CHRISTIANE MONTENEGRO BORTOLETO

MULTICAST SEMI-CONFIÁVEL PARA APLICAÇÕES MULTIMÍDIA DISTRIBUIÍDAS

Dissertação apresentada ao Programa de Pós-Graduação em Informática Aplicada da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Mestre em Informática Aplicada.

CURITIBA

2005

CHRISTIANE MONTENEGRO BORTOLETO

MULTICAST SEMI-CONFIÁVEL PARA APLICAÇÕES MULTIMÍDIA DISTRIBUIÍDAS

Dissertação apresentada ao Programa de Pós-Graduação em Informática Aplicada da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Mestre em Informática Aplicada.

Área de Concentração: *Sistemas Distribuídos*

Orientador: Prof. Dr. Lau Cheuk Lung

Co-orientador: Prof. Dr. Frank A. Siqueira

CURITIBA

2005

Bortoleto, Christiane Montenegro

Protocolo de Multicast Semi-Confiável para Aplicações Multimídia Distribuídas. Curitiba, 2005. 88p.

Dissertação – Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Informática Aplicada.

1. Multicast 2. Multicast Semi-Confiável 3. Multimídia I.Pontifícia Universidade Católica do Paraná. Centro de Ciências Exatas e de Tecnologia. Programa de Pós-Graduação em Informática Aplicada II-t

Página reservada à ata de defesa e termo de aprovação, que serão fornecidos pela secretaria após a defesa da dissertação e efetuadas as correções solicitadas.

“Existem mais pessoas que desistem do que pessoas que fracassam”.

(*Henry Ford*)

Agradecimentos

Agradeço em primeiro lugar a Deus, minha fonte de força e inspiração.

Agradeço aos professores Lau Cheuk Lung e Frank A.Siqueira por toda a ajuda e aconselhamento durante o desenvolvimento do protocolo. Agradeço também ao Alysson pelo empurrão inicial neste trabalho e todas as dicas preciosas durante todo o processo. Agradeço ainda ao Fábio Favarim, pela instalação e configuração do Simmcast, sem o qual seria impossível desenvolver e simular o protocolo.

Agradeço ao professor Carlos Maziero pelo apoio para que eu continuasse com este trabalho quando tive que mudar de cidade.

Agradeço ao Marinho Barcellos, criador do Simmcast, pela atenção com que resolveu todas as minhas dúvidas a respeito de seu simulador.

Agradeço aos membros da banca por aceitarem o convite de participar dela.

Agradeço à minha família, em especial meus pais, Luiz e Tânia, e minha avó, Orlanda, por ter insistido tanto para que eu seguisse em frente com o Mestrado mesmo com todos os obstáculos que surgiram.

Agradeço ao meu noivo querido, Rodrigo, pela paciência e carinho com que me aturou nos momentos mais críticos.

Agradeço ao meu colega, Francisco Carlos Lajús, por todo o incentivo e palavras de motivação durante esta longa caminhada.

E, finalmente, agradeço ao meu irmão, Edu, pelo auxílio técnico de última hora na hora de gravar o CD com apresentação da defesa.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

ACK	*Acknowledgement*
AER	*Active Error Recovery*
API	*Application Program Interface*
ARQ	*Automatic Repeat reQuest*
ARM	*Active Reliable Multicast*
FEC	*Forward Error Correction*
FIFO	*First In First Out*
GOP	*Group of Pictures*
IETF	*Internet Engineering Task Force*
LBRM	*Log-Based Receiver-Reliable Multicast*
LMS	*Light-weight Multicast Services*
MPEG	*Motion Picture Experts Group*
NACK	*Negative Acknowledgement*
OMG	*Object Management Group*
PGM	*Pragmatic General Multicast*
QMTP	*Quasi-reliable Multicast Transport Protocol*
QoS	*Quality of Service*
ReMIOP	*Reliable Multicast Inter-ORB Protocol*
RMANP	*Reliable Multicast Active Network Protocol*
RMCM	*Reliable Multicast for Core-based Multicast Trees*
RMRG	*Reliable Multicast Research Group*
RMTP	*Reliable Multicast Transport Protocol*
RMTP II	*Reliable Multicast Transport Protocol II*
RTP	*Real-time Transport Protocol*
RTT	*Round Trip Time*
SRM	*Scalable Reliable Multicast*

SRP	*Selective Retransmission Protocol*
TMTP	*Tree-based Multicast Transport Protocol*
TRAM	*Tree-based Reliable Multicast*
TUNA	*TUNAble Quasi-Reliable Multicast Protocol*
UDP	*User Datagram Protocol*
UMIOP	*Unreliable Multicast Inter-ORB Protocol*

Resumo

A transmissão de vídeo em redes de larga escala é uma tarefa bastante complexa devido aos requisitos de qualidade impostos pelas aplicações multimídia que se utilizam deste recurso. Além disso, muitas destas aplicações são interativas, exigindo que o atraso entre os pacotes seja reduzido. Congestionamentos, longos atrasos e falta de continuidade na entrega de pacotes, entre outros problemas, podem prejudicar de forma irreparável o resultado percebido pelo usuário final. Outra particularidade das aplicações multimídia distribuídas é que elas toleram atrasos pequenos, pois estes não são significativos para a percepção do usuário. Segundo [ABE00], a tolerância humana situa-se no intervalo de 40 a 600ms, de acordo com a aplicação.

Para atender às necessidades particulares das aplicações multimídia distribuídas, é proposto um protocolo multicast semi-confiável que objetiva incrementar a qualidade de fluxos (*streams*) de vídeo transmitidos em sistemas de larga escala sem sobrecarregar a fonte do vídeo e a rede de comunicação. Empregando o protocolo de comunicação IP multicast e tendo como base a hierarquia de quadros do padrão MPEG (*Motion Picture Experts Group)*, este protocolo avalia a necessidade de efetuar a retransmissão de pacotes perdidos com base na capacidade do quadro MPEG correspondente em incrementar a qualidade do fluxo de vídeo. O protocolo proposto emprega os receptores vizinhos para efetuar a retransmissão, permitindo que os quadros perdidos sejam recuperados mais rapidamente, o que é essencial para que estes sejam recebidos em tempo para serem exibidos. Além disso, esta estratégia evita sobrecarregar a fonte do fluxo de vídeo, tornando o protocolo mais escalável que as abordagens tradicionais que realizam a retransmissão a partir da origem. Os resultados das simulações apresentadas mostram que o protocolo proposto provê uma melhor qualidade de imagem aos receptores de fluxos de vídeo.

Palavras-Chave: Multicast, Multicast Semi-Confiável, Multimídia.

Abstract

Large-scale video transmission is a really hard task due to quality requirements of the application that uses this kind of feature. Besides, many applications are interactive, what demands reduced delay of the packets. Congestion, long delays and no stable delivery of the packets, besides other problems, can prejudice in an unrecoverable way the results observed by the final user. Another feature of distributed multimedia application is that reduced delays are not significant to the user's sensitivity. According to [ABE00], human tolerance can accept a 40 to a 600ms delay, depending on the application.

To support singular distributed multimedia application needs it is proposed a semi-reliable multicast protocol that aims to increase the quality of video streams transmitted in large-scale systems without overloading the video source and the communications network. This protocol, which is based on the IP multicast protocol and the MPEG standard, evaluates the necessity of retransmitting lost packets taking into account the capacity of the corresponding MPEG frames to improve the quality of the video stream. The proposed protocol relies on the neighboring receivers for retransmitting lost packets, resulting in much faster recovery, which is vital in order to receive retransmitted packets on time to be exhibited. Besides, this strategy avoids overloading the video source, making it more scalable than the traditional approach of retransmitting from the source. Simulation results presented in this paper show that the proposed protocol delivers video streams to the receivers with a better image quality.

Keywords: Multicast, Semi reliable Multicast, Multimedia.

Capítulo 1

Introdução

Aplicações multimídia distribuídas são intensamente desenvolvidas em diversas plataformas e topologias. No entanto, a estrutura de comunicação comum oferece riscos à qualidade do conteúdo multimídia em função da implementação das políticas de transmissão, roteamento (descarte de pacotes) e da complexidade na formatação dos dados. A camada de transporte pode implementar procedimentos para assegurar uma transmissão confiável dos pacotes, desde que adotado um protocolo adequado ao tipo de aplicação em questão. O termo "confiável", neste caso, não significa que não ocorrerão erros, mas sim que eles serão detectados caso ocorram e tratados de forma satisfatória.

Além disso, as aplicações multimídia distribuídas possuem requisitos bastante particulares. Um fluxo (*stream*) de áudio/vídeo, por exemplo, requer que os dados sejam recebidos no momento certo mas pode aceitar a perda de alguns dados. Um pacote que chega após um determinado prazo não contribui em nada na reprodução, ou seja, o efeito observável seria o mesmo que se o pacote tivesse sido perdido. Outro problema é o da variação do retardo, também chamado de *jitter*. Para aplicações multimídia, as informações devem ser apresentadas em intervalos regulares a fim de evitar a falhas de sincronização na reprodução. Quando essa variação é muito grande, há a necessidade de sincronizar o sinal através de uma operação chamada de sincronização intra-mídia, o que normalmente é realizado no receptor através da utilização de *buffers* e do atraso na reprodução. Contudo, apesar de melhorar a qualidade da reprodução, está técnica aumenta o atraso total no receptor, o que pode ser um problema caso a aplicação seja interativa.

Em muitas aplicações multimídia distribuídas (tais como *groupware*, CSCW, videoconferência, etc.), a transmissão de mídia pela Internet tem como destino não apenas um

receptor, mas um grupo de receptores. Devido a isto, abstrações de multicast têm recebido especial interesse da comunidade por apresentar um desempenho melhor que comunicações ponto-a-ponto. Na literatura, existe uma quantidade significativa de trabalhos em multicast melhor esforço (*best-effort*) para aplicações multimídia distribuída [BAR98] e multicast confiável [FLO95, LIA98, HAD94, BAR98] para aplicações que impõem confiabilidade na entrega das mensagens (ex: aplicações tolerantes a faltas). No entanto, entre esses dois extremos, encontram-se os protocolos de multicast semi-confiável, um tema de pesquisa bem mais recente e ainda pouco explorado [PIE00, MAN00, PER03, XU97, SCH99]. Multicast semi-confiável é uma abstração de comunicação em que os dados (pacotes) são qualificados, normalmente pela aplicação, antes da sua transmissão, estabelecendo níveis de importância ou prioridade na retransmissão de pacotes para correção de erros. Este conceito é adequado para a proposta desta dissertação, uma vez que as aplicações multimídia não precisam que todos os pacotes sejam recebidos pela aplicação para que o prejuízo para o usuário final seja mantido dentro de um limite aceitável. Desta forma, o objetivo principal deste trabalho é projetar e implementar um protocolo de difusão semi-confiável, baseado no receptor, mais eficiente e escalável, projetado para aplicações multimídia distribuídas baseada em grupo (ex: videoconferência, difusão de vídeo digital, etc.). O protocolo apresenta uma inovação em relação àqueles encontrados na literatura por ser especificado para difusão de fluxos de vídeo empregando codificadores que apresentem alguma forma de hierarquização de quadros, tal como no padrão MPEG, permitindo, desta forma, estabelecer níveis de prioridade na correção de quadros perdidos.

Além disso, buscou-se:

- Estudar o padrão MPEG e avaliar as características deste padrão que o levam a ser vantajoso para a transmissão de multimídia para aplicações distribuídas;
- Estudar os protocolos de multicast confiável e semi-confiável, verificando as propostas mais adequadas para a transmissão de multimídia;
- Estudar o simulador Simmcast [BAR01] para utilizá-lo na implementação e testes do protocolo proposto;
- Avaliar os resultados obtidos para o protocolo proposto e analisar sua aplicabilidade para redes de larga escala, como a Internet, por exemplo.

O capítulo 2 desta dissertação apresenta os principais conceitos relacionados ao padrão MPEG. São apresentados os principais padrões desta família MPEG e é explicado o funcionamento do algoritmo de compressão.

O capítulo 3 apresenta as características da transmissão multicast e multicast confiável, do qual também são citados alguns exemplos. É tratado ainda o conceito de multicast semi-confiável, trazendo alguns dos modelos que são encontrados na literatura e explicando-se as principais características de cada um deles.

O capítulo 4 trata do protocolo proposto por este trabalho. Neste capítulo, definem-se as premissas do trabalho e mostram-se o algoritmo de funcionamento do protocolo e a forma como são feitas as retransmissões e a entrega dos dados para aplicação.

O capítulo 5 apresenta o instrumento que foi utilizado para implementação e simulação do protocolo proposto, o Simmcast [BAR01], destacando as principais vantagens do uso desta ferramenta.

O capítulo 6 traz o conjunto de testes que foram realizados para avaliação do desempenho do protocolo proposto, com as discussões dos resultados obtidos.

O capítulo final traz a conclusão do trabalho e os possíveis trabalhos futuros que podem ser desenvolvidos a partir do que está proposto neste trabalho.

Capítulo 2

O Padrão MPEG

Para que um sinal de áudio/vídeo possa ser transmitido em uma rede, é necessário que ele seja codificado. Existem muitos esquemas de codificação tanto para áudio quanto para vídeo, com diferentes características de desempenho, complexidade e tempo de processamento. Comparando-se aos fluxos de texto e gráficos, o fluxo de um sinal de vídeo gerado por uma aplicação típica pode ser considerado de grande volume e, por isso, a compressão de dados, especialmente de áudio e vídeo, é necessária para se aperfeiçoar a utilização da largura de banda existente nas redes atuais e limitar a demanda por maior capacidade de armazenamento e transferência.

Os padrões mais importantes para compressão de vídeo são a família MPEG (*Motion Picture Experts Group*), estabelecida pela União Internacional de Telecomunicação (ITU). Os codificadores MPEG oferecem três grandes vantagens sobre os demais esquemas existentes: compatibilidade universal, grandes taxas de compressão e perda aceitável de qualidade na imagem final exibida [BER05].

O padrão MPEG trata separadamente vídeo e áudio, especificando como estes sinais são associados e sincronizados, possuindo assim três níveis: a camada de sistema, a camada de vídeo e a camada de áudio. A camada do sistema contém as informações sobre sincronização, acesso aleatório, administração *buffers* e uma marcação de tempo (*time code*) para cada quadro de vídeo [MAR99].

Neste capítulo, são explicados os principais conceitos e técnicas relativas a este padrão.

2.1. Família MPEG

Existem vários padrões diferentes dentro da família MPEG.

O MPEG-1, criado em 1991 [MAR99], é o padrão MPEG inicial, desenvolvido para codificar vídeos inteiros. Este padrão tem a taxa de bits de 1.5 Mbps, um quadro de 352x240 *pixels*, inclui três padrões de áudio e a maioria dos sistemas de vídeo usa a camada de áudio 1 ou 2 do MPEG-1. A camada de áudio 3 do MPEG-1, mais conhecida como MP3, é usada amplamente para áudio na Internet [VID05].

O MPEG-2 resulta em taxas de compressão de 3Mbps a 100 Mbps e suporta taxas de dados maiores apresentando, dessa forma, maior qualidade. O MPEG-2 é o padrão utilizado em aparelhos de DVD, na maioria dos satélites digitais da América do Norte e no sistema de TV digital norte-americana [VID05].

O MPEG-3 foi estabelecido para ser utilizado na televisão de alta definição (HDTV), mas percebeu-se que o MPEG-2 suportava perfeitamente do volume de informação requerido pela HDTV e este padrão foi abandonado [BER05].

O MPEG-4 tem o objetivo de ajustar-se melhor à Internet. Este padrão permite a transmissão de vídeo com uma qualidade superior comparada ao MPEG-1 com uma taxa de bits muito menor. O MPEG-4 também suporta uma grande variedade de elementos transmitidos separadamente e combinados para formar o quadro do vídeo, isto é, permite a manipulação de objetos dentro do *stream* (adição, subtração, manipulação de objetos, etc.).

O MPEG-7 é um padrão para descrição de objetos multimídia. Não é um formato de codificação de vídeo, mas uma maneira de descrever elementos em um fluxo multimídia de modo que possa ser acessado via banco de dados [MAR03].

2.2. Compressão

O objetivo de um sistema para compressão de vídeo é reduzir a taxa de transmissão, removendo a redundância e/ou informações de menor importância do sinal antes da transmissão [MAR99]. A compressão ocorre no codificador do transmissor de vídeo digital. No lado do receptor, um decodificador reconstrói uma aproximação da imagem a partir da informação remanescente após o processo de compressão.

Há dois tipos de redundância em um sinal de vídeo [MAR99]: redundância espacial e redundância temporal. A primeira são as informações redundantes que aparecem em uma mesma imagem, como uma cor de fundo, por exemplo. A segunda é observada em dois quadros consecutivos, por exemplo, quando o fundo permanece o mesmo e algum objeto muda de posição na imagem.

Antes de compreender o processo de compressão para eliminação da redundância, convém explicar alguns dos termos utilizados. Uma imagem é dividida em fatias (*slice*), como mostra a figura 2-1. As fatias, em unidades de 16 x 16 *pixels*, chamadas macroblocos. Os macroblocos, por sua vez, são divididos em blocos de 8x8 *pixels* [BER05].

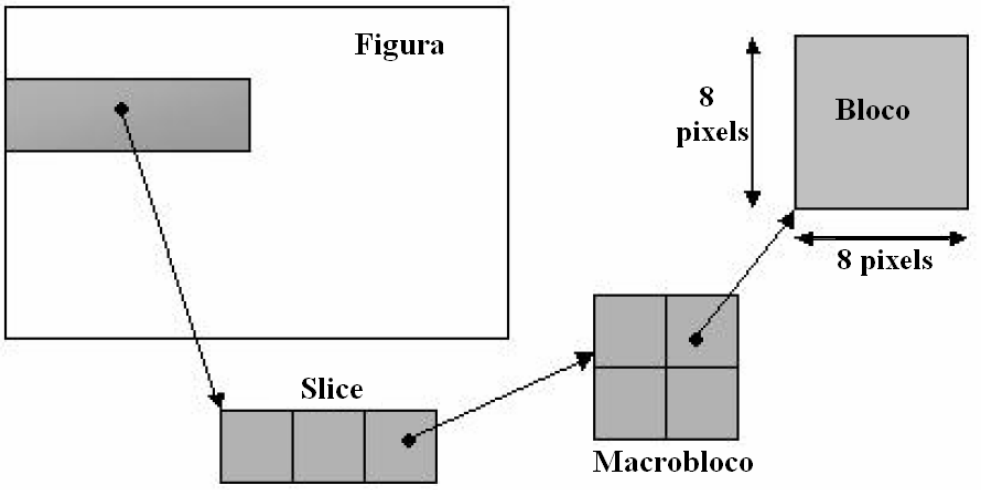

Figura 2-1. Estrutura da imagem [FER04]

O primeiro processo de compressão efetuado em um sistema MPEG é a sub-amostragem de cor. Esta sub-amostragem baseia-se na característica do sistema visual humano que é mais sensível a mudanças de luminância e menos sensível a variações de crominância [CAS05]. Como o objetivo é a compressão, o MPEG opera em um espaço de cores que permita tirar maior proveito desta propriedade da visão humana. Dessa forma, MPEG usa o espaço YUV, onde Y é uma componente de luminância e U e V, componentes de crominância. A Figura 2-2 mostra algumas representações de macrobloco depois de usadas diferentes taxas de sub-amostragem.

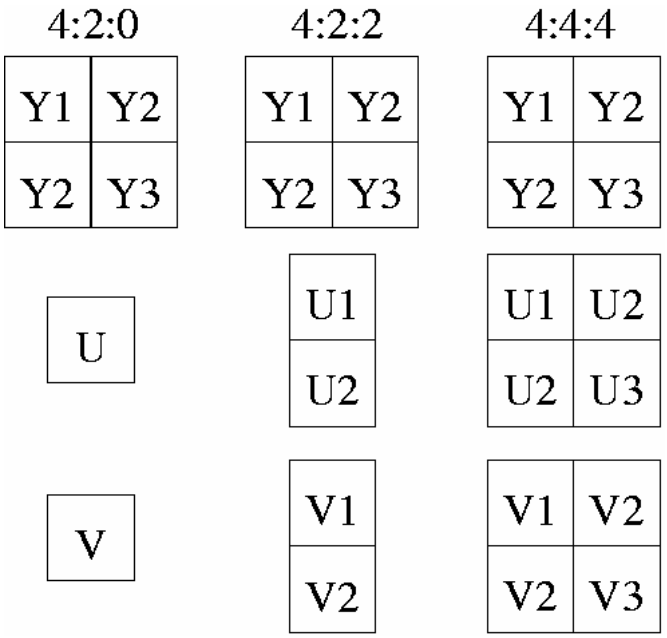

Figura 2-2. Macroblocos MPEG

Após a sub-amostragem, busca-se a eliminação da redundância espacial. Para isso, aplica-se a transformada discreta de co-seno (DCT) [PER99]. A DCT efetua o mapeamento entre a representação de uma imagem bidimensional e a sua representação no domínio da freqüência em cada um dos blocos de 8X8 *pixels*. O bloco resultante da transformação é um bloco também 8X8, mas composto por coeficientes transformados. A DCT não reduz diretamente o número de bits requerido para representar o bloco. A redução no número de bits vem do fato de que a transformação tende a concentrar a energia nos coeficientes de baixa freqüência. Os demais coeficientes possuem valor próximo a zero, podendo ser descartados. A distribuição não uniforme dos coeficientes transformados é um resultado da redundância espacial presente no bloco original [CAS05].

O próximo passo é a eliminação da redundância temporal, que é feita com a predição por compensação de movimento [MAR99]. Este tipo de predição busca compensar movimentos de translação dos blocos entre quadros consecutivos. Pela comparação dos macroblocos de um quadro com os de outro, verifica-se se a correlação entre os *pixels* destes quadros é alta, ou seja, se eles têm conteúdo similar ou idêntico. As coincidências podem ser ignoradas, pois é possível codificar um quadro através do cálculo do vetor de deslocamento em relação ao anterior, reduzindo-se significativamente a quantidade de informação a ser armazenada. Apenas um vetor de movimento é estimado, codificado e transmitido para cada

um destes blocos. O erro de estimação, ou seja, a diferença entre um quadro e sua estimação também é transmitido [PER99]. Este mecanismo é ilustrado nas Figuras 2-3 e 2-4.

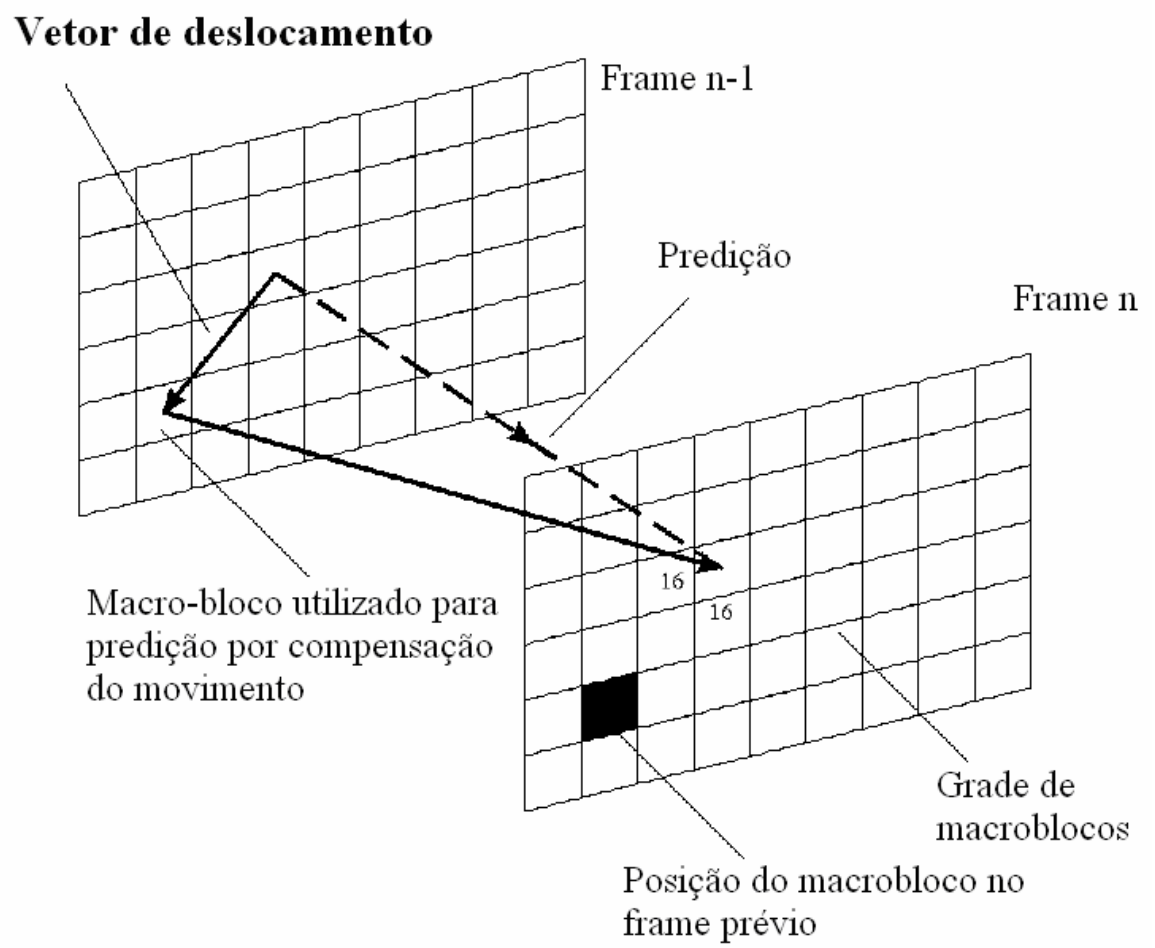

Figura 2-3. Predição por compensação de movimento – conceito [CAS05]

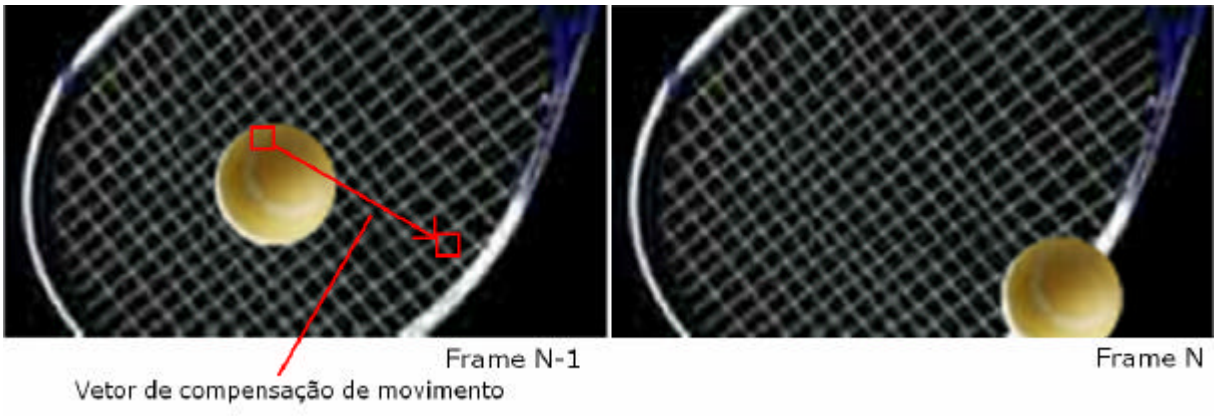

Figura 2-4. Predição por compensação de movimento – exemplo

É possível observar as vantagens do uso da predição de movimento compensado na Figura 2-5. O quadro (a) é um quadro a ser codificado em um dado instante de tempo e o quadro (b) é o quadro no instante anterior. Os vetores de movimento também podem ser vistos no quadro (b), mostrando o deslocamento médio de cada macrobloco no instante atual em relação ao anterior. O quadro (c) mostra a diferença simples entre os blocos, sem considerar o vetor de movimento, e o quadro (d) mostra o erro de predição entre os quadros considerando o vetor de movimento. Comparando-se os quadros, percebe-se que o sinal a ser codificado e transmitido é muito menor com o uso da compensação de movimento (quadro (d)) em comparação à diferença simples (quadro(c)).

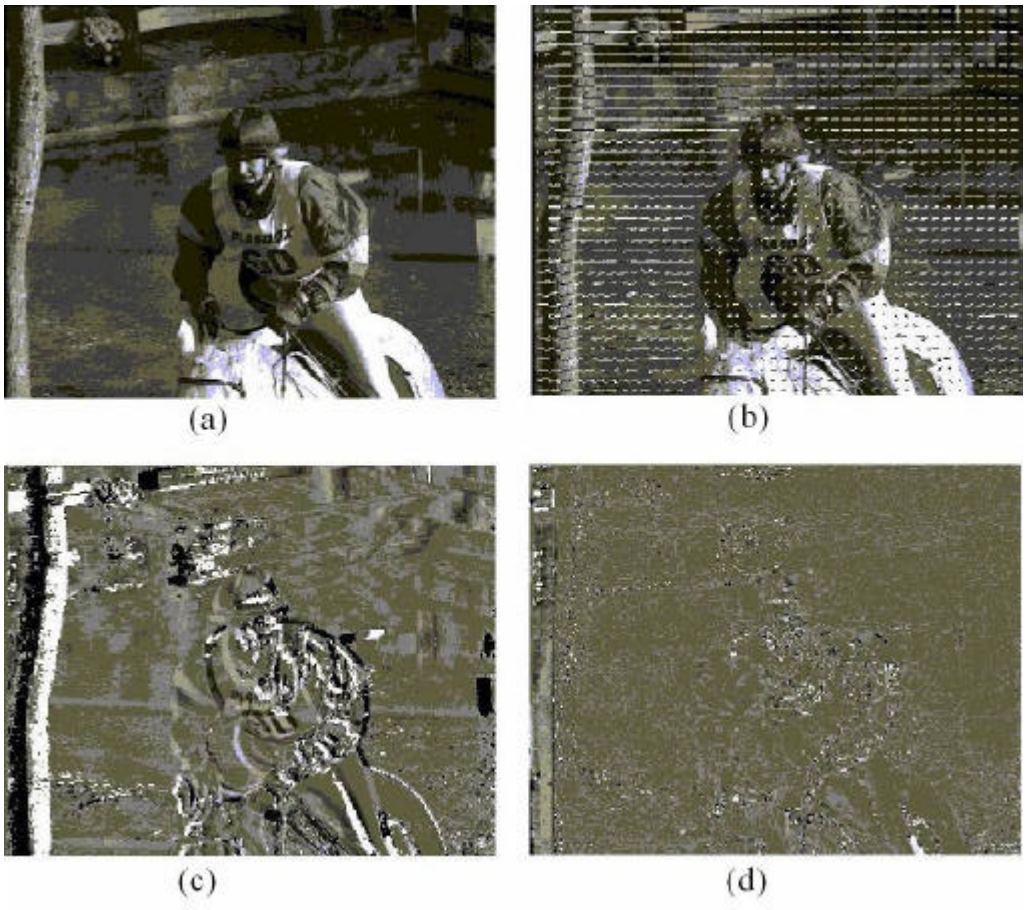

Figura 2-5. Efeito do vetor de movimento compensado [PER99]

Um método para determinar o movimento que ocorreu entre o bloco que está sendo codificado e o bloco no quadro de referência é a busca *block-matching*, em que várias tentativas de deslocamento são testadas e o melhor deslocamento é selecionado. Este processo é ilustrado pela figura 2-6.

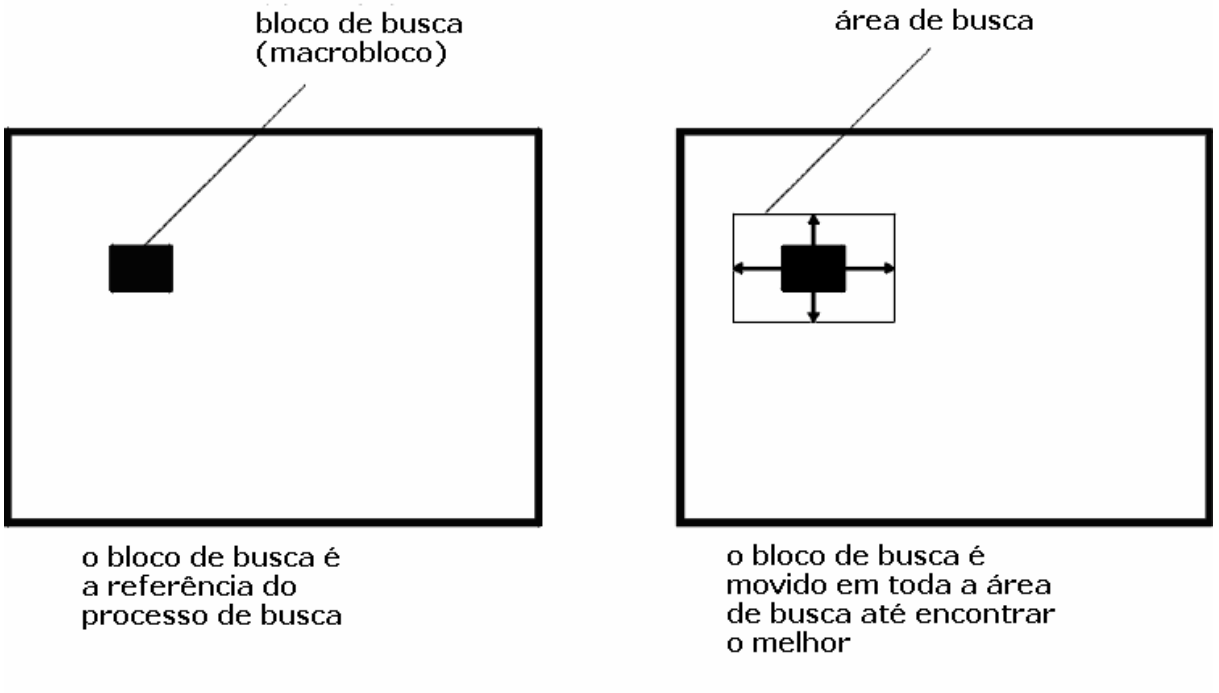

Figura 2-6. Processo de busca *block matching* [CAS05]

Outra forma de predição utilizada no padrão MPEG é a predição bidirecional, feita a partir de um quadro prévio e de um quadro consecutivo. Esta predição estabelece uma combinação linear destes dois quadros, interpolando os dois deslocamentos, como mostra a figura 2-7 [CAS05].

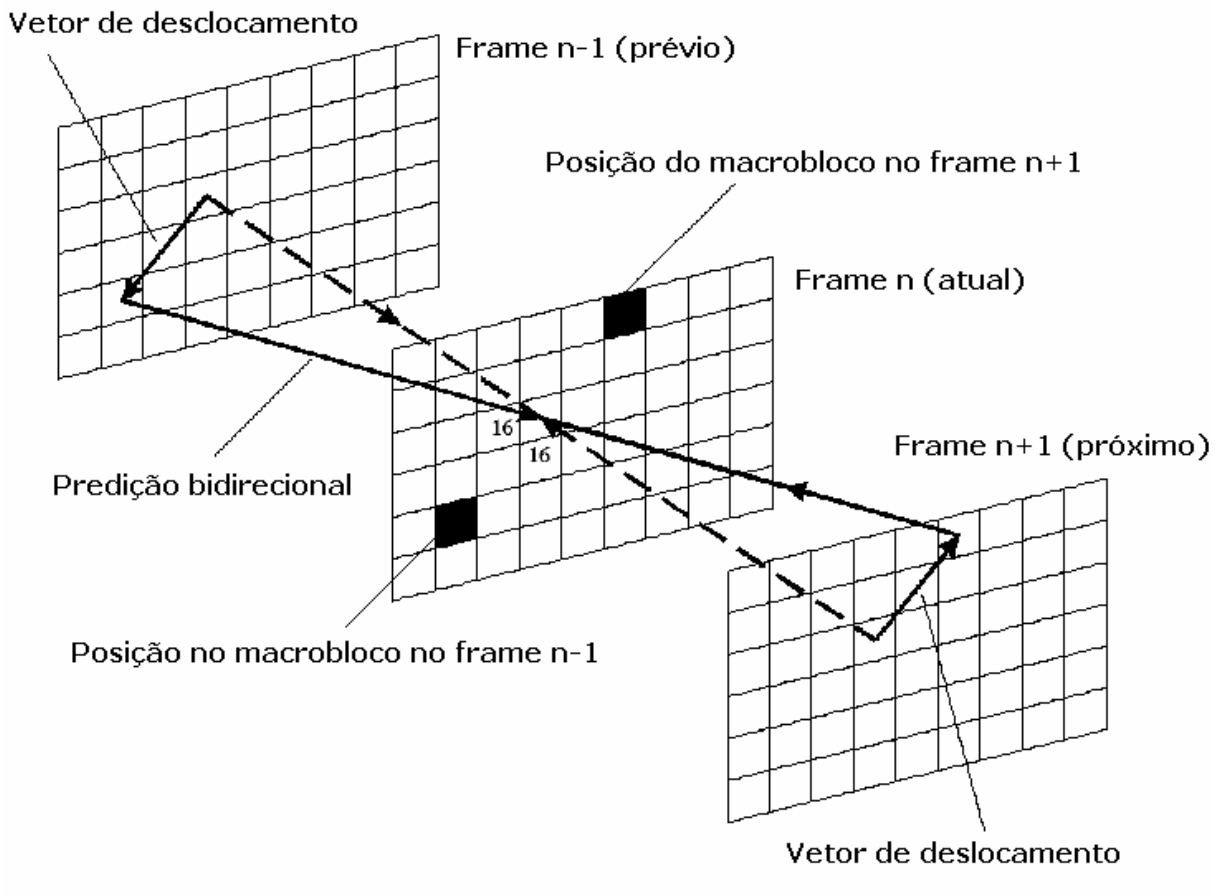

Figura 2-7 Predição bidirecional com compensação de movimento [CAS05]

O principal componente de um *stream* MPEG é o GOP (*Group of Pictures*), uma série, de tamanho N, de quadros onde cada um consiste em um cabeçalho da figura e os seus atuais dados e pode ser definido como a menor unidade do MPEG que ainda é passível de decodificação. Dentro do GOP, cada quadro recebe um número que determina o momento em que este deve ser reproduzido. O GOP compreende um quadro I (*Intracoded*), que são imagens completas codificadas individualmente, e seus referentes: P (*Predictive*), quadros codificados com predição relativa ao último quadro, e B (*Bidirectional*), quadros que carregam as diferenças entre o último e o próximo quadro.

Os quadros I são inseridos no *stream* de saída a uma taxa específica (por exemplo, uma ou duas vezes por segundo), com quadros P e B entre eles. Eles são os únicos completos no fluxo do MPEG, por isso os maiores arquivos, e têm a informação completa, o que os qualifica como pontos de entrada no fluxo, através de acesso randômico. Os quadros I não são

dependentes de outros quadros para serem decodificados, mas são necessários para decodificação de quadros P e B.

Os quadros P são baseados em previsão antecipada (*forward prediction*) e previstos pelo último quadro de referência, que pode ser um quadro do tipo ‘I’ ou ‘P’. O quadro do tipo ‘P’ possui a referência do vetor de movimento no passado, utilizado para referenciar o bloco do quadro anterior na mesma posição do quadro atual. Como apenas as mudanças entre o novo quadro e o quadro de referência precisam ser armazenadas, estes quadros são menores que os quadros do tipo ‘I’. Os quadros P são necessários para decodificação de quadros do tipo ‘B’.

Quadros do tipo ‘B’ (*backward prediction*) referem-se ao mesmo tempo ao quadro de referência seguinte e ao anterior, que podem ser do tipo ‘I’ ou ‘P’, e são os mais comprimidos do fluxo. O quadro do tipo ‘B’ possui a referência do vetor de movimento no futuro, utilizado para referenciar o bloco do próximo quadro na mesma posição do quadro atual. Como contêm muito pouca informação, quadros B nunca são usados como quadro de referência para os demais.

Este método de codificação faz com que alguns quadros sejam mais importantes que os outros. Se um quadro I é perdido durante a transmissão, não será possível decodificar os quadros que cheguem antes do próximo quadro I. A relação entre os quadros em um fluxo de vídeo MPEG é ilustrada na Figura 2-8.

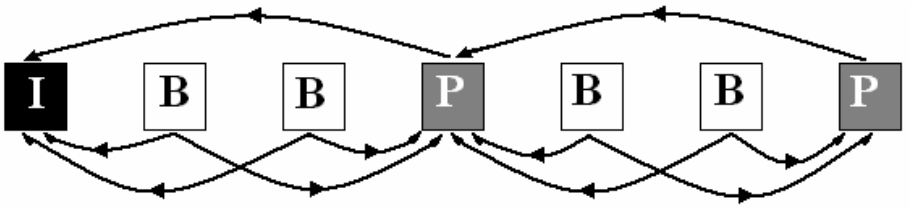

Figura 2-8. Relacionamento de quadros IPB

2.3. Conclusão

Este capítulo apresentou as principais características do padrão MPEG e seu esquema de codificação de sinais digitais de vídeo onde a compressão é utilizada para aumentar a eficiência dos espaços disponíveis.

Foi mostrado que nos algoritmos de codificação MPEG, o primeiro quadro da seqüência de vídeo, quadro ‘I’ (*intraframe*), é codificado sem referência a nenhum quadro, ao contrário dos quadros ‘P’ e ‘B’. Para os quadros ‘P’, o último quadro (‘I’ ou ‘P’) é armazenado e é estimado um vetor de movimento para cada macrobloco, que é codificado, agrupado ao erro de predição calculado e transmitido no *stream*. Para quadros ‘B’, um quadro é interpolado entre um quadro anterior e um quadro seguinte e a diferença entre ele e o quadro original é transmitida.

Com base nas características do fluxo MPEG mostradas neste capítulo, percebe-se que é possível estabelecer prioridades diferentes para cada um dos quadros que são transmitidos. Isto permite aplicar critérios para garantir a entrega daqueles considerados mais importantes, idéia fundamental para o protocolo proposto, conforme será mostrado no capítulo 4.

O capítulo seguinte descreve a idéia central dos protocolos de multicast, multicast confiável e semi-confiável e ilustra a adequação do último para transmissão multimídia.

Capítulo 3

Comunicação em Grupo

Para diversos tipos de aplicação, a comunicação entre os diversos *hosts*[1] conectados na rede é essencial. A forma mais simples de comunicação o modelo *unicast*, também chamado um-para-um, ponto-a-ponto ou ainda *peer-to-peer*, onde um único emissor envia mensagens a um único receptor. No entanto, em algumas situações é desejável que um emissor possa comunicar-se com vários receptores. Este mecanismo de comunicação que trata múltiplas conexões é chamado de *comunicação em grupo*. Este capítulo trata das principais propriedades e características deste tipo de comunicação, introduzindo conceitos que são utilizados amplamente nesta dissertação.

3.1. Grupo

Um grupo é uma coleção de processos que interagem entre si em algum sistema [TAN95] onde a propriedade mais importante é que, quando uma mensagem é enviada para o grupo, todos os seus membros a recebem. Esta forma é chamada comunicação um-para-muitos e está ilustrada na Figura 3-1.

Grupos são dinâmicos, a qualquer momento grupos existentes podem ser destruídos e novos grupos, criados. Um *host* pode juntar-se ou abandonar um grupo e pode ser membro de um ou vários grupos ao mesmo tempo. Por isso, são necessários mecanismos para gerenciamento dos grupos e seus membros.

[1] *Host* é qualquer máquina ou *gateway* que não sejam os roteadores multicast.

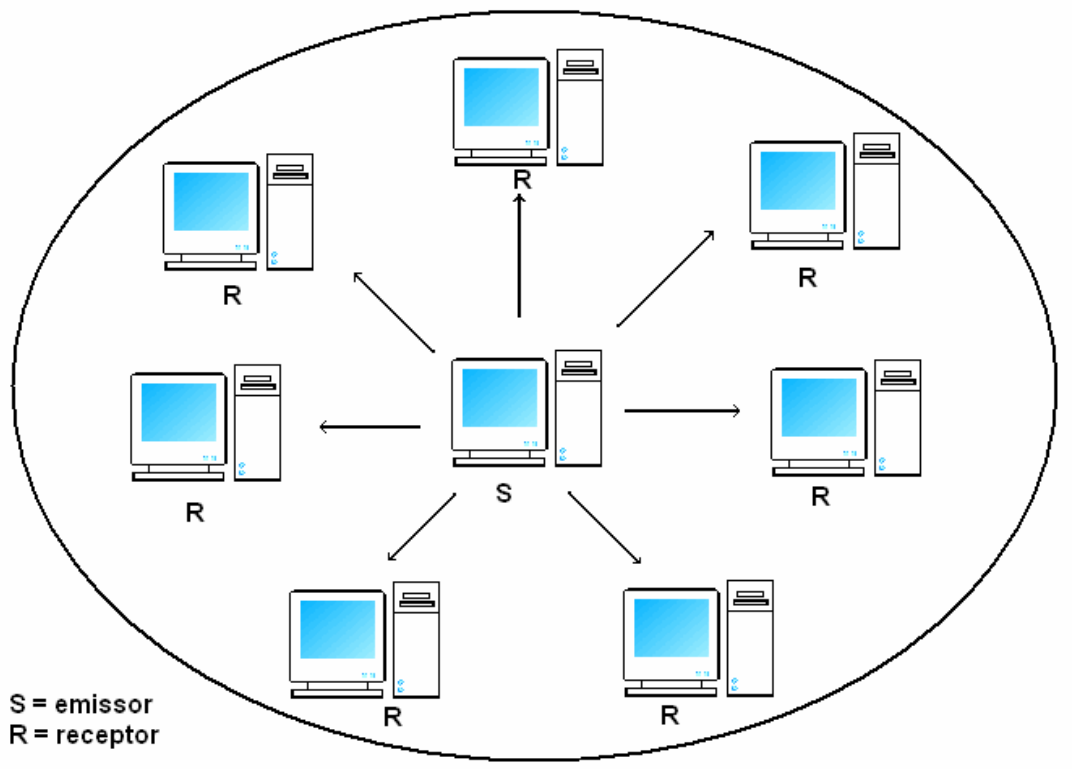

Figura 3-1. Comunicação um-para-muitos

As principais características a serem consideradas no projeto voltado para comunicação em grupo são: endereçamento, confiabilidade, ordenação, semântica de entrega, semântica de resposta e estrutura do grupo. [KAA94]

Quanto ao endereçamento, é possível criar um endereço especial de rede pelo qual diversos *hosts* possam receber as mensagens, ou seja, de forma que um pacote enviado para este endereço seja recebido por todos os membros do grupo. Esta técnica é o *multicast*. Em redes onde não é possível implementar o multicast, pode-se ter o *broadcast*, onde os pacotes que têm um determinado endereço são recebidos por todos os *hosts.* No entanto, esta técnica não é tão eficiente, pois todos os *hosts* da rede recebem a mensagem e cada um precisa avaliar se ela é endereçada ao grupo que ele pertence. Caso não seja, a mensagem é descartada, mas para isso já houve um gasto desnecessário de tempo e recursos. Em redes que apenas suportam comunicação *unicast*, a comunicação em grupo é feita gerando uma mensagem individual para cada membro, repetidamente. Neste caso, uma lista de endereços de cada membro deve ser mantida pelo servidor de grupo.

A confiabilidade trata da recuperação de falhas na entrega das mensagens e está relacionada com a semântica de resposta, que trata da expectativa do emissor em relação ao

retorno dos receptores. Aplicações diferentes requerem diferentes graus de confiabilidade. O emissor de uma mensagem multicast pode especificar o número de receptores, dos quais é esperada uma resposta. Nas comunicações um-para-muitos, o grau de confiabilidade pode ser expresso por: [SIN96]

- 0: Nenhuma resposta é esperada pelo emissor, dos receptores. Típica em aplicações assíncronas onde o emissor não precisa esperar pelos receptores.
- 1: O emissor espera uma resposta de um dos receptores pelo menos. Típica nas aplicações do tipo gerenciador de serviços onde o gerente solicita aos vários servidores e o primeiro a responder processa a requisição.
- m-em-n: O grupo consiste de n membros e o emissor espera a resposta de m (1<m<n) receptores. Típico em aplicações que utilizem algoritmos de consenso de maioria (votação) para controle de consistência de servidores replicados.
- todos: O emissor espera resposta de todos os membros do grupo. Típica em aplicações onde existam replicas de dados a serem atualizados em diferentes servidores e todas devem ser efetuadas ao mesmo tempo para garantir consistência

A ordenação, tratada na seção 3.3, diz respeito à entrega das mensagens na seqüência apropriada.

A semântica de entrega está relacionada à entrega bem sucedida de uma mensagem ao grupo. A entrega atômica, onde ou todos ou nenhum processo recebe a mensagem, é considerada a semântica ideal, mas sua implementação é mais complicada.

A estrutura do grupo diz respeito à classificação dos grupos em abertos ou fechados, como será discutido na seção seguinte.

3.2. Gerenciamento de Grupos

Como mostra a Figura 3-2, os grupos podem ser classificados em:

- Abertos: qualquer *host* da rede pode enviar mensagem para os processos do grupo.
- Fechados: apenas os membros do grupo podem trocar mensagens entre si, embora seja possível haver *unicast* entre um membro e um *host* externo ao grupo.

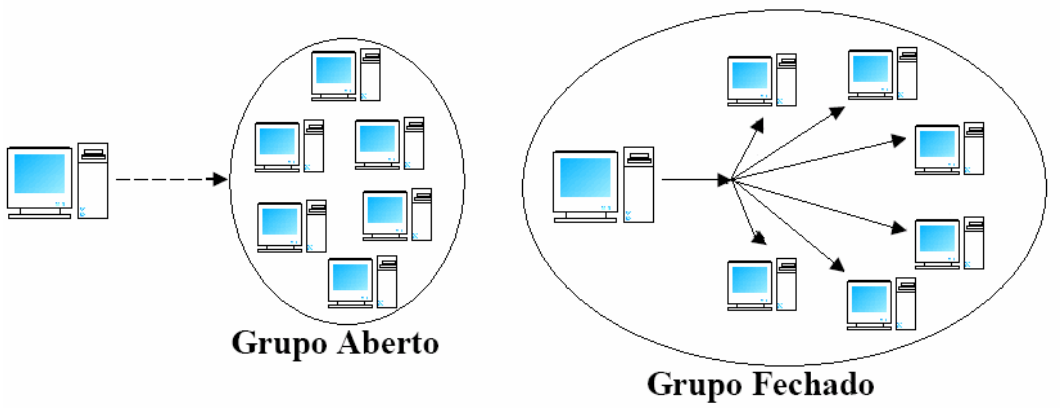

Figura 3-2. Tipos de grupo

A escolha adequada entre grupo aberto ou fechado está relacionada com o tipo de aplicação que será executada. Por exemplo, em uma aplicação de processamento paralelo, ou seja, processos trabalhando em conjunto para atender a um objetivo, não é necessária a interação de um processo externo ao grupo, sendo um bom exemplo de grupo fechado. Já uma aplicação onde um grupo é formado por servidores replicados pode receber mensagens de um cliente externo solicitando uma requisição de serviço.

Outra classificação quanto à organização está relacionada à estrutura interna do grupo. Em alguns grupos os processos são todos iguais, sem haver um "gerente" e com todas as decisões tomadas coletivamente. Tais grupos são os chamados grupos *pares*. Em outros, os grupos *hierárquicos*, há um processo coordenador que determina as ações dos demais. Neste caso, toda requisição é enviada a este coordenador e ele decide qual membro do grupo deve atendê-la. Estas duas formas de organização são ilustradas pela Figura 3-3.

A vantagem dos grupos pares é que se um *host* falhar, o funcionamento do grupo continua normalmente. Já nos hierárquicos, se o coordenador do grupo falhar, o grupo inteiro fica comprometido, havendo um ponto único de falha. A desvantagem dos grupos pares é que a tomada de decisão pode ser mais lenta e gerar mais carga sobre o *hosts*, já que a responsabilidade é compartilhada por todos os membros do grupo. Nos grupos hierárquicos, isso não acontece porque as decisões são tomadas pelo coordenador do grupo.

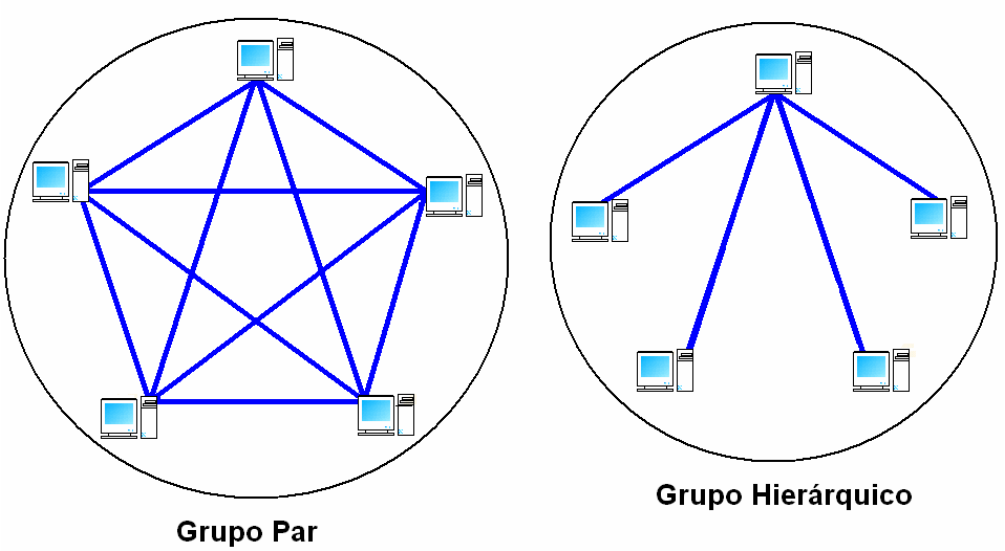

Figura 3-3. Grupo par e grupo hierárquico

Para controlar a junção e dissociação de membros de um grupo, há dois enfoques possíveis. O primeiro é centralizar este controle em um único servidor, um método simples, eficiente e fácil de ser implementado, mas que apresenta a desvantagem do ponto único de falha. O outro enfoque é gerenciar os grupos de forma distribuída.

Em um grupo aberto, um *host* externo ao grupo precisa apenas enviar uma mensagem anunciando sua presença para juntar-se ao grupo. Em um grupo fechado a única mensagem aceita vinda de um *host* externo é a solicitação para juntar-se ao grupo. Em ambos os casos, para abandonar o grupo basta o *host* encaminhar a ele uma mensagem de encerramento. Se um membro de um grupo falhar, ele deixa de fazer parte do mesmo. Como esse não é o procedimento padrão, os demais membros precisam descobrir esta saída por conta própria.

A partir do momento que um *host* se junta a um dado grupo ele deverá receber todas as mensagens enviadas ao grupo. Da mesma maneira, após deixar o grupo, ele não deve mais receber as mensagens endereçadas ao mesmo.

Para que seja possível a troca de mensagens dentro de um grupo ou o envio de mensagens de um *host* externo, no caso de grupos abertos, é preciso que haja uma maneira de identificar os grupos. Uma forma é atribuir ao grupo um único endereço, podendo ser multicast ou *broadcast*. Outra forma é enviar uma lista com todos os endereços (por exemplo, endereços IP) dos membros do grupo. Este método força todos os membros a conhecerem os demais participantes do grupo, deixando de ser transparente.

3.3. Propriedades da Comunicação em Grupos

As propriedades mais importantes da comunicação em grupo são: a atomicidade, a ordenação, a sobreposição e a escalabilidade. A atomicidade garante que as mensagens enviadas a um grupo ou são entregues corretamente a todos os processos do grupo ou a nenhum deles. O processo que enviou a mensagem recebe uma mensagem de erro se um ou mais participantes do grupo não a recebeu corretamente e os demais membros a ignoram.

A ordenação deve garantir que as mensagens cheguem na mesma ordem em que foram enviadas. Há quatro tipos de ordenação normalmente implementados na comunicação em grupo [TAN95]:

- Sem ordem, onde não há um mecanismo para garantir a entrega ordenada das mensagens. Tem o menor *overhead* porque não faz nenhum tipo de controle da ordem de entrega, mas não é adequado para várias aplicações.
- FIFO (*First In First Out*), que garante a entrega das mensagens na ordem em que foram enviadas.
- Causal, onde se utiliza o conceito de dependência entre as mensagens. No ordenamento causal, as mensagens estão em ordenamento FIFO e se um membro, após receber a mensagem M1, envia a mensagem M2, garante-se que os outros *hosts* recebam M1 antes de M2.
- Total, onde cada membro do grupo recebe todas as mensagens na mesma ordem.

A propriedade de sobreposição permite que um processo seja membro de diversos grupos ao mesmo tempo. E a escalabilidade busca garantir o bom funcionamento do grupo independente do número de participantes.

3.4. Conclusão

Este capítulo apresentou as principais características, mecanismos e propriedades da comunicação em grupos. Na forma mais geral, pode-se dizer que um grupo é um conjunto de processos com interesses em comum. Este conjunto pode mudar com o tempo, com a entrada ou saída de participantes. Um determinado processo pode pertencer a diferentes grupos. Em geral, os protocolos para este tipo de comunicação procuram garantir que uma mensagem, uma vez entregue a um participante de um determinado grupo, seja também entregue a todos

os demais (atomicidade). Outra propriedade em geral garantida pela comunicação em grupo é a ordem de entrega das mensagens aos diferentes processos.

As características da comunicação em grupo permitem que o processo de entrega de mensagens fique transparente à aplicação, já que ela não precisa ocupar-se do controle de membros. A aplicação apenas endereça suas mensagens ao grupo e deixa que o protocolo se ocupe da entrega.

O próximo capítulo descreve mais detalhadamente a tecnologia multicast, que permite a entrega de mensagens para um determinado grupo de *hosts* simultaneamente, de forma eficiente.

Capítulo 4

Multicast

Existem três tipos de transmissão em rede: as transmissões *unicast*, as transmissões *broadcast* e as transmissões *multicast*. Uma transmissão *unicast* ocorre quando se estabelece uma ligação entre duas máquinas, ponto-a-ponto. São transmissões muito comuns dentro de qualquer rede, mas podem se tornar bastante onerosas dependendo do tipo de tráfego necessário. As transmissões *broadcast* ocorrem quando se estabelece uma ligação em que os dados são transmitidos simultaneamente a todos os *hosts* da rede. O processo de *broadcast* é conhecido por difusão. Evoluindo-se este conceito chega-se à difusão seletiva, ou multicast.

A tecnologia multicast representa um serviço de rede no qual um único fluxo de dados, proveniente de uma determinada fonte, pode ser enviado simultaneamente para diversos receptores ou *hosts.* Cabe à infra-estrutura de rede transportar estes dados, replicando-o quando necessário, para todos os *hosts* interessados.

Os receptores são representados por um endereço de grupo ou endereço multicast, para onde a fonte envia os pacotes. Diversos *hosts* podem vincular e desvincular-se do grupo de forma dinâmica. Os dispositivos da rede devem fornecer um protocolo de roteamento para encaminhar os pacotes aos membros do grupo.

Uma solução TCP[1] (*Transmission Control Protocol*) para multicast não é possível: um único protocolo genérico não é capaz de atender aos requisitos de todos os tipos de aplicações multicast. Como a maioria dos mecanismos de transporte mais recentes foi projetada para aplicações específicas [OBR98], as aplicações multicast devem rodar no topo do protocolo

[1] TCP é um protocolo que provê um serviço confiável para transporte de mensagens.

UDP[2] (*User Datagram Protocol*) ou fazer interface direta com o IP[3] (*Internet Protocol*), fornecendo sua própria camada de transporte. O UDP fornece apenas o básico dos serviços da camada de transporte. Assim, se ocorrerem erros ou perdas, estes pacotes simplesmente não são entregues à aplicação [ABE00].

O multicast é voltado para aplicações do tipo um-para-muitos (um único host envia dados para dois ou mais hosts) e muitos-para-muitos (um número qualquer de hosts envia ou recebe dados de um grupo multicast). Entre as diversas aplicações que podem tirar proveito do uso de multicast podem-se citar: videoconferência, ensino à distância, jogos distribuídos, processamento concorrente, simulações distribuídas, etc.

Este capítulo descreve, inicialmente, o IP Multicast. Também são apresentados os protocolos de multicast confiável, citando alguns dos exemplos mais importantes. E, finalmente, são mostrados os pontos fundamentais dos protocolos de multicast semi-confiável e apresentadas algumas das abordagens mais significativas encontradas na literatura. Parte destas, inclusive, serviu de inspiração para a construção do protocolo proposto nesta dissertação.

4.1. Multicast IP

O Multicast IP [DEE89] foi desenvolvido a partir do protocolo IP, com o objetivo de tornar possível a difusão seletiva entre redes de computadores. Representa a transmissão de um datagrama IP para um grupo de *hosts* identificados por um único endereço IP. Não existe garantia de que os datagramas cheguem intactos a todos os membros do grupo ou na mesma ordem em que foram enviados.

O conceito de grupo é essencial para o multicast. No Multicast IP, cada grupo tem um identificador chamado ID do grupo multicast. Sempre que uma mensagem é enviada, este ID especifica o grupo de destino.

A associação a um grupo é dinâmica, os *hosts* podem participar ou abandoná-los a qualquer momento. Não há restrição quanto ao posicionamento geográfico ou ao número de membros em um grupo de *hosts*. Um *host* pode ser membro de um ou mais grupos ao mesmo tempo e não precisa ser membro de um grupo para enviar datagramas a ele.

---

[2] UDP é um protocolo que provê um serviço sem conexão não confiável para transporte de mensagens.

[3] IP é um acrônimo do inglês para *Internet Protocol* (Protocolo da Internet). É um protocolo usado pela fonte e *host* de destino para comunicação de dados através de troca de pacotes dentro da rede

Para gerenciar o multicast em redes de larga escala, como a Internet, foi criado o protocolo IGMP (*Internet Group Management Protocol*). Ele é responsável pelo gerenciamento das árvores de difusão. O IGMP é usado pelos *hosts*, para juntar-se e abandonar grupos de multicast, e pelos roteadores, para verificar periodicamente se os membros do grupo continuam ativos.

4.1.1. Transmissão

Datagramas IP Multicast são enviados usando a operação de u*nicast*: um módulo de protocolo de nível superior especifica unicamente um endereço de grupo como destino, ao invés de um endereço IP individual. Contudo, um conjunto de extensões é necessário:

- A interface de serviço deve permitir que protocolos de níveis superiores especifiquem o parâmetro TTL (*time-to-live*) de um datagrama multicast que esteja sendo enviado. Se o nível superior não especifica este parâmetro, é utilizado o valor padrão 1 para todos os datagramas. Assim, uma escolha explícita é necessária para propagar o multicast além de sua rede.
- Para *hosts* que possam estar associados a mais de uma rede, o serviço de interface deve permitir que os níveis superiores possam identificar que interface de rede será utilizada para realizar a transmissão multicast.
- Se o *host* que está enviando um datagrama é membro do grupo, uma cópia do datagrama deve ser enviada de volta para ele, a menos que tenha sido inibida pelo nível superior.

Para suportar o envio de pacotes Multicast IP, o módulo IP deve ser estendido para reconhecer endereços de grupo no roteamento dos datagramas enviados.

O endereço IP que indica o emissor de um datagrama deve ser um endereço individual pertencente à interface de saída. Um endereço multicast nunca deve ser colocado no campo de endereço fonte de um datagrama IP que esteja sendo enviado.

4.1.2. Recepção

Datagramas de Multicast IP que chegam são recebidos pelos protocolos de níveis superiores como se fossem datagramas *unicast*. A seleção de um protocolo de nível superior destino é baseada no campo ‘protocolo’ do cabeçalho IP, independente do endereço destino.

Contudo, antes que quaisquer datagramas sejam recebidos, um protocolo de nível superior deve requisitar ao módulo IP sua participação no grupo.

Para suportar a recepção de datagramas Multicast IP, o módulo IP deve ser estendido para conseguir manter uma lista de participantes de grupos associada a cada interface de rede. Quando um datagrama é recebido, o tratamento é o mesmo que aquele dispensado a datagramas destinados a um endereço individual de *host*.

Pacotes destinados a grupos aos quais o *host* não pertença são descartados. Também são descartados, em *hosts* com mais de uma interface de rede, datagramas que chegam via uma dada interface, mas destinado a um grupo para o qual o host pertence em outra.

4.1.3. MBone

Conceitualmente, para realizar multicast todos os *hosts* de uma rede precisariam suportar uma forma de roteamento de tráfego multicast. Na prática, todos os roteadores da Internet por onde circula o tráfego multicast devem implementar um ou mais protocolos de roteamento multicast. Além disso, roteadores conectados a sub-redes precisam implementar IGMP [DEE89] para exercer controle de grupo. Entretanto, a maioria dos roteadores da Internet, no início dos anos 90, não suportava qualquer tipo de protocolo de roteamento multicast. Isso levou a um problema: por um lado, o multicast IP não poderia ser empregado sem o suporte de fabricantes de roteadores e, por outro, os fabricantes de roteadores não dariam suporte a multicast IP antes que a tecnologia estivesse madura e robusta. O impasse foi quebrado criando-se roteadores com suporte a multicast nas sub-redes (IGMP) na periferia da Internet e as interconectando através de “túneis IP”.

O *Internet Multicast Backbone*, ou simplesmente *Mbone*, é uma rede *overlay* na Internet e por onde circula tráfego multicast. Em outras palavras, é um conjunto de sub-redes e roteadores interligados que suporta a entrega de tráfego multicast IP. O objetivo do *Mbone* foi construir um campo de testes para a tecnologia multicast IP que possibilitasse o emprego de aplicações de multicast sem esperar pela incorporação de suporte a multicast IP aos roteadores da Internet.

O *MBone* é composto de sub-redes que suportam multicast, denominados ilhas, conectadas umas as outras através de enlaces ponto-a-ponto reais ou virtuais. Cada uma das ilhas é composta por uma ou mais redes locais e por um *host* que realiza o roteamento multicast.

A comunicação entre estes roteadores pode ser realizada utilizando-se o conceito de túneis, enlaces virtuais ponto-a-ponto entre os roteadores, que possibilitam a transmissão de pacotes multicast entre os roteadores que não suportam esta forma de endereçamento, encapsulando-os dentro de pacotes *unicast*.

Os roteadores multicast têm a responsabilidade de replicar e distribuir os quadros de dados de multicast para os outros roteadores multicast e para a rede local, caso exista um *host* membro do grupo nela. A topologia de roteadores multicast do *MBone* possibilita uma distribuição de pacotes eficiente diminuindo a possibilidade de congestionamento nos nós ou enlaces de rede.

Quando um pacote multicast é enviado por um cliente, ele é capturado pelo roteador multicast da sub-rede. O roteador consulta sua tabela de roteamento e transmite o pacote para os roteadores multicast vizinhos correspondentes. O outro roteador receberá o pacote e consultará sua tabela de roteamento para decidir se o pacote deve ser enviado para algum outro roteador multicast, verificando também se há algum cliente em sua sub-rede que está inscrito neste endereço multicast e, caso haja, colocando-o na sub-rede para ser recebido pelo cliente.

O *MBone* possui uma topologia em malha e em árvore. As conexões entre os maiores provedores de serviços da Internet formam uma topologia em malha, criando os *backbones*[4] principais e os enlaces de reserva. Nas extremidades, a topologia é geralmente em árvore.

4.2. Multicast Confiável

São considerados protocolos de multicast confiável aqueles que fornecem um transporte totalmente confiável (entrega garantida) no topo de um serviço multicast não-confiável. Os principais mecanismos implementam controle de erro, controle de congestionamento, controle de sessão ou tratam de aspectos de segurança.

O mecanismo de confiabilidade requer alguma forma de conhecimento sobre a chegada dos pacotes ao receptor bem como um esquema de retransmissão ou recuperação dos pacotes perdidos [ABE00]. Podem ser utilizados:

---

[4] *Backbone* é a "espinha dorsal" de uma rede, geralmente uma infra-estrutura de alta velocidade que interliga várias redes.

- ARQ (*Automatic Repeat reQuest*): mediante o *feedback* (confirmação de entrega/pedido de retransmissão) o pacote perdido é retransmitido. A desvantagem é que o tráfego deste *feedback* pode causar um *overhead* (sobrecarga) desnecessário na rede.
- FEC (*Forward Error Correction*): envio de informações redundantes para correção dos possíveis erros. Este modelo não garante total confiabilidade e normalmente é usado em modelos híbridos combinado com soluções ARQ.
- Combinações de soluções ARQ e FEC.

Ainda não existe nenhuma especificação padrão para protocolos de multicast confiável. Contudo, existem vários protocolos deste tipo, alguns usados somente para pesquisa, outros para fins comerciais. O RMRG (*Reliable Multicast Research Group*) estuda o multicast confiável para auxiliar a IETF (*Internet Engineering Task Force*) a criar um conjunto de padrões.

Devido ao seu modelo de entrega, os protocolos de multicast confiável têm um impacto consideravelmente negativo na desempenho da rede. Eles podem gerar grandes quantidades de dados bem como tráfego de controle, o que pode causar congestionamento na rede e eventualmente levar a um colapso. Além disso, protocolos multicast podem tentar injetar pacotes na rede a uma taxa que ela não consegue suportar. Estes efeitos negativos tornam-se piores à medida que o grupo cresce. Dessa forma, são indispensáveis mecanismos efetivos para controle de congestionamento. Outro ponto importante são as técnicas para recuperação escalável de erros [OBR98].

A seguir, são relacionadas as propriedades do multicast confiável e apresentados alguns dos exemplos encontrados na literatura.

4.2.1. Propriedades do Multicast Confiável

Em [HAD94] é apresentada a definição de *broadcast* confiável, que pode também ser aplicada ao multicast confiável. Assim, para que a comunicação multicast possa ser considerada confiável, ela precisa apresentar as seguintes propriedades:

- Validade: se um processo correto difunde uma mensagem *m*, ele eventualmente a entrega.
- Acordo: se um processo correto entrega uma mensagem *m*, então todos os processos corretos eventualmente entregam *m*.

- Integridade: qualquer mensagem *m* é entregue no máximo uma vez aos processos corretos somente se foi enviada anteriormente através de *broadcast*.

A validade e o acordo garantem que uma mensagem enviada a partir de um processo correto, que não esteja provocando falhas, seja entregue a todos os processos corretos. E a integridade garante que não sejam entregues mensagens errôneas aos processos.

4.2.2. SRM

O SRM (*Scalable Reliable Multicast*) [FLO95] é um *framework* de multicast confiável para sessões consideradas "leves" e com *frames* em nível da aplicação. O SRM [FLO95] foi desenvolvido para atender à definição mínima de multicast confiável, por exemplo, entrega eventual de todos os dados a todos os membros do grupo, sem ênfase em nenhuma ordem de entrega em particular. Esta proposta também é fortemente baseada no modelo de entrega de grupos, onde os dados são enviados para um endereço multicast sem a necessidade de conhecimento prévio sobre os membros deste grupo. Cada receptor pode juntar-se ou sair do grupo sem afetar a transmissão de dados.

O SRM [FLO95] precisa do modelo de entrega por melhor esforço (*best-effort*) com possível duplicação e reordenação de pacotes. A confiabilidade é construída com uma base fim-a-fim. Os algoritmos do SRM [FLO95] ajustam dinamicamente os parâmetros de controle baseados no desempenho observado na sessão corrente. Isto permite o uso do SRM [FLO95] para vários tamanhos de grupo, topologias e larguras de banda com um desempenho alto e robusto.

O funcionamento é bastante simples: sempre que um membro gera novos dados, eles são difundidos no grupo. Cada membro é responsável por detectar perdas e pedir retransmissões através de NACKs (*Negative Acknowledgement*). A perda é normalmente detectada quando encontrada uma lacuna na seqüência de dados. Como é possível que o último objeto de uma seqüência se perca, cada membro envia mensagens periódicas informando o número de seqüência mais alto recebido até o momento.

4.2.3. RMTP e RMTP-II

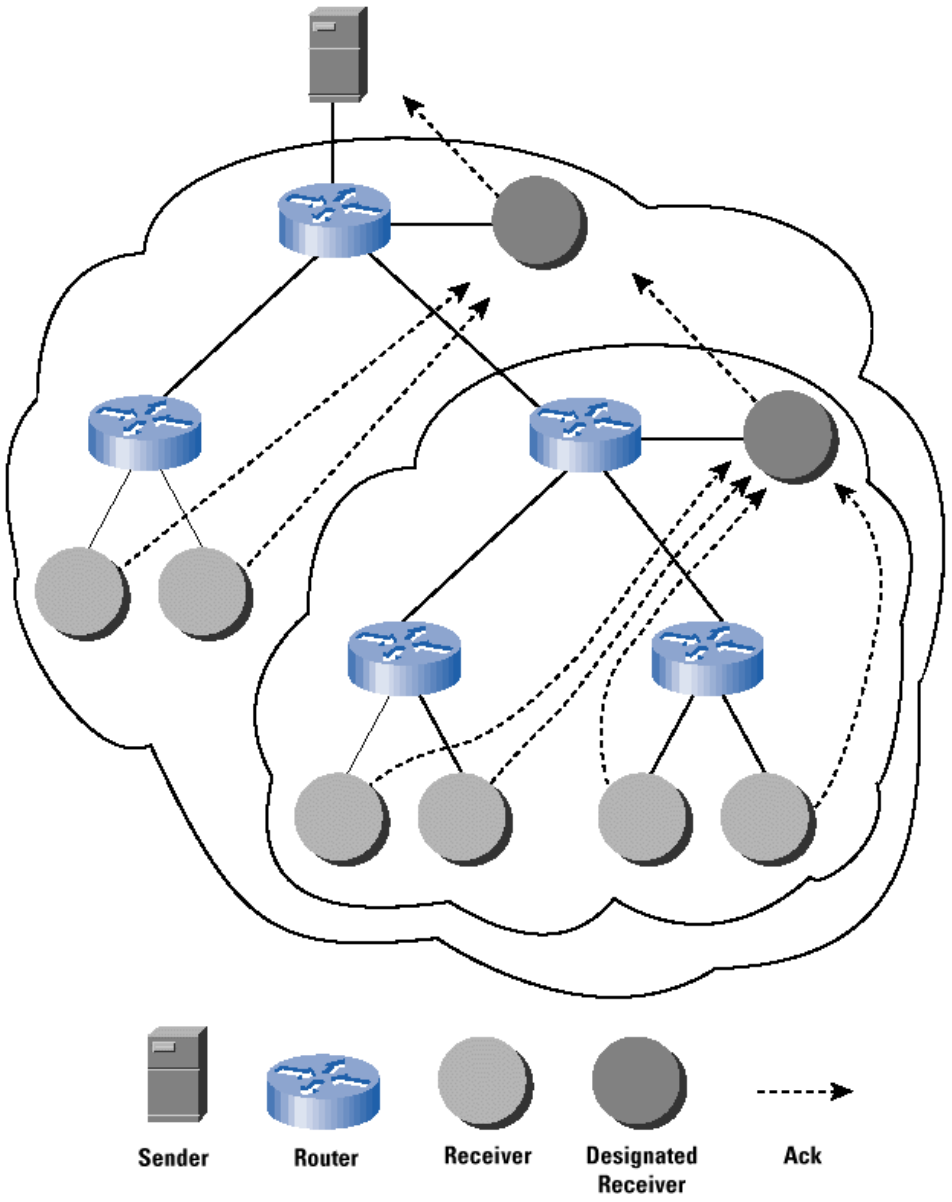

Figura 4-1. Receptores designados do RMTP [MIL98]

O RMTP (*Reliable Multicast Transport Protocol*) [LIN96] e o RMTP-II [WHE98] são protocolos baseados em árvore.

O RMTP é um protocolo de multicast confiável para a internet que fornece entrega dos dados seqüencial e sem perdas de um emissor para um grupo de receptores. É baseado em uma abordagem hierárquica com vários níveis, em que os receptores são agrupados em regiões hierarquizadas, com um receptor designado em cada grupo. Os receptores em cada

região enviam ACKs (*Acknowledgement*) periódicos para seus respectivos receptores designados, que os enviam para o receptor designado do nível mais alto ate que chegue ao emissor, evitando assim o problema de implosão de ACKs. Os receptores designados armazenam os dados recebidos para responder aos pedidos de retransmissão dos receptores em suas respectivas regiões, diminuindo assim a latência. Estes procedimentos estão ilustrados na Figura 4-1.

O RMTP-II é fortemente embasado no RMTP, mas melhorado para atender requisitos de sistemas em tempo real. Entre as melhorias, podem ser citados: suporte a NACKs e FEC (*Forward Error Correction*), um controle mais sofisticado da quantidade de ACKs gerados, suporte explícito a redes assimétricas, gerenciamento explícito da rede, membros do grupo contados, confiabilidade limitada por tempo e configuração automática dos receptores. Além disso, ainda apresenta a opção de configuração automática de receptores, o que permite ao protocolo ser "auto-configurável" se for usada uma única topologia de rede, sem o uso dos receptores designados.

4.2.4. Lorax

O Lorax [LEV96] é um protocolo que constrói e mantém uma única árvore para multicast confiável concorrente, eliminando a necessidade de uma árvore para cada emissor de um grupo multicast, e pode ser usado em combinação com vários protocolos de multicast confiável baseados em árvore.

Os receptores enviam mensagens de ACK para o nó acima dele na árvore periodicamente e, quando detecta uma perda, difunde uma mensagem de NACK no grupo. A estrutura interna de operação é semelhante à do RMTP. A entrega de mensagens de NACK e retransmissão de pacotes perdidos são feitas com um multicast de escopo limitado para evitar exposição dos pacotes de controle.

A desvantagem deste protocolo é que sua eficiência depende fortemente do algoritmo de roteamento multicast sobre o qual ele é utilizado.

4.2.5. TMTP

O TMTP (*Tree-based Multicast Transport Protocol*) [YAV96] é um protocolo que utiliza o mecanismo de melhor esforço (*best-effort*) para entrega e roteamento dos pacotes. Para o controle de fluxo e de erros, ele organiza dinamicamente os participantes em uma

árvore de controle hierárquica que aplica o envio de NACKs restritos e uma busca expandida para distribuir as funções de gerenciamento de estado e recuperação entre os membros do grupo multicast.

O TMTP, assim como o RMTP , organiza este grupo em uma hierarquia de sub-redes e um único receptor fica designado como representante de sua respectiva sub-rede, responsável pela recuperação de erros e tratamento de retransmissões locais. Além disso, o receptor designado pode também auxiliar na recuperação de erros em sub-redes vizinhas, desde que solicitado. A árvore de controle, formada pelos representantes de cada sub-rede, assegura a confiabilidade de entrega para cada membro do grupo multicast.

Cada receptor envia ACKs periodicamente apenas para o nó logo acima dele na hierarquia da árvore criada. Quando detecta uma perda de pacote, o receptor difunde em sua sub-rede um NACK, que, ao chegar ao receptor designado, pode ser difundido no nível hierárquico superior. Para limitar o escopo das retransmissões e envios de NACK, o TMTP utiliza um campo com um tempo limite para esse tipo de pacote. Como resultado disso, a recuperação de erros fica completamente localizada.

4.2.6. TRAM

O TRAM (*Tree-based reliable multicast*) [CHI98] é um protocolo de multicast confiável escalável desenvolvido para suportar transferências de grandes volumes de dados com um único emissor e vários receptores. Para implementar a recuperação de erros e suportar um grande número de receptores sem que haja grande impacto no emissor ele utiliza árvores dinâmicas, adotando muitas das técnicas do RMTP e do TMTP. Os receptores designados são escolhidos e a árvore é continuamente otimizada com base na população de receptores e na topologia da rede. O mecanismo de envio de ACKs é baseado em janela, como no RMTP , com algumas otimizações para reduzir riscos de implosões e sobrecarga.

O TRAM também inclui controle de fluxo, controle de congestionamento e outras técnicas adaptativas necessárias para operar de forma satisfatória com outros protocolos na grande variedade de conexões e características de clientes que formam as redes.

A confiabilidade na entrega dos dados é garantida para os receptores que se juntam à árvore e conseguem manter a velocidade mínima de retransmissão especificada pelo emissor. O TRAM constrói uma árvore de forma que os receptores designados para reparo fiquem próximos aos membros de sua respectiva sub-árvore. Isto permite que os reparos sejam feitos

com valores pequenos de TTL (*Time-to-live*), o que minimiza o consumo de largura de banda e evita processamento desnecessário nos receptores que não precisam da retransmissão.

4.2.7. LMS

O LMS (Light-weight Multicast Services) [PAP98] é uma extensão do Multicast IP [DEE89] com um conjunto de serviços simples que melhoram o encaminhamento nos roteadores para permitir soluções para multicast confiável altamente escaláveis e assistidas pela rede. Ele separa os componentes de controle de erro, mantendo os de transporte nos *hosts* e os de encaminhamento nos roteadores, onde podem ser mais eficientemente implementados.

No LMS, à medida que os receptores juntam-se à árvore de multicast, eles são organizados pelos roteadores em uma hierarquia onde cada roteador seleciona dinamicamente um *"host* pai". Se uma perda é detectada, todos os pedidos de retransmissão dos *"hosts* filhos" são enviados para o *host* pai, enquanto o pedido do pai é enviado para cima na hierarquia criada, assegurando que apenas um pedido seja enviado a partir de cada sub-árvore. Antes de enviar os pedidos ao *host* pai, um roteador insere um endereço das interfaces de saída e entrada dos pedidos que passam por ele. Este roteador é chamado de *turning point* do pedido, que identifica a raiz da sub-árvore que originou o pedido de retransmissão. Este processo assegura que um pedido de retransmissão encontre um receptor que tenha os dados solicitados, ou atinja a fonte. Em um caso ou em outro, a retransmissão é feita via *unicast* para o roteador *turning point*, que, por sua vez, difunde os dados para a sub-árvore afetada pela perda.

A implosão e a exposição de dados são tratadas pelo uso da hierarquia, que localiza a recuperação entre *hosts* pais e filhos. A hierarquia adapta-se rapidamente a mudanças de membros do grupo e de roteamento desde que os roteadores assegurem isso. A latência de recuperação de erros é minimizada porque, com o LMS, os *hosts* mais próximos da perda são envolvidos no processo e a recuperação das mensagens é enviada imediatamente. Finalmente, a hierarquia mantida pelos roteadores retira dos receptores toda informação de estado relacionada à topologia, como temporizadores, relacionamentos *host* pai/filho, etc.

4.2.8. Search Party

O *Search party* [COS99] segue a estrutura do LMS: os pedidos de retransmissão são enviados "árvore acima" até a raiz e retornam com a informação inserida sobre *turning point*.

O LMS torna o roteamento dos pedidos de retransmissão aleatório, trocando performance por robustez.

A responsabilidade de envio de retransmissões para uma conexão com perda em particular é dividida entre vários membros, em vez de um único receptor ou *host* designado. A desvantagem é um atraso na retransmissão aumentado e/ou uma maior sobrecarga (*overload*) de recuperação. Por outro lado, a responsabilidade distribuída pelo grupo faz com que um membro que funcione mal tenha um impacto limitado sobre os vizinhos, e um efeito desprezível no grupo como um todo.

Cada membro do grupo consegue detectar as perdas porque a fonte envia atualizações constantes que contém informações de estado suficientes, como o maior número de seqüência enviado até o momento. Quando um membro detecta uma perda, ele continuamente envia pedidos de reenvio até que uma retransmissão chegue. Cada membro conduz uma busca aleatória pelos dados perdidos. Como nenhum membro sabe onde ocorreu a perda, nenhum deles sabe como conduzir a busca. No entanto, como todos os membros participam da busca, independentemente de seu envolvimento nas perdas, eles promovem uma busca (*search party*) de tamanho e escopo corretos. Qualquer membro pode fazer retransmissão. Se for um receptor, ele envia o pacote solicitado para a sub-árvore de onde veio o pedido. Se for a fonte, ela faz um multicast.

Para evitar respostas duplicadas dentro da sub-árvore, cada membro guarda a informação sobre a última vez em que recebeu uma retransmissão dos dados e ignora pedidos que antecedam esta retransmissão.

4.2.9. RMCM

O RMCM (*Reliable Multicast for Core-based Multicast Trees*) [GAO00] é um protocolo para árvores de multicast baseadas no núcleo. São definidas novas opções IP para levar informações sobre o caminho no pacotes de dados e NACKs e é projetado um esquema para facilitar o direcionamento apropriado de NACKs e retransmissões pelos roteadores da árvore. Para realizar recuperação local é proposto um mecanismo que seleciona os *hosts* designados (replicadores) que têm mais chances de ter o pacote solicitado e para o qual os NACKs serão encaminhados. Também é proposta uma abordagem de ACKs atrasados de forma que tanto as fontes como os replicadores podem eliminar pacotes de dados que já foram recebidos por todos os membros do grupo.

Quando um receptor detecta uma perda, ele envia um NACK para seu roteador local contendo o número de seqüência do pacote perdido e a informação do caminho (*path_info*) para o par fonte-grupo multicast. Quando um roteador recebe um NACK, ele pode identificar a interface onde ocorreu a perda recuperando a *path_info*. Então, o roteador direciona o NACK usando o mecanismo semelhante ao *turning point* [PAP98]. Um replicador com os dados solicitados envia-os através de *unicast* ao *turning point* que, por sua vez, envia para a sub-árvore que sofreu a perda. O RMCM possui ainda um mecanismo para informar aos receptores e replicadores quando os pacotes de dados podem ser eliminados.

Os receptores enviam ACKs atrasados periodicamente, que incluem a informação *path_info*, para o roteador da árvore. Essas informações são levadas até a fonte para que ela determine qual o número de seqüência abaixo do qual as mensagens vão poder ser eliminadas.

4.2.10. ARM (I)

ARM (*Active Reliable Multicast*) [WEI98] é um protocolo baseado em NACKs em que os receptores são responsáveis por detectar e solicitar a retransmissão dos pacotes perdidos. Cada pacote de dados recebe um número de seqüência e os receptores detectam perdas pela ocorrência de lacunas na seqüência dos dados recebidos.

O receptor envia um NACK para o emissor assim que detecta uma perda. Os vários NACKs dos diferentes receptores são 'fundidos' em roteadores ativos ao longo da árvore de multicast de forma que só um NACK por mensagem perdida chegue à fonte. A fonte só responde ao primeiro NACK para cada mensagem, difundindo um pacote para reparo no grupo, ignorando os NACKs seguintes para mensagens que já foram retransmitidas.

Uma vez que os NACKs e mensagens de reparo também podem ser perdidos, um receptor pode reenviar um NACK se não receber a retransmissão dentro de um limite de tempo determinado. Para indicar novos NACKs, cada um contém um contador que indica quantas vezes o receptor pediu a retransmissão do pacote. O emissor guarda o contador com o valor mais alto de cada NACK, assume que a retransmissão anterior foi perdida e retransmite a mensagem mais uma vez.

Os roteadores ficam, neste protocolo, responsáveis por:

- Armazenar os dados para retransmissões locais. Se ele receber um NACK para uma mensagem que não possua, encaminha o pedido de retransmissão para as camadas mais altas da árvore de multicast.

- Fusão e cancelamento para evitar que cheguem NACKs duplicados na fonte
- Multicast parcial para retransmissões de forma que apenas os receptores que pediram retransmissão a recebam.

4.2.11. ARM(II)

O protocolo ARM (*Adaptive Reliable Multicast*) [YOO00] integra técnicas de ARQ e FEC. Tem como objetivos reduzir a sobrecarga de mensagens devido a pedidos de NACK, reduzir a quantidade de transmissões de dados e reduzir o tempo para que todos os receptores recebam os dados intactos (sem perdas). Durante a transmissão de dados, o emissor informa periodicamente aos receptores o número de pacotes que ainda serão transmitidos. Com base nesta informação, cada receptor estima se esta quantidade é suficiente para recuperar as perdas. Apenas se não for o receptor pede, com o envio de NACKs, que o emissor envie pacotes adicionais. Desta forma, o emissor pode ajustar dinamicamente a quantidade de informação redundante que vai enviar em seus pacotes (FEC).

A redução nos tempos de transmissão ocorre devido à eliminação da maioria dos pedidos de retransmissão como um resultado do ajuste dinâmico adequado da redundância enviada aos receptores. A redução na sobrecarga de mensagens é devida ao fato de o emissor codificar novos pacotes de reparo à medida que são solicitados, fazendo com que os receptores não recebam pacotes duplicados. Ale disso, os receptores não enviam NACKs se a redundância estimada for suficiente para uma recuperação completa dos pacotes perdidos, reduzindo o risco de implosão de NACKs. Finalmente, uma vez que o receptor tenha o número de pacotes suficientes para recuperar completamente os dados originais, ele pode abandonar o grupo de multicast, assim a árvore de roteamento multicast encolhe com o passar do tempo.

4.2.12. AER

O AER (*Active Error Recovery*) [KAS00] assume que os receptores não têm capacidade de transmissão multicast. Um emissor difunde os pacotes para um endereço multicast onde estão todos os receptores participantes do grupo e servidores de reparo. Quando é detectada uma perda, após um tempo de espera aleatório, é feito *unicast* de um NACK para o servidor de reparo logo acima na hierarquia da árvore multicast e é iniciado o temporizador da retransmissão. Este temporizador serve para evitar a implosão de NACKs. Se

o receptor recebe um NACK para o mesmo pacote, ele cancela o seu próprio. Se o servidor de reparo tem o pacote para atender ao receptor do nível mais abaixo, ele envia para os *hosts* abaixo dele na hierarquia. Caso contrário, ele envia o NACK para os receptores abaixo dele e começa o processo para obter o pacote a partir da fonte.

Se o temporizador da retransmissão expira sem que a retransmissão chegue o NACK é retransmitido.

Ao receber um NACK de um servidor de reparo, a fonte retransmite o pacote solicitado para todos os receptores e servidores de reparo. Estas retransmissões são interceptadas pelos servidores de reparo e só são encaminhadas para onde foram solicitadas.

4.2.13. PGM

A proposta do PGM (*Pragmatic General Multicast*) [SPE04] é ter a ajuda da infra-estrutura da rede, isto é, dos roteadores para ter escalabilidade. O PGM é um protocolo simples que faz interface direta com a camada IP (*Internet Protocol*).

Cada pacote do PGM contém um identificador de sessão de transporte, TSI (*Transport Session Identifier*) para identificar a sessão e a fonte de dados.

Como o PGM foi projetado também para atender a aplicações em tempo real, a periodicidade é um quesito importante. Esta questão é tratada por uma janela de dados e modo que se não são recebidos NACKs até que o tempo da janela se esgote, os dados simplesmente não ficam mais disponíveis para reparo.

O PGM é totalmente baseado em NACKs, que são enviados a partir do receptor para o roteador mais próximo e dali por diante seguem em *unicast* de roteador para roteador até que os dados de reparo sejam transmitidos. O cancelamento de NACKs é feito por qualquer roteador de uma sub-rede de receptores e depois todos os demais roteadores eliminam os NACKs duplicados em todas as conexões até a fonte de dados. Os roteadores também guardam o estado de onde os NACKs vêm na árvore de distribuição de forma a só encaminhar os pacotes de reparo para onde foram solicitados.

O funcionamento é ilustrado na Figura 4-2.

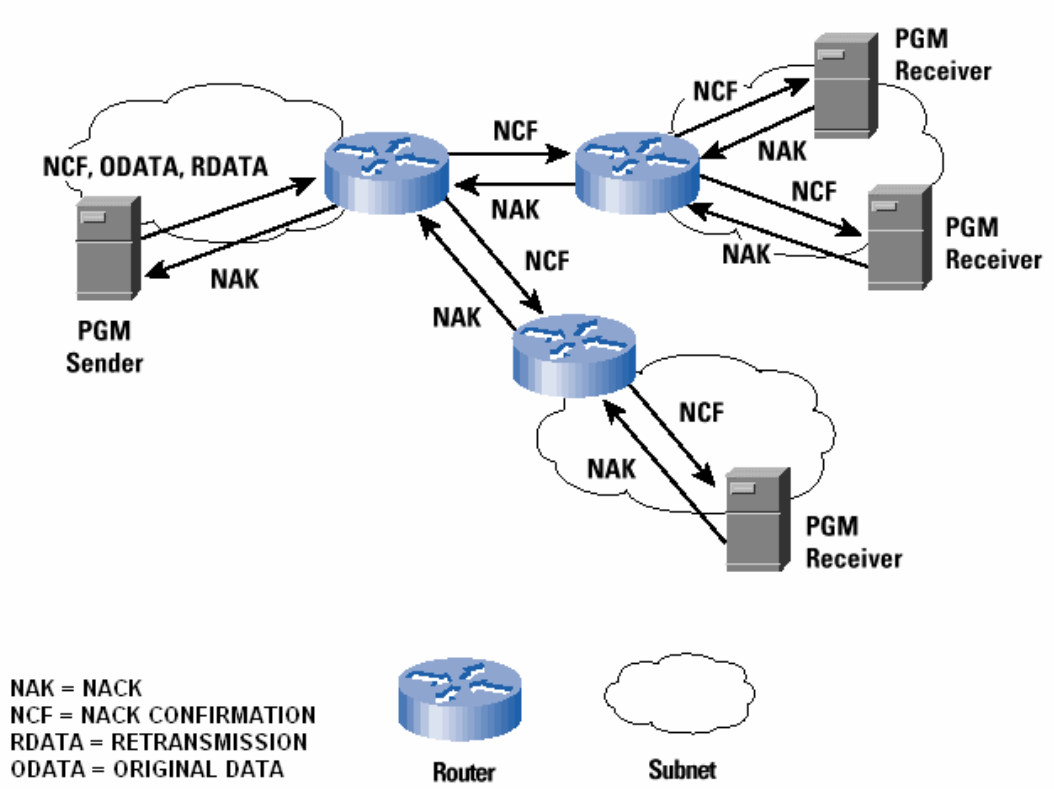

Figura 4-2. Funcionamento do PGM [MIL98]

4.2.14. ReMIOP

O ReMIOP [BES03] é um protocolo de multicast confiável desenvolvido especificamente para a plataforma CORBA. Ele é um protocolo escalável baseado no receptor, seguindo os moldes de outros protocolos de difusão confiável.

No CORBA, existe um protocolo de difusão não confiável baseado no IP multicast. Este protocolo é responsável pelo mapeamento de mensagens sobre a pilha UDP/ multicast IP e apresenta alto desempenho, mas sem garantias de entrega. O ReMIOP constitui um conjunto de extensões a este protocolo para a adição da confiabilidade. Além do mecanismo para controle de fluxo, ele utiliza o envio de NACKs para a recuperação de erros.

4.3. Multicast Semi-confiável

As aplicações multimídia têm requisitos diferentes das aplicações baseadas em interações convencionais cliente-servidor. Um *stream* de áudio, por exemplo, requer que os dados sejam recebidos no momento correto, mas é mais complacente com dados perdidos. Se um pacote de dados chega muito tarde, ele ‘perde’ o momento em que deveria ser

reproduzido. No exemplo do *stream* de áudio, isto causa lacunas no som ouvido pelo usuário. Em muitos casos, o pacote atrasado em uma aplicação multimídia não contribui em nada na reprodução e é equivalente a uma perda. No entanto, pequenas perdas no *stream* de reprodução podem ser substituídas ou ocultadas sem que o usuário perceba.

No projeto de uma aplicação multimídia, um protocolo multicast deve ser escolhido de forma a fornecer uma solução para as restrições temporais, bem como para as perdas e erros que venham a ocorrer durante a transmissão. Para isso é conveniente utilizar os protocolos de multicast semi-confiável [PIE00, MAN00, PER03] que buscam a garantia relaxada de entrega dos pacotes desde que respeitadas as limitações impostas pela aplicação.

Multicast semi-confiável é uma abstração de comunicação em que nem todos os pacotes são necessariamente retransmitidos quando solicitados, sendo dada uma prioridade para aqueles que tenham importância fundamental para manter a qualidade da informação requerida pela aplicação. Isto é, dado um conjunto de pacotes a ser difundido em um grupo de receptores, a entrega confiável é garantida apenas a uma parte destes: os de maior prioridade. A difusão confiável de um pacote implica em sua entrega a todos os receptores corretos (propriedade de acordo [HAD94]). Esta garantia pode ser conseguida com o uso de algum mecanismo de correção de erro, como a retransmissão dos pacotes. Para os demais pacotes do conjunto, os de menor prioridade, os erros de transmissão só são corrigidos se as condições da rede (tráfego, congestionamento, latência, etc.) permitirem. Caso contrário, a entrega é baseada no melhor esforço (*best-effort*), em que a prioridade de acordo pode ser violada.

Os pacotes (ou mensagens) a serem difundidos em um grupo de receptores podem ser classificados dentro de uma hierarquia, com base em alguma semântica da aplicação [PER03] ou em alguma regra do próprio protocolo multicast semi-confiável, estabelecendo níveis de importância ou prioridade na correção de erros (retransmissão seletiva [PIE00, MAN00]). Portanto, a correção de erros é efetuada baseada na prioridade e em parâmetros que reflitam o estado da rede. A adoção desta estratégia permite que sejam tomadas decisões pró-ativas [LEI04] para a recuperação de pacotes em caso de perda.

A seguir são descritas algumas das propostas encontradas na literatura para melhor ilustrar o conceito de multicast semi-confiável.

4.3.1. Janela de Confiabilidade

Em [CHI02], é proposto um mecanismo que permite a troca de confiabilidade por performance e escalabilidade em um protocolo de transporte multicast. Isto permite que a aplicação diga à camada de transporte para não recuperar pacotes que estejam "obsoletos" o suficiente. Em momentos de congestionamento e severas perdas de pacotes, este mecanismo ajuda a camada de transporte a trocar a confiabilidade por desempenho/escalabilidade.

Outro ponto interessante nesta proposta é a diferenciação dos receptores com menos capacidade. A camada de transporte entrega diferentes "camadas" de informação (com mais ou menos detalhes) para os diferentes receptores de acordo com sua capacidade. Não havendo reparo de certos pacotes perdidos para os receptores mais limitados, eles ficam com menor confiabilidade. A divisão em "camadas" de detalhes é feita de forma adaptativa e é diferente para cada receptor.

A cada operação de difusão de mensagem no grupo, há um retorno de um número para o emissor que fica guardado como uma referência. A mensagem difundida também inclui um número de referência que não pode ser maior do que o número mais alto retornado até o momento. Os números de referência não podem diminuir depois de difusões sucessivas. Este número de referência é usado para dizer à camada de transporte para não recuperar pacotes com um número de referência menor do que ele. Este parâmetro é chamado *forgetbefore* (esqueça o antes). Enviado na mensagem junto com o próprio número de seqüência da mensagem, o *forgetbefore* fornece a chamada 'janela de confiabilidade'.

No lado do receptor, no momento de envio de ACKs, qualquer pacote com um número de seqüência anterior ao atual *forgetbefore* é tratado como recebido.

O funcionamento deste protocolo baseia-se em uma janela de confiabilidade utilizada no cabeçalho dos pacotes de dados para indicar quais deles devem ser recuperados. A idéia é fazer com que as aplicações estabeleçam a janela de confiabilidade de modo significativo. Este mecanismo é útil para distribuição de conteúdo em larga escala onde o conteúdo torna-se obsoleto rapidamente, especialmente quando o novo conteúdo sobrescreve o anterior. Para garantir a ordem das mensagens, os pacotes que chegam ao receptor são colocados em uma fila antes de serem entregues à aplicação. Quando os pacotes que faltam na seqüência já estão obsoletos, ao chegarem são tratados como se já tivessem sido entregues à aplicação. Caso contrário, eles são inseridos na fila na posição apropriada.

Por não tentar recuperar alguns pacotes perdidos, este mecanismo pode afetar os algoritmos de fluxo e controle de congestionamento, especialmente se o controle de congestionamento está baseado nas perdas medidas para determinar a taxa de dados.

4.3.2. PTP

A plataforma *Prism* [TOU95] facilita o desenvolvimento de aplicações multimídia distribuídas em ambientes heterogêneos. Ela permite executar aplicações multimídia em estações de trabalho padrão e provê uma qualidade visual melhorada para estações de baixa velocidade. Oferece serviços de rede ajustados para a natureza especial de objetos multimídia (multicast, datagrama seguro, etc.). O plano de distribuição cuida das comunicações. Ele usa diferentes pilhas de protocolos e representações de dados e não tem conhecimento dos tipos de dados multimídia. O plano de distribuição oferece uma API padrão para o plano multimídia.

Um domínio do Prism é um conjunto de estações recebendo o mesmo grupo de informações, mas não é caracterizado por um único endereço de rede como nos domínios de multicast convencionais.

O protocolo de transporte do Prism, o PTP (*Prism Transport Protocol*) é projetado para preencher uma lacuna nas pilhas convencionais de protocolos para a Internet (TCP/UDP/IP), oferecendo além do serviço de multicast convencional, um mecanismo de transmissão confiável. O PTP é usado em aplicações multicast que precisam estar livres de perdas em domínios de endereçamento Prism.

Ele possui seu próprio esquema de endereçamento e um cabeçalho específico semelhante à abordagem RTP (*Real-time Transport Protocol*) [SCH96]. Não foi usado o esquema de endereçamento da Internet porque não há roteamento na camada de rede no PTP. O roteamento é implementado pela aplicação, pois ela tem um conhecimento melhor sobre os conteúdos e as prioridades.

O cabeçalho PTP contém o endereço, o número da unidade de informação (UI), o número de *offset*, o número de seqüência e a prioridade. O PTP é estruturado em duas camadas. A primeira, de cima para baixo, gerencia os chamados *buffers* circulares. A segunda trata as filas de pacotes de dados. Para cada endereço PTP, há uma instância de um *buffer* circular, um conceito baseado em uma estrutura entrelaçada de ponteiros de memória, que evita que haja cópias físicas da fila de entrada de dados.

No modo de operação para dados seguros, o protocolo atua como um protocolo de multicast semi-confiável. Quando detecta uma perda (recebendo um número de seqüência inesperado), o receptor retorna um NACK. Para evitar repetição de NACKs, a fonte dispara uma janela de inibição assim que recebe o primeiro, informando aos membros do grupo que não precisam enviar um NACK para aquele pacote.

4.3.3. SRP

O SRP (*Selective Retransmission Protocol*) [PIE00] utiliza um algoritmo de decisão específico por aplicação para determinar se um pedido de retransmissão do pacote perdido deve ser enviado ou não, ajustando o nível de perda e latência de acordo com a aplicação. Ele retransmite apenas uma porcentagem dos dados perdidos, sendo que a quantidade depende dos fatores de QoS (*Quality of Service*), incluindo perdas, latência, RTT (*Round Trip Time*), congestionamento da rede e qualidade desejada.

O SRP foi desenvolvido como um protocolo no nível de aplicação que usa UDP para transmissão de mensagens. Ele consiste em um servidor que envia um *stream* de multimídia em uma série de datagramas a uma taxa constante e em clientes que a recebem. Quando o cliente detecta que um pacote não chegou no momento esperado, ele decide se vai ou não pedir retransmissão da mensagem. Esta decisão é tomada por um algoritmo que leva em conta quantas perdas e quanta latência são toleradas pelo usuário e as perdas e latência medidas no momento. Se a decisão é retransmitir, um pedido é enviado para o servidor e o cliente aguarda a resposta. Se a resposta não chega quando esperada, a decisão sobre retransmissão é avaliada novamente. Se a decisão é não retransmitir, o cliente desiste daquela mensagem e retorna uma falha à aplicação receptora. Quaisquer mensagens recebidas durante a espera por retransmissão são colocadas em um *buffer* enquanto necessárias para a aplicação.

Os tempos de recepção esperados são gerados automaticamente à medida que as mensagens são recebidas. O tempo entre as chegadas é medido e é feita uma média. Espera-se que as mensagens cheguem em um RTT, medido por testes enviados ao servidor e que retornam periodicamente.

Durante o projeto do SRP, foram feitas as seguintes considerações:

- Sem *clocks* sincronizados entre o servidor e o cliente, a latência exata não pode ser conhecida. Por isso, assume-se que o tempo para envio de uma mensagem do

servidor ao cliente é o mesmo que do cliente ao servidor. Este tempo é estimado em aproximadamente metade do RTT.

- A aplicação cliente está lendo continuamente e tem um pequeno RTT.
- A aplicação do servidor envia mensagens a uma taxa constante. O servidor não tem controle de fluxo ou conhecimento do cliente. Isto permite que o cliente estime um tempo de chegada esperada.

O servidor SRP envia um *stream* multimídia para o cliente. A intervalos ímpares, a aplicação emissora solicita envio dos dados. O servidor SRP constrói uma mensagem de dados SRP controlando o número seqüencial dos dados e envia para o cliente via UDP. Estes dados também são armazenados em um *buffer* onde podem ser recuperados mais tarde caso seja necessário retransmitir. Se um pedido de retransmissão é recebido, os dados são recuperados a partir do *buffer* e reenviados. Se os dados não estão presentes, o pedido de retransmissão é ignorado. O cliente SRP é responsável por receber um *stream* multimídia do servidor e pedir retransmissão se necessário.

Se uma mensagem é recebida, verifica-se se o número de seqüência é o esperado. Se for maior (chegou muito cedo), a mensagem é armazenada no *buffer*. Se a mensagem que chega tem o número de seqüência esperado, é enviada para a aplicação emissora. Se ela em um número seqüencial menor (atrasada), é ignorada. Se os dados corretos não são recebidos antes de esgotado o tempo de retransmissão, então um algoritmo de decisão é consultado a respeito da retransmissão.

4.3.4. QMTP

Esta abordagem envolve o uso de uma combinação de controle de fluxo com base na taxa de envio, transmissões redundantes, e mensagens ocasionais de feedback para a entrega de QoS dentro de uma faixa aceitável para determinada aplicação [YAV93]. Foi desenvolvido um protocolo de transporte multicast chamado QMTP (*Quasi-reliable Multicast Transport Protocol*) [MAN93], que considera apenas o mecanismo de entrega multicast por melhor esforço (*best-effort*). Ele requer que a fonte especifique seus próprios requisitos de desempenho através dos parâmetros de serviço. Como as fontes de áudio e vídeo digital também podem tolerar alguns erros, permite-se que a aplicação especifique a tolerância.

A idéia central nesta abordagem é trocar largura de banda extra por uma redução na latência. A seqüência de ações nos pontos de missão e transmissão de um *stream* QMTP é a seguinte:

- A cada intervalo (tempo no qual monitora-se a qualidade do serviço de um *stream* de multicast), a fonte transmite quadros a uma taxa fixa.
- À medida que recebe os quadros, o receptor entrega-os à camada superior na seqüência em que devem ser apresentados. Os quadros que chegam muito tarde ou com erro são simplesmente descartados. O receptor mede a taxa de erro em cada intervalo e pode ou não enviar um NACK.
- Durante o intervalo, se algum NACK é recebido, a fonte coloca no *buffer* até o final deste intervalo, quando examina todos os NACKs recebidos e, se necessário, reduz a taxa de quadros. Também pode ser utilizada a estratégia da redundância para aumentar a probabilidade de recuperação de erros no receptor.
- Na estratégia de redundância, a fonte ou usa replicação (alguns pacotes escolhidos aleatoriamente são transmitidos duas vezes) ou transmite pacotes adicionais gerados pelo método FEC. Se não se atinge o QoS desejado após sucessivos intervalos, o QMTP pode abortar o serviço, a menos que a aplicação escolha reduzir os limites de QoS.
- Depois que a taxa de quadros é reduzida, se não forem recebidos novos NACKs após sucessivos intervalos, é possível aumentar a taxa de quadros gradualmente até a taxa desejada inicialmente.

Dada a natureza sensível a atrasos da comunicação multimídia, discute-se que o protocolo de transporte deve usar uma técnica FEC para evitar atrasos inerentes aos controles de erro e fluxo tradicionais. Resultados experimentais mostraram que a política probabilística funciona melhor com transmissões redundantes [YAV93].

4.3.5. TUNA

No artigo [WON98], os autores descrevem o TUNA (*TUNAble Quasi-Reliable Multicast Protocol*), uma adaptação da confiabilidade seletiva para disseminação de informações periódicas. A principal idéia do TUNA é suprimir NACKs de um receptor quando a “idade” esperada da informação perdida está abaixo de um limite especificado pelo

usuário. Em outras palavras, se a idade for maior que o limite, o receptor imediatamente envia um NACK. Senão, ele simplesmente espera pela próxima atualização da informação perdida. Assim, os receptores não têm garantia de receber todos os dados do emissor (por isso semi-confiável), mas a idade da informação é mantida dentro de um limite (é ajustável).

Para estimar a idade esperada, os autores propõem o STAP (*Statistics Announcement Protocol*), que usa o mecanismo de anúncio/escuta para enviar estatísticas das periodicidades das informações em intervalos regularmente espaçados. A vantagem deste mecanismo é não haver recuperação de erros explícita porque uma mensagem de STAP perdida é atualizada no próximo intervalo. Ele também reduz a quantidade de informações sobre o estado nos receptores porque eles não têm que manter as estatísticas. Uma vez que estas periodicidades podem mudar dinamicamente de acordo com a semântica das informações, os autores as representam em termos da média dos tempos entre os envios e seus desvios padrão.

O algoritmo de recuperação no TUNA é bastante simples e pode ser implementado com poucas linhas de código no receptor. Ele primeiro calcula a periodicidade das informações perdidas usando estatísticas do STAP. Ele adiciona a periodicidade ao tempo que o receptor recebeu a informação pela última vez para estimar quando deverá receber a informação perdida. Este valor é subtraído do momento atual, que é a "idade" atual dos dados. O algoritmo também prevê o tempo médio de espera até a próxima atualização. Juntos, a "idade" atual e o tempo de espera são a "idade esperada" que o receptor vai precisar para decidir se opta por não enviar o NACK. Se este valor é maior que o limite especificado pelo usuário, o algoritmo repara a perda explicitamente enviando um NACK. Senão, simplesmente espera pela próxima atualização da informação. Quando as estatísticas para as informações perdidas não estão disponíveis, ele utiliza a abordagem conservativa e sempre envia um NACK. Desta maneira, a "idade" da informação é mantida abaixo do limite.

As técnicas de recuperação de erro do TUNA são aplicáveis a informações que mudam com freqüência.

Neste mesmo artigo, os autores descrevem outra forma de semi-confiabilidade, baseada nos interesse dos receptores. Os interesses dos receptores normalmente seguem a distribuição de Zipf [GUN96], que diz que uma pequena porção de informação sendo disseminada é considerada interessante pela maioria dos usuários. Assim, são adicionadas seletivamente informações duplicadas deste tipo de informação no *stream* de dados. Não é necessário aumentar as periodicidades de todas as informações devido à distribuição não-

uniforme dos interesses dos receptores. Os autores especulam que um pequeno aumento na largura de banda para encaminhamento pode diminuir bastante o número de NACK gerados.

Enquanto o emissor tem conhecimento inerente sobre as periodicidades, ele tem que coletar os interesses dos receptores. Novamente é usado o anúncio/escuta para os receptores para enviar seus interesses periodicamente de forma escalável com intervalos proporcionais ao número de receptores. Como é impraticável manter uma visão global consistente de todos os interesses dos receptores, uma aproximação é calculada com uma amostra dos interesses. Esta abordagem é modelada segundo o protocolo SCUBA [AMI97], que prova que o tempo de convergência e assim a precisão do mecanismo de amostragem são independentes do tamanho da sessão de multicast.

4.3.6. XTP

O XTP (*Xpress Transport Protocol*) [STR92] busca fornecer um protocolo de transporte útil a sistemas de tempo real que permita a qualquer número de receptores participarem de uma sessão confiável um para muitos, conservando a largura de banda.

Para atingir seu objetivo, o XTP precisa realizar o processamento contínuo de pacotes. Assim, informações como o tipo de pacote, identificador de contexto, modos, *flags* e opções de processamento são enviadas no cabeçalho do pacote para acesso imediato no momento de sua chegada.

Uma premissa fundamental do XTP é separar políticas de mecanismos, especialmente nos modelos de comunicação e dispositivos de recuperação de erros. Desta forma, o usuário pode escolher o modelo mais apropriado para a sua aplicação, ao contrário do TCP, por exemplo. O XTP [STR92] também não impõe aos usuários um esquema específico para recuperação de erros. Em vez disso, é proposta uma série de meios de recuperação de erros com graus que variam de nenhuma a completa recuperação.

Para construção dos modelos de comunicação, é fornecido o mecanismo da *associação*, que é uma manutenção das informações de estado para comunicação entre os *hosts.* O XTP trata a comunicação multicast como uma extensão natural da associação. Há um bit de sinalização que indica aos receptores que eles são parte de uma comunicação multicast e assim devem empregar esquemas de reconhecimento apropriados para esta situação.

Com a separação das noções de modelo de comunicação e confiabilidade, no XTP, a aplicação pode estabelecer o grau apropriado de confiabilidade para o modelo de

comunicação escolhido. Este protocolo permite a retransmissão seletiva dos dados perdidos bem como outras políticas tradicionais e oferece graus de confiabilidade. O confiável é útil para serviços onde uma quantidade de erros parametrizada pode ser tolerada em alguns períodos de tempo. Por exemplo, dados de voz empacotados podem sofrer perdas intermitentes, ou uma rajada de perdas degradaria o serviço suficientemente para requerer alguma forma de recuperação. Se ocorrer uma perda aceitável de dados, o receptor deve estar apto a tentar a recuperação sem bloquear a entrega de mensagens enquanto aguarda a chegada dos dados retransmitidos.

Como o emissor é responsável pelo fornecimento dos dados perdidos, o receptor deve informar esta necessidade. O receptor acompanha a continuidade dos dados e à medida que ocorram as falhas, ele constrói uma lista dos dados recebidos corretamente. Quando recebe a solicitação de envio das informações de estado, ele coloca esta lista em um pacote de controle. Com estas informações, o emissor pode retransmitir os dados que estão faltando. Considerando que a retransmissão seletiva nem sempre é benéfica, o emissor pode ignorar estas informações e retransmitir os dados a partir da última seqüência cuja recepção foi confirmada.

4.3.7. WAIT

Na tese [MAN00], é proposta uma nova técnica chamada WAIT [MAN00], que evita os problemas de implosão e duplicidade resultantes da comunicação entre os receptores. Afirma-se que esta técnica recupera efetivamente a perda de pacotes e mantém a qualidade desejada (em termos de perda e de atraso) dos *streams* de áudio e vídeo em uma sessão de multicast.

Neste protocolo é preciso que na formação de um grupo os receptores tenham alguma informação sobre a topologia em que se encontram. Para fornecer esta informação, é proposto um mecanismo de roteamento e entrega dos dados ao receptor, onde cada roteador multicast adiciona seu endereço no pacote enviado pelo emissor antes de encaminhá-lo a todos os ramos da árvore.

A abordagem apresentada é motivada pelo argumento de que os roteadores devem ser mantidos o mais simples possível e não devem tomar parte nas questões de perda e recuperação. Assim, os *hosts* finais executam as tarefas de detectar a ocorrência da perda e

recuperá-la da forma mais efetiva possível em termos de latência de recuperação e sem os problemas de implosão e exposição.

A técnica apresentada trabalha diferentemente com perdas locais e perdas globais. No primeiro caso, se ocorre uma perda no ramo principal, o receptor mais próximo à raiz da sub-árvore é o primeiro a perceber. Então, ele deve enviar um NACK ao receptor e/ou emissor mais próximo e acima na hierarquia. O emissor e/ou receptor deve enviar o pacote de reparo para a raiz da sub-árvore onde aconteceu a perda para a recuperação. No segundo caso, se a perda ocorre no ramo de um grupo, deve ser feito *unicast* do reparo. Neste caso, a recuperação é feita sob a coordenação dos receptores mais próximos uns dos outros, evitando a sobrecarga gerada pelo processamento de pacotes de controle desnecessários.

Para fornecer informações aos receptores sobre os vizinhos é usado um mecanismo de traço de rota, que gera um *overhead* mínimo nos roteadores. Para ter um mecanismo de recuperação eficiente, observou-se que deveria haver uma comunicação explícita entre os receptores. Para isso, são escolhidos alguns deles como líderes do grupo de acordo com sua posição na árvore. Estes líderes enviam um pacote WAIT especial para os outros receptores designados em sua vizinhança. Estes pacotes funcionam como supressores de NACKs que possam vir a ser enviados por outros receptores, o que poderia levar a um congestionamento.

Para tratar a perda comum sofrida por alguns receptores abaixo do ponto da perda de uma forma efetiva, utiliza-se o mecanismo de *subcasting*, onde é feito *unicast* dos pacotes de reparo para a raiz da sub-árvore onde ocorreu a perda (o roteador que perdeu o pacote) e este faz o multicast para os receptores abaixo dele.

A abordagem apresentada tenta desenvolver um mecanismo onde nem todos os pacotes precisam ser recuperados, desde que ainda haja uma qualidade de multimídia aceitável para o usuário final. Sempre que os receptores registrarem mais perdas do que o estabelecido para a sessão, eles iniciam um processo de recuperação. A decisão de envio dos pacotes NACK e WAIT é tomada com base na porcentagem de perda, na latência esperada e no tamanho do *buffer* de reprodução.

Em resumo, o protocolo WAIT é uma abordagem para recuperação de perdas para aplicações de multicast de multimídia que leva em conta as características destas aplicações durante o processo de recuperação. Foi mostrado que levar ao conhecimento explícito para os outros receptores sobre a perda sofrida por um receptor pode auxiliar no processo. Descobrir a

raiz da sub-árvore onde ocorreu a perda ajuda a recuperação, reduzindo o *overhead* na rede e melhorando a latência ponto-a-ponto da sessão de multicast de multimídia.

4.3.8. Multicast Semanticamente Confiável

O multicast semanticamente confiável [PER03] foi desenvolvido para atender situações em que o mau desempenho de um ou parte dos receptores prejudica todo o grupo. Em vez de ajustar os parâmetros do emissor de dados ou excluir o receptor mais lento, a proposta é "enfraquecer" os requisitos de confiabilidade do sistema de forma que os receptores mais lentos não precisem receber todas as mensagens.

O multicast semanticamente confiável é um modelo baseado na obsolescência das mensagens. Uma mensagem torna-se obsoleta quando seu conteúdo ou seu propósito é sobreposto por outra mensagem. Este conhecimento é utilizado pelo protocolo para descartar algumas das mensagens dos *buffers* (do receptor e do emissor) em condições de sobrecarga (*overload*). Permitindo o descarte de mensagens obsoletas, o sistema consegue tolerar melhor as perturbações de desempenho sem precisar de recursos adicionais. Assim, garante-se também que os processos mais lentos recebam as mensagens "não-obsoletas", atingindo-se assim a confiabilidade semântica, um sistema onde toda a informação é entregue à aplicação mesmo que nem todas as mensagens o sejam.

A proposta é inspirada no uso de dois protocolos multicast em paralelo: um protocolo não confiável, usado para o conteúdo e um protocolo confiável, usado para os dados que descrevem o conteúdo das mensagens e informações relativas à sua obsolescência. Para evitar uma sobrecarga, quando o sistema não está congestionado esta informação não é levada em conta pelo protocolo e todas as mensagens são entregues à aplicação de maneira confiável. Usando a informação de obsolescência, o receptor também poderia avaliar a relevância das mensagens perdidas e pedir retransmissão quando for necessário. No entanto, a abordagem do multicast semanticamente confiável utiliza esta idéia apenas para explorar o conhecimento semântico no lado do emissor e evitar que a aplicação tenha que se envolver no tratamento de retransmissões.

O emissor envia as mensagens a uma determinada taxa de transmissão. Em cada receptor, as mensagens são colocadas em um *buffer* com capacidade limitada. Se uma mensagem não pode ser inserida em um dos *buffers*, o emissor fica bloqueado até que o espaço esteja novamente disponível. Um receptor mais rápido retira as mensagens de seu

buffer assim que elas chegam. Por outro lado, um receptor mais lento faz isso a uma determinada taxa. Em algum momento, o *buffer* do receptor mais lento ficará cheio. Quando isto acontece, o protocolo procura no *buffer* as mensagens obsoletas e as remove, liberando espaço para as mais recentes e desbloqueando o emissor.

É interessante notar que a retirada de mensagens obsoletas dos *buffers* começa pelos receptores mais lentos, que ficam com os *buffers* cheios antes dos demais. Isso reduz a probabilidade de receptores problemáticos perturbarem a ordem geral do sistema de transmissão.

4.3.9. LBRM

No LBRM (*Log-Based Receiver-Reliable Multicast*) [HOL95] a confiabilidade é obtida por um servidor de *logs* que grava informações sobre os pacotes transmitidos pela fonte, como ilustra a Figura 4-3. Quando um receptor detecta um pacote perdido, ele requisita o pacote ao servidor de *logs*. Este servidor não precisa estar acoplado com a fonte de dados, se os dois estiverem separados, a fonte precisa reter os dados até que tenha recebido um aviso de recebimento do servidor de *logs*.

O uso de um servidor de *logs* para confiabilidade generaliza o conceito de armazenamento em *buffers* dos protocolos de transporte tradicionais.

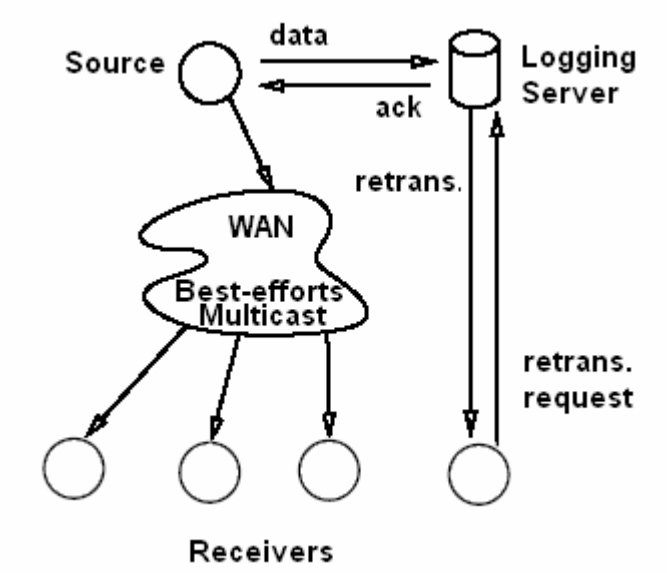

Figura 4-3. Confiabilidade no LBRM [HOL95]

Este protocolo é baseado no receptor, onde cada aplicação receptora define seus próprios requisitos de confiabilidade. O emissor apenas os torna possíveis através do serviço de *logs*, que permite a recuperação dos pacotes perdidos. Os receptores não são obrigados a recuperar todos os pacotes perdidos, e o emissor não confere se todos os receptores receberam cada um dos pacotes. Além disso, a ordem adequada dos pacotes é uma preocupação exclusiva da aplicação no receptor.

Nesta abordagem, três recursos precisam ser administrados com muito cuidado: largura de banda, carga computacional nos receptores e carga no servidor. Para tratar destes problemas, são apresentados alguns mecanismos: um pulso para rápida detecção de pacotes perdidos, um serviço de *logs* distribuído que reduz o número de pedidos de retransmissão e um mecanismo estatístico para detecção de perdas que escolhe dinamicamente entre estratégias de retransmissão *unicast* e multicast para melhorar o tempo de recuperação e minimizar a largura de banda utilizada.

4.3.10. Quadro Comparativo

A Tabela 4-1 apresenta um quadro comparativo dos protocolos de multicast semi-confiável descritos nesta seção, com seus pontos-chave e forma de retransmissão (ACK ou NACK).

A janela de confiabilidade apresenta-se como uma solução interessante para aplicações cujo conteúdo fica obsoleto rapidamente. Mas limitar as retransmissões apenas com base no número de seqüência e no parâmetro *forgetbefore* pode não ser a melhor alternativa para aplicações multimídia onde existe, por exemplo, uma ordem de importância diferenciada para cada tipo de dado transmitido.

O PTP pode funcionar bem na plataforma Prism, mas não é aplicável a outras plataformas por não usar o endereçamento padrão da Internet.

O método do SRP apresenta-se como uma boa solução para melhorar latência. Por utilizar os requisitos da aplicação para definir seus critérios de retransmissão, já é uma solução melhor do que a janela de confiabilidade para a transmissão de multimídia. A desvantagem desta abordagem é a necessidade de estar constantemente monitorando o tempo de envio das mensagens para os ajustes necessários no lado do emissor.

Tabela 4-1. Quadro comparativo dos protocolos de Multicast Semi-confiável

Protocolo	**Pontos-chave**	**Retransmissões**	**Emissor conhece os receptores?**
Janela de confiabilidade	Baseado na obsolescência das mensagens.	ACK/NACK	Sim
PTP (*Prism Transport Protocol)*	Usado para a plataforma Prism Janela de inibição para evitar explosão de NACKs.	NACK	Não
SRP (*Selective Retransmission Protocol)*	Pedido de retransmissão e retransmissão de pacotes com base nos requisitos de perda e latência da aplicação. Utilização de FEC.	NACK	Não
QMTP (*Quasi-reliable Multicast Transport Protocol*)	Combinação de controle de fluxo, transmissões redundantes e retransmissões ocasionais.	NACK	Não
TUNA (*TUNAble quasi-reliable multicast protocol*)	Baseado na obsolescência das mensagens. Utilização de FEC.	NACK	Não
XTP (*Xpress Transport Protocol*)	Adequado para sistemas de tempo real.	ACK	Sim
WAIT	Pedido de retransmissão e retransmissão de pacotes com base nos requisitos de perda e latência da aplicação.	NACK	Não
Multicast Semanticamente Confiável	Baseado na obsolescência das mensagens. Utiliza a retirada de mensagens dos *buffers*.	-	Sim
LBRM (*Log-Based Receiver-Reliable Protocol*)	Baseado em um servidor de *logs*	NACK	Não

O QMTP também apresenta descarte seletivo de pacotes de acordo com os requisitos da aplicação e avaliação de alguns parâmetros antes do envio de NACKs. Como a recuperação de erros não é baseada em retransmissão simples, mas através de FEC e/ou redução da taxa de transmissão, pode ocorrer de pacotes importantes não serem recuperados. Além disso, a técnica FEC empregada neste protocolo aumenta o consumo de largura de banda.

O TUNA também leva em conta a obsolescência das mensagens para decidir ou não pelo pedido de retransmissão. A recuperação de erros é limitada pela obsolescência limite (ajustável) das mensagens. Do ponto de vista estatístico, o método de recuperação parece eficiente, mas também não considera o caso de dados mais importantes que outros, deixando dúvida sobre sua aplicabilidade para multimídia.

O XTP é um protocolo desenvolvido para aplicações em tempo real onde o grau de confiabilidade é estabelecido pela aplicação. A proposta parece interessante, mas várias questões importantes não são mencionadas como o controle da latência, por exemplo.

As idéias propostas com o protocolo WAIT para recuperação seletiva de perda em sessões multimídia parecem bastante apropriadas, uma vez que ele é projetado para ser ajustado para diferentes requisitos de qualidade de uma sessão multimídia e reduzir o *overhead* na rede, além de tentar entregar melhor a qualidade das aplicações sobre a Internet com menor suporte do roteador.

O multicast semanticamente confiável endereça um problema que não parece ser tão crítico nos dias de hoje: situações em que um *host* seja tão lento a ponto de prejudicar o funcionamento de toda a comunicação multicast. O conceito de obsolescência das mensagens é bastante útil nas transmissões de multimídia, no entanto, o tratamento de erros baseado na manipulação dos *buffers* apresentado nesta abordagem não parece muito aplicável às redes atuais, pois os principais problemas encontrados estão relacionados a atrasos e perdas, não a *buffers* sobrecarregados.

Finalmente, o LBRM apresenta como inovação o conceito de recuperação através de um servidor de *logs*. Ficam dúvidas a respeito da eficiência deste método, uma vez que pode haver falhas na comunicação do emissor e deste servidor, ou deste servidor e dos receptores. E se o serviço de *logs* não for distribuído pode haver problemas ainda no tempo de recuperação dos pacotes perdidos.

4.4. Conclusão

Neste capítulo foram apresentadas as principais características dos mecanismos de multicast, apontando-se as principais características e representantes do tipo confiável e semi-confiável.

No capítulo anterior, viu-se que a transmissão de informações multimídia tem uma taxa de exibição fixa e os quadros devem ser recebidos e reproduzidos no receptor a uma taxa

similar para manter o significado original da seqüência. Assim, cada unidade de informação enviada deve ser recebida dentro de um determinado intervalo de tempo. Além disso, a fração de unidades de dados perdida (ou entregue após o instante no qual deveria ser exibida) deve estar também dentro de um limite aceitável. Portanto, o problema de QoS envolve encontrar os limites através das redes para erros de transmissão e possíveis flutuações entre os intervalos de entrega.

Considerando-se as informações apresentadas neste capítulo, pode-se concluir que um mecanismo de multicast confiável tradicional pode não ser adequado para este tipo de aplicação, por várias razões. Primeiramente, a estratégia de retransmissão com *timeout* obtém confiabilidade com um aumento de latência. Em segundo lugar, aplicações multimídia podem tolerar erros provenientes da perda e corrupção de pacotes dentro de um limite aceitável. Assim, um protocolo de transporte multimídia exige uma entrega semi-confiável dos pacotes, onde o atraso é mais relevante do que a entrega de todos os pacotes de um conjunto.

O próximo capítulo apresenta o protocolo proposto, descrevendo seus objetivos e forma de funcionamento.

Capítulo 5

Protocolo Proposto

Para adequar a transmissão multicast de forma a atender aos requisitos das aplicações multimídia distribuídas desenvolveu-se um protocolo de multicast semi-confiável para este tipo de aplicação. Adaptaram-se alguns conceitos usados em outros protocolos de multicast semi-confiável, apresentados no capítulo anterior, como a obsolescência das mensagens (utilizada nos protocolos: QMTP, TUNA, janela de confiabilidade e multicast semanticamente confiável) e a recuperação seletiva de pacotes, (citada no WAIT). Além disso, o protocolo proposto está baseado nas idéias do protocolo ReMIOP (*Reliable Multicast Inter-ORB Protocol*) [BES03] e em alguns conceitos de [LEI04] e [PIE00].

O protocolo proposto tira proveito da hierarquia de quadros definida no padrão MPEG em protocolos de difusão semi-confiável através da qualificação dos quadros MPEG e do encapsulamento desses em pacotes UDP. Uma vez qualificados, a recuperação de quadros (através da retransmissão dos pacotes perdidos) pode ser realizada pelo emissor (fonte) ou pelos receptores de acordo com o tipo de quadro perdido (I, P ou B) e, por exemplo, com os parâmetros da rede (tráfego, taxa de recepção, congestionamento, etc.). A idéia inicial é que todos os quadros do tipo I sejam entregues de forma confiável e que os quadros do tipo P e B sejam, principalmente o primeiro, retransmitidos (em caso de perda) se as condições do meio de transmissão permitirem.

Este capítulo apresenta o algoritmo desenvolvido e descreve os procedimentos para recepção e transmissão das mensagens no protocolo.

5.1. Funcionamento do Protocolo

A seguir é apresentado o algoritmo do protocolo proposto. Antes de detalhar seu funcionamento, algumas considerações iniciais precisam ser feitas:

- Assume-se que nem o emissor nem os receptores têm conhecimento dos membros do grupo multicast.
- O emissor envia apenas um quadro (‘I’, ‘P’ ou ‘B’) por pacote. Esta medida visa simplificar o processo de avaliação de relevância do pacote.
- Em cada pacote, além do quadro, são transmitidos: número de seqüência do quadro, tipo do quadro contido no pacote e quais os tipos dos oito quadros anteriores ao contido neste pacote. Estas informações são importantes no processo de avaliação de envio de NACKs e de retransmissão de pacotes, como será explicado posteriormente, pois garantem que, ao detectar um pacote perdido, o receptor consiga saber qual o tipo de quadro estava contido nele.
- Na implementação, tanto o emissor quanto os receptores possuem dois processos (*threads*) executados em paralelo: um para envio de mensagens ao grupo e outro para o recebimento. No emissor, enquanto um processo difunde as mensagens no grupo, o outro fica responsável por receber os pedidos de retransmissão que chegam. No receptor, enquanto um processo recebe as mensagens que chegam, o outro fica responsável pelo envio de NACKs e retransmissão de mensagens.
- Os valores para limitar o número máximo de pacotes perdidos foram estabelecidos empiricamente, bem como os parâmetros configurados no Simmcast para perda e delay, respectivamente 9% e 10 unidades de tempo da simulação.

ALGORITMO DO EMISSOR

```
A.1.   multicast(m)
A.2.   QUANDO R – recebe(m)
A.3.      SE m.tipo == NACK E m_N Î buffer_retransmissão
A.4.         SE m_N.quadro == “I”
A.5.            espera(aleatório (T))
A.6.            multicast(m_N)
A.7.         FIM-SE
A.8.         SE ((m_T.quadro == “P” E taxa_de_perda < 65%) OU
                (m_N.quadro == “B” E taxa_de_perda < 35% ))
A.9.               espera(aleatório (T))
A.10.              multicast(m_N)
A.11.        FIM-SE
A.12.     FIM-SE
A.13.  FIM-QUANDO
```

Figura 5-1. Algoritmo do Emissor

```
                              ALGORITMO DO RECEPTOR
B.1.  QUANDO R – recebe(m)
B.2.     SE m.tipo = DADOS
B.3.        SE agendado(m)
B.4.           cancela_agendado(m) {cancela o reenvio de m}
B.5.        FIM-SE
B.6.        quadro = m.tipo_do_quadro
B.7.        SE já_recebeu_mensagem(m) == FALSE
B.8.           cancela_agendado(NACKm) {cancela o envio de NACK para m}
B.9.           adiciona_ao_buffer (m.emissor, m)
B.10.          entrega(m)
B.11.          SE há mensagem faltando
B.12.             SE quadro == "I"
B.13.                espera(aleatório (T))
B.14.                multicast(NACKm)
B.15.             FIM-SE
B.16.          SENÃO SE ainda_relevante(quadro)
B.17.             taxa_de_perda = verifica_taxa_de_perda()
B.18.             SE ((quadro == "P" E taxa_de_perda < 65%) OU
                       (quadro == "B" E taxa_de_perda < 35%))
B.19.                espera(aleatório (T))
B.20.                multicast(NACKm)
B.21.             FIM-SE
B.22.          FIM-SE
B.23.       FIM-SE
B.24.    SENÃO SE m.tipo == NACK_N
B.25.       cancela_agendado(NACK_N) {cancela o envio de NACK para N}
B.26.       m_T = busca_no_buffer(m.emisor, m_n)
B.27.       SE m_T ¹ NULL
B.28.          SE m_T.quadro == "I"
B.29.             espera(aleatório (T))
B.30.             multicast(m_T)
B.31.          FIM-SE
B.32.          SENÃO
B.33.             taxa_de_perda = verifica_taxa_de_perda()
B.34.             SE ((quadro == "P" E taxa_de_perda < 65%) OU
                       (quadro == "B" E taxa_de_perda < 35%))
B.35.                espera(aleatório (T))
B.36.                multicast(m_T)
B.37.             FIM-SE
B.38.          FIM-SE
B.39.       FIM-SE
B.40.    FIM-SE
B.41. FIM-QUANDO
```

Figura 5-2. Algoritmo do Receptor

Os quadros (pacotes) são difundidos pelo emissor do grupo (linha A.1). Os receptores recebem estes quadros (linha B.1) e armazenam no *buffer* (linha B.9) para disponibilizá-los à aplicação. As perdas de pacotes são detectadas (linha B.11) quando os receptores encontram lacunas na seqüência dos pacotes recebidos, que é explicitada através de um número de ordem no cabeçalho de cada pacote. Quando um receptor detecta uma lacuna, ele avalia a necessidade de pedir a retransmissão do(s) pacote(s) perdido(s) (linhas B.12, B.16 e B.18).

Esta avaliação será feita com base nos requisitos de QoS da aplicação e nos parâmetros de rede. A QoS da aplicação leva em conta a relevância dos quadros perdidos para a reprodução da mídia no receptor, ou seja, se o quadro quando retransmitido ainda será útil no momento em que chegar. Os parâmetros de rede levam em conta a taxa de perda, o congestionamento da rede e o atraso aceitável para este pacote. Os valores que constam no algoritmo foram assumidos para possibilitar a simulação do protocolo, como será explicado a seguir.

Se um quadro perdido for considerado relevante, a retransmissão é solicitada através da difusão de uma mensagem de controle (NACK) no grupo (linhas B.14 e B.20). Caso contrário, o receptor simplesmente ignora o pacote perdido [Xu, 1997]. Qualquer processo (do emissor ou de um dos receptores) que receber este NACK e tiver o quadro solicitado, faz nova avaliação dos parâmetros (linhas A.4, A.8, B.28 e B.34) e, se for o caso, difunde-o novamente no grupo (linhas A.6, A.10, B.30 e B.36). As retransmissões são mais detalhadas na próxima seção.

5.2. Retransmissões

O protocolo baseia-se no princípio básico de retransmissão efetuada pelos receptores, onde estes compartilham a responsabilidade de ajudar os demais a recuperarem as perdas [FLO95]. Para análise dos parâmetros de decisão, convencionou-se empiricamente que a taxa de perda máxima para pedir retransmissão ou retransmitir pacotes contendo quadros do tipo B é de 35% e 65% para pacotes contendo quadros do tipo P. Isto significa que mesmo que 65% do total de pacotes estejam sendo perdidos, um pacote contendo um quadro P terá chance de ser retransmitido. Já a “tolerância” para pacotes contendo quadros B é menor, se 35% do total de pacotes forem perdidos, ele poderá não ser retransmitido.

A taxa de perda em cada receptor é calculada a partir de uma seqüência de amostragem: os últimos N pacotes difundidos. A cada N pacotes dessa seqüência verificam-se a quantidade de pacotes recebidos e as lacunas na seqüência. Portanto, a função verifica_taxa_de_perda calcula a porcentagem de pacotes perdidos a cada N pacotes difundidos. Na implementação, utilizou-se uma fila de N posições (N=50) com sistema FIFO (*First In First Out*), onde cada elemento recebe um indicador (verdadeiro ou falso) de recepção de pacotes. Cada lacuna detectada pelo receptor acrescenta um indicador com valor falso ao final da fila. A porcentagem de indicadores falsos em relação ao total de elementos da fila determina a taxa de perda. A Figura 5-3 mostra um exemplo do cálculo desta taxa.

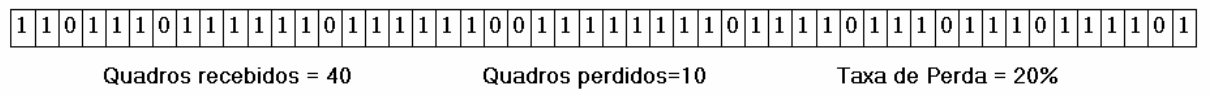

Figura 5-3. Cálculo da taxa de perda

A relevância de um quadro é verificada através do *buffer* de reprodução. A função verifica_relevância checa se o quadro que está faltando já está obsoleto, ou seja, se o momento de sua reprodução pela aplicação já passou. Na prática, verifica-se o número de seqüência dos quadros que estão *buffer* de reprodução e, de acordo com o número de seqüência do quadro perdido, é possível saber se ele já está obsoleto ou não.

Para evitar tanto a explosão de NACKs como a de retransmissões, é usada uma função de espera que interrompe a execução durante um tempo aleatório (T). Quando um receptor está prestes a enviar um NACK, ele espera por um tempo aleatório T (linhas B.13 e B.19). Se durante este tempo o receptor receber um NACK de um outro pedindo os mesmos pacotes, ele cancela a difusão do seu NACK (linha B.25). De modo similar, quando um receptor recebe um NACK e pode atendê-lo, espera por um tempo aleatório T antes de retransmitir o(s) pacote(s) solicitado(s) por esse NACK. No entanto, se durante o tempo T o receptor receber alguns ou todos os pacotes solicitados no NACK, pode cancelar a difusão dos pacotes já retransmitidos (linha B.4). O objetivo do tempo T é evitar que todos os processos do grupo atendam ao mesmo NACK e também diminuir a sobrecarga da rede.

5.3. *Buffer* de Reprodução e de Retransmissão

Um ponto importante para a implementação do protocolo consiste no gerenciamento do *buffer* de reprodução no receptor de forma a tratar adequadamente as perdas indiretas. Perdas indiretas ocorrem, por exemplo, quando um quadro P ou B não pode ser reproduzido devido à perda de um quadro do tipo I anterior a eles. Para auxiliar a solução do problema, foram criadas duas classes: *WaitElement* e *WaitWindow*.

A classe *WaitElement* é associada a um GOP. Ela representa um conjunto de pacotes recebidos onde o primeiro deles contém um quadro do tipo I. Um objeto *WaitElement* instancia três parâmetros:

- O número de seqüência do quadro I daquele conjunto,
- Um *flag* que indica se o quadro I já está presente no *buffer* auxiliar, e

- O *buffer* auxiliar, que armazena os pacotes daquele conjunto.

A Figura 5-4 ilustra este conceito.

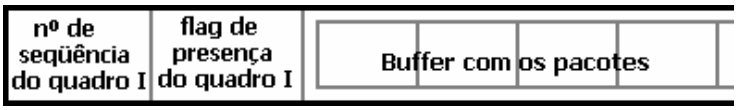

Figura 5-4. Classe WaitElement

A classe *WaitWindow* representa um *buffer* de armazenamento dos objetos *WaitElement*. Os pacotes que chegam ao receptor são colocados dentro do *WaitElement* correspondente, de acordo com o número de seqüência do quadro que contêm. Os objetos *WaitElement* são armazenados no objeto *WaitWindow* do receptor, ordenados pelo número de seqüência do quadro I do conjunto. A Figura 5-5 mostra como ficaria instanciada a classe *WaitWindow*, para a uma dada seqüência onde todos os quadros foram recebidos. A Figura 5-6 ilustra como seria a instância para a mesma seqüência caso alguns quadros fossem perdidos.

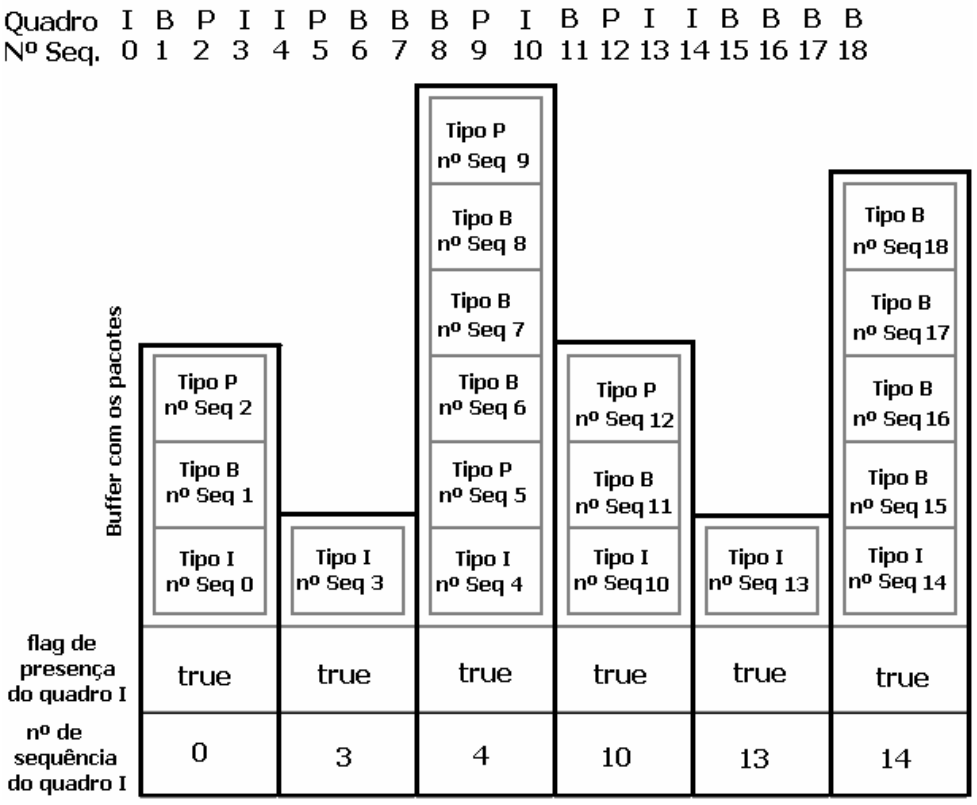

Figura 5-5. Objeto da classe *WaitWindow*

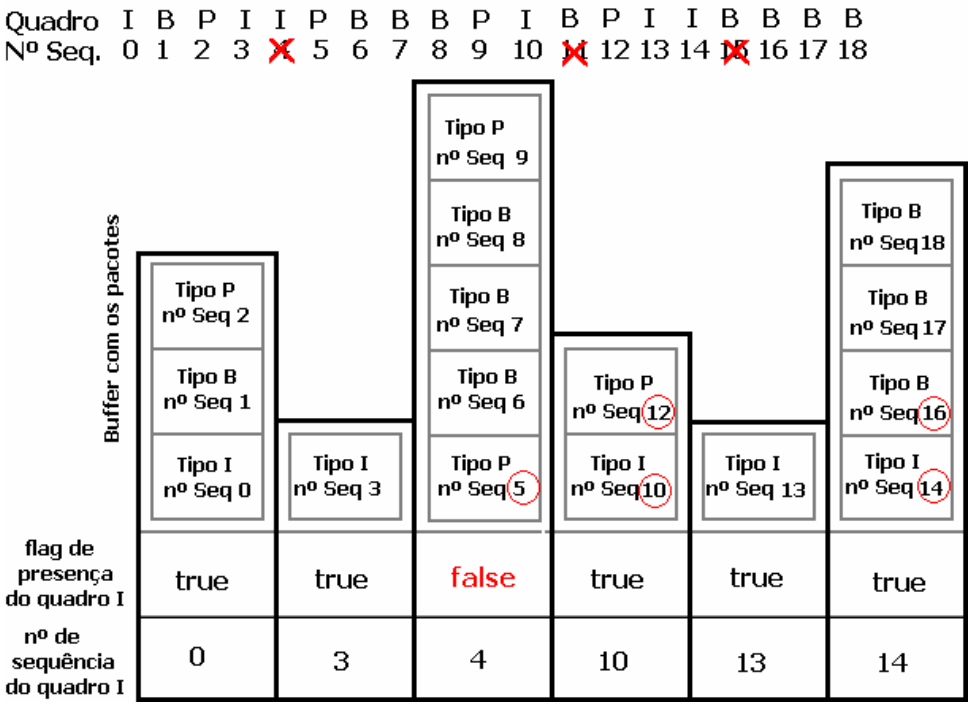

Figura 5-6. Objeto da classe *WaitWindow*

Quando um receptor recebe um pacote contendo um quadro I, é criada uma nova instância da classe *WaitElement*, com o número de seqüência deste quadro, o *flag* com o valor ‘verdadeiro’ e o quadro colocado no *buffer* auxiliar. Este objeto é então adicionado ao objeto *WaitWindow* na posição correspondente, de acordo com o número de seqüência do quadro I. Quando chega um pacote contendo um quadro P ou B, procura-se dentro do objeto *WaitWindow*, o *WaitElement* que contenha o quadro I que o precede e o pacote é adicionado ao respectivo *buffer* auxiliar na posição correta.

Quando é detectada a perda de um pacote contendo um quadro do tipo I, é criada uma instância de *WaitElement* com o número de seqüência do pacote que foi perdido e com o *flag* assumindo o valor ‘falso’, indicando que o quadro I daquele conjunto não está no *buffer* auxiliar. Dessa forma, se chegar o momento de reproduzir este conjunto de quadros antes da recepção do respectivo quadro I, todos os quadros daquele conjunto serão descartados para apresentação e o próximo conjunto que contenha um quadro I é que será colocado no *buffer* de reprodução. Se o quadro I correspondente a este conjunto chegar a tempo, ele é adicionado ao *buffer* auxiliar na posição correta e o *flag* passa a ter o valor ‘verdadeiro’.

Esse algoritmo permite que, quando os quadros são enviados para o *buffer* de reprodução, eles já estejam ordenados e que os descartes de quadros que sofreram perda indireta já tenham sido realizados.

A retransmissão de pacotes também ocorre a partir do objeto *WaitWindow*, uma vez que ele armazena todos os pacotes até o momento da reprodução dos quadros.

Com o emprego desta estrutura, cada receptor deve possuir uma entidade para cuidar do envio dos quadros para a aplicação. Isto é feito passando-se os quadros que estão na *WaitWindow* para o *buffer* de reprodução que será lido pela aplicação.

Além do *WaitWindow*, cada receptor possui um *buffer* de retransmissão que guarda os últimos pacotes (quadros) enviados para a reprodução. Os quadros que estão no *buffer* usado para reprodução é também usado para retransmissão, evitando a necessidade de redundância. Este *buffer* é utilizado por um receptor para atender pedidos de retransmissão de pacotes (NACK). O tempo em que os pacotes ficam disponíveis depende do tamanho deste *buffer*, que é limitado. Os pacotes mais antigos são descartados utilizando-se o método FIFO.

5.4. Controle de Fluxo

Para evitar congestionamentos e a explosão de NACKs é necessário que haja um controle de fluxo. No protocolo proposto, este controle foi feito de forma muito simplificada através dos mecanismos já descritos para cancelamento de NACKs e de retransmissões de mensagens.

Um mecanismo bastante interessante que poderia ser adaptado é o apresentado no LRMP [LIA98], onde o controle de fluxo e congestionamento é executado no emissor de acordo com as informações recebidas pela rede. Segundo [LIA98], o problema de congestionamento pode ser detectado pelo aumento do número de retransmissão ou pela indicação recebida dos receptores. O emissor mantém os dados transmitidos armazenados em um *buffer* para possíveis retransmissões. O tamanho deste *buffer* é que limita quantas mensagens podem ser retransmitidas.

Para responder rapidamente ao problema de congestionamento, a idéia é utilizar o menor número de seqüência de pacote cuja retransmissão foi solicitada. Se este número estiver fora da faixa abrangida pelo *buffer* de retransmissão, isso quer dizer que o emissor deve calcular a diferença entre o número de seqüência que está sendo transmitido e o mais baixo cuja retransmissão foi solicitada, para obter a taxa de transmissão atual R:

- Se a diferença for maior que metade do tamanho do buffer, a taxa de transmissão deve ser reduzia a R/4
- Se for maior que um terço do tamanho do buffer, a taxa deve ser reduzida a R/2
- Se for maior que um quarto do tamanho do buffer, a taxa deve ser reduzida a 3R/4.

Quando a indicação de congestionamento parte do receptor, um *flag* é usado para informar ao emissor. Ele deve ser "ligado" quando a diferença entre o último número de seqüência entregue à aplicação e o número de seqüência mais alto recebido do emissor for maior que a metade do tamanho buffer de recepção. Isto significa que o número de pacotes que sobram será muito alto para que o receptor consiga fazer reparo das perdas por falta de espaço em seu buffer. Ao receber esta indicação, o emissor deve reduzir sua taxa de transmissão para R/2, reduzindo a qualidade.

Se não ocorrer nada que exija uma diminuição na taxa de transmissão, e um oitavo do tempo do buffer de transmissão se passou desde o último aumento ou decréscimo, a taxa deve ser aumentada por um fator de 0,125 até a taxa máxima.

A adaptação desta abordagem também poderá contemplar a hierarquia dos quadros MPEG, de forma a ajustar a taxa de transmissão para cada tipo de quadro separadamente. Isto poderá melhorar a eficácia do protocolo, com maior garantia de entrega dos pacotes contendo quadros mais relevantes para a aplicação. Um cuidado necessário é adaptar esta idéia de forma que a qualidade não seja sempre nivelada por baixo, estabelecendo-se limites mínimos para R.

5.5. Conclusão

Este capítulo apresentou a proposta de um protocolo de multicast semi-confiável que busca aperfeiçoar a utilização de dados de vídeo que trafegam pela rede. Com base no padrão MPEG, este protocolo garante a entrega confiável dos quadros de vídeo ou parte deles. O protocolo proposto faz uso da técnica de retransmissão baseada no receptor e utiliza o descarte seletivo dos pacotes perdidos de acordo com a importância do quadro MPEG neles contidos.

As retransmissões podem acontecer a partir tanto do emissor quanto dos receptores. As estruturas de *buffers* criadas (classes *WaitElement* e *WaitWindow*) visam melhorar o processo de retransmissão bem como o envio correto dos quadros para a aplicação.

No próximo capítulo serão apresentados os resultados obtidos a partir das simulações do protocolo proposto e as conclusões correspondentes.

Capítulo 6

Simulação e Avaliação do Protocolo

Para avaliar o protocolo proposto, utilizou-se o Simmcast [BAR01]. Foram feitas comparações de desempenho em termos de recuperação de erros, tempo de recuperação, sobrecarga nos receptores e fator de qualidade de vídeo [LEI04] entre o protocolo proposto, um protocolo de multicast simples, um protocolo de multicast com envio de NACKs e retransmissões a partir da fonte e um protocolo de multicast confiável com retransmissão a partir dos receptores. Foram utilizadas duas topologias diferentes para mostrar as potencialidades do protocolo, especialmente em redes de larga escala.

Este capítulo apresenta o Simmcast [BAR01], mostra os resultados obtidos nas simulações e a avaliação dos mesmos.

6.1. Simmcast

Para as simulações do protocolo proposto escolheu-se um simulador de rede denominado Simmcast [BAR01]. O Simmcast é definido por seus criadores como “um *framework* para simulação discreta de eventos baseada em processos que contribui para o projeto e avaliação de protocolos multicast [BAR01]”. Este *framework* é baseado em Java, permitindo que os protocolos simulados sejam especificados utilizando-se esta linguagem e que os códigos utilizados nas simulações sejam utilizados mais tarde na implementação do protocolo. Escolheu-se esta ferramenta especialmente por ela oferecer um modelo para simulação bastante simples de entender e manipular, que permite aos usuários estabelecer o nível de detalhe desejado suas simulações. Uma leitura cuidadosa do tutorial [BAR05] possibilita ao usuário compreender seu funcionamento e utilizar todas as suas facilidades.

6.1.1. Características do Simmcast

O Simmcast [BAR01] é baseado na linguagem Java. Sua API (*Application Program Interface*) fornece as operações típicas de comunicação e temporização, bem como uma arquitetura apropriada para projeto e avaliação de protocolos multicast. Para executar uma simulação, o usuário precisa apenas adicionar ou estender as classes/interfaces do *framework* de acordo com as configurações do protocolo que deseja simular.

Conforme já foi mencionado, o ponto forte do Simmcast é sua simplicidade. Há apenas dez operações "primitivas" fornecidas ao usuário:

- *send:* envio de um pacote a um endereço destino
- *receive:* pedido de recepção de um pacote, bloqueando o processo até que o pacote esteja disponível.
- *tryReceive:* tenta receber um pacote, sem bloquear o processo, pois retorna "nulo" se o pacote não está disponível.
- *join:* juntar-se ao grupo de multicast
- *leave:* sair do grupo de multicast
- *setTimer:* configura um temporizador para expirar em um dado tempo
- *cancelTimer:* cancela o temporizador
- *onTimer:* método chamado quando o temporizador expira
- *sleep:* suspende a execução do processo durante um tempo determinado
- *wakeUp:* reinicia um processo suspenso.

Os blocos a serem combinados na construção do protocolo possuem uma classe associada a cada um. Os principais são:

- *Node:* é a entidade fundamental. Representa os nós que são conectados ao longo da rede. Também pode representar roteadores. Um objeto *Node* pode conter um ou mais processos *(thread)* executando simultaneamente.
- *Link:* é a conexão entre os nós. Suas propriedades são: largura de banda *(Bandwidth),* probabilidade de perda de pacotes *(Loss_conditions),* e atraso de entrega *(Delay).*
- *Group:* representa o grupo multicast.
- *Network:* representa a rede, uma combinação de nós *(Node)* e conexões *(Link).*
- *Packet:* representa os pacotes, a unidade de comunicação entre os nós da rede.

- *Thread:* é nesta classe que o usuário insere a lógica de seu protocolo, definindo as ações dos nós.
- *Timer:* classe utilizada para implementar eventos assíncronos.

A seguir descreve-se a utilização destes elementos para criar e executar uma simulação no Simmcast [BAR01].

6.1.2. Execução da Simulação

Para executar uma simulação no Simmcast, é preciso definir a lógica do protocolo e a topologia da rede. O protocolo é construído pela combinação dos blocos de construção (seção 6.1.1) e especialização da classe *Node* e das classes relacionadas a ela. A lógica de protocolo, ou seja, o comportamento dos nós, deve ser inserida em uma classe *NodeThread* associada aos nós criados pela classe *Node*.

Definido o protocolo, é preciso estabelecer sua conectividade e suas instâncias para definir a topologia da rede. É possível usar o padrão do Simmcast, onde a rede é um conjunto de conexões ponto-a-ponto entre os objetos da classe *Node*.

A definição da topologia é feita com a preparação de um arquivo de simulação com a descrição. Neste arquivo, com extensão ".sim", são definidos os nós, conexões e parâmetros da rede. No momento da execução da simulação (*run_simulation*), é necessário especificar o caminho deste arquivo na linha de comando.

6.1.3. Inserindo o Protocolo Proposto

Para simular o protocolo proposto, o primeiro passo foi criar o nó emissor e os nós receptores. Para isso, estendeu-se a classe *Node* para serem criadas as classes *SourceNode* e *SinkNode*, representando respectivamente a fonte de dados e o receptor.

Criados os nós, foram feitas quatro extensões da classe *NodeThread* para inserção da lógica do protocolo: *SourceThread*, *SourceReceiverThread*, *SinkThread*, *SendNackThread*.

A classe *SourceThread*, derivada da *SourceNode*, contém a lógica do emissor para o envio dos pacotes e controle de taxa de perda. A classe *SourceReceiverThread*, também derivada da *SourceNode*, contém a lógica do emissor para o monitoramento dos NACKs que chegam, avaliação dos parâmetros de controle e reenvio de pacotes. A classe *SinkThread*, derivada da *SinkNode*, contém a lógica do receptor para o recebimento e tratamento das

mensagens. A classe *SendNackThread*, também derivada da *SinkNode*, contém a lógica do receptor para decisão sobre o envio ou não de NACKs e de retransmissões de pacotes.

O diagrama de classes pode ser observado na Figura 6-1.

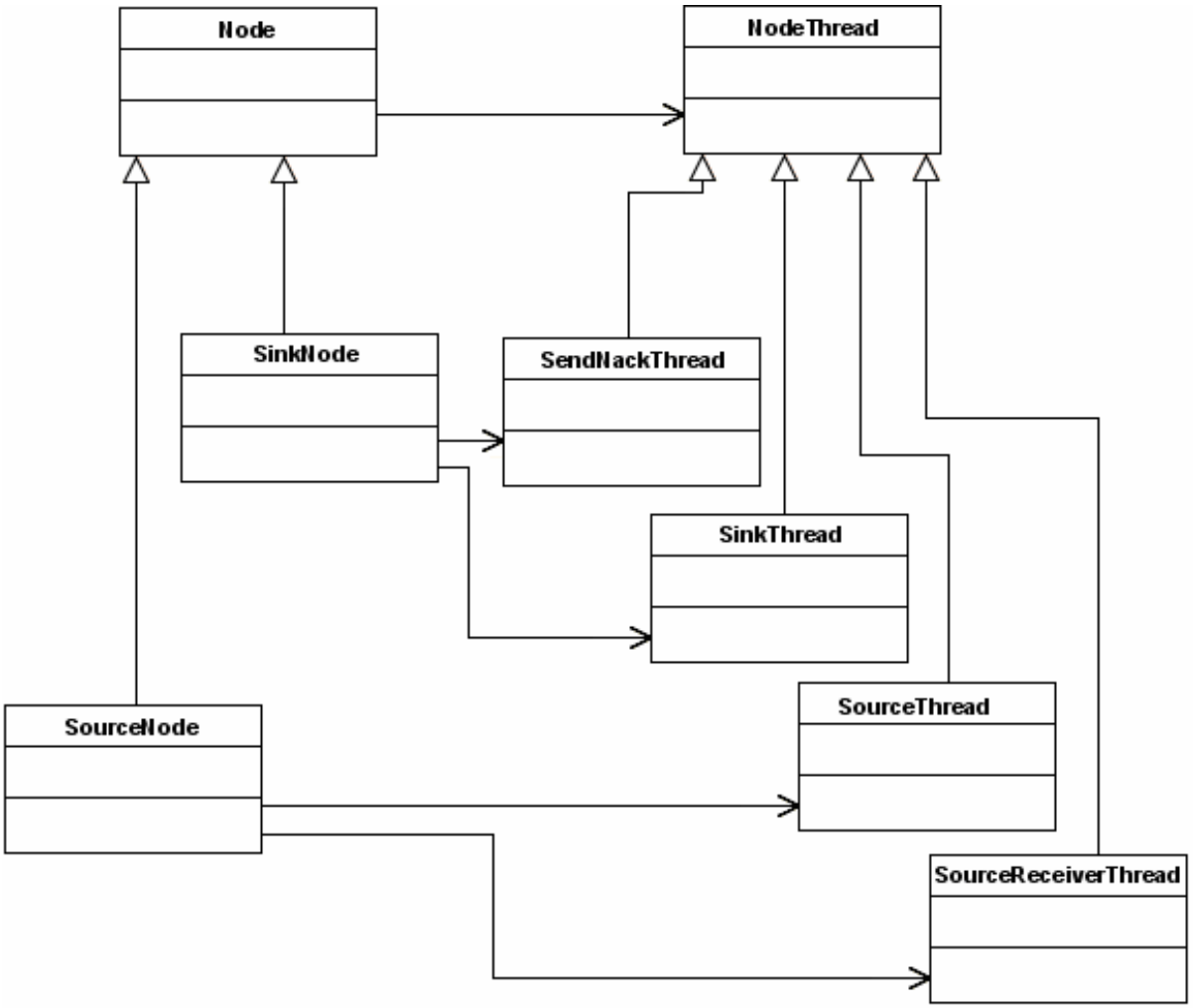

Figura 6-1. Classes do protocolo proposto para simulação no Simmcast

Todos os códigos-fonte do protocolo foram escritos com base em exemplos do próprio Simmcast.

Para visualizar a simulação, foram colocados no código alguns pontos em que se imprimem na tela as ações que estão sendo tomadas, como pode ser visto na figura 6-2. Para análise dos resultados, foram gerados arquivos de *log* que foram processados posteriormente em softwares de geração de planilha e gráfico.

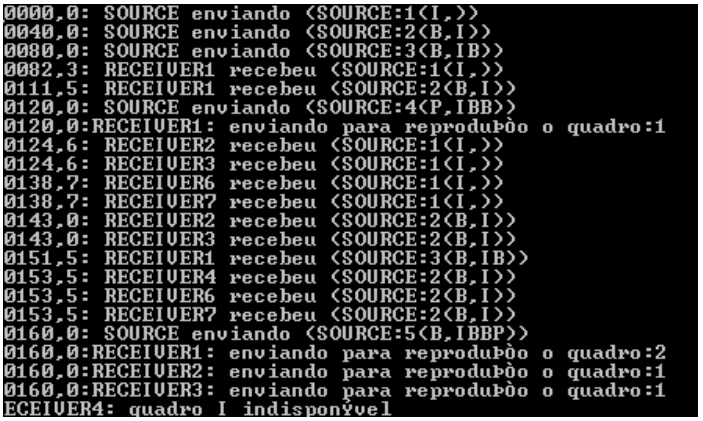

Figura 6-2. Imprimindo a simulação

6.2. Simulações e Resultados

As simulações foram realizadas para os quatro protocolos utilizando-se as mesmas condições de envio, ou seja, com os mesmos dados sendo transmitidos sob as mesmas condições de rede:

- Protocolo proposto;
- Multicast simples, onde a fonte realiza a difusão dos pacotes sem que haja qualquer tentativa de recuperação de perdas;
- Multicast com retransmissão simples, onde os receptores que detectam perdas solicitam retransmissão enviando um NACK para a fonte; e
- Multicast confiável com retransmissão a partir dos receptores vizinhos, onde há a tentativa de recuperar todos os pacotes perdidos.

Foram utilizadas duas topologias, que estão ilustradas nas Figuras 6-3 e 6-4. Configurou-se o arquivo de simulação do Simmcast com os dados mostrados na figura 6-5. O tamanho estabelecido para os *buffers* foi 50.

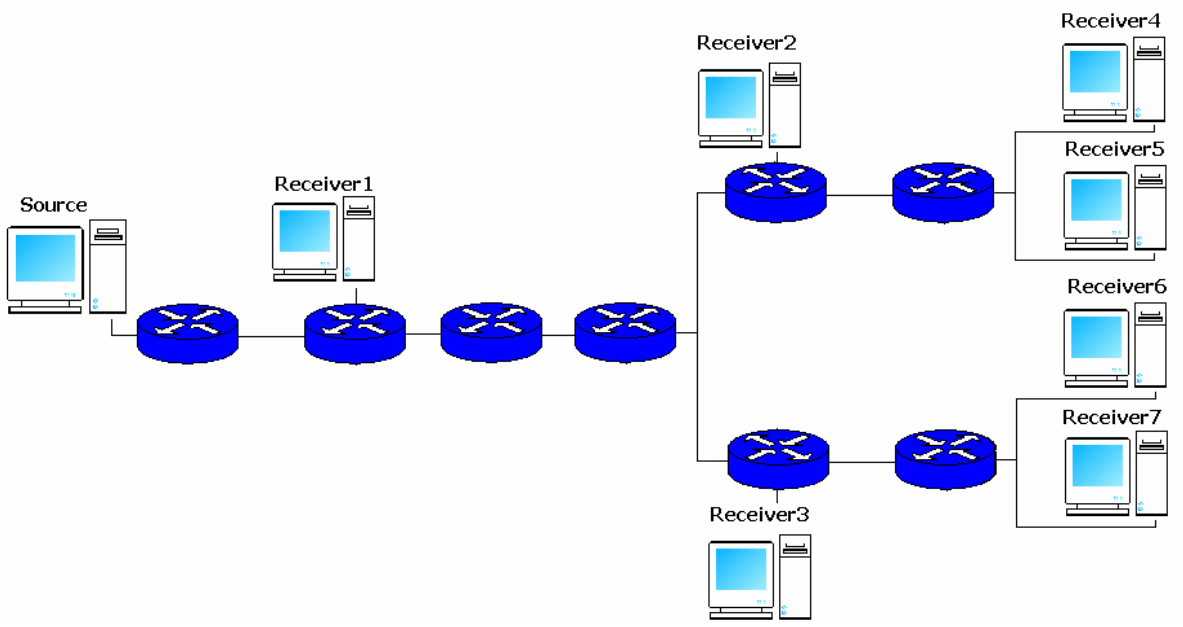

Figura 6-3. Topologia utilizada nas simulações – Cenário1

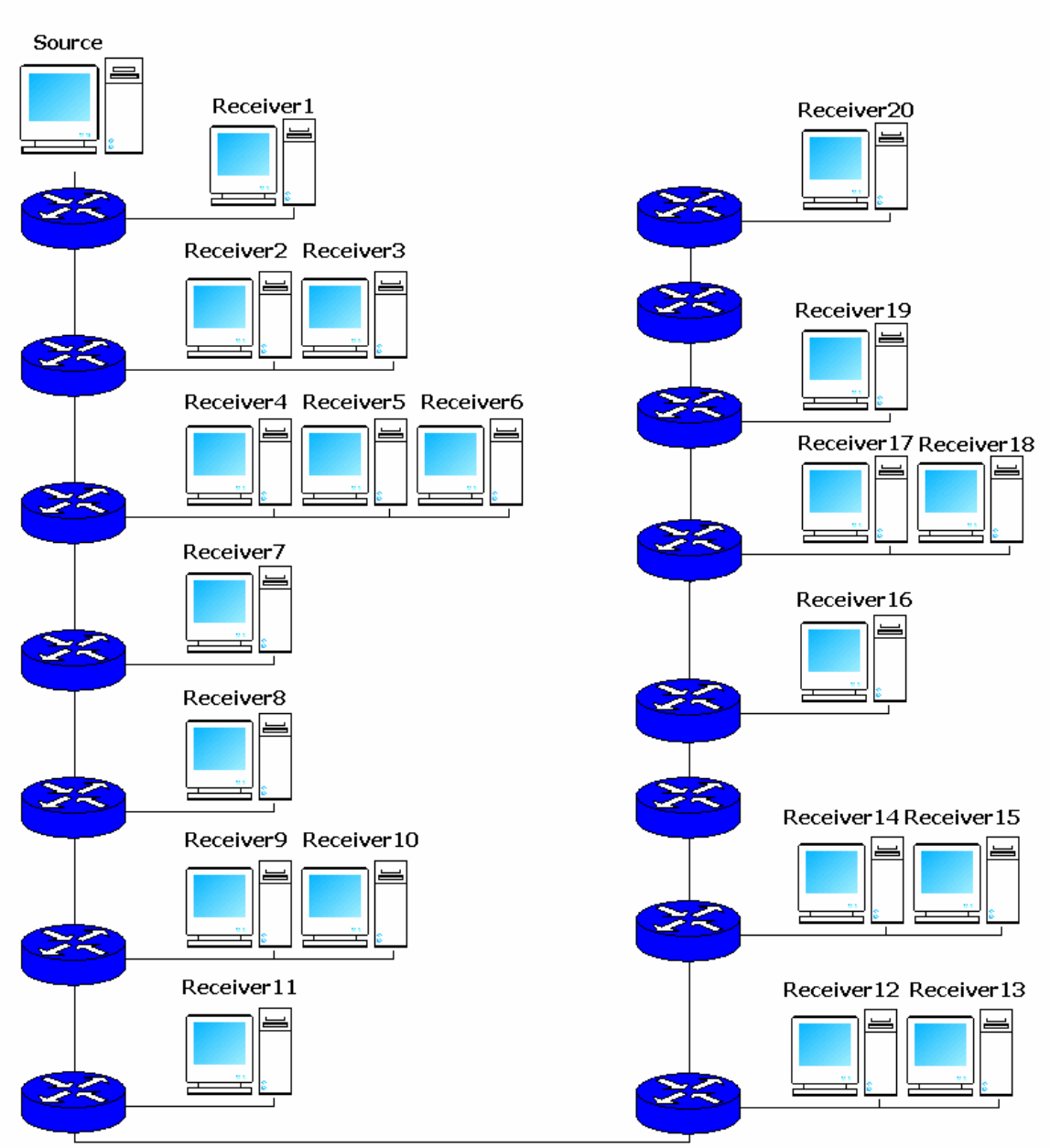

Figura 6-4. Topologia utilizada nas simulações – Cenário 2

```
macro !BANDWIDTH "10"
macro !DELAY "10.0"
new FIXED_DELAY simmcast.stream.FixedStream !DELAY
macro !QUEUESIZE "50"
macro !NO_LOSS "0.00"
macro !WITH_LOSS "0.09"
macro !N "10"
macro !LOSS_CONDITIONS "!WITH_LOSS"
```

Figura 6-5. Dados de simulação

Em termos de recuperação de pacotes não recebidos, o protocolo proposto apresentou uma média de 12,5% de recuperação (de todos os quadros perdidos) contra 8,5% do multicast com retransmissão simples e 82,5% do multicast confiável para o cenário 1. Para o cenário 2, o resultado foi de 68% de recuperação do protocolo proposto contra 4% do multicast com retransmissão simples e 82,35% do multicast confiável. O melhor desempenho do protocolo proposto e do multicast confiável pode ser atribuído, entre outras coisas, às retransmissões feitas a partir dos receptores vizinhos. A taxa de recuperação maior do multicast confiável em relação aos demais já era esperada, uma vez que ele busca recuperar todos os pacotes, independente de sua relevância temporal para a aplicação.

Quando analisados somente os pacotes que contêm quadros do tipo I, no cenário 1, o protocolo proposto apresentou uma recuperação de 42,3% de todos os quadros I, enquanto o multicast com retransmissão simples não recuperou nenhum quadro deste tipo e o multicast confiável, 94,1%. No cenário 2, o protocolo proposto apresenta uma média de 81,7% de recuperação contra 1% do multicast com retransmissão simples e 97% do multicast confiável. Este resultado permite afirmar que as retransmissões seletivas enfatizam a recuperação dos quadros mais importantes. Os valores médios da taxa de recuperação por tipo de quadro para cada cenário são mostrados na Tabela 6-1.

Tabela 6-1: Recuperação por tipo de quadro

	% de quadros recuperados					
	Cenário 1			**Cenário 2**		
	I	**P**	**B**	**I**	**P**	**B**
Protocolo Proposto	42,3	18,5	4,9	81,7	82,9	61,7
Multicast com retransmissão simples	0,0	1,6	4,9	1,0	2,1	3,4
Multicast Confiável com retransmissão a partir dos receptores	94,1	67,4	80,3	97,0	100,0	71,0

Outro aspecto interessante a ser observado é que a média de pacotes recuperados varia menos entre os receptores quando usado o protocolo proposto ou o multicast confiável com retransmissão a partir dos receptores. Para o multicast com retransmissão simples, esta média diminui bruscamente à medida que o receptor está mais distante da fonte transmissora, como ilustram os Gráficos 6-1 e 6-2. Para o cenário 1, mostrado no Gráfico 6-1, isso não fica tão evidente devido à menor variação da distância dos receptores em relação ao emissor neste cenário.

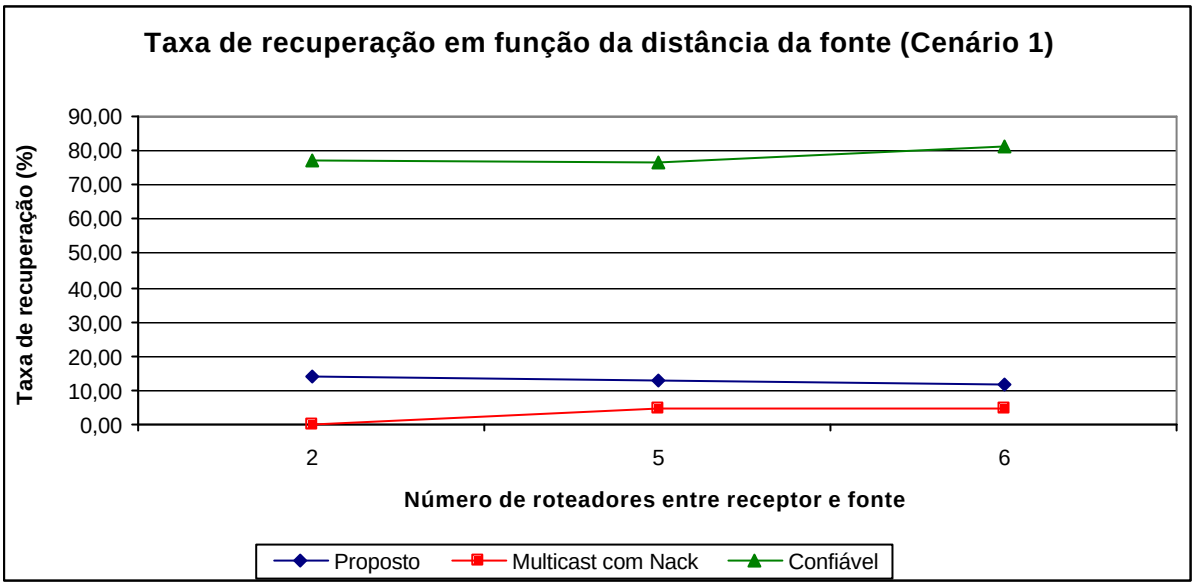

Gráfico 6-1. Recuperação em função da distância da fonte (Cenário 1)

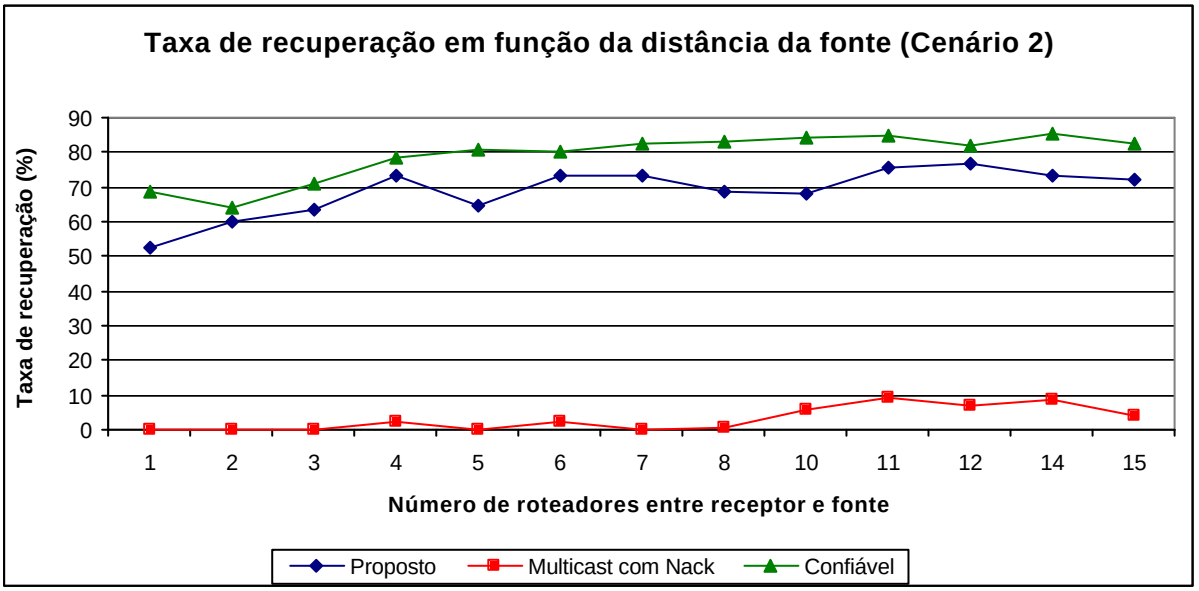

Gráfico 6-2. Recuperação em função da distância da fonte (Cenário 2)

Observou-se ainda que, para o protocolo proposto, a grande maioria dos pacotes recuperados nos receptores vem de seus vizinhos e não da fonte. No primeiro, 71% do total de pacotes recuperados foram transmitidos pelos próprios receptores, enquanto no segundo cenário a proporção foi de 93%. Esta diferença entre os dois cenários pode ser explicada pelo menor número de receptores do cenário 1 e a maior distância entre os grupos de receptores.

É apresentada na Tabela 6-2 uma observação mais detalhadas das fontes de recepção dos pacotes recuperados para cada receptor no cenário 1 (Figura 6-3). É possível perceber a forte participação dos receptores vizinhos no processo de recuperação dos pacotes.

Tabela 6-2. Fonte de Recuperação (Cenário 1)

Receptor	**FONTE DE RECUPERAÇÃO (Cenário 1)**							
	Receiver1	**Reciever2**	**Receiver3**	**Receiver4**	**Receiver5**	**Receiver6**	**Receiver7**	**Source**
Receiver1	0%	0%	0%	0%	0%	17%	0%	83%
Receiver2	33%	0%	11%	11%	11%	11%	0%	22%
Receiver3	33%	17%	0%	0%	0%	0%	17%	33%
Receiver4	33%	17%	17%	0%	0%	0%	33%	0%
Receiver5	13%	25%	38%	0%	0%	0%	13%	13%
Receiver6	25%	25%	13%	0%	13%	0%	0%	25%
Receiver7	29%	14%	14%	0%	0%	14%	0%	29%

A retransmissão seletiva a partir dos receptores implica também em um desempenho melhorado em termos de tempo de recuperação de pacotes perdidos no protocolo proposto. Para o primeiro cenário, o protocolo proposto apresenta uma média de 354ms de tempo de recuperação contra 1706ms do multicast com retransmissão simples e 473ms do multicast confiável. Para o segundo cenário, foram 95ms do protocolo proposto contra 962ms do multicast com retransmissão e 140ms do multicast confiável.

Ainda a respeito das retransmissões, outro ponto interessante a se avaliar é a quantidade de pacotes descartados pelos emissores, ou seja, pacotes que chegaram tarde demais para serem aproveitados. As informações mostradas nos Gráficos 6-3 e 6-4 mostram que a taxa de recuperação de pacotes é superior à de descarte[1]. Observa-se também que a maior parte dos pacotes descartados é do tipo I, um resultado natural uma vez que para este tipo de quadro não é feita nenhuma avaliação antes da retransmissão, isto é, sempre que algum nó recebe um NACK para um quadro I, o mesmo é retransmitido a não ser que antes chegue outro pacote já contendo a retransmissão (seção 5.2). A superioridade do desempenho

[1] Nestes gráficos foram considerados apenas os descartes de quadros que chegaram aos receptores tarde demais, não aqueles feitos a partir do objeto *WaitWindow* (seção 5.3) devido à falta de um quadro I.

do protocolo proposto no Cenário 2 evidencia sua melhor aplicabilidade para redes de larga escala: taxas de recuperações muito boas para todos os tipos de quadro e taxa de descarte de pacotes relativamente baixa, mostrando que a maior parte das retransmissões ocorre de forma bastante eficiente, atendendo às necessidades dos receptores.

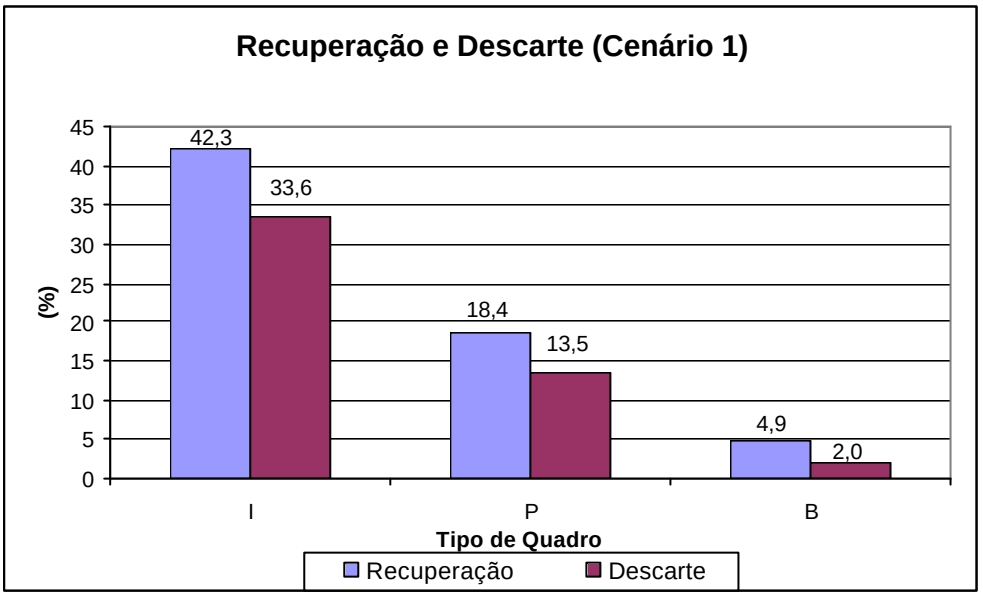

Gráfico 6-3. Recuperação e descarte de pacotes (Cenário 1)

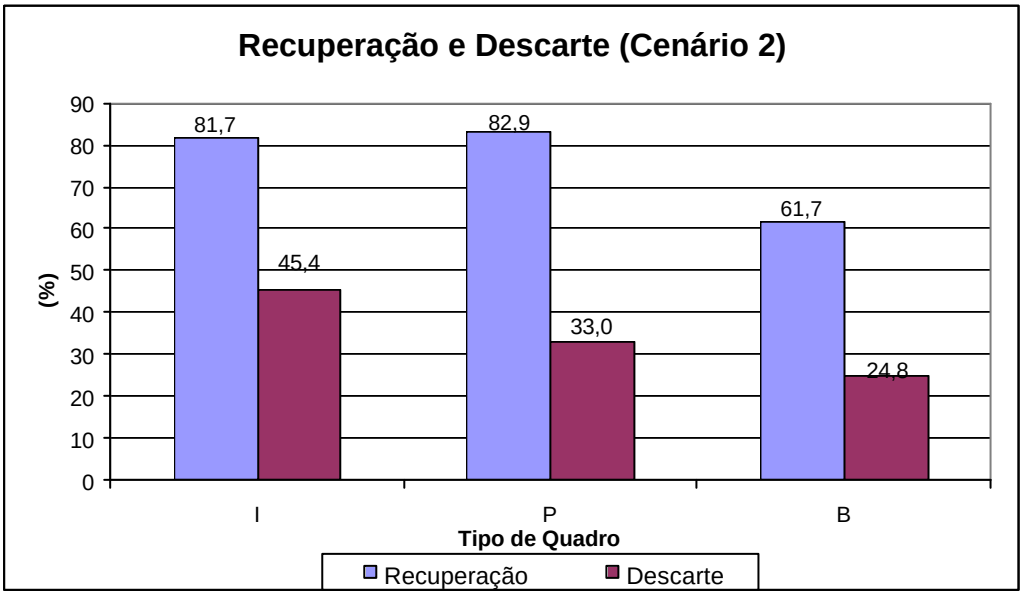

Gráfico 6-4. Recuperação e descarte de pacotes (Cenário 2)

Ainda quanto aos descartes, cabe uma comparação entre o protocolo proposto e o multicast confiável com retransmissão a partir dos receptores. Os dados da Tabela 6-3 mostram que a retransmissão seletiva dos pacotes com base em seu conteúdo faz com que as retransmissões sejam melhor aproveitadas, ou seja, otimizando os recursos da rede para que não sejam minimizadas as retransmissões de pacotes que não serão úteis no momento que chegarem à aplicação.

Tabela 6-3. Descarte por tipo de quadro

	% de quadros descartados					
	Cenário 1			**Cenário 2**		
	I	**P**	**B**	**I**	**P**	**B**
Protocolo Proposto	33,6	13,5	2,0	45,4	33,0	24,8
Multicast confiável	46,7	16,1	23,9	78,0	50,6	48,1

Quanto à sobrecarga (*overhead*) causada na rede, o protocolo proposto apresentou em média um aumento quase cinco vezes maior no número de pacotes enviados do que o multicast com retransmissão simples. Convém destacar que este aumento é um resultado de certa forma esperado uma vez que, no protocolo proposto, as retransmissões ocorrem a partir de todos os nós da rede e não somente da fonte. Quando comparado ao multicast confiável com retransmissão a partir dos receptores, o número de pacotes retransmitidos pelo protocolo proposto é duas vezes menor.

Um resultado bastante positivo foi a omissão de NACKs no protocolo proposto: em média, 37% dos NACKs foram evitados no cenário 1 e 33%, no cenário 2. Observou-se que essa porcentagem aumenta em função da distância da fonte, como mostram os Gráfico 6-5 e 6-6, para os cenários 1 e 2 respectivamente. Isto ocorre porque os receptores mais próximos da fonte detectam as lacunas na seqüência mais rapidamente enviando seus NACKs antes que os mais afastados possam fazê-lo.

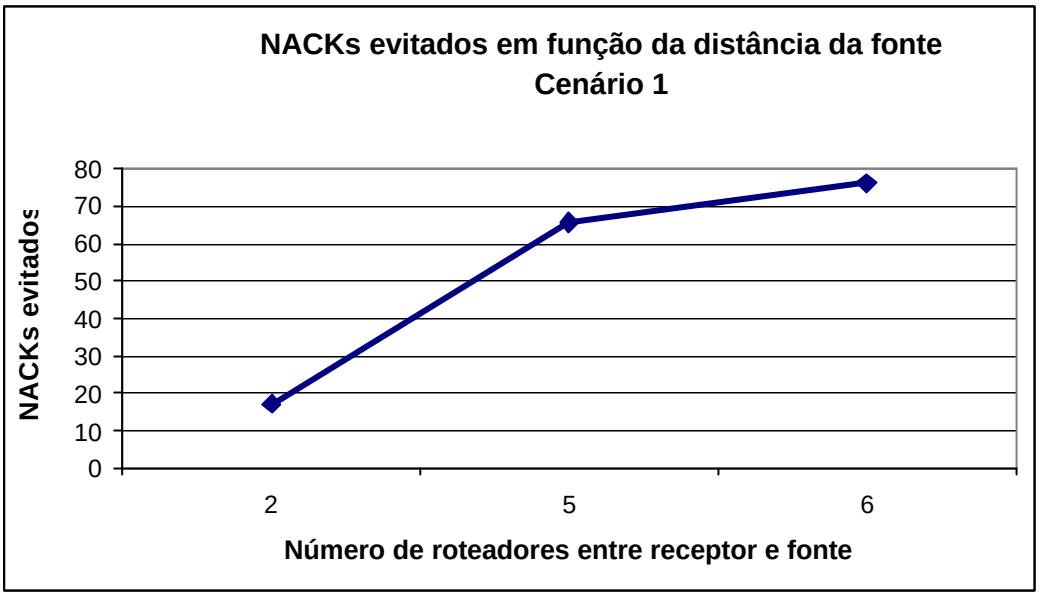

Gráfico 6-5. NACKs evitados em função da distância da fonte (Cenário 1)

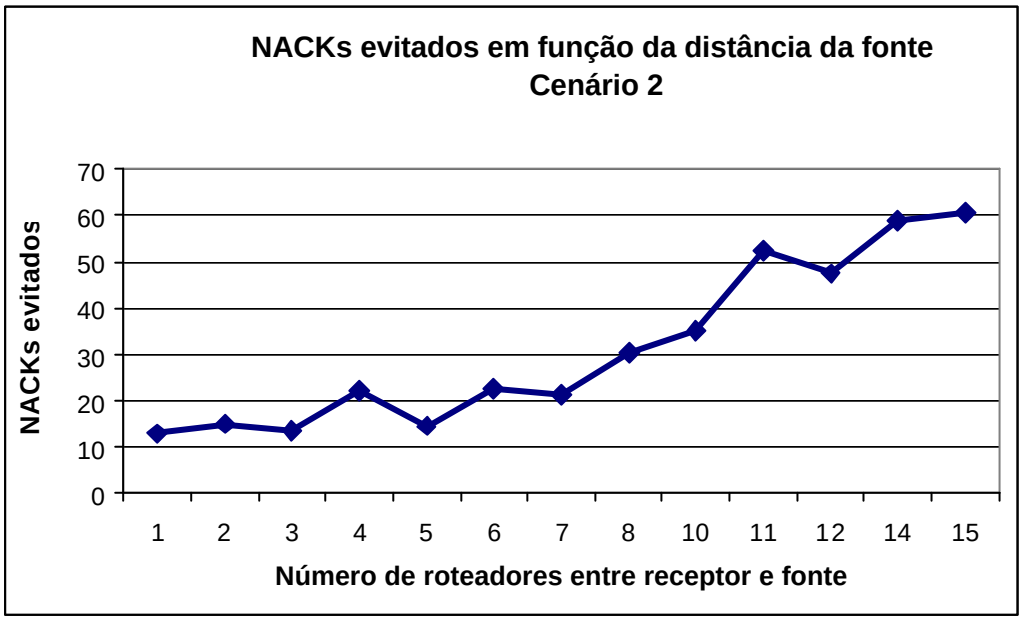

Gráfico 6-6. NACKs evitados em função da distância da fonte (Cenário 2)

A capacidade de recepção no protocolo proposto varia em função da distância da fonte e é ilustrada nos Gráficos 6-7 e 6-8. É possível perceber que o protocolo possui um desempenho bastante razoável comparado aos demais, ficando bastante próximo do desempenho do multicast confiável. Todos os protocolos testados apresentam uma queda no desempenho para os receptores mais afastados, algo mais claramente percebido no cenário 2.

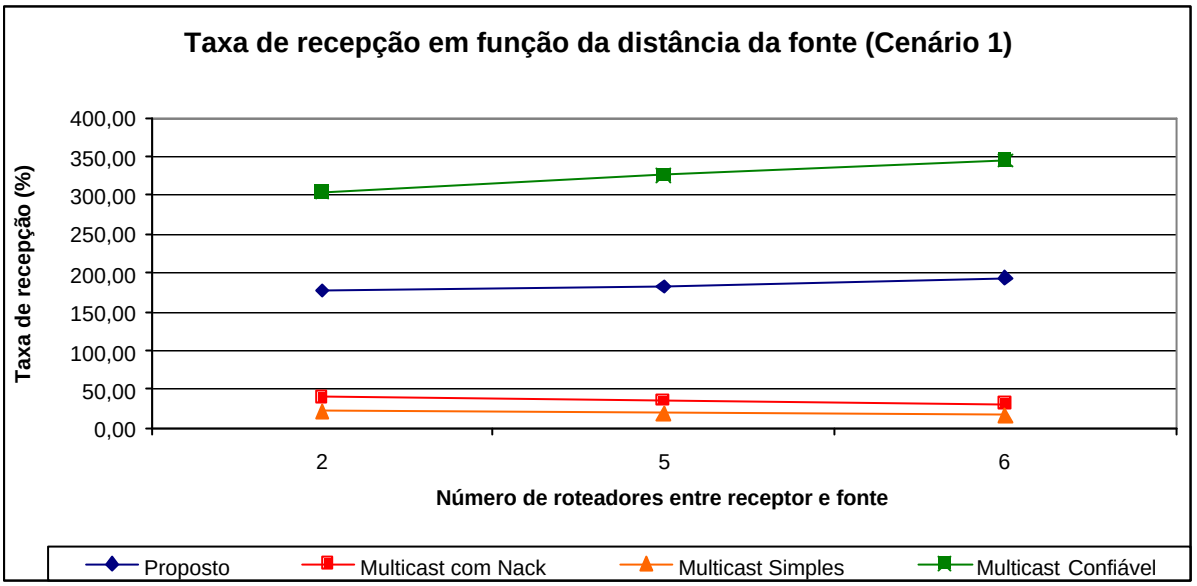

Gráfico 6-7. Capacidade de recepção (Cenário 1)

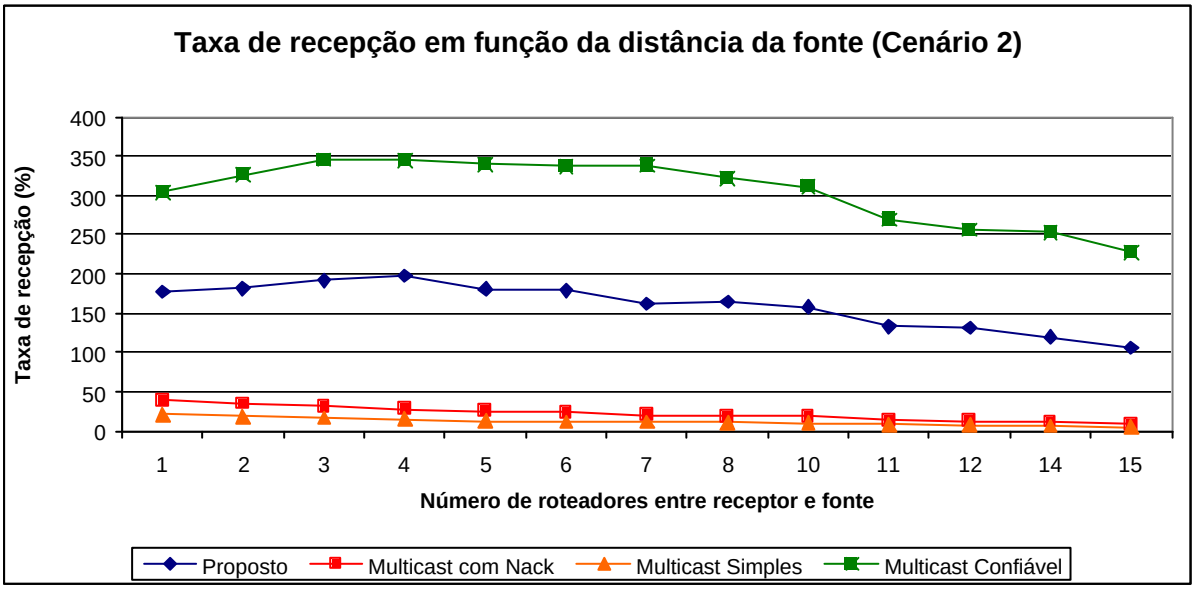

Gráfico 6-8. Capacidade de recepção (Cenário 2)

Um outro parâmetro adotado para avaliação do protocolo foi o Fator de Qualidade de Vídeo (q), apresentado por Leite [LEI04]. Ele consiste em uma métrica baseada na estrutura do GOP para avaliar a qualidade de vídeo. Sua fórmula, representada pelas equações 6.1 e 6.2, leva em conta tantos as perdas diretas, aquelas ocasionadas pela perda do próprio quadro,

quanto as indiretas, aquelas causadas pela falta de um outro quadro (por exemplo, um quadro P ou B que não pode ser reproduzido pela falta de um quadro I).

$$q = \frac{a_I * x_I + a_P * x_P + a_B * x_B}{a_I * N_{TI} + a_P * N_{TP} + a_B * N_{TB}}, 0 \le q \le 1 \quad (6.1)$$

onde,

j: Representa o tipo de quadro (se I, P ou B)

x_i*:* Representa o número de quadros do tipo *j* que são recebidos e mostrados

N_{Tj}*:* Representa o número total de quadros do tipo *j* presentes em um GOP

a_j*:* Representa o coeficiente relativo (quadros do tipo *j* no GOP)

e,

$$a_j = \frac{N_{Ij}}{N_{II} * N_{TI} + N_{IP} * N_{TP} + N_{IB} * N_{TB}} \quad (6.2)$$

onde,

N_{Ij}*:* Representa as perdas diretas e indiretas provocadas pela perda do quadro *j*

É importante lembrar que essa métrica busca avaliar as informações do fluxo de vídeo que são transportadas, mas não avalia diretamente a qualidade do ponto de vista do usuário final.

O fator q médio para o protocolo proposto foi 0,89 no cenário 1 e 0,90 no cenário 2. Para o multicast com retransmissão simples, estes valores foram 0,61 e 0,55, enquanto para o multicast simples os valores foram 0,51 e 0,45 e para o multicast confiável 0,66 e 0,85 nos cenários 1 e 2 respectivamente.

No Gráfico 6-9, relativo ao cenário 1, é possível perceber a performance superior do protocolo proposto em relação aos demais. Essa superioridade fica ainda mais evidenciada no cenário 2, Gráfico 6-10, onde mesmo para os receptores mais distantes, o fator q permanece acima de 0,8. Esta diferença de resultado entre os dois cenários justifica-se pelo número menor de receptores no primeiro, pois isso faz com que as vantagens provenientes da retransmissão a partir dos receptores vizinhos ficam mais discretas.

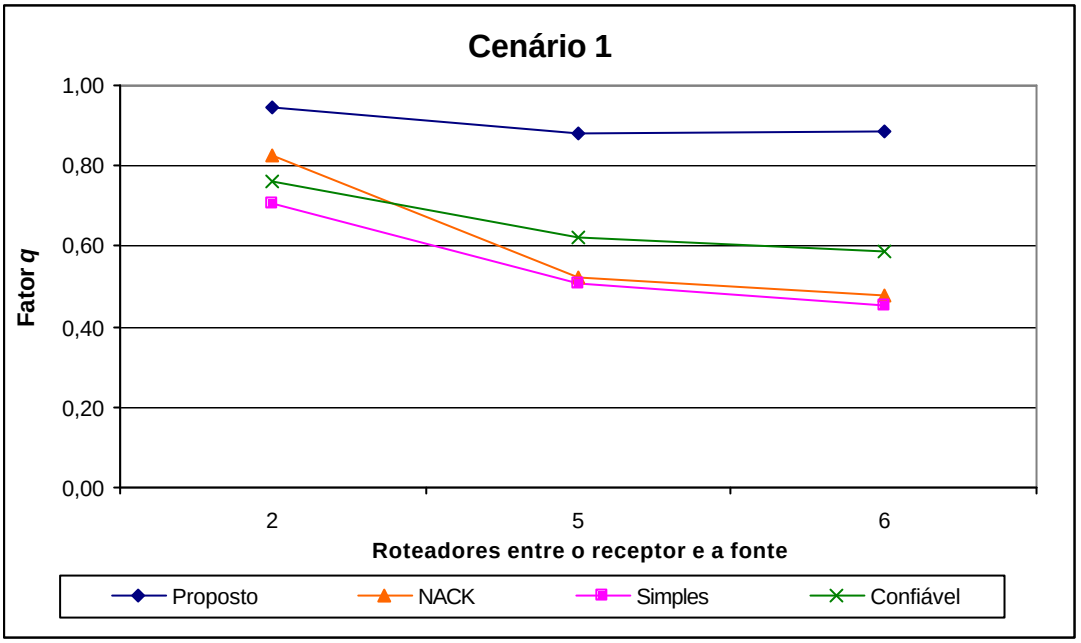

Gráfico 6-9. Fator q em função da distância da fonte (Cenário 1)

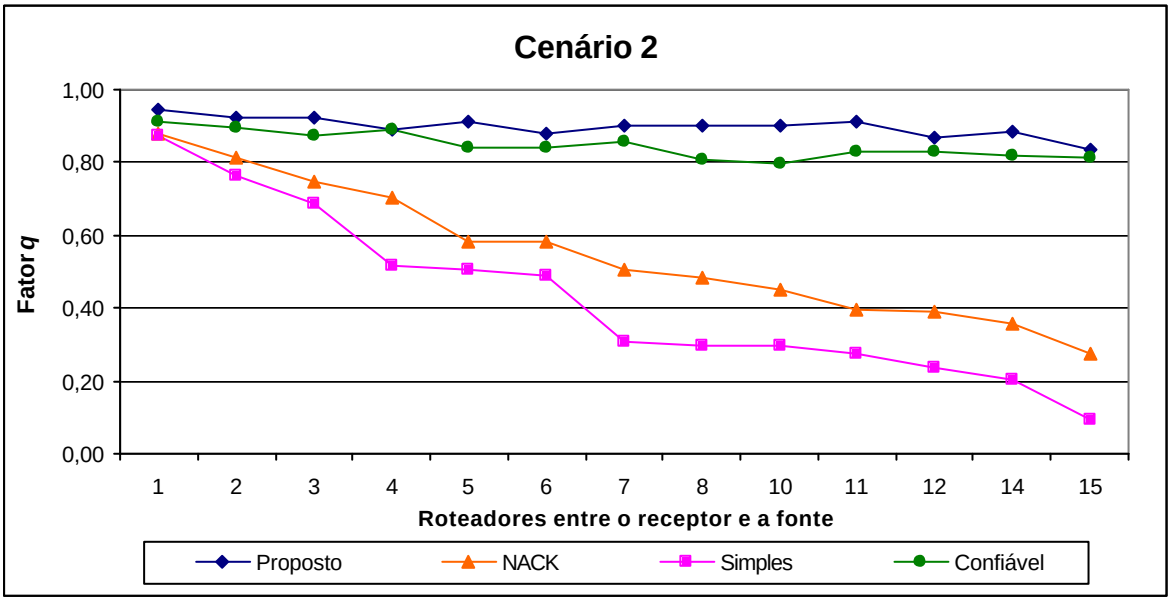

Gráfico 6-10. Fator q em função da distância da fonte (Cenário 2)

A Tabela 6-4 faz uma comparação do desempenho geral dos três protocolos testados.

Tabela 6-4. Comparação do desempenho dos protocolos

	Cenário 1				Cenário 2			
	Protocolo Proposto	**Multicast com retransmissão Simples**	**Multicast Simples**	**Multicast Confiável com retransmissão a partir dos receptores**	**Protocolo Proposto**	**Multicast com retransmissão Simples**	**Multicast Simples**	**Multicast Confiável com retransmissão a partir dos receptores**
Recuperação de pacotes perdidos (%)	12,5	8,5	-	82,5	68	2,8	-	82,35
Tempo médio de recuperação (ms de Simulação)	354	1706	-	473	95	962	-	140
Fator q médio	0,89	0,61	0,55	0,66	0,90	0,55	0,43	0,85

Uma análise geral do protocolo é apresentada para os dois cenários de simulação nos Gráficos 6-11 e 6-12. O que mais chama a atenção é a percepção de que quanto maior a quantidade de receptores conectados à rede, melhor o desempenho do protocolo proposto, mesmo com maior distância entre os mesmos e a fonte. Isso comprova o que foi dito anteriormente: o protocolo proposto é eficiente para redes de larga escala.

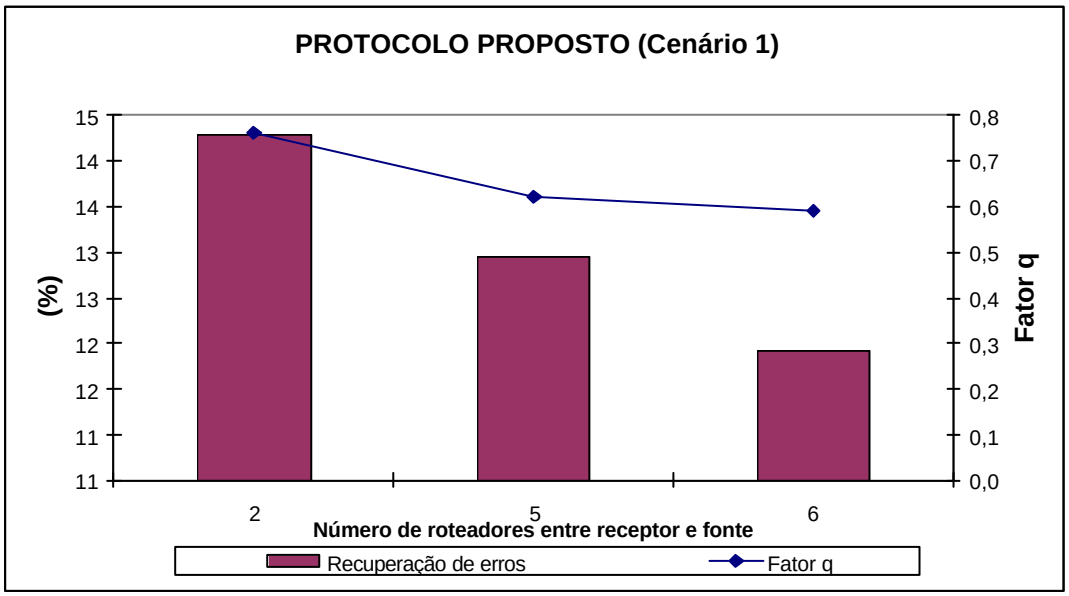

Gráfico 6-11. Análise geral do protocolo proposto (Cenário 1)

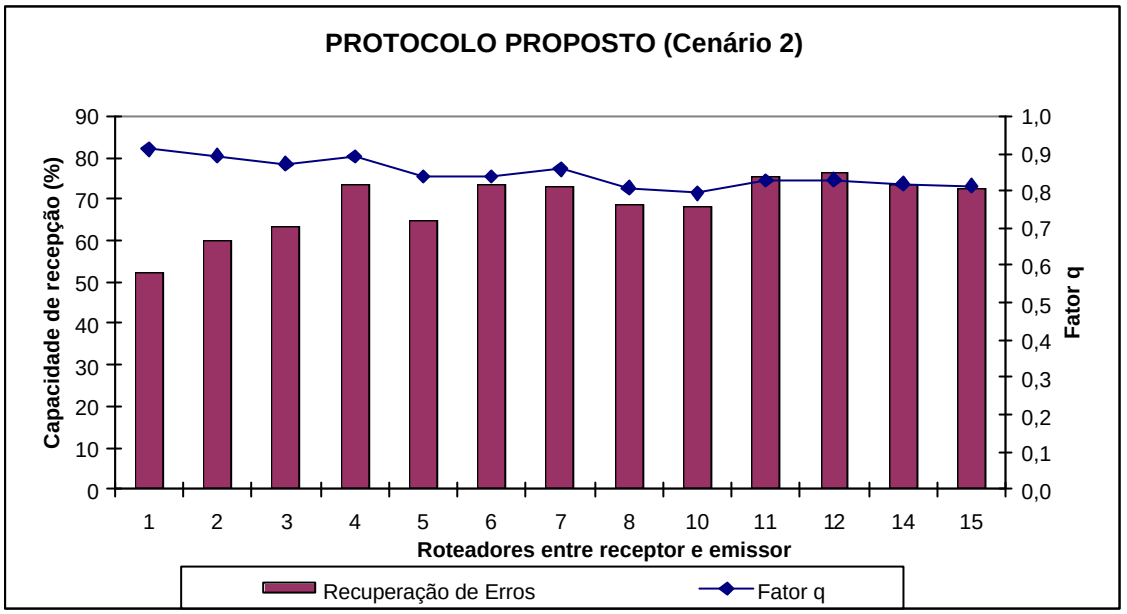

Gráfico 6-12. Análise geral do protocolo proposto (Cenário 2)

6.3. Conclusão

Os resultados das simulações foram bastante satisfatórios. Com os resultados obtidos, com destaque para o tempo médio de recuperação de pacotes e fator q, é possível afirmar que o protocolo proposto pode apresentar-se como uma solução bastante interessante para transmissão de multimídia, sendo mais aplicável em redes de larga escala.

O fator desfavorável ao emprego deste protocolo é o *overhead* relativamente alto que foi apresentado durante as simulações.

De qualquer maneira, em comparação com o multicast simples e o multicast com envio de NACK, percebe-se que o protocolo proposto é consideravelmente mais eficiente na maior parte dos quesitos avaliados, apresentando os melhores resultados para os receptores mais afastados da fonte. Em comparação ao multicast completamente confiável com retransmissão a partir dos receptores, percebe-se que a retransmissão seletiva com base na hierarquia dos quadros estabelecida pelo padrão MPEG faz com que o protocolo proposto ganhe em tempo de recuperação e relevância dos pacotes retransmitidos para a aplicação final.

Capítulo 7

Conclusão

A utilização da transmissão multicast semi-confiável é ideal para aplicações multimídia distribuídas devido aos requisitos particulares deste tipo de aplicação. Em vez da entrega de todos os pacotes, o essencial para estas aplicações é o respeito às restrições temporais de forma que, em caso de falhas ou perdas de pacotes, o resultado percebido pelo usuário final não seja afetado.

Com a adoção do padrão MPEG para a transmissão, é possível fazer uso do descarte seletivo dos pacotes. A hierarquia estabelecida entre os pacotes no processo de codificação deste padrão possibilita o desenvolvimento de algoritmos que, em caso de perdas de pacotes, priorizem a recuperação dos quadros que contenham as informações mais relevantes para recuperação da informação (quadros I, P e B nesta ordem). Em caso de congestionamento ou situações extremas de perda na rede, os pacotes de menor prioridade podem ser ignorados sem que haja grande prejuízo para a aplicação.

Este trabalho apresentou um protocolo que faz uso destes conceitos em associação com as retransmissões a partir dos receptores. Os resultados dos experimentos mostraram-se bastante satisfatórios e indicaram adequação do protocolo proposto para a transmissão de multimídia, em especial, para as redes de larga escala.

Durante o trabalho, o Simmcast [BAR01] mostrou-se uma ferramenta bastante poderosa e útil na implementação e simulação do protocolo.

Entre os possíveis trabalhos futuros, destacam-se:

- Simulações mais completas com variações nos tamanhos dos *buffers* do receptor, tempos de espera antes de pedir ou realizar a retransmissão de pacotes, etc.

- Desenvolvimento de um mecanismo eficiente de controle de fluxo e congestionamento;
- Estudo de novas estratégias de retransmissão (com TTL, hierarquia, etc);
- Analisar a possibilidade de utilizar mecanismos de FEC, talvez enviando dados redundantes junto com quadros que não ocupem todo o espaço de um pacote.
- Implementação;
- Adaptação para a plataforma CORBA, tirando proveito das facilidades oferecidas por ela e criando um novo paradigma para transporte de multimídia dentro desta tecnologia.

Na adaptação para a plataforma CORBA, a sugestão é utilizar como base o UMIOP (*Unreliable Multicast Inter-ORB Protocol*) [GRO01], padrão da OMG (*Object Management Group*), construindo sobre ele um protocolo de difusão semi-confiável. O padrão UMIOP é baseado em grupos abertos e multicast para difusão não-confiável. O conjunto de especificações UMIOP engloba um protocolo de difusão sem garantia de entrega, sem controle de erros e sem suporte a respostas, sobre o qual será adaptado o protocolo proposto.

Estes trabalhos estão sendo desenvolvidos por alunos de iniciação científica e mestrado do DAS (Departamento de Automação e Sistemas) da UFSC e do PPGIA (Programa de Pós-Graduação em Informática Aplicada) da PUC-PR.

Referências Bibliográficas

[ABE00] ABELEM,A., STANTON, M. *Comunicação Multiponto Multimídia: Problemas e Soluções.* Anais da 8ª Semana de Informática da Bahia – SEMINFO 2000.

[AMI97] AMIR, E., McCANNE, S. e KATZ, R *Receiver-driven bandwidth adaptation for light-weight sessions*. ACM Multimedia ’97, Seattle, Novembro de 1997.

[BAR98] BARCELLOS, M. *PRMP: A Scaleable Polling-based Reliable Multicast protocol*. Ph.D. Thesis, Department of Computing Science, Newcastle University, Newcastle upon Tyne, 200p. Outubro de 1998

[BAR01] BARCELLOS, M. et al. *Simmcast: a Simulation Tool for Multicast Protocol Evaluation*. Em: XI Simpósio Brasileiro de Redes de Computadores,2001

[BAR05] BARCELLOS, M. et al. *Simmcast* em http://inf.unisinos.br/~simmcast, 01/02/2005

[BER05] BERKELEY Multimedia Research Center (BMRC), University of California http://bmrc.berkeley.edu/frame/research/mpeg, 25/01/2005

[BES03] BESSANI, A., LUNG, L., FRAGA, J. (2003). *ReMIOP: Projeto e Implementação de um Mecanismo de Difusão Confiável no Corba* Em: XXI Simpósio Brasileiro de Redes de Computadores, 2003, Natal. SBRC'03. Porto Alegre: SBC, 2003. p. 957-964.

[CAS05] CASTRO, F. e CASTRO, M. *Introdução ao Sistema MPEG de Codificação de Vídeo*. Em http://www.ee.pucrs.br/~decastro/pdf/cs5.pdf , 29/01/2005

[CHI98] CHIU, D., HURST, S., KADANSKI, M., WESLEY, J. *Using Mutual Information for Selecting Features in Supervised Neural Net Learning*. IEEE Transactions on Neural Networks, Vol.5, no.4, 1994, p.537-550.

[CHI98] CHIU, D., HURST, S., KADANSKY,M., WESLEY, J. *TRAM: Tree-based Reliable Multicast Protocol*. Sun Microsystems Technical Report TR-98-66, 1998.

[CHI02] CHIU, D. e MULIK, J. *A Reliability Window for Flexible and Scalable Multicast Services*. Proceedings of ICC 2002, 2002.

[COS99] COSTELLO, A., MCCANNE, S. *Search party: using randomcast for reliable multicast with local recovery*. Proceedings of IEEE INFOCOM'99, 1999.

[CAL98] CALDER´ON, M., SEDANO, M., AZCORRA, A., ALONSO, C. *Active Networks Support for Multicast Applications*. IEEE Networks, 1998.

[DEE89] DEERING, S. E. *Host extensions for IP multicasting* (rfc 988). IETF Request For Comments,Julho de 1986.

[FER04] FERGUTZ, L., VENDRAMIN, J., FONSECA, K., *Aplicação de uma Política Inteligente de Descarte de Pacotes (PIDP) em Múltiplos Tráfegos de Vídeo MPEG* em XII Escola Regional de Informática SBC, Agosto de 2004

[FLO95] FLOYD, S., JACOBSON, V., MCCANNE, S., LIU, C.–G. AND ZHANG, L. *A reliable multicast framework for light-weight sessions and applications level framing*. Proceedings of SIGCOMM '95, 1995, pp. 342-356.

[GAO00] GAO, Y., GE, Y., HOU, J. *RMCM: reliable multicasts for core-based multicast trees*. 2000 International Conference on Network Protocols, 2000, pp.83-94.

[GRO01] GROUP, O. M. *Unreliable multicast inter-orb protocol specification*. OMG Standart, 2001.

[GUN96] GUNTHER, R., LEVITIN, L., SHAPIRO, B., WAGNER, P. *Zipf's law and the effect of ranking on probability distributions*, International Journal of Theoretical Physics, 1996, p.395-417.

[HAD94] HADZILACOS, V. e TOUEG, S. *A modular approach to the specification and implementation of fault-tolerant broadcasts.* Technical report, Department of Computer Science, Cornell University, New York – USA, 1994

[HOL95] HOLBROOK, H., SINGHAL, S., CHERITON, D. *Log-Based Receiver-Reliable Multicast for Distributed Interactive Simulation*. Proceedings of ACM SIGCOMM '95 Vol 25,Nº4, 1995, pp328-341.

[KAA94] KAASHOEK, M., TANENBAUM, A. *Efficient Reliable Group Communication for Distributed Systems.* Dept. of Mathematics and Computer Science Technical Report, Vrije Universiteit, Amsterdam.1994

[KAS00] KASERA, S. et Al. *Scalable fair reliable multicast using active services*. IEEE Networks, Special Issue on Multicast, 2000.

[LEI04] LEITE, C. *Uma Abordagem para o Transporte de Vídeo Digital Baseado em Técnicas Proativas de QoS*. Tese submetida à Universidade Federal de Santa Catarina como requisito parcial à obtenção do grau de Doutor em Engenharia Elétrica.

[LEV96] LEVINE, B., LAVO, D., GARCIA-LUNA-ACEVES,J.J. *The Case for Reliable Concurrent Multicasting Using Shared Ack Trees*. Boston, ACM Multimedia Conf., 1996, pp. 365-376.

[LIA98] LIAO, T (1998). “Light-weight reliable multicast protocol”. Internet-Draft, draft-liao-lrmp-00.txt Disponível em http://webcanal.inria.fr/lrmp, Outubro de 1998

[LIN96] LIN, J., PAUL, S. *RMTP: A Reliable Multicast Transport Protocol*. Proceedings of the IEEE Infocom'96, 1996, pp. 1414-1424.

[MAN00] MANE, P. *WAIT: Selective Loss Recovery For Multimedia Multicast*. MS Thesis Computer Science Department, WPI, 2000.

[MAN93] MANOJ, L. QMTP: *A Transport Protocol for Large Scale Dissemination of Multimedia Information*. Tese de Mestrado apresentada a Kentuck University. Mai. 1993

[MAR03] MARTÍNEZ, J, *MPEG-7 Overview (version 9),* International Organization for Standardization, Março de 2003

[MAR99] MARGI, C., BRESSAN, G. e RUGGIERO, W., *Um Mecanismo para Distribuição Segura de Vídeo MPEG*, EM II SECICOM, Novembro de 1999.

[MIL98] MILLER, C. *Reliable Multicast Protocols and Applications*. The Internet Protocol Journal - IPJ Vol 1, Issue 2, 1998

[OBR98] OBRACZKA, K. *Multicast Transport Protocols: A Survey and Taxonomy.* IEEE Communications Magazin, 1998, pp.94-102.

[PAP98] PAPADOPOULOS, C., PARULKAR, G., VARGHESE, G. *An error control scheme for large-scale multicast applications.* IEEE INFOCOM’98, 1998.

[PER03] PEREIRA, Jose et al *Semantically Reliable Multicast: Definition, Implementation, and Performance Evaluation*. IEEE Transactions on Computers, vol.52, nº2, 2003, p150-165.

[PER99] PEREIRA, G., YENIA, H., *MPEG-2: Um estudo do padrão de vídeo* http://www.dcc.ufmg.br/~gpereira/mpeg/, Março 1999, acessado em 29/01/2005.

[PIE00] PIECUCH, M. et al. *A Selective Retransmission Protocol for Multimedia on the Internet. Proceedings of SPIE International Symposium on Multimedia Systems and Applications.* 2000

[SCH96] SCHULZRINNE, H. et al. RTP: A Transport Protocol for Real-Time Applications. IETF RFC 1889, Janeiro de 1996.

[SCH99] SCHNEYER, S., GARCIA, J., BRUNSTROM, A., ASPLUND, K. *PRTP: A Partially Reliable Transport Protocol for Multimedia Applications*, Proceedings Int. Symposium on Intelligent Multimedia and Distance Education (ISIMADE), Baden-Baden, Germany, Agosto de 1999.

[SIN96] SINHA, P. *Distributed Operating Systems: Concepts and Design*. Wiley-IEEE Press, 1996.

[SPE04] SPEAKMAN, T. et Al *PGM Reliable Transport Protocol Specification*. RTFC3208 http://www.javvin.com/protocol/rfc3208.pdf (Jun. 2004)

[STR92] STRAYER, W. e WEAVER, A. *Is XTP Suitable for Distributed Real-Time Systems.* Proceedings of the International Workshop on Advanced Communications and Applications for High Speed Networks, 1992.

[TAN95] TANENBAUM, A. *Sistemas Operacionais Modernos*, Prentice-Hall do Brasil Ltda, 1995.

[TOU95] TOUTAIN, L., AFIFI, H., NARZUL, Jean-Pierre Le. *The Prism Distributed Multimedia Platform*. Proceedings of the IEEE Symposium on Computers and Communications, 1995, p. 384.

[VID05] Video Development Initiative (ViDe), Mary Fran, *The ViDe Videoconferencing Cookbook*, http://www.vide.net, 25/01/2005

[WHE98] WHETTEN, B., BASAVAIAH, M., PAUL, S., MONTGOMERY, T., RASTOGI, N., CONLAN, J., YEH, T. *The RMTP-II Protocol*. Internet-Draft (working draft), 1998

[WEI98] WEI,L., LEHMAN, H., GARLAND, S., TENNENHOUSE, D. *Active reliable multicast*. In IEEE INFOCOM' 98, 1998.

[WHE94] WHETTEN, B.,MONTGOMERY, T. e KAPLAN, S. A High-Performance Totally Ordered Multicast Protocol. Theory and Practice in Distributed Systems, Springer Verlag LCNS 938, Setembro de 1994, pp. 33-57.

[WON98] WONG, T. et. al. *Exploiting Application Level Framing in Reliable Multicast for Periodic Information Dissemination.Poster*. Proceedings of ACM Multimedia '98, Bristol, UK, 1998.

[XU97] XU, X., MYERS,A., ZHANG,H. and YAVATKAR,R. *Resilient Multicast Support for Continuous-Media Applications*, em Intl. Workshop on Network and OS Support for Digital Audio and Video (NOSSDAV97), 1997

[YAV93] YAVATKAR, R., MANOJ, L. *Optimistic Strategies for Large-Scale Dissemination of Multimedia Information*. Proceedings of the First ACM International Conference on Multimedia '93, 1993.

[YAV96] YAVATKAR, R., GRIFFIOEN, J., SUDAN, M. *A Reliable Dissemination Protocol for Interactive Collaborative Applications (TMTP).* Proceedings of ACM Multi-media, 1996

[YOO00] YOON, J., BESTAVROS, A., MATTA, I., *Adaptive Reliable Multicast*. Proceedings of IEEE International Conference on Communications (ICC) 2000, pp1542-1546.