Um serviço de replicação transparente de servidores CORBA utilizando sincronia virtual

M. João Monteiro Sandra Teixeira Hugo Miranda
Luís Rodrigues
Universidade de Lisboa
mjoao_monteiro@clix.pt {steixeira,hmiranda,ler}@di.fc.ul.pt

Resumo
Este artigo descreve um serviço transparente de suporte à replicação activa de servidores CORBA, que opera sobre qualquer servidor/cliente sem requerer a recompilação de código. A concretização recorre a um sistema de comunicação em grupo disponível na plataforma de suporte à comunicação Appia.

1 Introdução

O *Common Object Request Broker Architecture* (CORBA) [11] é actualmente uma das arquitecturas mais relevantes para o desenvolvimento de aplicações distribuídas baseadas em objectos. Este modelo fornece os mecanismos básicos para a invocação remota de objectos, realizada através do Object Request Broker (ORB). Adicionalmente, fornece também um vasto leque de serviços, como por exemplo, o Serviço de Nomes e o Serviço de Eventos [9].

A replicação de componentes é uma das técnicas utilizadas para aumentar a fiabilidade dos sistemas. Contudo, é necessário assegurar a coerência do estado das diversas réplicas, mesmo na presença de falhas. A especificação de tolerância a faltas para o CORBA (*ft-CORBA*) [3] enriquece o modelo cliente-servidor do ORB tradicional ao propor uma normalização para a interface de sistemas replicados.

A especificação contempla três alternativas de concretização da interface [3]: $i)$ os clientes acedem a um dos membros do conjunto de servidores utilizando o protocolo de comunicação tradicional (IIOP), delegando nesse membro a responsabilidade de assegurar a coerência das réplicas; $ii)$ os clientes utilizam o IIOP para aceder a um mediador que se torna responsável pela execução do pedido e pela coerência das réplicas; $iii)$ os clientes utilizam um protocolo de difusão em grupo para o envio dos pedidos.

Assumindo que a utilização de protocolos concebidos para a comunicação ponto-a-ponto prejudica o desempenho, cada uma das soluções apresenta diferentes ponderações de desempenho e flexibilidade. A alternativa $iii)$ maximiza o desempenho sacrificando a interoperabilidade dos sistemas, um dos pontos fortes de toda a estrutura que tem vindo a ser

definida pela OMG. Por sua vez, a alternativa $ii)$ é aquela que coloca os maiores desafios de desenvolvimento por requerer a concretização de uma interface entre o CORBA e um outro sistema de comunicação.

O artigo apresenta um sistema de replicação, desenvolvido em Java, que utiliza intermediários para a difusão dos pedidos pelas réplicas. As mensagens são distribuídas pelas réplicas utilizando o serviço de comunicação em grupo de uma plataforma de suporte à comunicação. O serviço pode operar transparentemente sobre qualquer servidor, sem requerer a recompilação do código. Para emular um servidor cuja interface é desconhecida, são utilizadas as ferramentas de invocação dinâmica do CORBA. O artigo mostra que estas ferramentas são por si insuficientes para os requisitos e apresenta uma solução.

O artigo está estruturado da seguinte forma: a secção 2 comenta alguns dos trabalhos já desenvolvidos na replicação de servidores CORBA. A secção 3 apresenta uma visão genérica sobre os serviços de comunicação utilizados neste trabalho; a secção 4 descreve a concretização do serviço de replicação. A secção 5 conclui o artigo e aponta direcções para a continuação deste trabalho.

2 Trabalho relacionado

A replicação de servidores CORBA tem sido activamente debatida na comunidade científica. Recentemente, a OMG propôs duas especificações que isolam as propriedades transaccionais da comunicação do serviço de replicação de objectos [12, 10]. Esta separação mostra-se incapaz de satisfazer os requisitos de algumas classes de aplicações, nomeadamente, as baseadas na decomposição em três níveis[1], devido às dependências que os serviços apresentam entre si e que não podem ser satisfeitas isoladamente [2].

A integração destes dois requisitos tem vindo a ser oferecida por diversas plataformas, que apresentam combinações que sacrificam ora o desempenho ora a transparência para os objectos replicados e para os seus clientes.

O Piranha [5] é um gestor de replicação CORBA que oferece também facilidades de administração, como a reinicialização de réplicas falhadas, migração e recuperação do estado de objectos. O Piranha é executado sobre o Electra, um CORBA Object Request Broker (ORB) que suporta a abstracção de grupos de objectos e que recorre aos serviços oferecidos por plataformas de comunicação em grupo. Ao contrário do serviço apresentado neste artigo, a replicação não é disponibilizada transparentemente para os objectos, o que limita a portabilidade das aplicações desenvolvidas.

O Interoperable Replication Logic (IRL) [6] é um sistema de replicação de objectos concretizado independentemente do ORB. O sistema define intermediários entre clientes e servidores que garantem a transparência da replicação e a tolerância a faltas. Para assegurar estas propriedades, o IRL centraliza todos os pedidos num componente replicado denominado ObjectGroup, que se pode apresentar como um ponto de estrangulamento do serviço.

[1] *three-tier*

O sistema Eternal [8] fornece tolerância a faltas transparente para aplicações CORBA. A plataforma apresenta uma combinação de mecanismos e serviços. Os primeiros são concretizados abaixo do ORB para garantir transparência e eficiência; os serviços são concretizados sobre o ORB, por forma a simplificar a sua utilização e a permitir a configuração pela aplicação. A coerência das réplicas é garantida através da utilização de um serviço de comunicação fiável com ordem total, da detecção de duplicados de invocações e de respostas, da transferência de estado coerente para uma réplica nova ou recuperada e, finalmente, do escalonamento coerente de actividades em execução concorrente.

3 Suporte à comunicação

A. Invocações remotas utilizando CORBA

Todos os objectos servidores CORBA têm associada uma referência única denominada Interoperable Object Reference (IOR). A comunicação entre clientes e servidores utilizando CORBA é mediada por Object Request Brokers (ORBs). Um ORB é uma entidade de *software* independente da plataforma que tem o objectivo de suportar a comunicação entre os objectos. Uma das responsabilidades do ORB é a localização de objectos a partir da sua referência. A interacção entre objectos clientes e objectos servidores é suportada por invocações remotas de métodos. Para garantir a inter-operabilidade entre ORBs de fabricantes diferentes, estes devem respeitar convenções de comunicação que definem o formato das mensagens e da representação dos dados. Estas normas estão agrupadas no Generic Inter-ORB Protocol (GIOP) e no Internet Inter-ORB Protocol (IIOP) [11].Um dos serviços normalizados pela OMG é o *Serviço de Nomes* [9] que mantém associações entre nomes de objectos e as suas IORs.

Na maioria das aplicações, o conjunto de métodos declarado na interface de um objecto é conhecido pelos clientes em tempo de compilação. Esta interface é descrita numa *linguagem de definição de interfaces* (IDL, Interface Definition Language) de modo a facilitar a criação automática de rotinas de adaptação que emulam uma invocação local no lado do cliente e encaminham os pedidos para os métodos apropriados no lado do servidor. Em alternativa, o CORBA prevê também uma *interface de invocação dinâmica* (DII, Dynamic Invocation Interface) que permite a definição, em tempo de execução, das chamadas aos métodos. A DII permite que toda a informação necessária à invocação, nomeadamente nome do método, sentido e tipo dos argumentos e do valor de retorno, seja definida dinamicamente. A informação necessária à realização de invocações dinâmicas pode ser obtida por consultas a um *repositório de interfaces* (IFR, InterFace Repository). A realização de invocações dinâmicas em CORBA conta ainda com um terceiro componente, a *interface esqueleto dinâmica* (DSI, Dynamic Skeleton Interface), que permite aos servidores determinarem, em tempo de execução, o método que irá atender um determinado pedido. Utilizando o DSI, os pedidos são entregues a uma *rotina de concretização dinâmica* (DIR, Dynamic Implementation Routine), que aceita como argumento uma estrutura contendo o nome do método invocado e os argumentos. Independentemente das acções tomadas, a DIR é responsável por preparar a resposta à invocação, formatando os

valores de retorno.

As ferramentas de invocação dinâmica desempenham um papel crucial na concretização de um serviço transparente de replicação CORBA: torna-se possível definir intermediários genéricos (que são colocados entre os clientes e servidores) capazes de interceptar e replicar invocações a métodos independentemente da interface dos mesmos.

B. Serviço de comunicação em grupo

Uma das abstracções mais relevantes para a concretização da replicação é a difusão fiável com ordem total, por vezes também designada por difusão atómica. Este serviço garante que todas as processos correctos recebem exactamente os mesmos pedidos pela mesma ordem. Em sistemas em que o número de réplicas pode variar dinamicamente, um *serviço de filiação* assegura que os diversos participantes recebem *vistas* actualizadas de quais as réplicas que estão activas. Nestes sistemas a noção de difusão fiável está intimamente ligada à noção de filiação no grupo. É pois importante ordenar a entrega das mensagens em relação à mudança de vistas, oferecendo aquilo que se designa por sincronia virtual [1], e que pode ser definida do seguinte modo:

Sincronia Virtual *A evolução dos membros do grupo é representada pela entrega de vistas com a filiação corrente do grupo. A sincronia virtual garante que se uma mensagem m é entregue numa vista V^i então todos os processos correctos p que pertencem a V^i também entregam m nessa vista.*

No sistema descrito neste artigo, a difusão atómica dos pedidos é assegurada pelo serviço de comunicação em grupo do *Appia* [7]. O *Appia* [2] é um sistema modular de suporte à comunicação, desenvolvido em Java. Integrando os protocolos adequados, é possível construir uma pilha de protocolos que oferece à aplicação o conjunto de propriedades que esta necessita. Cada módulo do *Appia* é uma camada, ou seja, um micro-protocolo responsável por garantir determinada propriedade. A combinação de camadas (pilha de protocolos) oferece um protocolo único com as propriedades desejadas. A troca de dados entre cada camada é feita através de eventos definidos pelo programador. Por ser desenvolvido em Java, o *Appia* assegura a portabilidade das aplicações num número significativo de plataformas.

O serviço de comunicação em grupo do *Appia* concretiza um serviço de Sincronia Virtual, que garante a gestão da filiação do grupo com detecção de falhas e a possibilidade do envio de mensagens em difusão ou ponto-a-ponto. Quando combinado com uma camada que ordena totalmente as mensagens, o serviço oferece as propriedades de difusão atómica.

[2] http://appia.di.fc.ul.pt

4 Serviço de replicação de servidores CORBA

A. Descrição geral

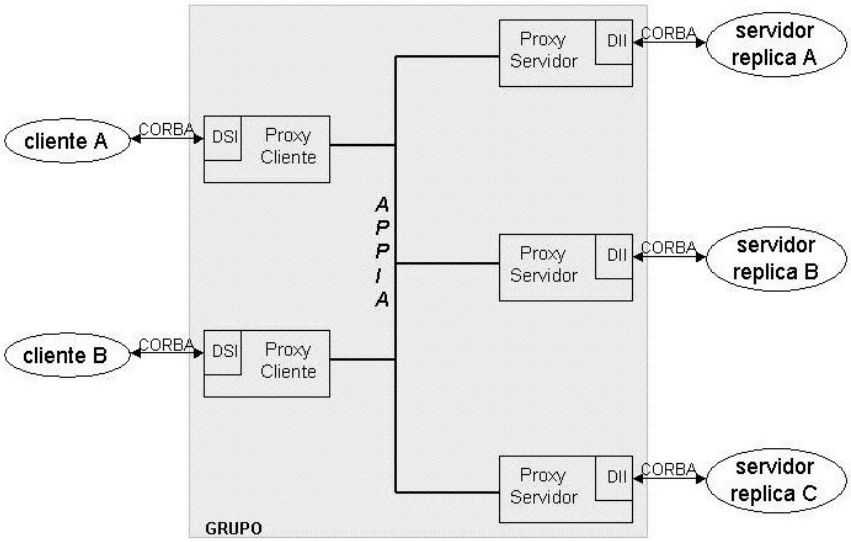

Figura 1: Relação entre os clientes, os intermediários e os servidores CORBA

A Figura 1 apresenta uma perspectiva geral dos componentes utilizados no serviço de replicação de objectos CORBA. Basicamente, entre o cliente e o(s) servidor(es) interpõem-se um conjunto de intermediários (*proxies*) cooperantes, encarregues de replicar os pedidos de modo fiável e ordenado e de recolher as respostas geradas pelas várias réplicas. Existem dois tipos de intermediários, o *intermediário do cliente* (IC) e o *intermediário do servidor* (IS). Cada intermediário tem associado respectivamente um cliente ou servidor, com os quais interagem utilizando CORBA. O conjunto dos intermediários forma um grupo de sincronia virtual que comunica entre si através do *Appia*.

Entre os componentes das arquitectura, observam-se as seguintes interacções. O IC recebe do cliente invocações CORBA e entrega-as a todos os IS que fazem parte do grupo, recorrendo ao serviço de difusão atómica do *Appia*. Os IS recebem dos IC os pedidos e enviam-nos aos servidores que lhes estão associados. As respostas dos servidores são posteriormente enviadas pelo IS, usando comunicação ponto-a-ponto, ao IC que realizou o pedido. Quando o IC recebe a primeira resposta do grupo de servidores, devolve-a ao cliente.

O sistema está preparado para suportar um número indeterminado de clientes e servidores em simultâneo estando apenas limitado pelas restrições que existam por parte do serviço de comunicação em grupo.

B. Descrição do processo

O serviço utiliza uma solução baseada em replicação activa com consolidação no destino [13], isto é, todos os servidores executam os pedidos e devolvem uma resposta. Cabe por isso ao IC a selecção de uma das respostas para entrega ao cliente. Este algoritmo foi seleccionado por ser o que apresenta um processo de recuperação de faltas mais simples.

Os intermediários apresentados na solução desenvolvida dispõem de duas interfaces distintas: uma interface é responsável por lidar com CORBA e a outra com *Appia*. A interface CORBA do IC comunica com o cliente através da interface esqueleto dinâmica onde está concretizada a rotina de concretização dinâmica. É esta rotina que recebe, trata e envia à interface *Appia* os pedidos do cliente. No IS, a interface CORBA é responsável por invocar os métodos pedidos pelos IC, através da interface de invocação dinâmica. A interface *Appia* recebe e envia pedidos e respostas. São as interfaces *Appia* dos intermediários que comunicam entre si através dos protocolos de comunicação em grupo do *Appia*.

1) Iniciação e localização de serviços

O IC personifica tanto quanto possível um objecto servidor mono-tarefa. Através de argumentos, é possível indicar ao intermediário qual o nome com que se deverá registar no serviço de nomes. Naturalmente, este deverá ser o *Serviço de Nomes* a utilizar pelo cliente quando tenta obter o IOR do servidor. Verificações adicionais, realizadas pelo cliente CORBA no momento em que estabelece uma ligação a um servidor,[3] são respondidas favoravelmente. Simultaneamente, o IC utiliza a sua interface para o serviço de difusão atómica do *Appia* para se registar no grupo de intermediários. Para além de se registar implicitamente através da mudança de vista gerada em resposta à alteração na filiação do grupo, o intermediário indica explicitamente aos restantes membros o seu tipo. Este processo facilita a identificação dos ISs, cuja presença é fundamental para que seja possível responder aos pedidos realizados.

Por sua vez, o IS localiza o servidor a que foi destinado, seja por procura num *Serviço de Nomes* (distinto daquele em que o IC se registou ou com uma outra identificação) ou pela recepção do seu IOR no momento em que é iniciado. A interface para o serviço de difusão atómica é utilizada pelo IS para se associar ao grupo e para difundir, em seguida, o IOR do servidor a que está associado.

As mensagens difundidas por todos os IC e IS repetem-se sempre que ocorre uma mudança de vista no grupo, ou seja, sempre que um membro é adicionado ou o abandona.

2) Processamento de pedidos

O IC recebe os pedidos através da sua interface esqueleto dinâmica. Para extrair a informação do pedido, recebido como argumento pela rotina de concretização dinâmica, o IC neces-

[3] Por exemplo as chamadas ao método `_is_a` que tentam assegurar que a invocação irá ser realizada a um objecto da classe correcta.

sita de conhecer a assinatura do método que foi invocado. Para isso utiliza a referência de um dos servidores (estas referências são difundidas pelos ISs sempre que é instalada uma nova vista). A partir desta referência, localiza um servidor e o repositório de interfaces correspondente. Conhecidos os tipos de dados recebidos, o IC extrai toda a informação do pedido e difunde-a para o grupo. Os dados são enviados utilizando as ferramentas de conversão do Java.

A interface *Appia* dos ISs recebe os pedidos difundidos pelos ICs e coloca-os numa lista, partilhada com a interface CORBA. Uma vez retirados da lista, os pedidos são entregues ao servidor através da interface de invocação dinâmica do IS. Tal como o IC, para compor o pedido, o IS necessita de conhecer a assinatura do método, recorrendo também ao repositório de interfaces correspondente. Cada servidor não recebe mais de um pedido em simultâneo do IS associado. Previnem-se dessa forma que as políticas de escalonamento de servidores multi-tarefa alterem a ordenação total já atribuída aos pedidos (isto pode representar uma diminuição na concorrência, mas trata-se de um custo associado à replicação activa [4]).

Quando a interface CORBA do IS recebe a resposta ao pedido, os seus dados são novamente convertidos através das ferramentas de conversão do Java e colocados numa mensagem de resposta, enviada ponto-a-ponto para o IC que formulou o pedido.

A interface *Appia* do IC recebe a resposta de cada IS e verifica se esta é a primeira a ser recebida. Nesse caso, entrega-a à interface CORBA que prepara a resposta (que inclui possíveis excepções retornadas pelo servidor ou os parâmetros de saída e o valor de retorno do método invocado) para o cliente. Caso contrário, descarta a resposta.

C. Tolerância a faltas

As propriedades da sincronia virtual mostram-se muito vantajosas no tratamento de faltas dos processos envolvidos. Sempre que a filiação do grupo é alterada (seja por se suspeitar uma falha, abandono controlado ou adição de um novo elemento), os membros activos da vista recebem uma sequência de duas mensagens. A primeira solicita a autorização para a instalação de uma nova vista. A segunda, que efectivamente instala a nova vista, contém a nova filiação do grupo. De notar que a nova vista só é instalada em todos os membros depois de serem satisfeitas duas condições:

1. As mensagens recebidas por pelo menos um membro do grupo são entregues a todos os membros correctos. Esta propriedade resulta da definição de sincronia virtual;
2. Todos os membros autorizam a instalação da nova vista.

Os processos são livres de reter a autorização de instalação de uma nova vista o tempo que desejarem. Enquanto não for concedida, os processos podem continuar a transmitir mensagens, que serão entregues ao maior número de processos possível, desde que registados na vista que está a terminar. No entanto, após autorizar a instalação de uma nova

vista, um processo deve abster-se de enviar mais mensagens para a rede até que a nova vista seja instalada.

Os IS só autorizam a instalação da nova vista quando todos os pedidos que têm em espera estão tratados. Todos os pedidos recebidos pelos IS depois de terem confirmado a aceitação da nova vista serão tratados mas a resposta só é enviada após a instalação da nova vista.

Por sua vez, os IC autorizam imediatamente a instalação de uma nova vista. Se existirem pedidos pendentes que foram recebidos em pelo menos um IS antes de este ter autorizado a instalação da nova vista, receberão ainda uma resposta. Caso contrário, se algum dos IS que recebeu o pedido instalar também a nova vista (ou seja, pelo menos um dos ISs não falhou) a resposta será entregue após a instalação da nova vista. O IC consegue detectar se receberá uma resposta comparando as listas de membros em ambas as vistas.

A falha de um IC tem um tratamento idêntico ao da falha de um servidor. Ao detectar a quebra da ligação, o ORB do cliente notifica a aplicação através do envio de uma excepção convencionada pelo CORBA, nomeadamente, a excepção COMM_FAILURE. O IC pode também levantar esta excepção intencionalmente. O estatuto de conclusão[4] da excepção poderá indicar dois valores:

COMPLETED_NO Se a operação foi recusada por não existirem servidores filiados no grupo;

COMPLETED_MAYBE Se a operação foi difundida mas não foi obtida resposta.

Os IS abandonam voluntariamente o grupo se, no seu processo de detecção periódica, concluem que o servidor a que estavam associados não está disponível.

D. Conversão de dados

O CORBA não disponibiliza métodos que permitam a uma aplicação a conversão dos tipos compostos em vectores de octetos. Para permitir a troca de dados dos pedidos e das respostas através do *Appia* foi necessário converter todos os tipos ao tipo CORBA genérico *Any*. Para esta conversão torna-se necessário recorrer ao repositório de interfaces para identificar o tipo de dados que está encapsulado pelo tipo genérico. Uma vez identificado este tipo, são usadas as classes de InputStream e OutputStream disponibilizadas nas interfaces CORBA.

E. Transferência de estado

No protótipo concretizado não foi desenvolvido nenhum suporte para facilitar a transferência de estado entre as réplicas activas e uma nova réplica que se pretenda juntar ao grupo. Neste momento, este problema é tratado directamente pela aplicação.

[4]CompletionStatus

```
module SRSCM{
  interface SRSCI{
    //metodos
    void retornaEstado(out Any estado);
    void actualizaEstado(in Any estado);
  };
};
```

Figura 2: IDL de extensão do serviço à sincronização de estado

No entanto, o serviço de replicação de servidores CORBA pode ser extendido para incluir algum suporte para automatizar a transferência de estado. Para isso, os servidores poderiam concretizar uma interface de captura e instalação de estado, definida pelos dois métodos apresentados na Figura 2.

O método retornaEstado seria invocado pelo IS que entrou no grupo a um dos servidores presentes na vista anterior. Esse servidor ficaria responsável por, numa estrutura convencionada pela aplicação, retornar o estado do objecto replicado. Após obter esta informação, o IS invocaria o método actualizaEstado na nova réplica, passando como argumento o resultado da operação anterior. O IS do novo membro do grupo seria responsável por garantir que durante o processo de transferência de estado não eram processados pedidos de clientes.

F. Discussão

A utilização de intermediários no suporte à replicação assegura a portabilidade das aplicações e a compatibilidade com todos os ORBs disponíveis no mercado. Por outro lado, quando comparada com a alternativa em que é utilizado apenas o protocolo IIOP, favorece o desempenho por utilizar um serviço de comunicação em grupo especializado. No entanto, a interface dinâmica do CORBA não dispõe do conjunto de funcionalidades que seriam esperadas.

Uma dificuldade prática na concretização deste serviço prendeu-se com a necessidade de seriar a informação recebida pelos intermediários para a transferir através do sistema de comunicação em grupo (e posteriormente, recuperar esta informação no intermediário remoto). Para obterem os dados contidos na mensagem CORBA e para realizar a invocação no servidor, os intermediários são forçados a ter conhecimento dos tipos dos argumentos e dos valores de resposta do método. Ou seja, não é possível um tratamento transparente dos dados. A assinatura dos métodos está disponível apenas no repositório de interfaces, acedido por invocações CORBA regulares, cuja utilização, prejudica naturalmente o desempenho do conjunto.

G. *Avaliação de desempenho*

Para a realização de testes de desempenho foi definido um servidor com três métodos, que, respectivamente, não recebe argumentos, recebe um inteiro e uma cadeia de caracteres. Todos os métodos retornam os valores recebidos. O cliente envia uma cadeia de 100 caracteres.

nº de Servidores	Intermediários	testeVOID	testeINT	testeSTRING
1	não	0.854	0.883	0.950
1	sim	26.794	26.906	22.172
3	sim	35.549	36.392	25.412
5	sim	36.514	36.929	27.193

Tabela 1: Desempenho do sistema com diferentes combinações do número de réplicas e de operações (ms)

A tabela 1 mostra que, como seria de esperar, a adição de intermediários penaliza significamente o desempenho. No entanto a plataforma de suporte à comunicação impede o crescimento linear quando aumentado o número de réplicas.

5 Conclusões

Este artigo apresenta um serviço de replicação de servidores CORBA, usufruindo do serviço de comunicação em grupo do *Appia*. O serviço apresenta-se completamente transparente para clientes e servidores e, por não requerer adaptações à interface dos servidores, mostra-se de fácil utilização e instalação. A flexibilidade obtida pela utilização de intermediários permite a sua utilização por clientes e servidores utilizando qualquer ORB compatível com a norma CORBA 2.0. Esta versatilidade potencia a sua utilização em ambientes heterogéneos, como aqueles que se podem encontrar nas aplicações mais recentes.

O serviço aqui apresentado não pretende substituir soluções comerciais que concretizam a especificação de tolerância a faltas para CORBA na sua quase totalidade. No entanto pode ser uma alternativa viável para aumentar a fiabilidade de alguns serviços específicos, sem recorrer à aquisição de um ORB especializado. O protótipo permitiu também aferir as potencialidades do *Appia*, permitindo acumular experiência que está a ser utilizada para afinar as interfaces e o desempenho deste sistema.

Referências

[1] K. Birman, T. Joseph. Reliable Communication in the Presence of Failures. *ACM, Transactions on Computer Systems*, 5(1), Fevereiro 1987.

[2] Svend Frølund, Rachid Guerraoui. Corba fault-tolerance: Why it does not add up. Em *Proceedings of the The Seventh IEEE Workshop on Future Trends of Distributed Computing Systems*, Tunisia, South Africa, Dezembro20 1999. Institute of Electrical and Electronics Engineers, Inc.

[3] Object Management Group. *CORBA 2.5*, capítulo 25 - Fault Tolerant CORBA. Número formal/01-09-62. OMG, 2001.

[4] R. Guerraoui, P. Eugster, P. Felber, Garbinato, K. Mazouni. Experiences with object group systems. *Software: Practice & Experience*, to appear.

[5] Silvano Maffeis. Piranha: A corba tool for high availability. *Computer*, 30(4):59–66, Abril 1997.

[6] Carlo Marchetti, Massimo Mecella, Antonino Virgillito, Roberto Baldoni. An interoperable replication logic for CORBA systems. Em *Proceedings of the International Symposium on Distributed Objects and Applications - DOA'00*, Antwerp, Belgium, Setembro21–23 2000. Institute of Electrical and Electronics Engineers.

[7] Hugo Miranda, Alexandre Pinto, Luís Rodrigues. Appia, a flexible protocol kernel supporting multiple coordinated channels. Em *Proceedings of The 21st International Conference on Distributed Computing Systems (ICDCS-21)*, páginas 707–710, Phoenix, Arizona, USA, Abril16–19 2001. IEEE Computer Society.

[8] L. E. Moser, P. M. Melliar-Smith, P. Narasimhan. A fault tolerance framework for CORBA. Em *Proceedings of the IEEE International Symposium on Fault-Tolerant Computing*, páginas 150–157, Madison, WI, USA, Junho 1999. IEEE.

[9] OMG. Corbaservices: Common object services specification. Specification 97-12-02, Object Management Group, Novembro 1997.

[10] OMG. Fault-tolerant CORBA using Entity Redundancy. Request for proposal, Object Management Group, Abril 1998.

[11] OMG. The common object request broker: Architecture and specification. Specification 99-10-07, Object Management Group, Outubro 1999.

[12] OMG. Transaction service specification. Specification formal/01-05-02, Object Management Group, may 2001.

[13] Paulo Veríssimo, Luís Rodrigues. *Distributed Systems for System Architects*, volume 1 de *Advances in Distributed Computing and Middleware*. Kluwer Academic Publishers, Boston, Janeiro 2001.