Uma Análise Comparativa das Estratégias de Suporte a Grupo sobre o CORBA

Jorge Ricardo Souza de Oliveira, Lau Cheuk Lung e Joni da Silva Fraga
Laboratório de Controle e Microinformática - LCMI-DAS-UFSC
Campus Universitário - Trindade - Florianópolis - SC
Caixa Postal 476 - CEP 88040-900
e-mail: {jorger, lau, fraga}@lcmi.ufsc.br

Resumo

Este artigo apresenta uma análise comparativa das abordagens para suporte ao processamento de grupo no CORBA. Na literatura são identificadas três abordagens: integração, serviço e interceptação. Cada uma destas abordagens apresentam características que procuram explorar aspectos de desempenho, transparência, conformidade com padrões abertos e facilidade de uso. Na abordagem de integração, o núcleo do ORB é alterado para permitir a execução dos mecanismos de processamento de grupo. Na abordagem de serviço, estes mecanismos são disponibilizados como um serviço CORBA. Finalmente, na abordagem de interceptação, o processamento de grupo é ativado transparentemente por meio de mecanismos interceptores. O estudo comparativo e as discussões são baseados nas implementações desenvolvidas, em um ORB CORBA, e nas análises desempenho destas.

Palavras chave: *sistemas distribuídos, comunicação de grupo, tolerância a faltas, sistemas abertos.*

Abstract

This paper presents a comparative study of approaches to support group processing over CORBA. In the literature, three approaches are identified: integration, service and interception. Each of these approaches has characteristics that aim to explore performance, transparency, open standard accordance and easy use aspects. In the integration approach, the ORB core is changed to allow the execution of group processing mechanisms. In the service approach, these mechanisms are made available as a CORBA service. Finally, in the interception approach, the group processing is activated transparently by interceptor mechanisms. The comparative study presented in this paper, besides discussions over existent group support environments over CORBA ORBs, presents implementations developed and performance analyses over them.

Key words: *distributed systems, group communication, CORBA, fault tolerance, open systems.*

1. Introdução

A comunicação de grupo tem se mostrado um paradigma útil, em sistemas distribuídos, no sentido de dar suporte a aplicações de trabalho cooperativo, mas sobretudo para o processamento replicado por razões de tolerância a falhas. Com isso, a introdução do conceito de grupo em padrões abertos para a programação distribuída tem sido alvo de vários trabalhos de pesquisa, produtos comercias e propostas de padronização. São exemplos destes esforços propostas de adição de mecanismos de processamento de grupo sobre as especificações CORBA os trabalhos apresentados em [3, 10, 15, 17]. Todavia, a OMG ainda não tem definido uma especificação padrão para o suporte a grupo na arquitetura CORBA [16]. Existe somente um primeiro documento *Resquest for Proposal* (*RFP*) requisitando propostas para a padronização de funcionalidades CORBA de suporte a aplicações tolerantes a faltas [19]. Os requisitos estipulados neste documento prevêem o fornecimento de serviços e facilidades CORBA para a construção de aplicações confiáveis. As propostas devem apresentar soluções abertas, não dependentes de protocolos ou ferramentas proprietárias.

Este trabalho tem por objetivo a apresentação e discussão de soluções para a adição de mecanismos de processamento em grupo ao padrão CORBA. Para tal foram identificadas na literatura três soluções básicas, as abordagens de integração [09, 15], de serviço [3, 12] e de interceptação [8, 17].

No decorrer deste trabalho é descrito um estudo comparativo destas abordagens, levantando

questões como: desempenho, transparência, conformidade com o padrão CORBA, flexibilidade e facilidade de uso. Neste sentido, além de discussões a respeito de ambientes de suporte a grupo sobre ORBs CORBA já existentes, são desenvolvidas implementações, sobre as quais são realizadas análises de desempenho. Este estudo faz parte do projeto *GrouPac*, que tem a finalidade de desenvolver um grupo de objetos de serviço para o suporte a aplicações confiáveis [11].

Este artigo está dividido da seguinte maneira. Na seção 2 são definidas as abordagens de integração, de serviço e de interceptação. Na seção 3 são apresentadas as implementações desenvolvidas e análises de desempenho. Na seção 4 são feitos comentários sobre as implementações. Finalmente, na seção 5 são levantadas as conclusões deste trabalho.

2. Abordagens de Suporte a Grupo no CORBA

2.1. Abordagem de Integração

A abordagem de integração consiste na construção ou na modificação de um *middleware* CORBA existente, adicionando mecanismos de processamento em grupo. Nesta abordagem, o *middleware* CORBA é alterado para que as aplicações não possam distinguir objetos simples de grupos de objetos, oferecendo um alto grau de transparência. A idéia principal nesta abordagem é que o processamento de grupo seja suportado por um sistema de comunicação de grupo abaixo do núcleo do ORB. Todas as chamadas que envolvam comunicação ou gestão de grupo são repassadas pelo núcleo do ORB a este suporte de mais baixo nível. Esta abordagem já foi adotada em algumas plataformas CORBA existentes, como o Orbix+Isis [7] e o Electra [13].

Na abordagem de integração, as referências de objeto passam a poder identificar tanto um objeto único, como um grupo de objetos. O ORB é responsável por distinguir estas referências. A figura 2.1 ilustra a execução de uma requisição em um grupo de objetos. No lado do cliente, a requisição quando passada ao ORB, é reconhecida pelo mesmo como uma requisição endereçada a um grupo, sendo convertida em uma chamada de difusão na ferramenta de mais baixo nível que dá suporte à comunicação de grupo. Em seguida, a operação adequada é invocada pelo ORB em cada membro do grupo de objetos servidores. Os resultados do processamento retornam pelo mesmo caminho, porém no sentido inverso. Finalmente, as repostas são coletadas no lado cliente, submetidas a alguma função de consenso e retornadas ao objeto que fez a requisição.

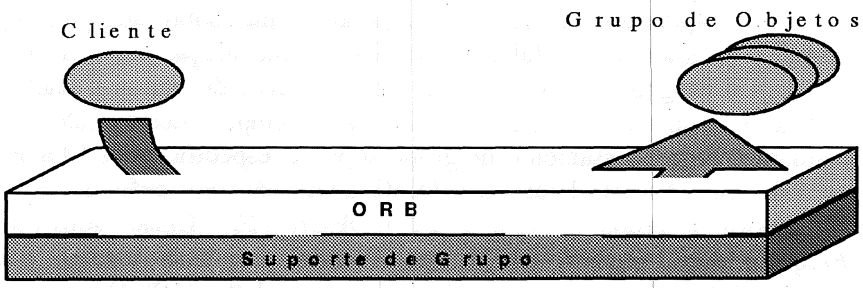

Figura 2.1 - Execução de uma requisição na abordagem de integração

As implementações da abordagem de integração podem fazer uso de adaptadores de objetos disponíveis nos servidores que fornecem acesso às facilidades para a gestão de grupo. Esta possibilidade é permitida pela especificação CORBA. Todavia, nos *middlewares* Orbix+Isis e Electra, o próprio adaptador de objeto básico é estendido, permitindo que todos os objetos do sistema possam tornar-se membros de grupos. Além disso, estes *middlewares* definem outras extensões ao padrão CORBA. No Orbix+Isis, as regras de mapeamento das estruturas definidas em IDL para a linguagem de implementação [15] foram estendidas para gerar os códigos de

gestão e comunicação de grupo. No Electra, as regras de mapeamento foram respeitadas, mas classes da implementação do ORB foram estendidas.

2.2. Abordagem de Serviço

A abordagem de serviço para a adição de suporte a grupo ao ORB está de acordo com a filosofia adotada pela OMG para o acréscimo de funcionalidades ao padrão CORBA. A plataforma básica de comunicação provida pelo CORBA, ou seja, o ORB, suporta apenas os mecanismos para a invocação de métodos de objetos remotos. Funções mais específicas são adicionadas ao ORB na forma de objetos de serviço definidos através de especificações *COSS* (*Common Object Services Specification*) da OMG [17]. Os objetos de serviço servem de *building blocks* para dar suporte ao desenvolvimento de diferentes aplicações.

A idéia básica na abordagem usando objetos de serviço é prover o suporte a grupos de objetos como um conjunto de serviços no topo do ORB, e não como parte do próprio ORB. Dessa forma, o serviço de suporte a grupo deve ser formado por uma coleção de objetos de serviço, que devem estar localizados em diferentes estações da rede. Estes objetos devem ser responsáveis pela gestão de grupos e ainda, pela entrega de mensagens aos objetos membros de grupo, mantendo as propriedades definidas pelo tipo de comunicação de grupo utilizado. A execução de uma requisição segundo a abordagem de serviço é ilustrada na figura 2.2.

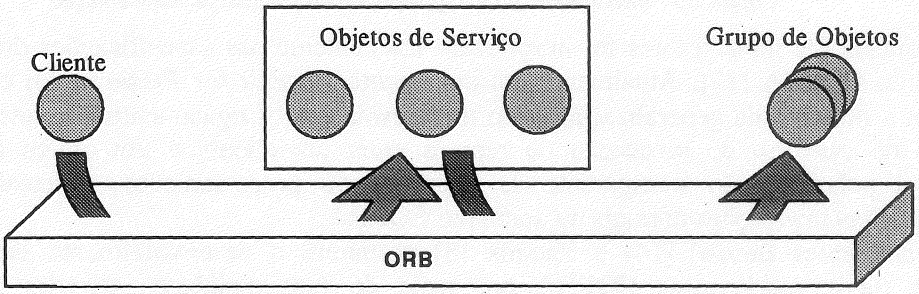

Figura 2.2 - Execução de uma requisição na abordagem de serviço

Como exemplos da abordagem de serviço, pode-se citar o *Object Fault-tolerance Service* (OFS) [9] e o *Object Group Service* (OGS) [3]. O OFS oferece suporte a tolerância a faltas por meio de mecanismos de replicação. Todavia, não é utilizada comunicação em grupo. O OGS oferece suporte a aplicações confiáveis por meio de um serviço de grupo de objetos. Este serviço segue a filosofia da OMG, sendo implementado a partir de objetos de serviço CORBA que podem ser utilizados em outros contextos. Tanto o OFS como o OGS estão de acordo com o padrão CORBA. Neste contexto, estes *middlewares* não utilizam ferramentas de suporte a grupo proprietárias e as regras de mapeamento IDL são respeitadas.

2.3. Abordagem de Interceptação

A abordagem de interceptação consiste no provimento de suporte a grupo em *middlewares* CORBA por meio da utilização de mecanismos de reflexão computacional. A reflexão computacional, ou simplesmente reflexão, consiste na capacidade de um sistema analisar e agir sobre si mesmo, ajustando-se a condições variáveis de seu ambiente [12, 25]. A programação reflexiva, segundo o paradigma de meta-objetos, permite separar o código funcional do não funcional nas aplicações [12]. O código funcional, ou nível base, relaciona-se com as computações no domínio da aplicação. O código não funcional, ou nível meta, é responsável por supervisionar a execução do código funcional. Técnicas de tolerância a faltas têm também sido implementadas refletidas, isto é, separadas dos aspectos funcionais de suas aplicações. Em [2],

são apresentadas discussões sobre o uso da reflexão computacional para implementar modelos de replicação em sistemas distribuídos. Em [5] apresentamos a extensão desta experiência para ambientes CORBA, porém envolvendo mecanismos não suportados pelo padrão CORBA.

A abordagem de interceptação prevê que as mensagens enviadas aos objetos servidores devem ser capturadas e mapeadas em um sistema de comunicação de grupo, de maneira transparente para a aplicação. Para tal, podem ser utilizados estruturas das linguagens de programação, interfaces do sistema operacional ou mesmo mecanismos do próprio ORB, conhecidos como interceptadores. O funcionamento desta abordagem usando interfaces do sistema operacional é ilustrado na figura 2.3.

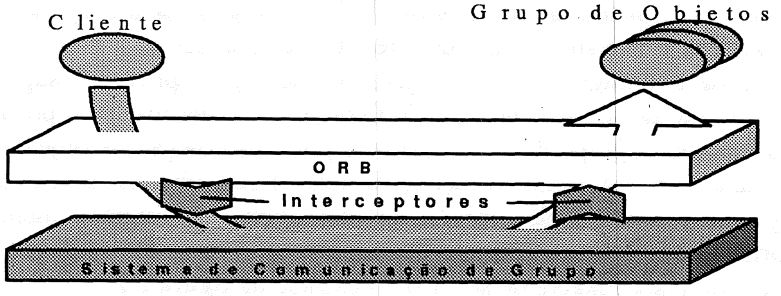

Figura 2.3 - Execução de uma requisição na abordagem de interceptação

O conceito de interceptores foi introduzido inicialmente nas especificações do serviço de segurança do CORBA [17]. Atualmente um documento *Request for Proposal* foi emitido pela OMG com o objetivo da generalização deste mecanismo [18]. Logicamente, um interceptador é interposto no caminho de invocação ou reposta entre um cliente e um objeto alvo, sendo responsável pela ativação transparente de controles ou processamentos especiais às quais estariam sujeitas invocações normais no ambiente CORBA.

Os ambientes Eternal [14] e Phoinix [8] permitem o desenvolvimento de aplicações confiáveis sobre *middlewares* CORBA por meio de funcionalidades ativadas a partir de mecanismos de interceptação. No Phoinix, estes mecanismos consistem em estruturas geradas por um compilador IDL estendido a partir das definições de interface dos objetos CORBA. Já no Eternal, a interceptação das chamadas aos objetos CORBA é realizada por mecanismos do sistema UNIX. Estes mecanismos capturam mensagens IIOP antes delas chegarem ao TCP/IP. Neste contexto, nenhum dos ambientes apresentados utiliza a estrutura de interceptores definida pelo padrão CORBA.

2.4. Considerações Sobre as Abordagens

Este item apresenta uma comparação informal entre as três abordagens, discutindo critérios como: transparência, facilidade de uso, portabilidade, interoperabilidade, conformidade com o padrão CORBA e desempenho. Esta comparação é baseada nas implementações das abordagens, isto é, o Orbix+Isis, Electra, OGS e Eternal. O OFS e o Phoinix são desconsiderados, já que os mecanismos para a tolerância a faltas oferecidos não suportam a comunicação em grupo.

A transparência oculta o processamento em grupo do programador, dando a ilusão de que as invocações são originadas e atendidas por um único objeto. Na abordagem de integração, os clientes não precisam saber que a operação invocada é atendida por um grupo. Todavia, em situações especiais, os clientes podem se beneficiar deste conhecimento. A abordagem por objetos de serviço pode ser utilizada com ou sem transparência. No primeiro caso, os clientes invocam diretamente os serviços oferecidos pelo grupo. Isto é realizado por meio da utilização de esqueletos e interfaces de invocação dinâmicos durante a comunicação entre clientes e servidores.

Nos segundo caso, clientes e servidores devem, respectivamente, invocar e atender operações específicas para a comunicação em grupo. A abordagem de interceptação obriga a transparência. Diferente das outras abordagens, um cliente não pode acessar todas as respostas de uma invocação.

A facilidade de uso é uma consideração importante, já que diminui o tempo de desenvolvimento e manutenção dos programas, tornando-os mais robustos e confiáveis. As abordagens de integração e interceptação apresentam maior facilidade de uso, já que o estabelecimento das estruturas de grupo é automático. A abordagem de serviço apresenta a configuração do suporte a grupo explícita, mas a comunicação em grupo pode ser transparente. Neste sentido, esta abordagem combina mecanismos de configuração flexíveis com suporte a grupo transparente.

A portabilidade implica na independência de ORBs específicos. Em particular, são consideradas a portabilidade do código do suporte a grupo e das aplicações desenvolvidas sobre este suporte. Na abordagem de integração, o código dos mecanismos de suporte a grupo é integrado ao ORB. Além disso, as aplicações desenvolvidas utilizam construções não padronizadas pelo CORBA. Neste sentido, os códigos do suporte e das aplicações não são portáveis. Na abordagem de serviço, tanto o código das aplicações como do suporte a grupo são portáveis, já que não dependem de características da implementação de um ORB específico. Em relação a abordagem de interceptação, mais especificamente no Eternal são utilizados mecanismos do Unix para suportar grupos. Desta forma, os mecanismos de grupo desta abordagem não são portáveis. Todavia, estes mecanismos não são referenciados no código das aplicações, tornando-as completamente portáveis.

A interoperabilidade implica na possibilidade de aplicações em ORBs distintos interagirem. Implementações que fazem uso de sistemas de comunicação proprietários não são interoperáveis. Este é o caso do Electra. O Orbix+Isis combina invocações sobre o Isis e sobre o IIOP. Desta forma, os objetos podem interoperar por meio de invocações IIOP ponto-a-ponto. O Eternal pode escolher quais as requisições devem ser interceptadas, também podendo interoperar sobre o IIOP. A abordagem de serviço utiliza apenas primitivas de comunicação do ORB, sendo completamente interoperável.

A abordagem de integração não está em conformidade com o padrão CORBA, já que as referências de objeto podem identificar grupos. Além disso, estruturas definidas pelo padrão CORBA são estendidas pelo Orbix+Isis e Electra. As abordagens de serviço e interceptação respeitam a especificação CORBA. Os mecanismos de grupo na abordagem de serviço são independentes do núcleo do ORB, sendo definidos por objetos de serviço e suas interfaces IDL. A abordagem de interceptação é completamente independente do ORB, sendo baseada em estruturas do protocolo IIOP.

A abordagem de integração apresenta o maior desempenho, já que os mecanismos de gestão e comunicação em grupo são executados por uma ferramenta especializada. Apesar do suporte a grupo no Eternal também ser baseado em uma ferramenta especializada, o seu desempenho depende do mapeamento das estruturas IIOP para o sistema de grupo. O desempenho da abordagem de serviço é comprometido devido a utilização de mecanismos de comunicação ponto-a-ponto do ORB. Além disso, devido a ser estruturado como uma pilha de serviços definidos por interfaces IDL, a sua execução apresenta uma sobrecarga.

3. Implementação das Abordagens

A seguir são discutidas aspectos de implementação e avaliações de desempenho envolvendo o Orbix+Isis. Estas implementações representam as abordagens de serviço e

interceptação. Todas as três abordagens avaliadas se utilizam da ferramenta Isis. Desta forma, espera-se que os mecanismos de comunicação e gestão de grupo possuam uma sobrecarga semelhante, permitindo que sejam avaliados apenas os processos de ativação destes mecanismos. Em particular, o desempenho das implementações das abordagens de serviço e de interceptação deve variar somente durante o estabelecimento das estruturas de grupo. Enquanto, na abordagem de serviço as próprias aplicações são responsáveis por requisitar a criação destas estruturas, na abordagem de interceptação esta criação é ativada transparentemente.

Para o desenvolvimento das implementações foram utilizados a linguagem Java e o ORB OrbixWeb [6]. Estas escolhas se fundamentam na produtividade da linguagem Java e no fornecimento de estruturas CORBA, como esqueletos e interfaces de invocação dinâmicos, pelo OrbixWeb.

3.1. Implementação da Abordagem de Serviço

A estrutura proposta como plataforma para o desenvolvimento das implementações é apresentada na figura 3.1. Esta estrutura prevê a justaposição de dois ambientes: o ORB e o suporte a grupo. Enquanto as comunicações ponto-a-ponto convencionais trafegam pelo ORB, as comunicações de grupo são implementadas usando estruturas de comunicação separadas. Estas estruturas separadas constituem o suporte a grupo, a interface de grupo e a ferramenta Isis [1].

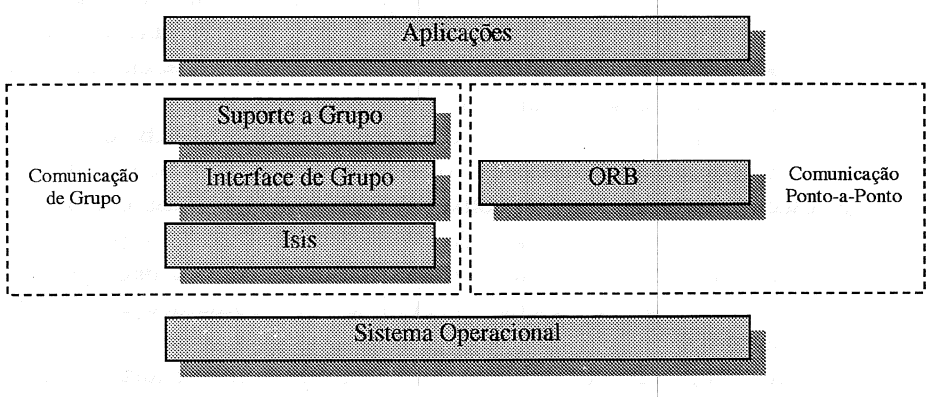

Figura 3.1 – Estrutura da implementação da abordagem de serviço

A ferramenta Isis oferece suporte de mais baixo nível para implementar os serviços de gestão e de comunicação de grupo. A interface de grupo visa livrar a implementação de características da ferramenta Isis. Esta interface consiste de objetos CORBA (objetos de interface) que definem um padrão para o acesso aos serviços do Isis, desacoplando a implementação do suporte a grupo de detalhes do Isis. Finalmente, o suporte a grupo propriamente dito consiste em um conjunto de objetos de serviço, chamados de *proxies*, que fornecem às aplicações uma interface para a utilização dos serviços de grupo. Na figura 3.2 é ilustrada a utilização de objetos *proxies* e interfaces na difusão de uma mensagem. Somente as comunicações entre interfaces clientes e servidores utilizam o Isis. Os demais objetos do modelo se utilizam do ORB em comunicações ponto-a-ponto.

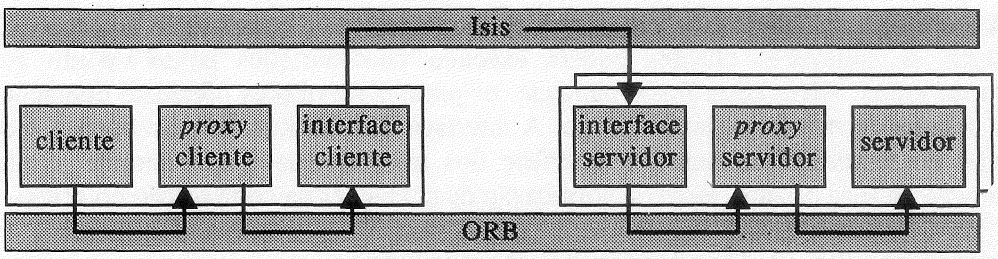

Figura 3.2 – Difusão de mensagem na implementação da abordagem de serviço

Devem ser criados objetos *proxies* e interfaces para cada grupo da qual um servidor deseja participar ou com o qual um cliente deseja comunicar-se. Estes objetos são criados a partir de fábricas[1]. O uso de fábricas é obrigatório, já que a linguagem IDL não define construtores. Na figura 3.3 é ilustrada a criação de *proxies* e interfaces servidores. Inicialmente, o servidor da aplicação requisita a uma fábrica a criação de um *proxy*. Em seguida, este *proxy* transparentemente requisita a outra fábrica a criação de uma interface. A partir de então, o *proxy* e a interface criados podem ser utilizados normalmente pela aplicação. A criação destas estruturas no lado cliente é análoga.

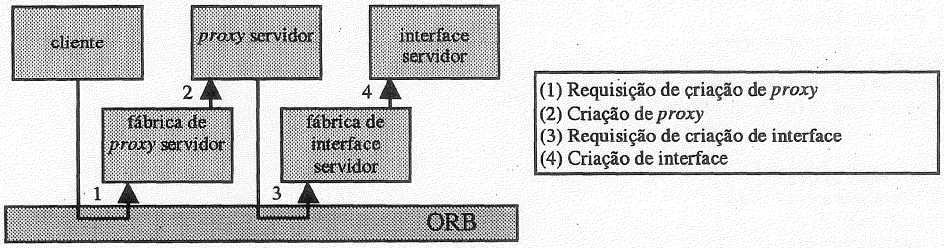

Figura 3.3 – Criação de *proxies* e interfaces servidores

Os *proxies* e interfaces devem ser criados em processos distintos, isto é, eles não devem residir no mesmo processo da fábrica. Cada fábrica é responsável pela criação de apenas um único tipo de objeto. Além disso, cada fábrica deve responder por um domínio. Em particular, espera-se que esteja disponível uma fábrica para cada máquina da rede, o que garante que *proxies* e interfaces não sejam objetos remotos, oferecendo um maior desempenho e confiabilidade às aplicações.

A função principal dos *proxies* é permitir a difusão de requisições dos clientes aos servidores. Esta tarefa é realizada por meio de esqueletos e interfaces de invocação dinâmicos. Em particular, os esqueletos dinâmicos permitem que os *proxies* clientes atendam requisições baseadas na interface do servidor. Por sua vez, as interfaces de invocação dinâmica (*DII: Dynamic Invocation Interface*) permitem que os *proxies* servidores invoquem operações definidas nos membros de grupo. O uso destas estruturas é necessário, já que em tempo de compilação, não é possível determinar as operações do grupo ao qual os *proxies* serão associados.

No lado servidor os *proxies* muitas vezes devem invocar operações nos membros de grupo. Isto acontece durante as mudanças de *view* do grupo e durante as transferências de estado. Neste contexto, os membros de grupo devem ser derivados de uma interface IDL padrão que define estas operações de gestão. Na figura 3.4 é ilustrada esta interface, além das interfaces dos *proxies* clientes e servidores e suas fábricas.

[1] Os objetos fábrica, referenciados em inglês como "*factory objects*", são definidos pela especificação do COSS de ciclo de vida. Em particular, um objeto fábrica é um objeto CORBA comum que é utilizado para criar outros objetos [17].

A única operação disponível em um *proxy* cliente retorna a composição do grupo. Com esta informação as aplicações clientes podem executar comunicações ponto-a-ponto com cada membro do grupo. Além desta funcionalidade, os *proxies* servidores oferecem operações para a entrada e saída de membros de um grupo. A interface das fábricas permite apenas a criação e destruição de *proxies*. Finalmente, a interface dos membros de grupo permite a obtenção e configuração de seus estados, além da notificação de alterações na composição do grupo.

3.2 Implementação da Abordagem de Interceptação

A implementação da abordagem de interceptação permite que os mecanismos de grupo sejam transparentemente adicionados a aplicações clientes e servidores. Esta transparência é alcançada a partir do uso de filtros do OrbixWeb. Estes filtros são associados às aplicações clientes e servidores e se responsabilizam pela comunicação com as fábricas para a criação de *proxies*. Além disso, os filtros fazem com que a entrada e saída de membros de grupo seja automática. Os filtros de processo interceptam todas as requisições e respostas chegando ou partindo de um processo cliente ou servidor, independentemente dos objetos de origem e destino.

```
module GroupService {

  // Group server member
  interface GroupServerMember {
    any getState ();
    void setState (in any state);
    void changeView (in GroupView view);
  };

  // Group server proxy
  interface GroupServerProxy {
    void joinGroup ();
    void leaveGroup ();
    GroupView getView ();
  };

  // Group server proxy factory
  interface GroupServerProxyFactory {
    GroupServerProxy create (in GroupServerMember groupServerMember, in string groupName);
    void destroy (in GroupServerProxy groupServerProxy);
  };

  // Group client proxy
  interface GroupClientProxy {
    GroupView getView ();
  };

  // Group client proxy factory
  interface GroupClientProxyFactory {
    GroupClientProxy create (in string groupName);
    void destroy (in GroupClientProxy groupClientProxy);
  };
```

Figura 3.4 – Interfaces IDL de membros de grupo, *proxies* e fábricas de *proxies*[1]

Na abordagem de interceptação a utilização dos filtros é prevista somente durante a associação de um cliente ou um servidor a um grupo. Neste contexto, os filtros devem interceptar apenas as comunicações entre as aplicações e o serviço de nomes. Segundo a especificação CORBA, cada referência de objeto é associada a um ou mais nomes. O serviço de nomes é utilizado para registrar e recuperar estas referências. A abordagem de interceptação, aproveita

[1] Na descrição destas interfaces estão citadas apenas as operações mais relevantes. Em particular, as operações dos *proxies* invocadas pelas interfaces foram suprimidas.

esta característica do padrão CORBA para definir nomes de grupo. Um nome de grupo identifica um serviço que é prestado por um grupo de servidores. Cada nome de grupo é prefixado pela *string* "grp://". Esta característica visa facilitar a sua identificação. Em especial, um mesmo servidor, deve poder cadastrar-se individualmente ou como um membro de grupo. Desta forma, todas as requisições ao serviço de nomes que especificarem um nome de grupo devem ser interceptadas e manipuladas pelos filtros.

Para obter uma referência a um objeto servidor, os clientes realizam uma requisição ao serviço de nomes. Caso esta requisição referencie um nome de grupo, um filtro deve interceptá-la. Então, este filtro deve entrar em contato com uma fábrica para criar um *proxy* de grupo. Em seguida uma referência deste *proxy* é retornada ao cliente. Neste contexto, o cliente tem a ilusão de que recebeu uma referência para um objeto comum, quando na verdade recebeu uma referência para um *proxy*. Esta situação é ilustrada na figura 3.5.

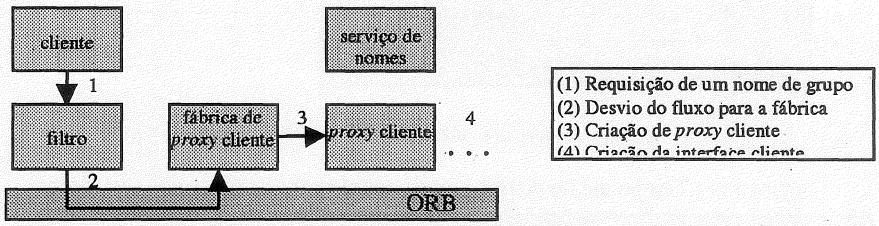

Figura 3.5 – Ativação das estruturas de grupo na abordagem de interceptação

Para cadastrarem-se, os servidores também devem realizar uma requisição ao serviço de nomes. Caso esta requisição especifique um nome de grupo, um filtro deve interceptá-la. De modo similar ao lado cliente, este filtro deve entrar um contato com uma fábrica para criar um *proxy* de grupo. Além disso, o filtro deve invocar a operação de junção ao grupo no *proxy*. A partir de então, o servidor passa a receber requisições de seu *proxy* como se ele fosse um objeto cliente qualquer.

4. Avaliações de Desempenho

As avaliações de desempenho foram realizadas com base em testes envolvendo três estações Ultra Sparc rodando o sistema operacional Solaris 2.5 sobre uma rede Ethernet 10 Mbs. O limite no número de estações se deve a condições da licença da ferramenta Isis disponível. Nestas avaliações de desempenho também deve ser considerado que enquanto no Orbix+Isis, as aplicações são desenvolvidas em C++, nas implementações da arquitetura proposta, as aplicações são desenvolvidas em Java. A linguagem Java 1.1 apresenta um desempenho cerca de dez vezes menor que a linguagem C [4]. Todavia, o impacto do uso da linguagem Java é bem menor, já que os mecanismos de difusão e gestão de grupo são providos pelo Isis. Além disso, um dos fatores mais importantes para o desempenho é o tempo de transmissão na rede.

Na figura 3.6 é apresentado o tempo médio de resposta no Orbix+Isis e nas implementações das abordagens de serviço e interceptação para uma invocação de um serviço prestado por um grupo de objetos. A média foi obtida a partir do tempo de resposta das primeiras mil invocações. Além disso, é ilustrada a evolução dos tempos de resposta em função do número de servidores presentes no grupo. Cada servidor é executado em uma máquina distinta. Como pode ser percebido, o Orbix+Isis possui um desempenho superior. Embora com o aumento do número de servidores presentes no grupo exista um aumento no tempo de resposta, a diferença de desempenho entre as abordagens estudadas permanece constante. Neste contexto, caso o número de servidores seja muito grande, espera-se que a diferença de desempenho seja desprezível. A justificativa para a diferença de desempenho entre estas abordagens é a utilização de

comunicações ponto-a-ponto entre objetos intermediários nas implementações das abordagens de serviço e de interceptação.

A fim de avaliar melhor o desempenho das abordagens, foram levantados tempos de resposta parciais a uma invocação de um serviço prestado por um grupo de objetos. Conforme o ilustrado na figura 3.7, os tempos de resposta parciais envolvem o tráfego entre os processos participantes do serviço de grupo. Neste tráfego são considerados o tempo para o envio de uma requisição e o retorno de seu resultado.

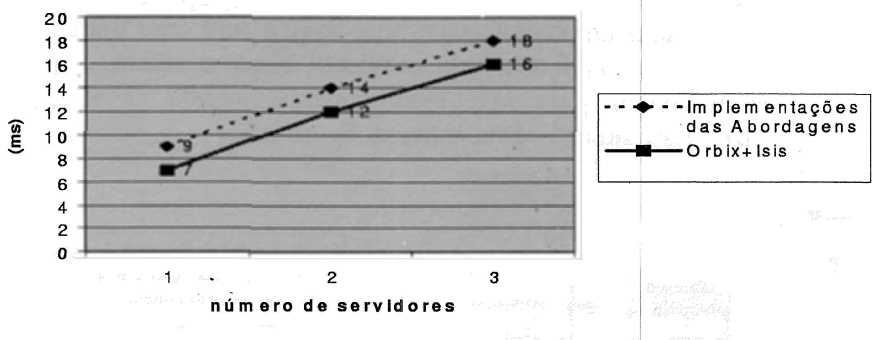

Figura 3.6 – Tempo médio de resposta de acordo com o número de servidores

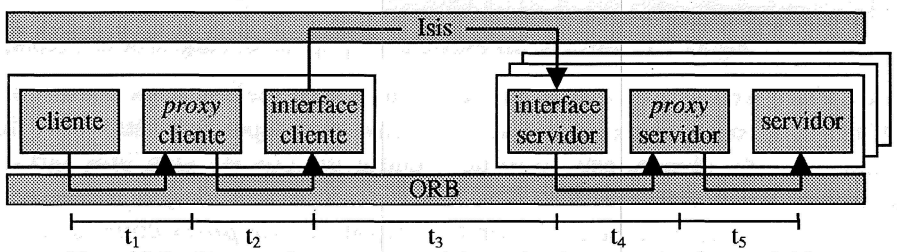

Figura 3.7 – Tempos de resposta parciais nas implementações desenvolvidas

Na figura 3.8(a) é apresentado o impacto de cada tempo parcial na execução de uma invocação a um serviço prestado por um grupo de servidores. Pode ser percebido que o aumento do número de servidores implica apenas no aumento no tempo de tráfego entre as interfaces clientes e servidores. Esta característica é esperada, já que a comunicação entre as aplicações e os *proxies* e entre os *proxies* e as interfaces, envolve apenas tráfego local. Além disso, é somente na comunicação entre as interfaces que ocorre a execução da difusão em grupo. A seguir são apresentadas alternativas de implementação da arquitetura proposta que visam o aumento de desempenho. Em particular, estas alternativas buscam a diminuição do tempo envolvido nas comunicações locais. Neste contexto, estas comunicações deixariam de ser intermediadas pelo ORB.

A primeira alternativa de implementação da arquitetura proposta envolve a eliminação dos processos onde residem as interfaces. Neste contexto, ao invés de entrar em contato com uma fábrica, os *proxies* instanciam diretamente as interfaces. Desta forma, os *proxies* e interfaces deixam de residir em processos distintos, fazendo com que a interação entre estas estruturas passe a ser executada em um mesmo espaço de endereçamento. Esta alternativa não causa nenhum impacto sobre as aplicações clientes e servidores, já que estas aplicações mantém a mesma interface com os *proxies*. Neste contexto, esta alternativa pode ser implementada segundo as abordagens de serviço e interceptação. Na figura 3.8(b) são apresentados tempos médios de resposta desta alternativa. Em particular, o tempo t_{234} envolve as comunicações do *proxy* cliente ao *proxy* servidor e vice-versa. Segundo estas avaliações, pode-se perceber que houve uma considerável diminuição nos tempos de resposta.

	1	2	3
t_1	0,5	0,5	0,5
t_2	0,5	0,5	0,5
t_3	7	12	16
t_4	0,5	0,5	0,5
t_5	0,5	0,5	0,5
t	9	14	18

(a) Implementações propostas

	1	2	3
t_1	0,5	0,5	0,5
t_{234}	7,5	12,5	16,5
t_5	0,5	0,5	0,5
t	8,5	13,5	17,5

(b) Ausência de fábrica de interfaces

	1	2	3
t	8	13	17

(c) Ausência de fábricas

Figura 3.8 – Tempos médios de resposta parciais em variações da arquitetura proposta

A segunda alternativa de implementação da arquitetura proposta envolve tanto a eliminação dos processos onde residem as interfaces, como a eliminação dos processos onde os *proxies* residem. De forma semelhante à alternativa anterior, as aplicações clientes e servidor passam a instanciar diretamente os *proxies*. Esta alternativa, embora também apresente um considerável aumento de desempenho, conforme ilustrado na figura 3.8(c), aproximando-se do alcançado pelo Orbix+Isis, possui algumas desvantagens. Esta alternativa não consiste em uma solução CORBA, já que o código dos *proxies* é disponibilizado às aplicações como parte de uma biblioteca.

Outro fator de desempenho que deve ser observado é o tempo necessário para o estabelecimento das estruturas de grupo, isto é, os *proxies* e interfaces. Na implementação da abordagem de serviço este tempo é de 2 ms, já na abordagem de interceptação são necessários 3 ms. Esta diferença se deve ao uso de filtros na abordagem de interceptação. No caso do Orbix+Isis, as estruturas de grupo são inerentes a qualquer aplicação. Desta forma, não é necessário tempo para a sua criação.

4.1. Considerações sobre as Implementações

Um serviço de suporte a grupo pode ser implementado tanto no topo de um sistema de comunicação de grupo específico, bem como, sobre os mecanismos de comunicação do ORB [3]. Todavia, a última abordagem parece ser mais interessante, já que o uso de um canal separado para as comunicações de grupo implicaria em uma solução proprietária. Uma potencial desvantagem do uso dos mecanismos de comunicação do ORB é o comprometimento do desempenho, uma vez que as comunicações são ponto-a-ponto. Entretanto, a OMG recomenda assumir-se implementações de CORBA eficientes quando projeta-se um serviço [17].

A especificação CORBA define os interceptadores como estruturas que ativam transparentemente um ou mais serviços [17]. Neste sentido, qualquer serviço CORBA pode ser invocado diretamente pelas aplicações ou ativado por um interceptor. A implementação da abordagem de interceptação segue esta diretriz. Desta forma, a abordagem de interceptação consiste apenas em uma evolução da abordagem de serviço.

5. Conclusão

Neste trabalho são discutidas várias soluções para a adição de mecanismos de processamento de objetos em grupo ao padrão CORBA. Estas soluções podem ser agrupadas em três abordagens. Na abordagem de integração o núcleo do ORB é alterado para permitir a execução dos mecanismos de processamento em grupo. Na abordagem de serviço, estes mecanismos são disponibilizados como objetos de serviço e suas interfaces IDL. Finalmente, na abordagem de interceptação, o processamento em grupo é ativado transparentemente por meio de

estruturas de reflexão computacional. Em particular, apenas as abordagens de serviço e interceptação estão em conformidade com o padrão CORBA. Todavia, para apresentarem portabilidade e interoperabilidade, as implementações destas abordagens não devem utilizar sistemas de comunicação em grupo proprietários.

Além disso, neste trabalho é apresentado um estrutura geral para o desenvolvimento das implementações, representando as abordagens de serviço e interceptação, sobre as quais são realizadas análises de desempenho. Estas avaliações apontaram uma sobrecarga mínima em relação à abordagem de integração, representada pelo Orbix+Isis. Todavia, as implementações desenvolvidas são baseadas em mecanismos de difusão e gestão de grupo fornecidos pela ferramenta Isis, consistindo em soluções proprietárias. Desta forma, futuramente pretende-se desenvolver uma solução completamente aberta. Neste contexto, a gestão e a difusão em grupo devem ser desenvolvidas somente a partir de objetos CORBA. Ademais, atualmente as implementações abordam apenas a replicação de servidores, não oferecendo primitivas para a comunicação entre grupos e portanto, não suportando replicação de clientes. Posteriormente, com o desenvolvimento de mecanismos para tal, será possível um grupo de objetos enviar uma requisição a um servidor da mesma forma que um cliente único o faria.

Referências Bibliográficas

[1] BIRMAN, K. & COOPER, R. & JOSEPH, T. A. & MARZULLO, K. MAKPANGOU, M. & KANE, K. & SCHMUCK, F. & WOOD, M. **The Isis System Manual**. Department of Computer Science, Cornell University, setembro de 1990.

[2] FABRE, Jean-Charles & NICOMETTE, Vincent & PÉRENNOU, Tanguy & STROUD, Robert J. & WU, Zhixue. **Implementing Fault Tolerant Aplications using Reflexive Object-Oriented Programming**. 25th IEEE International Symposium on Fault-Tolerant Computing, Passadena-CA, pp. 488-498, junho de 1995.

[3] FELBER, Pascal & GUERRAOUI, Rachid & SCHIPER, André. **A CORBA Object Group Service**. Ecole Polytechnique Fédérale de Lausanne, Computer Science Department, Technical Report, 1997

[4] FLANAGAN, David. **Java in a Nutshell**. Editora O'Reilly, 2a edição, maio de 1997.

[5] FRAGA, Joni & MAZIERO, Carlos & LUNG, Lau C. & LOQUES FILHO, Orland G. **Implementing Replicated Services in Open Systems Using a Reflexive Approach**. 3rd International Symposium on Autonomous Decentralized Systems, pp 273-280, Berlim - Alemanha, abril de 1997.

[6] IONA Technologies PLC. **OrbixWeb Programmer's Guide**, novembro de 1997.

[7] ISIS Distributed Systems, Inc. & IONA Technologies, Ltd. **Orbix+Isis Programmer's Guide**, Document D070-00, 1995.

[8] LIANG, Deron & CHOU, S. C. & YUAN, S. M. **Phoinix: A Fault-Tolerant Object Service in OMA**. 1996.

[9] LIANG, Deron & FANG, Chen-Liang. & YUAN, Shyan-Ming. **A Fault-Tolerant Object Service on CORBA**. submetido a Journal of Systems and Software, 1998.

[10] LISBÔA, Maria Lúcia. **Arquiteturas de Meta-nível**, Simpósio Brasileiro de Engenharia de Software SBES'97, Fortaleza-CE, Brasil, 1997.

[11] LUNG, Lau & FRAGA, Joni da Silva & FARINES, Jean-Marie. ***CosNamingFT* – Um Serviço de Nomes Tolerante a Falhas em Conformidade com o Padrão OMG**. Simpósio Brasileiro de Redes de Computadores – SBRC, 1999.

[12] MAES, P. **Concepts and Experiments in Computational Reflection**, in Proc. OOPSLA 87, pp. 147-155, 1987.

[13] MAFFEIS, Silvano. **Electra 2.0b Programmer's Manual**. Dept. of Computer Science, Cornell University, 1996.

[14] NARASIMHAN, P. & MOSER L. E. & MELLIAR-SMITH P. M. **Exploiting the Internet Inter-ORB Protocol Interface to Provide CORBA with Fault Tolerance**. Proceedings of the 3rd Conference on Object-Oriented Technologies and Systems, junho de 1997.

[15] OMG. **IDL C++ Language Mapping Specification**. OMG Document 94-9-14, 1994.

[16] OMG. **The Common Object Request Broker: Architecture and Specification**. OMG, 1995.

[17] OMG. **CORBAservices: Common Object Services Specification**. OMG, 1995.

[18] OMG. **Portable Interceptor RFP- Request for Proposal OMG** Document Orbos/ 98-05-05, maio de 1998.

[19] Object Management Group. **Fault-Tolerant CORBA Using Entity Redundancy**. RFP orbos/98-04-01, October, 1998.