ReMIOP: Projeto e Implementação de um Mecanismo de Difusão Confiável no CORBA

Alysson Neves Bessani[1*], **Lau Cheuk Lung**[2] , **Joni da Silva Fraga**[1]

[1]DAS - Departamento de Automação e Sistemas
UFSC - Universidade Federal de Santa Catarina
Campus Universitário, Caixa Postal 476 - CEP 88040-900 - Trindade - Florianópolis - SC

[2]PPGIA - Programa de Pós-Graduação em Informática Aplicada
PUC-PR - Pontifícia Universidade Católica do Paraná
R. Imaculada Conceição, 1155 - Prado velho - CEP 80215-901 - Curitiba -PR

neves@das.ufsc.br, lau@ppgia.pucpr.br, fraga@das.ufsc.br

Abstract. *OMG published a specification of a unreliable multicast mechanism for distributed applications developed in CORBA (UMIOP). However, many fault-tolerant or groupware applications demand for more restrictive warranties of agreement and ordering (for instance, reliable multicast with FIFO, causal or total ordering) of the available supports for group communication. OMG still does not have available a set of specifications to support those requirements. This paper presents an important contribution in that direction. We proposed the ReMIOP, an extension to the UMIOP protocol, for the conception of a reliable multicast mechanism in CORBA middleware. Performance measures comparing ReMIOP, UMIOP and UDP sockets for IP multicast communication is presented in order to evidence the costs for adding reliable and unreliable multicast in the middleware level.*

Resumo. *A OMG publicou uma especificação de um mecanismo de difusão não confiável para aplicações distribuídas desenvolvidas em CORBA (UMIOP). No entanto, muitas aplicações tolerante a faltas ou aplicações groupware exigem garantias mais restritivas de acordo e ordenação (por exemplo, difusão confiável com ordenação FIFO, causal, total, etc) dos suportes de comunicação de grupo disponíveis. A OMG ainda não tem disponível um conjunto de especificações que atenda a esses requisitos. Este artigo apresenta uma importante contribuição nesse sentido. É proposto o ReMIOP, uma extensão ao protocolo UMIOP, para a concepção de um mecanismo de difusão confiável em middleware CORBA. Medidas de desempenho comparando o ReMIOP, UMIOP e o uso de sockets UDP para comunicação multicast IP são apresentadas no sentido de evidenciar os custos da inclusão de difusão confiável e não confiável em nível de middleware CORBA.*

---

[*]Bolsista CNPq.

1. Introdução

Quando a arquitetura CORBA (*Common Object Request Broker)* foi introduzida pela OMG (*Object Management Group* [OMG, 2001a]) apenas comunicações ponto a ponto (usando interfaces de invocação estática ou dinâmica) eram disponíveis através do ORB (*Object Request Broker*). As mensagens que trafegam através desse canal seguem uma sintaxe de transferência própria definida pelo GIOP (*General Inter-ORB Protocol*). Esta sintaxe torna as mensagens envolvidas nas comunicações independentes das implementações de ORBs e das conseqüências de um ambiente heterogêneo. O mapeamento do GIOP sobre a camada de transporte TCP/IP é concretizado através do protocolo IIOP (*Internet Inter-ORB Protocol*). A combinação IIOP com TCP/IP corresponde a uma boa solução para comunicações entre objetos distribuídos segundo o modelo cliente/servidor, abordando aspectos como controle de erro, ordenação FIFO, etc.

Comunicações ponto a ponto têm se mostrado, no geral, bastante efetivas em aplicações distribuídas suportadas pelo CORBA. No entanto, muitas dessas aplicações teriam um melhor desempenho em termos de tempo e complexidade de mensagens se tivessem ao seu dispor mecanismos de comunicação multiponto. Usualmente, essas aplicações dependem de abstrações como grupos de objetos ou da necessidade da disseminação de dados sobre vários *hosts* da rede. Portanto, aplicações orientadas a grupo poderiam usufruir melhor os serviços de baixo nível de uma rede.

Para tentar suprir a necessidade de comunicações multiponto em nível de *middleware* CORBA, a OMG publicou as especificações UMIOP (*Unreliable Multicast Inter-ORB Protocol*) [OMG, 2001b]. O UMIOP corresponde a um conjunto de especificações de um serviço de difusão não confiável para ser incluído como parte do ORB. O protocolo definido nestas especificações, o MIOP (*Multicast Inter-ORB Protocol*), é responsável pelo mapeamento do GIOP sobre a pilha UDP/*multicast* IP. O *multicast* IP compreende um conjunto de extensões ao protocolo IP que o habilita na concretização de comunicações multiponto [Deering, 1986]. Este protocolo se caracteriza pela ausência de garantias e pelo alto desempenho, especialmente em redes locais. Várias são as aplicações que utilizam *multicast* IP, principalmente em sistemas de difusão multimídia na Internet.

A difusão não confiável do UMIOP, o modelo menos restritivo de comunicação de grupo, pode ser empregado para algumas aplicações distribuídas, por exemplo, em vídeo conferência, em que a perda de alguns quadros pode não representar a degradação (ou perda) da informação transmitida. No entanto, aplicações tolerante a faltas, aplicações *groupware*, entre outras, na maioria dos casos, exigem garantias mais restritivas de acordo e ordenação (por exemplo, difusão confiável, ordenação FIFO, causal, total, etc) dos suportes de comunicação de grupo disponíveis. A OMG ainda não tem disponível um conjunto de especificações que atenda a esses requisitos. Este problema está sendo tratado pela OMG em etapas. O primeiro passo, portanto, foi a publicação das especificações UMIOP. Esperamos que esta iniciativa motive a publicação de outras RFPs (*Request For Proposal*) por parte da OMG, no sentido da especificação de suportes a comunicação de grupo que forneçam as garantias acima citadas.

O projeto de integração do UMIOP em um ORB e a sua implementação foi apresentado em [Bessani et al., 2002]. Dando continuidade a este trabalho, o presente artigo apresenta nossas contribuições no sentido da concepção de um suporte de difusão confiável (*Reliable Multicast*) no ORB, baseado nas especificações UMIOP. O modelo

proposto, chamado de ReMIOP, está em conformidade com as especificações CORBA e UMIOP da OMG – as extensões necessárias para assegurar difusão confiável não alteram as interfaces dessas especificações. A inclusão de um protocolo de difusão confiável acima da camada MIOP é implementada usando a técnica de *plugins*. No sentido de avaliar essa proposta, foram realizados vários experimentos e medidas envolvendo aspectos de desempenho com nossa implementação. Medidas de desempenho do ReMIOP (a pilha ReMIOP/MIOP/UDP/*multicast* IP), UMIOP (a pilha MIOP/UDP/*multicast* IP) e o uso de sockets UDP para comunicação *multicast* IP (a pilha UDP/*multicast* IP) são apresentadas no sentido de evidenciar os custos da inclusão de difusão confiável e não confiável em nível de *middleware* CORBA.

Este trabalho é parte do projeto GROUPPAC (*http://www.lcmi.ufsc.br/grouppac*) que corresponde a um conjunto de objetos de serviço desenvolvidos com a finalidade de facilitar a implementação de aplicações distribuídas tolerantes a faltas. No GROUPPAC devem conviver as duas pilhas de protocolos IIOP/TCP/IP e ReMIOP/MIOP/UDP/*multicast* IP possibilitando um mais amplo espectro de suportes de protocolos aos modelos de comunicação de objetos distribuídos, conforme a figura 2.

O artigo apresenta na seção 2 as iniciativas da OMG de introdução de comunicação de grupo no CORBA. O MJACO é apresentado na seqüência, na seção 3. Na seção 4, é introduzido o modelo ReMIOP, a extensão ao MIOP para difusão confiável no CORBA. Detalhes de implementação do nosso serviço de difusão confiável e da integração do ReMIOP são descritos na seção 5. Na seção 6, são apresentados e discutidos alguns experimentos com nosso ORB *multicast*. A seção 7 cita alguns trabalhos relacionados e, finalmente, na seção 8, são levantadas as conclusões finais e as perspectivas futuras deste trabalho.

2. Comunicação de Grupo no CORBA

Em relação à introdução de mecanismos de comunicação de grupo no CORBA existem duas iniciativas significativas na OMG, a primeira delas utiliza a noção de grupo e permite o uso de um suporte de comunicação de grupo proprietário como meio para se implementar aplicações tolerante a faltas em nível de objetos (FT-CORBA [OMG, 2000]), a outra propõe a utilização do ORB como um mecanismo de comunicação de grupo de alto desempenho sem garantias de confiabilidade (UMIOP [OMG, 2001a]). Desta forma, estas duas especificações podem ser complementares, e indicam uma tendência, dentro da OMG, de se tentar montar um mecanismo de comunicação de grupo padronizado que ofereça níveis diferenciados de garantia para diferentes aplicações.

O padrão FT-CORBA, que introduziu o conceito de grupo de objetos na arquitetura CORBA, define um conjunto de objetos de serviço que oferecem funcionalidades como gerência de grupos, transferência de estado, detecção e notificação de falhas, entre outros. Um tipo de suporte que o FT-CORBA utiliza, porém não padronizado pela OMG, é o serviço de comunicação de grupo. As especificações definem que este serviço quando incluíndo certas propriedades de comunicação [Hadzilacos and Toueg, 1994] forma as bases necessárias na implementação de replicação ativa [Schneider, 1990]; entretanto, estas mesmas especificações não definem as semânticas deste suporte e muito menos os protocolos que ele deve implementar.

As especificações UMIOP, por outro lado, definem um serviço de comunicação de grupo não confiável baseado em *multicast* IP, podendo ser consideradas como um primeiro passo em direção a criação de um mecanismo de comunicação de grupo interoperável e padronizado pela OMG. Extensões nessas especificações, que implemente propriedades mais fortes de acordo e ordenação, seriam bastante adequadas ao FT-CORBA.

2.1. UMIOP

Em 1999 a OMG iniciou o processo de especificação de um protocolo de difusão não confiável baseado em *multicast* IP e um modelo de grupo de objetos que desse suporte a este protocolo em ORBs CORBA. Este processo culminou com o lançamento das especificações UMIOP. O objetivo do padrão UMIOP é fornecer um mecanismo de comunicação multiponto sem garantias de entrega dentro da arquitetura CORBA.

O protocolo utilizado pelo UMIOP é o MIOP. Este protocolo mapeia mensagens GIOP para UDP/*multicast* IP. A função básica do protocolo MIOP é segmentar e encapsular as mensagens GIOP enviadas a grupos em vários pacotes (coleções). Estes pacotes contém um cabeçalho, definido nas especificações, que contém uma série de campos que permitem a remontagem da mensagem original nos receptores.

Uma vez os pacotes devidamente arranjados a difusão da mensagem no grupo é feita através do protocolo de transporte UDP, que fornece uma interface quase direta para os serviços IP, ou neste caso, *multicast* IP. O *multicast* IP define um conjunto de extensões ao protocolo IP que permitem a este realizar comunicações de um para muitos (difusão seletiva). As principais características deste protocolo são grupos abertos (não é necessário ser membro do grupo para difundir uma mensagem neste), sem *membership* (lista de membros), sem confiabilidade (assim como o IP) e acessíveis via endereços IP classe D (de `224.0.0.0` a `239.255.255.255`).

Utilizando o MIOP, e o *multicast* IP num nível inferior, é possível realizar a transferência de mensagens GIOP entre instâncias de ORBs. Entretanto, o modelo de objetos convencional do CORBA, que especifica que uma referência de objeto deve corresponder a uma única implementação do objeto, não é adequado para dar suporte a grupos. Além disso, a semântica de invocação ponto a ponto do CORBA é confiável em relação a entrega de mensagens e ordem definida pelo emissor que podem ser definidas com ou sem espera de resposta, ao contrário do que define o MIOP. Portanto, um novo modelo de objetos, que representasse grupos, teve de ser definido no UMIOP. Este modelo não define um identificador de objetos, mas sim um identificador de grupo que pode ser associado a múltiplos identificadores de objetos, que são utilizados pelo POA (*Portable Object Adapter*) para a ativação das implementações correspondentes [OMG, 2001b]. A semântica de entrega e ordenação de mensagens no UMIOP é sem garantias, só são suportadas via MIOP, requisições sem resposta.

Um grupo de objetos no UMIOP consiste em informações de identificação do grupo (um identificador único para este) e informações sobre como acessá-lo na rede de comunicações (endereço IP classe D e uma porta). Estas informações ficam contidas na referência de grupo UMIOP, detalhada a seguir.

2.2. Referência de Grupo UMIOP

Uma referência, ou IOR (*Interoperable Object Reference*), serve como identificação única de um objeto (implementação) no CORBA. Cada IOR contém um ou mais perfis que permitem ao ORB acessar a implementação do objeto através de algum mecanismo de transporte. Por exemplo, perfis IIOP, que nos seus campos definem o endereço da implementação (tipicamente um IP e porta) e um identificador único do objeto no ORB servidor (chave de objeto), são utilizados para acessar implementações via TCP/IP.

Para dar suporte a grupos as especificações UMIOP definem uma IOR de grupo que permite endereçar um grupo de zero ou mais objetos. Uma IOR de grupo utiliza um tipo de perfil diferente do perfil IIOP para difundir mensagens via UDP/*multicast* IP. Este perfil, definido na especificação como perfil UIPMC, contém todas as informações necessárias para o acesso ao grupo (endereço IP classe D e porta) em nível de transporte. Uma outra estrutura, com a identificação lógica do grupo, chamada `TagGroupComponent`, é utilizada para identificação de objetos membros em nível de ORB. Além destes componentes, a IOR de grupo, pode também conter dois perfis IIOP, um para o encaminhamento de requisições que exijam resposta e outro com um *gateway*, que pode realizar a difusão de mensagens no grupo quando o cliente não for capaz de fazê-lo.

A figura 1 apresenta o formato completo da IOR de grupo, definida pela OMG como parte das especificações UMIOP. A criação de uma IOR de grupo deve ser feita através da especificação de informações sobre o grupo (para o preenchimento das estruturas `UIPMC_ProfileBody` e `TagGroupComponent`) e das IORs com o perfil IIOP de grupo e do *gateway*. Esta criação é feita através da especificação destas informações em uma URL ”corbaloc”, ou através de métodos do MGM (*Multicast Group Manager*), um objeto de serviço opcional das especificações UMIOP que disponibiliza uma série de operações para o gerenciamento de grupos.

3. MJACO

Partindo do estudo das especificações do UMIOP realizamos um esforço de implementação para o desenvolvimento de um ORB que atendesse estas especificações. Este esforço culminou no MJACO [Bessani et al., 2002], uma extensão do *JacORB*, um ORB CORBA de alto desempenho e de código aberto que implementa as especificações CORBA 2.3 (*http://www.jacorb.org*). A arquitetura do MJACO foi definida de tal forma a permitir o convívio de duas pilhas de protocolos (IIOP/TCP/IP e MIOP/UDP/*multicast* IP) no mesmo ORB, contribuindo, deste modo, para uma melhor interoperabilidade e portabilidade.

A figura 2 apresenta a arquitetura de integração do UMIOP ao ORB implementada no MJACO. Nesta figura temos o ORB com as duas pilhas de protocolos: uma para a comunicação ponto a ponto, baseada em IIOP, utilizando os serviços do TCP/IP, e outra para comunicação multiponto formada pelo MIOP, que utiliza UDP/*multicast* IP como mecanismo de transferência de seus pacotes. O nosso modelo de integração apresenta os vários elementos definidos na especificação que compõem o suporte para os dois modelos de comunicação. Foram adicionados ainda outros componentes e extensões, não definidas

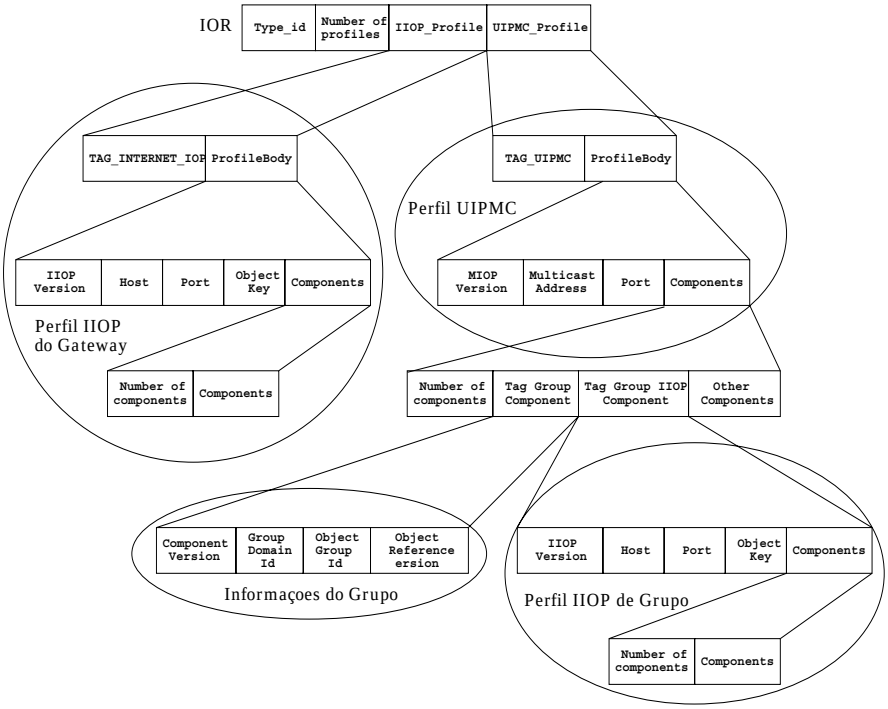

Figura 1: IOR de grupo *multicast*.

na especificação, cuja finalidade é facilitar o convívio das diferentes pilhas e melhorar a eficiência de utilização do conjunto.

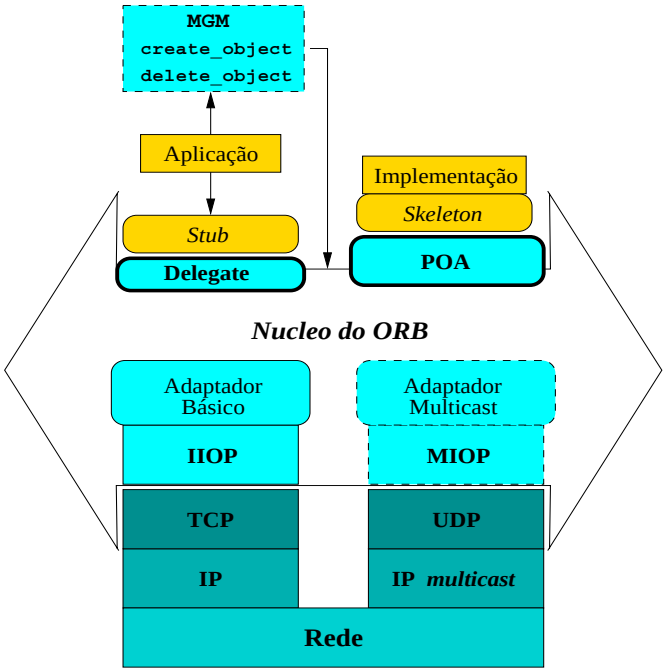

Figura 2: Arquitetura do MJACO.

Um componente fundamental que faz parte do nosso modelo de integração é o Adaptador *Multicast*, responsável pela gestão dos *sockets multicast* utilizados na recepção

de pacotes MIOP e pela entrega de mensagens endereçadas a grupos aos POAs ativos no ORB. O módulo UMIOP cumpre as tarefas, previstas nas propostas de padrão, no que se refere à tradução das mensagens GIOP em coleções de pacotes MIOP e vice-versa.

O POA e o *Delegate* são os principais componentes do ORB a serem alterados para a introdução do UMIOP. A alteração do *Delegate* se dá em alguns pontos para dar suporte ao envio de mensagens GIOP para grupos, já que ele é o primeiro componente interno ao ORB a ser ativado quando ocorre uma chamada de método no *stub*. Em nossa abordagem, é nele que é escolhida qual das pilhas de protocolos será utilizada para envio de uma mensagem GIOP correspondente ao método chamado. O POA, por sua vez, além do acréscimo de quatro primitivas para a manipulação de grupos de objetos descritas nas especificações da OMG, deve ser alterado para o processamento de requisições endereçadas a grupos, realizando buscas em uma tabela de grupos ativos para obtenção das implementações membros do grupo para o qual a mensagem está endereçada.

4. ReMIOP - Reliable MIOP

Como mencionado anteriormente, o MIOP, protocolo definido nas especificações UMIOP, é não confiável, portanto inadequado para utilização em diversos tipos de aplicações, em especial em serviços de replicação para tolerância a faltas, que não admitem perda de mensagens. Desta forma, propomos um conjunto de extensões ao MIOP, chamada de ReMIOP, para que este apresente propriedades de confiabilidade (garantia de entrega de mensagem). O ReMIOP é um protocolo de difusão confiável escalável iniciado pelo receptor (usa pedidos de retransmissão - NACKs) [Levine and Garcia-Luna-Aceves, 1998, Pingali et al., 1994] nos moldes de protocolos como o SRM [Floyd et al., 1997], LRMP [Liao, 1998] e TRM [Sabata et al., 1996].

As premissas que embasam as propriedades de confiabilidade do ReMIOP levam em consideração um suporte de comunicação com características de sistemas assíncronos, portanto, sem garantias de limites de tempo na realização de transferências de mensagens ou na execução de operações remotas. Quanto ao modelo de faltas são admitidas somente faltas de parada nos *hosts* e faltas de omissão no sistema de comunicação. Uma outra premissa fundamental para a consistência do ReMIOP é a que define que após $Od + 1$ difusões de uma mesma mensagem, nenhum *host* correto fica sem receber o pacote (ver seção 4.1).

De um modo geral, o ReMIOP opera da seguinte maneira: as mensagens (pacotes MIOP) são difundidas pelo emissor no grupo sem saber quais são seus membros. Os receptores detectam perdas de pacotes quando encontram lacunas na seqüência das mensagens recebidas. Quando um membro detecta a falta de um pacote, o mesmo difunde uma mensagem de controle (NACK) no grupo pedindo a retransmissão. Quem receber esta mensagem, seja o emissor ou algum receptor que tenha o pacote requerido, o difunde novamente no grupo para que os que não o receberam tenham a possibilidade de recebê-lo. Este protocolo inclui ainda mensagens de controle difundidas periodicamente no grupo pelos receptores para reportar o estado de seus *buffers* aos emissores, a fim de que estes possam ajustar sua taxa de envio através do algoritmo de controle de fluxo.

O algoritmo apresentado na figura 3 descreve, de maneira simplificada, os procedimentos executados para a recepção e o envio de mensagens via ReMIOP.

```
{To R − multicast(m)}
T_d ← calculate_delay() {controle de fluxo no emissor}
schedule_multicast(T_d, m) {envio da mensagem é agendado}

{To R − receive(m)}
U − receive(m)
if m.type = DATA then {m.type: tipo da mensagem m}
   if search_in_buffer(m.sender, m) = NULL then {m.sender: emissor de m}
      add_to_buffer(m.sender, m)
      R − deliver(m)
      if there are missing messages then
         wait(random(T_nack)) {supressão de NACKs}
         R − multicast(NACK_m)
      end if
   end if
else if m.type = NACK then
   cancel_scheduled(m) {cancela envio de m, se agendado}
   for all m_n ∈ m.nacked do {m.nacked: lista de mensagens requeridas}
      m_r ← search_in_buffer(m_n.sender, m_n)
      if m_r ≠ NULL ∧ nacks_{m_r} ≤ Od then {nacks_m é o número de nacks para m}
         wait(random(T_repair)) {supressão de reparos}
         R − multicast(m_r)
         nacks_{m_r} ← nacks_{m_r} + 1
      end if
   end for
else if m.type = STATE then
   update_send_rate(m.states)
end if
```

Figura 3: Algoritmo ReMIOP simplificado.

O envio de mensagens, representado pela operação $R - multicast(m)$ no algoritmo da figura 3, é realizado em dois passos: o cálculo do tempo estimado de espera para o envio da mensagem a fim de que a taxa de envio corrente, que segue o algoritmo de controle de fluxo, seja respeitada; e o agendamento da difusão (via *multicast* não confiável) a partir deste tempo estimado.

O procedimento para a recepção e entrega de mensagens, indicado na figura 3 pela operação $R - receive(m)$, se inicia com a recepção não confiável de uma mensagem e logo em seguida trata de maneira diferenciada cada um dos três tipos de mensagens definidas para o ReMIOP:

1. Se a mensagem for de dados a primeira tarefa é verificar se esta já foi recebida (já está no *buffer*); neste caso nada é feito. Se a mensagem está sendo recebida pela primeira vez, a mesma é incluída no *buffer* de recepção e então entregue à aplicação. Depois disso é verificado se existem mensagens faltando, pelo controle de numeração por emissores, e NACKs são enviados para as suas recuperações;
2. Se a mensagem recebida for um NACK, então este é cancelado (mecanismo de su-

pressão de NACKs) e para cada mensagem requisitada no NACK é agendada uma mensagem de reparo, se o receptor tiver como repara-la. O envio desta mensagem de reparo é atrasado por um tempo aleatório, porém limitado, para que não exista uma explosão de mensagens de reparos no grupo;

3. Finalmente, se a mensagem for uma notificação de estado de algum receptor do grupo (estado dos *buffers* de recepção), o sistema leva em consideração este estado e atualiza a taxa de transferência do protocolo.

Os mecanimos utilizados para agregar confiabilidade ao protocolo apresentado são detalhados a seguir.

4.1. Retransmissões

Como a possibilidade de perda de mensagens existe e é considerável, principalmente em sistemas de larga escala, o ReMIOP inclui um tipo de mensagem de controle para permitir o pedido de retransmissões. Esta mensagem contém o identificador dos pacotes MIOP extraviados, assim, quem a receber, se tiver os pacotes pedidos, pode difundi-los novamente no grupo. Além deste mecanismo, que caracteriza um protocolo iniciado pelo receptor, algumas otimizações foram adicionadas ao protocolo a fim de evitar, na medida do possível, as inundações de NACKs e retransmissões. Dentre estas modificações podemos citar a utilização de um mecanismo RINA (*Receiver Initiated Nack Avoidance*)[Pingali et al., 1994] e a postergação de retransmissões. Estas duas otimizações postergam a difusão de NACKs e retransmissões, respectivamente, por intervalos de tempo aleatórios esperando que algum outro membro do grupo faça essa difusão.

Além disso, como uma otimização opcional, foi introduzido o parâmentro grau de omissão (Od - *Omission Degree*). Neste caso, o emissor e os receptores que forem recebendo o pacote podem retransmitir este até o limite $Od + 1$. Em canais com faltas de omissão é admissível considerar que não mais que Od retransmissões de um mesmo pacote seja perdido em um intervalo de tempo de referência. Testes podem ser feitos em redes concretas para determinar Od com qualquer grau de probabilidade desejado [Veríssimo et al., 1997]. Se um receptor não recebe um pacote após $Od + 1$ retransmissões de um emissor, com Od computado considerando somente faltas acidentais, então é possível assumir que o receptor é faltoso (*crash*). Note que Od é apenas um parâmetro que pode ser utilizado no protocolo. Se Od é um valor muito alto, então o protocolo executará tal como aqueles protocolos que assumem canais confiáveis [Hadzilacos and Toueg, 1994].

4.2. Controle de Fluxo

O controle de fluxo é um ponto fundamental em qualquer mecanismo de difusão confiável. Através deste mecanismo é possível evitar perdas de pacotes nos *hosts* e a decorrente explosão de NACKs. O mecanismo de controle de fluxo implementado no ReMIOP baseia-se no modelo proposto para o protocolo LRMP [Liao, 1998].

O mecanismo empregado para realizar o controle de fluxo no ReMIOP utiliza mensagens de estado, que contém o estado dos *buffers* de recepção dos membros dos grupos[1]. Através destas informações o emissor recebe dados a respeito da ”velocidade”de

[1]Cada receptor tem um *buffer* de recepção para cada emissor.

seus receptores, e usa uma função de atualização da taxa de transmissão de acordo com a capacidade dos receptores. Este mecanismo se utiliza de dois tipos de *buffers*: um para emissores e outro para receptores.

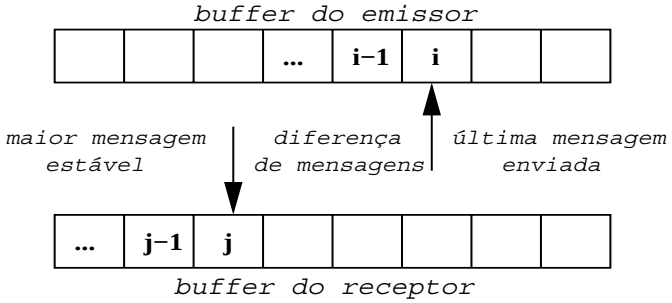

Figura 4: O controle de fluxo no ReMIOP.

Na figura 4 o *buffer* do emissor, também chamado *buffer* de envio, com tamanho fixo, é utilizado para armazenar as mensagens enviadas e por enviar. O tamanho deste *buffer* define quantas mensagens antigas podem ser reenviadas por ele em caso de pedidos de retransmissão. No receptor, o *buffer* serve para armazenar as mensagens recebidas. A diferença entre o número de seqüência da última mensagem enviada pelo emissor (i no *buffer* de envio), e a maior mensagem estável (j no *buffer* de recepção) é o parâmetro utilizado para ajustar a taxa de envio das mensagens.

Seja a diferença $\delta = i - j$ de tal modo que: quanto maior δ, menor deve ser a taxa de envio para que os receptores lentos (cujo j é muito menor que i) consigam "consumir"corretamente as mensagens do emissor. O principal objetivo deste algoritmo é evitar retransmissões ajustando a taxa de envio de tal forma que todos os membros do grupo, mesmo os que estão em áreas congestionadas da rede possam receber as mensagens.

A taxa de transmissão dos emissores varia sempre entre os valores $[R_{min}, R_{max}]$. Onde R_{min} e R_{max} são definidos pela aplicação. A taxa de transmissão inicial é definida como $R = (R_{min} + R_{max})/2$, e a cada 8 envios de pacotes R é incrementado em $0.125R$. A cada recepção de uma mensagem de estado, o emissor ajusta a taxa de transmissão (R) de acordo com a seguinte regra (onde $size$ é o tamanho do *buffer* do emissor, i.e. sua capacidade máxima):

$$new\_rate(R, \delta, size) = \begin{cases} R & \text{se } \delta \leq size/5 \\ 0.75R & \text{se } size/5 < \delta \leq size/4 \\ 0.50R & \text{se } size/4 < \delta \leq size/3 \\ 0.25R & \text{se } size/3 < \delta \end{cases} \quad (1)$$

Na equação 1 [Liao, 1998], a função new_rate define uma nova taxa de envio baseando-se na taxa atual (R), no δ e no tamanho do *buffer* ($size$). A regra de ajuste de fluxo apresentada mostra que a taxa de envio em um emissor é sempre definida pelo receptor mais lento (aquele para o qual δ é maior).

Através desse algoritmo de controle do fluxo e da premissa (informal) de que o sistema opera a maior parte do tempo em condições normais (sem congestionamentos e falhas de omissão), é possível garantir que todas as mensagens enviadas chegarão aos

membros do grupo.

4.3. Integração do ReMIOP ao MIOP

Os pacotes de dados enviadas com o ReMIOP são extamente iguais aos difundidos via MIOP, portanto, dados enviados por emissores que utilizam ReMIOP podem ser recebidos por receptores MIOP sem o menor problema.

As mensagens de controle ReMIOP são encapsuladas também em pacotes MIOP e difundidas normalmente no grupo *multicast* IP da mesma forma que os pacotes de dados. Entretanto, receptores MIOP não tem como entendê-las, e devem ignorá-las, para tanto, alguns cuidados básicos são tomados no preenchimento dos campos do cabeçalho MIOP destes pacotes para permitir que eles sejam descartados.

O preenchimento do cabeçalho dos pacotes MIOP que contém mensagens de controle ReMIOP segue as seguintes regras: *(i)* o identificador de mensagens deve sempre ter o mesmo valor, e este valor não deve ser utilizado por mensagens de dados; *(ii)* o campo com o número do pacote dentro da mensagem contém sempre o valor 0; *(iii)* o campo que define a quantidade de pacotes que formam esta mensagem contém sempre o valor 2; e *(iv)* é marcado um bit no campo `flags` para indicar que o pacote em questão é uma mensagem de controle ReMIOP. Os demais campos do pacote são preenchidos de maneira convencional de acordo com os dados enviados e a especificação do protocolo.

Definindo os campos do cabeçalho MIOP da forma especificada os receptores de pacotes ReMIOP que tiverem condição de processar estas mensagens perceberão através do campo `flags` a natureza do pacote e os tratarão adequadamente (como um pacote de controle ReMIOP). Os receptores que não implementam o ReMIOP entenderão o pacote como o primeiro elemento de uma coleção de pacotes de tamanho 2. Como o segundo pacote dessa coleção nunca vai chegar, ele nunca vai ser liberado para as camadas superiores do ORB, e será descartado por ocasião de seu *timeout*, de acordo com as especificações do MIOP [OMG, 2001b].

5. Implementação - Estratégias Plugáveis no MJACO

A fim de disponibilizar os serviços de difusão confiável fornecidos pelo ReMIOP no MJACO, foi implementado um mecanismo de *plugins* que permite a integração de estratégias de confiabilidade que podem extender a pilha *multicast* do ORB. Através deste mecanismo, vários protocolos de características diversas podem ser implementados sobre o serviço de difusão não confiável. A figura 5 apresenta a arquitetura do mecanismo.

Na figura 5 temos a estratégia de confiabilidade carregada como uma camada da pilha *multicast*. A camada que pode ser ”plugada”fica entre as duas camadas (segmentação e MIOP) que caracterizam a implementação do MIOP. Num nível mais baixo temos a camada MIOP que encapsula blocos de dados em pacotes MIOP e os envia via UDP/*multicast* IP, já acima do *plugin* temos a camada de segmentação/desegmentação de mensagens, esta camada é responsável por desmontar mensagens longas em coleções de pacotes MIOP e remonta-las nos receptores.

O mecanismo de *plugin* apresentado nada mais é do que uma implementação do padrão de projeto *Strategy*. Este padrão permite que o comportamento, método na terminologia da orientação a objetos, de um componente, ou objeto, seja trocado, através da

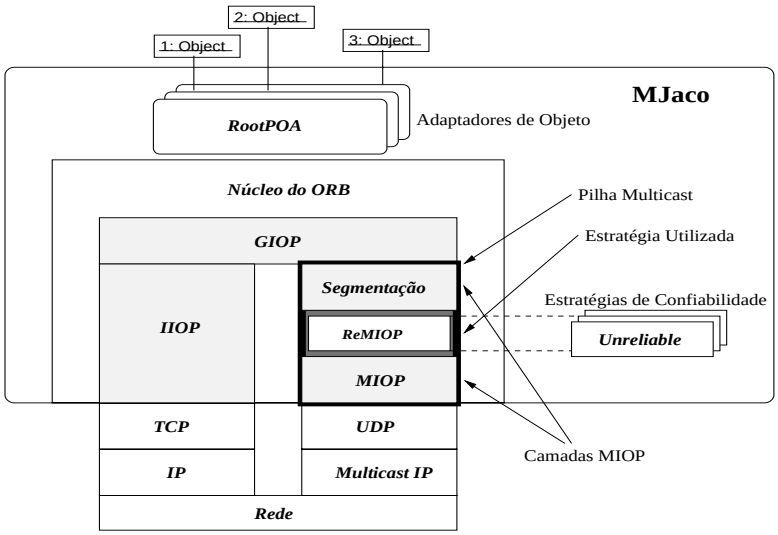

Figura 5: Arquitetura do MJACO com as Estratégias Variáveis.

substituição deste (o comportamento) que fica encapsulado em um objeto (a estratégia). No caso do MJACO, temos o componente pilha *multicast* (*Multicast Stack*) que utiliza como mecanismo para garantir confiabilidade de comunicação uma estratégia, encapsulada em um objeto, que pode ser trocada, inclusive em tempo de execução, sem a necessidade de se alterar o código do ORB.

6. Resultados Obtidos

A fim de verificar o desempenho de nosso serviço de difusão confiável em ORB CORBA, realizamos uma série de testes comparativos da utilização do MJACO usando a estratégia ReMIOP, com o protocolo MIOP e usando *sockets multicast*. Estes testes foram realizados em 4 máquinas de perfis idênticos[2] ligadas a um mesmo HUB. Desta forma, objetivou-se testar o desempenho da implementação MJACO+ReMIOP, portanto não foram consideradas arquiteturas de rede mais complexas como as encontradas na Internet. Testes mais elaborados serão realizados em trabalhos futuros.

O programa de teste utilizado mede o tempo necessário para um membro do grupo difundir uma mensagem de tamanho variável no mesmo e receber confirmação de recebimento de todos os membros do grupo (inclusive dele mesmo), chamado tempo de *round trip*.

Foram realizados vários experimentos utilizando-se valores diferentes para R_{max} (ver seção 4.3), que define a taxa de envio máxima em bits por segundo em que o ORB difunde mensagens em um grupo. Quanto maior o R_{max} utilizado, mais rápidas são as difusões, e maiores são as possibilidade de haver perdas de mensagens tanto de dados quanto de controle. Foram realizados também experimentos envolvendo o MJACO usando o MIOP puro, que pode ser considerado como o ReMIOP com $R_{max} = \infty$, e com *sockets multicast*. O grafico da figura 6 apresenta o resultado destes experimentos em uma escala logarítmica.

[2]Pentium IV 1.6GHz, 256 Mbytes de memória RAM, sistema operaciona Mandrake Linux 9 (kernel 2.4) e placas de rede Ethernet 100Mbps.

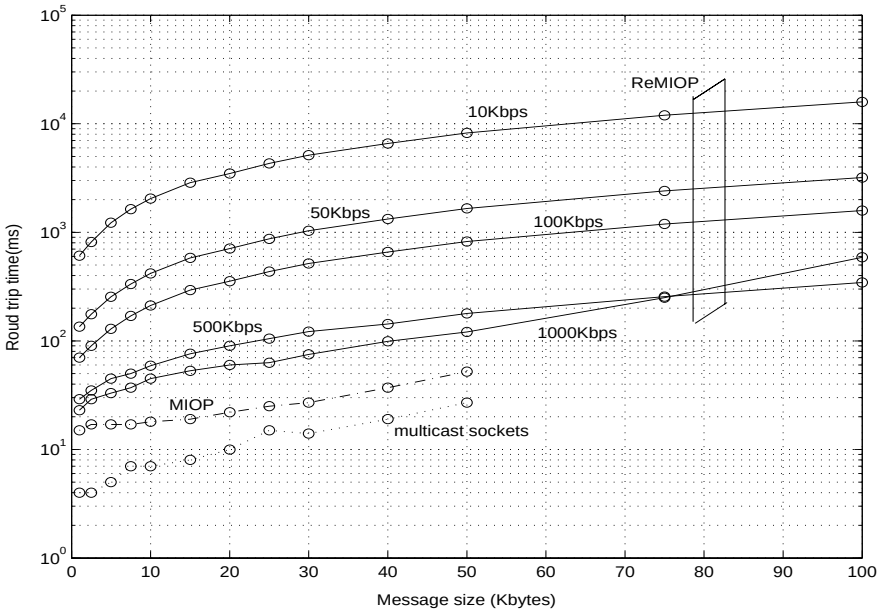

Figura 6: Desempenho do MJACO.

Pela figura 6 é possível verificar que quanto maior o valor de R_{max} mais o comportamento do ReMIOP se aproxima do MIOP puro. A curva com menor valor de R_{max} ($10Kbps$), é extremamente confiável (e lento) a tal ponto que, em nossos testes, nenhum pacote foi perdido, e por conseguinte, nenhum NACK foi difundido no grupo, entretanto o tempo necessário para a realização de um *round trip* chega a ser quase 3 casas decimais de magnitude maior do que o tempo necessário utilizando MIOP. Em curvas com valores maiores de R_{max}, o tempo de *round trip* já é bem menor, e a quantidade de pacotes perdidos, NACKs e retransmissões é bem maior, chegando ao ponto de, na curva com $R_{max} = 1000Kbps$, termos o tempo de *round trip* igual ou maior que a curva de $R_{max} = 500Kbps$ para mensagens grandes. Este resultado se deve a quantidade de pacotes perdidos (inclusive NACKs) e retransmissões empregadas.

O gráfico da figura 6 também apresenta curvas, a título de referência, com o tempo de *round trip* para o MIOP puro (MJACO com *plug-in* `Unreliable`) e para *sockets multicast* (implementado sem o MJACO). Estas curvas vão até mensagens de 50Kbytes, pois para mensagens maiores que estas as perdas de pacotes impossibilitam a utilização de nosso programa de testes já que não é possível completar um *round-trip* sem que todas as mensagens sejam transmitidas corretamente.

7. Trabalhos Relacionados

Existem vários trabalhos na literatura que visam incorporar comunicação de grupo a arquitetura CORBA, entretanto, apenas recentemente, com a publicação das especificações UMIOP, abriu-se a possibilidade de se implementar mecanismos de comunicação de grupo interoperáveis baseados em *multicast* IP. Um trabalho semelhante ao nosso é o protocolo RMIOP [Picard et al., 2001] que também propõe uma extensão ao MIOP com características confiáveis. Este trabalho também utiliza uma política de envio de NACKs e, adicionalmente, faz uso de ACKs para confirmação de mensagens GIOP recebidas

(cada receptor envia um ACK ao emissor após receber todos os pacotes de uma coleção). Este tipo de política necessita de informações de *membership* pois o emissor tem que saber quem são os membros do grupo na hora de coletar os ACKs. O uso de *membership* (não detalhado pelos autores), neste caso, aumenta a complexidade do protocolo RMIOP e degrada o desempenho quando receptores são removidos ou adicionados dinamicamente no grupo. A nossa abordagem não necessita de *membership*, e em nossa concepção, este tipo de serviço deve ser implementado em nível de objetos, utilizando as facilidades oferecidas pelo FT-CORBA. Além disso, em [Picard et al., 2001] não é utilizado mecanismo de controle de fluxo. A implementação do RMIOP foi feita em C++ sobre o ORBacus 3.1 (*http://www.iona.com*), um ORB proprietário que provê um mecanismo de *plugins* bastante genérico para a integração de novos protocolos. Em nosso trabalho utilizamos um ORB de código aberto pela possibilidade irrestrita de se realizar extensões no mesmo.

Outro trabalho que integra difusão confiável em um ORB CORBA, realizado pelo mesmo grupo que idealizou o RMIOP, é descrito em [Gransart and Geib, 1999]. Neste trabalho a difusão confiável é implementada através da biblioteca LRMP integrada ao ORBacus como um *plugin*, portanto, não se trata de uma solução padronizada e interoperável. Mas vale lembrar que apesar de apresentar uma solução simples, este trabalho precede as especificações UMIOP, e nele são introduzidas várias idéias que foram posteriormente aproveitadas nestas especificações.

Na literatura existem vários protocolos de comunicação multiponto confiáveis com ”melhor esforço”[Levine and Garcia-Luna-Aceves, 1998], estes protocolos se caracterizam geralmente pela ausência de confirmações de mensagens e pela preocupação com a escalabilidade. Entre os mais influentes protocolos deste tipo podemos citar o SRM [Floyd et al., 1997], o TRM [Sabata et al., 1996] e o LRMP [Liao, 1998], este último tendo grande influência na concepção do ReMIOP. Apesar dos protocolos citados terem obtido moderado sucesso, o IETF (*Internet Engineering Task Force*) não adotou nenhum destes como protocolo de transporte multiponto confiável para a Internet, ao invés disso uma série de mecanismos que definem um protocolo de comunicação confiável estão sendo definidos para que aplicações diferentes, com requisitos diferentes, possam montar protocolos diversos a partir de mecanismos padronizados [Handley et al., 2000, Whetten et al., 2001].

A idéia de protocolos plugáveis em sistemas de comunicação de grupo e a configuração dinâmica de pilhas de protocolos com base em microprotocolos disponíveis já foi largamente utilizada na definição de garantias de serviço diferentes em suportes de comunicação de grupo. O sistema HORUS [van Renesse et al., 1996], utiliza camadas de microprotocolos para fornecer serviços com diferentes níveis de garantias em termos de confiabilidade e segurança, assim como o sistema Ensemble [van Renesse et al., 1998], uma evolução deste implementado em ML, utilizado também em pesquisas com formalização de protocolos e demonstração automática de propriedades. Um outro sistema tem se tornado um padrão de facto quando se trata de comunicação de grupo, principalmente na comunidade Java de software livre, é o Javagroups [JavaGroups, 2003]. Este sistema que começou como uma versão em Java do HORUS, atualmente já implementa um grande número de funcionalidades e vem sendo usado em vários projetos para a implementação de alta disponibilidade e balanceamento de carga.

8. Conclusão

O objetivo principal deste trabalho foi, a partir de um mecanismo de difusão não confiável padronizado, propor extensões para obter um protocolo de difusão confiável, mais adequado a implementação de garantias mais restritivas de acordo e ordenação, implementado como parte de um ORB CORBA. O modelo de integração, usando o mecanismo de *plugins*, é bastante flexível e não compromete aspectos de interoperabilidade e portabilidade do ORB como um todo. O ORB é capaz de realizar tanto invocações usando o ReMIOP, MIOP ou IIOP. Além disso, as abordagens de invocação estática e dinâmica no CORBA são mantidas mesmo com as extensões propostas – só não o retorno de invocação (resposta), tal como afirmado na especificação UMIOP.

A implementação do ReMIOP torna agora possível a concretização de outras soluções de comunicação de grupo na arquitetura CORBA. Estas soluções estão sendo aproveitadas no projeto GROUPPAC [Lung et al., 2000], que implementa a especificação *Fault-Tolerant* CORBA, no sentido da concepção de modelos de replicação ativa [Schneider, 1990], inexistentes nas atuais especificações da OMG. A própria OMG já está reunindo um grupo para trabalhar em uma RFP sobre difusão confiável no CORBA. Acreditamos que este serviço de multicast com garantias deva ser também implementado sobre o UMIOP.

Além disso, foram também apresentados neste artigo algumas medidas de desempenho da implementação ReMIOP comparando-a com o MIOP e com *sockets multicast* IP para verificar os custos dessa qualidade de serviço disponibilizado em nível de *middleware*. Estes desenvolvimentos, que tiveram como base o modelo de integração proposto, foram concretizados sobre o JacORB. Estas implementações podem ser obtidas na WEB no seguinte endereço (*http://grouppac.sourceforge.net/*).

9. Agradecimentos

Agradecimentos especiais ao CNPq pelo suporte financeiro (processo 551948/02-7) e aos revisores anônimos pelos comentários valiosos na melhora do artigo.

Referências

Bessani, A. N., da Silva Fraga, J., and Lung, L. C. (2002). Mjaco - integração do multicast ip na arquitetura corba. In *Anais do 20o. Simpósio Brasileiro de Redes de Computadores*, Buzios - RJ - Brasil.

Deering, S. E. (1986). Host extensions for ip multicasting (rfc 988). IETF Request For Comments.

Floyd, S., Jacobson, V., Liu, C.-G., McCane, S., and Zhang, L. (1997). A reliable multicast framework for light-weight session and application level framing. *IEEE/ACM Transactions on Networking*.

Gransart, C. and Geib, J.-M. (1999). Using an orb with multicast ip. In *Proceedings of PCS99: Parallel Computing Systems Conference*, Ensenada - Mexico.

Hadzilacos, V. and Toueg, S. (1994). A modular approach to the specification and implementation of fault-tolerant broadcasts. Technical report, Department of Computer Science, Cornell University, New York - USA.

Handley, M., Floyd, S., Whetten, B., Kermode, R., Vicisano, L., and Luby, M. (2000). The reliable multicast design for bulk data transfer (rfc 2887). IETF Request For Comments.

JavaGroups (2003). Javagroups: A toolkit for reliable multicast communication. http://www.javagroups.com.

Levine, B. N. and Garcia-Luna-Aceves, J. J. (1998). A comparison of reliable multicast protocols. *Multimedia Systems*, 6(5):334–348.

Liao, T. (1998). Light-weight reliable multicast protocol. Disponível em http://webcanal.inria.fr/lrmp/.

Lung, L. C., da Silva Fraga, J., Farines, J. M., and Oliveira, J. R. (2000). Experiências com comunicação de grupo nas especificações fault tolerant corba. In *Anais do 18o. Simpósio Brasileiro de Redes de Computadores*, Belo Horizonte - MG - Brasil.

OMG (2000). Fault-tolerant corba specification v1.0. OMG Standart.

OMG (2001a). The common object request broker architecture specification v2.6. OMG Standart.

OMG (2001b). Unreliable multicast inter-orb protocol specification v1.0. OMG Standart.

Picard, S. L. D., Degrande, S., and Gransart, C. (2001). A corba based platform as communication support for synchronous collaborative virtual environments. In *9th ACM Multimedia Conference*, Ottawa - Canada.

Pingali, S., Towsley, D., and Kurose, J. F. (1994). A comparison of sender-initiated and receiver-initiated reliable multicast protocols. In *Proceedings of the Sigmetrics Conference on Measurement and Modeling of Computer Systems*, pages 221–230, New York, NY, USA. ACM Press.

Sabata, B., Brown, M., Denny, B., and Heo, C. H. (1996). Transport protocol for reliable multicast: Trm. In *Proceedings of the International Conference on Networks*, Orlando - Flórida - USA.

Schneider, F. B. (1990). Implementing fault-tolerant service using the state machine aproach: A tutorial. *ACM Computing Surveys*, 22(4):299–319.

van Renesse, R., Birman, K. P., and Maffeis, S. (1996). Horus: A flexible group communication system. *Communications of the ACM*, 39(4):76–83.

van Renesse, R., Birman, K. P., Mark Hayden, A. V., and Karr, D. (1998). Building adaptative systems using ensemble. *Software - Pratice and Experience*, 28(9):963–979.

Veríssimo, P., Rodrigues, L., and Casimiro, A. (1997). Cesiumspray: a precise and accurate global clock service for large-scale systems. *Journal of Real-Time Systems*, 12(3):243–294.

Whetten, B., Vicisano, L., Kermode, R., Handley, M., Floyd, S., and Luby, M. (2001). Reliable multicast transport building blocks for one-to-many bulk-data transfer (rfc 3048). IETF Request For Comments.