Difusão Seletiva Semanticamente Confiável Baseado na Técnica de Escalonamento (m-k)-*firm*

Wilian Queiroz[1], Lau Cheuk Lung[2],
Luciana de O. Rech[2], Luiz A. de Paula Lima Junior[1]

[1] Programa de Pós-Graduação em Informática Aplicada - CCET
Pontifícia Universidade Católica do Paraná - Curitiba - PR - Brasil.
[2] Programa de Pós-Graduação em Ciência da Computação - INE - CTC
Universidade Federal de Santa Catarina - Florianópolis - SC - Brasil.
Computer Science Editorial, Tiergartenstr. 17,
lau.lung@inf.ufsc.br, {wqueiroz, laplima}@ppgia.pucpr.br, lrech@das.ufsc.br

Abstract. This paper proposes a new semi-reliable *multicast* algorithm based on the (m, k)-*firm* scheduling technique, where in each consecutive k window messages sent by a sender, at least m of these messages must be received by the receiver. To assure this restriction, message recovery mechanisms from reliable *multicast* protocols can be used. This algorithm is mainly targeted to applications that may suffer losses, as long as these losses do not occur consecutively and do not overrun a specified maximum value of message, without any degradation of the quality of service.

Palavras-chave: *multicast*, algoritmos distribuídos, tempo real.

1 Introdução

A necessidade de comunicação em grupo vem crescendo com o decorrer dos anos, em boa parte pela necessidade de disseminação de informações pela Internet a um grande número de usuários. Distribuição de conteúdo, ensino à distância, jogos *multipliers*, aplicações de áudio/vídeo, *stock quotes*, são exemplos de aplicações que podem se beneficiar com ferramentas de comunicação em grupo. Estas ferramentas podem ser implementadas a partir de protocolos de difusão seletiva (ou *multicast*) e tem como vantagens [1]: o desempenho otimizado da rede (economia de banda passante); o suporte a aplicações distribuídas; a facilidade de crescimento em escala (permite, sem esforços adicionais, a distribuição de mensagens para poucos ou muitos receptores); e a maior disponibilidade da rede.

Devido à existência de inúmeras variações de funcionalidades disponibilizadas por aplicações distribuídas, o *multicast* teve que se adaptar a estas variações, e conseqüentemente aos seus requisitos, tais como [2]: latência de entrega; ordenamento de mensagens; escalabilidade e entrega. Com base nestes requisitos e na existência de aplicações que fazem uso do mecanismo *multicast*, onde tais aplicações podem possuir requisitos diferentes no que diz respeito à tolerância a faltas na entrega de mensagens, os protocolos *multicast* podem ser agrupados em três categorias: não-confiável, confiável, e semi-confiável. O protocolo S-RM (*Semantically Reliable Multicast*) [3] é uma derivação deste último grupo de protocolos. O protocolo S-RM

toma como base a obsolescência de mensagens, levando em consideração que em algumas aplicações distribuídas algumas mensagens podem subscrever ou conter informações suficientes para tornar mensagens enviadas anteriormente irrelevantes (ou obsoletas). Baseado nesse princípio, este tipo de protocolo relaxa a confiabilidade, de acordo com a semântica da aplicação. Isto é, se uma mensagem mais recente é recebida, as anteriores perdidas não precisam mais ser recuperadas.

Este artigo propõe um novo algoritmo de *multicast* semi-confiável baseado na técnica de escalonamento *(m, k)-firm* [4]. Este algoritmo foi desenvolvido para aplicações em que a abordagem de obsolescência da mensagem [3] não é suficiente. A técnica *(m, k)-firm* foi proposta originalmente para lidar com situações de sobrecarga transientes em abordagens de escalonamento tempo-real de processos em Sistemas Operacionais.

Neste artigo, a técnica *(m, k)-firm* é explorada em um algoritmo de *multicast* semi-confiável. Nesta proposta, cada participante de um grupo *multicast* possuirá um parâmetro *(m, k)-firm* que indicará qual a taxa máxima tolerável de perdas na transmissão, isto é, este parâmetro estabelece quantas *m* mensagens de uma janela de *k* mensagens consecutivas deverão ser recebidas, de um mesmo emissor, com sucesso pelo receptor. Caso esta taxa máxima seja ultrapassada, $(k - m) / k$ | k seja $\neq 0$, uma falha é apontada e o receptor deve tentar recuperar esta perda.

O artigo está estruturado da seguinte forma: a seção 2 apresenta uma síntese sobre *multicast*. Na seção 3 é abordada a técnica de escalonamento *(m, k)-firm*. Na seção 4, são apresentados alguns dos principais trabalhos relacionados. O protocolo proposto é apresentado nas seções 5 (estudo de caso) e 6, juntamente com simulações, resultados e experimentos realizados. Na seção 7 são apresentadas as conclusões finais.

2 Conceitos Gerais de *Multicast*

A tecnologia *multicast*, também conhecida como difusão seletiva, é a base de um serviço de rede no qual uma determinada fonte pode enviar uma mensagem simultaneamente para diversos receptores interessados. A infra-estrutura de rede fica encarregada de replicar o fluxo de dados para todos os receptores que registraram interesse em receber esta mensagem. Este mecanismo utiliza um conjunto de receptores que definem um grupo *multicast*, sendo dinamicamente alocado, e válido durante toda a sessão *multicast*. Transmissores utilizam o endereço *multicast* como IP destino para transmissão dos pacotes que desejam enviar para os membros do grupo.

Existem três tipos de protocolos *multicast*: confiável, não-confiável e semi-confiável. Os protocolos *multicast* confiáveis visam fornecer uma "entrega garantida", acrescentando certo nível de confiabilidade na entrega de mensagens aos receptores. Os seguintes aspectos devem ser abordados neste tipo de protocolos [2]: controle de erro, de fluxo, de congestionamento, de sessão e de implosão. A necessidade da existência destes diversos mecanismos de controle gera uma desvantagem, onde um único membro do grupo mais lento pode degradar o desempenho do protocolo como um todo. Por outro lado, os protocolos *multicast* não-confiável não fornecem nenhuma garantia de entrega.

Uma forma encontrada para tratar a questão da confiabilidade foi relaxar o requisito de consistência forte dos protocolos *multicast* confiáveis, surgindo assim um novo tipo, os protocolos *multicast* semi-confiáveis. Este tipo de protocolo não garante

a entrega de uma mensagem a todos os membros do grupo, mas somente a um subconjunto de mensagens com determinada prioridade ou importância. Isto implica que nem todas as mensagens que foram perdidas serão retransmitidas. É empregado um algoritmo para verificar quais mensagens são relevantes para que não haja degradação da qualidade do serviço da aplicação.

As aplicações distribuídas que fazem uso de mecanismos de difusão seletiva, têm diferentes requisitos no que diz respeito à entrega confiável de pacotes, por exemplo: garantia de ordem total na entrega, troca de confiabilidade total por um melhor tempo de entrega e necessidade de mais de um membro como fonte de dados. Por existirem estas diferenças, as propostas de protocolos *multicast* seguiram diferentes direções para atendê-las. Entre as propostas existentes podemos citar: WAIT [5], [6], S-RM [3], SRM [7], STORM [8], TUNA [9], PRTP [10], PGM [11].

Durante o ciclo de vida de uma sessão *multicast*, três importantes eventos podem ocorrer [12]: dinâmica dos grupos, da rede e de tráfego. O primeiro e o segundo aspecto preocupam-se em manter a qualidade do serviço levando em conta a entrada e saída de membros do grupo, bem como sua atualização, o terceiro preocupa-se com o controle de fluxo, controle de congestionamento e controle de erro.

3 A Técnica *(m, k)-firm* para Especificar *Deadlines*

Essa técnica considera um modelo de tarefas onde uma tarefa periódica pode perder eventualmente *deadlines* sem que ocorra uma falha no sistema, desde que as perdas não ultrapassem um patamar especificado, considerando uma janela de execuções consecutivas. A idéia básica desse modelo de tarefas é a captura da semântica de algumas aplicações que executam periodicamente, e que podem "perder" algumas ativações, desde que essas perdas não ocorram consecutivamente, dentro de um valor máximo especificado pela semântica da aplicação. Nesse modelo de tarefas, cada tarefa possui um deadline *(m, k)-firm* que especifica que em uma janela de k ativações periódicas, para permanecer em um estado correto, essa tarefa precisa ter pelo menos m ativações com seus *deadlines* atendidos [4].

A notação *(m, k)-firm* [4] é genérica, expressando que em determinadas aplicações existe um limite superior tolerado de perdas de deadlines que podem ocorrer sem que haja uma degradação considerável da qualidade do serviço, este limite pode ser dado por k-m. Caso o limite superior seja excedido, uma falha dinâmica é assumida no sistema. Um deadline especificado na forma *(m, k)-firm* é capaz de representar vários tipos de restrições. Em particular, ($m = k \geq 1$) caracteriza uma restrição como *hard-deadline*, e um vasto número de valores para k com $(k - m) / k$ igual à taxa máxima permitida de perda representa uma restrição *soft-deadline*. Na Seção 6 será apresentada a proposta da adoção da técnica *(m, k)-firm* em algoritmos de *multicast* semi-confiável.

4 Trabalhos Relacionados

Diversas propostas envolvendo o assunto foram estudadas, por motivo de limitação de espaço, serão brevemente descritos apenas alguns dos trabalhos que serviram de inspiração para o desenvolvimento do algoritmo apresentado neste artigo.

O STORM [8] possui como idéia básica uma distribuição de NACKs e reparos de pacotes sobre uma estrutura onde todos os nós participantes são considerados como aplicações *end-point*. O TUNA (*Tunable Quasi-Reliable Multicast Protocol*) [9] possui políticas apropriadas para a confiabilidade seletiva, utilizando propriedades estatísticas para prover ao receptor, informações para serem utilizadas nas definições de decisão de confiabilidade.

O WAIT [5] combinou o SRM [7] e o PGM [11] conseguindo um melhor desempenho, pois reduz a carga na rede, alcançando uma melhor qualidade. Foi projetado para se ajustar a diferentes requisitos de qualidade de uma sessão multimídia e reduzir a carga na rede, além de uma melhor qualidade de serviço nas aplicações sobre a Internet com menor suporte do roteador. O PGM, definido em [11], é um protocolo de transporte *multicast* confiável para aplicações que requerem uma difusão seletiva de uma simples origem para múltiplos receptores.

No protocolo S-RM [3] foi observado que em algumas aplicações distribuídas algumas mensagens podem subscrever ou conter informações suficientes para tornar mensagens enviadas anteriormente irrelevantes (ou obsoletas). Baseado neste princípio são definidos protocolos que relaxam a confiabilidade, de acordo com a semântica da aplicação, oferecendo um melhor desempenho e escalabilidade.

Em [3] levou-se em consideração que para cada par de mensagens (m, m') existe uma relação de obsolescência, podendo esta relação ser expressa como $m \subset m'$, onde se lê: m é obsoleta para m'. Isto significa que nesta relação se m' é entregue a aplicação e m não, isto não afeta a mesma. A notação $m \subseteq m'$ é utilizada como abreviatura para $m \subset m' \vee m = m'$. A idéia é que as mensagens carreguem uma informação de controle relativa à sua relação de obsolescência, que deverá ser utilizada somente para prevenir congestionamentos. Quando verificado uma situação de ocupação total do buffer, o protocolo busca mensagens obsoletas no mesmo para serem eliminadas. Cabendo a cada membro do grupo a responsabilidade de verificação e eliminação das mensagens obsoletas de seu buffer local. Com a desocupação do buffer, o protocolo retoma a operação confiável.

5 Cenários e Estudo de Caso

Verificamos três diferentes cenários onde demonstramos algumas das possíveis variações de combinação da técnica de escalonamento *(m, k)-firm* e do *multicast*. No primeiro cenário, consideramos a situação que a restrição *(m, k)-firm* é estabelecida no início do processo e se mantêm fixa até o final. No segundo cenário o grupo *multicast* possui mais de um tipo de restrição, sendo uma para cada tipo de receptor. No terceiro cenário, consideramos uma restrição *(m, k)-firm* inicial para todos os receptores, esta restrição será dinamicamente alterada de acordo com a semântica atual, podendo assim chegar a uma restrição diferente para cada receptor. Isso permite certa flexibilidade ao receptor para alterar sua restrição, caso julgue necessário.

Uma aplicação para o primeiro cenário seria um sistema computacional que recebe dados de radares para traçar a trajetória correta de um artefato aéreo (*radar tracking*). Quanto mais pontos na trajetória melhor será a qualidade do traçado. No segundo cenário, enquadram-se as aplicações que possuem receptores com diferentes necessidades de confiabilidade de entrega, por exemplo, em uma aplicação de jogo distribuído. É possível a existência de diferentes papéis para os participantes

(jogadores ou ouvintes). Essa diferença de papéis torna clara as diferentes restrições *(m, k)-firm* existentes, ou seja, os "jogadores" necessitam de uma restrição *(m, k)-firm* muito mais rigorosa (*m*/*k* próximo de 1) do que os "ouvintes", visto que se um participante "ouvinte" não receber alguma mensagem isto não implicará no resultado do jogo. O terceiro cenário é um aperfeiçoamento dos anteriores, acrescentando a eles a captura da semântica da aplicação e alterando dinamicamente a restrição *(m, k)-firm* em tempo de execução, tendo com isto um aumento da restrição *(m,k)-firm* quanto a disponibilidade de recurso de rede o permita, ou sua diminuição caso seja verificada alguma falta do mesmo, por exemplo em um princípio de congestionamento.

6 *Multicast* (m, k)-*firm*

A utilização da restrição *(m,k)-firm* pode ser empregada em diversas aplicações distribuídas, mas com maior aproveitamento nas aplicações que apresentem características de tolerância a faltas na recepção da mensagem.

Nesta proposta, cada participante de um grupo *multicast* possuirá um parâmetro *(m, k)-firm* que indicará qual a taxa máxima tolerável de perdas na transmissão. Este parâmetro estabelece quantas ***m*** mensagens de uma janela de ***k*** mensagens consecutivas deverão ser recebidas, de um mesmo emissor, com sucesso pelo receptor. Caso esta taxa máxima seja ultrapassada, (k–m)/k ⌊ k seja ≥ 1, uma falha é apontada e o receptor deve tentar recuperar (solicitar retransmissão) as mensagens perdidas até que o seu *(m, k)-firm* seja atendido. Vale observar que o algoritmo pode ser considerado confiável quando os valores de ***m*** e ***k*** forem iguais (m = k ≥ 1).

6.1 Aplicando a Restrição (m, k)-*firm*

O presente algoritmo possui um histórico (tupla que armazena o estado das últimas N mensagens recebidas). Cada participante do grupo *multicast* possui um buffer que armazena mensagens recebidas por certo tempo, permitindo que este coopere no protocolo de comunicação, agilizando assim o reparo de mensagens. Como esse buffer é finito com um tamanho N, cada histórico também vai ser finito com um tamanho fixo N.

Sabendo o número de seqüência da última mensagem que chegou e que está armazenada no histórico, é possível saber o estado desta e das últimas N–1 mensagens. A Figura 1 representa um histórico com tamanho de 5 *slots* de mensagens (N=5). Nesse histórico, cada mensagem recebida é representada pelo valor 1, as mensagens que ainda estão faltando são representadas pelo valor 0 (isso facilita a implementação eficiente de um histórico, usando uma seqüência de bits 0s e 1s).

No exemplo da Figura 1, caso a última mensagem recebida tenha o número de seqüência 4000, é possível verificar o estado das mensagens cujo número de seqüência esteja no intervalo [3996-4000], e determinar que a mensagem 3999 ainda não havia chegado (daí o valor 0 no histórico).

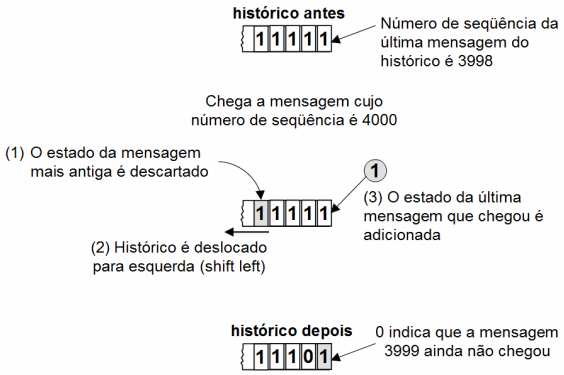


Figura 1 – Histórico de participante.

6.2 Algoritmo *Multicast* (m, k)-*firm*

Em uma implementação do algoritmo *Multicast (m, k)-firm* um emissor difunde a um grupo de receptores mensagens de dados. Estes, por sua vez, a cada mensagem recebida, verificam suas restrições *(m,k)-firm*. Ao verificarem uma distorção nesta restrição, enviam uma solicitação de reparo ao grupo. Como tal mensagem é recebida por todos os membros do grupo, os quais possuem um buffer local onde armazenam as mensagens enviadas ou recebidas, todos podem participar no reparo [7], agilizando assim o restabelecimento da restrição *(m, k)-firm* do receptor solicitante.

Na Figura 2 é apresentado de forma simplificada o algoritmo *multicast (m-k)-firm*. São utilizados alguns procedimentos (p_n), funções (f_n) e transações (t_n). A notação {auto} expressa uma execução automática. Consideram-se três tipos de mensagens: de dados ($\in D_{data}$), de solicitação de reparo ($\in D_{nack}$) e de reparo propriamente dita ($\in D_{repair}$). Este último tipo é apenas utilizado no ato de envio do reparo, auxiliando na prevenção de implosões deste tipo de mensagens. Neste algoritmo, a letra "m" representa uma mensagem e a letra "M" representa um dos parâmetros da restrição *(m, k)-Firm*.

A estrutura de uma mensagem no algoritmo de *Multicast (m, k)-firm* possui um identificador do emissor (*senderID*), um número de seqüência da mensagem deste emissor, um tipo (*type*) que indica a finalidade da mensagem e um conteúdo (*content*), que é a informação a ser entregue à aplicação.

Algumas variáveis são utilizadas para armazenar o tempo de espera para envio das mensagens: T_{data} para as mensagens de dados, T_{repair} as mensagens de reparo, e T_{nack} para as mensagens de solicitação de reparo (*NACK*). Enquanto *Sq* define o tamanho máximo dos bufferes; e S_{msg} define a quantidade de mensagens que devem ser solicitadas caso haja uma distorção da restrição *(m-k)-firm.* M representa a quantidade mínima de mensagens que devem ser recebidas de uma janela de K mensagens; K representa a quantidade de mensagens que compõem a janela de controle de restrições. As constantes TP_{data}, TP_{repair} e TP_{nack} definem os tipos de mensagens de dados, de reparo e de solicitação de reparo, respectivamente.

```
p1: scheduleMsg(m)
   delay(m.type)
   addMsg(Qsend,m)
p2: sendRepair(m)
   cid ← m.content
   for all cid | not newMsg(Qrec,cids[i]) do
     scheduleMsg(getMsg(Qrec,cid[i]))
     updateStateMsg(Qrec,m.id,TPrepair)
   end-for

f1: mkFirm(m)
  Qtmp←{∀m1 ∈ Qrec | m1.id.sender =
      m.id.sender ∧ m1.id.seq ≥ m.id.seq - K }
  if m.id.seq >= K ∧ Qrec.size < M
    then
      cids←{∀ids | id ⊄ Qtmp} ∧ cids.size ≤ Smsg
      m2 ← new message;
      m2.content ← cids
      m2.type ← TPnack
      scheduleMsg(m)
    end-if
t1 : receive( m) {auto}
    if m.sender ≠ self then
      if m ∈ Ddata ∨ m ∈
        Drepair then
      if isNewMsg(Qrec,m) then
        addMsg(Qrec,m)
        mkFirm(m)
      else
        updateStateMsg(Qrec,m.id,TPData)
      end-if
      else
        if m ∈ Dnack then
          sendRepair(m)
        end-if
      end-if
```

Figura 2 – Definição algoritmo *Multicast (m, k)-firm*.

O algoritmo proposto apresenta o seguinte funcionamento: um determinado emissor difunde a um grupo de receptores interessados nas mensagens de dados. Após recebimento (t_i) de cada mensagem pelos receptores, os mesmos são individualmente responsáveis pela detecção de falha no recebimento de alguma mensagem através da verificação de sua restrição *(m, k)-firm* [4] (f_1), bem como pelo envio ao grupo de uma mensagem de solicitação de reparo (p_2), sendo esta solicitação composta pelas identificações das mensagens que são necessárias para restabelecimento da restrição *(m, k)-firm*. Todos os membros do grupo *multicast* recebem as mensagens de *NACK*, e pelo fato de todos possuírem um buffer local com todas as *N* últimas mensagens recebidas, todos podem participar no reparo [7]. Isto é importante porque diminui a sobrecarga no remetente original e faz o processo de reparo mais rápido se a distância entre o remetente do *NACK* e o remetente original for grande. Para minimizar o número de *NACKs* e reparos, estas duas operações são precedidas pelo *back-off* exponencial. Isto significa que em vez de emitir o pacote diretamente, o receptor *p* espera um tempo aleatório (p_1), baseado na distância relativa ao outro receptor, e se outro receptor emitir um pacote correspondente, o receptor *p*, ao receber este pacote, cancela seu envio [7]. O receptor solicitante continua a emitir pedidos de reparo até que o reparo seja recebido ou a mensagem em questão se torne obsoleta ao processo, por exemplo, depois de *t* tentativas.

6.3 Simulações e Resultados

Para avaliar o desempenho do algoritmo proposto, foram realizados testes comparativos. Consideramos um sistema simples, com apenas um *sender*, um *fast receiver* e um *slow receiver* – mesmo modelo utilizado em [3]. Na simulação de um ambiente *multicast*, utilizamos o framework *Simmcast* [2], o qual configuramos de forma a obter uma igualdade no ambiente de execução. Para isto, alguns dos parâmetros utilizados na execução dos testes foram mantidos fixos em todas as simulações para uma melhor clareza na visualização dos resultados obtidos. Tais

parâmetros são: percentual de mensagens perdidas durante a comunicação de 30%; tempo de espera para envio de uma mensagem de NACK ou de retransmissão de 80ms; tempo de espera de 200ms no *slow receiver* para recebimento de uma nova mensagem; tamanho do buffer de armazenamento das mensagens de 40 *slots*.

A Figura 3 apresenta uma comparação, levando em consideração o envio de mensagens de solicitação de reparo (NACK), da simulação de execução de um protocolo *Multicast* Confiável (RM) e o nosso algoritmo utilizando 4 tipos de restrições diferentes: (4,4)-*firm*, janela de confiabilidade total; (3,4)-*firm*, janela de confiabilidade com até 25% de perdas; (2,4)-*firm*, janela de confiabilidade com até 50% de perdas; (1,4)-*firm*, janela de confiabilidade com até 75% de perdas.

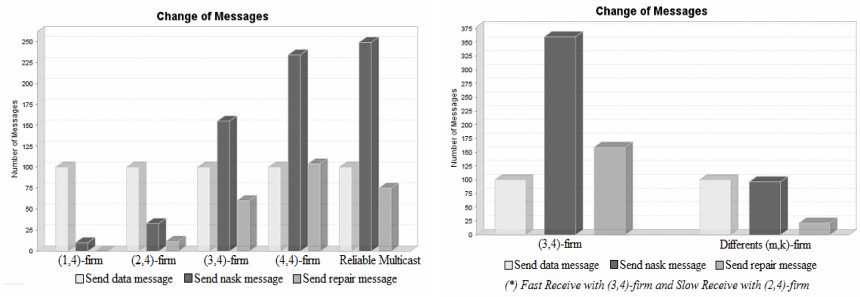


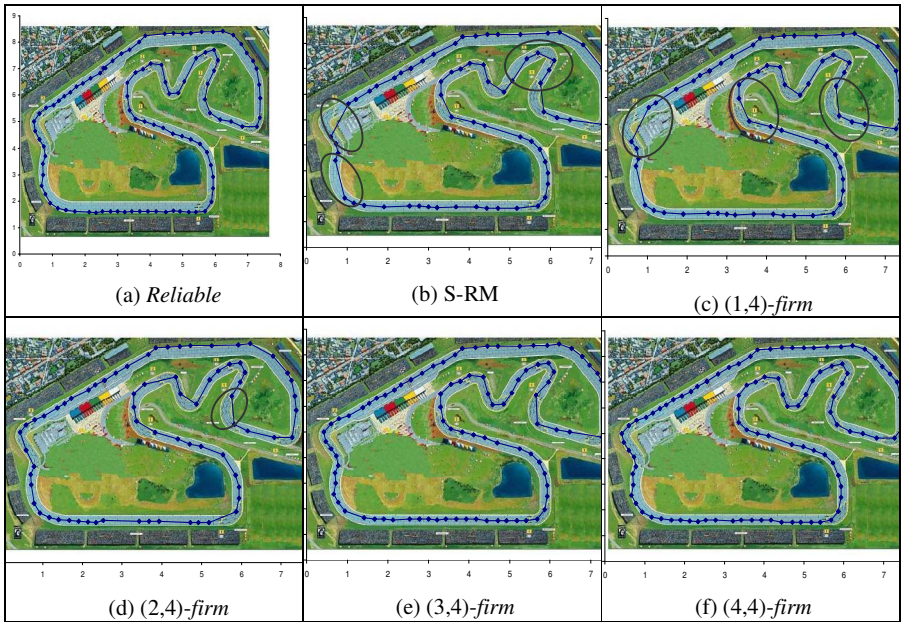
Figura 3 – Mensagens de solicitação de reparo.

Analisando a Figura 3(a), percebe-se que o algoritmo com restrição (4,4)-*firm* se assemelha com a execução do RM, situação esperada devido aos níveis de confiabilidade serem iguais (100%). Nas demais restrições apresentadas, verificou-se uma queda significativa do número de NACK enviados.

Na Figura 3(b) buscou-se efetuar uma comparação entre duas estratégias diferentes de utilização do nosso algoritmo, representando respectivamente os cenários 1 e 2 apresentados na seção 5 deste artigo. Em um dos casos, utilizamos a restrição (3,4)-*firm* tanto para o *fast receiver* como para o *slow receiver*, no outro caso, para o *fast receiver* utilizamos uma restrição (3,4)-*firm* e para o *slow receiver* utilizamos uma restrição (2,4)-*firm*. Pode-se observar claramente o ganho ao utilizarmos restrições diferentes para o *fast receiver* e *slow receiver*. Houve uma queda de aproximadamente 66% nas mensagens de NACK, e conseqüentemente houve redução na retransmissão de mensagens, somando uma queda no total geral de mensagens transmitidas na comunicação de 65%. Também foi analisada a ocupação do buffer de recebimento pelo *throughput* onde verificou-se que a restrição e ocupação do buffer são diretamente proporcional, isto é, quanto menor a restrição menor a ocupação.

A Figura 4 apresenta uma simulação em que receptores recebem dados discretos sobre a posição de um automóvel numa pista. Os dados fornecem a posição (um ponto) e são utilizados para traçar a trajetória do automóvel. Para isso, é preciso que se receba o máximo de pontos possíveis. A Figura 4(b) ilustra a utilização do protocolo S-RM por um receptor lento (*Slow Receiver*). O uso deste protocolo não

trouxe benefício, pois permitiu perdas de mensagens em intervalos consideráveis - basta receber um dado (um ponto da trajetória) mais recente para eliminar a necessidade de recuperar os dados anteriores perdidos. As Figuras 4(c), 4(d), 4(e) e 4(f) apresentam os resultados das simulações utilizando o algoritmo *Multicast* (m, k)-*firm* com as restrições (1,4)-*firm*, (2,4)-*firm*, (3,4)-*firm* e (4,4)-*firm*. Notou-se algumas perdas de qualidade também no algoritmo *Multicast (m, k)-firm*, nas restrições mais leves (Figuras 4(c) e 4(d)) onde realmente não se obteve uma boa garantia na entrega. Os círculos indicam falhas na trajetória em relação à pista real.



(a) *Reliable* (b) S-RM (c) (1,4)-*firm*

(d) (2,4)-*firm* (e) (3,4)-*firm* (f) (4,4)-*firm*

Figura 4 – Trajetória do artefato móvel – *Slow receiver*.

7. Conclusões

Em nossos estudos verificamos que a utilização de protocolos semanticamente confiáveis traz grandes vantagens às aplicações distribuídas em sistemas heterogêneos. Nos protocolos semi-confiáveis padrão, a forma de recuperação dos erros é igual para qualquer aplicação que os utilize. Já nos protocolos semanticamente confiáveis, cada aplicação tem a possibilidade de definir sua relação de obsolescência.

Este trabalho apresentou um novo algoritmo de difusão semanticamente confiável para otimização de troca de mensagens e buffer de retransmissão para aplicações que permitem a perda de algumas mensagens sem que haja considerável degradação do serviço. Com base na semântica de cada aplicação, este trabalho traz uma forma de parametrização desta semântica, através da definição de restrições. Seu desempenho é obtido pelo método de retransmissão baseado no receptor que utiliza a restrição *(m,k)*-

firm para verificação de descarte seletivo de mensagens perdidas de acordo com a janela de confiabilidade estabelecida.

O grande diferencial desta proposta é a delegação de responsabilidade aos receptores, ao contrário do S-RM em que a responsabilidade da semântica da aplicação está contida somente no emissor. Mas com esta responsabilidade também vem à flexibilidade dos receptores poderem, com base no ambiente ou na finalidade do mesmo, atribuir uma restrição adequada ao momento, de forma adaptativa, contribuindo de uma maneira geral ao desempenho do grupo como um todo.

Referências

[1] Rede nacional de Ensino e Pesquisa, http://www.rnp.br/noticias/2004/not-040128b.html
[2] Barcellos, M., ROESLER, V.: M&M: *Multicast*ing & Multimídia. Em: SBC, (2000).
[3] Pereira, J., Rodrigues, L., Oliveira, R. Semantically Reliable *Multicast*: Definition, Implementation, and Performance Evaluation. IEEE Trans. On Computers, Vol.52, N°2, (2003).
[4] Hamdaoui M., Ramanathan P. A dynamicpriority assignment technical for streams with (m,k)-*firm* Deadline, IEEE Trans. on Computers, 44(4), pp.1443-1451. (1995).
[5] Mane, P.: WAIT: Selective Loss Recovery For Multimedia *Multicast*. MS Thesis Computer Science Department, WPI, (2000).
[6] Bortoleto, C., Lau C. L., Siqueira, F., Bessani, A. N. e Fraga, J. da S.: A Semi-reliable *Multicast* Protocol for Distributed Multimedia Applications in Large Scale Networks. 8th Int. Conference on Management of Multimedia Networks and Services. Proceedings of MMNS 2005. LNCS vol. 3754. pp. 109-120. Barcelona, Espanha. (2005).
[7] Floyd, S., Jacobson, V., McCanne, S., Ching-Gung L. A Reliable *Multicast* Framework for Light-weight Sessions and Application Level Framing, SIGCOMM, Agosto (1995).
[8] Xu, R. X., Myers, A., Zhang, H., Yavatkar, R.: Resilient *multicast* support for continuous-media applications. International Workshop on Network and Operating System Support for Digital Audio and Video, Maio (1997).
[9]Wong, T, Henderson, T., Raman, S., Costello, A., Katz, R.: PolicyBased Tunable Reliable *Multicast* for Periodic Information Dissemination. Workshop on Satellite Based Information Services, Dallas, TX, Outubro (1998).
[10] Schneyer, S., Garcia, J., Brunstrom, A., Asplund, K.: PRTP: A Partially Reliable Transport Protocol for Multimedia Applications, Symposium on Intelligent Multimedia and Distance Education (ISIMADE), Baden-Baden, Alemanha, Agosto (1999).
[11] T. Speakman, J. Crowcroft, J. Gemmell, D. Farinacci, S. Lin, D. Leshchiner, M. Luby, T. Montgomery, L. Rizzo, A. Tweedly, N. Bhaskar, R. Edmonstone, R. Sumanasekera, L. Vicisano.: PGM Reliable Transport Protocol Specification, RFC 3208, Dezembro (2001).
[12] Striegel, A., Manimaran, G.: A survey of QoS *multicast*ing issues. IEEE Communications Magazine, 40(6):82–87, Junho, (2002).