Avaliação de Desempenho de Protocolos para Multicast com Conhecimento de Grupo baseados em Polling

Marinho Barcellos, André Detsch

PIPCA- Programa de Pós-Graduação em Computação Aplicada
Centro de Ciências Exatas e Tecnológicas
Unisinos - Universidade do Vale do Rio dos Sinos
Av. Unisinos, 950 - São Leopoldo, RS - CEP93022-000
{marinho,detsch}@exatas.unisinos.br

Resumo. *Aplicações que necessitam controlar a composição do grupo de receptores enquanto dados são enviados, com confirmação gradual e positiva do recebimento dos mesmos, demandam o uso de protocolos com semântica ao estilo* TCP*-multicast. O presente trabalho demonstra que protocolos de transporte multicast confiável baseados em polling são adequados a aplicações que exigem o envio de dados de forma confiável, eficiente e relativamente escalável via Internet a membros de um grupo cuja composição a aplicação remetente precisa conhecer e controlar. Estes requisitos não são atendidos por protocolos de transporte multicast orientados a receptor, pois estes sacrificam confiabilidade e eficiência em prol da escalabilidade; em contraste, protocolos baseados em polling sacrificam escalabilidade em prol da confiabilidade. Este trabalho descreve cinco protocolos multicast baseados em polling e os compara através de experimentos de simulação em uma topologia de rede com 451 nós. Os resultados obtidos comprovam a eficácia de protocolos de polling em evitar implosão, seu baixo custo de rede e alto throughput para os grupos analisados, e medem o impacto do tamanho da janela e da banda disponível no desempenho dos protocolos.*

Palavras-chave: multicast confiável, avaliação de desempenho, *polling*.

Abstract. *Applications that require control over the membership of a receiving multicast group while data is sent, with gradual positive acknowledgement of receipt, demand the use of protocols with* TCP*-multicast semantics. This paper shows that reliable multicast transport protocols based on polling are adequate to applications that seek the efficient, reliable and relatively scalable transmission via Internet to members of a group whose membership the sending application needs to be aware of and control. These requirements are not met by receiver-oriented multicast protocols, because these give priority to scalability over reliability; in contrast, polling-based protocols trade-off some scalability for reliability. This work describes five multicast polling-based protocols and compares them through simulation experiments using a network topology with 451 nodes. The results advocate the efficiency of certain polling protocols in preventing implosion, and show that these protocols can achieve low network cost with high throughput for the group sizes considered. Finally, simulation was used to evaluate the impact of window size and available bandwidth in the protocols throughput.*

Keywords: reliable multicast, performance evaluation, *polling*.

1 Introdução

Multicast é um paradigma de comunicação onde uma mensagem é enviada a um grupo de receptores identificados por um endereço único. Como área, multicast tem feito parte da agenda de diversos grupos de pesquisa em redes de computadores e sistemas distribuídos há muitos anos. Em sistemas distribuídos, são investigados protocolos e sistemas de *comunicação em grupo*, particularmente com aplicação em tolerância a falhas, buscando algoritmos e protocolos com garantias de ordenamento e atomicidade na entrega de mensagens, e controle consistente de grupo mediante falhas ([7, 8, 10, 12]).

Na metade dos anos 90, em função da esperada larga disponibilidade de IP multicast, grupos de pesquisa em redes de computadores passaram a investigar protocolos de transporte multicast com mecanismos *escaláveis* de detecção e recuperação de erros ([6, 9, 15]). Tais protocolos são conhecidos na literatura como *scalable reliable multicast protocols,* mesmo embora não ofereçam confirmação positiva de recebimento dos dados (ACK); ao invés disso, tais protocolos objetivam enviar com alguma confiabilidade dados ou multimídia de um remetente para um grupo com número arbitrário de receptores de maneira igualmente eficiente. Como resultado desta pesquisa, foram sugeridos no final dos anos 90 diversos protocolos multicast confiável *baseados em receptor*, com ênfase em escalabilidade em detrimento das garantias de confiabilidade (típicas de protocolos *orientados a remetente*). Em protocolos de transporte multicast confiável baseados em receptor, o remetente desconhece a composição do grupo receptor. Tais protocolos são altamente escaláveis e atendem a várias classes de aplicações na Internet.

Entretanto, há aplicações multicast que necessitam de garantias mais fortes de confiabilidade, em que o remetente está **ciente da composição do grupo**. Um exemplo é o suporte a protocolos de comunicação em grupo que assumem a existência de uma camada subjacente genérica que entrega mensagens ao estilo TCP (serviço TCP-multicast, conforme definido em [3]). Posto que a simples extensão de TCP não é escalável, mesmo para pequenos grupos, outra alternativa deve ser usada. Uma possível solução são os protocolos baseados em *polling* ([1, 13, 14]). Em termos gerais, protocolos de transporte multicast baseados em polling mantêm o controle sobre o andamento da transmissão no remetente, que por sua vez determina o volume do fluxo de *feedback* enviado por receptores de forma a maximizar o *throughput* e minimizar a largura de banda demandada.

Neste artigo, explora-se um conjunto de protocolos de transporte multicast baseados em polling, adequados a aplicações em que um nó necessita transmitir dados de maneira confiável para um grupo cuja composição ele determina e controla. São modeladas versões generalizadas de protocolos, avaliando as mesmas através de simulações utilizando configurações de rede Internet.

O restante do trabalho está organizado da seguinte maneira: a Seção 2 descreve os mecanismos comuns de detecção e recuperação de perdas presentes em protocolos multicast confiável baseados em polling. Baseado nesses mecanismos, cinco protocolos genéricos, representativos da classe, são modelados; suas especificações se encontram na Seção 3. A Seção 4 descreve o *framework* comum em que as simulações foram realizadas, e a Seção 5 o comparativo de desempenho dos protocolos, estabelecendo-se o *trade-off* entre seu throughput efetivo e custo. A Seção 6 encerra o artigo com conclusões e discussão de trabalhos futuros.

2 Controle de Erro para Protocolos Multicast com Polling

O objetivo do protocolo de transporte multicast confiável com conhecimento sobre composição do grupo receptor é transmitir pd pacotes de dados de um remetente para um grupo de N receptores, de forma a entregar a cada receptor uma cópia **idêntica** dos dados enviados, e

obter no remetente uma **confirmação gradual e positiva** dos receptores à medida que os dados são recebidos e passados à aplicação. Para preservar a generalidade do serviço, *pd* não é conhecido no início da transmissão, e pode ser arbitrariamente grande. A quantidade de memória disponível aos protocolos é finita, bem como a banda passante dos links que interligam os nós. O funcionamento dos modelos de protocolo analisados neste artigo foi estabelecido seguindo-se uma série de pressupostos, esclarecidos a seguir.

Em primeiro lugar, assume-se que uma fase de estabelecimento de sessão precede a transmissão dos dados, de forma a se ter o grupo formado: o remetente conhece cada receptor, e cada receptor conhece o remetente. Segundo, assume-se que a velocidade em que dados são gerados pela aplicação remetente é igual ou superior à capacidade do protocolo transferir dados; isto é, sempre haverá dados prontos para serem transmitidos. Semelhantemente, assume-se que os dados são imediatamente consumidos pelos receptores. Note que a remoção destas duas premissas não afeta o funcionamento dos protocolos aqui descritos, mas apenas dificulta a avaliação do seu desempenho. A combinação destes dois pressupostos permite que não se utilize mecanismos de controle de fluxo. Por outro lado, a rede pode fazer com que pacotes sejam perdidos, arbitrariamente atrasados, reordenados, ou duplicados; em todos esses casos, o protocolo é responsável pela detecção e recuperação em situações de inconsistência.

Os protocolos descritos na próxima seção implementam o mesmo estilo de detecção e recuperação de perdas de pacotes, baseado em janela deslizante e temporizadores. O remetente envia pacotes de dados via multicast, e guarda o estado de cada receptor em uma *janela de transmissão*. Cada receptor mantém o estado dos pacotes já recebidos, em uma *janela de recepção*. A janela de recepção avança conforme os pacotes são recebidos em ordem. Uma janela de transmissão avança quando o pacote mais à esquerda teve seu recebimento confirmado (através de um pacote de feedback) pelo receptor correspondente. Além das N janelas, o remetente computa uma *janela de transmissão global*, que consolida informações sobre todos os receptores. A janela de transmissão global está atrelada à janela de transmissão mais atrasada, de forma que seu avanço só ocorre quando o recebimento do pacote mais à esquerda tiver sido confirmado por todos os receptores. O remetente necessita esperar por confirmações positivas de todos os receptores em função da semântica do serviço prestado pelo protocolo à aplicação; a aplicação pode, em teoria, configurar o protocolo de forma a remover do grupo um receptor muito atrasado.

Conforme indicado acima, para atualizar uma janela de transmissão, o remetente necessita obter feedback do receptor correspondente. Para tal, ele transmite um poll (pacote POLL) que requisita uma resposta (pacote RESPONSE) do receptor. POLLs podem ser explícitos em pacotes de controle ou embutidos em pacotes de dados. Em função de um ou mais POLLs, cada receptor vai, mais cedo ou mais tarde, enviar um RESPONSE ao remetente. Ao recebê-lo, o remetente atualiza o estado interno relativo ao receptor em questão (modificando a janela de transmissão de acordo com ACKs e NACKs na resposta, conforme descrito posteriormente).

A atualização do estado interno do remetente vai ser influenciada por dois fatores principais: a **freqüência** na qual as respostas são recebidas de um determinado receptor, e a **quantidade** de informações que cada resposta contém. A freqüência de solicitação de feedback é dada pelo protocolo em questão. Em um extremo, uma resposta pode ser solicitada apenas após todos os pacotes terem sido enviados; no outro extremo, uma resposta será solicitada para cada pacote de dados enviado. A quantidade de informação contida em uma resposta, da mesma forma, pode variar consideravelmente, referenciando de um pacote a uma janela inteira.

A relação acima representa um trade-off: protocolos de polling reduzem a quantidade de pacotes de feedback para evitar implosão, mas em contrapartida, aumentam o montante de informação por pacote, demandando mais processamento. Entretanto, sabe-se empiricamente que em geral receber e processar dois pacotes requer tipicamente mais tempo do que receber

e processar um pacote maior. Implosões podem ser evitadas a partir do momento em que o aumento do processamento por pacote é relativamente pequeno frente à grande redução no volume de pacotes de respostas gerados.

Sistema de janelas e detecção de perda

A seguir, descreve-se em maior detalhe o mecanismo de janela implementado pelos protocolos de polling considerados. O remetente mantém uma janela de transmissão (sw_i) para cada receptor (R_i), que por sua vez mantém o estado dos pacotes de dados recebidos em uma janela de recepção (rw_i). Desta forma, cada janela sw_i representa uma janela rw_i correspondente, sendo a primeira atualizada de acordo com informações recebidas de R_i através de respostas (pacotes RESPONSE). Uma resposta contém uma cópia ("snapshot") de rw_i no momento em que a ela foi enviada. Janelas tem tamanho ws pacotes, e são compostas de:

- $w[1..dp]$: vetor de bits que representa o estado de todos os pacotes entre 1 e dp (número total de pacotes) – na prática, possui ws entradas;
- le: número de seqüência do pacote mais à esquerda da janela, ou seja, o primeiro bit 0;
- re: número de seqüência do pacote mais à direita da janela, dado por $le + ws - 1$;
- hr: maior número de seqüência de pacote registrado em w até o momento.

Quando um RESPONSE é recebido pelo remetente, sw_i é atualizada em função da cópia de rw_i embutida. Este processo permite ao remetente detectar NACKS em sw_i. Uma entrada $w[seq]$ é interpretada como ACK, NACK ou estado indefinido de acordo com as seguintes regras:

- ACK seq: $seq < le \vee w[seq]$
- NACK seq: $seq \leq hr \wedge \sim w[seq]$
- estado indefinido: $seq > hr$

Além da detecção da perda de pacotes de dados descrita acima, é necessário que o remetente trate perdas de pacotes RESPONSE e POLL. O remetente mantém uma tabela de *round trip times* (RTTs) atualizada dinamicamente, de onde pode ser obtida uma estimativa de *timeout de retransmissão* para cada receptor (denominada rto_i, de *retransmission timeout*). A forma de fazer estas estimativas foi baseada no TCP. O valor rto_{max} é definido como o maior timeout de retransmissão no conjunto de rto_i's. No restante, a forma de implementação do controle de erro para POLLs e RESPONSEs varia conforme o protocolo analisado. As descrições a seguir foram desenvolvidas com base nos conceitos acima descritos.

3 Modelos de Protocolos baseados em Polling

Nesta seção, são apresentados cinco modelos gerais de protocolos baseados em polling: PET, PrP, PSEW, RBP e PeP.

3.1 PET - *Poll Every Time*

Neste protocolo, cada pacote de dados contém (implicitamente) um POLL que solicita um RESPONSE de cada receptor. Ele se assemelha ao protocolo *Full Feedback* (vide [2]).

A cada pacote transmitido, o remetente programa um temporizador para aguardar por respostas. O temporizador é configurado com base no mais longo timeout de retransmissão, rto_{max},

de forma a permitir o recebimento de respostas de todos os receptores. Cada receptor envia um RESPONSE contendo uma cópia da própria janela. Para cada RESPONSE recebido, o remetente atualiza a respectiva janela de transmissão, conforme descrito na seção anterior. Quando um pacote se torna *fully acked*, o remetente cancela o temporizador associado a esse pacote. Caso o temporizador expire, o remetente repete o processo, retransmitindo e reprogramando o temporizador. Note que um RESPONSE carrega *confirmações cumulativas* de recebimento, de forma que a perda de um desses pacotes pode ser compensada por respostas posteriores oriundas do mesmo receptor.

Caso não ocorra nenhuma perda, N respostas são geradas para cada pacote. Estima-se que mesmo para pequenos grupos, esta política acarrete um grande número de perdas de pacotes devido à implosão. O PET usa a estratégia da *saturação* para atualizar o estado sobre os receptores no remetente: existe um alto grau de redundância nas respostas (pois cada resposta traz informações sobre uma janela inteira) e mesmo com alto grau de perdas, algumas respostas acabam chegando ao remetente, garantindo o progresso da transmissão.

3.2 PrP - *Probabilistic Polling*

Neste protocolo, os receptores respondem aos pacotes de dados de acordo com uma probabilidade (p), contida no pacote. Isto é, todo pacote é um POLL em potencial. O valor de p é um parâmetro do protocolo, com $0 < p \leq 1$; a quantidade de respostas enviadas por receptores é diretamente proporcional a p. Dessa forma, o valor de p deve ser tal que, considerando o tamanho do grupo, a quantidade de pacotes RESPONSE enviados não exceda a capacidade do remetente nem dos links próximos.

Na operação do protocolo, o remetente transmite pacotes (com POLLs embutidos) de forma contínua até que terminem os dados, ou a janela tranque. Neste último caso, o remetente espera até que a janela seja destrancada por eventuais respostas sendo recebidas. Para evitar *deadlock*, a espera pelo destrancamento da janela é limitada em rto_{max}. Caso a janela destranque, a transmissão de **novos** pacotes prossegue; senão, quando o tempo se esgota, o remetente retransmite cada um dos pacotes ainda não confirmados por todos os receptores e volta a esperar por um novo período de até no máximo rto_{max}. Este processo se repete até que a janela seja destrancada e novos pacotes possam ser enviados.

Como vantagem, este esquema probabilístico permite solicitar um número aproximado de respostas, sem precisar que o remetente indique/controle quem são os receptores a responder. Por outro lado, uma vez que o polling é feito de forma não determinística, o remetente não tem como saber de quais nem de quantos receptores um RESPONSE deve ser esperado. Uma das implicações disso é que não é possível usar temporizadores associados a POLLs. Outra é que não é possível solicitar um RESPONSE de um ou outro receptor específico, obrigando o remetente a repetir pollings probabilísticos até que as respostas desejadas sejam obtidas; este problema é ainda maior quando há poucos pacotes de dados na janela que ainda não foram confirmados por todos receptores, posto que assim um pequeno número de POLLs probabilísticos será enviado a cada rto_{max}.

3.3 PSEW - *Poll Subgroups after Each Window*

Neste protocolo, a operação é dividida em duas etapas: transmissão de uma janela e obtenção de respostas. Cada RESPONSE se refere à janela por completo, reduzindo substancialmente o número de pacotes de feedback e o custo de rede (largura de banda). Além disso, receptores são divididos em **subgrupos**, o que permite que um POLL seja eficientemente enviado com multicast diretamente a um determinado subgrupo (cada subgrupo corresponde a um grupo IP multicast diferente). Abaixo o protocolo é descrito em maior detalhe.

Conforme anteriormente indicado, o remetente trabalha em duas fases: de *dados* e de *polling*. Durante a fase de *dados*, são (re-)transmitidos todos os pacotes de dados entre *le* e *re*[1] cuja recepção não tenha sido confirmada por um ou mais receptores. Após (re-)transmitir o pacote *re*, o remetente passa para a fase de polling.

Esta fase tem como objetivo assegurar que, a partir do seu início, pelo menos uma resposta de cada receptor seja obtida. Para isso, o remetente irá enviar um POLL para cada subgrupo usando multicast. Ele então aguarda até que chegue uma resposta de cada receptor de cada subgrupo. Para prevenir *deadlock*, o remetente limita esta espera: rto_{max} após o envio do último POLL, o processo é repetido para todos os subgrupos contendo pelo menos um receptor cujo RESPONSE não tenha sido recebido. A espera pode ser interrompida quando o último RESPONSE necessário chegar, fazendo com que o remetente volte à fase de dados.

Para evitar implosões de feedback, as transmissões de pacotes de POLL são sempre espaçadas. Tanto o espaçamento entre os envios de POLL quanto o tamanho dos subgrupos são parâmetros do protocolo. Quanto maior o subgrupo, maior será a concentração temporal na chegada de RESPONSEs; em compensação, menos POLLs serão necessários para gerar as respostas desejadas, e mais eficiente será o processo de recuperação de POLLs ou RESPONSEs perdidos. Quanto maior o espaçamento entre os POLLs, menor a probabilidade de perdas por implosão, causadas pela chegada simultânea de RESPONSEs de subgrupos diferentes; por outro lado, um espaçamento muito longo poderá refletir no desempenho do protocolo.

3.4 RBP - *Response-Bucket Polling*

A idéia básica deste protocolo é limitar a quantidade de respostas em trânsito através de um mecanismo de *balde de fichas (token bucket)*, que garante que em nenhum momento haverá mais respostas chegando do que o limite estabelecido pelo balde.

O protocolo possui duas partes independentes: uma que envia pacotes de dados de acordo com o que é permitido pela janela (ou seja, $seq \leq re$), e outra que envia POLLs quando necessário e sujeito à disponibilidade de fichas.

O remetente mantém um balde de fichas, onde cada ficha representa uma permissão para solicitar uma resposta a um receptor. Quando um POLL é enviado para algum receptor (sempre via unicast), uma ficha é retirada do balde, e quando uma resposta é recebida, uma ficha é devolvida ao balde. Caso o balde se encontre vazio, nenhuma nova resposta pode ser solicitada (novos POLLs não podem ser enviados). Ao enviar um POLL, o remetente (re-)programa um temporizador para limitar a espera de uma resposta gerada por *esse* POLL (apenas um temporizador por receptor).

O remetente controla a necessidade de enviar POLLs a receptores através de um vetor V, cuja i-ésima entrada indica (caso marcada) se um POLL deve ser enviado ao receptor R_i. Uma entrada $V[i]$ é marcada sempre que for (re-)transmitido um pacote de dados a R_i, ou expirar o temporizador relativo a R_i. Para enviar POLLs, o remetente percorre V de maneira circular procurando por uma entrada marcada. Caso encontrada (p.ex., i), o remetente envia um POLL ao receptor R_i, desmarca $V[i]$ e retira uma ficha do balde. Este processo é repetido até que o balde esteja vazio, ou que nenhum receptor precise receber um POLL.

Quando o temporizador expira, uma ficha é devolvida ao balde, e a posição em V referente ao receptor cujo temporizador expirou é marcada, de forma que seja, posteriormente, enviado um POLL para o receptor. Note que devido à existência de apenas um temporizador por receptor, o mecanismo de controle de respostas pendentes não é preciso. Para remover essa imprecisão, seria necessário manter um temporizador por resposta pendente, o que seria demasiado custoso em termos de processamento e estado no remetente, além de tornar o protocolo mais complexo.

[1] relembrando, pacote mais à esquerda e mais à direita da janela de transmissão global, respectivamente.

A perda de pacotes é detectada através de NACKs identificados na janela recebida em respostas. Perdas são recuperadas através de retransmissões com multicast. Uma vez encontrado um NACK com seqüência *seq*, o processo de retransmissão é iniciado de maneira a enviar o pacote de dados *seq*. Esse processo consiste na criação de um temporizador associado à *seq* e programado para expirar em rto_{max}; o que ocorre apenas caso ainda não exista em andamento um temporizador associado à *seq*. Tal temporizador é necessário para suprimir retransmissões desnecessárias; ele permite a chegada de mais respostas referentes à *seq*.

O tamanho do balde de fichas determina o número máximo de respostas em trânsito. Esse valor é um parâmetro de entrada do protocolo, e deve ser configurado em função da capacidade do remetente e dos links próximos, tal como nos protocolos PrP e PSEW. Um valor excessivamente alto levará a implosões; por outro lado, um número muito baixo poderá levar ao "sufocamento" do remetente: trancamento da janela de transmissão devido à ausência de feedback.

3.5 PeP - *Periodic Polling*

Este protocolo representa uma derivação do protocolo RBP, conforme seção anterior, com as semelhanças descritas a seguir. Primeiro, tal como o RBP, o funcionamento é dividido em duas partes, transmissão de dados e transmissão de POLLs; segundo, o remetente mantém um vetor V para controlar a necessidade de envio de novos POLLs; terceiro, o processo de retransmissão de dados é idêntico.

A diferença básica entre os protocolos RBP e PeP é que o limitante para a transmissão de POLLs deixa de ser a quantidade de respostas em trânsito e passa a ser o intervalo de transmissão entre POLLs. O envio de POLLs, como indicado pelo nome, está baseado em um esquema periódico: a cada tempo T, o remetente verifica em V se é necessário enviar um POLL a algum receptor. Caso positivo, um POLL é enviado da mesma forma que em RBP. O tempo T é um parâmetro do protocolo que é configurado de forma a resultar em uma taxa de respostas adequada.

4 Modelo de Simulação

Os modelos de protocolos descritos na seção anterior foram estudados através de simulação. Os experimentos foram realizados com dois objetivos: primeiro, analisar o comportamento dos protocolos de polling em função dos seus parâmetros de configuração; segundo, comparar o desempenho dos protocolos em função da sua escalabilidade, valendo-se de métricas como throughput e largura de banda. Esta seção apresenta uma visão geral do modelo de rede empregado nas simulações, assim como as configurações específicas utilizadas nos experimentos.

4.1 Simulador

As simulações foram conduzidas no simulador de redes VINT ns ([5]), um dos simuladores de rede mais utilizados em avaliação de protocolos. O ns é um simulador de rede de nível de pacote implementado através de uma abordagem de eventos discretos. Experimentos utilizando este simulador podem representar desde a camada de enlace até a camada de aplicação. O ns foi usado inicialmente em pesquisas de desempenho do TCP, mas desde então teve seu uso expandido a uma série de áreas, como pesquisas de dinâmica quanto à interação de múltiplos protocolos, políticas de enfileiramento em roteadores, protocolos multimídia, protocolos de multicast confiável e controle de congestionamento. O ns tem ainda a vantagem de já possuir suporte a uma grande quantidade de tecnologias e protocolos reais.

Figura 1: Topologia de rede empregada nas simulações.

4.2 Topologia

Não existe algo como uma topologia "representativa" da Internet; tal problema é reconhecido na comunidade e tem sido alvo de pesquisas nos últimos anos. O ns não oferece suporte à geração automática de topologias; entretanto, existem ferramentas geradoras de topologias cujas saídas podem ser convertidas para o formato esperado pelo mesmo. Neste artigo, foi usada a ferramenta gt-itm ([11]), a partir do qual foi gerada uma topologia usando o modelo *Transit-Stub*. O modelo *Transit-Stub* produz grafos hierárquicos compondo domínios *transit* e *stub* interconectados.

Todos os experimentos foram conduzidos sobre a mesma topologia de rede, com 450 nós, irrespectivo do tamanho do grupo (que variou entre 1 e 400 receptores). O diâmetro dessa topologia de rede é 16 *hops*. O limite inferior para latência máxima encontrada nessa rede, com filas vazias (*unloaded network*), é de 622ms. Na prática, a latência total nos experimentos é maior, em função da soma dos tempos de recepção e transmissão decorrentes do encaminhamento em cada nó intermediário, bem como dos tempos de enfileiramento.

A essa rede, foi acrescentado um nó contendo o remetente, e o mesmo ligado ao resto da topologia através de um link de 1ms com um nó de um *transit domain*. Considerando o nó remetente como raíz, a árvore multicast de caminho mais curto resultante utilizada nos experimentos possui altura 11. O limite inferior para latência máxima entre raiz e um nó folha qualquer nesta árvore é de 411ms. A inclusão deste nó extra, para o remetente, tem por objetivo garantir a existência de apenas **um** link de chegada no nó remetente, permitindo limitar a capacidade de recebimento de feedback por parte do remetente. A Figura 1 ilustra a topologia utilizada; note que o remetente (nó 0) está destacado, na parte inferior esquerda da figura.

Sem perda de generalidade, no nó remetente a fila de saída do link foi configurada com um valor artificialmente alto, suficiente para que nunca ocorresse um descarte de pacote nesta fila. Tal medida foi adotada para que o remetente pudesse ordenar o envio de mais de 50 pacotes[2] sem pausa. Isto não representa uma limitação do modelo, posto que o envio espaçado de pacotes (de acordo com a banda do link de saída) poderia ser facilmente incorporado aos protocolos; por simplicidade, esse gerenciamento foi passado ao ns. Note que, no mesmo link, a fila na direção oposta não teve sua capacidade alterada, possibilitando implosões de feedback.

Diversas topologias lógicas com grupos de receptores foram sobrepostas à topologia com 450 nós, mantendo-se sempre a localização do remetente e do primeiro receptor do grupo. Para diminuir o impacto natural da latência nos experimentos, e permitir uma avaliação justa entre grupos de diferentes tamanhos, este primeiro receptor foi posicionado propositadamente no nó com maior latência mínima até o remetente (considerando uma rede sem carga). Além disso, a localização dos nós receptores restantes foi aleatória e observou as seguintes regras: (a) apenas um receptor por nó; (b) escolha *incremental* dos nós, de forma que, por exemplo, o grupo com 200 nós conterá todos os nós pertencentes ao grupo com 100.

Na topologia gerada, todos os links foram configurados com igual taxa de perda induzida. Em um conjunto de experimentos, a perda em cada link foi ajustada para 0,1%, e no outro, para 0% (ou seja, sem perda). Note que a chance de perda é acumulada através dos diversos links: a probabilidade de um pacote do remetente chegar ao receptor mais distante (11 *hops*) é de $(1 - 0,001)^{11}$, levando a uma probabilidade de perda $\cong 1\%$. Além das perdas de dados em função dessa taxa de erro, pacotes de feedback podem ser descartados devido à implosão. Independente de taxas de perdas associadas aos links, pacotes podem ser reordenados devido ao esquema de roteamento implementado pelo ns.

Por fim, considerando que o objetivo dos experimentos neste estudo é comparar cada protocolo isoladamente, ao invés de investigar sua interação com outros protocolos, optou-se por não empregar fontes geradoras de tráfego de fundo (como FTP e *cbr*).

4.3 Configuração e Métricas dos Protocolos

Nos experimentos, os protocolos foram avaliados com a mesma quantidade de dados a transmitir, $dp = 1000$ pacotes, e os mesmos tamanhos de pacotes: 1000 bytes para pacotes de dados (com ou sem POLL embutido), 50 bytes para pacotes de POLL, e $50 + \lceil ws/8 \rceil$ bytes para pacotes de RESPONSE (ws é o tamanho da janela, em pacotes, conforme definido na Seção 2). No início da sessão, não existe uma estimativa de RTT entre o remetente e cada receptor; o RTT foi inicializado com 1s, valor esse superior aos RTTs medidos durante a transmissão.

Para avaliação dos protocolos, as seguintes métricas foram escolhidas: *throughput efetivo*, custo de rede e quantidade de implosões. O throughput efetivo é determinado dividindo-se a quantidade de dados transferidos de maneira confiável pelo período de transmissão. Define-se *período de transmissão* como o intervalo compreendido entre o envio do primeiro pacote de dados e o recebimento da última confirmação positiva necessária (quando todos os pacotes tiverem sido confirmados por todos os receptores). Desconta-se aí o período inicial, para estabelecimento de sessão, e o período final, para fechamento da mesma, sobrecarga presente em todos os protocolos confiáveis com controle de grupo.

Para efeito de avaliação, o *número de implosões* foi definido como a diferença entre o número de pacotes RESPONSE enviados por receptores e aqueles recebidos pelo remetente. Este cálculo foi possível apenas em configurações onde os links foram atribuídos taxa de erro zero. Note que, em teoria, perdas por implosão podem ocorrer em qualquer ponto da rede, não apenas no remetente.

[2] valor *default* para filas de pacotes no ns.

A terceira métrica escolhida foi a largura de banda utilizada pelos protocolos. Para contabilizar o somatório de todos os bits transmitidos através de todos os links, seria necessário gravar arquivos de traço (*trace*) registrando cada troca de pacotes feita pelo `ns`. Por uma questão de viabilidade técnica, optou-se por simplificar a contabilização de largura de banda; define-se *custo de rede* como o somatório de todos os bits transmitidos pelo remetente e pelos receptores.

5 Simulação

5.1 Parâmetros de Configuração dos Protocolos

Conforme descrito na Seção 3, os modelos de protocolos possuem parâmetros de entrada que influenciam seu comportamento. Em uma série de experimentos preliminares, avaliou-se o impacto da variação dos parâmetros de entrada para cada protocolo, de forma a encontrar-se valores que produzissem os melhores resultados em termos de throughput e custo de rede.

Seja t_{send} o tempo necessário para enviar um pacote de dados através do link de saída, t_{recv} o tempo necessário para receber uma resposta através do link de chegada, e N o número de receptores, os seguintes parâmetros são especificados em cada protocolo:

- PET: não há parâmetros de entrada específicos;
- PrP: a probabilidade de resposta, p, é dada por $p = 0,5 \times \frac{(t_{send}/t_{recv})}{N}$, o que significa que, dado um fluxo contínuo de dados *downstream,* receptores geram um fluxo *upstream* de respostas 50% da capacidade do remetente (na prática, o envio de dados pode não ser contínuo, diminuindo a taxa de respostas verificada);
- PSEW: subgrupos com 50 receptores e intervalo entre o envio de dois POLLs subseqüentes dado por $t_{recv} \times 50$, que representa o tempo necessário para receber através do link de chegada as respostas de todos os membros de um subgrupo;
- RBP: balde com 100 fichas, que permite a existência de no máximo 100 respostas "pendentes" em um dado momento;
- PeP: intervalo entre dois POLLs subseqüentes dado por $t_{recv} \times 10$, ou seja, o remetente solicita o envio de respostas em uma taxa igual a 10% de sua capacidade máxima de receber respostas.

Além dos parâmetros individuais acima listados, protocolos compartilham um argumento, tamanho da janela (ws), que influencia decisivamente o desempenho de protocolos baseados em polling. Estes protocolos evitam implosão reduzindo a quantidade de feedback, fazendo com que a janela de transmissão seja atualizada menos freqüentemente, o que pode levar ao trancamento da mesma.

No primeiro experimento, avalia-se o impacto do **tamanho da janela** no throughput dos protocolos de transporte multicast baseados em polling, para configurações com 400 receptores. Na Figura 2, o eixo x representa o tamanho da janela, em pacotes, e o eixo y, o throughput em Kbits/s. Foram avaliados como potenciais tamanhos de janela valores entre 50 e 500 pacotes (50 e 500 Kbytes), para cada um dos protocolos exceto PET (o número de implosões causadas por este protocolo em grupos com 400 receptores é tão alto que inviabilizou a execução de simulações). Primeiramente, como esperado, a Figura 2 mostra que o throughput dos protocolos melhora à medida que o tamanho da janela aumenta. Três protocolos, PSEW, RBP e PeP, obtiveram throughput entre 650 e 720 Kbits/s, mais de 65% do throughput máximo atingível,

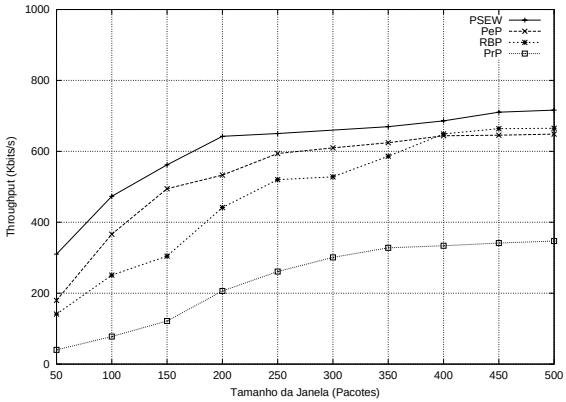


Figura 2: Impacto do tamanho da janela no throughput dos protocolos.

quando a janela aproxima 500 pacotes. Estes mesmos três protocolos não apresentaram throughput inferior a 160 Kbits/s, em qualquer um dos tamanhos de janela avaliados. O melhor e pior throughput foi observado nos protocolos PSEW e PrP, respectivamente, para todos os tamanhos de janela. Em função destes resultados, optou-se por fixar em 200 o tamanho de janela para os demais experimentos.

5.2 Implosão

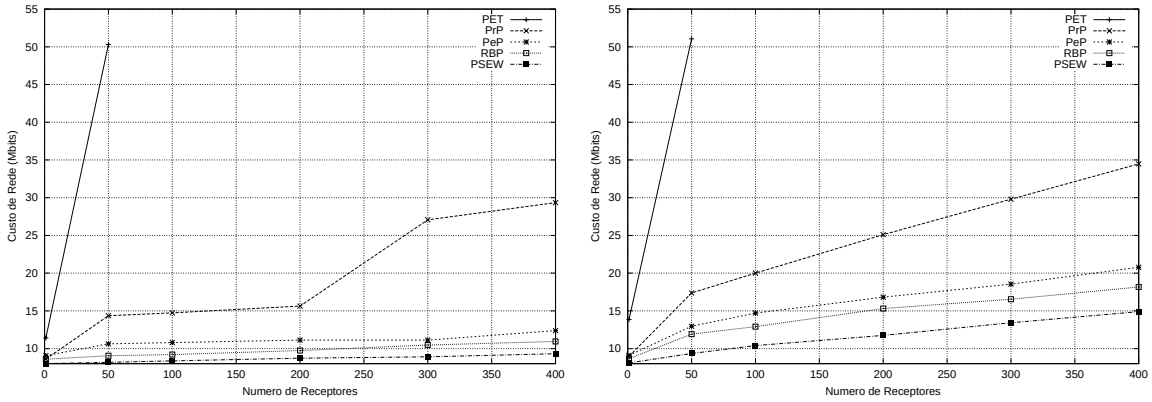
Neste experimento foi medido o número de pacotes perdidos devido à implosão, para cada protocolo, em função do aumento do grupo. A expectativa é que, à exceção do protocolo PET, implosão seja em geral evitada.

Foram realizadas simulações variando-se o número de receptores entre 1 e 400, utilizando os parâmetros estabelecidos nas Seções 4.3 e 5.1 (os parâmetros dos protocolos foram configurados de maneira conservadora, no sentido de evitar implosões). Como o objetivo deste experimento é isolar perdas por implosão, manteve-se a taxa de erro nos links em 0. Todos os protocolos, com exceção do PET, mostraram-se eficientes em evitar implosões, pois não foram registradas perdas de respostas em **nenhum** dos casos avaliados. Em contraste, um altíssimo número de perdas por implosão foi observado nas simulações com PET: enquanto que com um receptor não houve implosão, com 50 receptores (segundo valor testado) houve mais de 40 mil respostas perdidas.

Tal e qual em uma rede real, o alto número de implosões que PET ocasiona inviabilizou a realização de experimentos com mais de 50 receptores, tanto na avaliação de implosão quanto nos experimentos relatados a seguir.

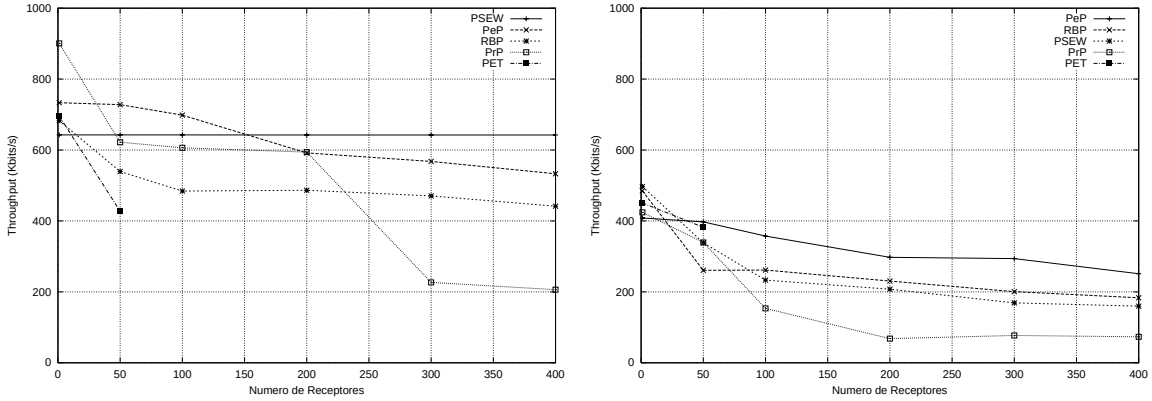
5.3 Custo de Rede

Este experimento avalia o custo de rede demandado por cada protocolo. A Figura 3 mostra os resultados obtidos pelos cinco protocolos em cenários sem e com perdas, respectivamente (a) e (b). Em primeiro lugar, observa-se que, devido às implosões, o custo de rede de PET sobe agressivamente. Segundo, que o *ranking* entre os protocolos é mantido para todos os tamanhos de grupo: do melhor ao pior, PSEW, RBP e PeP, então PrP, e finalmente PET. O custo de rede do PrP é comparativamente alto, que decorre do menor controle do remetente sobre os receptores. O custo mínimo (ótimo) de rede teórico é de 8 Mbits (1000 transmissões multicast de 1000 bytes), assumindo-se um cenário perfeito, sem perdas. Observa-se que o custo de rede dos protocolos PSEW, PeP e RBP é bem próximo a esse valor ótimo.



(a) sem perdas (b) com perdas

Figura 3: Comparativo do custo de rede entre os protocolos, para diferentes tamanhos de grupo.



(a) sem perdas (b) com perdas

Figura 4: Comparativo de throughput entre os protocolos, para diferentes tamanhos de grupo.

No cenário com perdas, o custo de rede é maior para todos os protocolos, pois parte dos pacotes enviados é perdida e gera retransmissões, contribuindo para o aumento da banda necessária. A Figura 3(b) confirma o comportamento esperado: a inclinação das curvas é maior frente ao gráfico sem perdas porque a chance de um pacote multicast necessitar uma retransmissão cresce com o número de receptores.

5.4 Throughput

Este experimento avalia o throughput (efetivo) de cada protocolo, conforme descrito na Seção 4.3. Nas Figuras 4(a) e (b), o eixo x representa o tamanho do grupo, em número de receptores, enquanto o eixo y mostra o throughput em Kbits/s (de 0 a 1000, taxa ótima).

Em primeiro lugar, como nos casos anteriores, a curva referente ao PET está limitada a dois pontos (por motivos já explicados), suficientes para demonstrar a baixa escalabilidade do protocolo.

Segundo, no cenário sem perdas (Figura 4(a)), nota-se claramente que o protocolo PSEW

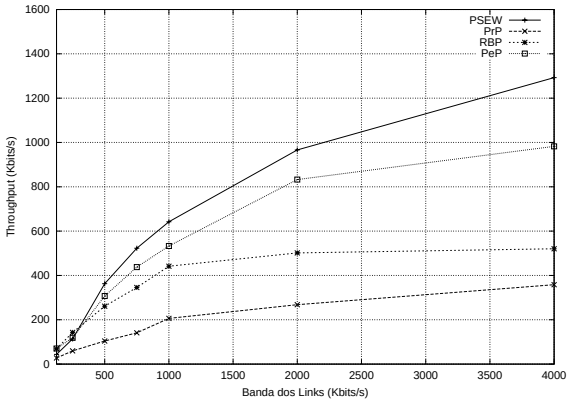


Figura 5: Impacto da variação da banda passante em relação ao throughput

mostra-se "perfeitamente escalável" (há uma queda quase imperceptível, inferior a $0,1\%$) quanto ao throughput para todos os tamanhos de grupo avaliados, com taxa efetiva superior a 640 Kbits/s. Relembrando (vide Seção 3.3), neste protocolo o remetente envia uma janela de pacotes e (apenas) então passa a solicitar, espaçadamente, respostas a cada subgrupo de receptores (cada subgrupo corresponde a um grupo IP multicast diferente). Novos dados são transmitidos somente após a chegada de uma resposta de cada um dos receptores. Dado esse funcionamento, espera-se que o throughput do PSEW seja mais afetado quando do aumento da latência, o que não foi o caso nos experimentos, pois a latência máxima se manteve independente do tamanho do grupo (vide Seção 4.2).

Globalmente, o PeP apresenta o segundo melhor desempenho; conforme pode ser visto na Figura 4(a), para grupos de até 150 receptores, PeP mostra-se superior ao PSEW. Observa-se ainda que os protocolos RBP e PeP sofrem uma queda suave de throughput à medida que o grupo cresce, com vantagem consistente do PeP sobre RBP; em contraste, o desempenho de PrP cai agudamente.

Diferentemente do cenário sem perdas, Figura 4(b) mostra que o melhor throughput pertence ao PeP. PSEW e RBP apresentam resultados similares, com PrP bastante abaixo a partir de 100 receptores. Em cenários com perdas de dados, POLLs e respostas, o PSEW não consegue repetir o bom desempenho devido aos longos atrasos decorrentes de sua política de detecção e recuperação de perdas.

5.5 Impacto da Variação da Banda

A largura de banda passante disponível nos links de uma topologia de rede possui forte impacto em qualquer protocolo de comunicação. Nos protocolos de polling, em particular, a redução da banda disponível levaria, mais cedo ou mais tarde, a uma redução nos níveis de feedback que o protocolo poderia aceitar antes que implosão ocorresse. Com exceção do PET, os protocolos de polling podem evitar implosão, porém dependem do feedback para atualizar a janela de transmissão no remetente e prosseguir de maneira eficiente.

No último experimento relatado, avalia-se o impacto da largura de banda passante no throughput dos protocolos, considerando o maior grupo receptor examinado (400 receptores). Os demais parâmetros foram mantidos (em particular, janela de 200 pacotes e pacote de dados com 1000 bytes). Foram avaliadas bandas variando entre 125 Kbits/s e 4 Mbits/s. No eixo x, representa-se a banda disponível (igual para todos os links), medida em Kbits/s, contra o throughput, no y. Na Figura 5, percebe-se que, conforme esperado, todos os protocolos avaliados apresentaram melhor desempenho em função do aumento de banda. Destacam-se nesse sentido

os protocolos PSEW e PeP, que tiraram maior proveito do incremento em banda disponível.

6 Conclusão

O assunto "protocolos para transporte multicast confiável e escalável" tem sido largamente discutido na literatura, conforme apontado na Introdução. No entanto, a imensa maioria dos protocolos multicast escaláveis propostos seguem a filosofia orientado a receptor. Já o uso de um esquema orientado a remetente baseado em polling ainda não foi discutido a contento. [13] sugere o uso de polling em protocolos de transporte multicast com o intuito de evitar implosão de feedback; tal trabalho analisa o custo de rede de protocolos de forma analítica e experimental, considerando redes locais (com reduzido número de receptores e RTTs homogêneos). Em [1, 2], um protocolo multicast baseado em polling, denominado PRMP, é proposto com base no planejamento temporal para chegada de respostas; são apresentadas duas versões do protocolo, uma plana e outra hierárquica, que são avaliadas através de simulações para "pequenos" grupos (menos de 100 receptores). Em contraste, o presente artigo descreve versões generalizadas de protocolos de polling, os compara através de simulação (com um simulador que emprega um modelo de rede mais detalhado, o `ns`), e baseando-se em configurações com maior número de nós de rede e receptores.

Aplicações que necessitam controlar a composição do grupo de receptores enquanto dados são enviados, com confirmação gradual e positiva do recebimento dos mesmos, demandam o uso de protocolos com semântica ao estilo TCP-multicast ([3]), ou seja, com controle de erro baseado no remetente. Para evitar implosão decorrente dos pacotes de confirmação (ACKs), é necessário o uso de uma tecnologia eficiente para controle do feedback gerado pelos receptores. Polling resolve este problema: reduz o volume de feedback, a partir do momento em que o remetente informa explicitamente quando cada receptor deve responder.

Protocolos de transporte multicast baseados em polling podem ser usados quando a aplicação exige que o remetente envie dados de forma confiável, eficiente e relativamente escalável[3] via Internet a membros de um grupo cuja composição ela (a aplicação remetente) precisa conhecer e controlar. Estes requisitos não são atendidos por protocolos de transporte multicast orientados a receptor.

Os resultados de simulação apresentados na Seção 5 mostram que protocolos de polling podem utilizar suas informações sobre receptores para evitar eficazmente perdas por implosão, e portanto são uma alternativa eficiente para o transporte multicast confiável dependendo da ordem de magnitude do tamanho do grupo. Os resultados mostram também que um protocolo de transporte multicast baseado em polling pode apresentar custo de rede bem próximo ao valor ótimo, isto é, mínimo (vide PSEW, PeP e RBP na Figura 3). Tal é explicado pelo fato que multicast transporta de maneira eficiente os dados downstream do remetente aos receptores, e mecanismos de polling reduzem efetivamente a quantidade de feedback produzida por receptores. Por fim, um dos protocolos (PSEW) apresentou excelente throughput, sem perda de desempenho com o aumento do grupo em cenários sem perdas induzidas, e no cenário com perdas, a maioria dos protocolos lidou eficientemente com a detecção e retransmissão de pacotes perdidos; em particular, o protocolo PeP (periódico) teve seu throughput diminuindo em apenas 32% (de 400 para 270 Kbits/s) quando o grupo aumentou de 1 para 400 receptores (com a ressalva que a distância até o receptor mais longe foi igual em todos os casos, 11 hops).

Estes resultados demonstram que a abordagem baseada em polling pode ser utilizada eficientemente em configurações de grupo em larga escala, com até algumas centenas de nós, sem necessidade de acrescentar suporte específico em roteadores ou a configuração (tipicamente complexa) de nós em forma hierárquica.

[3] pelo menos até algumas centenas de receptores.

Como trabalhos futuros, vislumbra-se o uso dos modelos genéricos no projeto de novos protocolos direcionados a aplicações específicas de multicast, como suporte a sistemas de comunicação em grupo com tolerância a falhas (p.ex., NEWTOP [8]) e, no contexto do processamento de alto desempenho, a distribuição de tarefas em agregados de computadores. Por fim, será estudada a interação desses protocolos de polling com demais protocolos da Internet, através de simulações envolvendo fluxos de outros protocolos e tráfego de fundo.

Agradecimentos

Os autores gostariam de agredecer à FAPERGS, que financiou parte deste trabalho, a Guilherme B. Bedin e Hisham H. Muhammad, que contribuíram através de discussões sobre os protocolos, e por fim, aos revisores anônimos deste trabalho, que fizeram sugestões de como melhorar o texto.

Referências

[1] M. P. Barcellos, P. D. Ezhilchelvan, "An End-to-End Reliable Multicast Protocol Using Polling for Scalability", In IEEE INFOCOM'98, San Francisco, April 98, pp.1180-1187.

[2] M. P. Barcellos, "PRMP: A Scaleable Polling-based Reliable Multicast Protocol". Ph.D. Thesis, Department of Computing Science, Newcastle University, Newcastle upon Tyne, October 1998, 200pp.

[3] M. P. Barcellos, A. Detsch, G. B. Bedin, H. H. Muhammad, "Efficient TCP-like Multicast Support for Group Communication Systems", IX Brazilian Symposium on Fault-Tolerant Computing (SCTF), Proceedings, SBC (Porto Alegre), Florianópolis, 5-7 March 2001, pp.192-206.

[4] K. Birman, "Building Secure and Reliable Network Applications", Manning: Prentice Hall, 1996. 500p.

[5] L. Breslau et alli, "Advances in Network Simulation", IEEE Computer, v.33, n.5, pp. 59-67, May 2000.

[6] J. Byers, M. Luby, M. Mitzenmacher, "Fine-Grained Layered Multicast", INFOCOM2001

[7] D. Dolev and D. Malki, "The Transis Approach to High Availability Cluster Communication", Communications of the ACM, v.39, n.4, April 96, pp. 64-70.

[8] P. Ezhilchelvan, R. Macedo, and S. Shrivastava, "Newtop: A Fault-Tolerant Group Communication Protocol". In IEEE 15th Intl. Conf. Distributed Computing Systems, pp.296-306, May 1995.

[9] S. Floyd, V. Jacobson, S. McCanne, C. Liu and L. Zhang, "A Reliable Multicast Framework for Light-Weight Sessions and Application Level Framing", IEEE/ACM Transactions on Networking, v.5, n.6, Dec. 1997, pp. 784-803.

[10] H. Garcia-Molina, A. Spauster, "Ordered and Reliable Multicast Communication", ACM TOCS, v.9, n.3, Ago. 1991, pp.242-271.

[11] Georgia Tech Internetwork Topology Models (gt-itm), http://www.cc.gatech.edu/projects/gtitm/, acessada em 10/11/2001.

[12] M.A. Hiltunen, R.D. Schlichting, "A Configurable Membership Service", IEEE Transactions on Computers, v.47, n.5, pp.573-586, May 1998.

[13] L. Hughes, M. Thomson, "Implosion-Avoidance Protocols for Reliable Group Communications", Proc. of 19th Conf. on Local Computer Networks, Minneapolis, Minnesota, October 1994.

[14] C. Liu, P. Ezhilchelvan, and M. Barcellos, "A Multicast Transport Protocol for Reliable Group Applications", First International Workshop on Networked Group Communication - NGC'99, Springer-Verlag, 1999, pp.170-187.

[15] C. Maihöfer, "A Bandwidth Analysis of Reliable Multicast Transport Protocols", II Workshop on Networked Group Communication, Stanford, Nov. 2000.

[16] G. Morgan, S. K. Shrivastava, P. D. Ezhilchelvan and M. C. Little, "Design and Implementation of a CORBA Fault-Tolerant Group Service", In 2nd IFIP WG 6.1 International Working Conference on Distributed Applications and Interoperable Services, Helsinki, June 99.

[17] P. Radoslavov, C. Papadopoulos, R. Govindan, D. Estrin, "A Comparison of Application-level and Router-assisted Hierarchical Schemes for Reliable Multicast", In INFOCOM2001.