MJaco – Integração do Multicast IP na Arquitetura CORBA

Alysson Neves Bessani[*], Joni da Silva Fraga[*] e Lau Cheuk Lung[§]

[*] Laboratório de Controle e Microinformática (LCMI) – DAS – CTC – UFSC
Campus Universitário – Caixa Postal 476 – Trindade - CEP 88040-900 – Florianópolis – SC – Brasil
e-mail: {neves,fraga}@lcmi.ufsc.br

[§]Departamento de Informática
Faculdade de Ciências da Universidade de Lisboa
Bloco C5, Campo Grande, 1749-016 Lisboa – Portugal
e-mail: lau@di.fc.ul.pt

Resumo

Este artigo apresenta nossas experiências na integração do MIOP (*Multicast Inter-ORB Protocol*) em um ORB que segue as especificações da OMG. O modelo proposto para essa integração permite a coexistência de duas pilhas de protocolos distintas (IIOP/TCP/IP e MIOP/UDP/IP multicast), possibilitando um mais amplo espectro de suporte de protocolos aos modelos de comunicação de objetos distribuídos. Esse modelo de integração é discutido neste texto, evidenciando a conformidade de nossa abordagem com as especificações CORBA. No sentido de avaliar o MJaco, foram realizados vários testes envolvendo aspectos de interoperabilidade, desempenho e escalabilidade. Os resultados obtidos mostram que não há uma perda acentuada de eficiência com o nosso modelo de integração. Além disso, o Mjaco traz as vantagens da programação de objetos distribuídos às aplicações que antes se limitavam ao uso de sockets UDP e outras interfaces de mais baixo nível.
Palavras chave: *IP Multicast, Middleware, Protocolos de Comunicação, Sistemas Abertos.*

Abstract

This paper presents our experiences for integrating OMG MIOP (Multicast Inter-ORB Protocol) specifications into an ORB. We proposed a model of integration which allows the coexistence of two different protocol stacks (IIOP/TCP/IP and MIOP/UDP/IP multicast), making possible a wider spectrum of middleware support for distributed objects communication. That integration model is discussed in this paper, giving evidence of the accordance of our approach with the CORBA specifications. In order to evaluate that integration, differents tests were made considering the interoperability, performance and scalability aspects. The obtained results show that there is not a significant loss of performance with that integration model, which brings the advantages of the objects distributed programming for applications that before were limited to the UDP sockets and other lower-level interfaces.
Keywords: *IP Multicast, Middleware,Internet Protocol, Open Systems.*

1. Introdução

A arquitetura CORBA (*Common Object Request Broker Architecture* [OMG01]) introduzida pela OMG (*Object Management Group*[1]), corresponde nos dias de hoje as mais bem sucedidas especificações de *middleware* para suporte a sistemas de objetos distribuídos. O principal componente desta arquitetura é o ORB (*Object Request Broker*) que entre outras coisas implementa as semânticas de comunicação definidas na arquitetura. As mensagens geradas nas comunicações seguem uma sintaxe de transferência própria: o *General Inter ORB Protocol* (GIOP). 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 o TCP/IP é concretizado através do IIOP, *Internet Inter ORB Protocol.*

O IIOP, seguindo as características do TCP, foi definido para comunicações ponto a ponto. Apesar da combinação IIOP – TCP/IP corresponder a uma boa solução para comunicações que seguem este modelo – abordando aspectos como controle de erro, ordenação FIFO, etc. – muitos outros paradigmas de comunicação quando implementados sobre estes protocolos não aproveitam as características dos níveis mais baixos da rede. O reflexo desta dificuldade sempre recai em custos no desempenho destas comunicações.

Muitas aplicações distribuídas dependem de abstrações como grupos de objetos ou da necessidade da disseminação de dados entre vários *hosts* da rede. Estas aplicações necessitam de um melhor aproveitamento dos serviços de uma rede. Assim, em 1999 a OMG publicou um RFP (*Request For Proposal*) onde definia uma série de requisitos para um serviço de *multicast* não confiável que deveria ser suportado pelo *multicast IP*. O *multicast* IP compreende um conjunto de extensões ao protocolo IP que o habilita na concretização de comunicações multiponto [Dee86]. 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 nas áreas de multimídia.

Em 2001, como resposta a RFP, duas propostas foram submetidas a OMG: a primeira vinha da *Inprise* (antiga *Borland*) e a outra foi desenvolvida por um conjunto de empresas e instituições de pesquisa, dentre as quais destacam-se a *Eternal Systems*, *IONA*, *Object Oriented Concepts* e a *University of California, Santa Barbara*. Esta segunda proposta foi a vencedora e se tornou a base das especificações a serem homologadas pela OMG em meados de 2002. Como conseqüência destes esforços de padronização, surgiu então o MIOP (*Multicast Inter-ORB Protocol*) [OMG01b], protocolo responsável pelo mapeamento do GIOP sobre a pilha UDP/*multicast* IP. O MIOP é o elemento chave para tornar disponível um serviço de *multicast* não confiável em suportes CORBA de objetos distribuídos.

A introdução do paradigma de grupo em padrões abertos tem sido alvo de vários trabalhos de pesquisa e propostas de padronização [Dee85, Dee86, Mel90, Ren95, OMG01b, Mos01, Bar01]. Prover suporte de grupo a aplicações distribuídas envolve uma combinação de protocolos que tratam do gerenciamento de grupo (*membership*), detecção de falhas, transferência de estado e comunicação de grupo. Dentro da OMG, essas abstrações estão sendo padronizadas separadamente, os três primeiros são tratados nas especificações *FT-CORBA* [OMG00]. Todavia, a OMG ainda não tem publicado uma especificação para a comunicação de grupo na arquitetura CORBA que atenda, por exemplo, diferentes níveis de garantias nas várias versões possíveis deste paradigma. A OMG está tratando deste problema em etapas. O primeiro passo foi a criação de um grupo de trabalho para definir a especificação do suporte para *multicast* não confiável – o modelo menos restritivo de comunicação de

[1] entidade formada por mais de 1000 instituições que visa promover a tecnologia de orientação a objetos.

grupo. Essa iniciativa deverá motivar a publicação de outras *RFPs*, no sentido da concepção de suportes a comunicação de grupo que forneçam também garantias mais restritivas de acordo e ordenação (por exemplo, *multicast* confiável, ordem FIFO, causal, total, etc) .

O objetivo neste artigo é apresentar as nossas experiências na integração do MIOP em um ORB que segue as especificações da OMG. O nosso modelo de integração é discutido neste texto, evidenciando a conformidade de nossa abordagem com as especificações CORBA. Este trabalho é parte do projeto *GroupPac* (*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 MIOP/UDP/*multicast* IP possibilitando um mais amplo espectro de suportes de protocolos aos modelos de comunicação de objetos distribuídos.

No sentido de avaliar nossos trabalhos nesta integração, foram realizados vários testes envolvendo aspectos de interoperabilidade, desempenho e escalabilidade em nossa implementação. Neste artigo, apresentamos um estudo comparativo que realizamos com o nosso serviço de *multicast* não confiável, baseado na pilha MIOP/UDP/*multicast* IP, e o uso de *sockets* UDP para comunicação *multicast* IP que é prática corrente em aplicações de multimídia, por exemplo. Os resultados obtidos mostram que há uma perda de eficiência com o nosso modelo de integração, mas que é compensada pelas vantagens da programação de objetos distribuídos em aplicações que antes se limitavam ao uso de *sockets* UDP e outras interfaces de mais baixo nível.

O artigo apresenta na seção 2 uma descrição sucinta do *multicast* IP. Aspectos da especificação descrevendo o MIOP e as estruturas necessárias a nível de ORB para suportar *multicast* não confiável são apresentados na seqüência, na seção 3. Na seção 4, é introduzido o nosso modelo de integração do MIOP que procura preservar as duas pilhas de protocolos: uma para comunicações ponto a ponto e outra multiponto. Detalhes de implementação do nosso serviço de *multicast* não confiável e da integração do MIOP são descritos na seção 5. Na seção 6, são apresentados e discutidos alguns testes no nosso ORB *multicast*. A seção 7 cita alguns trabalhos relacionados e, finalmente, na seção 8, são levantadas as conclusões finais deste trabalho.

2. *Multicast* IP

O conjunto de extensões adicionadas ao protocolo IP para o suporte comunicação multiponto é chamado de *multicast* IP (RFCs 966 [Dee85] e 988 [Dee86]). A partir deste serviço é possível enviar um pacote IP a um conjunto de *hosts*, denominado de grupo. Cada grupo de *hosts* é identificado por um endereço IP (semelhante a um endereço de *host*[2]). Quando o destino de um datagrama é um grupo, é feita uma tentativa de entrega a todos os *hosts* membros do mesmo, entretanto, a exemplo do *unicast IP* não existe qualquer garantia a respeito desta entrega. Ou seja, este serviço não confiável não fornece garantias quanto à

[2] Quanto ao endereçamento, o *multicast* IP define que cada grupo deve ser identificado por um endereço IP classe D (que começa com 1110 e vai de 224.0.0.0 a 239.255.255.255) associado a ele, desta forma, para um emissor enviar um datagrama a um grupo, basta que ele defina o endereço IP do grupo no campo destino do datagrama IP. Um outro ponto importante sobre o endereçamento de grupos é a tradução de endereços *multicast* IP para endereços de enlace. Este serviço, que é executado pela camada de enlace, deve permitir em uma rede *Ethernet*, por exemplo, o mapeamento de endereços *multicast* IP para endereços *multicast Ethernet*. Para redes que não suportam *multicast* (em nível de enlace) os endereços *multicast* IP devem ser mapeados para endereços de *broadcast* (atingindo todos os nós da rede local), com os descartes possíveis nos *hosts* que não pertencerem ao grupo.

entrega dos pacotes a todos os membros do grupo ou à integridade dos pacotes; aspectos de ordem também não são garantidos.

As especificações do *multicast* IP definem dois tipos de grupos: grupos permanentes e grupos temporários. Os grupos permanentes estão sempre presentes e não precisam ser criados, já os grupos temporários devem ser criados e duram enquanto houver processos pertencentes a estes. Os grupos definidos no *multicast* IP são grupos abertos no sentido que um emissor não pertencente ao grupo pode enviar mensagens ao mesmo. Além disso, não existem informações a respeito de quais são os membros de um grupo, no sentido global. Cada *host* sabe a que grupo pertence, porém desconhece os outros membros.

O gerenciamento dos membros de um grupo é feito de maneira dinâmica, de tal forma que a associação de endereços IP a *hosts* seja bastante flexível. Assim, um *host* pode pertencer a um ou mais grupos, ou a nenhum grupo em determinado momento. Esta associação dinâmica é realizada através das três operações de gerenciamento, que o módulo IP deve prover. Estas operações são mostradas na tabela 1. Note que não existe operação para destruição de grupos. O que se justifica pelo fato de grupos permanentes não poderem ser destruídos e grupos temporários só existirem enquanto seu número de membros é maior que zero.

Operações	Descrição
`CreateGroup`	Cria um grupo temporário; o *host* invocador é o único membro de início
`JoinGroup`	Adiciona o *host* invocador ao grupo especificado (permanente ou temporário)
`LeaveGroup`	Remove o *host* invocador do grupo especificado

Tabela 1 - Operações de gerenciamento no *multicast* IP.

As operações de gerenciamento são implementadas através do protocolo IGMP, *Internet Group Management Protocol* [Dee86], utilizado na comunicação entre os *hosts* de uma rede local com o agente *multicast*. Toda rede que suporta *multicast* IP deve conter pelo menos um agente *multicast* ativo que, usualmente implementado no *gateway* da rede, é a entidade responsável pela criação e manutenção de grupos bem como pela troca de mensagens no contexto da *Internet*. Cada agente *multicast* deve saber quais grupos tem membros em sua rede.

3. Modelo de Objetos MIOP e o Suporte de *Multicast* Não Confiável

O modelo atual de objetos CORBA (pilha IIOP/TCP/IP) especifica que uma referência de objeto[3] deve corresponder a uma única implementação. Além disso, a semântica de invocação do CORBA tem tratamento de erro que adiciona confiabilidade em relação à entrega e, também, ordenação FIFO de mensagens que podem ser definidas com ou sem espera de resposta.

A entrega de mensagens GIOP via serviço *multicast* não-confiável define um contexto de comunicação completamente diferente do citado acima. O objetivo nas especificações MIOP (*Multicast Inter-ORB Protocol*) é prover mecanismos para o *multicast* não-confiável entre objetos distribuídos. Estes mecanismos se reduzem à necessidade de estruturas possibilitando o mapeamento de mensagens GIOP em pacotes IP, e alguma forma de gestão consistente de grupos em nível de objetos distribuídos.

[3] Um identificador de objeto também chamado de referência ou IOR (*Interoperable Object Reference*) é peça fundamental na arquitetura CORBA. É único no espaço do sistema e interoperável, atravessando diferentes implementações de ORBs. Esta estrutura é uma espécie de ponteiro que liga o *stub* cliente a implementação do objeto no servidor.

Na gestão em nível de objetos, é necessário que se introduza a noção de grupo e de seu correspondente identificador (referência de grupo). Esta gestão é simplificada: como não existem requisitos de confiabilidade no MIOP, não é necessário um serviço de *membership* no sentido de se ter conhecimento de quantos e quem são os membros do grupo.

Na verdade um grupo de objetos no MIOP consiste em informações sobre como acessá-lo através da rede. Estas informações estão contidas na estrutura `UIPMC_ProfileBody`, disponível a partir da referência de grupo (figura 1). O perfil UIPMC difere do tradicional perfil IIOP presente em IORs de objetos simples nos seguintes pontos:

- Não existe chave de objeto[4];
- O endereço IP do *host* do perfil *IIOP* é substituído no *UIPMC* por um campo que deve conter um endereço de *multicast* IP, ou um *alias* deste endereço.

A ausência de chaves e endereços de *hosts* de objetos neste perfil é pertinente: este serviço de comunicação não confiável foi definido sob a premissa da não existência de *membership* (não são conhecidos quantos e quem são os membros do grupo). Mesmo assim, é possível a partir do perfil UIPMC chegar a um objeto membro do grupo nos POAs (*Portable Object Adapter*[5]) disponíveis em *hosts* receptores da mensagem.

Este cenário é bem mais complexo do que o dos perfis IIOP, que identificam unicamente uma implementação de interface. Além de informações de versão do protocolo, do endereço IP e a porta que o grupo atende, o perfil UIPMC contém também uma série de componentes como, por exemplo, um perfil IIOP para envio de requisições que necessitam de resposta (o MIOP é definido para requisições *oneway*) e uma estrutura, a `TagGroupTaggedComponent`, que contém informações como a versão do grupo, o *id* do domínio e o *id* do grupo[6]. Este último componente é obrigatório e deve sempre estar presentes nos perfis *UIPMC*.

Além do perfil UIPMC (obrigatório), uma IOR de grupo pode possuir opcionalmente um ou mais perfis *IIOP* de *gateways* para serem utilizados por clientes que não suportam *multicast* IP.

Uma *IOR* de grupo é usada então, basicamente, em três tipos de invocações:

- A um *gateway IIOP*, nos casos em que o cliente, através de seu ORB, não é capaz de realizar multicast;
- A servidores que suportam o MIOP, implementam a mesma interface e pertencem ao mesmo grupo;
- A um membro do grupo em servidor IIOP que deve responder a operações *two-way*.

A figura 1 apresenta a estrutura completa de uma IOR de grupo, com os perfis UIPMC e IIOP de *gateway* e de objeto membro.

Conforme já citado, o perfil UIPMC não define chaves de objetos para o acesso a membros de um grupo. Assim, um POA deve utilizar as informações associadas ao grupo (*id*, domínio e versão) definidas no perfil UIPMC para identificar os membros locais do grupo.

---

[4] A chave de um objeto (*ObjectKey*) é o identificador do objeto dentro do ORB, ela é utilizada, juntamente com o endereço IP e a porta do ORB, na localização da implementação de um objeto.

[5] Componente do ORB responsável pelo registro e ativação de objetos, bem como pelo encaminhamento das mensagens recebidas a estes.

[6] A mesma função do *ObjectKey* no IIOP é desempenhada pelo *id de grupo* no perfil UIPMC.

Esta associação entre grupos e implementações de membros locais requer uma nova tabela interna ao POA. Esta tabela, chamada *Mapa de Grupos Ativos*, define para as informações de grupo de um UIPMC o conjunto de implementações correspondentes, acessíveis a partir do adaptador. Desta forma, a inclusão de membros a grupos é feita sempre localmente através de chamadas aos POAs ativados no ORB. A fim de que o POA permita a associação de implementações de objetos a referências de grupo, as especificações do MIOP prevêem quatro novas operações para a interface `PortableObject::POA`. Estas operações permitem associar e desassociar implementações registradas a referências de grupos; bem como, listar que implementações pertencem a determinado grupo no POA − fornecendo com isto uma espécie de serviço de *membership* local.

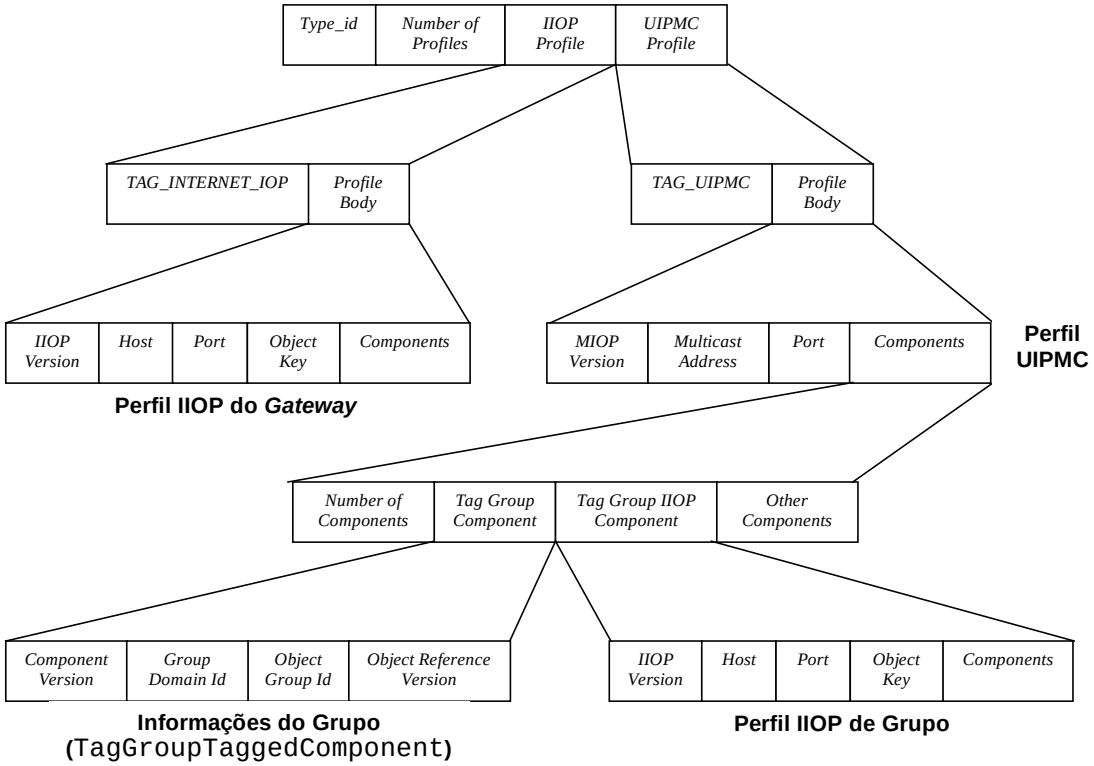

Figura 1 - IOR de grupo.

O pequeno trecho de código em *Java* da figura 2, mostra a associação de uma implementação de objeto a uma referência de grupo.

```
...
1 org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args,null);
2 org.omg.CORBA.Object poaObj = orb.resolve_initial_references(“RootPOA”);
3 POA poa = POAHelper.narrow(poaObj);
4 poa.the_POAManager().activate();
5 org.omg.CORBA.Object group = orb.string_to_object(“corbaloc:” +
6   “miop:1.0@1.0-mydomain-1-2/225.1.2.5:7676;” +
7   “iiop:1.1@localhost:1236/MyObjectKey”);
8 MyGroupMemberImpl member = new MyGroupMemberImpl();
9 byte[] memberId = poa.activate_object(member);
10 poa.associate_reference_with_id(member,memberId);
11 orb.run();
...
```

Figura 2 - Associação no POA de uma *IOR* de grupo com uma implementação de objeto.

Nas linhas de 1 a 4 na figura 2, referências ao ORB e ao POA são obtidas e ativadas. A seguir, nas linhas de 5 a 7, uma referência a um grupo é criada através de uma URL codificada no mecanismo *corbaloc*[7]. Esta URL define um perfil UIPMC para o grupo e um perfil IIOP para operações com resposta. Note que nesta URL estão descritas informações a respeito dos endereços a serem usados na comunicação (MIOP: *225.1.2.5* porta 7676 e IIOP: *localhost* porta 1236) e as informações lógicas do grupo (MIOP: *1.0-mydomain-1-2* e IIOP: *MyObjectKey*). Tendo obtido uma referência ao grupo, as linhas 8 e 9 apresentam a instanciação de uma implementação da interface `MyGroupMember` e sua ativação pelo POA. Na linha 10 uma das novas operações do POA é chamada para a associação da implementação ativada à referência de grupo, e finalmente, na linha 11 o ORB é ativado e está pronto para receber requisições.

Na listagem de código da figura 2, nota-se que a criação de grupos e a associação de membros a estes, se comparada a outros suportes de grupo (por exemplo, o FT-CORBA), é feita de maneira muito complexa, envolvendo manipulações de estruturas de mais baixo nível. Para contornar este problema, a especificação do MIOP define um objeto de serviço opcional chamado MGM - *Multicast Group Manager*, que é introduzido para oferecer uma interface mais evoluída de gerência de grupos de objetos *multicast*. O MGM contém operações para a criação e destruição de grupos menos complexas em termos de uso que os mecanismos usuais como o *corbaloc*. Outras operações são para o gerenciamento de propriedades dos grupos gerados. Como exemplos destas propriedades podemos citar a definição de componentes IIOP de grupo, de *gateway*, e do endereço *multicast* e porta utilizada pelo grupo.

3.1. Protocolo *Multicast* Inter-ORB

Até agora foi apresentado o modelo de objetos do MIOP que provê o suporte a grupos, a outra parte da especificação trata do protocolo MIOP propriamente dito. Uma mensagem GIOP é sempre encapsulada em um conjunto de pacotes MIOP. Um pacote MIOP é composto de um cabeçalho (*PacketHeader*) e um bloco de dados GIOP. Um dos requisitos da RFP que originou a especificação do MIOP é a integração do mesmo com a pilha UDP/*multicast* IP.

O tamanho máximo de um bloco de dados GIOP que pode ser contido em um pacote MIOP geralmente depende do tamanho do *frame* suportado pelo hardware de rede utilizado. Redes *Ethernet*, por exemplo, suportam *frames* de 1518 bytes. O formato do cabeçalho MIOP é apresentado na figura 3.

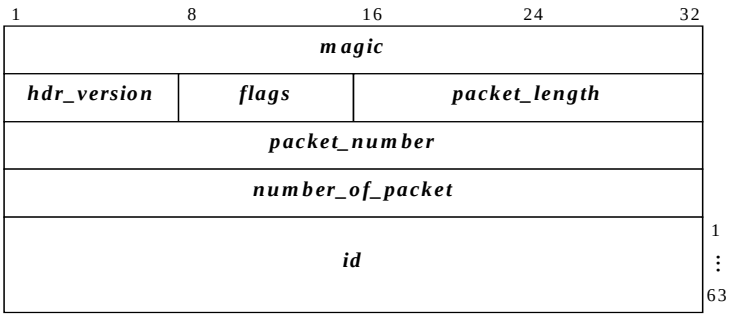

Figura 3 - Cabeçalho do pacote MIOP.

[7] Este mecanismo define uma estrutura de URL (*Uniform Resource Locator*) para criação de referências a objetos. Por exemplo, a URL: "*corbaloc:iiop:1.1@jung.lcmi.ufsc.br:1245/ATF%1235655%*", cria uma referência a um objeto registrado em um ORB no *host jung.lcmi.ufsc.br*, porta 1245, cuja chave de objetos é ATF%1235655%.

O principal objetivo do cabeçalho da figura 3, definido pela estrutura `PacketHeader_1_0` na especificação, é permitir a reconstrução de mensagens GIOP no receptor. O campo *magic* identifica o bloco de dados como um pacote MIOP, para tanto ele deve conter sempre o valor literal de 4 bytes 'MIOP'. Já o campo *Hdr_version* define a versão do protocolo, na versão atual da especificação o valor deste campo deve ser *10H* (versão 1.0). O campo *flags* de 8 bits define alguns bits de controle que indicam a codificação de dados (*big* ou *little endian*) e se o pacote representa o fim de mensagem. No campo *packet_length* está definido o tamanho do bloco de dados GIOP (em bytes). *Packet_number* define o número do pacote na coleção. Em *number_of_packets*, campo opcional (deve conter 0 quando não utilizado), é informado ao receptor quantos pacotes devem chegar para completar esta mensagem. Finalmente, o campo *id* identifica a mensagem relativa ao pacote. O único requisito associado a este campo é que cada coleção de pacotes deve ter um identificador único de mensagem.

Conforme já citado, o MIOP é um protocolo baseado em UDP/*multicast* IP e, portanto, não confiável – a possibilidade da perda de pacotes existe e é dependente da rede utilizada. Mecanismos usuais de temporização devem ser utilizados para aguardar um pacote e, caso este não chegue, a coleção de pacotes referente a mensagem GIOP correspondente, obrigatoriamente, é descartada.

4. Uma Abordagem de Integração do MIOP

O modelo de grupo apresentado na seção anterior e definido nas especificações do MIOP, é bastante simples se considerarmos a não existência de garantias na comunicação e a ausência de serviços de *membership*. É exatamente o mesmo modelo de grupo utilizado no *multicast* IP. Portanto, esta especificação do MIOP se concentra apenas em definir um modelo básico de grupo na arquitetura CORBA, deixando a expectativa que modelos mais complexos de comunicação de grupo serão cobertos em futuras especificações da OMG através de incrementos a este modelo básico.

A nossa intenção é ter este mecanismo de *multicast* não confiável disponível no *GroupPac* onde, junto como os aspectos de gerência de grupo do FT-CORBA, formaria um conjunto de funcionalidades para o suporte de diferentes modelos de grupos. Num primeiro passo, os nossos esforços se concentraram na integração deste suporte de *multicast* não confiável em um ORB.

4.1 Abordagem de Integração

A figura 4 apresenta a nossa arquitetura de integração do MIOP no ORB, implementada no projeto *MJaco*. O *MJaco* é uma extensão do *JacORB* [Bro97], um ORB/CORBA convencional. A arquitetura *MJaco* é proposta de 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.

Na figura 4 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 o UDP/*multicast* IP como mecanismo de transferência de seus pacotes. O nosso modelo de integração apresenta os vários elementos definidos em especificação que compõem o suporte para os dois modelos de comunicação. Foram adicionados ainda outros componentes e extensões, não definidas em especificações, cuja finalidade é facilitar o convívio das diferentes pilhas e melhorar a eficiência de utilização do conjunto.

Nossa abordagem inclui um objeto de serviço local MGM+ que estende o MGM das especificações da OMG adicionando a este funcionalidades para criação e gerenciamento de grupos em nível de objetos. Na criação de referências de grupos, implementada através de chamadas ao ORB (e montagens de URLs *corbaloc*), o MGM+ pode realizar o registro da referência criada em um servidor de nomes CORBA, tornando-a automaticamente disponível para qualquer outra aplicação[8]. As outras facilidades do MGM+ dizem respeito a mudanças de objetos membros de um grupo (gerenciamento local da mudança de membros do grupo), são possíveis através de algumas operações, como `add_member` ou `delete_member`. Estas operações centralizam no MGM+ as interações com o POA na associação entre grupos e implementações de membros locais através da tabela interna *Mapa de Grupos Ativos* (seção 3).

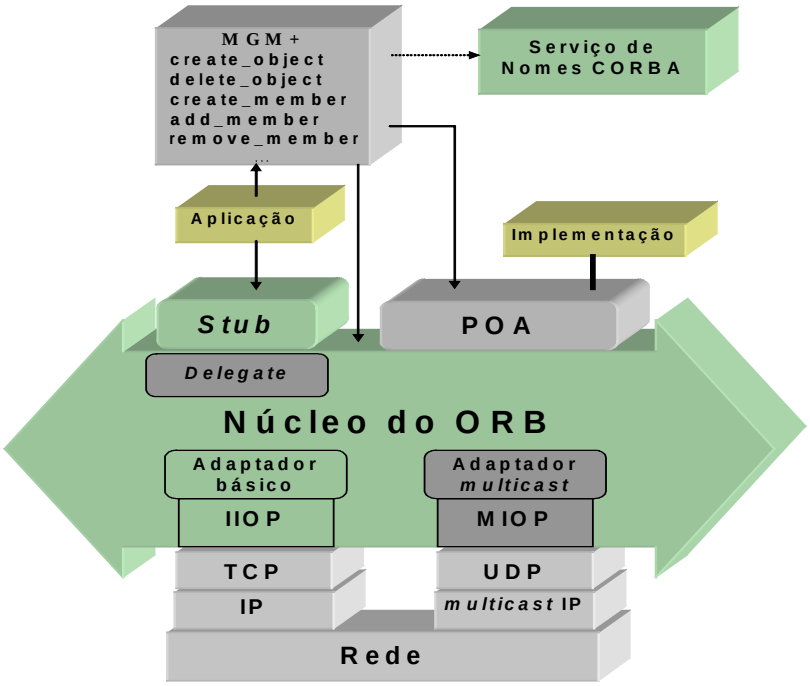

Figura 4 - Arquitetura do *MJaco*.

Um outro componente que faz parte do nosso modelo de integração é o Adaptador *Multicast*, responsável pela gerência de *sockets multicast* utilizados na recepção de pacotes MIOP e pela entrega de mensagens endereçadas a grupos aos POAs ativos no ORB. O módulo MIOP 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. O *Delegate* é alterado em alguns pontos para dar suporte ao envio de mensagens GIOP para grupos; ele é o primeiro componente interno do 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 das quatro primitivas descritas nas especificações da OMG, deve ser alterado para o processamento de requisições endereçadas a grupos, realizando buscas na tabela de grupos ativos para obtenção das implementações membros do grupo para o qual a mensagem está endereçada. O POA também é o componente

[8] Que deve ser previamente especificado em uma das propriedades do objeto.

responsável pela ativação do adaptador *multicast* para a execução das operações de gerenciamento definidas pelo *multicast* IP. Por exemplo, quando a operação `associate_reference_with_id`[9] é chamada para o registro do primeiro membro de um grupo no ORB, o POA chama o adaptador *multicast* para que este crie um *socket* e execute a operação `JoinGroup` do *multicast* IP no endereço definido no perfil UIPMC presente na referência de grupo. A partir de então, o ORB passa a receber mensagens destinadas a este grupo. As outras operações do POA relacionadas a grupos, também resultam na execução de operações de gerenciamento do *multicast* IP.

Este modelo de integração foi concretizado no *JacORB* mas pode ser reproduzido em outros ORBs provavelmente com poucas modificações.

5. *MJaco*: Implementação do MIOP no JacORB

Para implementação das especificações do MIOP e do modelo apresentado na seção 4, escolheu-se o *JacORB* (http://www.jacorb.org), um ORB não comercial e de código aberto, disponibilizado pela *Freie Universität Berlin*. Este ORB foi escolhido devido a sua reconhecida qualidade e desempenho e pela nossa experiência com o mesmo nos projetos *GroupPac* [Lau00] e *JaCoWeb* [Wan01].

A figura 5 apresenta um diagrama de classes em UML com as classes que processam a requisição dentro do cliente *JacORB*. O conjunto de classes mostrado interage desde a chamada do método no *stub* por um cliente CORBA até o envio da mensagem GIOP correspondente através da conexão TCP. O *stub* é o responsável pela serialização da chamada do método, deixando o envio da mensagem a cargo da classe *Delegate*. Esta classe mantém uma conexão TCP aberta para o ORB servidor, onde a implementação correspondente ao *stub* está registrada e implementa as operações referentes à interface `CORBA::Object`, base de todas as interfaces definidas para objetos CORBA.

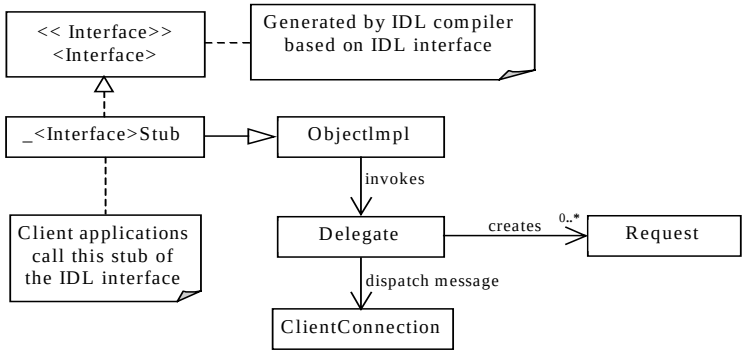

Figura 5 – Processamento de requisições no ORB cliente.

No lado do servidor, o processamento de requisições é bem mais complexo. A figura 6 apresenta um diagrama de classes com as principais participantes do processamento da requisição desde sua chegada pelo *socket* até sua execução pela implementação apropriada. Um objeto da classe *Listener* "escuta" uma porta de rede a espera de requisições via TCP/IP. Quando uma requisição é recebida nesta porta, o objeto cria uma nova instância da classe *RequestReceptor*, que é a responsável por interpretar a requisição e encaminhá-la ao POA destino[10]. Uma vez no POA, a requisição é colocada em uma fila de processamento para

[9] Uma das quatro novas operações do POA definidas nas especificações do MIOP. Esta operação é responsável pela associação de uma implementação a uma referência de grupo (veja exemplo na figura 2).
[10] O POA em que a implementação destino está registrada.

execução pela implementação destino (subclasse de *Servant*). Note que várias classes presentes na figura 6 têm o estereótipo *thread*, estas classes representam os pontos da estrutura do *JacORB* que implementam sua arquitetura *multithreading*.

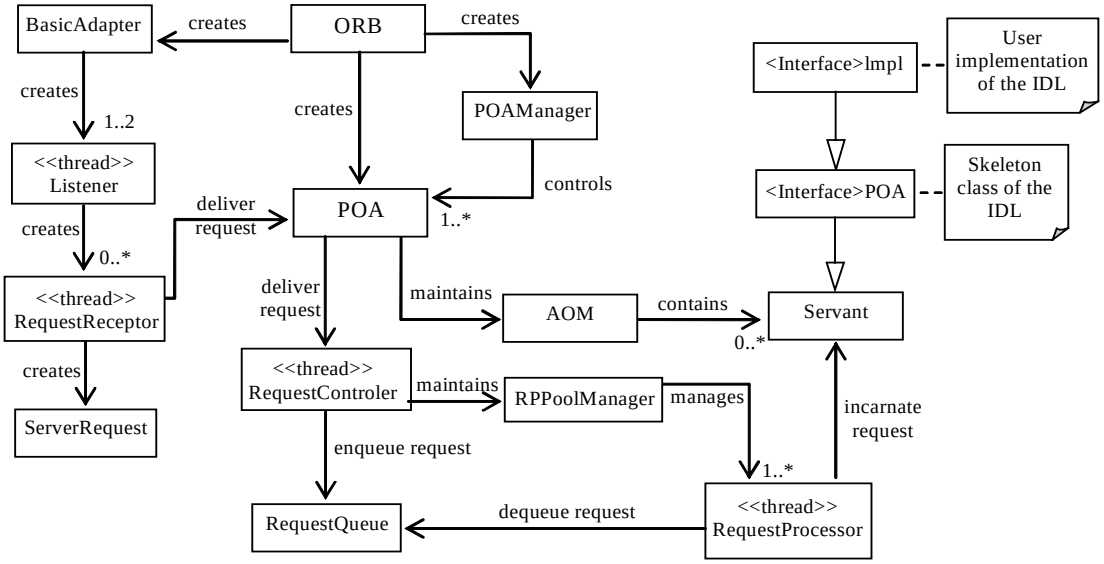

Figura 6 – Processamento de requisições no ORB servidor.

Para o processamento de mensagens MIOP, uma nova estrutura de classes teve de ser implementada e adicionada àquelas presentes nas figuras 5 e 6, representando os mecanismos de processamento de requisições IIOP do *JacORB*. A figura 7 apresenta um diagrama com esta estrutura.

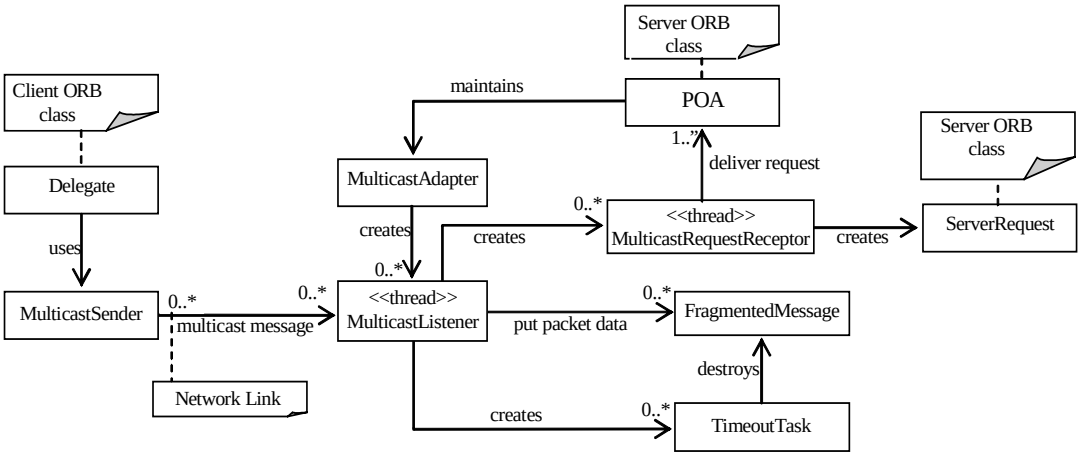

Figura 7 – Classes de extensões para processamento de mensagens MIOP.

Neste diagrama podemos identificar o ponto do processamento de requisições do ORB que é alterado de modo que mensagens também possam ser enviadas via *multicast*: a classe *Delegate*. É nesta classe que, quando o objeto destino corresponde a uma referência de grupo e a mensagem é *oneway*, o processamento é desviado para a classe *MulticastSender*, que encapsula a requisição GIOP em uma coleção de pacotes MIOP para serem enviados via *multicast* IP. No servidor, um objeto da classe *MulticastListener* é criado para cada grupo em que o ORB tem membros registrados (um *listener* para cada porta e uma porta para cada

grupo) pelo adaptador *multicast* (classe *MulticastAdapter*). Estes *listeners* recebem os pacotes MIOP e armazenam os fragmentos da mensagem encapsulada neles. Quando a mensagem está completa, uma *thread* (classe *MulticastRequestReceptor*) é retirada do *Pool de Threads*[11] e disparada para unificação dos fragmentos na mensagem GIOP original que é repassada a todos os POAs do ORB. Em cada POA o *Mapa de Grupos Ativos* (classe AGM), relaciona as informações do grupo contidas no cabeçalho da mensagem GIOP com o conjunto de identificadores de objetos pertencentes a este grupo que foram registrados neste POA.

Note que a requisição é encaminhada a todos os POAs ativos no ORB, mesmo aqueles que não registram implementações de objetos pertencentes ao grupo destino, este algoritmo de entrega foi desenvolvido visando tornar o AGM o mais simples possível e evitar o *overhead* de processamento imposto pela busca de POAs destino. Outro fator que justifica esta implementação é o fato de que na maioria das aplicações, não se ativa um grande número de POAs.

6. Resultados Obtidos

Nesta seção são apresentamos alguns testes simples realizados com o objetivo de validar e verificar o desempenho da implementação do *MJaco*. Os testes foram executados em uma rede local usando computadores com a mesma configuração[12], conectados através de um mesmo *hub*. Foram implementadas duas versões de um programa distribuído que mede o tempo de *round-trip*[13]: uma utilizando *sockets multicast* e outra fazendo uso do *MJaco*.

O primeiro experimento foi montado da seguinte forma: duas instâncias do programa foram iniciadas em cada uma das quatro maquinas, totalizando oito membros em um mesmo grupo. Um destes membros é o transmissor que envia uma mensagem de tamanho variável para o grupo e que, posteriormente, fica aguardando o recebimento de todas as confirmações dos membros receptores do grupo. Esse processo é repetido por 10000 vezes. O experimento foi executado com cada uma das versões do programa exemplo e os resultados obtidos são apresentados na figura 8.

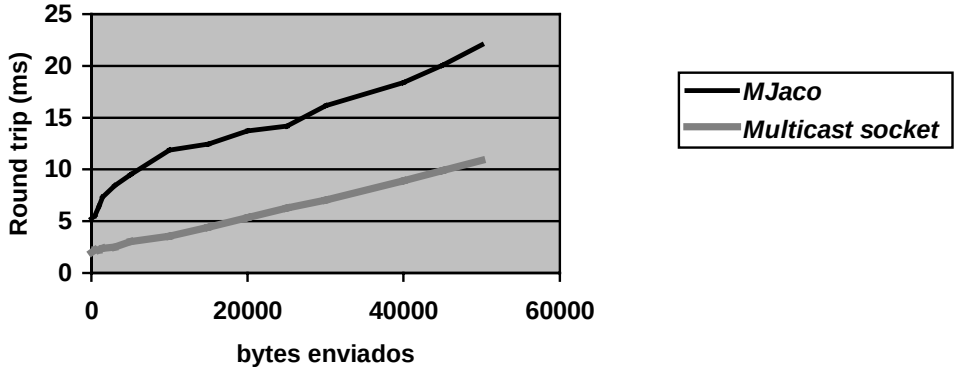

Figura 8 - Desempenho do *MJaco* em relação a *Sockets Multicast*.

No gráfico da figura 8 é possível verificar que a utilização do *sockets multicast* resulta em um desempenho aproximadamente 60% melhor (em média) do que a utilização do *MJaco*.

[11] Estrutura de programação que mantém um conjunto de *threads* criadas e prontas para serem executadas.

[12] Pentium III 900Mhz com 198 Mb de memória RAM e utilizando interfaces de rede Ethernet de 10Mbps. O sistema operacional utilizado é o Suse Linux 7.3.

[13] Representa o tempo entre o início da transmissão de uma mensagem a todos os membros do grupo e a recepção pelo transmissor da última das mensagens de confirmação emitidas pelos membros receptores.

O que já era esperado, visto que o *MJaco* coloca toda uma camada de software para gerenciamento de objetos e requisições bem como para a serialização das mensagens (ver figura 4); estrutura essa que possivelmente as aplicações que utilizam-se de *sockets multicast* teriam de implementar de alguma forma, e que o nosso pequeno programa *sockets multicast* de teste não implementa. Vale ressaltar também que a implementação do *MJaco* foi realizada o menos dependente possível do *JacORB*, o que acarreta algumas perdas de desempenho, que devem ser minimizadas conforme estes dois projetos forem se integrando e as implementações forem evoluindo.

Um ponto importante a se enfatizar é que, na configuração apresentada, a partir das mensagens de 50000 bytes, o sistema começa a perder mensagens. Este comportamento se verificou tanto na versão com *sockets multicast* quanto na que utiliza o *MJaco*, e deve-se a não confiabilidade do UDP, que não possui mecanismos de controle de fluxo para evitar os descartes de pacotes decorrentes de *buffers* cheios, durante o alto congestionamento da rede nos testes.

O segundo experimento realizado visa avaliar a escalabilidade do *MJaco*. Os testes foram realizados da seguinte forma: em cada *host* foi iniciada uma instância do programa teste especificando-se quantos objetos do grupo de testes deveriam ser instanciados e registrados nela. Um dos vários objetos criados nos *hosts* do sistema é o transmissor, ou seja, é ele que transmite a mensagem com 4Kbytes de dados e que faz a medição do tempo de *round-trip*. O número de mensagens difundidas durante um *round-trip* é `4n+1` onde `n` é o número de objetos pertencentes ao grupo registrados em cada *host* (uma mensagem e um reconhecimento para cada objeto membro do grupo).

O processo de *round-trip* foi repetido para um número variado de objetos por *host* e os resultados são mostrados na figura 9.

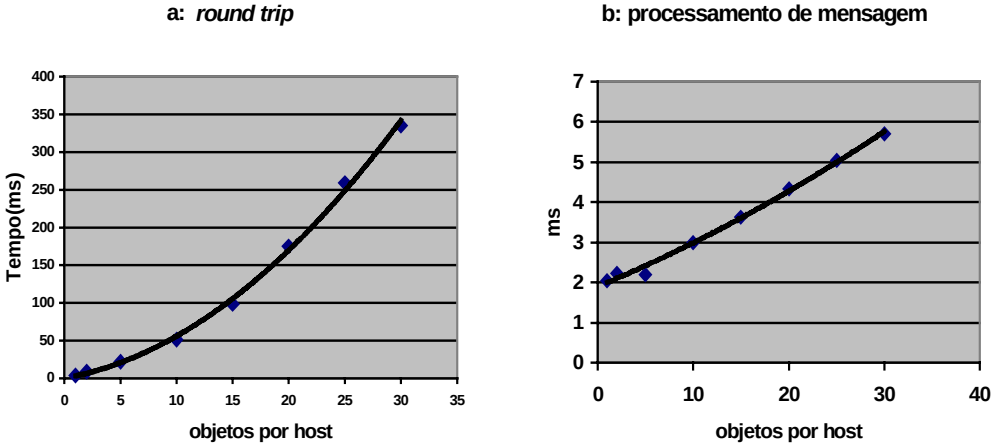

Figura 9 - Desempenho do *MJaco* em relação a escalabilidade.

O gráfico *a* da figura 9 mostra a situação óbvia em que, à medida que adicionamos membros no sistema, o tempo de *round-trip* aumenta bastante. Isto se deve claramente ao aumento substancial na quantidade de mensagens geradas na rede, ocupando o meio físico. Por exemplo, com 2 objetos por *host* (8 membros no grupo), temos 9 mensagens trocadas (1 mensagem de dados e 8 de reconhecimentos) para um *round-trip*. Para 10 objetos por *host* temos 41 mensagens trocadas. Levando em conta o tempo de duração de um *round-trip* e dividindo o mesmo pelo seu número de mensagens, é obtida uma estimativa de tempo médio para transferência de uma mensagem no sistema. O gráfico *b* da figura 9 apresenta estes valores em função das composições dos grupos. Neste gráfico é possível notar que a curva

que representa este tempo médio das mensagens é linear[14], e aumenta aproximadamente 0,1 ms para cada novo objeto incluído em cada *host* do sistema, o que demonstra o potencialidade para escalabilidade do *MJaco*, mesmo estando em sua primeira versão.

7. Trabalhos Relacionados

Devido ao fato das discussões e da especificação do MIOP serem recentes, são praticamente inexistentes outras experiências sobre o mesmo na literatura. Além do *MJaco*, existe a iniciativa de integração do MIOP em outro ORB: a equipe que desenvolve o ORB *TAO* faz um esforço também nesta direção [Hun01]. Entretanto, até a submissão deste artigo, a versão *multicast* deste ORB não estava disponível.

Um outro trabalho de integração do *multicast* IP ao suporte de *middleware* aparece no projeto *JavaGroups* [Ban99]. Neste projeto, um conjunto de ferramentas é feito disponível no sentido de adicionar mecanismos de comunicação de grupos a sistemas de objetos distribuídos que possuem como base o RMI. O *JavaGroups* trabalha com a abstração de canais na comunicação de grupos onde propriedades são feitas disponíveis a partir de ferramentas como, por exemplo, o *Ensemble* [Hay98]. Mas o seu canal *default* compreende uma pilha de protocolos construída sobre o UDP/*multicast* IP. Esta pilha é configurável, permitindo a construção de modelos mais complexos de comunicação de grupo através da inclusão de propriedades de ordenação e de confiabilidade ao canal *default*.

Em relação a inclusão de novos protocolos de transporte em ORBs CORBA, a OMG lançou um RFP chamado *"Extensible Transport Framework for Real Time CORBA"* a respeito disso. Este RFP obteve duas respostas [OMG01c, OMG02]. Até a submissão deste artigo não havia sido escolhida a proposta vencedora, porém quando esta for definida pela OMG, sua implementação será considerada no projeto *MJaco*.

8. Conclusão

O *Unreliable Multicast Inter-ORB Protocol* é o primeiro passo para outras soluções de comunicação de grupo na arquitetura CORBA. Atualmente estamos realizando estudos sobre protocolos de difusão confiável e avaliando como estes podem ser adaptados ao CORBA, e em especial as especificações do *Unreliable Multicast*, já que o objetivo do projeto é implementar um mecanismo de difusão confiável sobre o MIOP no *MJaco*. Portanto, questões sobre tratamento de perdas de pacotes, análise de desempenho e etc, não foram discutidos aqui. O objetivo principal deste projeto é, a partir de um mecanismo de difusão não confiável, propor extensões para obter protocolos de difusão com garantias mais restritivas de acordo e ordenação.

Estas soluções deverão ser aproveitadas no projeto *GroupPac* [Lau00, Lau01], que implementa a especificação *Faul-Tolerant* CORBA, no sentido da concepção de modelos de replicação ativa [Sch90], inexistentes nas atuais especificações da OMG. A própria OMG já está reunindo um grupo para trabalhar em uma *RFP* sobre um *multicast* confiável no CORBA[15]. Estes serviços de *multicast* com garantias devem ser implementados sobre o MIOP.

Neste artigo foram apresentadas nossas soluções para a integração do *Multicast* IP em um ORB CORBA. O modelo de integração proposto não compromete aspectos de interoperabilidade e portabilidade do ORB como um todo. O ORB é capaz de realizar

[14] Excluindo-se os valores quase constantes obtidos inicialmente (onde a mensagem de dados ainda tem grande influência devido a seu tamanho).

[15] Informação da composição de um *working group* em *reliable multicast* obtida na lista de discussão da OMG sobre tempo real.

invocações tanto usando o IIOP (convencional) como o MIOP. Este modelo pode muito bem ser adotado para integrar outros protocolos de comunicação, desde que esses tenham uma API disponível.

Além disso, foram também apresentadas neste texto as nossas experiências nas implementações do *MJaco*. Estes desenvolvimentos que tiveram como base o modelo de integração proposto, foram concretizados sobre o *JacORB*, um ORB *Java free*. Estas implementações podem ser obtidas na WEB no seguinte endereço (http://grouppac.sourceforge.net/).

Foram também realizados vários testes sobre o *MJaco*. Estes testes apontam alguns custos no desempenho não favoráveis ao ORB. Os resultados obtidos mostram que há uma perda de eficiência, mas talvez compensada, se levarmos em conta as vantagens da programação de objetos distribuídos. O gerenciamento de objetos e *threads*, totalmente a cargo do POA e do ORB, e a interoperabilidade fornecida pelo padrão CORBA adicionam simplicidade a aplicações que antes se limitavam ao uso de *sockets* UDP e outras interfaces de mais baixo nível. Esta arquitetura também tem disponível todo um *framework* de objetos de serviços (com serviços de nomes, transações, concorrência, tempo, entre outros) que podem ser de grande utilidade para as aplicações que utilizam *multicast*.

Agradecimentos

Agradecimentos especiais a Ricardo Sangoi Padilha e Luciana Moreira Sá de Souza, bolsistas do LCMI-DAS-UFSC, pela inestimável contribuição nas implementações do *MJaco*.

Referências

[Ban99] B. Ban, **"Design and Implementation of a Reliable Group Communication Toolkit for Java"**, Disponível em http://www.cs.cornell.edu/Info/Projects/JavaGroupsNew/, Tech Report, 1999.

[Bar01] F. A. R. Barros, M. Stanton, **"Modelo de Serviço de Difusão IP: Aperfeiçoamentos ou Mudanças"**, XIX Simpósio Brasileiro de Redes de Computadores - SBRC'2001, SBC, Florianópolis – SC. Maio de 2001.

[Bro97] G. Brose, **"JacORB: Implementation and Design of a Java ORB"**, Procs. of DAIS'97, IFIP WG 6.1 International Working Conference on Distributed Aplications and Interoperable Systems, Cottbus, Germany, Chapman & Hall, Sep. 1997.

[Dee85] S. E. Deering, D. R. Cheriton, **"Host Groups: A Multicast Extension to the Internet Protocol"**, Internet Engineering Task Force – RFC 966. Disponível em www.ietf.org 1985.

[Dee86] S. E. Deering, **"Host Extensions for Multicast IPing",** Internet Engineering Task Force – RFC 988. Disponível em www.ietf.org 1986.

[Hay98] Mark Hayden, **"The Ensemble System"**, Relatório Técnico, Cornell University, 2001.

[Hun01] F. Hunleth, **"Design and Implementation of the Unreliable Multicast InterORB Protocol in TAO"**, Relatório Técnico, Washington University, St. Louis, 2001.

[Lau00] Lau C. L., J. Fraga, J. Farines, J. R. Oliveira, **"Experiências com Comunicação de Grupo nas Especificações Fault Tolerant CORBA"**, XVIII Simpósio Brasileiro de Redes de Computadores - SBRC'2000, SBC, Belo Horizonte – MG. Maio de 2000.

[Lau01] Lau C. L., J. Fraga, R. Padilha, L. Souza, **"Adaptando as Especificações FT-CORBA para Redes de Larga Escala"**, XIX Simpósio Brasileiro de Redes de Computadores - SBRC'2001, SBC, Florianópolis – SC. Maio de 2001.

[Mos01] L. E. Moser, P. M. Melliar-Smith, P. Narasimhan, R. Koch and K. Berket, **"A Multicast Group Communication Protocol, Engine, and Bridge for CORBA"**, Concurrency and Computation Practice & Experience, vol. 13, no. 7, pp. 579-603, Junho 2001.

[Mel90] P. Melliar-Smith, L. Moser and V. Agrawala, **"Broadcast Protocols for Distributed System"**, IEEE Transactions on Parallel and Distributed Systems, 1(1):17-25, Janeiro 1990.

[OMG00] Object Management Group, **"Fault-Tolerant CORBA Specification V1.0"**, OMG document: ptc/2000-04-04, Disponível em www.omg.org. 2000.

[OMG01] Object Management Group, **"The Common Object Request Broker Architecture Specification V2.6"**, OMG Document. Disponível em www.omg.org, 2001.

[OMG01b] Object Management Group, **"Unreliable Multicast Inter-ORB Protocol Specification"**, OMG document. Disponível em www.omg.org, 2001.

[OMG01c] Object Management Group, **"Extensible Transport Framework Revised Submission"**, OMG document: orbos/01-10-05. Disponível em www.omg.org, 2001.

[OMG02] Object Management Group, **"Objective Interface Systems Extensible Transports for Real-Time CORBA Revised Submission"**, OMG document: mars/02-04-03. Disponível em www.omg.org, 2001.

[Ren95] R. V. Renesse and K. P. Birman, **"Protocol Composition in Horus"**, Dept. of Computer Science of the Cornell University, Março de 1995.

[Sch90] F. B. Schneider, **"Implementing Fault-Tolerant Service Using the State Machine Approach: A Tutorial"**, ACM Computing Survey, 22(4):299-319, Dezembro de 1990.

[Wan01] M. S. Wangham, Lau C. L., C. M. Westphall and J. Fraga. **"Integrating SSL to the JaCoWeb Security Framework: Project and Implementation"**, In: IEEE/IFIP International Symposium on Integrated Network Management, Seattle, NJ:IEEE Press, 2001. p.779-792.