GroupPac: Um *Framework* para Implementação de Aplicações Tolerantes a Falhas

Lau Cheuk Lung, Joni da Silva Fraga, Luciana Moreira Sá de Souza, Ricardo Sangoi Padilha
Laboratório de Controle e Microinformática - LCMI-DAS-UFSC
Campus Universitário - Trindade - Florianópolis – SC - Caixa Postal 476 - CEP 88040-900
e-mail: {lau, fraga, luciana, padilha}@lcmi.ufsc.br

Resumo

Nos últimos anos, a área de sistemas distribuídos têm avançado na direção da padronização aberta. Muitas aplicações tolerantes a falhas estão seguindo o paradigma de orientação a objetos e considerando o CORBA como uma alternativa de se adequar a sistemas abertos. No entanto, as especificações do CORBA inicialmente não apresentaram uma visão muito clara em termos de conceitos e de necessidades de suporte para a introdução da noção de grupo, e especificamente para tolerância a falhas. Devido a isto, o padrão CORBA tem sido objeto de extensão para garantir o suporte para grupo. Este artigo apresenta como proposta um *framework* para suporte a aplicações tolerante a falhas em sistemas abertos usando abstrações de grupo.

Palavras chave: tolerância a falhas, objetos de serviço, CORBA e sistemas abertos.

Abstract

Recently, Distributed Systems have been moving forward ope systems. In this way, many fault-tolerant applications are following the object-oriented paradigm and using CORBA specifications. However, the CORBA specifications are not very clear in terms of concepts for supporting fault toletance or group abstractions. So, some works have been proposed for providing group processing in the OMA archetecture. In the literature, three approaches are identified for that: integration, service and interception approaches. This paper presents a framework proposed for supporting group abstractions, based in the service and interception approaches. This framework is founded in some Common Object Services develloped following the OMG principles.

Keywords*: fault-tolerant, object services, CORBA e open systems.*

1. Introdução

Nos últimos anos, a área de sistemas distribuída tem avançado na direção da padronização aberta. Esta tendência vem no sentido de combater soluções proprietárias devido aos altos custos de desenvolvimento de sistema que ela traz, por exemplo, custos provenientes da dificuldade de manutenção de software, interoperabilidade e portabilidade. Atualmente, muitas aplicações tolerantes a falhas estão seguindo o paradigma de orientação a objetos e considerando o CORBA [OMG 96] como uma alternativa de se adequar a sistemas abertos. Todavia, a OMG ainda não tem definido uma especificação padrão no que se refere a tolerância a falhas na arquitetura CORBA. Existe somente uma proposta de especificação revisada no sentido de prover funcionalidades CORBA para a construção de aplicações tolerantes a falhas. 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.

Comunicação de grupo tem se mostrado um paradigma útil em sistemas distribuídos, no sentido de dar suporte às aplicações, e sobretudo no que se refere a processamento replicado por razões de tolerância a faltas. Com isso, a introdução do conceito de grupo em padrões abertos para a programação distribuída tem sido alvo de vários trabalhos de pesquisa, produtos comerciais e propostas de padronização.

A partir do estudo publicado em [ISIS 93] surgiram diversas experiências no sentido da inclusão em *middleware* CORBA suporte a grupo de objetos. Em particular, podemos citar o Orbix+Isis [IONA 95], Electra [Maffeis 95], OFS [Sheu 98], Phoinix [Liang 96], OGS [Felber 98] e o Ethernal [Naras 97]. Este texto tem por objetivo inicial a apresentação e discussão de soluções para a adição de mecanismos de suporte a grupo no CORBA. Para tal, são identificadas na literatura três soluções básicas para esse problema, identificadas como as abordagens de integração, de serviço e de interceptação.

Este artigo apresenta o projeto e a implementação do pacote de serviços *GroupPac* para a inclusão de tolerância a falhas no CORBA, seguindo a linha de soluções abertas preconizadas pela OMG. Este pacote de serviços consiste de um conjunto de objetos de serviço (ou componentes) que são especificados segundo os requisitos apresentados na *COSS* (*Common Object Services Specification*) do padrão OMG. De acordo com estes requisitos, os objetos de serviço do *GroupPac* servem como blocos de construção para a implementação de técnicas de tolerância a falhas ou de aplicações orientadas a grupo (ex: aplicações de trabalho cooperativo). Esses objetos de serviço podem ser combinados constituindo-se em *frameworks* que permitem a implementação de diferentes esquemas de tolerância a falhas. A solução adotada no *GroupPac* consiste em combinar alguns aspectos da abordagem de serviço e de interceptação oferecendo, deste modo, uma melhor transparência dos serviços de grupo à aplicação. No sentido de mostrar as potencialidades desse pacote de serviços, é discutido o *framework* que implementa um suporte de grupo para replicações ativas. As soluções apresentadas nesse artigo visam demonstrar que o *GroupPac* consegue atender aos requisitos relacionados a portabilidade, interoperabilidade, reusabilidade e conformidade com o padrão OMG, além de ser uma solução não dependente de ferramentas de comunicação de grupo proprietário.

O artigo apresenta na seção 2 as abordagens presentes na literatura envolvendo o suporte a grupo no CORBA. Na seção 3, o pacote *GroupPac* tem descrito seus objetos de serviço. Na seção 4 é discutido o *framework* correspondente a um modelo de replicação ativa, constituído a partir dos objetos de serviço do *GroupPac*. Na seção 5 são descritos aspectos da implementação do *framework* constituído a partir do *GroupPac*. Por fim, nas seções 6 e 7 são feitos as considerações gerais e conclusão deste trabalho.

2. As Abordagens de Suporte a Grupo no CORBA

Apresentamos neste item uma descrição dos aspectos conceituais das abordagens identificadas na literatura referentes a inclusão de suporte a grupo em *middlewares* CORBA. Essas abordagens são divididas em: integração, serviço e interceptação. A abordagem de integração consiste na construção ou na modificação de um *middleware* CORBA existente, adicionando mecanismos de processamento em grupo ao ORB, tornando-os partes integradas. 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 são repassadas pelo núcleo do ORB a este suporte de mais baixo nível. Algumas plataformas CORBA existentes, como o Orbix+Isis [IONA 95] e o Electra [Maffeis 95] são exemplos dessa abordagem de integração.

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 serviços comuns definidos através de especificações *COSS* (*Common Object Services Specification*) da OMG [OMG 97]. Os objetos de serviço servem de *building blocks* para dar suporte ao desenvolvimento de diferentes tipos de aplicações.Como exemplos da abordagem de serviço, pode-se citar o *Object Fault-tolerance Service* (OFS) [Sheu 98] e o *Object Group Service* (OGS) [Felber 98].

A abordagem de interceptação prevê que as mensagens enviadas aos objetos servidores devem ser capturadas e mapeadas em um sistema de comunicação de grupo, separado do *middleware* CORBA, de maneira transparente para a aplicação. Uma vez que as mensagens são capturadas do ORB, esta abordagem não requer a modificação do *middleware* CORBA utilizado. Na concretização dessa captura, podem ser realizadas em nível de aplicação, interfaces do sistema operacional ou mesmo mecanismos do próprio CORBA.

O desvio ou captura em nível de aplicação é bastante comum na literatura [Joshi 97, Fabre 95, Chiba 93]. A idéia consiste em desenvolver em nível de aplicação meios para redirecionar uma invocação de método. A abordagem de interceptação no provimento de suporte a grupo em *middleware* CORBA pode ser interpretada como uma forma de reflexão computacional [Maes 97]. Protocolos meta-objeto [Kiczales 91] é a maneira usual para implementar a abordagem de interceptação, desviando a requisição e implementando a comunicação de grupo como um acréscimo de funcionalidade que se executa a nível de aplicação. Em [Fabre 95], são apresentadas discussões sobre o uso de protocolos meta-objetos para implementar modelos de replicação em sistemas distribuídos. Em [Fraga 97, Lau 97 e Fabre 98] é estendida esta experiência com protocolos meta-objetos implementando modelos de replicação em ambientes CORBA

2.1. Discussão 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 dinâmica durante a comunicação entre clientes e servidores. No 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 as 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. O Pacote de Serviços *GroupPac*

O *GroupPac* [Lau 98], que está sendo desenvolvido atualmente no nosso laboratório, segue essa linha de soluções abertas definida pela OMG onde 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. Esse pacote de serviços para objetos distribuídos é baseado nos modelos e conceitos definidos pela OMG para objetos de serviço [OMG 97]. Conceitualmente, dentro dessa 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 falhas. 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 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 da OMG. Na falta dessas interfaces padronizadas, 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

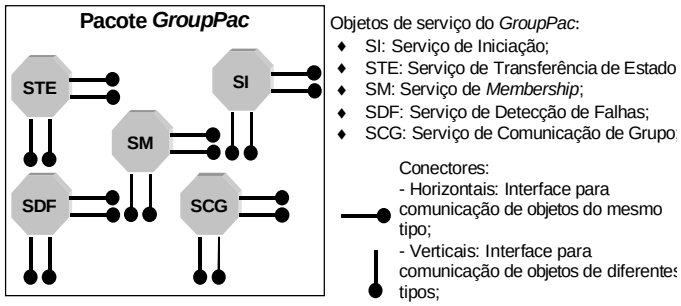

Figura 4. O pacote de objetos de serviço do *GroupPac*.

serão um problema dentro de uma perspectiva de sistema aberto. As nossas definições em relação a esses serviços são discutidas a seguir.

Na figura 4, 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.

O objeto SI (Serviço de Iniciação) é responsável pela criação e iniciação 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, Causal ou Total), construídos a partir de alguns objetos de serviço (por exemplo, o SM) e de comunicações simples, ponto a ponto, em nível de ORB.

Os serviços descritos acima podem ser combinados permitindo a implementação de diferentes esquemas de tolerância a falhas, sem depender de soluções proprietárias, como em [Maffeis 95], [IONA 95] e [Naras 97]. Um experimento implementado inicialmente, usando o *GroupPac*, foi o serviço de nomes replicado - *CosNamingFT* [Lau 99]. Esse serviço de nomes segue as especificações CORBA. A técnica de replicação usada nesse serviço é o primário/*backup* (técnica de replicação passiva), construída a partir dos objetos de serviço definidos no *GroupPac*. Na seqüência deste texto, apresentamos um *framework* que através da combinação dos serviços do *GroupPac* permite que se ofereça suporte para técnicas de replicação ativa.

4. O *Framework* para Replicação Ativa Usando o *GroupPac*

Os objetos de serviço do *GroupPac* são combinados de forma a implementar a técnica de replicação ativa para dar suporte a tolerância a falhas. A técnica de replicação ativa define que todas as réplicas não faltosas do grupo são ativas: recebem, executam e respondem a todas requisições dos clientes. Por ter todas as réplicas ativas executando o mesmo conjunto de requisições é necessário que o conjunto de réplicas tenha um comportamento determinista que é conseguido assegurando propriedades de acordo e ordem sobre as requisições recebidas [Schneider 90]. A abordagem de réplica ativa, no sentido do modelo de máquinas de estado, é capaz de tolerar todo o espectro de faltas (*crash*, omissão, temporização, valor e arbitrária) [Schneider 90]. Neste trabalho nos limitaremos a faltas por *crash*.

O suporte de grupo necessário para garantir o determinismo das réplicas ativas é montado a partir dos objetos de serviço do *GroupPac*. A nossa proposta de suporte de grupo combina características das abordagens de serviço e de interceptação, apresentadas no item 2 desse texto. Deste modo, os aspectos de gestão de réplicas são providos por um conjunto de objetos de serviço em nível do ORB, segundo a

abordagem de serviço. Aspectos relacionados com a comunicação de grupo seguem a abordagem de interceptação onde conceitos como o de interceptor, definido pela OMG [OMG 98a], são usados.

A vantagem nessa combinação de aspectos dessas abordagens é que, uma vez configurado o suporte de comunicação, os grupos ficam totalmente transparentes para seus clientes. Na essência poderemos dizer que todo o suporte a grupo estará disponível na forma de objetos de serviço comum e que alguns desses serviços serão ativados de forma transparente, através do uso de interceptores.

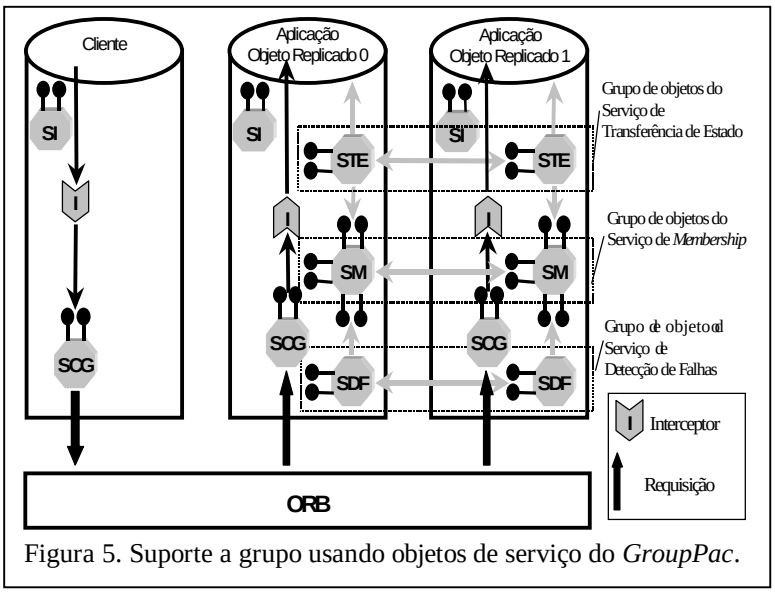

Figura 5. Suporte a grupo usando objetos de serviço do *GroupPac*.

Os objetos de serviço do *GroupPac* vão formar o suporte de grupo em que cada objeto replicado da aplicação ocupa junto com as réplicas dos objetos SI, STE, SM, SDF e SCG do nosso *framework* o mesmo espaço de endereçamento (um único processo UNIX). A figura 5 ilustra o uso dos objetos do *GroupPac* no suporte a grupo. Na falha de um destes objetos é considerado como falha da réplica do todo (*crash* da estação). Os serviços STE, SM, SDF e SCG, são dispostos na forma de grupos de objetos de serviço. Por exemplo, levando em conta as conexões horizontais da figura 5, os objetos SM de cada réplica formam o grupo que fornece o serviço de *membership*.

4.1. Serviço de Gestão de Grupo Replicado

Para dar suporte ao modelo de replicação ativa é necessário um serviço de gestão que forneça informação de quais réplicas estão operacionais e que também controle as entradas e saídas dinâmicas das réplicas do grupo. A combinação dos grupos de serviço SM e SDF suprem essas necessidades. O serviço de detecção de falhas implementado pelo grupo SDF tem a finalidade de detectar falhas de *crash* de réplicas. Esse serviço implementa um protocolo com premissas de comunicações ponto-a-ponto, confiáveis e assíncronas entre os pares comunicantes. Na figura 6, cada objeto do grupo SDF envia mensagens do tipo "você está ativo ?", periodicamente (período T(s)), ao membro anterior da ordenação em anel virtual do grupo. Ao não receber resposta de seu antecessor dentro de um prazo pré-definido, uma réplica R_i considera a possibilidade do *crash* de seu antecessor R_{i-1}, devendo então ativar o método `leave_group` do objeto SM da réplica R0. Essa ativação indica a suspeita em relação a réplica R_{i-1}.

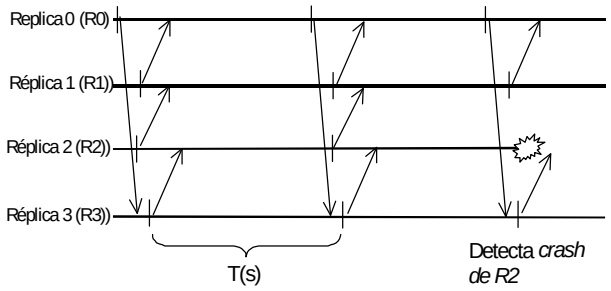

Figura 6. O serviço de detecção de falhas.

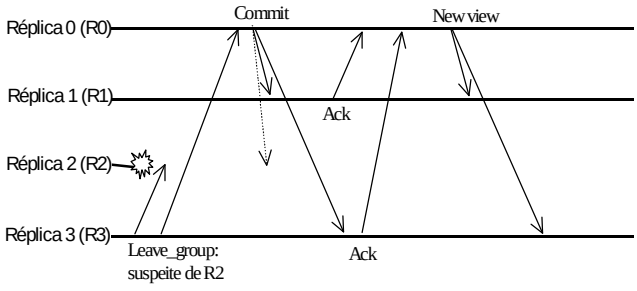

Figura 7. Situação: *crash* da réplica R2

O serviço de *membership* é então executado (ou seja, os objetos do grupo SM são ativados). O protocolo implementado nesse serviço é de decisão centralizada envolvendo interações entre as réplicas em duas fases [Ricciardi 91]. A coordenação na obtenção de uma nova lista de membros (*new view*) é sempre centralizada no objeto SM da réplica de menor *rank* (no caso acima, o R0). A figura 7 mostra um exemplo de funcionamento desse protocolo. O objeto SDF da réplica R3 ao detectar o *crash* da réplica R2 envia a mensagem "suspeite de R2" para o objeto SM da réplica R0. Neste ponto, o protocolo de *membership* é ativado e a réplica R0 difunde uma mensagem *commit* para detectar quem ainda continua no grupo, as réplicas que estão ativas devem dar um reconhecimento a esta mensagem (*Ack*). Após um prazo de espera pré-definido, o primário produz uma nova lista de membros a partir dos *Ack* recebidos [Renesse 95] [Ricciardi 91].

4.2. Serviço de Transferência de Estado

O serviço de transferência de estado (STE) do *GroupPac* é também incorporado às réplicas da aplicação, para garantir que cada nova réplica no grupo tenha seu estado consistente em relação as outras. Na figura 8, são apresentados os passos executados através desse serviço para a transferência de estado. Uma nova réplica ao tentar se juntar a um grupo deve invocar o método `join_group` na réplica R0. A referência de objeto da réplica R0 é obtida através do método `resolve` do objeto *CosNamingFT* – servidor de nomes do sistema [Lau 99]. A partir desta invocação os objetos SM de cada réplica do grupo, que compõem o serviço de *membership* em si, interagem para decidir pela entrada dessa nova réplica (*commit* e *ack*). Se a nova réplica é aceita, o serviço de transferência de estado é ativado. Esse serviço consiste na obtenção do estado da réplica mais antiga do grupo (R0), usando o método `get_state`, e transferi-lo para a nova réplica usando o método `set_state` (figura 8). Só assim, a nova lista de membros (new view) é instalada.

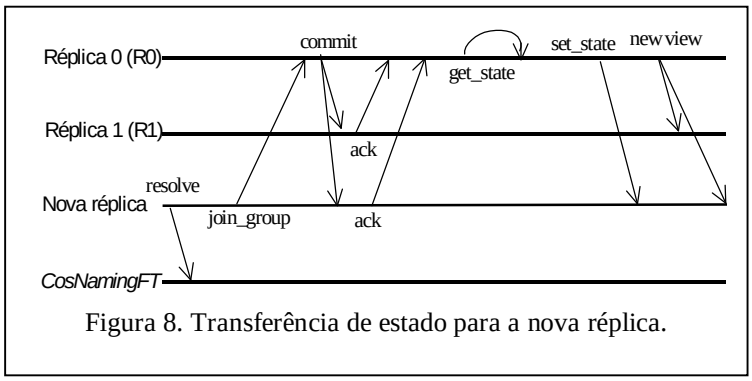

Figura 8. Transferência de estado para a nova réplica.

4.3. Serviço de Comunicação de Grupo (SCG)

Esse serviço através de suas interfaces deve prover acesso a comunicação confiável de grupo, com diferentes políticas de ordenação de mensagem (FIFO, Causal ou Total). A construção desse serviço é feita a partir de outros objetos de serviço do *GroupPac* (por exemplo, o SM e o SDF,) e de comunicações confiável, ponto-a-ponto, via ORB.

A implementação da abordagem de interceptação, no que se refere a comunicação de grupo, permite que os mecanismos necessários sejam transparentemente acionados não envolvendo as aplicações clientes e servidoras. Esta transparência é alcançada a partir de mecanismos que desviem as requisições do cliente. O uso de interceptor, mecanismo definido nas especificações CORBA, é introduzido neste artigo como a solução que consideramos a mais adequada às recomendações da OMG. O interceptor captura a requisição de cliente e a redireciona para o grupo de objetos de serviço SCG (serviço de comunicação de grupo, figura 9).

Na figura 9, a invocação de um cliente é desviada por um interceptor I e redirecionada para o serviço SCG. Os SCGs de cada objeto (cliente e grupo de servidores replicados) interagem de forma a alcançar o acordo e a ordem das mensagens antes de entrega-las aos interceptores que por fim, enviam-

na à aplicação. Para realizar o *multicast* o SCG do lado cliente acessa o serviço de nomes (*CosNamingFT*) para obter a referência de objeto do SM da réplica R0 (a mais antiga do grupo). Com isso, o SCG do cliente pode requisitar a lista de membros atual, com suas respectivas referências de objeto, para realizar o *multicast* no grupo replicado.

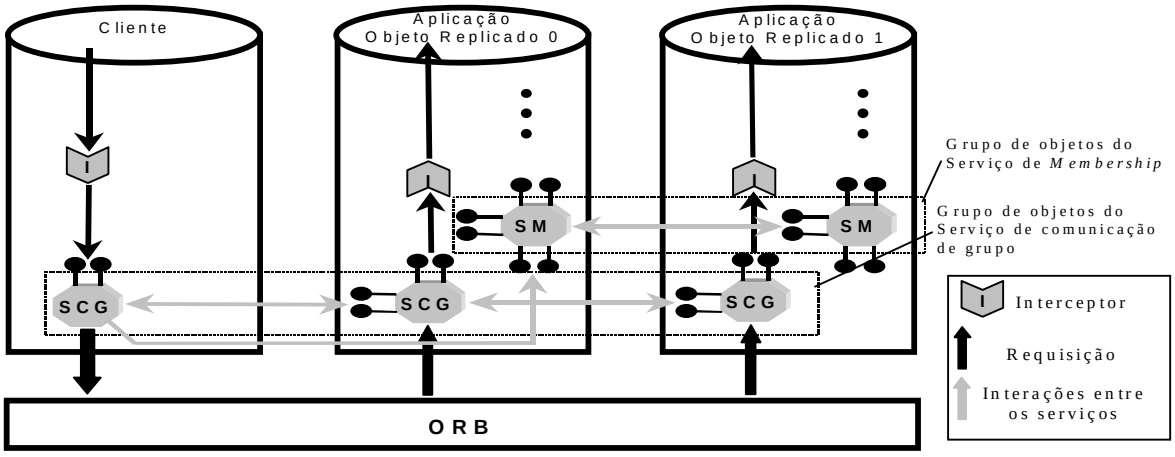

Figura 9. O serviço de comunicação de grupo.

5. Implementação

Discutiremos nesta seção alguns aspectos relacionados com a implementação do *framework* proposto. Todos os serviços deste pacote são implementados na linguagem Java, o que traz a vantagem da portabilidade, e suas interfaces especificadas segundo o padrão IDL/OMG. As implementações foram realizadas na plataforma de desenvolvimento *OrbixWeb* [IONA 97]. Esse *middleware* foi completamente desenvolvido na linguagem Java, seguindo as especificações do CORBA 2.0. Os objetos do *OrbixWeb* se comunicam usando o protocolo IIOP, também padronizada pela OMG.

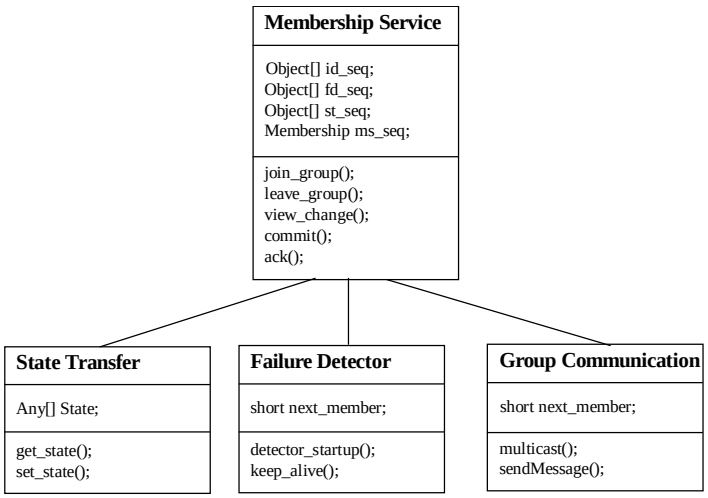

Figura 10. O serviço de comunicação de grupo.

As IDLs dos serviços do *GroupPac* são apresentados na figura 11. Cada uma dessas interfaces será representada por um objeto de serviço em tempo de execução. Cada réplica do objeto de aplicação é mapeada juntamente com os objetos de serviço necessários para o suporte a grupo em processamento. Como cada serviço do *GroupPac* é um conjunto de objetos CORBA, as comunicações entre os mesmos são também através do ORB. Na nossa implementação, no sentido de facilitar, a gestão das referências de objeto que envolve esses serviços, é definido como repositório das mesmas o serviço de *membership* (cada objeto SM que compõem este grupo mantém uma cópia do conjunto dessas referências na estrutura `AllServRefs`). O acesso a essas referências de objeto é local e garantido através de herança da interface *MembershipService* (figura 10). Com isso, cada objeto de serviço desejando realizar uma interação com seus pares complementares de grupo podem fazer através da obtenção da *IOR* (*Interoperable Object Reference*) junto ao objeto SM localizado no seu processo réplica.

```
module GroupPac {
  typedef sequence<any> State;
  typedef sequence<string> MembersCrash_seq;
  // Membership service IDL interface - SM
  interface MembershipService {
    struct ServiceRefs {
      Object fd;
      Object st;
      MembershipService ms;
    };
    struct AllServRefs {
      sequence<string> id_seq;
      // id: Identifier
      sequence<Object> fd_seq;
      // fd: Failure Detector Reference
      sequence<Object> st_seq;
      // st: State Transfer Reference
      sequence<MembershipService>ms_seq;
      // ms: Membership Reference
    };
    boolean join_group(in ServiceRefs servRefs,
                       out string rank);
    boolean leave_group(in MembersCrash_seq
                        membersCrashId);
    void view_change(in short newRank, in short
                     new_view_number);
    boolean ack(in string memberAck);
  };
  // Failure Detector IDL interface - SDF
  interface FailureDetector: MembershipService {
    void keep_alive();
  };
  // State Transfer IDL interface - STE
  interface StateTransfer: MembershipService {
    void get_state(in Object obj, in string
                   stg);
    void set_state(in short id, in State state);
  };
};
```

Figura 11. A IDL dos servicos utilizados do *GroupPac*.

A estrutura `ServiceRefs` é utilizada pelo objeto SI no momento em que uma nova réplica solicita a entrada no grupo replicado. Após iniciar os objetos de serviço, o objeto SI invoca o método `join_group` do objeto SM enviando a estrutura `ServiceRefs`, que contém as referências de cada objeto de serviço (fd, st e ms da figura 11), pertencente ao processo solicitante. Após o protocolo de *membership* ter sido executado e decidido que o novo processo pode ser inserido no grupo, o objeto SM da réplica R0 difunde o novo `AllServRefs` para todos os objetos SM do grupo invocando o método `view_change`. O método `leave_group` é invocado no SM da réplica R0 pelos objetos do grupo de detecção de falhas (SDF) na detecção de um *crash* de uma réplica do grupo (seção 3).

A implementação dos objetos detetor de falhas (SDF) se fundamenta basicamente no método `keep_alive.` Cada membro do grupo SDF invoca esse método no membro anterior da seqüência de anel virtual do grupo. A detecção de *crash* ocorre quando uma exceção do CORBA indicando falha de comunicação é sinalizada.

Na figura 12, é apresentada a definição da interface do serviço de comunicação grupo SCG. Esta interface oferece dois métodos `multicast` e `sendMessage`. O método `multicast` é invocado pelo interceptor do cliente para solicitar ao SCG a difusão da requisição ao grupo replicado. Dentro deste método é implementada a operação de obtenção da referência de objeto do SM, através do servidor de nomes. Uma vez de posse da lista de membros atual a difusão (*multicast*) da requisição para o grupo replicado é realizada através da invocação do método `sendMessage` em cada membro do grupo replicado. Outras trocas necessárias para as propriedades de acordo e de ordem sobre a requisição, entre objetos SCG, são também realizadas usando o método `sendMessage.` Os parâmetros utilizados pelos métodos são do tipo lista de `any` (sequence Any) padrão OMG o que permite uma ampla variedade de combinação de dados na requisição. A requisição é montada segundo a abordagem de invocação dinâmica (DII).

```
// Group Communication IDL interface - SCG
interface GroupMulticast {
  typedef sequence<any> Request;
  typedef sequence<any> Reply;
  Reply multicast(in Request request);
  Reply sendMessage(in Request request);
};
```

Figura 12. IDL do serviço de comunicação de grupo

Um serviço de suporte a grupo, como um conjunto de objetos de serviços comuns, pode ser implementado tanto no topo de um sistema de comunicação de grupo específico, bem como, sobre os mecanismos de comunicação do ORB [Felber 98].

Nós tivemos uma experiência anterior onde implementamos o conjunto de serviços de gestão (SM, SDF e STE) e de comunicação de grupo (SCG), usando o suporte oferecido pela ferramenta ISIS. As discussões referentes a essa experiência estão

presentes em [Lau 00]. Todavia, esse tipo de implementação apresenta a desvantagem da dependência em relação a uma ferramenta específica. As nossas soluções atuais em relação ao *GroupPac* envolve a implementação completa dos serviços, usando só recursos do *middleware* CORBA. Portanto os algoritmos citados no item 3, referentes aos objetos SM, SDF e STE foram implementados. Em relação ao SCG, estamos implementando o *Abcast* [Birman 91] que centraliza a coordenação do *commit* de uma requisição no SCG do cliente, garantindo propriedades de acordo e ordem total sobre as requisições difundidas no grupo. 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.

Na implementação dos interceptores foram usados os mecanismos de filtros presentes no OrbixWeb. Os filtros do OrbixWeb, embora funcionem de forma bastante semelhante aos interceptores definidos pela especificação CORBA, possuem uma desvantagem. Estes filtros, diferente dos interceptores, não podem ser transparentemente associados aos objetos CORBA. Neste contexto, cada aplicação deve criar seu próprio filtro. Esta característica diminui um pouco a transparência da abordagem de interceptação. Entretanto, a criação de um filtro ainda é bem mais simples do que o acesso direto a todos os serviços de grupo.

6. Considerações Gerais

Na literatura são encontrados diversos trabalhos de pesquisa e mesmo produtos que tratam da inclusão de mecanismos para tolerância a falhas na arquitetura CORBA. Além disso, existe um grupo de interesse especial em tolerância a falhas, formada recentemente na OMG, com o objetivo de estender o padrão. A RFP (*Request for Proposal*) [OMG 98] editada por este grupo apresenta um conjunto de requisitos desejáveis para o suporte as aplicações tolerantes a falhas na arquitetura CORBA. Muitos desses requisitos ainda não são satisfeitos por soluções clássicas encontradas na literatura. A incompatibilidade maior está na necessidade de se manter as propriedades de sistema aberto. Mesmos os esforços encontrados na literatura, como o Electra e o Orbix+Isis, exemplos da abordagem de integração, que apesar de oferecerem um alto grau de transparência e de desempenho, não são completamente compatíveis com as especificações da OMG – não permite, por exemplo, a interoperabilidade com outros ORB.

Abordagens que se baseia em objetos de serviço comum (COSS) para implementar mecanismos e suportes de tolerância a falhas estão mais de acordo com os princípios da OMG. As funcionalidades necessárias são adicionadas como objetos de serviço e não representam em modificações do ORB, não apresentando qualquer modificação semântica na comunicação normal entre clientes e servidores. As soluções sendo apresentadas como objetos de serviço com interfaces genéricas definidas pela OMG devem atender as expectativas de soluções abertas. O *GroupPac* que está sendo desenvolvido atualmente no nosso laboratório, segue essa linha de soluções abertas. Esse pacote de serviços, conceitualmente, segue essa filosofia de objetos de serviço como um conjunto de blocos de construção que podem ser arranjados de diferentes formas no sentido de compor diferentes esquemas ou arquiteturas de serviços de aplicação com propriedades de tolerância a falhas. É importante ressaltar que esse *framework* foi proposto no sentido de atender aos requisitos apresentados no RFP da OMG e que deverá refletir a evolução dos trabalhos do grupo de interesse criado na OMG para tratar com as padronizações referentes a tolerância a falhas nas especificações CORBA.

Os exemplos presentes na literatura que seguem essa mesma linha são, de alguma maneira, limitados. O OFS [Sheu 98] oferece meios na forma de objetos de serviço comum para que se implemente serviços de aplicação replicados tomando como base técnicas de replicação passiva. Porém nenhum suporte para comunicação de grupo necessário na implementação de técnicas de replicação

ativa é apresentado no OFS. No OFS os objetos de serviço são definidos com um fim específico, não apresentando a possibilidade de uso dos mesmos em diferentes esquemas de tolerância a falhas. Se tomarmos o OGS como exemplo, em termos de suporte a grupo, esse *middleware* é completamente compatível com o padrão CORBA. No entanto, como exposto em [Felber 98], aspectos de tolerância a falhas não foram devidamente tratados.

Por último, na abordagem de interceptação, o sistema Eternal apresenta uma solução em que é dependente das funcionalidades do Unix e de um suporte de grupo proprietário o que dificulta aspectos de portabilidade. Já no Phoinix é proposta uma extensão da IDL do CORBA para geração de *proxys* para comunicação de grupo. O que de fato afeta o aspecto de portabilidade e interoperabilidade com outras plataformas.

7. Conclusão

Nesse trabalho, os objetos de serviço do *GroupPac* são arranjados no sentido de formar um suporte de grupo em um ambiente aberto. A nossa proposta de grupo combina características das abordagens de serviço e de interceptação, apresentadas no item 2 desse artigo. Todo o suporte a grupo está disponível na forma de objetos de serviço comum sobre o ORB e alguns desses serviços, envolvidos com a comunicação de grupo, são ativados de forma transparente, através do uso de interceptores. Isto é, todas as invocações clientes para os objetos da aplicação são desviadas pelos interceptores para um serviço de comunicação de grupo. O uso de interceptores garante um alto grau de transparência em relação aos serviços de grupo do *GroupPac* e um alto grau de portabilidade, uma vez que os interceptores utilizados são mecanismos do próprio ORB. Em relação a interoperabilidade, o *framework* proposto se utiliza apenas das primitivas de comunicação do ORB (IIOP), o que o torna completamente interoperável com outras plataformas CORBA.

A solução apresentada neste trabalho é bastante similar aos apresentados em [Fraga 97] e [Lau 97]. Naqueles trabalhos, os mecanismos de tolerância a falhas eram implementados segundo os preceitos da reflexão computacional. O paradigma reflexivo usado seguia o conceito de meta-objetos [Kiczales 91]. A gestão do modelo de replicação definidas nos protótipos não era implementada usando facilidades ou serviços de objeto CORBA mas sim, faziam parte da aplicação na forma de meta-objetos. O modelo de implementação de replicações se mostrou bastante flexível, mudar de técnica de replicação se resumia a simples troca de meta-objetos. As soluções adotadas naquele modelo também estavam fundamentadas em suportes de grupo proprietários.

Bibliografia

[Birman 91] 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. Submitted to Comm. ACM.

[Fabre 95] 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.

[Fabre 96] M. Killijian, J. Fabre, J. Ruiz-Garcia, S. Chiba, **"A Metaobject Protocol for Fault-Tolerant CORBA Application"**, Proceedings of 17th IEEE Symposium on Reliable Distributed Systems (SRDS'98), West Lafayette (USA), 20-23 Octobre 1998.

[Felber 98] P. Felber, "**The CORBA Object Group Service – A Service Approach to Object Groups in CORBA**", PhD. Thesis, École Polytechnique Fédérale de Lausanne, Lausanne, 1998.

[Fraga 97] 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.

[Fritzke 98] U. Fritzke J., J-M. Farines, F. Bachmann, J. Fraga, **"Projeto e Implementação de uma Aplicação *Groupware* com Editor Cooperativo de Objetos Gráficos e *Talk* Multi-Usuário em Ambientes Distribuídos Heterogêneos"**, 16 Simpósio Brasileiro de Redes de Computadores, Rio de Janeiro, Maio 1998.

[IONA 95] Isis Distributed Systems Inc., IONA Techonologies, Ltd. **"Orbix+Isis Programmer's Guide"**, 1995. Document D070-00.

[IONA 97] IONA Techonologies, Ltd. "**OrbixWeb Programmer's Guide**", 1997. www.iona.com.

[ISIS 93] Isis Distributed Systems Inc., Ithaca, N.Y, "**Object Groups: A Response to the ORB 2.0 RFI**", April 1993.

[Joshi97] R. K. Joshi at al., "**Message Filters for Object-Oriented Systems**", Software-Practice and Experience, vol. 27(6), 677-699, 1997.

[Kiczales 91] G. Kiczales, J. des Rivières, D. G. Bobrow, "**The Art of the Metaobject Protocol**", MIT Press, 1991.

[Lau 97] Lau C. L., J. Fraga, C. Maziero, **"Uma Abordagem Reflexiva Usando Suporte de Grupo para Implementar Técnicas de Replicação em Ambientes Heterogêneos"**, Anais do VII Simpósio de Computadores Tolerantes a Falhas SCTF 97 - SBC - Campina Grande, PB, Julho 1997.

[Lau 98] Lau C. L., **"*GroupPac*: Object Services to Group Processing"** Departamento de Automação e Sistemas – UFSC, Setembro, 1998.

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

[Lau 00] Lau C. L., J. S. Fraga, J. Farines, J. R. S. Oliveira, **"Experiências com Comunicação de Grupo nas Especificações Fault Tolerant CORBA"**, Anais do XVIII Simpósio Brasileiro de Redes de Computadores, Belo Horizonte – BH, Maio 2000.

[Liang 96] D. Liang, S. C. Chou, S. Yuan, **"Phoinix: A Fault-Tolerant Object Service in OMA"**. 1996.

[Maes 97] P. Maes, **"Concepts and Experiments in Computational Reflection"**, OOPSLA 87 Proceedings, pp. 147-156, October 1987.

[Maffeis 95] 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.

[Naras 97] 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.

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

[OMG 97] Object Management Group, "**CORBAservices: Common Object Services Specification**", OMG Document March, 1997. www.omg.org.

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

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

[Renesse 93] Robbert V. Renesse, **"A MUTS Tutorial"** Dept. of Computer Science of the Cornell University, Mar 1993.

[Renesse 95] Robbert V. Renesse and Kenneth P. Birman, **"Protocol Composition in Horus"** Dept. of Computer Science of the Cornell University, Mar 1995.

[Ricciardi 91] A. Ricciardi and K. Birman**, "Using Process Groups to Implement Failure Detection in Asynchronous Systems"**. In Tenth ACM Symposium on the Principles of Distributed Computing . August, 1991.

[Schneider 90] F. B. Schneider, **"Implementing Fault-Tolerant Service Using the State Machine Approach: A Tutorial",** ACM Computing Survey, 22(4):299-319, Dec 1990.

[Shapiro 86] M. Shapiro, **"Structure and Encapsulation in Distributed Systems: The Proxy Principle"**, 198-204 IEEE, 1986.

[Sheu 98] G. Sheu, Y. Chang, D. Liang, S. Yuan, W. Lo, **"A Fault-Tolerant Object Service on CORBA"**, Proceedings of the IEEE Symposium Reliable Distributed Systems – SRDS 98, 1998.