Experiências com Comunicação de Grupo nas Especificações *Fault Tolerant* CORBA

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

Resumo

Este artigo apresenta um estudo sobre o desempenho e as facilidades de incorporação das diferentes abordagens para suporte a comunicação de grupo no CORBA. Na literatura são identificadas três abordagens: integração, serviço e interceptação. Cada uma destas abordagens apresenta 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: *comunicação de grupo, tolerância a faltas, avaliação de desempenho, sistemas abertos.*

Abstract

This paper presents performance analyses of approaches to support group communication in 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.

Keywords: *group communication, fault tolerance, performance analysis, open systems.*

1. Introdução

Protocolos de comunicação de grupo têm se mostrado um paradigma útil em sistemas distribuídos. Isto é devido à necessidade de dar suporte, por exemplo, às aplicações de trabalho cooperativo e para o processamento replicado por razões de tolerância a faltas. Diante disto, um esforço considerável vem sendo feito no sentido da introdução do conceito de grupo em padrões abertos para a programação distribuída, com propostas de padronização bem como no desenvolvimento de produtos comerciais. São exemplos destes esforços, as propostas de suporte para processamento de grupo nas especificações CORBA (*Common Object Request Broker Architecture*)[3, 8, 11, 14, 15]. Todavia, aOMG (*Object Management Group*) [17], recentemente, liberou uma proposta de especificação revisada do *Fault Tolerant* CORBA (FT CORBA) [20], que define um conjunto de interfaces de serviços e facilidades úteis na implementação de técnicas de replicação em ambientes distribuídos heterogêneos. Na verdade, estas especificações foram geradas a partir de uma submissão envolvendo várias empresas, propondo um conjunto de soluções para suportes à tolerância a faltas em *middleware* CORBA.

No entanto, nestas especificações não existe a definição de interface para a comunicação de grupo, fundamental na implementação de qualquer técnica de replicação em sistemas distribuídos. Segundo o documento [20], as necessidades relacionadas com a comunicação de grupo, para dar suporte às técnicas de replicação, podem ser supridas por

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 5.3(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 SCG. 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.

6. Conclusão

Técnicas de replicação para tolerância a faltas são facilmente implementadas quando baseadas em facilidades de suportes para comunicação de grupo. Isto é evidenciado quando estes suportes fornecem várias primitivas de comunicação envolvendo diferentes tipos de ordenação e acordo sobre as mensagens. A proposta de especificação revisada do *Fault Tolerant* CORBA (FT CORBA) [20], definem um conjunto de serviços essenciais para tolerância a faltas no CORBA. Estes serviços são disponíveis como objetos de serviço COSS com suas interfaces definidas em IDL/CORBA. Entretanto, nenhuma interface é definida para comunicação de grupo nessas especificações. A OMG enfatiza que as necessidades relacionadas com a comunicação de grupo podem ser supridas por ferramentas proprietárias. A especificação só impõe que os serviços de suporte a grupo fornecidos por essa ferramenta sejam transparentes para os objetos da aplicação.

Neste trabalho foram discutidas as abordagens no suporte a grupo em *middlewares* CORBA. 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 de interceptação estão em conformidade com o padrão CORBA.

Além disso, neste trabalho é apresentada uma estrutura geral *(framework)* que permite associar o ORB com qualquer ferramenta de comunicação de grupo existem - neste caso, o Isis. Para o desenvolvimento das implementações, representando as abordagens de serviço e interceptação, ambas se utilizam das facilidades da ferramenta proprietária Isis - o que nas especificações em [20] é permitida. Nas implementações e mais no Orbix+Isis, foram realizadas medidas de desempenho. Estas avaliações apontaram uma sobrecarga mínima das implementadas em relação à abordagem de integração, representada pelo Orbix+Isis. Portanto, as medidas de desempenho demonstram que adotando a abordagem de serviço ou interceptação, na forma de objetos de serviço ou interceptores, respectivamente, encapsulando uma ferramenta de comunicação de grupo proprietária, não impõe um alto custo de desempenho. Além disso, ambas as abordagens apresentadas neste trabalho adotam a extensão IOGR *(Interoperable Object Group Reference)* para referenciar grupos de objetos, o que garante uma melhor transparência, portabilidade e interoperabilidade do sistema.

Presentemente estamos desenvolvendo uma solução mais ao nível de *middleware.* Neste contexto, as funções de gestão e de difusão em grupo são realizadas pela combinação

entre estas abordagens é a utilização de comunicações ponto-a-ponto entre objeto intermediários nas implementações das abordagens de serviço e de interceptação.

A fim de avaliar melhor o desempenho das abordagens, foram levantados tempos d resposta parciais a uma invocação de um serviço prestado por um grupo de objetos. Conform ilustrado na figura 5.2, os tempos de resposta parciais envolvem o tráfego entre os processo participantes do serviço de grupo. Neste tráfego são considerados os tempos para o envio d uma requisição e o retorno de seu resultado.

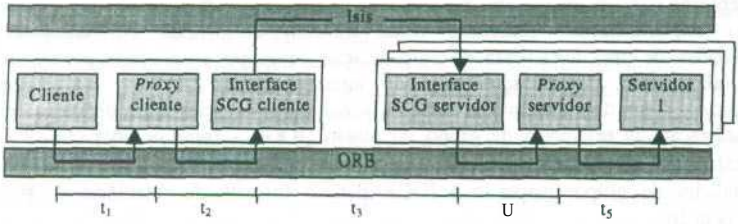

Figura 5.2. Pontos onde os tempos de resposta parciais foram medidos.

Na figura 5.3(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 *t* os *proxies* e entre os *proxies* e as interfaces SCG, 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.

	1	2	3
t,	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 5.3. Tempos médios de resposta parciais (milisegundos) em variações da arquitetura proposta.

A primeira alternativa de implementação da arquitetura proposta envolve a eliminação dos processos onde residem as interfaces SCG. 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 mantêm a mesma interface com os *proxies*. Neste contexto, esta alternativa pode ser implementada segundo as abordagens de serviço e interceptação. Na figura 5.3(b) são apresentados tempos médios (em ms) 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.

potencial desvantagem do uso dos mecanismos de comunicação do ORB é o comprometimento do desempenho, uma vez que as comunicações são ponto-a-ponto através do ORB.

A especificação CORBA define os interceptadores como estruturas que realizam e/ou ativam transparentemente um ou mais serviços [19]. Neste sentido, qualquer serviço CORBA pode ser invocado diretamente pelas aplicações ou ativado por um interceptador. A implementação da abordagem de interceptação segue esta diretriz. Desta forma, a abordagem de interceptação consiste apenas em uma evolução da abordagem de serviço.

As implementações realizadas neste trabalho consistiram em encapsular as funcionalidades de uma ferramenta de comunicação de grupo proprietária por objetos de serviço (abordagem de serviço) ou por interceptadores CORBA (abordagem de interceptação). Além disso, as implementações seguiram as especificações definidas em [20] para a utilização da referência de grupo de objeto (IOGR - *Interoperable Object Group Reference*). Neste caso, as interfaces SCG que negociam diretamente com o Isis, para obter as funcionalidades de grupo, mapeiam a IOGR para o formato de referência de processo utilizado pelo Isis.

5. Avaliações de Desempenho

As medidas 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. 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.

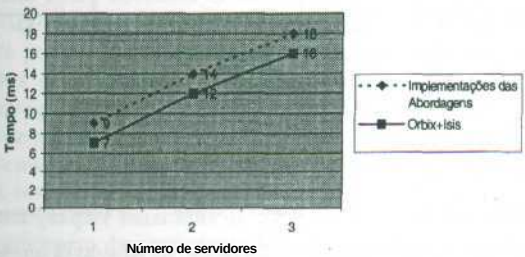

Figura 5.1. Tempo médio **de** resposta de acordo com o número de servidores.

Na figura 5.1 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

4.3. 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/interceptadores 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 sejam automáticas. 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.

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 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 4.5.

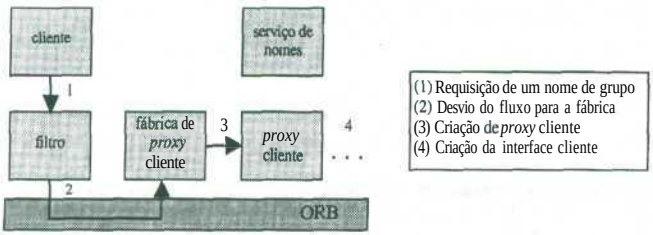

Figura 4.5. Ativação das estruturas de grupo na abordagem de interceptação.

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.

4.4. 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 dificulta o aspecto da interoperabilidade. Uma

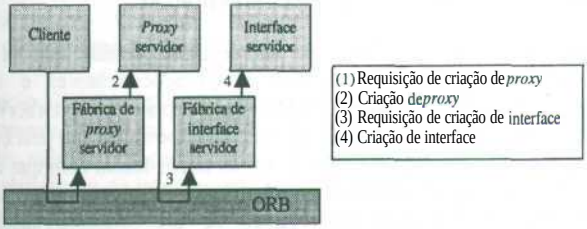

Figura 4.3. Criação de *proxies* e interfaces de servidores.

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 dinâmica. 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.

```
module GroupService {
  interface GroupServerMember {                          // Group server member
    any getstate ();
    void setState (in any state);
    void changeView (in GroupView view);
  h
  interface GroupServerProxy {                           // Group server proxy
    void joinGroup 0;
    void leaveGroup ();
    GroupView getView ();
  };
  interface GroupServerProxyFactory {                    // Group server proxy factory
    GroupServerProxy create (in GroupServerMember groupServerMember, in string groupName);
    void destroy (in GroupServerProxy groupServerProxy) ;
  };
  interface GroupClientProxy {                           // Group client proxy
    GroupView getview ();
  };
  interface GroupClientProxyFactory {                    // Group client proxy factory
    GroupClientProxy create (in string groupName);
    void destroy (in GroupClientProxy groupClientProxy);
  };};
```

Figura 4.4. Interfaces IDL de membros de grupo, *proxies* e fábricas de *proxies*[3].

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 4.4 é ilustrada esta interface, além das interfaces dos *proxies* clientes e servidores e suas fábricas.

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

convencionais trafegam pelo ORB, as comunicações de grupo são implementadas usando estruturas de comunicação separadas. Estas estruturas separadas constituem o suporte de grupo, a interface de grupo e a ferramenta Isis [1].

Figura 4.1 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 confiável. A interface de grupo, representado pelo objeto SCG, 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 dita, consiste de um conjunto de objetos de serviço, chamados de *proxies*, que fornecem às aplicações uma interface para a utilização dos serviços de grupo. Na figura 4.2, 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 o ORB em comunicações ponto-a-ponto.

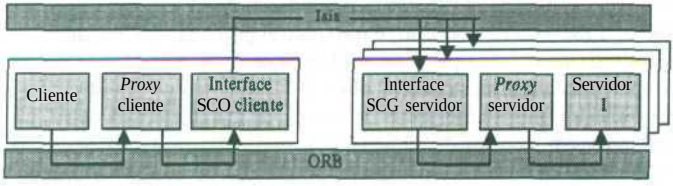

Figura 4.2 - Difusão de mensagem na implementação da abordagem de serviço

Devem ser criados objetos *proxies* e interfaces SCG 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[2]. O uso de fábricas é obrigatório, já que a linguagem IDL não define construtores. Na figura 4.3 é ilustrada a criação de *proxies* e interfaces de 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 SCG. A partir de então, o *proxy* e a interface SCG criados podem ser utilizados normalmente pela aplicação. A criação destas estruturas no lado cliente é análoga.

Os *proxies* e interfaces SCG 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.

[2] Os objetos fábrica, referenciados em inglês como *"factoryobjects"*, 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 [18].

da OMG. Na falta dessas interfaces padronizadas no inicio do projeto, nós tivemos que definir os objetos de serviço a partir de suas interfaces, especificadas em IDL do padrão CORBA. A opção por determinados algoritmos no desenvolvimento desses serviços ou ainda, a construção dos mesmos no topo de uma ferramenta de tolerância a falhas como o ISIS, Horus ou Totem, não é um fator muito importante na implementação desses serviços. Pois, uma vez que a OMG defina seus serviços comuns para tolerância a falhas e as suas interfaces genéricas e padronizadas, as escolhas de implementação dos mesmos não serão um problema dentro de uma perspectiva de sistema aberto. As nossas definições em relação a esses serviços são discutidas a seguir.

Na figura 3, são apresentados os objetos de serviço SI, STE, SM, SDF e SCG - todos fazendo parte do *GroupPac*. Cada um desses objetos possui uma interface horizontal e uma vertical, ambas definidas em IDL/CORBA: a primeira permite as comunicações entre objetos de serviço do mesmo tipo, e a segunda constitui-se no meio pelo qual fornecem serviços para outros objetos.

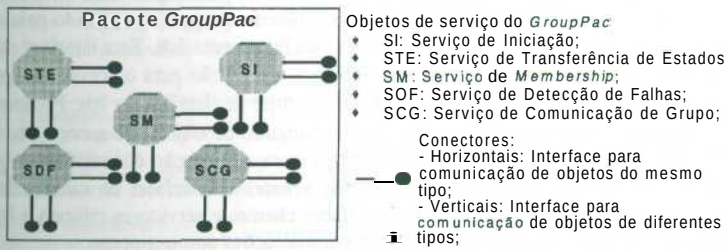

Figura 3. O pacote de objetos de serviço do *GroupPac*.

O objeto SI (Serviço de Iniciação) é responsável pela criação e inicialização dos outros objetos de serviço do pacote. É a partir desse objeto que a configuração necessária ao suporte de uma aplicação replicada é construída. O objeto STE (Serviço de Transferência de Estados) oferece funcionalidades para transferências de estados de um objeto para outro. Esse serviço é usado, por exemplo, em modelos de replicação ou ainda para migração de objetos. O objeto SM (Serviço de *Membership) é* responsável pelo serviço de gestão de grupos de réplicas (grupos de objetos). Essa gestão deve ser transparente à aplicação, exercendo um controle dinâmico nas entradas e saídas de objetos de um grupo pela manutenção de listas atualizadas de seus membros (*membership*). O objeto SDF (Serviço de Detecção de Falhas) envolve um conjunto de procedimentos de detecção de falhas de objetos em um grupo. O SDF opera em conjunto com o objeto SM: quando o SDF detecta um *crash* de um dos objetos do grupo, essa falha é imediatamente reportada ao objeto SM para que esse gere uma nova lista de membros.

Por último, temos o objeto SCG (Serviço de Comunicação de Grupo) que oferece um conjunto de facilidades para comunicação de grupo. Este serviço fornece um conjunto de protocolos confiáveis de comunicação de grupo, com diferentes políticas de ordenação de mensagem (FIFO, Causai ou Total), construídos a partir de alguns objetos de serviço (por exemplo, o SM) e de comunicações simples, ponto a ponto, à nível de ORB. Nessa versão do *GroupPac*, o objeto SCG é implementado de forma a encapsular os mecanismos de comunicação de grupo de um sistema proprietário, neste caso o Isis [1].

4.2. Implementação da Abordagem de Serviço

A estrutura proposta como plataforma para o desenvolvimento das implementações é apresentada na figura 4.1. Esta estrutura prevê a justaposição de dois ambientes: o ORB e o suporte de grupo representado pelo objeto SCG. Enquanto as comunicações ponto-a-ponto

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 de grupo 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 para comunicações de grupo. Além disso, devido a ser estruturado como uma pilha de serviços definidos por interfaces IDL, a sua execução apresenta uma sobrecarga.

4. Implementação das Abordagens

Apresentamos neste item o pacote de serviços de grupo de objetos *GroupPac*, a ferramenta que desenvolvemos para a realização deste trabalho. Em seguida, são discutidos aspectos de implementação e avaliações de desempenho envolvendo as três abordagens de suporte a grupo no CORBA (item 3). Todas as três abordagens avaliadas se utilizam da ferramenta Isis para comunicação de grupo. Desta forma, espera-se que os as implementações das abordagens possuam uma sobrecarga semelhante, permitindo que sejam avaliados apenas os custos do processo de ativação destes mecanismos. Em particular, os desempenhos das implementações das abordagens de serviço e de interceptação devem variar somente durante o estabelecimento das estruturas de grupo. Enquanto que, 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 uma melhor avaliação comparamos o produto comercial Orbix+Isis, que segue a abordagem de integração, com nossas implementações.

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.

4.1. Pacote de Serviços *GroupPac*

O *GroupPac* [11], que foi desenvolvido no nosso laboratório, segue essa linha de soluções abertas definida pela OMG em que qualquer nova funcionalidade acrescida nas especificações CORBA é definida na forma de objeto de serviço comum, mantendo com isso o ORB inalterado. A idéia básica é fornecer um conjunto de serviços e facilidades para a construção de aplicações tolerantes a falhas. Dentro da filosofia de objetos de serviço, o *GroupPac* oferece um conjunto de blocos de construção (*"building blocks"*) - os objetos de serviço - que podem ser arranjados de diferentes formas no sentido de compor diferentes esquemas ou arquiteturas de serviços de aplicação com propriedades de tolerância a faltas. Além disso, esses objetos de serviço de grupo podem ser combinados para dar suporte a aplicações não enfatizando necessariamente a tolerância a falhas. Aplicações distribuídas, tais como *groupware*, ou mais precisamente aplicações de trabalho cooperativo, podem se utilizar os objetos desse pacote para implementar características ou facilitar aspectos da coordenação nessas aplicações.

Os serviços do *GroupPac* devem representar soluções abertas, na forma de interfaces genéricas a serem definidas na evolução dos trabalhos de padronização do grupo de interesse

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.

3.4. 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 comunicação de grupo.

A transparência oculta a noção de 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 é utilizada sem transparência. Os clientes e servidores devem, respectivamente, invocar e atender operações específicas para a comunicação em grupo. No entanto, estas operações podem ser invocadas através de objetos *proxy*, conseguindo assim, um certo grau de transparência dos serviços de 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 considerados 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, apenas, 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

faltas por meio de técnicas de replicação passiva. 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 é 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 modelo de referência OMA *(Object Management Architecture)* [18]. Neste contexto, estes *middlewares* não utilizam ferramentas de suporte a grupo proprietária de baixo nível e as regras de mapeamento IDL são respeitadas.

3.3. 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 [10, 13]. A programação reflexiva, segundo o paradigma de meta-objetos, permite separar o código funcional do não funcional nas aplicações [13]. 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, 12] apresentamos a extensão desta experiência para ambientes 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 [10], interfaces do sistema operacional [15] ou mesmo mecanismos do próprio ORB [5,12], conhecidos como interceptadores. O funcionamento desta abordagem usando interfaces do sistema operacional é ilustrado na figura 2.3.

O conceito de interceptores foi introduzido inicialmente nas especificações do serviço de segurança do CORBA [18]. Atualmente um documento *Request for Proposal* foi emitido pela OMG com o objetivo da generalização deste mecanismo [19]. 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.

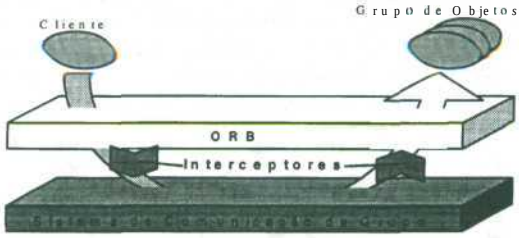

Figura 2.3. Interceptação no nível do sistema operacional.

Os ambientes Eternal [15] 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

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.

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 [16] 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.

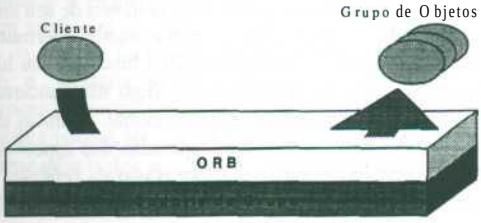

Figura 2.1. Execução de uma requisição na abordagem de integração.

3.2. 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 tal como as especificações *COSS (Common Object Services Specification)* da OMG [18]. 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 podem estar distribuídos 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

- Replicação passiva fria: apenas um membro (o primário) da replicação executa o método requisitado. Em caso de falha do primário, uma réplica *backup* é ativada para assumir o papel de nova primária. A réplica ativada tem o seu estado atualizado partindo do último *checkpoint*, com a execução de requisições extraídas do *log*[1], na ordem dos seus registros de salvamento, após o último *checkpoint;*
- Replicação passiva quente: apenas um membro (o primário) executa o método invocado na interface de serviço. As outras réplicas operam como *backups* onde os métodos invocados não são executados. O estado do primário é transferido para os *backups* periodicamente.

A OMG, como é de sua característica, se limita em definir interfaces de serviço genéricas, tentando atender diferentes abordagens de tolerância a faltas. Por exemplo, protocolos de comunicação de grupo confiável, base ftindamental para a implementação de técnicas de replicação, não são padronizados - como era de se esperar, pela complexidade dos mesmos e pela quantidade de algoritmos envolvidos. A OMG enfatiza, neste caso, o uso de soluções proprietárias (figura 1). Todavia, para garantir um mínimo grau de interoperabilidade, os sistemas de comunicação de grupo devem adotar a IOGR (*Interoperable Object Group Reference*),a referência de grupo de objetos definida em [20]. A IOGR é uma extensão da IOR, de um simples objeto, para uma referência interoperável de grupo de objetos. Um IOGR permite a um cliente referenciar a um grupo de objetos como a uma entidade única.

Na seqüência discutimos as abordagens identificadas na literatura como soluções para suporte de comunicação de grupo no CORBA. Estas abordagens são analisadas diante das recentes especificações FT CORBA.

3. Abordagens de Suporte a Grupo no CORBA

3.1. 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 ORB 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 [14]. Para a implementação do Orbix+Isis, são utilizadas facilidades de gestão e comunicação de grupo fornecidos pela ferramenta Isis [01]. Já o Electra é estruturado de modo que possa ser executado sobre várias ferramentas. Atualmente, existem implementações do Electra sobre o Isis e Horus [22].

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

[1] Mecanismo que registra o conjunto de requisições (mensagens) que chegam no servidor

especificações aconselham um arranjo de detectores de faltas estruturados em forma hierárquica, com diferentes domínios de detecção.

O serviço notificador de faltas é também replicado e tem a função de enviar mensagens de notificação ao gerente de replicação, a partir dos registros de faltas enviados pelos detectores de faltas dos três níveis. O gerente de replicação *(membership)* é responsável pelo gerenciamento dos membros dos objetos replicados. Controla as entradas (criação de réplicas) e saídas (normal ou por falta) de réplicas de um grupo de objetos. Na criação de novas réplicas, o objeto de serviço gerente de replicação usa o objeto fábrica genérica (um COSS) para criar novas cópias do objeto servidor da aplicação. O objeto fábrica genérica negocia com os objetos fábrica locais para a criação de uma nova réplica nas diferentes estações de um sistema distribuído. Esse processo de criação utiliza o serviço de *logging* e *checkpoint*, que são mecanismos para registro e atualização de estados, que a partir de uma réplica identificada como primária (objeto servidor 1), faz as transferências de estado para as novas réplicas que se juntam a replicação.

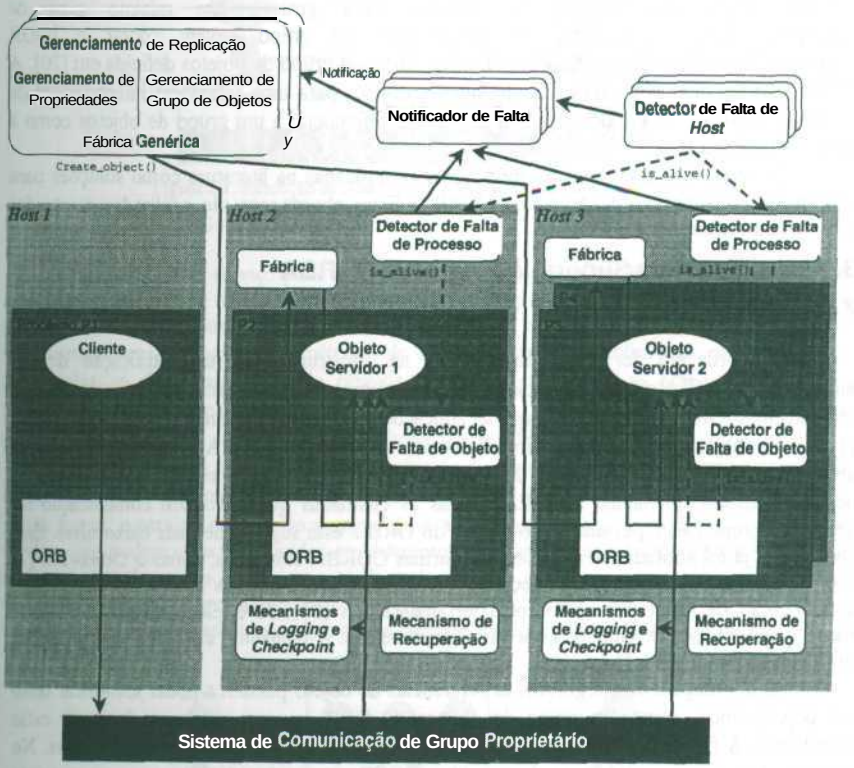

Figura 1. Arquitetura de tolerância a falta CORBA.

Além disso, as especificações definem, também, as técnicas de replicação que podem ser suportadas pela arquitetura FT CORBA, assim definidas:

- Replicação ativa: todos os membros executam independentemente o mesmo método invocado na interface de serviço. Neste caso, eventuais falhas são mascaradas: as réplicas não faltosas produzem o resultado requerido adaptada à técnica de ajuste usada na coleta das respostas do servidor replicado;

uma ferramenta proprietária de comunicação de grupo. A especificação só impõe que os serviços fornecidos por essa ferramenta sejam transparentes aos objetos da aplicação,

Este trabalho tem por objetivo apresentar e discutir as soluções para a adição de suporte de comunicação de grupo em *middlewares* CORBA [17]. As discussões seguem no sentido de apresentar soluções que integrem um ORB/CORBA com uma ferramenta proprietária que suporte serviços de comunicação de grupo. Para tal foram identificadas na literatura três soluções básicas: as abordagens de integração [7, 14], de serviço [3, 9, 11] e de interceptação [5, 8, 12, 15]. As comparações realizadas entre estas abordagens consideram questões como: desempenho, transparência, conformidade com o padrão CORBA, flexibilidade e facilidade de uso. Neste sentido, além de discussões sobre ORBs CORBA com suportes a grupo, já existentes, são desenvolvidas implementações destas abordagens, visando o estudo comparativo.

Estes trabalhos fazem parte do projeto *GroupPac*, que tem a finalidade de desenvolver um conjunto de objetos de serviço para o suporte à aplicações confiáveis em sistemas abertos [5, 11, 12, 21]. A abordagem adotada nesse projeto, definida como híbrida, consiste em combinar as abordagens de serviço e de interceptação. Utiliza mecanismos de filtros/interceptores em nível de aplicação para invocação dos serviços de gerenciamento e comunicação de grupo.

Este artigo está dividido da seguinte maneira. Na seção 2 é apresentada uma descrição resumida das especificações de tolerância a faltas no CORBA. Na seção 3 são definidas as abordagens de integração, de serviço e de interceptação. Na seção 4 são apresentados o *GroupPac*, as implementações desenvolvidas e os comentários sobre as implementações. Na seção 5 são feitas as avaliações de desempenho. Finalmente, na seção 6 são levantadas as conclusões deste trabalho.

2. As especificações *Fault Tolerant* CORBA

Os trabalhos de introdução da tolerância a faltas nos padrões CORBA (FT CORBA) são recentes e, seguindo os calendários da OMG, só foi liberada uma proposta de especificação revisada em dezembro de 1999 [20]. Estas especificações definem um conjunto de serviços essenciais para a construção de aplicações confiáveis em ambientes abertos. Nessa primeira especificação foi apresentado um conjunto de interfaces e protocolos para gerenciamento de replicação. gerenciamento de faltas. gerenciamento de recuperação e *logging* e interoperabilidade. No gerenciamento de replicação são definidas as propriedades de tolerância a faltas e os objetos de serviço para gerenciamento de propriedade, gerenciamento de grupo e objetos fábrica, os quais são responsáveis pela criação remota de objetos réplicas. Para gerenciamento de faltas são definidas as interfaces de detecção de faltas, notificação de faltas e análise de faltas. Finalmente, no gerenciamento de recuperação e *logging* são definidas os mecanismos para transferência de estado de objeto e recuperação de réplicas faltosas. Entretanto, essas especificações deixam em aberto alguns pontos importantes relacionados à tolerância a faltas, permitindo, deste modo, extensões proprietárias.

Na figura 1, é apresentada a arquitetura de tolerância a faltas do CORBA. Nela são apresentados os vários objetos de serviço (COSS), que fornecem no *middleware*, funcionalidades básicas para a construção de servidores tolerantes a faltas. Segundo as especificações, o serviço de detecção à faltas é dividido em três níveis, detectores de faltas em nível de objeto, processo e *host*. Os detectores de faltas nos diferentes níveis são baseados em mecanismos de *timeout*. O detector de falta em nível de *host* se utiliza de replicações. Isto se explica pelo fato de quanto maior o número de réplicas maior a probabilidade de se detectar, de forma precisa, um membro faltoso. Em termos de sistemas de larga escala, as

de objetos de serviço. 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] K. P. Birman, **"The Process Group Approach to Reliable Distributed Computing"**, Technical Report Tr91-1216, Cornell University Computer Science Department, Ithaca, N.Y., July 1991.

[2] J. Fabre, V. Nicomette, T. Pérennou, R. J. Stroud and Z. Wu, **"Implementing Fault Tolerant Applications using Reflective Object-Oriented Programming"**, Proceedings of the 25th IEEE International Symposium on Fault-Tolerant Computing, Pasadena (CA), June 1995.

[3] P. Felber, **"The CORBA Object Group Service - A Service Approach to Object Groups in CORBA"**, PhD. Thesis, École Polytechnique Fédérale de Lausanne, Lausanne, 1998.

[4] D. Flanagan, **"Java in a Nutshell**. Editora O'Reilly, 2a edição, maio de 1997.

[5] J. Fraga, C. Maziero, Lau C. L. and O. Loques, **"Implementing Replicated** Services **in Open** Systems **Using a Reflective Approach",** Proceedings of the 3th IEEE International Symposium on Autonomous Descentralized Systems - ISADS 97, Berlin - Germany, April 1997.

[6] IONA Techonologies, Ltd. **"OrbixWeb Programmer's Guide"**, 1997. www.iona.com.

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

[8] D. Liang, S. C. Chou, S. Yuan, **"Phoinix: A Fault-Tolerant Object Service in OMA"**. 1996.

[9] D. Liang, C. Fang, S. Yuan, **"A Fault-Tolerant Object Service on CORBA",** submitted to the Journal of Systems and Software, 1998.

[10] M. L. Lisboa, **"Arquiteturas de Meta-nível"**, Simpósio Brasileiro de Engenharia de Software SBES'97, Fortaleza-CE, Brasil, 1997.

[11] Lau C. L., J. S. Fraga, J. Farines, M. Ogg, A, Ricciardi, **"CosNamingFT - A Fault-Tolerant CORBA Naming Service"** 18* IEEE Symposium on Reliable Distributed Systems - SRDS'99, Lausanne, Suiça, October 1999.

[12] Lau C. L., J. S. Fraga, C. Maziero, **"MetaFT - A Reflective Approach to Implement Replication Techniques in CORBA"** 19th International Conference of the Chilean Computer Science Society - SCCC' 99, IEEE Computer Society, Talca, Chile November 1999.

[13] P. Maes, **"Concepts and Experiments in Computational Reflection"**, OOPSLA 87 Proceedings, pp. 147-156, October 1987.

[14] Maffeis, S. **"Adding Group Communication and Fault-Tolerance to CORBA",** In Proceedings of the 1995 USENIX Conference on Object-Oriented Technologies (Monterey, CA, June 1995), USENIX.

[15] P. Narasimhan, L. E. Moser, P. M. Melliar-Smith, **"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.

[16] Object Management Group, **"IDL** C++ **Language Mapping Specification",** Document 94-9-14, 1994.

[17] Object Management Group, **"The Common Object Request Broker 2.0/IIOP Specification",** Revision 2.0, OMG Document 96-08-04, 1996.

[18] Object Management Group, **"CORBAservices: Common Object Services Specification", OMG** Document March, 1997. www.omg.org.

[19] Object Management Group, **"Portable Interceptor"**, RFP- Request for Proposal OMG Document Orbos/98-05-05, maio de 1998.

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

[21] J. S. Oliveira, Lau C. L., J. S. Fraga, **"Uma Análise Comparativa das Estratégias de Suporte a Grupo sobre o CORBA",** XXV Conf. Latinoamericana de Informática - CLEI'99, Asunción, Paraguay, Set. 1999.

[22] Robbert V. Renesse and Kenneth P. Birman, **"Protocol Composition in Horus"** Dept. of Computer Science of the Cornell University, Mar 1995.