Gerenciamento de Filiação em *Middleware* Tolerante a Faltas*

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

[1]DAS - Departamento de Automação e Sistemas
UFSC - Universidade Federal de Santa Catarina
Florianópolis - Santa Catarina

[2]PPGIA - Programa de Pós-Graduação em Informática Aplicada
PUC-PR - Pontifícia Universidade Católica do Paraná
Curitiba - Paraná

{alchieri,neves,fraga}@das.ufsc.br,lau@ppgia.pucpr.br

Abstract. *FT-CORBA is one of the most important standards for fault-tolerant middleware. It defines a set of services to manage the membership of object group replicas in the CORBA architecture. Even after almost one decade of research on how to implement this standard, still, there is no system that supports FT-CORBA without relying on single point of failure. This paper presents an architecture to implement the FT-CORBA standard that does not require any centralized service (single point of failure) and supports application and infra-structure objects replication. An implementation of this architecture in the* GROUPPAC *system is also presented. To validate our design, several experiments were carried. These experiments showed the costs of these mechanisms in various fault-loads.*

Resumo. *O FT-CORBA é um dos mais importantes padrões para middleware tolerante a faltas. Este padrão define uma série de serviços para gerenciamento de grupos de objetos (réplicas) na arquitetura CORBA. Mesmo após quase uma década de pesquisas relacionadas a formas de se implementar e suportar esse padrão, ainda hoje não existe um sistema que proveja uma infra-estrutura FT-CORBA livre de pontos únicos de falha. Este trabalho apresenta um esquema para concretização do padrão FT-CORBA que não recorre a pontos únicos de falhas, suportando replicação de objetos de aplicação e dos serviços fornecidos pela infra-estrutura, bem como sua implementação no sistema* GROUPPAC. *Visando validar este esquema, alguns experimentos que medem o impacto dos mecanismos em diversas situações de falhas são apresentados.*

1. Introdução

A crescente dependência por parte da sociedade de sistemas distribuídos cada vez mais complexos, que executam em sistemas heterogêneos, tem levado ao desenvolvimento de uma série de tecnologias e padrões que permitem uma maior simplicidade no desenvolvimento e facilidade de integração destes sistemas. As plataformas de *middleware* baseadas em objetos distribuídos proporcionam uma simplificação da programação de sistemas distribuídos através do acesso a métodos disponibilizados por objetos ativos em diferentes processos na rede. Dentre as arquiteturas de objetos distribuídos destaca-se o CORBA

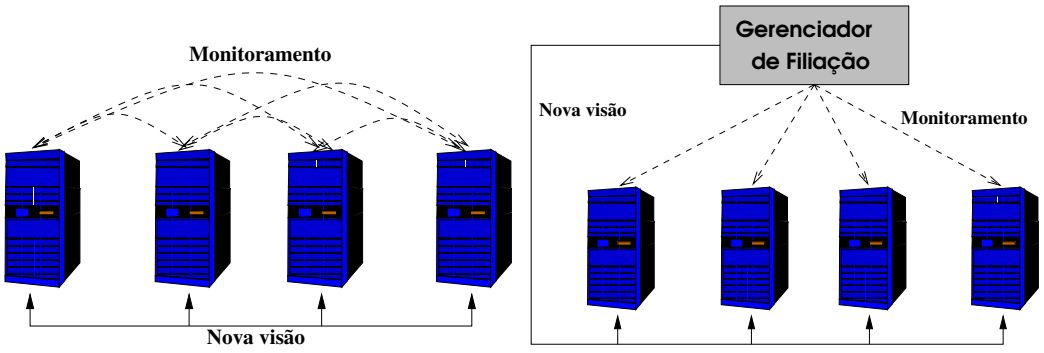
*Realizado com recursos do CNPq (projeto número 401802/2003-5).

(*Common Object Request Broker Architecture*) [Object Management Group, 2004] por ser um padrão aberto, completamente independente de plataforma e que oferece suporte a praticamente todos os serviços necessários em um sistema distribuído.

Um aspecto importante para a confiabilidade dos sistemas baseados em objetos distribuídos é o suporte a replicação de objetos de tal forma que seus métodos possam ser executados mesmo em caso de falhas em alguns dos objetos (réplicas). Esta capacidade visa manter a disponibilidade dos serviços providos pelos sistemas mesmo quando da ocorrência de faltas. No caso da arquitetura CORBA, o suporte a replicação se dá através do padrão FT-CORBA [Object Management Group, 2004][1]. Vários sistemas têm implementado essa arquitetura e ela tem sido a base para diversos projetos relacionados a *middleware* tolerante a faltas [Moser et al., 1999, Natarajan et al., 2000, Lung et al., 2001, Baldoni et al., 2002a, Baldoni et al., 2002b, Bessani et al., 2004, Wilson and Totten, 2005]. O principal desafio envolvido na implementação destes sistemas é suportar o conceito de grupo de objetos (réplicas) e gerenciar falhas nestes grupos mantendo a integridade dos estados de seus membros de acordo com os requisitos de diferentes modelos de replicação.

Muitos destes desafios estão relacionados à resolução do problema de **filiação** (*membership*) [Cristian, 1988], usualmente resolvido através da implementação de **protocolos de filiação** de tal forma que na percepção de falhas no sistema os próprios participantes da computação distribuída possam remover o processo faltoso do grupo (Figura 1(a)). Estes protocolos também tratam as entradas e saídas espontâneas nos grupos.

A arquitetura FT-CORBA define um serviço de filiação centralizado fornecido a partir de um gerenciador de replicação (*Replication Manager*). Este serviço não requer um protocolo distribuído de filiação para as decisões sobre os participantes dos grupos. A simplicidade nas decisões deste serviço tem como custo a sua vulnerabilidade como ponto único de falha na arquitetura (Figura 1(b)).



(a) Filiação Distribuída. (b) Filiação Centralizada.

Figura 1. Gerenciamento de filiação em sistemas tolerantes a faltas.

As especificações FT-CORBA sugerem que os serviços centralizados sejam também replicados para eliminar pontos únicos de falha. Entretanto, os implementadores destas especificações não têm nenhum tipo de indicação de como gerenciar esta replicação. Note que a principal função do gerenciador de replicação (e dos serviços de filiação de um modo geral) é notificar os processos (ou objetos) do sistema a respeito

[1]A descrição do FT-CORBA se encontra no capítulo 23 da especificação principal do CORBA.

das alterações em seus grupos. Sendo este serviço replicado, a grande dificuldade é a recorrência do problema: Quem vai avisar o grupo de gerenciadores de replicação sobre alterações, motivadas por faltas, neste grupo?

Para resolver este problema é preciso definir que ações devem ser tomadas pelo sistema de tal forma a permitir que em caso de falha de algum dos gerenciadores de replicação, o serviço continue operacional, i.e. onde alterações nos demais grupos gerenciados sejam informadas aos processos filiados aos grupos e, além disso, o próprio grupo de gerentes de replicação seja informado sobre a mudança na sua filiação.

De fato, a implementação desta qualidade de serviço, apesar de sua aparente simplicidade, envolve a resolução de uma série de problemas não triviais relacionados ao fato de como resolver os casos de falhas em réplicas do gerenciador de replicação. Além disso, o esforço em termos de implementação é também considerável. Esta complexidade se reflete no fato de que mesmo hoje, após mais de 10 anos de pesquisa em *middleware* tolerante a faltas CORBA, **nenhuma** das implementações conhecidas do padrão FT-CORBA [Moser et al., 1999, Natarajan et al., 2000, Baldoni et al., 2002b, Wilson and Totten, 2005] implementam um sistema completamente replicado, i.e. sem pontos únicos de falhas. Em especial, todas têm no gerenciador de replicação e notificador de faltas seu "calcanhar de Aquiles".

Neste trabalho propomos um esquema de gerenciamento de replicação completamente tolerante a faltas, o qual não emprega um protocolo de filiação convencional (Figura 1(a)), e apresentamos sua integração ao sistema GROUPPAC, uma implementação livre do padrão FT-CORBA [Lung et al., 2001, Bessani et al., 2004]. Fundamentais para a concretização deste esquema são algumas funcionalidades já oferecidas pelo GROUPPAC, como suporte a difusão com ordem total [Bessani et al., 2004], e outras desenvolvidas especialmente para a implantação desse esquema (mas que também são interessantes por elas próprias), como redirecionamento de invocações e supressão de mensagens duplicadas. A implementação deste esquema no GROUPPAC torna este sistema a primeira implementação do padrão FT-CORBA não dependente de pontos únicos de falha, i.e. "completamente tolerante a faltas". Apesar de todo o trabalho ser descrito no contexto do FT-CORBA, acreditamos que o esquema pode ser aplicado em qualquer *middleware* onde o conceito de objetos de serviço possa ser implementado.

O texto está organizado da seguinte forma: As especificações FT-CORBA são brevemente discutidas na seção 2. A seção 3 apresenta o GROUPPAC e seus mecanismos, os quais são usados para a concretização do esquema de replicação, resultando em uma implementação do FT-CORBA completamente tolerante a faltas. Na seção 4, o esquema de replicação da infra-estrutura FT-CORBA é apresentado e seus principais aspectos de implementação são discutidos. Vários testes de desempenho do sistema são apresentados na seção 5. Na seção 6, algumas experiências similares encontradas na literatura são relatadas. Finalmente, na seção 7, temos as conclusões do trabalho.

2. O Padrão FT-CORBA

O padrão FT-CORBA surgiu de um esforço da OMG (*Object Management Group*) para a especificação de uma arquitetura para objetos distribuídos tolerantes a falhas [Object Management Group, 2004]. Este padrão introduz suporte à tolerância a faltas na arquitetura CORBA fazendo uso de técnicas de replicação de objetos. O suporte é construído a partir de um conjunto de objetos de serviço usados no gerenciamento de

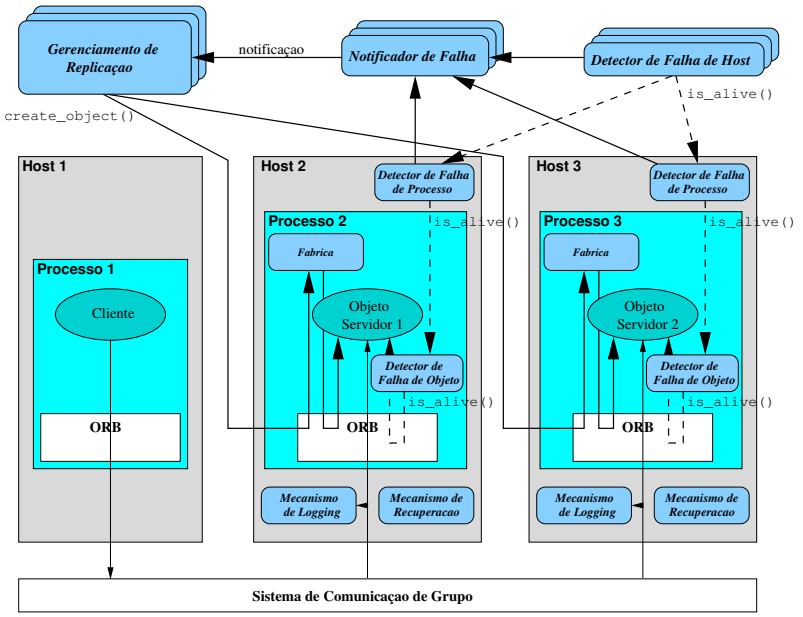


Figura 2. Arquitetura FT-CORBA.

objetos de aplicação replicados, chamado infra-estrutura de replicação. Estes objetos fornecem suporte a vários tipos de replicação: ativa [Schneider, 1990], passiva (morna ou fria) [Budhiraja et al., 1993] e sem estado. Os objetos de serviço no FT-CORBA estão divididos em três módulos:

- **Gerenciamento de Replicação (SGR):** Responsável pelo ciclo de vida dos grupos. Duas funcionalidades são oferecidas: **Gerenciamento de Propriedades**, onde as características funcionais da replicação (tipo de replicação usada, número mínimo de réplicas, etc.) são definidas; e o **Gerenciamento de Grupos**, que oferece mecanismos para a criação de membros (através de **Fábricas de Objetos**) e para o controle da filiação dos grupos definidos;
- **Gerenciamento de Falhas (SGF):** Responsável pela detecção, notificação, análise e diagnóstico de falhas em objetos. Este serviço trabalha em conjunto com o SGR para que este último mantenha a filiação dos grupos sempre atualizada;
- **Gerenciamento de Recuperação e Logging (SRL):** Responsável pela consistência de estados das réplicas. Este serviço define mecanismos para a recuperação de réplicas e do próprio grupo. Entre estes mecanismos está o suporte para a construção dos *logs* usados no armazenamento de requisições enviadas ao grupo. Cabe ao SRL também a atualização de membros através de *checkpoints*.

A Figura 2 apresenta a arquitetura FT-CORBA. Nesta figura não é explicitado suporte dos serviços FT-CORBA no *host* 1 (cliente). Isto mostra que para o cliente a replicação do servidor deve ser completamente transparente, a não ser pelo tratamento da referência de grupo (ver Seção 3.1.). A criação de membros do grupo é feita através do gerenciador de replicação (SGR), que delega esta funcionalidade chamando, no *host* alvo, o método remoto `create_object()` do objeto fábrica que é o responsável local pela criação e ativação de novos membros do grupo (ver *hosts* 2 e 3 na figura).

A detecção de falhas no FT-CORBA (SGF) segue uma hierarquia, de acordo com três níveis de abstração: objeto, processo e *host*. Em cada processo que mantém membros de um grupo existe um detector de falhas de objetos. Este detector verifica perio-

dicamente, através da chamada de método `is_alive()`, se os objetos em seu processo ainda estão em atividade. Para o nível de *host* é definido o detector de falhas de processos que faz o monitoramento dos processos do *host*. Por fim, em nível de sistema, completando a hierarquia, ocorre a detecção de falhas em *hosts*. Os detectores de falhas de *hosts* enviam periodicamente requisições aos detectores de falhas de processos (invocando o método `is_alive()` destes últimos) para verificar se estes, e por conseqüência seus *hosts*, ainda estão ativos.

Quando qualquer um dos detectores presentes no sistema observa uma parada em um objeto, processo ou *host*, esta é sinalizada ao notificador de falhas (ver Figura 2), que a repassa ao SGR para que este atualize a filiação do grupo. Se a falha ocorrer no membro primário de uma replicação passiva, cabe ao serviço de gerenciamento de replicação escolher um novo primário do grupo de objetos, e ao mecanismo de recuperação atualizar o estado deste objeto de tal forma que ele possa continuar processando requisições do ponto onde o objeto primário falhou. A fim de obter uma melhor confiabilidade dos sistemas que usam suporte FT-CORBA, os objetos de serviço (gerenciador de replicação, notificador de falhas e detector de falhas de *host*) devem ser replicados. A idéia em se ter um gerenciador de replicação tolerante a faltas consiste na replicação e auto-gerenciamento destes objetos de serviços.

Em relação à interoperabilidade com grupos onde objetos são mantidos por diferentes ORBs que implementam o FT-CORBA, a OMG criou um formato padronizado para a referência de grupo de objetos: IOGR (*Interoperable Object Group Reference*). Uma IOGR consiste em um conjunto de perfis IIOP[2] que identificam cada membro do grupo, sendo o membro primário (replicação passiva) identificado através de uma *tag* especial em seu perfil. Qualquer um destes perfis pode ser usado pelo cliente para se comunicar com qualquer membro do grupo através de comunicação ponto a ponto.

3. GROUPPAC: Implementando o Padrão FT-CORBA

O GROUPPAC [Lung et al., 2001, Bessani et al., 2004] é uma implementação completa do padrão FT-CORBA desenvolvida pelo grupo de sistemas distribuídos do LCMI/DAS/UFSC. Inicialmente desenvolvido para dar suporte a sistemas de larga escala, o GROUPPAC foi uma das primeiras implementações do padrão FT-CORBA desenvolvidas, e ainda hoje é uma das poucas disponibilizadas livremente (*http://grouppac.sf.net*). Neste sistema, todos os objetos de serviços são implementados e gerenciados de forma replicada, aumentando o grau de tolerância a faltas fornecido.

Em termos de modelo de sistema, o GROUPPAC é construído sobre as premissas de faltas de parada, detecção de falhas perfeitas [Chandra and Toueg, 1996], uso de serviços de comunicação de grupo e canais ponto a ponto confiáveis.

A arquitetura do sistema segue o modelo proposto pelo FT-CORBA (Figura 2), acrescida do uso de um serviço de nomes replicado onde todas as referências a grupos são mantidas para acesso por parte dos clientes. A arquitetura completa é descrita em vários artigos [Lung and da Silva Fraga, 2000, Lung et al., 2001, Bessani et al., 2004]. Nesta seção apresentaremos apenas os mecanismos fundamentais para a implementação da infra-estrutura de gerenciamento de replicação tolerante a faltas.

[2]Cada perfil IIOP (*Internet Inter-ORB Protocol*) contém informações de acesso a um objeto através da rede TCP/IP: endereço IP, porta e identificador do objeto no ORB destino.

3.1. SCG: Serviço de Comunicação de Grupo

O modelo de replicação ativa, proposto para o FT-CORBA, baseia-se na idéia de um grupo fechado de réplicas acessado por um conjunto arbitrário de clientes leves através do mecanismo de comunicação usual do CORBA (chamada de métodos remotos via IIOP/TCP/IP). A partir daí, a especificação prevê a utilização de ferramentas de comunicação de grupo proprietárias para difusão de mensagens com diferentes níveis de qualidade de serviço [Object Management Group, 2004]. Entretanto, estas mesmas especificações não definem como estas ferramentas de comunicação de grupo devem ser integradas na arquitetura FT-CORBA.

Diante disto, foi criado um *framework* de **suporte de comunicação de grupo** (SCG) para o GROUPPAC [Bessani et al., 2004]. O SCG define um mecanismo padrão através do qual a infra-estrutura FT-CORBA interage com diferentes ferramentas de comunicação de grupo visando prover serviços de comunicação com as propriedades necessárias para a replicação ativa, especialmente difusão com ordem total [Défago et al., 2004]. Através deste mecanismo é possível integrar qualquer suporte de comunicação de grupo à arquitetura FT-CORBA concretizando adaptadores que são acoplados ao GROUPPAC através de *plugins*. A Figura 3 apresenta este modelo.

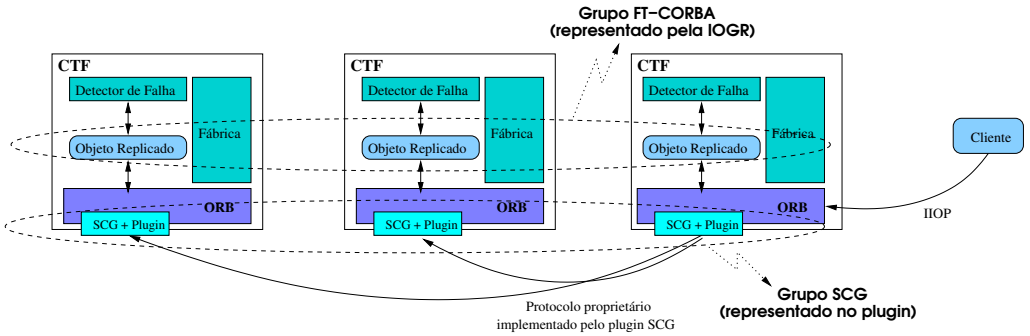


Figura 3. Replicação ativa no GROUPPAC.

Nesta figura, o cliente, de posse da referência de grupo (IOGR), envia uma requisição ponto a ponto a um dos objetos listados na IOGR (representado por um dos perfis IIOP na IOGR), este objeto é chamado **réplica ponte**. O receptor desta requisição serve de ponte para acesso aos protocolos de comunicação de grupo (encapsulados nos *plugins* e ferramentas de comunicação de grupo). Estes protocolos são executados atendendo ao requisito de que todas as réplicas do serviço executem a mesma seqüência de requisições [Schneider, 1990]. As respostas, se houverem, são enviadas de volta à ponte, que retorna o resultado ao cliente[3]. Portanto, é através de réplicas ponte e de *plugins* que o grupo fechado definido pelo FT-CORBA fica acessível a clientes IIOP. Vale lembrar que o SCG é utilizado apenas na replicação ativa. A replicação passiva é inteiramente implementada usando mecanismos de comunicação usuais do CORBA [Lung et al., 2001].

3.2. Redirecionamento e Reinvocação

Conforme apresentado, um cliente que acessa um objeto replicado precisa manipular uma referência de grupo (IOGR), que contém vários perfis IIOP, um para cada réplica do objeto. As requisições ao grupo devem ser direcionadas a uma destas réplicas, chamada

[3]Se o tipo de replicação for ativa, a primeira resposta devolvida por uma réplica é retornada ao cliente. Já no caso da replicação ativa com votação, todas as respostas de réplicas não faltosas (faltas de *crash*) são recebidas para então o resultado ser computado e devolvido.

representante do grupo (a primária na replicação passiva ou a ponte na replicação ativa). Além disso, falhas em réplicas ou na rede devem ser processadas de forma transparente para a aplicação: quando uma falha ocorrer durante a comunicação ponto a ponto entre o cliente e o representante do grupo, faz-se necessário que a requisição seja redirecionada e reenviada a outro representante do grupo.

O Algoritmo 1, apresenta a lógica de redirecionamento implementada nos clientes do GROUPPAC. Este algoritmo é implementado em um interceptador portável [Object Management Group, 2004] instalado no ORB cliente, resultando em um mecanismo completamente transparente para a aplicação.

Algoritmo 1 Interceptador na Infra-estrutura de Tolerância a Faltas (ORB cliente)

1: {Inicialização}
2: $redirected \leftarrow \emptyset$ {Conjunto com os identificadores dos grupos já redirecionados.}
3: $name\_service \leftarrow$ referência para o serviço de nomes
Require: $send\_request(request)$ {Envio de uma requisição.}
4: $reference \leftarrow request.target\_reference$
5: **if** $reference.group\_id \notin redirected$ **then**
6: **if** $reference.replication\_style =$ ACTIVE **then**
7: $target \leftarrow get\_bridge\_from\_reference(reference)$
8: **else** {$reference.replication\_style$ is PASSIVE}
9: $target \leftarrow get\_primary\_from\_reference(reference)$
10: **end if**
11: $redirected \leftarrow redirected \cup \{reference.group\_id\}$
12: FORWARD_REQUEST($request, target$)
13: **end if**
Require: $receive\_exception(request)$ {Recebimento de uma exceção.}
14: $group\_id \leftarrow request.target\_reference.group\_id$
15: $current\_reference \leftarrow name\_service.resolve(group\_id)$
16: $redirected \leftarrow redirected \setminus \{group\_id\}$
17: FORWARD_REQUEST($request, current\_reference$)

Na inicialização é criado um conjunto vazio (*redirected*), que armazena o identificador de grupos que já sofreram o redirecionamento, e obtida uma referência ao serviço de nomes (*name_service*). Toda vez que uma requisição for enviada pelo cliente, o método *send_request* é invocado em seu interceptador, fazendo com que as requisições sejam redirecionadas quando necessário. Na linha 4 é recuperada a referência para o grupo que a requisição esta sendo enviada. Na linha 5 é realizado um teste, para determinar se ainda não foram redirecionadas as requisições para o grupo ao qual esta requisição está sendo enviada. O redirecionamento propriamente dito é realizado nas linhas 6-12. Se o tipo de replicação for ativa uma função é executada para determinar a réplica ponte (*get_bridge_from_reference*), para onde a requisição será redirecionada. Se o tipo da replicação for passiva, a função *get_primary_from_reference* é utilizada para determinar a réplica primária do grupo. Depois de selecionada a réplica para onde a requisição será enviada, o identificador do grupo é adicionado ao conjunto *redirected*, para marcar que este grupo já foi redirecionado (linha 11). Por fim, na linha 12, a função FORWARD_REQUEST faz com que a requisição seja enviada à réplica escolhida (esta função gera uma exceção que faz com que o método *send_request* seja re-executado, porém a condição da linha 5 será falsa).

Ainda existe a possibilidade de ocorrer alguma falta durante o processamento na réplica ponte (ver Seção 3.1.), na réplica primária (replicação passiva), ou na própria rede (ruptura de conexão TCP, etc...). Neste caso o método `receive_exception` é chamado no interceptador. Na linha 14 é determinado o identificador do grupo e na linha 15 a nova referência para este grupo é obtida no serviço de nomes[4]. Em seguida, o identificador do grupo é retirado do conjunto que possui os identificadores dos grupos que não precisam de redirecionamento e finalmente, na linha 17, a função FORWARD_REQUEST tendo como parâmetro a nova referência do grupo fará com que o método `send_request`, descrito acima, seja executado novamente. Um fator que faz com que este algoritmo seja mais eficiente do que outros encontrados na literatura é que uma nova referência é obtida no servidor de nomes apenas na ocorrência de faltas na réplica ponte (replicação ativa) ou na réplica primária (replicação passiva).

Além do interceptador instalado no cliente, ainda existe a possibilidade de algum redirecionamento nas réplicas devido a mudanças da réplica primária em uma replicação passiva. Isto pode ser feito através de ferramentas administrativas ou pela própria aplicação (quando o controle da replicação não for automático) através de interfaces do gerenciador de replicação. Portanto, o ORB de cada réplica também ativa um interceptador bastante simples que, ao receber uma requisição na replicação passiva, verifica se sua réplica é a primária. Caso ela não seja, a réplica apenas encontra o novo primário de seu grupo e indica que a requisição deve ser redirecionada para esse objeto através da chamada a função FORWARD_REQUEST.

3.3. Supressão de Mensagens Duplicadas

Para manter a consistência e a integridade dos estados dos objetos replicados, as implementações do FT-CORBA devem garantir que nenhum objeto execute a mesma requisição mais de uma vez. Existem duas situações onde, requisições duplicadas podem ser executadas por réplicas dos grupos:

1. Uma requisição precisar ser reenviada a outra réplica devido a uma falha durante o processamento na réplica ponte, neste caso é possível que algumas réplicas do grupo (ou até mesmo todas) já tenham executado esta requisição;
2. Em chamadas aninhadas, onde um grupo de réplicas que utiliza replicação ativa é cliente de outro grupo, neste caso as réplicas do segundo grupo receberão uma requisição de cada réplica do primeiro grupo. Note que, se o segundo grupo utilizar replicação passiva, somente a réplica primária receberá requisições duplicadas.

Para evitar execuções repetidas de uma mesma invocação, é necessário que o cliente identifique de forma única cada uma de suas requisições, através da adição de identificadores únicos a elas. A fim de que esta identificação seja transparente a aplicação, esta adição se dá em um interceptador portável instalado no cliente.

Quando a requisição chega a réplica ponte (replicação ativa) ou a réplica primária (replicação passiva), ela é interceptada por outro interceptador que adiciona outras informações ao seu contexto, como o tipo de replicação utilizado e o número de membros pertencentes ao grupo invocado. Caso esta requisição envolva uma invocação a outro grupo (mensagens aninhadas) as novas informações são concatenadas com as já existentes no contexto desta requisição. Através destas informações é possível identificar se uma

[4]A nova versão da referência terminará por ser publicada no serviço de nomes pelo gerenciador de replicação.

requisição já foi executada por determinada réplica. Existe um *buffer* no interceptador instalado em cada réplica para o armazenamento das respostas das requisições e, em caso de recebimento de requisições duplicadas, a resposta da requisição já executada é enviada ao cliente, sem que ocorra a re-execução do método no objeto destino.

Um problema que surge aqui é como realizar a coleta de lixo (respostas a requisições já executadas) de tal forma que esse *buffer* não cresça indefinidamente. No caso da replicação passiva, as respostas são eliminadas do *buffer* de acordo com o tipo do cliente que invoca a requisição:

- **Não pertence a um grupo**: neste caso a resposta pode ser removida do *buffer* assim que for enviada ao cliente.
- **Pertence a um grupo (passiva)**: neste caso apenas a réplica primária irá invocar a requisição, assim a resposta é eliminada do *buffer* tão logo ela seja enviada e a réplica saiba que o próximo *checkpoint* no grupo cliente foi executado.
- **Pertence a um grupo (ativa)**: neste caso a resposta precisa ficar armazenada no *buffer* até que todos os membros do grupo invoquem a requisição.

Contudo, no caso da replicação ativa, uma resposta somente é eliminadas do *buffer* de uma réplica quando, além das condições acima, a réplica tiver certeza de que a requisição foi completamente atendida, i.e. sua resposta foi enviada ao cliente requisitante. Isto se faz necessário devido ao fato de faltas poderem ocorrer durante o processamento de uma requisição pela réplica ponte. As réplicas ponte notificam outras réplicas do grupo sempre que uma requisição é completamente atendida. A fim de evitar uma comunicação extra de mensagens para realizar essa informação, utiliza-se um mecanismo de *piggybacking* [Amir et al., 1992], onde listas de requisições completamente atendidas são enviadas "de carona" durante a difusão de novas requisições no grupo.

4. Replicando a Infra-estrutura FT-CORBA

O principal objetivo do FT-CORBA é eliminar pontos únicos de falhas, e portanto garantir disponibilidade dos serviços em face a ocorrência de faltas no sistema. Assim, cada serviço previsto na arquitetura FT-CORBA (Figura 2) e implementado pelo GROUPPAC deve ser replicado. A Figura 4 apresenta uma visão geral de como as funcionalidades descritas na seção anterior são integradas para implementar de tolerância a faltas na arquitetura do GROUPPAC.

Como pode ser observado na Figura 4, temos o serviço de detecção de falhas que monitora todas as réplicas, inclusive as pertencentes à infra-estrutura, e caso alguma falha seja detectada os dados referentes a mesma são repassados ao serviço de notificação de falhas. Este serviço por sua vez, é responsável por notificar a falha a todos os interessados, isto é, ao serviço de gerenciamento de replicação e aos processos que executam o algoritmo de difusão com ordem total usado pelo SCG, pois este protocolo pode depender deste serviço para manter a filiação atualizada dos grupos. Quando o serviço de gerenciamento de replicação recebe a notificação de alguma falha, a réplica faltosa é retirada do grupo (atualização da filiação) e o serviço de nomes é atualizado com a nova referência do grupo, a qual não contém a réplica faltosa.

O serviço de detecção de falhas utiliza um tipo especial de replicação, independente do serviço de gerenciamento de replicação, já que os próprios detectores têm de identificar falhas em réplicas do serviço. A idéia empregada é baseada em um anel lógico onde cada detector monitora os objetos do sistema e também o próximo detector no anel.

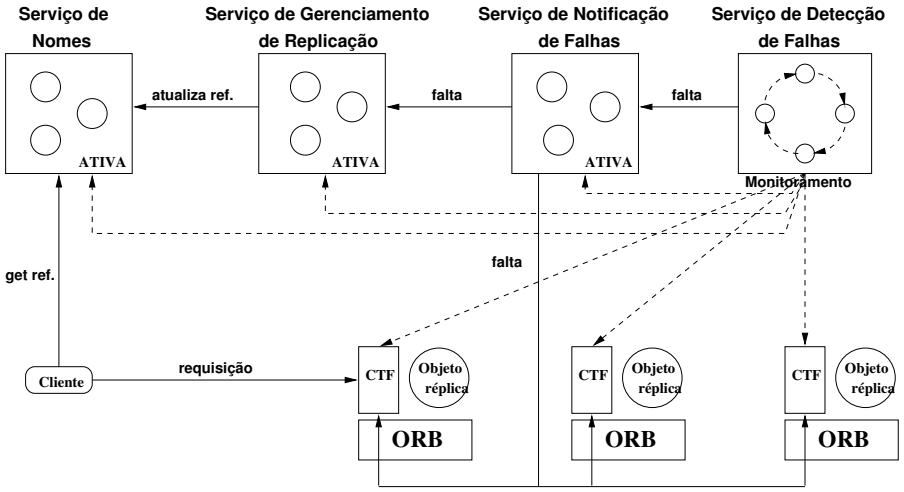


Figura 4. Replicação da infra-estrutura no GROUPPAC.

Toda falha detectada por um membro do grupo de detectores é **votada** no grupo de tal forma a melhorar a precisão do serviço de detecção de falhas em face as oscilações de um sistema assíncrono [Lung and da Silva Fraga, 2000].

Os serviços de gerenciamento de replicação e de notificação de falhas são mantidos por objetos de serviço replicados através de replicação ativa. Este é o único tipo de replicação possível para estes serviços uma vez que na replicação passiva somente a réplica primária de cada grupo executa as requisições. Caso este tipo de replicação fosse utilizado, na ocorrência de uma falha na réplica primária do grupo de notificadores de falhas, ninguém notificaria os interessados sobre a falha. Se a falha ocorresse no membro primário do grupo dos gerenciadores de replicação, ninguém atualizaria a filiação do grupo e sua referência.

O último serviço replicado em nossa infra-estrutura de tolerância a faltas é o serviço de nomes. O tipo de replicação empregada é também a ativa uma vez que os interceptadores clientes sempre recorrem a este serviço no caso de falhas no primário de uma replicação passiva (Seção 3.2.). Desta forma, este tipo de replicação não poderia ser usado neste serviço devido a inexistência de réplicas capazes de responder a estes interceptadores em caso de falha na réplica primária deste.

Note que os mecanismos detalhados na seção anterior são fundamentais para implementação desse esquema: os grupos se baseiam em replicação ativa, portanto é fundamental a existência do SCG juntamente com um *plugin* com suporte difusão com ordem total; grupos de objetos invocam outros grupos (ex. notificador de falhas invoca métodos do gerenciador de replicação), logo o tratamento de invocações aninhadas (duplicadas, já que são grupos baseados em replicação ativa) é necessário; e o cliente deve acessar um novo representante do grupo em caso de falha no atual, o que fica a cargo do mecanismo de redirecionamento e reinvocação.

4.1. Processamento de Faltas na Infra-estrutura FT-CORBA

Quando ocorre uma falha em alguma réplica de objeto, o gerenciador de replicação deve atualizar a filiação e publicar uma nova versão da referência do grupo correspondente no serviço de nomes. No entanto, quando a réplica faltosa pertence a própria infra-estrutura de tolerância a faltas, esta tarefa torna-se muito mais complexa.

Conforme discutido anteriormente, o protocolo de difusão com ordem total usado pelo SCG pode depender dos serviços da infra-estrutura do FT-CORBA para funcionar corretamente. Mais especificamente, este protocolo precisa ser notificado sobre a ocorrência de faltas (utiliza o serviço de notificação de falhas), para poder manter a filiação dos grupos atualizada, uma descrição detalhada deste protocolo pode ser encontrada em [Bessani et al., 2004]. Um fator a ser abordado aqui é que o mesmo utiliza o paradigma de ordenador fixo[5] [Défago et al., 2004]. Neste caso, alguns cuidados a mais são necessários, pois o membro faltoso pode ser o ordenador do grupo de notificadores de faltas, e neste caso não existirá um processo para definir o número de seqüência da mensagem e conseqüentemente ela não será entregue. Para superar este problema, enfraquecemos a difusão das mensagens dos detectores de falhas para os notificadores de falhas de tal forma que estas não sejam ordenadas (requerem apenas difusão confiável). Este enfraquecimento só é possível devido ao fato do método invocado pelos detectores de falhas não alterar o estado dos notificadores de falhas. Assim, o notificador poderá notificar os processos que executam o protocolo de difusão para que estes escolham um novo ordenador se necessário, além do serviço de gerenciamento de replicação, para que este atualize a filiação e a referência do grupo.

Se o *plugin* SCG não utilizar as informações da infra-estrutura de tolerância a faltas para atualização do grupo [Bessani et al., 2004], a modificação descrita acima não é necessária. Entretanto a própria ferramenta de comunicação de grupo usada pelo *plugin* deve implementar algum tipo de detecção de falhas e gerenciamento de filiação.

No caso de ocorrência de alguma falha no grupo de gerenciamento de replicação, as réplicas restantes (não faltosas) serão as responsáveis por atualizarem a filiação do grupo e publicarem uma nova referência do grupo no serviço de nomes. Visto que tanto o serviço de gerenciamento quanto o serviço de notificação utilizam replicação ativa, os maiores problemas ocorrem quando a falha ocorre na réplica ponte. Porém, neste caso os mecanismos de redirecionamento e reinvocação (ver seção 3.2.), juntamente com o SCG, providenciarão a entrega da mensagem para o grupo.

Finalmente, quando da ocorrência de uma falha em qualquer réplica do serviço de nomes, o gerenciador de replicação primeiro atualiza a filiação deste grupo, para depois publicar uma nova referência do serviço de nomes no próprio. Note que se a falha ocorrer na réplica ponte, os mecanismos de redirecionamento e reinvocação escolherão outra réplica para desempenhar esta função.

5. Experimentos

Esta seção descreve alguns experimentos realizados para avaliar as conseqüências da replicação da infra-estrutura de tolerância a faltas no GROUPPAC. A aplicação exemplo utilizada foi um serviço de armazenamento remoto que provê uma operação para o armazenamento de um bloco de dados retornando seu resumo criptográfico. O ambiente utilizado foi uma rede local com 5 máquinas com CPU Athlon XP de 2,6 GHz, 512 Mb de memória e sistema operacional GNU Linux (kernel 2.6).

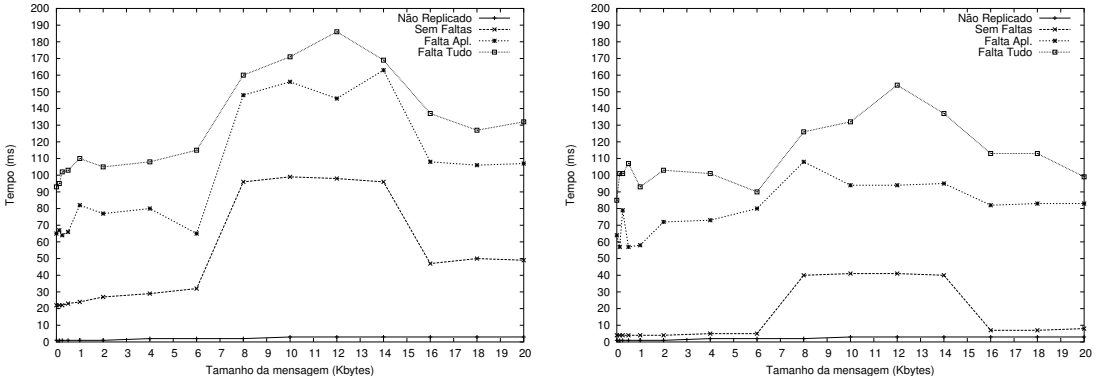
Foram realizados dois experimentos. No primeiro mediu-se o tempo médio necessário para o processamento de uma requisição para um objeto replicado. Este tempo foi calculado a partir da média dos tempos obtidos em 500 invocações sucessivas ao grupo

[5]Neste tipo de protocolo, um processo é o responsável por definir o número de seqüência de todas as mensagens difundidas no grupo, desta forma garantindo ordem total.

de objetos. Foram realizados testes utilizando ambos os estilos de replicação: ativa (com votação) e passiva (morna). Para cada um dos estilos três cenários foram analisados: **(1)** execução sem falhas, **(2)** execução com falha na aplicação e **(3)** execução com falha na aplicação e em todos os serviços da infra-estrutura. No cenário 2, no caso da replicação ativa, a falha é injetada na réplica ponte, já na replicação passiva a réplica primária é a faltosa. Isto faz com que (em ambos os casos) o grupo fique indisponível até que a infra-estrutura entre em ação e trate a falha, atualizando o grupo. No cenário 3, são injetadas falhas ao mesmo tempo em réplicas da aplicação, do notificador de falhas e do gerenciador de replicação. Este experimento modela o pior caso possível, quando réplicas críticas da aplicação e da infra-estrutura falham ao mesmo tempo.

O GROUPPAC foi configurado da seguinte forma para realização do experimento: detectores de faltas no estilo de monitoramento *Pull* (detectores verificam ativamente os objetos), granularidade de detecção por membro (cada detector monitora cada membro de cada grupo), *timeout* de 5 segundos e intervalo de monitoração de 1 segundo. Estes dois últimos valores foram escolhidos de forma bastante conservativa visando evitar a todo custo falsos positivos na detecção de faltas. Além disso, na replicação ativa utilizou-se o *plugin* MJACO como ferramenta de comunicação de grupo [Bessani et al., 2004]. Este *plugin* implementa difusão com ordem total usando os protocolos de comunicação do próprio ORB e fazendo uso dos serviços de detecção e notificação de falhas do GROUPPAC. Na replicação passiva os *checkpoints* foram configurados para serem executados automaticamente a cada 2 segundos.

Todos os cenários do experimento foram executados em 5 máquinas (um cliente e quatro réplicas) e os resultados são apresentados na Figura 5. Nos casos onde falhas são injetadas (cenários 2 e 3), durante o processamento da seqüência de requisições, o objeto de aplicação executa `System.exit(0)` ao receber a invocação, fazendo com que o processo pare (provocando falha em todas as réplicas de objetos nele ativadas). No caso 2, o processo que falha contém apenas a réplica do objeto de aplicação, enquanto que no caso 3 ele contém réplicas de todos os serviços da infra-estrutura e do objeto da aplicação. Para fins de comparação, em cada um dos gráficos também são apresentados os custos da mesma invocação da aplicação sem o suporte a tolerância a faltas (sem replicação).



(a) Replicação Ativa (b) Replicação Passiva

Figura 5. Desempenho do GROUPPAC.

Conforme esperado, os resultados na replicação ativa (Figura 5(a)) são piores que os obtidos com replicação passiva (Figura 5(b)). Note que no cenário 1 (sem falhas), a

replicação passiva obtém resultados muito próximos da execução sem replicação[6]. Isto era esperado pois o único *overhead* neste cenário está nos interceptadores e no serviço de *logging*. No caso da replicação ativa, um protocolo de difusão com ordem total é executado, portanto o tempo requerido para invocar um método é muito maior[7].

Nos cenários com falha, o tempo médio requerido para a chamada de um método remoto foi maior (especialmente na replicação passiva), mesmo com o custo do tratamento da falta diluído nas 500 requisições executadas. Isto mostra o alto custo da detecção e processamento de faltas, o que não chega a nos surpreender uma vez que o GROUPPAC foi construído para operar de forma otimizada em ambientes sem faltas.

No cenário 2 da replicação ativa (Figura 5(a)), ocorre perda de desempenho mesmo na ocorrência de faltas apenas em réplicas da aplicação. O que a princípio parece ser uma contradição, já que neste tipo de replicação não existe recuperação de réplicas, se justifica pelo fato de que a falha foi injetada justamente na réplica responsável pela ordenação de mensagens, e portanto o protocolo de difusão com ordem total fica paralisado até que o notificador de falha envie a filiação atualizada do grupo e um novo seqüenciador possa ser definido. Caso a falha fosse injetada em outro membro que não o ordenador, ou outro tipo de protocolo completamente distribuído fosse utilizado, as diferenças entre esse cenário e o cenário sem falhas seriam menores.

O segundo experimento realizado objetivou mensurar o custo da recuperação de falhas na infra-estrutura e analisar cada uma das fases de execução do esquema de processamento de faltas. Neste experimento, mediu-se o tempo médio necessário para que um cliente execute uma operação invocada no grupo ao mesmo tempo em que uma falha ocorre, e por conseguinte o tempo para processamento e recuperação da falha, utilizando diferentes valores de *timeout* para o serviço de detecção de faltas nos dois tipos básicos de replicação. O tipo de falha injetada corresponde ao cenário 3 do experimento anterior (réplicas dos serviços da infra-estrutura e da aplicação falham).

A Figura 6 apresenta os tempos totais requeridos para recuperação e indica as percentagens de tempo gastas em cada uma das três fases da recuperação: **Detectar**, tempo entre a ocorrência da falta e o momento que o serviço de notificação recebe esta informação; **Processar**, tempo entre o fim da fase anterior e o momento em que uma nova referência para o grupo faltoso é publicada no serviço de nomes; e **Cliente**, tempo gasto pelo cliente no redirecionamento da invocação e seu processamento correto (após a recuperação do serviço).

Nesta figura podemos perceber que o tempo de recuperação está diretamente relacionado com a configuração dos *timeouts*: quanto menor o *timeout*, mais rápida a detecção de faltas e menor o tempo necessário na recuperação de falhas. Conforme esperado, o tempo médio para recuperar uma falta foi maior na replicação passiva do que na replicação ativa. Isto se deve ao fato de que este tipo de replicação requer, além do restabelecimento da infra-estrutura, a escolha de uma nova réplica primária e a atualização de seu estado. Na replicação ativa nenhuma recuperação é necessária para a aplicação.

Note que os resultados obtidos demonstram que a maior parte do tempo de recuperação é gasta na detecção da falha. O alto custo dessa fase se deve ao algoritmo

[6]O comportamento "anômalo" no gráfico, particularmente com requisições de 6 a 16 Kbytes, só pode ser explicados como algum tipo de defeito no código do JACORB no que diz respeito ao processamento de invocação dinâmica de métodos (DII e DSI) utilizados pela infra-estrutura de tolerância a faltas.

[7]Para uma análise detalhada deste protocolo em casos sem falha, ver [Bessani et al., 2004].

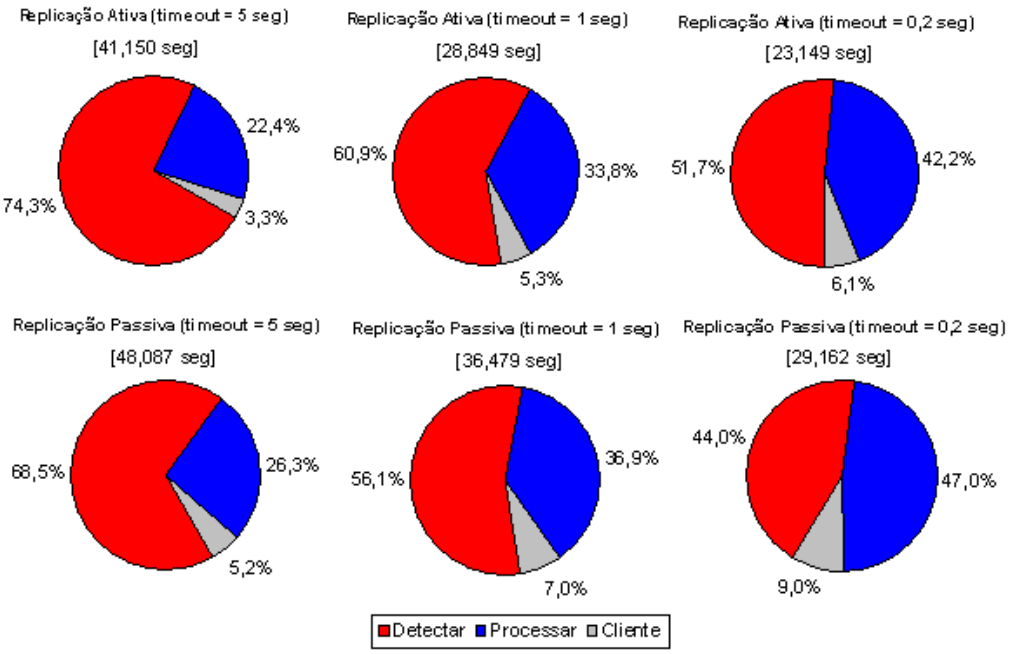


Figura 6. Percentagens de tempo gasto em cada fase da recuperação.

de votação empregado pelo serviço de detecção de falhas (ver Seção 4.). Note ainda que na replicação passiva, a percentagem gasta na detecção de falhas é sempre menor que na ativa, chegando a ser ultrapassada pelo custo do processamento no cenário com *timeout* de 0,2 seg. Isto se deve aos passos requeridos na recuperação da aplicação usando replicação passiva, conforme discutido anteriormente.

6. Trabalhos Relacionados

Existem vários projetos semelhantes ao GROUPPAC que utilizam técnicas de replicação de objetos para suportar tolerância a faltas em *middleware* para objetos distribuídos (ver [Felber and Narasimhan, 2004] para uma descrição do estado da arte), entretanto apenas quatro dos sistemas implementados seguem as especificações FT-CORBA: ETERNAL [Moser et al., 1999], IRL [Baldoni et al., 2002b], DOORS [Natarajan et al., 2000] e o suporte ao FT-CORBA implementado no TAO [Wilson and Totten, 2005].

Cada um destes sistemas suporta tolerância a faltas a partir de uma arquitetura que reflete diferentes decisões de projeto. O ETERNAL utiliza interceptação em nível de sistema operacional para acessar o sistema de comunicação de grupo TOTEM [Moser et al., 1996]. Esta arquitetura torna bastante simples a implementação do FT-CORBA, porém cria dependências entre o *middleware* e serviços específicos do sistema operacional e do TOTEM. O sistema IRL (*Interoperable Replication Logic*) fornece um *framework* de suporte a arquitetura FT-CORBA que pode ser integrado com relativa facilidade a qualquer ORB implementado em Java. O sistema DOORS e os serviços FT-CORBA do TAO suportam apenas subconjuntos das especificações FT-CORBA não oferecendo, por exemplo, suporte adequado a replicação ativa. Apesar dessa diversidade, o ponto fundamental que diferencia todos esses sistemas do GROUPPAC é o fato de que **todos** eles recorrem a serviços centralizados não-replicados (em geral os serviços de notificação de falhas e/ou gerenciamento de replicação), enquanto que o GROUPPAC suporta replicação em todos os seus componentes, eliminando pontos únicos de falha. Desta

forma, a arquitetura apresentada neste trabalho não têm paralelo no contexto de tolerância a faltas em *middleware* para objetos distribuídos.

Para suportar a replicação de toda a infra-estrutura FT-CORBA, alguns mecanismos tiveram de ser desenvolvidos. Dentre estes, um de fundamental importância para a concepção de aplicações complexas (onde os objetos apresentam muitos relacionamentos, e conseqüentemente chamadas de métodos aninhadas) baseadas em FT-CORBA é a supressão de mensagens duplicadas. Dentre os sistemas citados, o ETERNAL e o IRL também relatam suporte a tal mecanismo. Este suporte, assim como no GROUPPAC, é baseado na adição de um identificador único a cada requisição de tal forma a evitar sua re-execução. Entretanto, as especificidades dos mecanismos diferem substancialmente. No ETERNAL, os identificadores são dependentes da ferramenta de comunicação de grupo (TOTEM), e portanto todo o mecanismo está baseado nesta ferramenta [Moser et al., 1999]. O mecanismo do IRL, por outro lado, é portável e pode ser usado em diferentes implementações do CORBA, diferindo do GROUPPAC apenas no algoritmo de coleta de lixo do *buffer* de respostas: enquanto no GROUPPAC uma resposta é removida após ser enviada a todos os possíveis clientes que podem requisitar a execução do método (ver Seção 3.3.), no IRL é associado um tempo estimado pelo ORB cliente a cada resposta e, após decorrido esse tempo (contando-se a partir da produção da resposta), ela é removida [Baldoni et al., 2002a]. Este mecanismo não oferece nenhum tipo de garantia de que uma invocação não será re-executada, ainda mais se considerarmos as imprecisões de um sistema assíncrono.

Existem muitos trabalhos que buscam solucionar o problema da filiação em sistemas distribuídos [Cristian, 1988]. Em geral, estes trabalhos objetivam manter uma visão (lista de processos ativos) consistente em todos os processos de um grupo fechado, através da execução de protocolos de detecção de falhas e difusão de visões, seguindo o modelo da Figura 1(a). Estes protocolos são a base de grande parte dos sistemas de comunicação de grupo (ex. [Moser et al., 1996, Amir et al., 1992]). No caso do GROUPPAC, o problema é um pouco mais complexo tendo em vista que além de avisar os objetos do sistema sobre falhas em outros objetos, um conjunto aberto e desconhecido de clientes deve ter acesso também a visões atualizadas e corretas de um grupo de réplicas do serviço (Figura 4). De fato, sistemas de comunicação de grupo são previstos no FT-CORBA, porém são insuficientes para a resolução do problema como um todo.

7. Conclusão

Mesmo após 10 anos de pesquisas e um grande número de trabalhos propondo arquiteturas, mecanismos e implementações de suportes à tolerância a faltas em objetos distribuídos, é interessante notar que não existe ainda uma proposta em conformidade com padrões abertos (FT-CORBA) que não esteja sujeita a pontos únicos de falha.

Neste artigo, apresentamos a arquitetura de gerenciamento de filiação do GROUPPAC, que permite a replicação de todos os serviços requeridos pela infra-estrutura, eliminando completamente pontos únicos de falhas e preenchendo esta lacuna. A implementação desta arquitetura requer uma série de serviços da infra-estrutura de tolerância a faltas, em especial, acesso a comunicação de grupo com suporte a difusão com ordem total, supressão de requisições duplicadas e redirecionamento.

O enorme esforço requerido para a concepção e implementação deste esquema no

GROUPPAC[8] aliado a ausência de outros trabalhos que implementam essa funcionalidade nos dá evidências de que a completa eliminação de pontos únicos de falha seja justamente os "20% do sistema que tomam 80% do tempo" de acordo com o folclore do desenvolvimento de software, no que diz respeito a implementação de infra-estruturas FT-CORBA.

Referências

Amir, Y., Dolev, D., Kramer, S., and Malki, D. (1992). Transis: A Communication Subsystem for High Availability. In *FTCS-22: 22nd International Symposium on Fault Tolerant Computing*.

Baldoni, R., Marchetti, C., Panella, R., and Verde, L. (2002a). Handling FT-CORBA compliant interoperable object group references. In *Proceedings of the 7th International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS'02)*, pages 37–44.

Baldoni, R., Marchetti, C., and Termini, A. (2002b). Active Software Replication through a Three-tier Approach. In *Prooceedings of the 21st Symposium on Reliable Distributed Systems (SRDS'02)*, pages 109–118.

Bessani, A. N., da Silva Fraga, J., Lung, L. C., and Alchieri, E. A. (2004). Active Replication in CORBA: Standards, Protocols and Implementation Framework. In *Proceedings of the 4nd DOA - Distributed Objects and Applications*. Lecture Notes in Computer Science vol 3291.

Budhiraja, N., Marzullo, K., Schneider, F., and Toueg, S. (1993). The primary-backup approach. In *Distributed systems*, pages 199–216. ACM Press/Addison-Wesley Publishing Co., 2nd edition.

Chandra, T. D. and Toueg, S. (1996). Unreliable failure detectors for reliable distributed systems. *Journal of the ACM*, 43(2).

Cristian, F. (1988). Agreeing on Who is Present and Who is Absent in a Synchronous Distributed System. In *Proceedings of the 18th International Conference on Fault-Tolerant Computing*.

Défago, X., Schiper, A., and Urbán, P. (2004). Total order broadcast and multicast algorithms: Taxonomy and survey. *ACM Computing Surveys*, 36(4):372–421.

Felber, P. and Narasimhan, P. (2004). Experiences, strategies, and challenges in building fault-tolerant CORBA systems. *IEEE Transactions on Computers*, 53(5):497–511.

Lung, L. C. and da Silva Fraga, J. (2000). Detecção de Falha para Redes de Larga Escala no Fault Tolerant CORBA. In *Anais do II Workshop de Testes e Tolerância a Falhas - WTF 2000*.

Lung, L. C., da Silva Fraga, J., Padilha, R., and Souza, L. (2001). Adaptando as Especificações FT-CORBA para Redes de Larga Escala. In *Anais do XIX Simpósio Brasileiro de Redes de Computadores - SBRC'2001 - SBC*.

Moser, L. E., Melliar-Smith, P. M., Agarwal, D. A., Budhia, R. K., and Lingley-Papadopoulos, C. A. (1996). Totem: A Fault-Tolerant Multicast Group Comunication System. *Communications of the ACM*, 39(4):54–63.

Moser, L. E., Melliar-Smith, P. M., Narasimhan, P., Tewksbury, L. A., and Kalogeraki, V. (1999). The Eternal System: an Architecture for Enterprise Applications. In *Proceedings of the 3rd International Enterprise Distributed Object Computing Conference (EDOC'99)*.

Natarajan, B., Gokhale, A., Yajnik, S., and Schmidt, D. C. (2000). DOORS: Towards High-performance Fault Tolerant CORBA. In *Proceedings of the 2nd DOA - Distributed objects and Applications*.

Object Management Group (2004). Common Object Request Broker Architecture (CORBA) Specification v3.0.3. OMG Standart formal/04-03-12.

Schneider, F. B. (1990). Implementing Fault-Tolerant Service Using the State Machine Aproach: A Tutorial. *ACM Computing Surveys*, 22(4):299–319.

Wilson, D. and Totten, S. (2005). FT-CORBA services in TAO. *http://www.cs.wustl.edu/∼schmidt*.

[8]Dentre as várias funcionalidade integradas no projeto ao longo de seus vários anos [Lung and da Silva Fraga, 2000, Lung et al., 2001, Bessani et al., 2004], o esquema proposto neste artigo foi sem sombra de dúvida o mais complexo de se implementar.