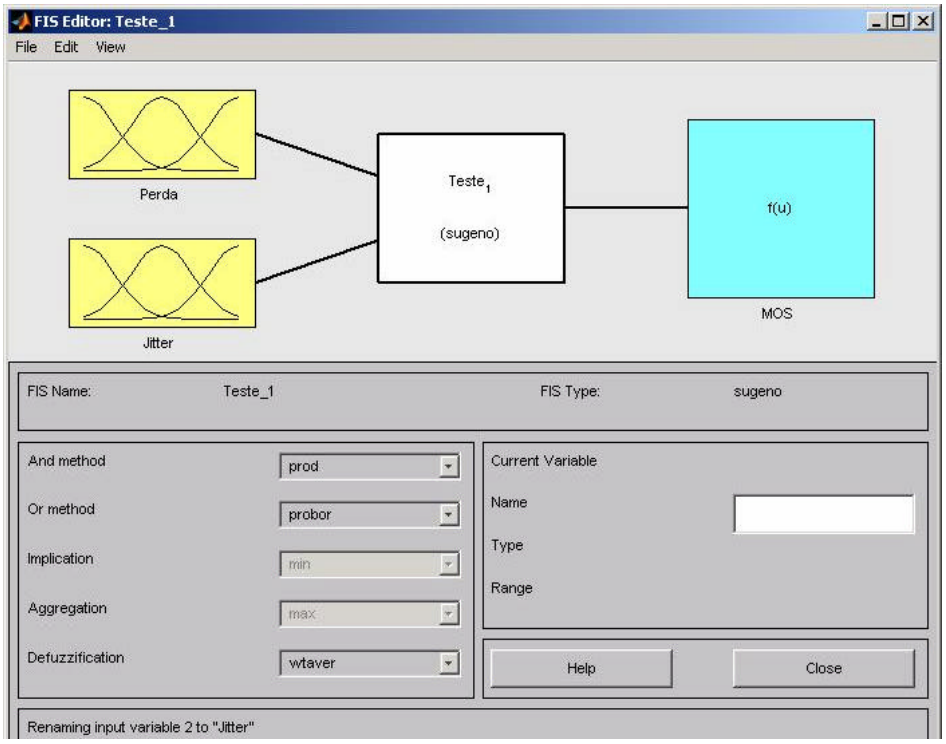


Figura 5-11 – Estrutura ANFIS adotada

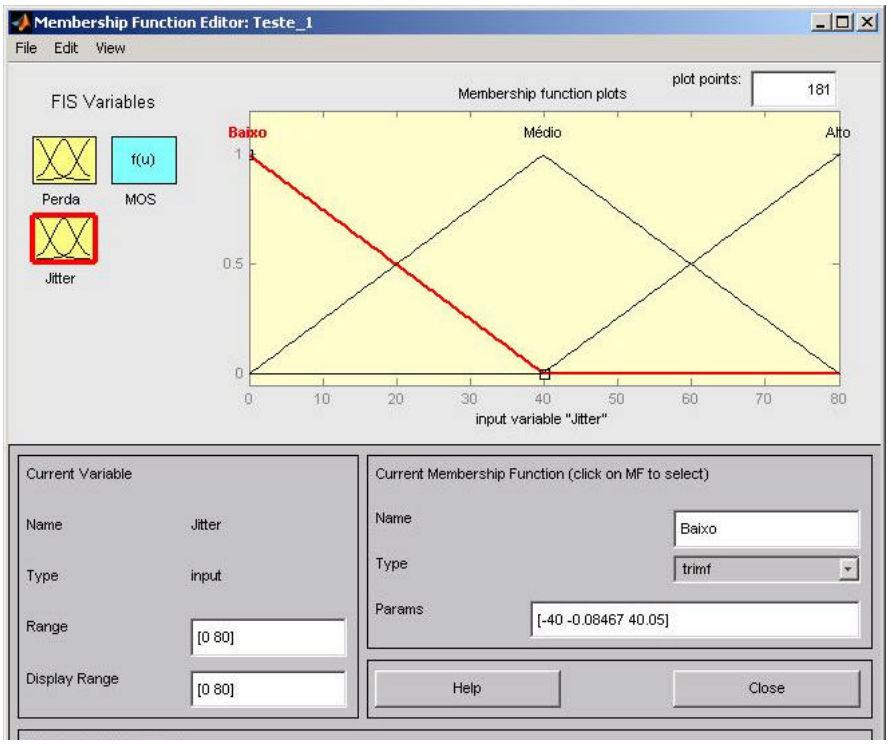


Figura 5-12 – Funções de Pertinência para a variável de entrada jitter

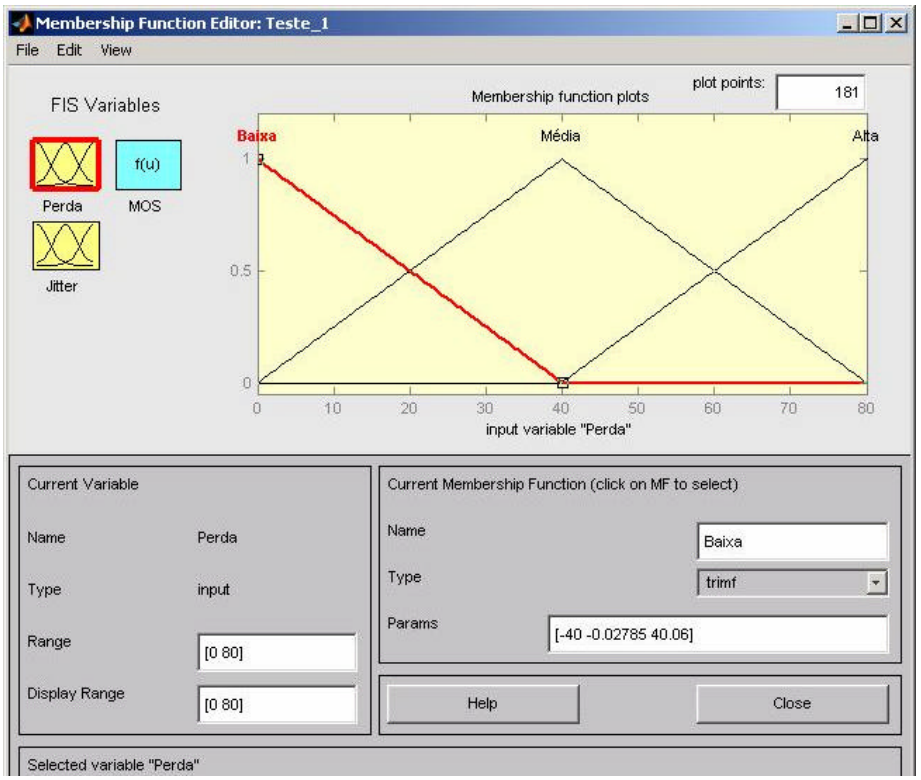


Figura 5-13 – Funções de Pertinência para a variável de entrada perdas

5.3.2 Arquitetura resultante do ANFIS

Com a definição das duas variáveis de entrada do sistema e das três funções de pertinência para cada entrada (por exemplo, *jitter* baixo, médio e alto), é possível a definição de nove regras. Estas dividem o espaço de entrada (as possibilidades das variáveis de entrada) em nove regiões fuzzy, como ilustrado na figura 5.14.

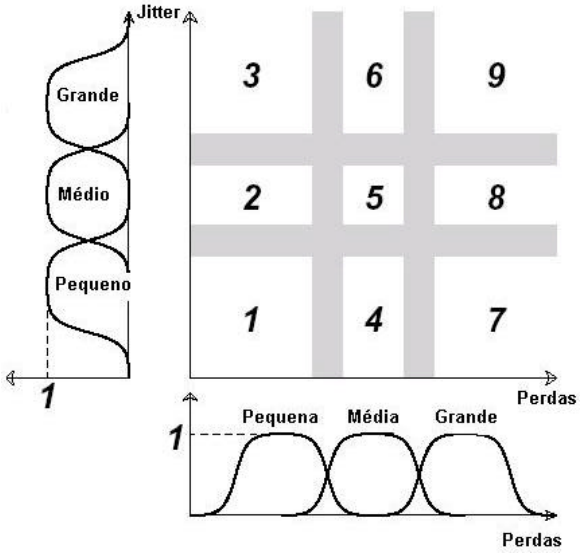


Figura 5-14 – Divisão do espaço de entrada em regiões fuzzy

A topologia resultante para as nove regras é uma extensão da figura B9 (com apenas duas regras) estando ilustrada na figura 5.15.

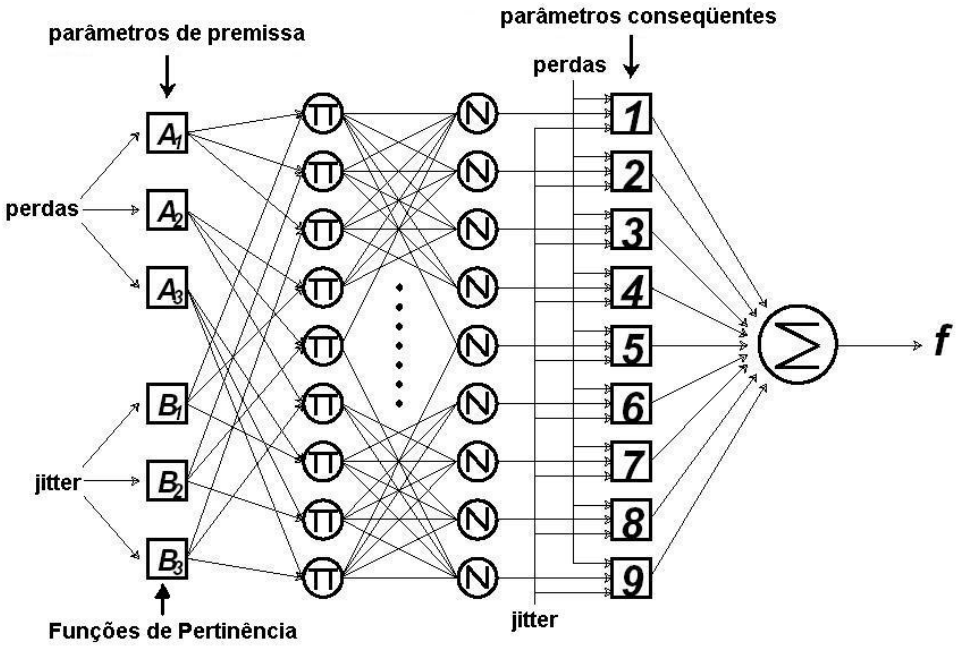


Figura 5-15 – Topologia ANFIS usada

5.3.3 Treinamento do ANFIS

Os parâmetros de premissa e os conseqüentes são ajustados, pelo processo de treinamento da rede neural contida no ANFIS. Este treinamento objetiva minimizar o erro ente os dados de entrada, contidos na tabela 5.1 ou na figura 5.10, e a saída MOS. Ou seja, após o processo de treinamento, quando o erro estiver próximo de zero (com uma precisão tolerável pela aplicação) pode-se dizer que a saída corresponderá aos dados de entrada. Como exemplo, se a perda for de 20% e o jitter for de 20%, a qualidade MOS deverá ser de 2,2 (vide tabela 5.1).

A figura 5.17 ilustra o processo de treinamento do ANFIS, feito pelas ferramentas contidas no MATLAB. A figura 5.18 ilustra uma ferramenta do MATLAB para a verificação da correta programação da rede neuro-fuzzy. Tem-se, como exemplo, as variáveis de entrada assumindo os valores de jitter = 23,23% e perdas = 20%. O classificador fornece como resultado uma qualidade subjetiva de 2,67.

Note-se que esses valores (2,2 e 2,67) diferem ligeiramente, isto é, o que fora programado originalmente e a qualidade MOS indicada pelo classificador, para as condições muito próximas de *jitter* e perda. Estas discrepâncias ocorrem devido às aproxi-

mações, não ótimas, entre os pontos de treinamento (25 – relativos às condições de rede utilizadas na tabela 5.1) e a superfície que efetivamente será usada, pelo classificador, para interpolá-los. Este problema pode ser visualizado através da figura 5.16: a ferramenta indicou que o número de pares de treinamento é menor que o número de parâmetros a serem ajustados no ANFIS (45 – vide item B5). Com isto, têm-se poucos pontos para ajustar um "grande" número de parâmetros, o que permite muitas soluções, inclusive as não ótimas. Se a questão fosse a inversa, ou seja, poucos parâmetros para ajustar uma superfície que deve passar por muitos pontos, o MATLAB precisaria de mais esforço computacional para satisfazer a todos os pontos. Em conseqüência, o treinamento da estrutura neuro-fuzzy seria mais eficiente, produzindo uma curva com melhor ajuste.

A fim de melhorar a precisão do classificador neuro-fuzzy, duas soluções são possíveis:

- aumentar o número de pares de treinamento (está relacionado ao escopo do problema, pois depende do número de estados de rede, e conseqüentemente do número de vídeos que os 15 observadores devem avaliar. Este número não pode aumentar muito, pois os observadores ficariam cansados, ao avaliar, digamos, 49 vídeos em seqüência). O número de pares de treinamento poderia aumentar para 36, no máximo, o que ainda não supera os 45 parâmetros. Esta solução não é desejável.
- otimizar a criação da estrutura neuro-fuzzy. Esta etapa fora procedida automaticamente pelo MATLAB, gerando nove regras e produzindo a estrutura da figura 5-15, com muitos parâmetros. Uma estrutura mais enxuta, parecida com a da figura B9, terá menos parâmetros e, conseqüentemente, seu treinamento com os mesmos 25 pares será mais preciso. Esta é a solução desejável para aumentar a precisão do classificador.

```
ANFIS info:
        Number of nodes: 35
        Number of linear parameters: 27
        Number of nonlinear parameters: 18
        Total number of parameters: 45
        Number of training data pairs: 25
        Number of checking data pairs: 0
        Number of fuzzy rules: 9

Warning: number of data is smaller than number of modifiable parameters.
```

Figura 5-16 - Informações sobre a estrutura ANFIS gerada

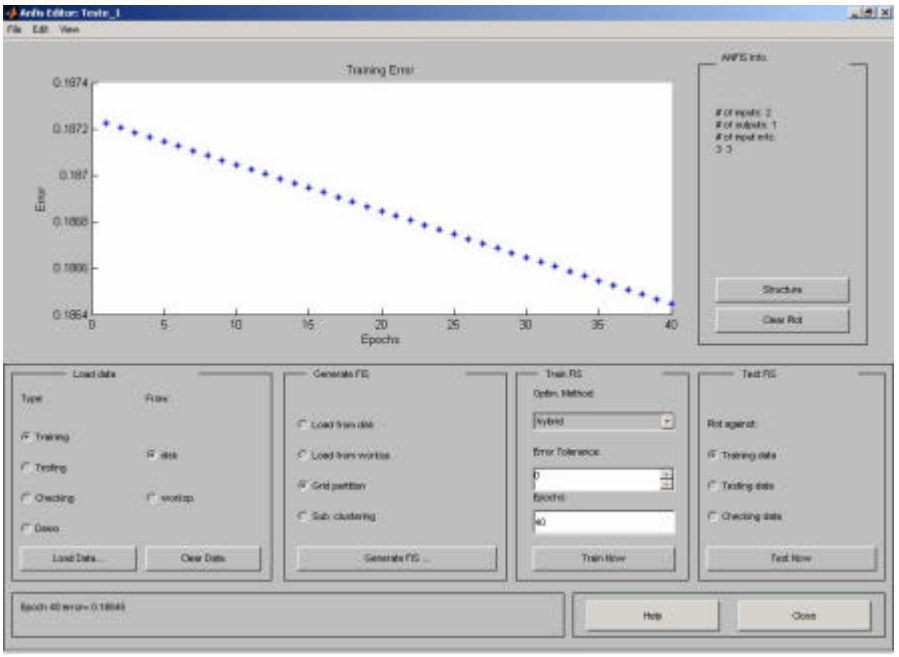


Figura 5-17 – Processo de treinamento do ANFIS

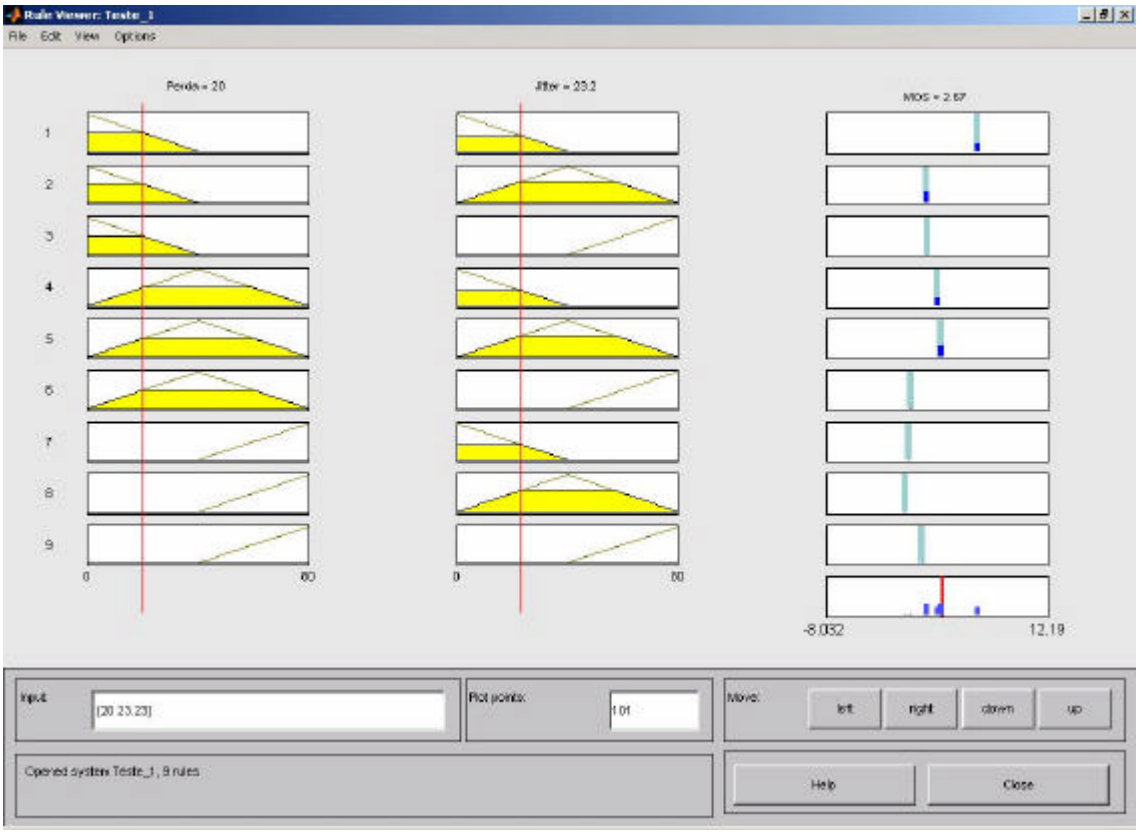


Figura 5-18 – Verificação do treinamento

5.4 Avaliação da qualidade pelo Classificador

No receptor, o Inspetor de Desempenho, pertencente à aplicação AVReceiver.java (versão original na seção "UI-Based Application for Reception of Media over RTP" das páginas de soluções do JMF, disponível na WEB pelo endereço: http://java.sun.com/products/java-media/jmf/2.1.1/solutions/ToolsRx.html), usa diversas informações disponíveis para alimentar o Classificador Neuro-Fuzzy (previamente programado conforme item 5.2) e obter a qualidade do vídeo recebido. Essas informações são as seguintes:

- provenientes dos Sender Reports (do RTCP, vide item A.3.1.2) e geradas pelo transmissor;
- presentes nos Receiver Reports (vide item A.3.1.1), gerados localmente no próprio receptor; e
- as Estatísticas da Conexão RTP, descritas no item 4.4.3.

Note-se, inicialmente, que as estatísticas acima obtidas são aqui tratadas a cada intervalo de reportagem (vide seção A.3.2), aproveitando o funcionamento natural dos protocolos RTP/RTCP. Caso seja necessária uma observação mais intensiva, o intervalo de reportagem pode ser alterado através da mudança dos parâmetros da sessão. Esta estratégia inclui também as estatísticas que são contínuas, isto é, não são zeradas a cada intervalo de reportagem, como por exemplo: o número de pacotes enviados pelo transmissor e os perdidos, através das variáveis *PktContIntervalo* e *PDUlostintervalo*, respectivamente, como pode-se observar extrato de código da aplicação, presente no Anexo F, item F3. Ou seja, por uma questão de coerência, computa todas as estatísticas referentes à mesma escala de tempo, que consiste no intervalo entre dois pacotes SR consecutivos.

A aplicação AVReceiver.java (presente no Anexo F, item F2), a cada intervalo de reportagem, computa a taxa média de perdas e o jitter médio, executa o programa Classificador.exe, escrito em "C", passando-lhe esses dois parâmetros, mais a estrutura Neuro_Fuzzy.fis (de Fuzzy Inference System) e recebe como retorno a qualidade subjetiva do vídeo, isto é, o MOS. A figura 5.19 ilustra três Sender Reports sendo recebidos (note-se que os intervalos de tempo entre suas chegadas não são constantes), o jitter e a perda sendo computados e a correspondente MOS retornada.

```
SR 21:11:17       jitter: 11.053993938371201
                  perda : 16.363636

                  MOS:    3.089000272505

SR 21:11:20       jitter: 11.773534787502209
                  perda : 26.666666

                  MOS:    2.537847653413
```

Figura 5-19 – Funcionamento do Inspetor de Desempenho

O arquivo Classificador.exe consiste de dois programas, fismain.c e fis.c que estão disponíveis no diretório "/fuzzy" das ferramentas do MATLAB (vide MATLAB, 2004). O programa *fismain.c* inicialmente recebe três parâmetros, todos do tipo string: o *jitter*, a perda e o nome do arquivo, *"Neuro_Fuzzy.fis"*. Este arquivo possui tamanho total de 2 Kbytes. Com este, é criada uma estrutura do tipo FIS e, em seguida, chama as funções de classificação neuro-fuzzy constantes no programa *fis.c*. A estrutura ANFIS criada, fora previamente treinada de acordo com os procedimentos no item 5.2.3.

Para o levantamento de um gráfico que demonstre o correto funcionamento do Inspetor de Desempenho, foram executadas as seguintes etapas:

1. o mesmo vídeo de teste é sempre enviado pelo transmissor;
2. os parâmetros correspondentes a UM estado da rede (no total de 25, conforme a tabela F1) são programados no Emulador Paramétrico;
3. para cada estado de rede (que é registrado na tabela de testes F1), um vídeo resultante destas degradações é salvo para ser avaliado posteriormente por observadores (em número de 15, nesta dissertação);
4. cada observador emite sua nota sobre a qualidade de cada um dos 25 vídeos a que assiste;
5. para cada uma dos 25 estados de rede, é computada a média das 15 notas referentes aos julgamentos dos observadores, obtendo-se o MOS para este estado de rede;
6. os dados presentes na tabela F.1 são usados para treinar o Classificador Neuro-Fuzzy; e

7. o Inspetor de Desempenho mede o estado presente da rede e consulta o Classificador que, estando previamente treinado, pode automaticamente avaliar o estado da rede, sem interferência humana.

Um extrato deste funcionamento é ilustrado na figura 5.20, enquanto que o gráfico na figura 5.21 ilustra a superfície MOS correspondente a todos os estados programados no Emulador Paramétrico.

Perda(%)	Jitter(ms)	MOS_MatLab	MOS_Inspetor_Desempenho
20	60	2,87	2.870399815373
10	40	3,1	3.099780990956
15	20	3,04	3.036341138584
5	23	3,87	3.870911046399
50	53	1,56	1.564508401849

Figura 5-20 – Extrato do funcionamento do Inspetor de Desempenho

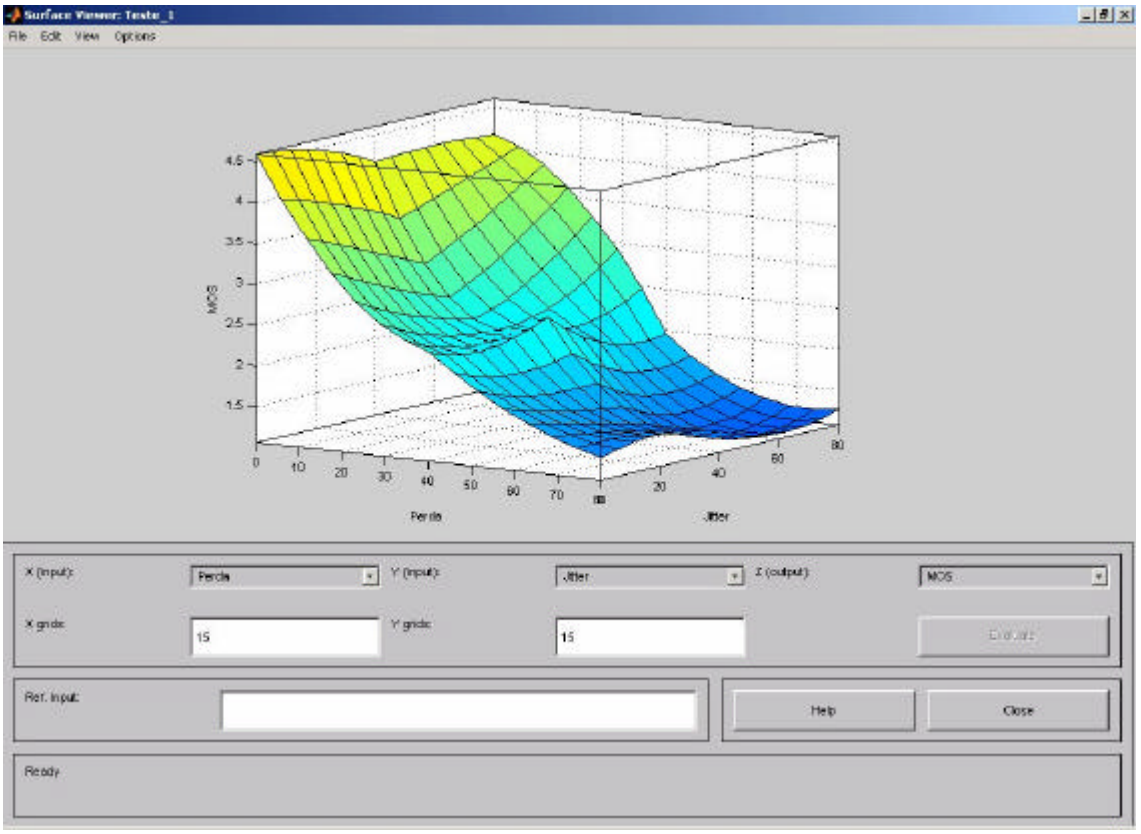


Figura 5-21 – Superfície MOS versus condições de rede gerada

Capítulo 6 Conclusões

Esta dissertação apresentou a concepção de um sistema de gerência distribuída e pró-ativa para a distribuição de vídeo de tempo real, que controla a admissão de novos fluxos e a QoS fim-a-fim de aplicações, segundo uma política preemptiva de prioridades. Ocorrendo falhas na distribuição, a gerência procura degradar graciosamente a qualidade dos fluxos de vídeo sendo distribuídos. Os sistemas de distribuição de vídeo e de gerência são construídos sobre uma infra-estrutura que utiliza componentes disponíveis comercialmente.

6.1 Sobre o trabalho realizado

O SCDM é um sistema cuja implementação é viável, usando-se tecnologias disponíveis comercialmente. Foram utilizados mecanismos e protocolos já consagrados, tais como o STP, OSPF, DiffServ e UDP/RTP.

A descrição e os requisitos que devem orientar a operação do SCDM foram elencados nos itens 3.1 e 3.2. As diretrizes do sistema e a definição de sua Gerencia foram especificados no item 3.3.

Conforme mencionado no Capítulo 4, uma parte do SCDM foi implementada: o Inspetor de Desempenho com Classificador Neuro-Fuzzy e um Emulador Paramétrico. Os modelos matemáticos usados na caracterização do *jitter* e das perdas de pacotes são adotados por outros autores. Uma adaptação foi necessária, devido à capacidade (precisão) de medidas do sistema operacional, que consistiu na troca do modelo exponencial (tempo contínuo) para o geométrico (eventos discretos). Verificamos também que o codec usado para a transmissão de mídia deverá contemplar perdas de pacotes seguidos, conforme o modelo simplificado de Gilbert indica.

A comprovação da aplicabilidade da metodologia de avaliação subjetiva da qualidade dos vídeos reside nos gráficos consistentes de MOS *versus* estado da rede. Esta informação permite ao Inspetor de Desempenho enviar alarmes se a MOS diminuir de um limite, determinado sempre pelo usuário final e não por parâmetros de rede, que não lhe traduzem um sentimento intuitivo de seu problema.

6.2 Sobre o trabalho que pode ser realizado

Como possibilidades de continuação desta dissertação, destacamos:

- melhorias no Inspetor de Desempenho;
- dimensionamento de uma infra-estrutura de rede IP capaz de suportar o trafego a ser distribuído pelo SCDM; e
- a implementação da Gerencia.

O Inspetor de Desempenho pode ser aperfeiçoado pelas seguintes linhas de ação:

- manter a estrutura Neuro-fuzzy.fis (tamanho de 2Kbytes) em memória, não sendo necessário passá-la como argumento a cada Sender Report, recebido por RTCP;
- otimizar a criação das regras de pertinência, o que facilitará o treinamento da estrutura neuro-fuzzy, conforme descrito no item 5.2.3;
- definir um limiar de MOS, sempre dependente da aplicação, para o envio de alarmes, permitindo à Gerencia tomar atitudes no sentido de prevenir a piora da QoS e posterior quebra da apresentação da mídia nos clientes;
- usar os "itens" no Buffer como parâmetro de entrada no Classificador Neuro-Fuzzy. A definição se "itens" refere-se a pacotes ou frames de vídeo depende da implementação do codec, que disponibilizará o acesso a estes itens, a fim de efetuar as medidas necessárias destas quantidades. O parâmetro "itens" no Buffer indica o estado futuro da apresentação da mídia nos clientes, pois mesmo ocorrendo uma piora da QoS, por uma certa quantidade de tempo, a apresentação nos clientes poderá ser mantida pelo consumo de seus buffers.

A Gerencia deve também ser tolerante a falhas, o que lhe indica uma característica distribuída, podendo-se usar a tecnologia peer-to-peer. Sugere-se a ferramenta JXTA, para esta tarefa. O item 3.3.2.3 contém alguns tópicos que deverão ser observados para a implementação de uma Gerencia Distribuída, enquanto que as características gerais a serem satisfeitas estão detalhadas nos itens 3.3.2.1 e 3.3.2.2.

Para o correto dimensionamento da rede IP capaz de suportar o trafego e que não seja demasiadamente onerosa, deve-se seguir algumas etapas:

- determinação dos tipos de mídia que serão distribuídos pelo SCDM;
- determinação dos fluxos de mídia que trafegarão simultaneamente pela rede;
- especificação do CODEC a ser utilizado para compressão de mídia, esquemas de correção de erros e recuperação de perdas de pacotes;
- modelagem estatística destes tipos de tráfego a fim de determinar-se a quantidade de trafego que entrara efetivamente na rede; e
- com o tráfego entrante caracterizado, é possível através de simulações, a determinação de uma topologia de rede onde haja disponibilidade dos recursos da rede, com uma probabilidade suficientemente elevada para as aplicações a serem suportadas pelo SCDM.

Anexo A Protocolos UDP e RTP/RTCP

A.1 UDP

O User Datagram Protocol (UDP) fornece um conjunto mínimo de extensões ao IP. O cabeçalho do UDP, mostrado na figura A1, compreende 64 bits que representam as portas fonte e destino, um campo do comprimento, e o *checksum*.

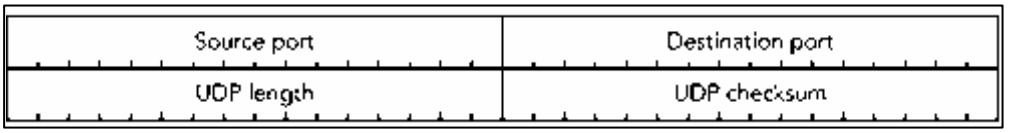


Figura A-1- Cabeçalho UDP

As portas fonte e destino identificam os destinos finais dentro dos computadores comunicando-se e permitem multiplexação de diversos serviços em portas diferentes. Alguns serviços funcionam em portas padronizadas, enquanto que outros usam uma porta definida dinamicamente, durante a negociação da sessão. O campo de comprimento é redundante com o aquele no encabeçamento do IP. O checksum é usado para detectar corrupção da carga útil (*payload*) e é opcional (é ajustada a zero para as aplicações que não o usam).

O uso das portas e do checksum, não garante nenhuma confiabilidade ao transporte dos dados (embora o checksum permita a detecção de erros na carga, que a camada IP não detecta), nem interfere na temporização de entrega do pacote. Uma aplicação que usa o UDP fornece pacotes de dados à camada de transporte, que os entrega a uma porta na máquina do destino (ou a um grupo das máquinas se o multicast for usado). Esses pacotes podem ser perdidos, atrasados, ou desordenados no trânsito, exatamente como observados para o serviço do IP.

A.2 RTP

O RTP (SCHULZRINNE, H. et al, 2003) executa muitas das tarefas atribuídas tipicamente a um protocolo da camada de transporte, contudo executa também algumas

tarefas da camada de sessão (controlando diferentes conexões de transporte e a identificação de um participante de uma forma abstraída de transporte) e da camada de apresentação (definindo representações padrão para mídias).

A figura A2 mostra o empacotamento da carga de mídia a ser transportada dentro de um pacote RTP, e este dentro de pacote UDP e IP.

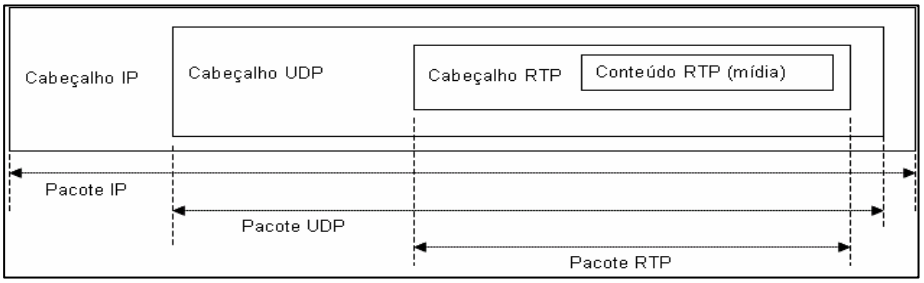


Figura A-2 - Encapsulamento da carga de mídia em pacotes RTP/UDP/IP

A.2.1 Sessões RTP

Sessão multimídia: um conjunto de sessões RTP simultâneas entre um grupo comum de participantes. Por exemplo, uma videoconferência pode conter uma sessão RTP de áudio e uma sessão RTP de vídeo.

Sessão RTP: uma associação entre um conjunto de participantes que se comunicam via RTP. Um participante pode estar envolvido em diversas sessões RTP ao mesmo tempo. Em uma sessão multimídia, cada tipo de mídia será normalmente transportado por uma sessão RTP separada, com seus próprios pacotes RTCP, a menos que a própria codificação componha diversas mídias em um único fluxo de dados.

Um participante distingue sessões RTP pelo uso de pares diferentes de endereços de transporte, cada um compreendendo um endereço de rede mais um par de portas para RTP e RTCP. Todos os participantes de uma sessão RTP podem compartilhar um mesmo par comum de endereço de transporte do destino, como no caso de uma transmissão multicast, ou os pares podem ser diferentes para cada participante, como no caso unicast.

A.2.2 O cabeçalho RTP

A figura A3 ilustra os campos do cabeçalho RTP. Os campos principais são explanados a seguir.

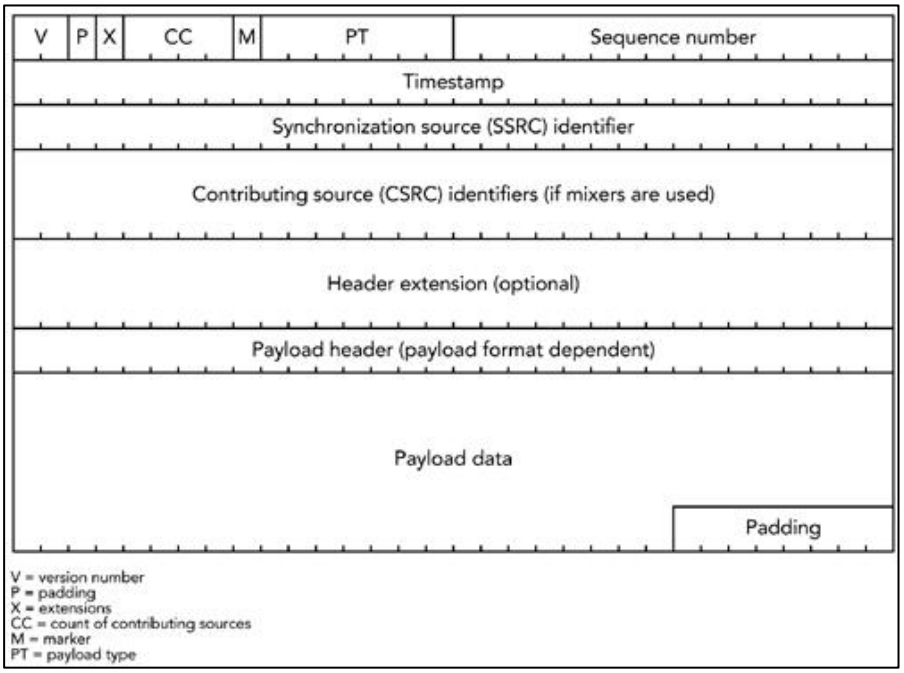


Figura A-3 – O Pacote RTP

PT (PAYLOAD TYPE) - identifica o tipo de mídia transportado por um pacote RTP. A aplicação na recepção examina o tipo de carga para determinar como tratar os dados -- por exemplo, passando-os por um decompressor particular. A interpretação do campo de *payload* relaciona-se a um perfil RTP (definido por uma RFC), que relaciona os números deste campo às especificações do formato da carga, conforme a tabela A1.

A escolha do formato da carga tem outras implicações: define o *clock* da mídia RTP, o formato do cabeçalho e do esquema de empacotamento da própria carga. Para atribuições estáticas, o clock é especificado no perfil (ou seja, na RFC correspondente). Por exemplo, para o formato de carga VIDEO/H263-1998 o clock é fixado em 90 khz.

Payload Type Number	Payload Format	Specification	Description
0	AUDIO/PCMU	RFC 1890	ITU G.711 μ-law audio
3	AUDIO/GSM	RFC 1890	GSM full-rate audio
8	AUDIO/PCMA	RFC 1890	ITU G.711 A-law audio
12	AUDIO/QCELP	RFC 2658	PureVoice QCELP audio
14	AUDIO/MPA	RFC 2250	MPEG audio (e.g., MP3)
26	VIDEO/JPEG	RFC 2435	Motion JPEG video
31	VIDEO/H261	RFC 2032	ITU H.261 video
32	VIDEO/MPV	RFC 2250	MPEG I/II video

Tabela A-1 - Alguns formatos de mídia transportados pelo RTP

SEQUENCE NUMBER - é usado identificar pacotes e fornecer uma indicação ao receptor se os pacotes estiverem sendo perdidos ou recebidos fora de ordem. O número de seqüência é um inteiro sem sinal de 16 bits, que é incrementado de uma unidade a cada pacote de dados enviado e retorna a zero quando o valor máximo for excedido. Uma conseqüência importante de o campo ter apenas 16 bits é que o retorno à zero do número de seqüência ocorre com relativa freqüência. Isto significa que as aplicações não devem confiar em números de seqüência como identificadores originais do pacote. Recomenda-se que seja usado um número de seqüência estendido de 32 bits, ou maior, para identificar os pacotes. Um exemplo: os 16 bits menos significativos sendo o número de seqüência do pacote RTP e os 16 bits superiores sendo uma contagem do número de vezes que a seqüência retornou a zero:

extended_seq_num = seq_num + (65536 * contagem_retorno_zero) .

TIMESTAMP - indica o instante de amostragem do primeiro octeto da mídia em um pacote, e é usado para agendar a apresentação da mídia no receptor. O *timestamp* é um inteiro sem sinal de 32 bits, que aumenta a uma taxa dependente da mídia e retorna a zero quando o valor máximo permitido no campo for excedido. Em *codecs* de vídeo típicos, com clock de 90 kHz, o retorno à zero ocorre aproximadamente a cada 13 horas; enquanto que com áudio de 8 kHz o intervalo é de aproximadamente 6 dias. O valor inicial do timestamp é escolhido aleatoriamente, ao invés de partindo de zero. Da mesma forma que o número de seqüência, esta precaução é usada para dificultar ataques em um fluxo criptografado.

O instante de amostragem deve ser derivado de um relógio de referência que seja incrementado linear e monotonicamente, a fim de permitir cálculos de sincronismo e jitter. A freqüência do relógio é dependente do formato da mídia transportada como carga e é especificada estaticamente no perfil ou no formato da carga (ou seja, na RFC correspondente). Estas exigências para o incremento do relógio da mídia não implicam necessariamente na mesma ordem de amostragem da mídia e o respectivo envio dos pacotes. Após a geração dos frames de mídia, conseqüentemente obtendo seus timestamps, estes podem ser reordenados antes do empacotamento. Como resultado, pacotes RTP consecutivos podem conter timestamps que não são crescem monotonicamente se seus dados não forem transmitidos na ordem em que foram amostrados, como no caso de vídeo MPEG interpolado (os números de seqüência dos pacotes transmitidos conti-

nuarão monotônicos). O receptor tem que reconstruir a ordem do timestamp para reproduzir a mídia originalmente transmitida.

Os timestamps em pacotes RTP não são necessariamente únicos dentro de um mesmo ciclo de retorno à zero. Se dois pacotes contiverem dados do mesmo instante de amostragem, terão o mesmo timestamp. A duplicação dos timestamps ocorre quando um frame de vídeo é dividido em múltiplos pacotes RTP para a transmissão (os pacotes terão números de seqüência diferentes, mas o mesmo timestamp).

SYNCHRONIZATION SOURCE (SSRC) - é um inteiro de 32 bits escolhido aleatoriamente pelos participantes quando se juntam à sessão. Tendo escolhido um SSRC, o participante usa-o nos pacotes que envia. Devido ao fato dos valores de SSRC serem escolhidos localmente, dois participantes podem selecionar o mesmo valor. Tal fato caracteriza uma colisão, que pode ser detectada quando uma aplicação recebe um pacote de outra que contem o mesmo SSRC escolhido por essa. Se um participante detectar uma colisão entre o SSRC que esteja usando e o escolhido por outro, deve enviar um pacote BYE (RTCP) para o participante com o SSRC original e escolher um outro SSRC. Este mecanismo de detecção de colisão garante que o SSRC seja único para cada participante dentro de uma mesma sessão.

CONTRIBUTING SOURCES (CSRCs) - sob circunstâncias normais, os fluxos RTP são gerados por uma única fonte, mas quando múltiplos fluxos RTP passam através de um misturador ou de um tradutor, o fluxo resultante terá diversas origens, que podem ter contribuído com o pacote RTP. A lista das fontes contribuintes (CSRCs) identifica os participantes que contribuíram com um pacote de RTP, mas não são responsáveis por sua temporização e sincronismo. Cada fonte contribuinte é um inteiro de 32 bits, correspondendo ao SSRC do participante que contribuiu com este pacote. O comprimento da lista de CSRCs é indicado pelo campo **CC** no cabeçalho RTP.

A.3 RTCP

O protocolo do controle do RTP (RTCP) executa quatro funções:

1- a função básica é fornecer informações sobre a qualidade da distribuição dos dados. A emissão de relatórios a todos os participantes permite o diagnóstico de problemas

como locais ou globais. Esta função é executada pelos relatórios do remetente (SR) e do receptor (RR) de RTCP, descritos adiante.

2- o RTCP carrega um identificador, persistente ao nível de transporte, para uma fonte de RTP chamado Nome Canônico (CNAME). Visto que o identificador SSRC pode mudar em caso de conflito ou se um programa for reiniciado, os receptores precisam do CNAME para cada manter informações de cada participante e para poder associar a este diversos fluxos, em várias sessões RTP relacionadas.

3- as primeiras duas funções requerem que todos os participantes enviem pacotes RTCP, portanto a freqüência de envio destes deve ser controlada para que o RTP seja escalável para um grande número de participantes. Como cada participante envia seus pacotes de controle a todos os outros, cada um pode independentemente observar o número total dos participantes. Este número é usado calcular a taxa em que os pacotes são enviados.

4- uma quarta função, que é opcional, é a de transmitir informações mínimas de controle da sessão, como por exemplo, a identificação dos participantes seja mostrada na interface do usuário.

A.3.1 Tipos de pacotes RTCP

- **SR (Sender Reports):** para estatísticas de transmissão e de recepção dos participantes que são emissores ativos
- **RR (Receiver Reports)**: para estatísticas da recepção dos participantes que não são emissores ativos e, em combinação com o SR, para os emissores ativos que relatam em mais de 31 fontes
- **SDES (Source Description items)**: informações que descrevem as fontes, incluindo CNAME.
- **BYE**: indicam o fim da participação de um membro
- **APP (Application-specific functions)**: definidos para uso experimental conforme novas aplicações e ferramentas forem sendo desenvolvidas

A.3.1.1 O pacote Receiver Report (RR)

Um pacote de relatório do receptor (RR) é identificado pelo valor 201 no campo PT do cabeçalho RTCP e tem o formato ilustrado em figura A4 abaixo. Um pacote RR contém o SSRC (fonte de sincronismo) do receptor que está gerando este relatório, seguido por zero ou mais blocos de relatório, indicados no campo RC, provenientes de outros participantes (*reportees*), visto que uma sessão RTP pode conter um transmissor e vários receptores e que todos estes recebem pacotes dos demais.

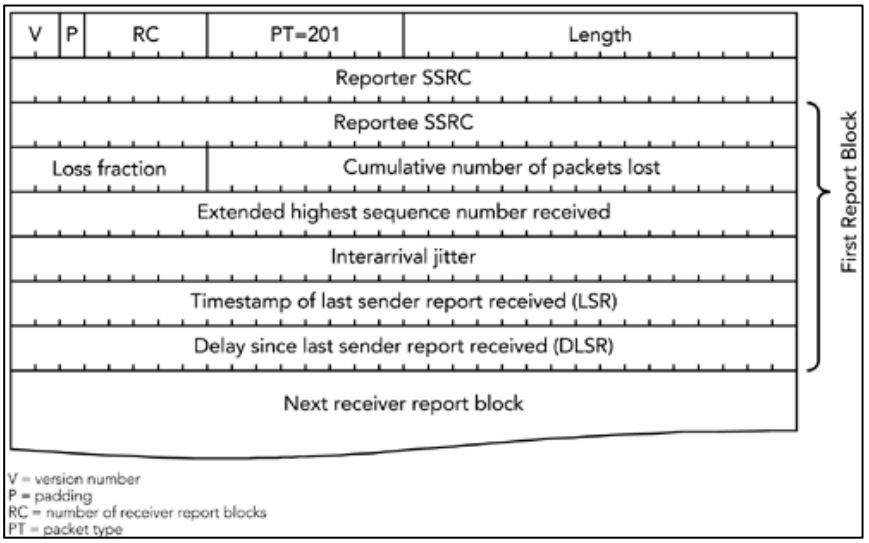


Figura A-4 - O pacote Receiver Report (RR)

Cada bloco do relatório descreve a qualidade da recepção de um único participante (*reportee*) que tenha recebido pacotes RTP durante o presente intervalo de reportagem. No máximo 31 blocos de relatório podem estar em cada pacote de RTCP RR. As estatísticas presentes num bloco de relatório referem-se à qualidade da recepção relativa à fonte de sincronismo (ou seja, de dados) deste *reportee*.

INTERARRIVAL JITTER - é uma estimativa da variância do tempo de trânsito na rede dos pacotes emitidos de uma fonte (SSRC) para um dado receptor (*reportee*). Este *jitter* é medido em unidades de timestamp, sendo expresso como um inteiro sem sinal de 32 bits, como o timestamp do RTP.

Para calcular-se a variância no tempo de trânsito na rede, são necessárias medidas deste tempo do trânsito. Devido ao fato de que o emissor e o receptor, não necessariamente, têm relógios sincronizados, é impossível medir o tempo absoluto de trânsito.

Conseqüentemente, calcula-se o tempo de trânsito relativo como sendo a diferença entre o timestamp RTP de um pacote e o relógio do receptor (em unidades de timestamp RTP) no instante da chegada, medido nas mesmas unidades. Devido à falta de sincronismo entre os relógios do emissor e o do receptor, o tempo relativo do trânsito inclui uma constante desconhecida. Isto não é um problema, visto que o interesse está somente na variação do tempo do trânsito.

O *jitter* entre pacotes J é definido como sendo o desvio médio da diferença D temporal entre dois pacotes no receptor em relação ao tempo quando foram enviados pelo emissor. Se Si for o timestamp RTP do pacote i, e Ri for o instante de chegada, em unidades de timestamp RTP, para o pacote i, então para os pacotes i e j, D pode ser expresso como:

$$D(i,j) = (Rj - Ri) - (Sj - Si) = (Rj - Sj) - (Ri - Si)$$

O *jitter* entre pacotes deve ser continuamente calculado à medida que cada pacote i é recebido, proveniente da fonte SSRC_n, computando-se a diferença D entre este pacote i e seu precedente i-1, na ordem em que chegaram (não necessariamente na mesma seqüência em que foram enviados), de acordo com a fórmula:

$$J(i) = J(i-1) + (|D(i-1,i)| - J(i-1))/16$$

Sempre que um pacote RR é enviado, o valor corrente de J será usado.

Precisão da medida do *jitter*:

Devido ao cálculo do *jitter* basear-se no timestamp RTP, que representa o instante de amostragem dos primeiros dados do pacote, qualquer variação entre os instantes de amostragem e o de transmissão efetiva do pacote, afetará o *jitter* resultante calculado pela fórmula acima.

Essa diferença temporal ocorrerá, por exemplo, em pacotes de áudio com duração variável. Ocorrerá também para os codecs de vídeo porque o timestamp é o mesmo para todos os pacotes de um mesmo frame, mas esses pacotes não são todos transmitidos ao mesmo tempo.

A variação do instante de transmissão reduz a precisão do cálculo do *jitter* como uma medida do comportamento da rede, mas seu uso é apropriado considerando-se que o *buffer* do receptor deve compensar esse fato. Quando o cálculo do *jitter* é usado como uma medida comparativa, o componente (aproximadamente constante) devido à varia-

ção do instante de transmissão é cancelado (do transmissor menos a do *buffer* do receptor) de modo que uma mudança no *jitter* da rede possa então ser observada.

LAST SENDER REPORT (LSR) - indica o tempo de recepção do último pacote SR (RTCP). É formado com os 32 bits mais significativos do campo NTP TIMESTAMP (64 bits) contido no último pacote SR recebido por este receptor (reportee), proveniente de sua fonte (SSRC). Se ainda nenhum SR for recebido, este campo é ajustado para zero.

DELAY SINCE LAST SENDER REPORT (DLSR) - é o tempo, expresso em unidades de 1/65.536 segundos, entre a recepção do último pacote SR proveniente da fonte (SSRC) deste receptor (*reportee*) e a emissão deste bloco de relatório de recepção. Se nenhum pacote SR foi recebido, o campo de DLSR será zerado.

Calculando o tempo de ida-e-volta na rede:

Um emissor pode usar os campos LSR e DLSR para calcular o tempo de ida e volta (RTT) entre ele e cada um dos receptores que recebam seus dados. Ao receber um pacote RR que lhe seja relacionado, o emissor subtrai o campo LSR do instante corrente, a fim de obter o atraso entre a emissão do SR e a recepção deste RR. O emissor então subtrai o campo DLSR, removendo o retardo introduzido pelo receptor, obtendo assim o tempo de ida e volta na rede. Note-se que o valor calculado para o tempo de ida e volta na rede exclui o processamento nos terminais (dado por DLSR), como por exemplo, o efeito da bufferização da mídia no receptor a fim de suavizar o *jitter*.

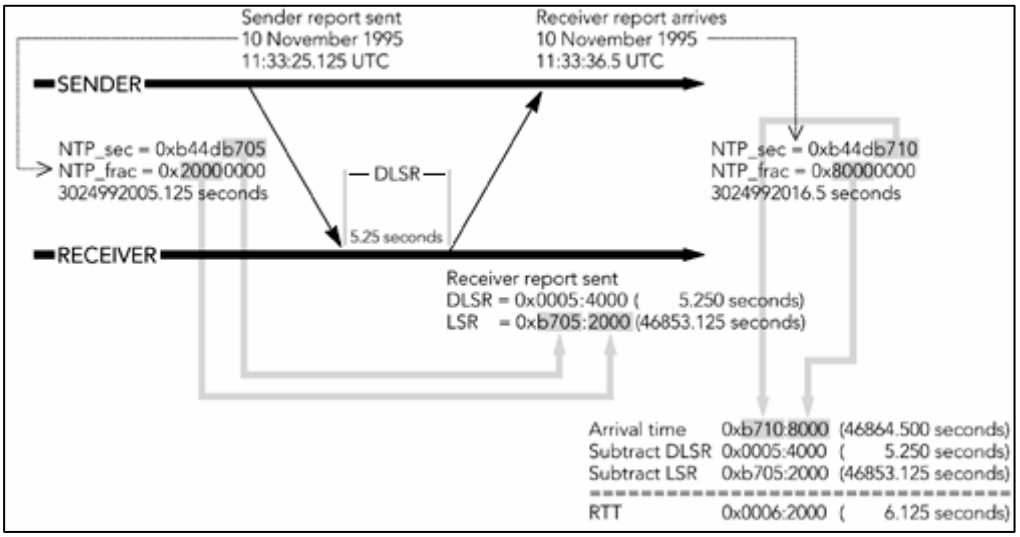


Figura A-5 - Cálculo do tempo de ida-e-volta

A.3.1.2 O pacote Sender Report (SR)

Um pacote de relatório do remetente é identificado por um valor no campo tipo de pacote de 200 e tem o formato ilustrado na figura A6 abaixo. Um SR contém informações do emissor em 24 octetos, seguido por zero ou mais blocos de relatório do receptor.

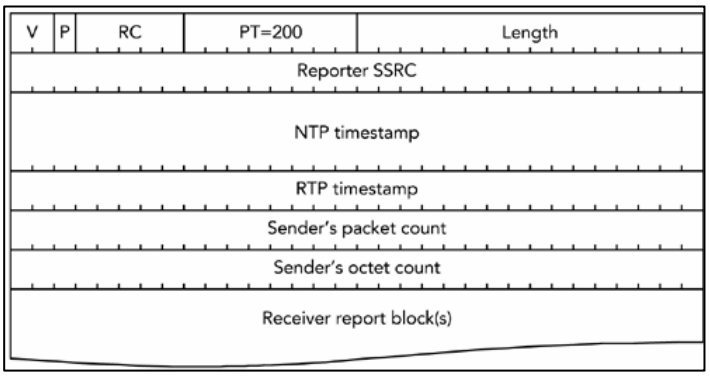


Figura A-6 - O pacote Sender Report (SR)

NTP TIMESTAMP - indica o tempo do relógio de referência quando este pacote SR foi enviado, de modo que possa ser usado em combinação com os timestamps retornados nos relatórios de recepção (RR) de outros receptores para medir a o tempo de propagação de ida e volta àqueles receptores.

O tempo do relógio de referência (data e hora absolutas) é representado usando o formato do Network Time Protocol (NTP), que está em segundos relativos à 0h UTC em 1º de janeiro de 1900. A resolução máxima do timestamp NTP é um número com ponto fixo sem sinal de 64 bits, com a parte inteira nos primeiros 32 bits e a parte fracionária nos últimos 32 bits. Em alguns campos onde é necessária uma representação mais compacta, apenas os 32 bits do meio são usados, isto é, os 16 bits inferiores da parte inteira e os 16 bits superiores da parte fracionária.

RTP TIMESTAMP - corresponde ao mesmo instante que o timestamp NTP, porém é expresso nas mesmas unidades e com o mesmo *offset* aleatório que o timestamp do pacote de dados (mídia via RTP). O valor deste campo geralmente não será o mesmo que o timestamp RTP do pacote de dados adjacente, devido ao offset aleatório entre os timestamps relativos à amostragem dos dados (RTP Timestamp) e o envio deste pacote SR (NTP Timestamp). Se as fontes estiverem sincronizadas pelo mesmo timestamp NTP, será possível o sincronismo inter e intra-mídia, usando-se este campo (do SR) e o

campo Timestamp dos pacotes de dados RTP.

A.3.1.3 Pacotes Compostos

Os pacotes RTCP nunca são emitidos individualmente, mas agrupados em um pacote composto para a transmissão, ilustrado na figura A6. Algumas regras governam a estrutura dos pacotes compostos:

- se o participante que gera o pacote composto for um emissor ativo dos dados, o composto deve começar com um pacote SR
- se não for emissor ativo, o composto deve começar com um pacote RR
- depois de SR/RR, o pacote deve ser um SDES. Este pacote deve incluir um item CNAME, e pode incluir outros itens.
- os pacotes BYE devem ser colocados como o último pacote do composto.
- outros pacotes RTCP a serem enviados podem ser incluídos em qualquer ordem.

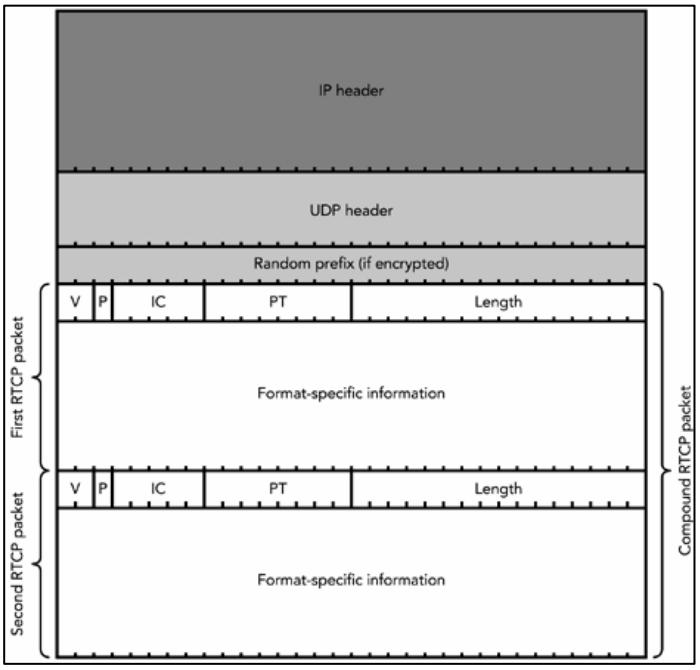


Figura A-7- Formato dos pacotes RTCP compostos

Todos os participantes em uma sessão devem enviar pacotes RTCP compostos e, conseqüentemente, receberão os pacotes RTCP compostos enviados por todos os restantes. Note-se que a realimentação é enviada a todos: tanto de forma unicast a um tradu-

tor, que então redistribuirá os dados, como de forma multicast.

A natureza peer-to-peer do RTCP permite a cada participante um conhecimento global da sessão: a presença dos participantes restantes, a qualidade da recepção, e opcionalmente, detalhes pessoais tais como o nome, o endereço do e-mail, a localização e o número de telefone.

A.3.2 Intervalo de Reportagem

Os pacotes compostos RTCP são enviados de forma aleatória. O tempo médio entre a emissão de pacotes RTCP é conhecido como o intervalo de reportagem. Este é calculado com base em diversos fatores:

- a largura de banda alocada ao RTCP. Esta é uma fração fixa - geralmente 5% - da largura de banda da sessão, que é definida como a taxa de bits de um único fluxo de mídia (áudio ou vídeo) multiplicado pelo número de emissores simultâneos. A largura de banda da sessão é fixa durante a sua duração, sendo fornecida como um parâmetro de configuração da aplicação RTP. Para maiores detalhes vide “Configurando uma Sessão RTP”, no item D.4 mais adiante;
- o tamanho médio dos pacotes RTCP enviados e recebidos; e
- o número total de participantes e da fração destes que são emissores. Isto requer que a implementação do RTP mantenha uma base de dados de todos os participantes, avaliando se são emissores (isto é, se pacotes de dados RTP ou os pacotes SR foram recebidos deles) ou receptores (se apenas pacotes RR, SDES ou APP foram recebidos).

Anexo B O Classificador Neuro-Fuzzy

Neste anexo, é apresentada uma breve introdução à lógica fuzzy e funções de pertinência. Não se pretende aprofundar em conceitos de conjuntos difusos, aritmética difusa, raciocínio difuso ou em modelos fuzzy. Para estes e outros tópicos, consultar (JANG, J.-S., 1995) como referência. O objetivo aqui é mostrar tópicos selecionados, relativos ao Classificador Neuro-Fuzzy adotado e sua implementação no MATLAB 7.0.

B.1 Por que usar a lógica Fuzzy?

Tem-se uma lista de características gerais sobre a lógica fuzzy:

- A lógica Fuzzy é conceitualmente fácil de compreender. Os conceitos matemáticos necessários ao raciocínio fuzzy são muito simples.
- A lógica Fuzzy é tolerante a dados imprecisos. Na maioria dos conceitos existe certa imprecisão associada.
- A lógica Fuzzy pode modelar funções não-lineares da complexidade arbitrária. Pode-se criar um sistema fuzzy para ser ajustado ao conjunto de dados de entrada e saída da função a ser modelada. Este processo é executado facilmente por técnicas adaptáveis, como os sistemas que estão disponíveis no ferramental do MATLAB.
- A lógica Fuzzy pode ser misturada com as técnicas convencionais de controle. Os sistemas Fuzzy não necessariamente substituem estes métodos, mas, em muitos casos, simplificam sua execução.
- A lógica Fuzzy pode ser construída sobre a experiência de peritos
- A lógica Fuzzy é baseada na língua natural. A base para a lógica fuzzy é a mesma para a comunicação humana.

A última observação é talvez a mais importante e merece mais discussão. A linguagem natural, usada pelas pessoas diariamente, foi formatada por milhares dos anos da história humana para ser conveniente e eficiente. As sentenças escritas desta forma representam um marco para uma comunicação eficiente.

B.2 Introdução à Lógica Fuzzy

Em oposição à lógica clássica de dois valores, a lógica nebulosa é multivalores, ou seja, atribui a uma afirmação não o valor 'verdadeiro' ou o 'falso', mas um grau de veracidade dentro de um intervalo numérico. Além disto, é possível um tratamento das implicações lógicas seguindo regras naturais de raciocínio, analisando condições e estipulando conseqüências (MEDEIROS, A. V. et al, 2001). Seu fundamento advém da teoria dos conjuntos nebulosos, que permite a manipulação de valores incertos, expressões verbais abstratas (e.g. pequeno, próximo, muito rápido, etc.).

No entanto, embora seja possível modelar fenômenos através da lógica *fuzzy*, é requerido um processamento computacional sobre esse modelo. É na etapa de "fuzzificação" ou "nebulização" que tais informações são convertidas em números *fuzzy* para então ocorrer a formulação e execução de uma estratégia de controle. As grandezas provenientes do domínio do mundo real (valores exatos ou *crisp*), captadas por sensores, dispositivos computadorizados ou mesmo provenientes de outros segmentos do processo de controle, sofrem essa conversão por meio da definição de um conjunto de variáveis nebulosas (funções lingüísticas de pertinência) que descrevem a entidade no domínio de abrangência. Normalmente utilizam-se mnemônicos para descrever essas variáveis, por exemplo: variação da pressão, erro, temperatura.

B.3 Funções de Pertinência

Uma função de pertinência (Membership Function - MF) é uma curva que define como cada ponto no espaço da entrada (exata ou *crisp*) é mapeado a um valor de pertinência (ou ao grau de pertinência) variando entre 0 e 1. O espaço da entrada é referenciado como o universo de discurso (MATLAB, 2004).

Um dos exemplos mais geralmente usados de um conjunto fuzzy é o de altura de pessoas. Neste caso, o universo de discurso consiste de todas as alturas potenciais, digamos de 0,7 a 2,50 metros, e a palavra alta corresponderia a uma curva que definisse o grau com que uma pessoa é alta. Se ao conjunto de pessoas altas for dado um limite bem definido (*crisp*), podemos dizer que, por exemplo, uma pessoa com mais de 1,90 metros será considerada alta.

A abordagem convencional, mostrada na figura B.1, faz uma distinção que apresenta um problema: o limiar de decisão de quando considerar uma pessoa é extremamente rígido. Digamos, pela figura B1, que esse limiar seja 1,90 metros. Então, enquanto alguém com 1,95m será considerado alto, outra pessoa com 1,89 metros não o será! A abordagem fuzzy contorna esta rigidez: a função de pertinência informaria "o quão alto" é uma pessoa, conforma a figura B2.

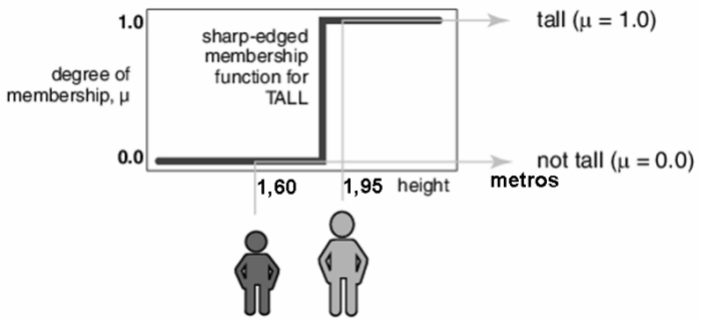


Figura B-1- Função de Pertinência convencional

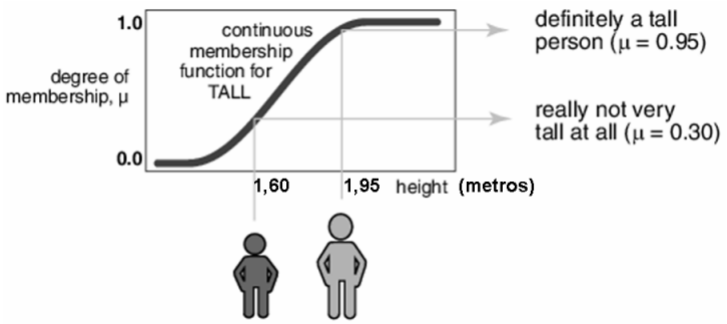


Figura B-2 - Função de Pertinência Fuzzy

B.4 Inferência Fuzzy

A inferência fuzzy é o processo de mapeamento de uma dada entrada do sistema a uma saída deste, usando a lógica fuzzy. Este mapa fornecerá uma base às decisões que podem ser feitas, ou os padrões que podem discernidos (ou seja, classificados).

O ferramental de lógica Fuzzy do MATLAB decompõe o processo de inferência fuzzy em cinco etapas: fuzzificação das variáveis da entrada, aplicação do operador fuzzy (E ou OU, dependendo da regra) no antecedente, implicação lógica do antecedente ao conseqüente (raciocínio ou inferência), agregação dos conseqüentes através de regras e desfuzzificação.

A figura B3 exemplifica um sistema de inferência fuzzy com duas entradas, três regras e uma saída. O fluxo de informação ocorre da esquerda para a direita, das duas

entradas para a saída. A execução paralela das regras é um dos aspectos mais importantes dos sistemas de lógica fuzzy.

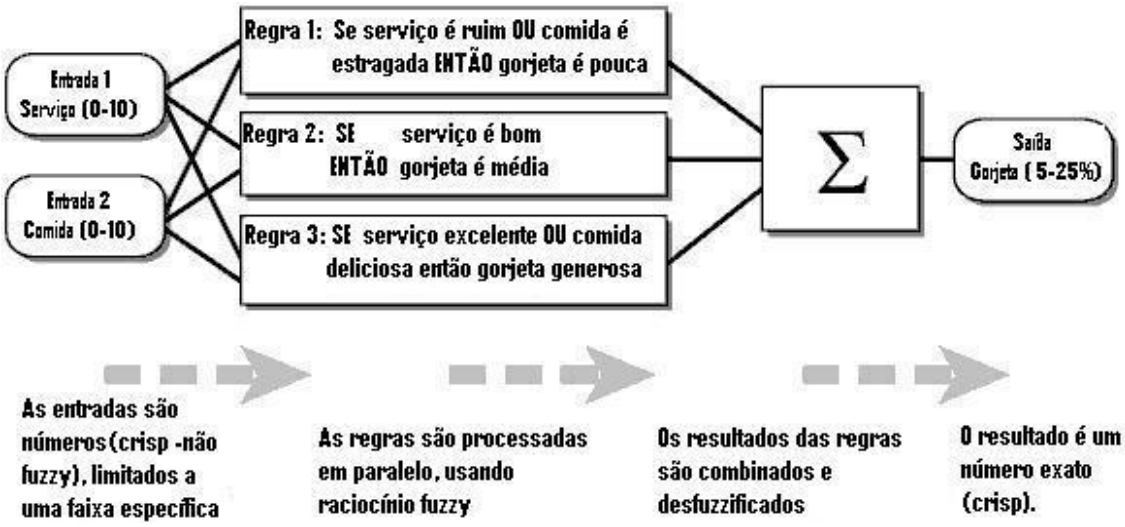


Figura B-3 - Exemplo de um sistema de inferência fuzzy

Etapa 1 - Fuzzificar entradas. Determinar para cada entrada, o grau de pertinência relativo a seu respectivo conjunto *fuzzy*, via funções de pertinência. Na figura B4, a entrada COMIDA é sempre um valor numérico (crisp) limitado ao universo do discurso da variável (neste caso o intervalo entre 0 e 10) e a saída é um grau de pertinência fuzzy no conjunto lingüístico qualificando (sempre o intervalo entre 0 e 1). Neste caso, o alimento é delicioso ao grau 0,7.

Figura B-4 - Fuzzificação de uma entrada

Etapa 2. Combinar entradas. A inferência convencional, com um antecedente e um conseqüente, é ilustrada abaixo:

premissa 1 (fato) : x é A (a entrada x pertence ao conjunto A)
premissa 2 (regra): SE x é A ENTÃO y é B,
conseqüência : y é B (conclusão) (Equação B1)

Se o antecedente de uma regra tiver mais de um fator, conforma a figura B5, o operador fuzzy está aplicado para obter um número que representa o resultado do antecedente combinado para esta regra.

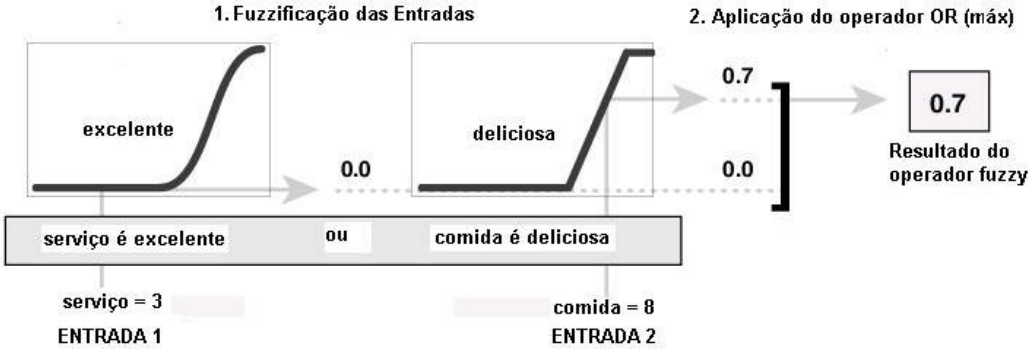


Figura B-5 - Combinação de duas entradas em uma regra do modelo de inferência

Etapa 3. Aplicar a implicação lógica. Esta etapa implementa o raciocínio (inferência) fuzzy. A inferência fuzzy ocorre pela composição máx-min dos valores retornados pelas funções de pertinência. Este processo pode ser implementado pelas funções OR-AND. Um conseqüente é um conjunto fuzzy, representado por uma função de pertinência, que pondera apropriadamente as características lingüísticas (entradas) que lhe são atribuídas, conforme a figura B6. A entrada para o processo da implicação é um número dado pelo antecedente, e a saída é um conjunto fuzzy. A implicação é executada para cada regra do modelo.

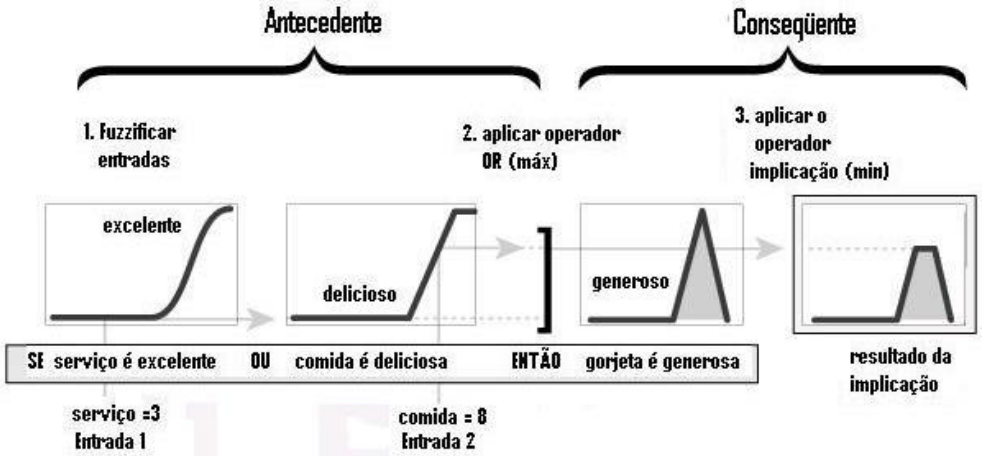


Figura B-6 - Processo de Implicação Fuzzy (modelo de Mandami)

Para a apresentação das duas últimas etapas (agregação dos conseqüentes e desfuzzificação) faz-se necessária uma diferenciação entre dois modelos de inferência fuzzy: Mandami e Sugeno.

B.4.1 Inferência com o modelo de Sugeno

A figura B6 ilustra um modelo fuzzy onde o resultado da implicação é também um conjunto fuzzy. Este é o modelo de Mandami. A diferença principal entre Mamdani e Sugeno é que as funções de pertinência das saídas de Sugeno são lineares ou constantes. Uma regra típica em um modelo de Sugeno tem o formato geral:

SE entrada_1 = x E entrada_2 = y ENTÃO saída z = ax + by + c. (Equação B2)

Este modelo permite a simplificação das duas últimas etapas da inferência *fuzzy* - agregação dos conseqüentes e desfuzzificação – para uma simples média.

A figura B7 ilustra todo o processo de inferência fuzzy, adotando-se o modelo de Sugeno de ordem zero, visto que as saídas constituem-se apenas das constantes "c" da equação B2.

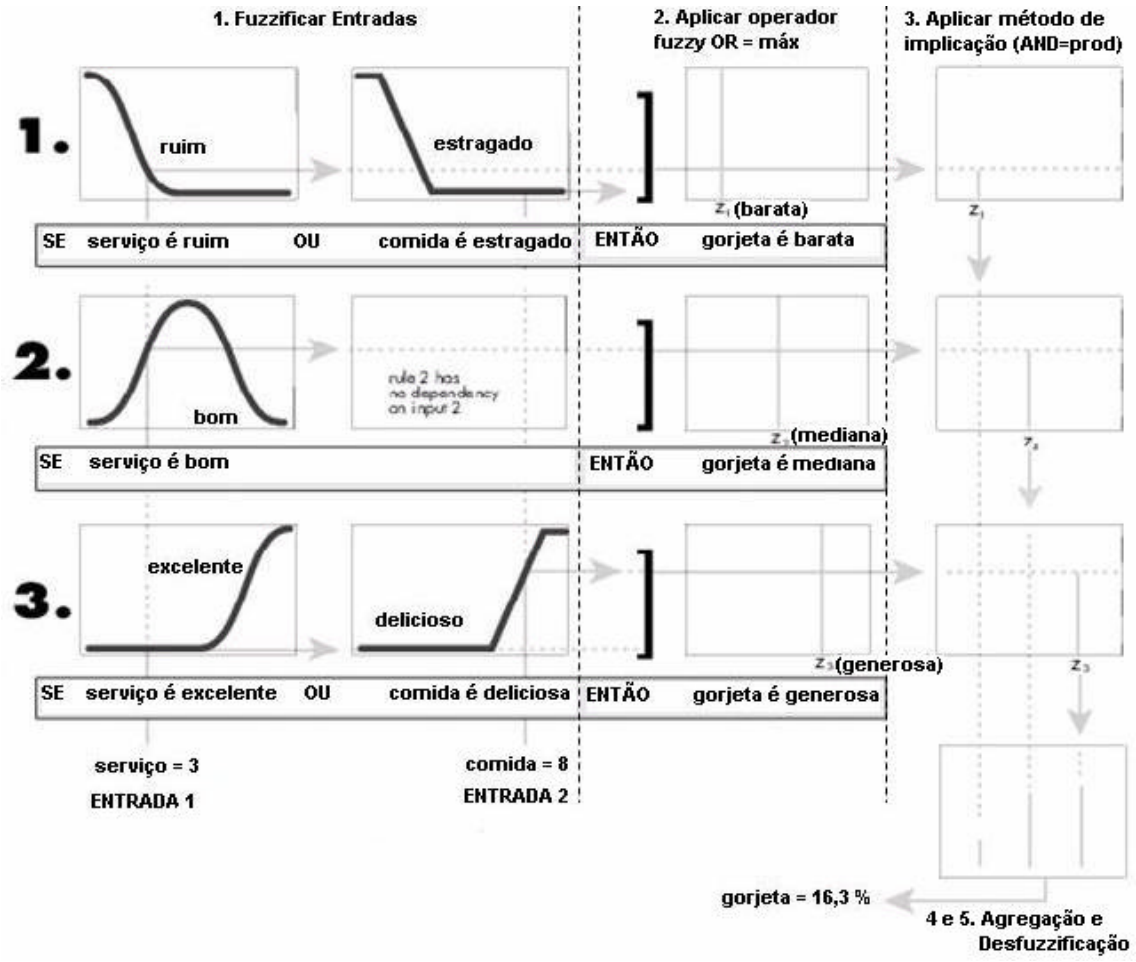


Figura B-7 - Sistema de inferência fuzzy de Sugeno de ordem zero.

B.5 ANFIS

ANFIS é um sistema adaptável de inferência neuro-fuzzy, no qual os parâmetros das funções de pertinência podem ser aprendidos, através de treinamento. Este treinamento dar-se-á através de um algoritmo tipo *backpropagation* sozinho, ou em combinação com um método tipo mínimos quadrados. Isto permite que o mapeamento fuzzy que o ANFIS implementa aprenda os dados que estão sendo modelados. O sistema ANFIS implementa o modelo de inferência fuzzy de Sugeno de primeira ordem.

Vamos considerar, por simplicidade, um sistema com duas entradas *x* e *y* e uma saída *f*. Para o modelo de Sugeno de primeira ordem temos:

Regra 1: SE x é A_1 E y é B_1 , ENTÃO $f_1 = p_1.x + q_1.y + r_1$

Regra 2: SE x é A_2 E y é B_2 , ENTÃO $f_2 = p_2.x + q_2.y + r_2$.

A figura B8 ilustra o mecanismo de raciocínio para este modelo. A arquitetura ANFIS equivalente, que implementa esse modelo está ilustrada na figura B9, onde os nós de uma mesma camada têm funções similares, descritas abaixo.

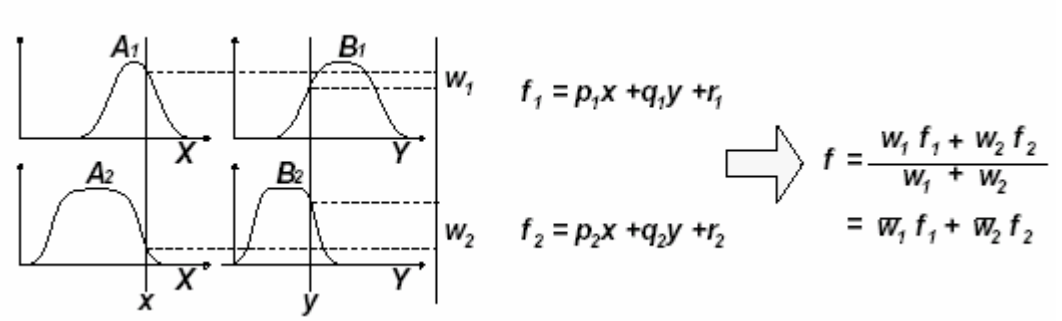


Figura B-8 - Modelo de Sugeno de primeira ordem

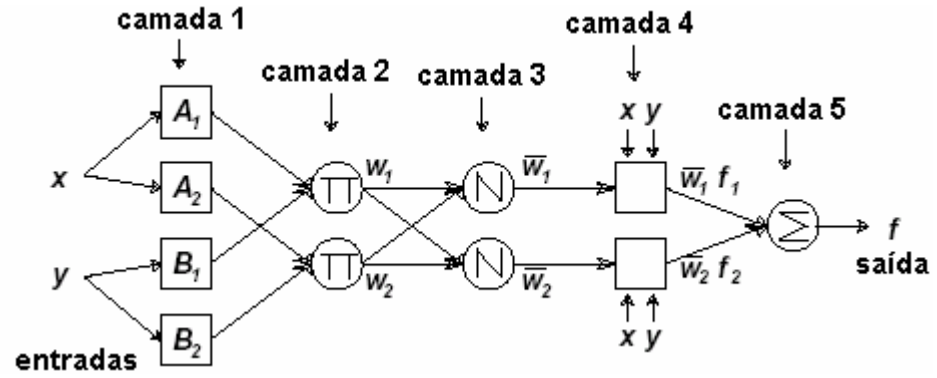


Figura B-9 – Arquitetura ANFIS

Camada 1: os nós desta camada são adaptativos e têm a seguinte forma:

$$O_{1,i} = \mu_{A_i}(x), \quad \text{para } i = 1, 2, \text{ ou}$$
$$O_{1,i} = \mu_{B_{i-2}}(y), \quad \text{para } i = 3, 4.$$

onde x e y são as entradas e A_i e B_{i-2} são as funções de pertinência associadas aos nós, ou seja, esta camada fuzzifica as entradas. Os parâmetros das funções de pertinência (referenciados como parâmetros de premissa) podem ser ajustados, pelo processo de treinamento.

Camada 2: nós não são ajustáveis que multiplicam, através do operador fuzzy AND, os sinais de entrada. A saída aqui representa o peso de cada regra.

Camada 3: calcula o peso de cada regra, mas normalizado-o em relação ao peso total.

Camada 4: os nós são adaptativos, com a seguinte função:

$$O_{4,i} = \overline{w}_i f_i = \overline{w}_i (p_i x + q_i y + r_i)$$

onde os parâmetros p, q e r (os conseqüentes) são ajustáveis.

Camada 5: calcula a soma dos sinais de entrada, computando a saída *f*.

O sistema ANFIS pode aprender um mapeamento entre as entradas e a saída (como visto no item 5.2) através do ajuste dos parâmetros (treinamento) de premissa e conseqüentes, das camadas 1 e 4 respectivamente.

Anexo C Vídeo H.263 transmitido sobre RTP

A seguir será feita uma rápida caracterização de vídeo digital, sua codificação em H.263 e sua transmissão sobre o RTP. Após, procura-se caracterizar estatisticamente o tráfego a ser transmitido usando a distribuição do tamanho dos pacotes e do tempo entre estes.

C.1 Princípios de Vídeo Digital

Vídeo digital consiste de frames de vídeo (imagens) que são apresentadas a uma taxa definida. Como exemplo, a taxa de 30 frames/segundo é usada pelo padrão NTSC. Cada frame de vídeo consiste de elementos de imagens (*pixels* ou *pels*). O formato do frame especifica o tamanho dos frames individuais em termos de pixels. O formato ITU-R/CCIR-601 (o formato comum da TV) tem 720×480 pixels, enquanto o formato CIF tem 352×288 e QCIF tem 176×144 pixels. Os formatos CIF e QCIF são normalmente usados nos estudos relacionados a redes. Cada pixel é representado por três componentes: a componente de luminância (Y) e duas componentes de crominância, HUE (U) e intensidade (V). Uma representação alternativa é a RGB, que pode ser convertida de (e para) YUV com o uso de uma matriz fixa de conversão.

Desde que o sistema visual humano é menos sensível à informação de cor do que à informação do luminância, as componentes de crominância são sub-amostradas numa razão de um conjunto de U e V para cada quatro amostras de Y (esquema 4:1:1). Logo, para cada imagem no padrão CIF, existem 352×288 amostras de Y, 176×144 de U e 176×144 amostras de V, totalizando 152.064 amostras. Como cada amostra é tipicamente quantizada em 8 bits, tem-se 152.064 bytes para cada frame CIF de vídeo não comprimido, correspondendo a uma taxa de bits de 36.5 Mbps (SEELING, P., 2004).

C.2 H.263 sobre RTP

Para transmitir-se fluxos de vídeo codificados em H.263 para a Internet, a saída do codificador pode ser diretamente empacotada por RTP. Para cada frame de vídeo, o

fluxo de bits em H.263 pode ser transportado pela carga útil (*payload*) RTP sem alterações, incluindo o código de início da figura, o cabeçalho inteiro da figura, além de quaisquer códigos de comprimento fixo ou variável (BORMANN, C. et al, 1988).

O RTP não garante um serviço confiável de entrega nem da ordem dos dados, assim um pacote pode sofrer perdas na rede. Para conseguir a melhor recuperação possível para estas perdas, o decodificador necessita processar outros pacotes que estão sendo recebidos.

Logo, é desejável o processamento independente de cada pacote em relação aos demais. Alguma informação ao nível de frame é incluída em cada pacote, como o formato da fonte e *flags* para que características opcionais ajudem ao decodificador operar correta e eficientemente na presença de perdas de pacotes. Estas *flags* fornecem também informações sobre as opções de codificação usadas nos fluxos de vídeo, podendo ser usadas por ferramentas de gerência da sessão.

C.2.1 Cabeçalhos H.263

Para os fluxos de vídeo H.263, cada pacote RTP carrega somente um pacote de vídeo H.263 (o que não ocorre com áudio, por exemplo, onde 160 amostras estão contidas em um único pacote RTP). O cabeçalho da carga útil estará sempre presente em cada pacote de vídeo. Três formatos (modos "A", "B" e "C") são definidos para o cabeçalho da carga H.263. No modo "A", um cabeçalho de quatro bytes estará presente antes da carga real de vídeo comprimido. Este modo permite a fragmentação ao nível de GOB. No modo "B", um cabeçalho de oito bytes H.263 é usado e cada pacote começa nos limites de um MB, sem a opção dos PB-*frames*. Finalmente, um cabeçalho de doze bytes é definido no modo "C" para suportar a fragmentação ao nível de MB, para os *frames* que são codificados com a opção PB-*frames*.

C.2.2 Seleção de modo para o cabeçalho H.263

É possível a mistura de pacotes que transportam fluxos de vídeo H.263 contendo modos diferentes. Os modos devem ser cautelosamente selecionados, com base na MTU da rede, nas opções de codificação H.263 e dos protocolos das camadas inferiores de rede.

O modo “A” deve ser usado para os pacotes que começam em um GOB ou com o código de início de figura, enquanto que os modos “B” ou “C” devem ser usados sempre que um pacote deve começar no limite do MB. Estes modos são necessários para aqueles GOBs cujos tamanhos são maiores do que o MTU da rede. Recomenda-se o uso do modo “A” sempre que possível. A vantagem principal deste modo sobre os “B” e “C” é simplicidade da primeira.

Para o caso dos vídeos usados neste trabalho e o codec H.263 implementado no JMF, o modo “C” com cabeçalho de 12 bytes sempre foi selecionado.

C.3 Caracterização estatística do tráfego H.263

Espera-se que as aplicações utilizando vídeo codificado MPEG-4 e H.263 venham contribuir com uma grande parcela do tráfego em redes infra-estruturadas e wireless. Nesta dissertação, adotou-se o H.263 como codificador para os vídeos a serem transmitidos, devido às características de implementação do JMF.

C.3.1 Parâmetros de codificação para H.263

Os autores de (FITZEK, F.H.P., 2001) apresentaram e estudaram uma biblioteca publicamente disponível de vídeos codificados em H.263 e MPEG-4. Os *traces* de tamanho do *frame* foram gerados para as codificações abaixo, sobre 10 seqüências de vídeo com 60 minutos cada.

O codificador foi programado para comprimir o formato de vídeo QCIF (176× 144 pel) com uma taxa de *frames* de 25 f/s. Logo, usando-se este padrão e o esquema 4:1:1 de amostragem temos: 176x144 (=25344) amostras de Y, 88x72 amostras de U e de V (totalizando 12.672 amostras de crominância). Com 8 bits por amostra, cada frame QCIF tem 38.016 bytes e o fluxo não comprimido é de 7.603.200 bits/s.

Foram também desabilitadas as seguintes opções negociadas[1] de codificação: vetores irrestritos do movimento, codificação aritmética com base na sintaxe e predição avançada.

Nota 1: Além ao algoritmo de codificação básico, quatro opções negociadas são incluídas no padrão H.263 para otimizar o desempenho: vetores irrestritos do movimento, codificação aritmética com base na sintaxe, predição avançada e *frames* tipo PB. Todas estas opções podem ser usadas juntas ou separadamente, vide (ITU-T H.263, 1996).

Esta decisão teve como base a conclusão, por parte desses autores, de que estas características trouxeram somente pouca melhoria na qualidade vídeo, ao retardar dramaticamente o codificador. Foram permitimos frames do tipo PB. Cada vídeo foi codificado em quatro diferentes taxas de bits: 16 Kbps, 64 Kbps, 256 Kbps, e taxa variável (VBR) (isto é, sem ajustar uma taxa de bits alvo).

C.3.2 Análise estatística dos *traces* H.263

A tabela C1 ilustra as primeiras 10 linhas do *trace* do filme "Silêncio dos Inocentes", que foi codificado com uma taxa de bits alvo de 256 Kbps. Ela ilustra na linha *n*, *n* = 1,…, N, os seguintes dados para o frame *n*: o tempo cumulativo T*n*-1 (até o frame *n* - 1) de sua exposição, seu tipo (I, P ou PB) e seu tamanho X*n* em bytes.

Como ilustrado por cada linha do *trace*, os T*n* são múltiplos inteiros do período básico de frame, ?=40 ms (=1/25 fps), do codificador H.263 usado. Note-se, entretanto que, alguns frames estão "saltados" pois o codificador tenta manter a taxa alvo de bits. Isto resulta em períodos variáveis do frame.

0	I	12539
360	P	3981
600	PB	6203
760	PB	5884
1000	PB	6749
1160	PB	6425
1400	PB	7849
1640	PB	5983
1800	PB	6183
2040	PB	7052

Tabela C-1 – Extrato de trace H.263 para o filme "Silêncio dos Inocentes"

A figura C1 dá as funções de densidade de probabilidade $P(tn = l.?)$, l=1, 2, …, dos períodos de frame para três *traces* H.263 gerados. Observa-se nos gráficos em (b) e (c), a tendência geral de que menores taxas alvo de bits resultam em períodos maiores do frame, isto é, mais frames são "saltados" a fim de manter a taxa final de saída, o que confirma o funcionamento esperado do codificador H.263.

De outra forma, para as codificações em VBR (isto é, sem uma taxa alvo de bits especificada) o H.263 tipicamente não "salta" nenhum frame. Não obstante, como ob-

servado na figura C1.a, a maioria dos *frames* codificados têm um período de frame de 2? (lembrar que ?=40 é o período básico). Isto ocorre porque o codificador produz principalmente frames do tipo PB (isto é, dois frames consecutivos são codificados como uma entidade). Se nenhum frame for "saltado", o período do frame PB seria de 2.?, ao ser emitido pelo codificador. Entretanto, no decodificador, o frame tipo B é mostrado primeiro com um período de ? e então o frame tipo P com período de ?.

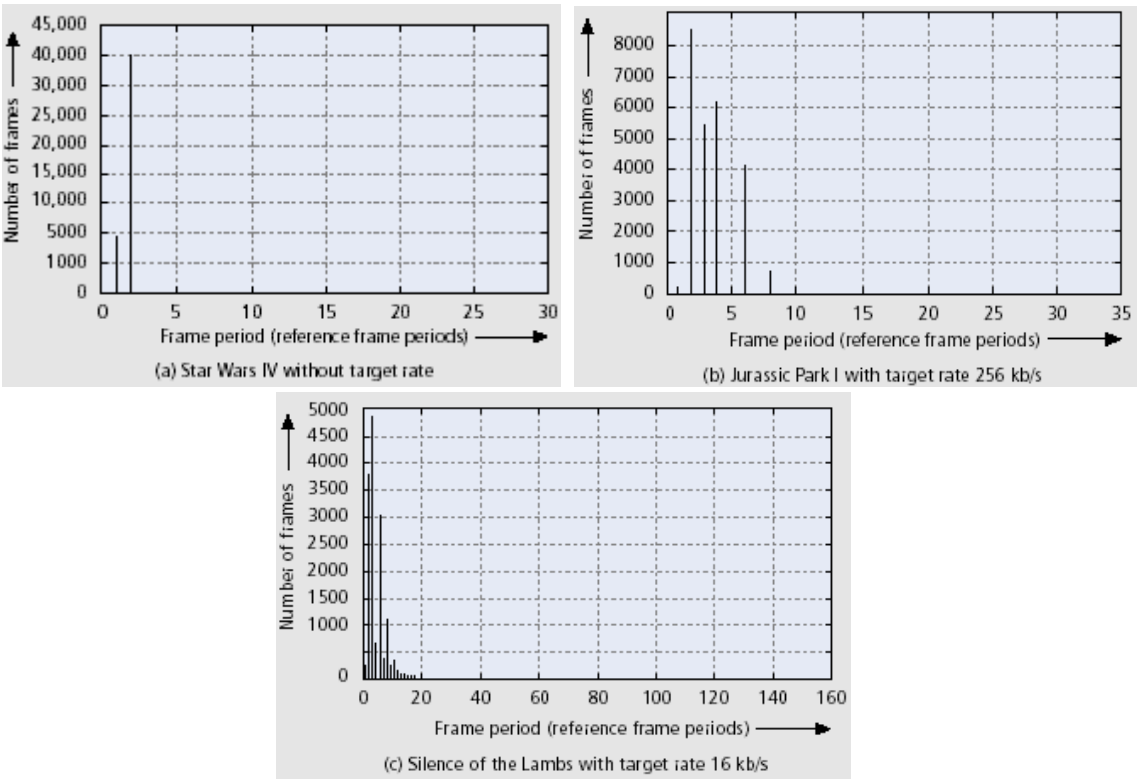


Figura C-1 – Densidade de probabilidade para períodos de frame H.263

C.3.2.1 Taxas de Bits e Tamanhos de Frames

A tabela C2 fornece um panorama das estatísticas para os tamanhos de frame. Nesta tabela, o codificador H.263 foi programado para gerar taxas alvo de bits para alguns vídeos.

A fim de obter a taxa alvo média de bits para o fluxo de vídeo, o codificador procura manter a mesma relação de compressão para todos os vídeos. Note-se que, para uma dada taxa alvo, todos os vídeos codificados têm a mesma taxa de bits média, mesmo que seus tamanhos médios de frame sejam diferentes. Para a taxa alvo de 16 Kbps, por exemplo, a codificação do filme *"Soccer"* possui tamanho médio de frame de 655

bytes, enquanto para *"Silence of the Lambs"* foi 370 bytes. Estas diferenças demonstram que o codificador, para manter uma taxa alvo de bits média, precisa "saltar" mais frames de *"Soccer"*, conseqüentemente o período médio para este vídeo é maior.

Comparando-se as codificações 256 Kbps com as VBR observa-se que alguns casos VBR têm razões de compressão maiores que os vídeos correspondentes comprimidos com a taxa alvo de 256 Kbps. Como exemplo, vemos que para o filme *"Star Wars IV"*, a razão é de 65 com VBR e de 29,7 com 256 Kbps.

A codificação VBR, entretanto, possui uma variação maior nos tamanhos médios de frame (variando de 903.78 a 3993.31), sendo esta variação tanto maior quanto for sua razão de compressão. É importante notar que para um vídeo codificado pelo H.263, com o tamanho de frame variável, o tamanho de frame é somente uma componente das estatísticas do fluxo de vídeo.

Rate	Trace	Comp. ratio YUV:H.263	Mean $\bar{X}$ [byte]	CoV $s_X/\bar{X}$	Peak/Mean $X_{max}/\bar{X}$
16 kbps	*Jurassic Park I*	476.36	476.36	0.67	20.83
	Silence of the Lambs	476.43	369.85	0.67	33.90
	Star Wars IV	476.43	326.02	0.61	11.32
	Soccer	476.30	654.56	0.60	7.10
64 kbps	*Jurassic Park I*	118.96	1132.02	0.36	7.89
	Silence of the Lambs	118.95	1129.94	0.41	11.10
	Star Wars IV	118.95	1153.32	0.43	7.11
256 kbps	*Jurassic Park I*	29.73	4533.67	0.35	2.61
	Silence of the Lambs	29.73	4453.81	0.39	5.00
	Star Wars IV	29.73	4563.53	0.33	3.58
VBR	*Jurassic Park I*	17.08	3993.31	0.64	4.55
	Silence of the Lambs	25.19	2703.46	0.99	10.27
	Star Wars IV	65.79	1048.21	0.66	8.58
	Mr. Bean	25.00	2662.61	0.60	6.09
	First Contact	44.64	1512.82	0.78	7.68
	From Dusk Till Dawn	19.30	3378.48	0.58	4.81
	The Firm	57.17	1241.80	0.80	7.39
	Formula 1	14.25	3826.30	0.47	3.69
	Soccer	10.18	5583.62	0.50	4.05
	ARD News	20.17	3442.46	0.77	4.45
	ARD Talk	30.70	2374.17	0.55	5.59
	N3 Talk	27.96	2545.62	0.57	5.48
	Office-Cam	84.01	903.78	0.36	5.74

Tabela C-2 – Estatísticas de tamanhos de frame para o H.263

Para os objetivos desta dissertação, necessita-se considerar apenas os tamanhos de frame em conjunto com seus períodos associados. A figura C2 ilustra os histogramas de tamanho *Xn* de frame para três *traces*. Observa-se, por exemplo, que a codificação com taxa alvo de 256 Kbps para *"Jurassic Park I"* tem uma distribuição bimodal pronunciada dos tamanhos do frame.

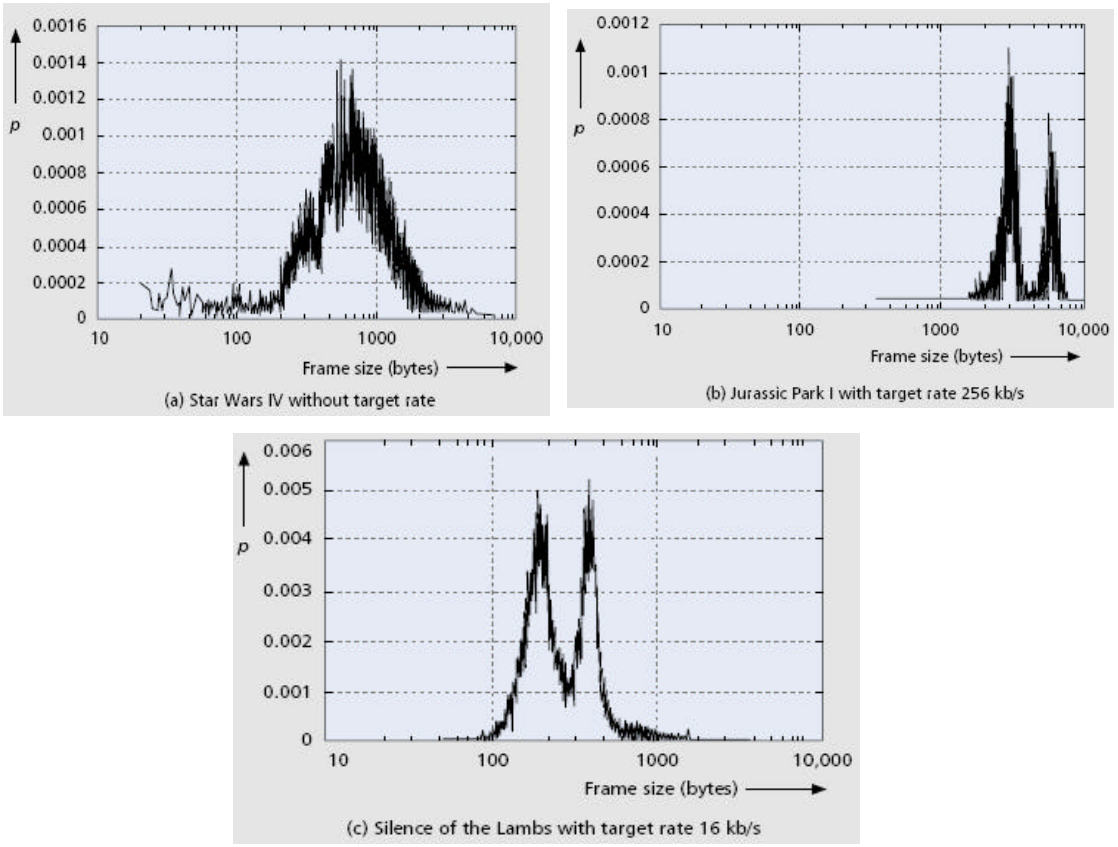


Figura C-2 – Histogramas para tamanhos de frame H.263

C.3.2 Medidas efetuadas com o JMF

Na tabela C3 abaixo, são mostrados extratos de dois *traces*, gerados com aplicativos escritos em JMF (um transmissor e um receptor), contendo as seguintes informações:

(a) número do pacote com seu respectivo tamanho em bytes

(b) tempos entre pacotes, contabilizados de 10 em 10 pacotes (*lag* = 10)

As análises sobre estes dados estão feitas no capítulo 4.

Pacote número	Tamanho (bytes)	Tempos, a cada 10 pacotes (ms)
1	649	130
2	951	140
3	921	241
4	914	180
5	985	70
6	953	170
7	647	191
8	948	160
9	896	40
10	893	170
11	986	141
12	969	190
.....	.....	160
894	891	40
895	923	140
896	271	141
897	946	170
898	758	140
899	976	100
900	634	.....
901	948	

Média = 765,3917869

(A) (B)

Tabela C-3 – Extratos de traces com tamanhos e tempos entre pacotes, para o H.263

Anexo D O Java Media Framework (JMF)

D.1 O que é JMF ?

Fundamentalmente, o JMF é uma extensão de Java para manipulação de áudio e vídeo. A JMF API (Java Media Framework Application Programming Interface) é uma API oficial que estende o núcleo da plataforma Java (SUN MICROSYSTEMS, 1999), (DEITEL, P.J, 2002).

JMF provê uma arquitetura unificada que contempla um protocolo de mensagens para o controle da aquisição, processamento e a entrega de mídias de tempo-real. O JMF é projetado para suportar a maioria de tipos de mídia padrão, tais como AIFF, AU, AVI, GSM, MIDI, MPEG, QuickTime, RMF e WAV.

Características principais da JMF API

- independência da plataforma. Um programa executará em qualquer lugar que funcione com Java.
- manipulação integrada e uniforme de áudio e vídeo como objetos de mídia.
- uma estrutura de processamento uniforme que suporta todas as operações com mídia (por exemplo, efeitos).
- captura de mídia de dispositivos tais como câmeras e microfones.
- recepção de fluxos de mídia transmitidos através da rede.
- transmissão de fluxos de mídia (através da rede).
- extensível para suportar novos formatos e plug-ins.

Arquitetura de Alto Nível

Dispositivos como fitas de vídeo (VHS) vídeos cassetes (VCRs) fornecem um modelo familiar para a gravação, processamento e apresentação de mídias. Para se assistir a um filme usando um VCR é necessário a introdução de uma mídia, contida na

fita de vídeo. O VCR lê e interpreta os dados da fita e emite sinais apropriados à televisão e aos alto-falantes. Este modelo está ilustrado na figura D1.

O JMF usa este mesmo modelo básico. Um objeto *DataSource* encapsula o fluxo de mídia da mesma forma que a fita e um objeto *Player* provê os mecanismos de processamento e controle similares a um VCR.

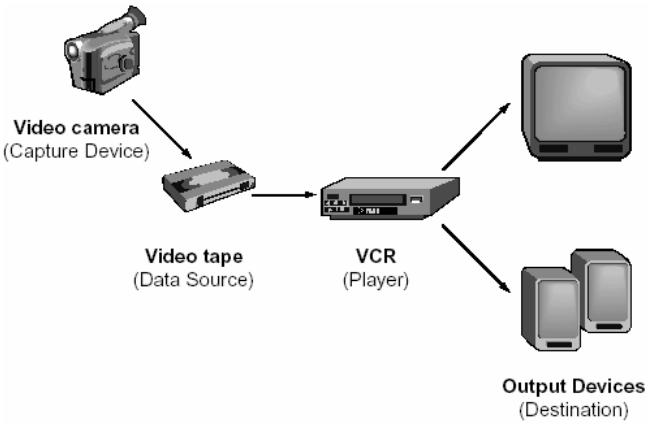


Figura D-1 – Modelo básico de processamento de mídia

D.2 Modelos da Arquitetura do JMF

A arquitetura de classes do JMF está calcada em uma concepção construída a partir de modelos: de tempo, de dados e de eventos. A partir destes modelos, são concebidas as interfaces e as classes do JMF. A figura D2 ilustra um diagrama de herança para algumas das interfaces relevantes ao SCDM.

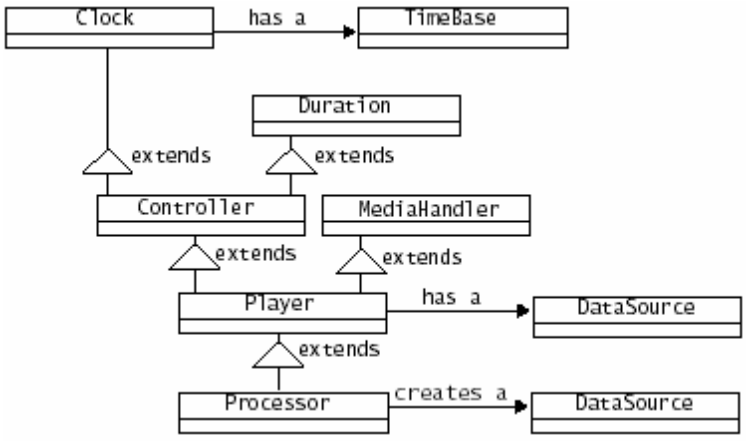


Figura D-2 – Diagrama de heranças de classes no JMF

D.2.1 Modelo de Tempo

O JMF fornece um modelo em camadas para o tempo. Na camada mais interna existe uma representação "exata" de um instante de tempo no nível do nanosegundo (dependente da precisão do Sistema Operacional que o suporta). As camadas seguintes (TimeBase) suportam o conceito de uma fonte que envia pulsos constantemente: um relógio.

As camadas mais externas suportam abstrações de tempo como tendo começado ou parado e estando em um dentre alguns estados potenciais. Estas camadas permitem ao programador exercer o controle da temporização (por exemplo, alterando a taxa de um filme) assim como ter informações de processos dependentes do tempo, no nível apropriado para a tarefa. Este controle é fundamental para os programas em JMF.

A Interface Clock

As classes que implementam a interface TimeBase fornecem uma fonte inalterável de tempo através da emissão constante de pulsos. Entretanto, o controle de uma mídia significa prover controle sobre as propriedades temporais dessa: com a possibilidade de começar ou pará-la em posições arbitrárias bem como ter o controle de sua taxa (por exemplo, adiantá-la rapidamente ou rebobiná-la em um VCR). As interfaces *Controller, Player, e Processor* (descritas adiante), centrais à funcionalidade do JMF, estendem a interface `Clock`.

Um objeto *media time* da interface *Clock* representa a posição atual dentro do fluxo de mídia - o início do fluxo é definido como tempo zero para *media time*, enquanto que o final do fluxo terá o máximo *media time* para essa mídia. Note-se que isto apenas aplica-se a mídias pré-armazenadas em um arquivo, por exemplo. Para fluxo de mídia, a duração deste somente pode ser determinada ao seu final - o tempo necessário para apresentá-lo.

Clock provê uma transformação de tempo de TimeBase àquele associado com a mídia. Esta é uma transformada linear simples que requer três parâmetros: a taxa (p. ex., do filme), o tempo de início da mídia e o tempo inicial da base de tempo. A partir desses, o tempo da mídia pode ser determinado.

D.2.2 Modelo de Dados

Para gerenciar os dados, são utilizados os *DataSources* e os *DataSinks.* Estas classes permitem a abstração tanto da localização quanto do protocolo utilizado para o acesso à mídia.

Um *DataSource* é o meio pelo qual os *Players,Processors,* ou DataSinks obtém seus dados. A criação destes objetos sempre envolve um *DataSource*: seja explicitamente através método da criação (como na criação de um *DataSink*), ou criado como a parte do processo maior (como na criação de um *Player,* onde um *MediaLocator* seja fornecido).

Um *DataSink* é usado para ler uma mídia de um *DataSource* e enviá-la a algum destino - geralmente diverso do dispositivo da apresentação (*Player*). Um *DataSink* pode escrever dados em um arquivo, através da rede, ou funcionar como um difusor (broadcast) de RTP.

Depois de identificada a localização, o próximo passo é identificar o formato da mídia. Para isso é utilizada a classe *Format*. Essa classe origina duas outras classes filhas:

- *AudioFormat*: especifica atributos de trilhas de áudio; e
- *VideoFormat*: especifica atributos de trilhas de vídeo.

D.2.3 Modelo de Eventos

O JMF utiliza-se de um mecanismo estruturado para reportar eventos, a fim de manter os programas informados sobre o estado da mídia, permitindo-lhes responder às condições de erro derivadas de mídia. Como exemplo, temos a ausência de dados e condições de indisponibilidade de recursos.

Quando um objeto precisa disparar um evento que informe seu estado atual, esse instancia um *MediaEvent*. São quatro as classes filhas de *MediaEvent*:

- *ControllerEvent*: classe base para os eventos disparados por um *Controller*;
- *DataSinkEvent*: classe base para os eventos disparados por um *DataSink*;

- *GainControlEvent*: esse tipo de evento é disparado por um *GainControl*, para sinalizar uma mudança de estado. *GainControl* é uma interface específica para trilhas de áudio, por isso não será aprofundada nesse trabalho; e
- *RTPEvent*: classe base para todas as notificações de um *SessionManager* (a ser visto adiante).

O recebimento e tratamento de eventos do tipo *ControllerEven*t é feito por classes que implementam a interface *ControllerListener*, possuindo o método *controllerUpdate (ControllerEvent* event), que será chamado a cada *ControllerEvent* gerado.

O recebimento e tratamento de eventos do tipo *DataSinkEvent* é feito por classes que implementam a interface datasink.DataSinkListener e que possuam o método dataSinkUpdate(*DataSinkEvent* event), que será chamado a cada *DataSinkEvent* gerado.

D.3 Apresentação e processamento de mídia no JMF

No JMF, o processo de apresentação/processamento é modelado pela interface *Controller*. Esta define os estados básicos e um mecanismo que permite a um objeto controlar, apresentar, ou capturar mídia.

A interface *Controller* estende diretamente a interface *Clock* em três áreas:

- provê um mecanismo de eventos, pelo qual os estados podem ser monitorados;
- provê um mecanismo pelo qual os objetos podem exercer um controle adicional sobre o objeto *Controller*; e
- amplia o conceito de "parado" para cinco estados relacionados à alocação de recursos, de modo que os processos consumidores de tempo possam ser controlados por meio do disparo de eventos.

Estes cinco estados representam o ciclo de vida de um objeto *Controller* (p.ex., um *Player*) desde sua criação até estar pronto para iniciar. As transições que conduzem um objeto *Controller* para além do estado *Prefetched* (pronto para começar), estão sob o controle do programa, enquanto que os erros ou outros eventos podem conduzi-lo a uma transição no sentido inverso (menos pronto para começar). Vide figura D3.

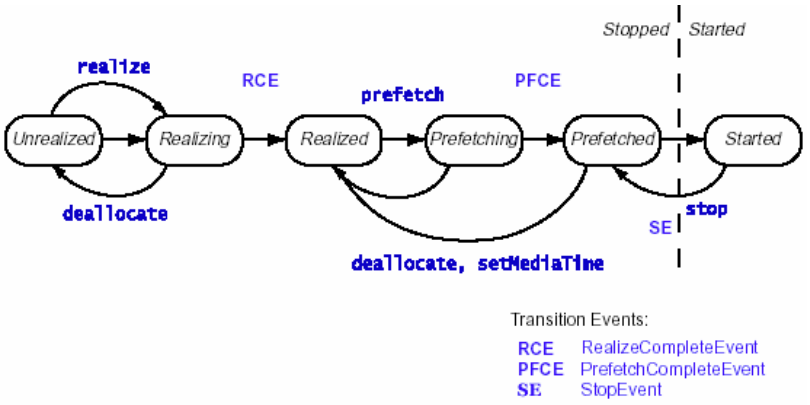


Figura D-3 – Diagrama de Transição de estados para um objeto tipo *Controller*.

Algumas das operações executadas antes que a mídia possa ser apresentada, podem consumir muito tempo. Devido a este fato, o JMF permite o controle sobre sua execução, informando através de eventos quando cada tarefa estiver pronta.

Os programas (objetos) podem ser informados de mudanças de estado do objeto *Controller*, adicionando-se como ouvintes para tais eventos (da classe *ControllerEvent*, pertencente ao modelo de eventos). Desta maneira, um programa pode iniciar operações bem como responder prontamente à medida que o objeto *Controller* move-se através de seus vários estágios.

Conforme a figura D.3, os cinco estados do objeto *Controller* que ampliam de *Clock* o conceito (estado) de "tempo parado" são:

- Unrealized - o *Controller* foi criado, mas não começou a executar sua primeira tarefa de alocar recursos. Todos os *Controller* começam neste estado.
- Realizing - o *Controller* está adquirindo as informações sobre os recursos necessários ao seu funcionamento. Este é um estado de transição que deve resultar (caso não ocorram erros) em "Realized". O tempo para esta transição é variável.
- Realized - o *Controller* adquiriu as informações sobre todos os recursos necessários à execução de sua tarefa e o tipo de mídia a ser apresentada. Provavelmente o *Controller* já adquiriu todos os recursos, com exceção dos que envolvem recursos exclusivos do sistema (p.ex. alocando algum dispositivo de hardware).
- Prefetching - o processador está iniciando os procedimentos iniciais, tais como iniciar a carga da mídia no buffer ou adquirir recursos de hardware. Este é um estado de transição que deve resultar em "Prefetched". O tempo para esta transição é variável.
- Prefetched - o *Controller* adquiriu todos os recursos necessários e está pronto para ser iniciado.

Como exemplo, ao se executar o método *Start*, o *Controller* vai para o estado *Started*. Um *Controller* (p. ex. um *Player*) que está no estado *Started*, tem tanto seu objeto *Time base time* como *media time* mapeados e seu objeto *Clock* já está funcionando, mesmo que o *Player* possa estar esperando por um instante particular para começar a apresentar sua mídia.

Apresentadores (*Players*)

Um *Player* processa um fluxo de entrada e entrega-o em um instante preciso de tempo. Um *DataSource* é usado para entregar um fluxo de entrada ao Player. O destino depende do tipo de mídia que está sendo apresentada. Um objeto *Player*, estendendo as interfaces Controller e MediaHandler, pode encontrar-se em um dos estados do objeto *Controller*.

Processadores (*Processors*)

Os processadores também podem ser usados para apresentar mídia. Um processador é um tipo especializado de *Player* que provê controle sobre qual processamento será executado no fluxo de mídia de entrada. Enquanto que o processamento realizado por um *Player* é definido pelo JMF, o objeto *Processor* permite que o usuário defina qual processamento será realizado nos dados de entrada (*DataSource*), por exemplo, pela introdução de um Codec. Um processador suporta os mesmos controles de apresentação que um *Player*. Vide figura D4.

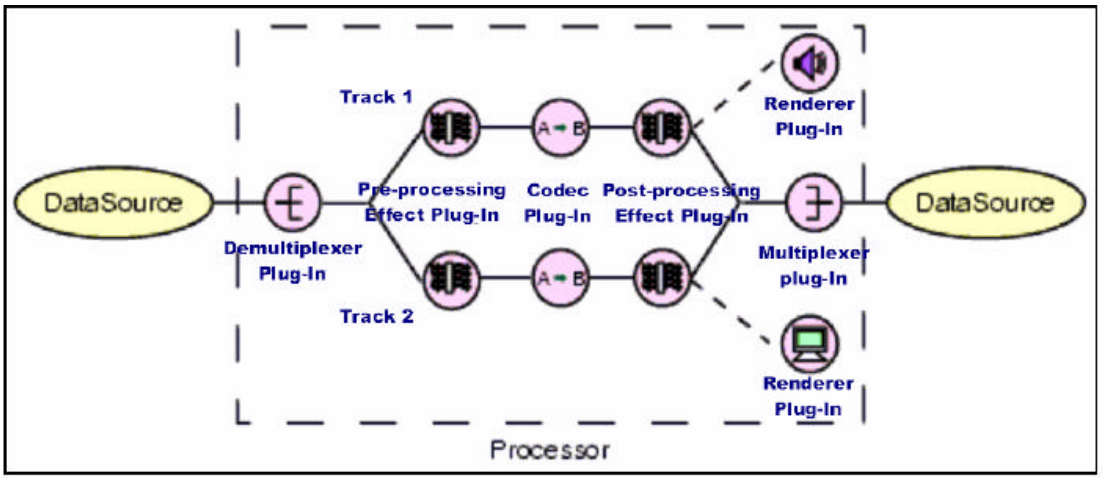


Figura D-4 - Fluxo de processamento efetuado por um *Processor.*

Abaixo são enumerados os estágios do fluxo interno de dados de um *Processor*:

- Demultiplexação: é feita por um *plug-in* do tipo *Demultiplexer* e consiste na análise dos dados de entrada e divisão desses dados em diferentes trilhas de áudio e vídeo;
- Pré-processamento: nesse estágio são aplicados efeitos sobre uma trilha de entrada específica, visando facilitar o trabalho *Codec*. O pré-processamento é feito por um *plug-in* do tipo *Effect*, que, diferente do *Codec*, não está habilitado a modificar o formato da mídia;
- Transcodificação: é feita por um *plug-in* do tipo *Codec* e consiste no processo de conversão das trilhas da mídia de um formato para outro;
- Pós-Processamento: nesse estágio são aplicados efeitos sobre a trilha transcodificada. Assim como o pré-processamento, o pós-processamento é feito por *plug-ins* do tipo *Effect*;
- Multiplexação: é o processo inverso da demultiplexação. Nele várias trilhas são convertidas em um único segmento de dados. Esse processo é feito por *plug-ins* do tipo *Multiplexer*;
- Renderização: é feita por *plug-ins* do tipo *Renderer* e consiste no processo de redistribuição da mídia para exibição ao usuário.

D.4 Transmissão de fluxos multimídia

As classes do JMF relacionadas ao protocolo RTP são extensões a esse. Não substituem nem aperfeiçoam as funcionalidades do núcleo do JMF, como por exemplo, as encontradas nas classes do *Player*, do *Processor*, do *DataSource*, do *DataSink*, entre outras. Estes permanecem inalterados, constituindo o centro de manipulação de mídias, mas agora as localizações das mídias podem ser originadas tanto de um arquivo local bem como através de um fluxo proveniente da Internet.

Formatos e Conteúdos RTP:

Nem todos os conteúdos e formatos de mídia oferecidas pelo JMF estão disponíveis para uso com o RTP. Como exemplo, um fluxo transportado por um formato tipo JPEG_RTP significa que a mídia foi inicialmente comprimida por um codec JPEG e, em seguida, empacotada num pacote RTP. O código a seguir ilustra a criação de um

objeto chamado obj, do tipo Format, cujo conteúdo será comprimido por JPEG e packetizado por RTP:

```
Format obj = new Format(Format.JPEG_RTP);
```

Classes relacionadas ao RTP:

As seguintes funcionalidades são oferecidas pelas classes relacionadas ao RTP:

- gerenciamento de sessões, participantes e fluxos de mídias (por exemplo, começar, terminar, adicionar, e assim por diante)
- monitoração (através das interfaces *listener*) de eventos de RTP (por exemplo, novos participantes que juntaram-se à sessão)
- obtenção e geração de estatísticas (por exemplo, qualidade da conexão)

A classe principal para manipulação de dados RTP é a *RTPManager*. Um objeto do tipo *RTPManager* atua como um intermediário entre os objetos do JMF para manipulação de mídia (por exemplo, o `Player`, ou o `Processor`) e os detalhes relativos ao protocolo e à sessão RTP. Esta classe facilita a programação em JMF, pois libera o usuário de preocupações com detalhes de implementação.

Configurando uma sessão RTP:

Um objeto *RTPManager* pode ser instanciado após sua pré-configuração apropriada ser executada: deve-se fornecer um endereço de multicast ou de unicast assim como um objeto de *DataSource*.

O *RTPManager* deve ser inicializado com o método *initialize*(). Existe uma versão multi-argumento de *initialize*() que permite um controle mais fino dos parâmetros da sessão RTP, como por exemplo, a porcentagem da largura de banda consumida pelo tráfego RTCP e pela criptografia (se alguma) empregada. Isto permite a definição do intervalo de reportagem da sessão, mencionada anteriormente no item A.3.2.

O método *addTarget*() é usado especificar o alvo de uma sessão de RTP. Para a transmissão, este alvo é o endereço de transporte (IP + porta) do receptor. Para a recepção, esse alvo é o endereço do transmissor. O método do *addTarget*() abre a sessão, fazendo com que os relatórios de RTCP sejam gerados assim como os eventos *SessionEvents* apropriados.

O método do *createSendStream*() cria um novo objeto tipo SendStream a partir de um *DataSource* (a saída de um *Processor*) existente. Esta etapa é necessária apenas para a transmissão dos dados.

D.4.1 Estatísticas da Sessão RTP

Uma tarefa importante no gerenciamento de uma sessão RTP é a monitoração da qualidade da conexão que está sendo experimentada por todos os seus participantes. Particularmente, a gerência inteligente pode, por exemplo, ajustar o tráfego de saída em resposta às mudanças na rede. O protocolo RTCP fornece um mecanismo básico para este tipo da monitoração através da emissão periódica de relatórios, pelos participantes. No JMF, os relatórios de RTCP são representados pelas interfaces *Report* e *Feedback*.

A interface *Report* é subdividida em *SenderReport* e em *ReceiverReport*. Os objetos do tipo *Report* podem ser obtidos para cada participante da sessão, através do método *getReports*(). Estes objetos possuem um método chamado *getFeedbackReports*() que permitem o acesso a estatísticas do tipo: o número dos pacotes perdidos, o inter-arrival jitter, e de outras propriedades do fluxo

Estatísticas globais estão disponíveis através dos objetos *GlobalReceptionStats* e *GlobalTransmissionStats*, obtidos a partir do *RTPManager* responsável para a sessão.

Um objeto *GlobalReceptionStats* fornece métodos para determinar-se o número de pacotes que não foram transmitidos por falha, o número de pacotes recebidos com erros, e o número de colisões locais.

Um objeto *GlobalTransmissionStats* fornece seis métodos para a determinação da qualidade de transmissão da sessão inteira: o número total de bytes emitidos, o número de transmissões com falhas e o número de colisões locais e remotas.

Anexo E Pseudocódigos e tabelas relativas ao Emulador Paramétrico

E.1 Pseudocódigo do Transmissor Sintético

```
Instancia novos objetos: soquete UDP e pacote
while(count < 4000L)
{
        Copia “count” e o tempo corrente como carga do pacote
                envia   o pacote através do soquete UDP
        Incrementa count
        thread suspende-se por “X” milisegundos
}
```

Note-se que “X” é o intervalo de emissão de pacotes do transmissor.

E.2 Pseudocódigo do Receptor Sintético

```
Instancia novos objetos: soquete UDP e pacote
while(count < 4000L)
{
        recebe o pacote pelo soquete UDP

        //   SECAO DE JITTER
        obtém “tempo_transmissao_pacote_atu” da carga do pacote recebido
        mede o tempo atual e o salva em “tempo_recepcao_pacote_atu”
        calcula o tempo de trânsito relativo (vide item A.3.1.1), da seguinte forma:
        diff = (tempo_recepcao_pacote_atu - tempo_transmissao_pacote_atu) -
                (tempo_recepcao_pacote_ant - tempo_transmissao_pacote_ant);
        calcula o jitter, com base na RFC3550 (vide item A.3.1.1), da seguinte forma:
        jitter_atu = jitter_ant + (Math.abs(diff) - jitter_ant)/16.0;

        //  SESSÃO DE PERDAS
        calcula  “tam_episodio_perda”
        se  ( tam_episodio_perda  >=  1)  (há uma perda)
                incrementa vetor de perdas na posição indexada por  “tam_episodio_perda”
}
        imprime vetor de perdas
```

E.3 Jitter entre pacotes

Pacote número	Diferença de tempo entre o pacote atual e o anterior (ms)
1	0
2	10
3	20
4	10
5	20
6	10
7	20
8	10
9	21
10	10
11	20
12	20
13	10
14	10
15	20
16	10
17	20
18	20
19	10
20	10

Tabela E-1 – Diferença entre os tempos em uma seqüência de pacotes recebidos

Valores do *jitter*	Condições (que o Jitter medido deve ter)	Número de Ocorrências
10	Se J = 10 ms	1789
20	Se 10 < J = 20	725
30	Se 20 < J = 30	399
40	Se 30 < J = 40	265
50	Se 40 < J = 50	187
60	Se 50 < J = 60	104
70	Se 60 < J = 70	80
80	Se 70 < J = 80	47
90	Se 80 < J = 90	35
100	Se 90 < J = 100	14
110	Se 110 < J = 120	9
120	Se 120 < J = 130	2
130	Se 130 < J = 140	5
140	Se 140 < J = 150	4
400	Se 150 < J = 400	0

Tabela E-2 – Exemplo de histograma de Jitter

Anexo F Pseudocódigos e tabelas relativas ao Inspetor de Desempenho

F.1- Condições de Rede versus MOS

Condições de rede		Julgamento dos Observadores sobre a qualidade final do vídeo recebido					número	MOS
JITTER	PERDA	Excelente	Bom	Razoável	Ruim	Péssimo	avaliações	
0	0	7	8				15	4,466667
0	20	1	4	9	1		15	3,333333
0	40			4	10	1	15	2,2
0	60			2	9	4	15	1,866667
0	80		1		1	13	15	1,266667
20	0	12	2	1			15	4,733333
20	20			5	8	2	15	2,2
20	40		1	8	6		15	2,666667
20	60				4	11	15	1,266667
20	80				9	6	15	1,6
40	0	3	10	2			15	4,066667
40	20		1	8	6		15	2,666667
40	40			6	8	1	15	2,333333
40	60				8	7	15	1,533333
40	80			1	2	12	15	1,266667
60	0	4	9	2			15	4,133333
60	20		3	9	3		15	3
60	40			2	10	3	15	1,933333
60	60				3	12	15	1,2
60	80			1		14	15	1,133333
80	0	5	9		1		15	4,2
80	20		8	5	2		15	3,4
80	40			4	7	4	15	2
80	60		1		2	12	15	1,333333
80	80			1	2	12	15	1,266667

Tabela F-1 – Condições de Rede *versus* MOS

F.2 - Pseudocódigo do Receptor

Aplicação: AVReceiver.java

Recebe um "target" (peer remoto para iniciar uma conexão)
Instancia um objeto RTPManager
Adiciona ao RTPManager:

- SessionListener: verifica quando um novo participante junta-se à sessão
- ReceiveStreamListener: - estatísticas do fluxo recebido;
  - permite a criação de objetos DataSource e com isto, o Processor;
  - detecta o fim da sessão

- RemoteListener - estatísticas do peer remoto (o transmissor)

Cria soquetes para o target
Abre sessão para o target
Fica aguardando os eventos do JMF para acionar o "Listener" correto

F.3 Extrato de código do RemoteListener

```
public void update( RemoteEvent event) {

String              timestamp           = getTimestamp();
SessionManager      mgr;
mgr = event.getSessionManager ();
        if( event instanceof ReceiverReportEvent) {
        ReceiverReport rr= ((ReceiverReportEvent) event).getReport();
        if( rr != null) {
                Participant participant  = rr.getParticipant();
                 Vector   rcvFB = rr.getFeedbackReports();
                        // quando houver dados iniciais, rcvFB.size() > 0
                        if ( rcvFB.size() > 0 )  {}
                    } else
                        System.out.println(" *** RR nulo !!!! ");
        } else if ( event instanceof SenderReportEvent) {
                SenderReport sr = ((SenderReportEvent) event).getReport();
                if( sr != null) {
                        Participant participant= sr.getParticipant();
                        Feedback feed = sr.getSenderFeedback();
                        if (feed != null) {
                        System.out.println("\nSR " + timestamp /* +" vindo de "+sr.getSSRC() */ +
                                        "  jitter: " + mutex.get_jitter() + " Occup: " + mutex.consulta_seq() +
                                        " buf: " + mutex.consulta_tempo() );
                        Vector  fluxos = mgr.getReceiveStreams();
                        if ( fluxos.size() > 0 )
                        {
                        ReceptionStats rs = ( (ReceiveStream)fluxos.elementAt(0) ).getSourceReceptionStats();
                        int PDUlostintervalo = rs.getPDUlost() - prevPDUlost;
                        prevPDUlost = rs.getPDUlost();
                        int     PktContAtual            = rs.getPDUlost() + rs.getPDUProcessed() ;
                        long  PktContIntervalo          = PktContAtual - prevPktCont;
                        prevPktCont                     = PktContAtual;

                        perda = (new Float( 100*(float)PDUlostintervalo / (float)PktContIntervalo )).toString();
                        System.out.println(" perda : " + perda + "\n");
                        }

                        //   CHAMA O CLASSIFIC.  NEURO-FUZZY E IMPRIME A CLASSIFICACAO
                        if ( perda != null ) {  // ou seja, tem um Report.
                                String comando = null;
                                comando = "Classificador " + perda;
                                comando += " " + (new Double(mutex.get_jitter() ) ).toString();
                                comando += " Neuro_Fuzzy.fis";
                                try {
                                        String outlist[] = runCommand(comando);
                                        // mostra a saída do Classificador:  o MOS
                                        for (int i = 0; i < outlist.length; i++)
                                                System.out.println(outlist[i]);
                                }
                                catch (IOException e) {  System.err.println(e);   }
                        }
                        } else
                                System.out.println("  feed nulo !!!");
          }
          else
                System.out.println(" +++++ SR nulo !!!! ");
        } // fim_SR
  } // fim_update
```

Referências:

3Com Corp, 2004, "3Com XRN™ Technology Brief". In: *3Com Corp*, white paper disponível em (último acesso em 15/10/06):
http://www.3com.com/other/pdfs/products/en_US/500928.pdf.

APOSTOLOPOULOS, G., WILLIAMS, D., KAMAT, S., GUERIN, R., ORDA, A., and PRZYGIENDA, T., 1999, "QoS Routing Mechanism and OSPF Extensions". In: *Internet Engineering Task Force, RFC 2676.*

BLAKE, S., BLACK, D.L., CARLSON, M., DAVIES, E., WANG, Z. and WEISS, W., 1998, "An Architecture for Differentiated Services". In: *Internet Engineering Task Force, RFC 2475*

BOLOT, J.C., FOSSE-PARISIS, J., and TOWSLEY, D., 1999, "Adaptive FEC-based error control for Internet telephony". In: *Proceedings of IEEE INFOCOM*, pages 1453-1460, New York, NY, USA.

BORMANN, C. et al, 1988, "RTP Payload Format for the 1998 Version of ITU-T Rec. H.263 Video (H.263+)". In: *Internet Engineering Task Force, RFC 2429*

CADZOW, J., 1987, *Foundations of Digital Signal Processing and Data Analysis.* New York, Macmillan (ed).

CECÍLIO, E.L, 2002, *Uma Arquitetura de Gerenciamento de Desempenho Pró-ativo Distribuído Usando Tecnologia Ativa*, Dissertação de Mestrado, NCE/UFRJ.

CECÍLIO, E.L, DUMONT, A. P. M., CORREIA, R. de B, RUST, L. F. da C. e PIRMEZ, Luci, 2003, "Uma Arquitetura de Gerenciamento de Desempenho Pró-ativo Distribuído Usando Tecnologia Ativa". In: *XXI Simpósio Brasileiro de Redes de Computadores SBrC*.

CHANDRA, T.D. e TOUEG, S., 1996, “Unreliable Failure Detectors for Reliable Distributed Systems”. In: *Journal of the ACM*, vol. 43 No.2, March 1996, pp. 255-267

CHEN, W., TOUEG, S. e AGUILERA, M., 2000, “On the quality of service of failure detectors”. In: *Proc. of the First Int. Conf. on Dependable Systems and Networks.*

CISCO, 2004, “OSPF Design Guide”. In: *Cisco Systems*, white paper disponível no endereço (último acesso em 15/10/06): http://www.cisco.com/warp/public/104/1.pdf

CISCO, 2005a, “Cisco IOS XR Multicast Configuration Guide”. In: *Cisco Systems*, white paper part number: OL-5552-05, disponível no endereço (último acesso em 16/10/06): http://www.cisco.com/application/pdf/en/us/guest/products/ps5763/c2001/ccmigration_09186a00803dcd65.pdf

CISCO, 2005b, “Cisco IOS Quality of Service Update”. In: *Cisco Systems*, white paper disponível no endereço (último acesso em 15/10/06): http://www.cisco.com/en/US/products/ps6558/prod_presentation_list.html

CORREIA, R. de B., 2003, *Rerroteamento de fluxos utilizando redes MPLS e tecnologia ativa*, Dissertação de Mestrado, NCE/UFRJ.

CROZIE, K. R., 1997, “Implementing HSRP in the Enterprise Network”. In: *Cisco Systems*, white paper disponível no endereço (último acesso em 15/10/06): http://www.cisco.com/warp/public/cc/so/cuso/epso/entdes/hsrp_wp.pdf.

DEITEL, P.J, DEITEL, H.M., 2002, "Java Media Framework and Java Sound". In: Deitel & Associates, Inc (eds), *Advanced Java™ 2 Platform How to Program*, 4 ed., chapter 22, Prentice-Hall Publishing

FERGUSON, P., HUSTON, G, 1998. “Quality of Service on the Internet: Fact, Fiction or Compromise ?”. In: *Internet Conference*, Genebra, Suiça.

FIGUEIREDO, D.R. e SILVA, E.S., 1999, “Efficient mechanisms for recovering voice packets in the Internet”. In: *Proc. of IEEE/Globecom´99*

FITZEK, F.H.P., REISSLEIN, M., 2001, *"MPEG–4 and H.263 Video Traces for Network Performance Evaluation*". In: IEEE Network, November/December 2001

GHINEA, G., THOMAS, J.P., FISH, R.S., 1999, "Quality of Perception to Quality of Service Mapping Using a Dynamically Reconfigurable Communication System". In: *IEEE Global Telecommunications Conference*, Rio de Janeiro, Brazil.

GOERENDER, S. e MACÊDO, R.J.A., 2002, "Um Modelo para Tolerância em Sistemas Distribuídos com QoS". In: *XX Simpósio Brasileiro de Redes de Computadores SBrC´02*

GOMES, R.L., 2001, *Autoria e Apresentação de Documentos Multimídia Adaptativos em Redes*, Dissertação de Mestrado, Núcleo de Computação Eletrônica - UFRJ.

HINDEN, R., 2004, "Virtual Router Redundancy Protocol (VRRP)", Internet Engineering Task Force, *RFC 3768.*

HUITEMA, C.; 2000, *Routing in the Internet*, 2nd Edition – Prentice Hall

ITU-R BT.500-10, 2000, "Methodology for the subjective assessment of the quality of television pictures", International Telecommunication Union, March 2000.

ITU-T H.263, 1996, "Video coding for low bitrate communication", International Telecommunication Union, May 1996.

ITU-T P.910, 1999, "Subjective video quality assessment methods for multimedia applications", International Telecommunication Union, September 1999.

JACOBSON, V., 1988, "Congestion avoidance and control" In: SIGCOMM '88, Proceedings of the ACM Symposium on Communications Architectures and Protocols, pages 314-329, Stanford, CA, USA, August 1988.

JANG, J.-S. R. and SUN, C.-T.,1995, "Neuro-Fuzzy Modeling and Control". In: *Proceedings of the IEEE*.

KAWAMURA, R. and STADLER, R., 2000, "Active Distributed Management for IP Networks". In: *IEEE Communications Magazine*, Abril de 2000.

KROPOTOFF, A.B., 2002, *Um Emulador Paramétrico de Conexões Fim-a-Fim em Redes IP*, Dissertação de Mestrado, COPPE/UFRJ

KOEHLER, E., 2003, "Solutions Reference Design: IP Multicast Television Distribution" In: *Nortel Networks*, white paper disponível no endereço (último acesso em 15/10/06): http://www142.nortelnetworks.com/bvdoc/setips/july03/multicastiptvsrd.pdf.

KUROSE, J. F., KEITH W.R., 2003, *Computer Networking – A Top-Down Approach Featuring the Internet*, 2 ed, New York - Addison-Wesley Computing

MACÊDO, R.J.A., e RAMON, F.L.L., 2004, "Improving the Quality of Service of Failure Detectors with SNMP and Artificial Neural Networks". In: *Simpósio Brasileiro de Redes de Computadores SbrC´04*

MARKOVSKI, V., 2000, *Simulation and Analysis of Loss in IP Networks*, Master of Applied Science Thesis, SIMON FRASER UNIVERSITY.

MATLAB, 2004, "Fuzzy Logic Toolbox". In: The MathWorks Inc, *MATLAB Online Documentation*, version 7.0R14.

MEDEIROS, A. V., MAITELLI, A. L., FILHO, O. G., 2001, "Otimização das Funções de Pertinência de um Controlador Nebuloso utilizando Algoritmos Genéticos". In: *V Simpósio Brasileiro de Automação Inteligente*, BAURÚ – SP.

MELO, E. T. L., 2001, *Qualidade de Serviço em Redes IP com DiffServ: Avaliação através de Medições*, Dissertação de Mestrado, Universidade Federal de Santa Catarina

MIEGHEM, V.P., et al, 2001, "Hop-by-Hop Quality of Service Routing". In: *Computer Networks*, no 37, p. 407-423.

MOHAMED, S., RUBINO, G., 2002, “A Study of Real-Time Packet Video Quality using Random Neural Networks”, In *IEEE Transactions on Circuits and Systems for Video Technology* , vol. 12, num 12, p. 1071- 1083

MOON, S.B., SKELLY, P., and TOWSLEY, D., 1999, “Estimation and removal of clock skew from network delay measurements”. In: *Proc. IEEE INFOCOM*, vol. 1, pp. 227–234.

MORAES I. M. e DUARTE, O. C. M. B., 2004, “Uma Comparação entre Protocolos de Roteamento Multicast para Distribuição de TV na Internet”. In: *XXI Simpósio Brasileiro de Telecomunicações,SBT'04.*

NEWMAN, D., 2003, “XRN Interconnect architecture”, Network World Global Test Alliance, disponível no endereço a seguir (ultimo acesso em 11/06/06): http://www.networkworld.com/reviews/2003/0303xrnrev.html?page=1

OLSSON, O., STOPPIANA, M., e BAINA, J., 1997, “Objective methods for assessment of video quality: state of the art", In *IEEE Transactions on Broadcasting*, vol. 43, no. 4.

SCHULZRINNE, H., CASNER, S., FREDERICK, R. and JACOBSON, V., 2003, “RTP: A Transport Protocol for Real-Time Applications”, In: *Internet Engineering Task Force, RFC 3550.*

SEELING, P., REISSLEIN, M. and KULAPALA, B., 2004, “Network Performance Evaluation Using Frame Size and Quality Traces of Single-Layer and Two-Layer Video: a Tutorial”. In: *IEEE Communications Surveys***,** Vol. 6, No. 3

SLUMAN, C., 1989, “A Tutorial on OSI Management”, In *Computer Networks Magazine*, Vol. 17, n^{os} 4 e 5, outubro de 1989.

SILVA, E.S., LEÃO, R.M.M., NETO, B.R. e CAMPOS, S., 2002, “Performance Issues on Multimedia Applications”. In: *Lecture Notes in Computer Science*, Vol. 2459, Pages: 374 – 404.

STEINMETZ, R., 1996, “Human Perception of Jitter and Media Synchronization”. In *IEEE Journal on Selected Areas in Communications*, v.14, n.1, p.61-72.

SUN MICROSYSTEMS, INC, 1999, *Java Media Framework API Guide*, In: Sun Microsystems, manual disponível no endereço a seguir (ultimo acesso em 15/10/06): http://java.sun.com/products/java-media/jmf/2.1.1/specdownload.html.

TRUMP, T., 2000, “ Estimation of Clock Skew in Telephony over Packet Switched Networks”. In: *Proc. IEEE ICASSP*, vol. 5, pp. 2605–2609.

YAJNIK, M., MOON, S., KUROSE, J., and TOWSLEY, D., 1999, “ Measurement and modelling of the temporal dependence in packet loss”. In: *IEEE INFOCOM*, pages 345-352, New York, NY, USA.

WADDINGTON, D.G., HUTCHISON, D., 1998, “End-to-end QoS Provisioning through Resource Adaptation”. In: *IFIP Conference on High Performance Networking*.