Incorporando Escalabilidade ao Protocolo MCF *

Marcelo Dias Nunes — bill@gta.ufrj.br
Otto Carlos Muniz Bandeira Duarte — otto@gta.ufrj.br

Grupo de Teleinformática e Automação – GTA
COPPE/EE - Programa de Engenharia Elétrica
Universidade Federal do Rio de Janeiro
Caixa Postal 68504 - CEP 21945-970 - Rio de Janeiro - RJ - Brasil

Resumo

Os serviços de IP Multiponto e Mbone na Internet possibilitaram o surgimento das mais diversas aplicações multidestinatárias. Enquanto seu padrão de utilização não for bem conhecido, está claro que somente os protocolos que suportarem grupos dos mais variados tamanhos serão bem sucedidos. Para que isso seja possível, é preciso levar em consideração os problemas de implosão de reconhecimentos e escalabilidade. Portanto, este artigo discute as principais classes de protocolos multidestinatários, dando ênfase à escalabilidade, e propõe extensões e modificações ao protocolo MCF, que é um protocolo multidestinatário e configurável, com gerenciamento de grupo por múltiplas fichas, incorporando mecanismos que o tornem escalável.

1 Introdução

A recente popularidade de aplicações multimídias e/ou de tempo real distribuídas tem acelerado o desenvolvimento de protocolos multidestinatários. Alguns serviços que podem se beneficiar do suporte de um protocolo multidestinatário são: trabalho cooperativo (CSCW - *Computer Supported Cooperative Work*), vídeo-conferência e difusão de informação.

Atualmente, a disseminação dos serviços de IP Multiponto e Mbone tem feito com que a extensão geográfica e o tamanho dos grupos de comunicação aumente consideravelmente. Centenas de participantes podem estar envolvidos em uma única comunicação multidestinatária, além de estarem provavelmente espalhados por todo o planeta. Logo, mecanismos eficientes e escaláveis de gerenciamento de comunicações multidestinatárias, incluindo controle de fluxo e de erros, têm se tornado cada vez mais essenciais. No que diz respeito à confiabilidade, um dos maiores desafios que afligem a comunidade Internet, dado o rápido crescimento dos grupos multidestinatários, é o gerenciamento de reconhecimentos para evitar o que se conhece por **problema da implosão de reconhecimentos**: basicamente, o transmissor deve esperar pelo reconhecimento positivo de **todos** os receptores, que enviam pacotes de controle, incluindo reconhecimentos e informações de controle de tráfego. À medida que o número de receptores aumenta, o transmissor é sobrecarregado com pacotes dos receptores. Essa, visivelmente, não é uma estratégia escalável e acarreta uma série de problemas.

O mecanismo descrito acima é apenas uma das principais categorias de protocolos multidestinatários existente, os **protocolos controlados pelo transmissor**. Além deste, existem ainda os **protocolos controlados pelo receptor**, os **protocolos em árvore** e os **protocolos em anel**.

O protocolo MCF [1] é um protocolo completamente configurável, ou seja, o usuário pode selecionar os mecanismos que mais lhe aprouverem: *checksum*, controle de fluxo por taxa ou janela (ou nenhum controle) e recuperação de erros por *go-back-N* ou retransmissão seletiva (ou nenhum mecanismo de recuperação). Além disso, um mecanismo de fichas gerenciadas por um servidor centralizado oferece suporte a comunicações multiponto-a-multiponto, pois pode-se criar tantas fichas quanto for o número de transmissores desejados. Este artigo propõe, portanto, extensões e modificações ao protocolo MCF no sentido de torná-lo escalável e de minimizar o problema da implosão de reconhecimentos. Para isso, é feito um estudo detalhado deste problema, bem como de algumas soluções conhecidas.

Este artigo está dividido da seguinte maneira: a seção 2 discute as principais classes de protocolos multidestinatários confiáveis, dando ênfase à escalabilidade. A seção 3 descreve de maneira sucinta o protocolo MCF e propõe extensões e modificações. Finalmente, conclusões são apresentadas na seção 4.

2 Escalabilidade em comunicações multidestinatárias confiáveis

Enquanto o padrão de utilização de aplicações multidestinatárias não é bem conhecido, é evidente que grupos dos mais variados tamanhos deverão ser suportados. Um protocolo que não é escalável para grupos grandes não será uma plataforma multidestinatária viável. Visando a melhor solução em termos de escalabilidade, esta seção descreve as quatro principais estratégias de multiponto confiável.

*Este trabalho foi realizado com recursos da UFRJ, FUJB, PROTEM-CC, CNPq, FAPERJ, CAPES e COFECUB.

2.1 Protocolos controlados pelo transmissor

Um protocolo multidestinatário confiável controlado pelo transmissor [2, 3] é aquele que exige que o transmissor receba reconhecimentos (daqui em diante referenciados pelo seu acrônimo ACK, de *acknowledgement*) de **todos** os receptores, antes que lhe seja permitido liberar da memória os dados associados aos ACKs. Dessa maneira, o transmissor precisa conhecer o conjunto de receptores. Essa estratégia sofre claramente do problema de implosão de ACKs. O grande número de pacotes retornados resulta em maior *overhead* de processamento no transmissor, introduzindo atraso na transferência de dados, e provoca excesso na utilização de banda-passante e de *buffers* de memória, acarretando perdas de pacotes.

Embora não seja confiável, o *Xpress Tansfer Protocol* [4] é um exemplo de protocolo controlado pelo transmissor, no sentido em que este decide, através de comandos específicos, quando deseja receber reconhecimentos positivos dos receptores.

2.2 Protocolos controlados pelo receptor

Nesta categoria de protocolos [2, 3], os receptores enviam reconhecimentos negativos (NAKs) ao transmissor quando uma retransmissão é necessária, devido a erros, falhas na seqüência ou estouro de temporizadores. Dessa forma, o transmissor é incapaz de decidir quando pode liberar os dados da memória com segurança, pois recebe informação dos receptores apenas quando pacotes são perdidos e não quando são corretamente entregues. Sendo assim, algum mecanismo de reconhecimento positivo deve ser utilizado para possibilitar a liberação dos dados da memória do transmissor. Tal mecanismo pode ser submetido ao controle do transmissor, como no XTP, ou ao receptor, mediante, por exemplo, o recebimento de uma janela de pacotes.

Se vários receptores detectarem erros de transmissão, pode ocorrer um problema de implosão de NAKs. Surgiram então os chamados **protocolos controlados pelo receptor com supressão de NAKs**: toda vez que um receptor detecta uma perda, ele espera durante um intervalo de tempo aleatório e só depois transmite ao grupo (transmissor e demais receptores) um NAK. Quando um receptor recebe um NAK relativo a um pacote não recebido e para o qual ele tenha disparado um temporizador para enviar seu próprio NAK, o receptor suspende esse temporizador, não envia o NAK e dispara um novo temporizador. O estouro do temporizador sem a recepção do pacote correspondente sinaliza perda do pacote. Com este esquema, espera-se que, teoricamente, apenas **um** NAK seja enviado de volta ao transmissor no caso de uma perda, para todo o conjunto de receptores.

O XTP foi um dos primeiros protocolos a definir algum tipo de controle de implosão. Ele utiliza dois algoritmos chamados *damping* e *slotting*:

- *Damping*: uma vez que os receptores difundem seus ACKs para o grupo, e não especificamente para o transmissor, todos os membros podem observar os ACKs enviados por seus pares. Em uma comunicação livre de erros, espera-se que um ACK enviado por um receptor seja duplicado por todos os outros receptores em resposta a um pedido de reconhecimento do transmissor. Na ocorrência de erros, receptores que detectassem erros similares enviariam ACKs contendo informações similares, havendo sempre alguma porcentagem de duplicatas. O que o algoritmo *damping* faz é impedir que um receptor envie seu ACK se alguma duplicata for detectada na rede. Esta técnica reduz a quantidade total de ACKs transmitidos pelo número de duplicatas canceladas, às custas de forçar todos os receptores a lerem todos os ACKs. Para grupos grandes (mais de dez membros), este mecanismo pode melhorar o desempenho global, principalmente se for utilizado em conjunto com a técnica de *slotting*.
- *Slotting*: o algoritmo *slotting* impõe um atraso aleatório antes da transmissão dos ACKs. Dentre as vantagens trazidas pelo uso desta técnica, pode-se citar duas principais: em primeiro lugar, o envio de ACKs é distribuído no tempo e não mais concentrado, melhorando o desempenho no sentido em que evita uma súbita sobrecarga no tráfego da rede. Em segundo lugar, a distribuição dos ACKs no tempo aumenta a possibilidade do exercício da técnica de *damping*.

Um exemplo de protocolo controlado pelo receptor que utiliza supressão de NAKs é o algoritmo *Scalable Reliable Multicast* (SRM) [5]. O SRM não é exatamente um protocolo, mas uma arquitetura para comunicações multidestinatárias confiáveis, que utiliza princípios de *Application Level Framing* (ALF) [6].

É ainda importante chamar a atenção para o fato de que os mecanismos de supressão de NAKs descritos acima são eficientes somente em redes de difusão, ou seja, redes que oferecem esse serviço inerentemente. Um exemplo seriam redes com topologia em **barramento**, como o Ethernet.

2.3 Protocolos em árvore

Os protocolos baseados em árvore [3] dividem o conjunto de receptores em grupos, chamados grupos locais [7]. Sem perda de generalidade, assume-se que cada grupo consiste em não mais do que N filhos e um líder. Filhos também podem ser líderes de um grupo. Reconhecimentos dos filhos são enviados apenas ao seu líder. Um filho deve enviar seu reconhecimento ao pai tão logo receba um pacote correto, não esperando portanto pelos reconhecimentos de seus próprios filhos (se houver). Esses ACKs são conhecidos como reconhecimentos hierárquicos (HACKs).

Uma árvore N-ária é composta de três tipos de nós: fontes, folhas e *hops*. A **fonte** é a originadora de um novo pacote e tem no máximo N filhos dos quais deve processar HACKs e para os quais deve enviar retransmissões. **Folhas** estão na base da árvore e não têm filhos. *Hops* são líderes de grupos entre a fonte e as folhas. Eles enviam HACKs aos seus próprios líderes de grupo, um nível acima na árvore, e coletam HACKs de seus filhos, retransmitindo quando necessário. Eles não liberam os dados da memória até que todos os seus filhos tenham reconhecido corretamente a transmissão.

O *Reliable Multicast Transport Protocol* (RMTP) [8] é um exemplo de protocolo em árvore. Enquanto o protocolo em árvore genérico envia um HACK para cada pacote ou janela de pacotes, o RMTP envia HACKs periodicamente, para conservar banda-passante.

Quanto ao tipo de rede que melhor se adapta a essa categoria de protocolos, está claro tratar-se de redes em **malha**, como as redes ATM. Sua implementação sobre redes em barramanto é visivelmente impraticável.

2.4 Protocolos em anel

Protocolos *token-ring*, para multiponto confiável, foram desenvolvidos originalmente para oferecer suporte a aplicações que exigem **ordenação** das transmissões em todos os receptores [3]. A idéia básica é ter um receptor, designado *token-site*, responsável pela difusão de ACKs para o grupo. Se não receber um ACK do *token-site* dentro de um certo intervalo de tempo, a fonte transmissora de dados retransmite o pacote. Caso contrário, poderá dealocá-lo de sua memória. O ACK enviado pelo *token-site* serve basicamente a três propósitos:

- atribuir um número de seqüência **global** ao pacote reconhecido, garantindo sua ordenação em todos os receptores;
- possibilitar aos demais receptores a detecção de perdas de pacotes. Os receptores enviam NAKs ao *token-site*, que é responsável pela retransmissão seletiva de pacotes perdidos. Por isso, o *token-site* somente poderá liberar os dados da memória quando a ficha for passada à frente;
- passar a ficha ao próximo membro do anel de receptores, caso este esteja apto a recebê-la. Para isso, o candidato a novo *token-site* deverá ter recebido corretamente todos os pacotes que o anterior recebeu.

O mecanismo descrito acima é o princípio básico de funcionamento do protocolo *Reliable Multicast Protocol* (RMP) [9]. Ele oferece um serviço multidestinatário totalmente distribuído e confiável sobre serviços de datagrama não confiáveis, como o IP Multiponto.

Além da óbvia topologia em anel (redes FDDI, por exemplo), essa categoria de protocolos pode ser implementada sobre redes em malha, configurando um anel "lógico".

3 Extensões ao protocolo MCF

O protocolo MCF (Multidestinatário, Configurável e baseado em Fichas) [1] é um protocolo multiponto-a-multiponto desenvolvido pelo Grupo de Teleinformática e Automação (GTA) da COPPE-UFRJ. A definição deste protocolo passou por uma fase de estudo detalhado de mecanismos utilizados por protocolos de alta velocidade. O protocolo MCF foi o primeiro passo no sentido de criar um serviço de comunicações multisdestinatárias adequado a redes de alta velocidade, para utilização em um ambiente multimídias, que encontra-se atualmente em desenvolvimento pelo GTA, incluindo estudos de controle de congestionamento e sincronização.

O protocolo MCF foi originalmente implementado em plataformas PC e posteriormente portado para um ambiente Sun SPARC sob sistema operacional Solaris 2.x. Ele segue uma arquitetura de implementação especial desenvolvida pelo GTA [10]. Esta arquitetura define módulos especializados e otimizados, promovendo a interface entre o sistema de protocolos e o sistema operacional, bem como interfaces padronizadas entre as diversas camadas do sistema de protocolos:

- A arquitetura de implementação utiliza o mecanismo de troca de mensagens assíncrona, definindo uma interface padrão entre camadas e garantindo flexibilidade, modularidade e fraco acoplamento.
- Uma utilização ordenada da memória torna seu gerenciamento mais simples e rápido. Logo, para evitar fragmentações excessivas, o módulo **gerente de memória** divide a memória em regiões distintas para transmissão e recepção.
- Foi criado um módulo **escalonador de tarefas**, que implementa uma fila FIFO de tarefas (na forma de ponteiros) prontas para execução. Ao invés de um *loop* de execução de todas as tarefas correspondentes às máquinas de estado dos protocolos, o escalonador sempre executa a primeira tarefa da fila de prontas.
- A implementação do módulo **gerente de temporizações** segue uma analogia com um relógio real e é simulada através de um contador em módulo. Este mecanismo é bastante eficiente, pois evita testes desnecessários a cada interrupção de hardware e permite a criação de um grande número de temporizadores durante a execução do programa.

Visando atender à maior variedade possível de aplicações, o protocolo MCF é completamente configurável, oferecendo como opcionais mecanismos que na maioria das vezes são impostos ao usuário. O cálculo de *checksum* para detecção de erros pode ser desabilitado, pois é uma importante fonte de *overhead* de processamento. Os mecanismos de controle de fluxo são opcionais e oferecidos na forma de controle de fluxo por janela e controle da taxa de transmissão. Os mecanismos de recuperação de erros também são opcionais e oferecidos na forma dos tradicionais *go-back-N* e retransmissão seletiva.

O protocolo MCF é um protocolo multiponto-a-multiponto que utiliza fichas para implementar essa característica. Existem três entidades diferentes no MCF: o receptor, o transmissor, que detém uma ficha e também pode receber dados, e o **servidor de fichas**, que concede as fichas e também pode funcionar como transmissor e receptor.

O mecanismo de gerenciamento de fichas original do MCF é simples: qualquer estação que deseje transmitir dados para o grupo, cujo endereço é fornecido pelo servidor de fichas, deve enviar primeiramente ao servidor um **pedido de ficha**. Se tiver alguma ficha disponível, o servidor a concede à estação solicitante e dispara um temporizador, o `TTIMER` (de *token timer*), que representa o tempo máximo com o qual a estação pode permanecer com a ficha. Caso não disponha de nenhuma ficha, o endereço da estação é armazenado em uma fila FIFO interna. Uma vez em poder da ficha, a estação já pode inciar a difusão de dados para o grupo. Quando terminar, a estação devolve a ficha ao servidor, que interrompe o TTIMER associado àquela ficha e a concede automaticamente ao próximo endereço da fila de pedidos, se houver algum. Entretanto, se o temporizador TTIMER estourar e a ficha não tiver sido devolvida, o servidor dispara um novo temporizador, o WTIMER (de *wait timer*), e envia um pedido de devolução **imediata** da ficha. Esse temporizador, de duração bem menor que o TTIMER, é reiniciado três vezes até que a ficha seja retornada. Se não receber a ficha de volta, o servidor a considera perdida e incrementa o contador de fichas. A figura 1 exemplifica o mecanismo descrito, para o caso de duas estações competindo por uma ficha. As primitivas de serviço que aparecem nesta figura, T-TOKEN.Request, Indication e Response, são, respectivamente, primitivas de pedido, informação de recebimento e devolução de uma ficha. As mensagens PASS e TOKEN correspondem, respectivamente, ao pedido de ficha e à ficha propriamente dita.

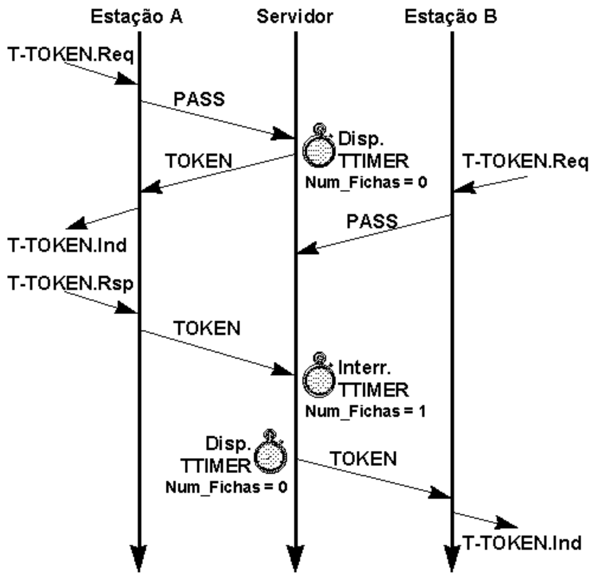

Figura 1: Exemplo de gerenciamento de fichas.

O conceito de servidor único e centralizado de fichas pode ser criticado por não ser dinâmico, pois o servidor é sujeito a falhas, prejudicando assim a escalabilidade do MCF. Entretanto, deve-se ter em mente que o MCF foi projetado inicialmente para redes locais de microcomputadores PC compatíveis. Entretanto, sua migração para estações Sun conectadas à Internet estimulou o questionamento de suas limitações, que o confinavam a redes locais. Sendo assim, as modificações que serão apresentadas a seguir visam reduzir o problema da implosão de ACKs e tornar o protocolo MCF mais flexível e escalável para utilização em redes de longa distância.

Considerando um grupo espalhado fisicamente por várias subredes interconectadas, seria interessante manter um servidor de fichas em cada subrede local. O número total de servidores em um grupo está relacionado ao número total de transmissores simultâneos e à sua distribuição geográfica. As entidades MCF, ao se associarem a um grupo, utilizariam como parâmetro de adesão a "distância", em termos de tempo de ida-e-volta, ao servidor de fichas mais próximo: o novo membro difundiria uma mensagem para o grupo e se associaria ao servidor que respondesse primeiro. Se, após um determinado intervalo de tempo, nenhum servidor respondesse, a nova estação se autoproclamaria servidor de fichas daquela subrede.

Outra importante modificação se refere à realocação do servidor de fichas. Para isso, é preciso fazer algumas extensões ao mecanismo de gerenciamento de fichas original: assim que devolver a ficha ao servidor, a estação deve disparar um temporizador, o CTIMER (de *confirmation timer*). Dentro desse intervalo de tempo, a estação esperará pela **confirmação** de recepção da ficha pelo servidor. Se CTIMER estourar, a estação deve reiniciá-lo e enviar ao servidor um pedido de confirmação imediata da devolução da ficha. Essa operação é repetida ainda por duas vezes, até que a confirmação seja recebida. Caso contrário, a estação considera que aconteceu algum problema com o servidor de fichas e se declara novo servidor, difundindo uma mensagem para todos os membros do grupo e informando seu endereço. Portanto, a **primeira** estação a detectar uma falha no servidor será o novo servidor. Adicionalmente, a mensagem representativa da ficha pode conter o número máximo de fichas, de maneira que, mediante uma alteração, o novo servidor conheça esse número. Qualquer estação, com pedido de ficha pendente, que receber uma mensagem comunicando o endereço de um novo servidor, deve refazer seu pedido à ele. Quaisquer fichas devolvidas ao servidor original se perderão; contudo, isso não constitui em problema, uma vez que o novo servidor é inicializado com o número máximo de fichas.

O serviço multidestinatário oferecido pelo protocolo MCF é não-confiável, no sentido em que o transmissor não conhece o número de participantes do grupo. O algoritmo *bucket*, utilizado pela primeira vez pelo protocolo XTP, é oferecido também opcionalmente pelo MCF como mecanismo de controle de erros em conexões ponto-a-multiponto. Entretanto, esse algoritmo não impede que todos os receptores enviem seus ACKs, gerando uma súbita sobrecarga de tráfego na rede. Para resolver esse problema de implosão de ACKs, duas soluções estão sendo estudadas: configuração em árvore e uso concomitante do algoritmo *bucket* com os algoritmos de *damping* e *slotting*.

Na primeira opção, o servidor de fichas, que como já foi dito seria local a uma subrede, concentraria os ACKs de seus "filhos", limitados em quantidade de acordo com, por exemplo, a capacidade de processamento do servidor de fichas. Assim, o transmissor, esteja ele em qual subrede estiver, receberá tantos ACKs quantos forem os servidores no grupo como um todo.

Há uma tendência em se implementar a segunda opção, pelo menos em um primeiro momento, pois todos os mecanismos descritos nesta seção que utilizam o conceito de grupos locais exigem serviços de roteamento, ainda não analisados. Sua implementação seria complicada, mas uma alternativa é a utilização de protocolos de roteamento já existentes, como o ST-II ou o RSVP [11].

4 Conclusões

Em vista das categorias de protocolo confiável apresentadas na seção 2, este artigo propôs modificações ao protocolo MCF, para adaptá-lo à realidade das aplicações multidestinatárias, tornando-o escalável e reduzindo a implosão de ACKs.

Utilizando a idéia básica do conceito de grupos locais, que é a distribuição geográfica em subgrupos para utilização com redes de longa distância, atribuir-se-á um servidor de fichas para cada subrede (de maneira estática, a princípio, para fins de teste). Isso resolverá principalmente o problema do atraso na aquisição de fichas em um grupo constituído por várias subredes interconectadas.

Quanto ao mecanismo de realocação automática do servidor de fichas, este pode ser prontamente implementado, bastando para isso a criação de uma nova PDU, representando a confirmação de devolução da ficha, e um novo temporizador.

Quanto ao problema da implosão de reconhecimentos, os algoritmos de *slotting* e *damping* mostram-se mais adequados à implementação imediata. Não obstante, está prevista a incorporação de um mecanismo para estruturar o protocolo MCF hierarquicamente de maneira dinâmica em grupos locais, elegendo o servidor de fichas como líder ou "pai" da subrede, ficando responsável pelo reconhecimento de ACKs e retransmissões locais. Estudos comprovam [3] que protocolos hierárquicos mantêm a vazão (teórica) constante, independentemente do número de receptores, mesmo quando a probabilidade de perdas não é desprezível; também permitem que a fonte trabalhe com memória finita, ofereça atrasos fim-a-fim limitados e processe apenas mensagens trocadas nos grupos locais. Entretanto, esse tipo de implementação esbarra em problemas de roteamento, que devem ser pesquisados em maior detalhe. Além disso, nada impede a utilização de supressão de reconhecimentos nos grupos locais.

Referências

[1] M. D. Nunes e O. C. M. B. Duarte, "MCF: Um protocolo multidestinatário configurável com gerenciamento de acesso por fichas de permissão", in *XIV Simpósio Brasileiro de Redes de Computadores*, Fortaleza, pp. 159–177, maio 1996.

[2] S. Pingali, D. Towsley e J. F. Kurose, "A comparison of sender-initiated and receiver-initiated reliable multicast protocols", *Performance Evaluation Review*, vol. 22, pp. 221–230, maio 1994.

[3] B. N. Levine e J. J. Garcia-Luna-Aceves, "A comparison of known classes of reliable multicast protocols", in *Proceedings of International Conference on Network Protocols*, Columbus, outubro 1996.

[4] W. T. Strayer, *Xpress Transfer Protocol Specification, Revision 4.0.* Santa Barbara: XTP Forum, março 1995.

[5] S. Floyd, V. Jacobson, S. McCanne, C. G. Liu e L. Zhang, "A reliable multicast framework for light-weight sessions and application level framing", *ACM Computer Communication Review*, pp. 342–356, agosto 1995.

[6] D. D. Clark e D. L. Tennenhouse, "Architectural considerations for a new generation of protocols", in *ACM SIGCOMM*, Philadelphia, pp. 200–208, setembro 1990.

[7] M. Hofmann, "Adding scalability to transport level multicast", in *3rd COST 237 Workshop - Multimedia Telecommunications and Applications*, Barcelona, novembro 1996.

[8] J. C. Lin e S. Paul, "RMTP: a reliable multicast transport protocol", *IEEE Infocom*, pp. 1414–1425, março 1996.

[9] B. Whetten, S. Kaplan e T. Montgomery, "A high performance totally ordered multicast protocol". Disponível em `ftp://research.ivv.nasa.gov/pub/doc/RMP/RMP_dagstuhl.ps`, agosto 1994.

[10] M. D. Nunes, C. V. de Albuquerque e O. C. M. B. Duarte, "An efficient implementation architecture for layered communication systems", in *38th Midwest Symposium on Circuits and Systems*, Rio de Janeiro, agosto 1995.

[11] S. Paul, K. K. Sabnani e D. M. Kristol, "Multicast transport protocols for high speed networks", in *Proceedings of International Conference on Network Protocols*, 1994.