UM ARQUIVO SEGURO E TOLERANTE A FALTAS PARA CORBA UTILIZANDO FRAGMENTAÇÃO E DISPERSÃO COM REDUNDÂNCIA

Cristina Silva **Luís Rodrigues**
Departamento de Informática da Faculdade de Ciências da Universidade de Lisboa
Campo Grande
LISBOA
Tel.: +351-01-7500171, Fax: +351-01-7500084,
E-mail: csilva@navigators.di.fc.ul.pt, ler@di.fc.ul.pt

Sumário
Apresenta-se o desenho de um arquivo Corba seguro e tolerante a faltas baseado na técnica de Fragmentação e Dispersão com Redundância. Esta Técnica consiste em fragmentar os dados confidenciais e distribuir os fragmentos resultantes, de forma redundante, por diversos servidores.

1. INTRODUÇÃO

A Fragmentação e Dispersão com Redundância (FDR) é uma técnica que pode ser utilizada para se obter segurança e tolerância a faltas [1]. Consiste em fragmentar os dados confidenciais e distribuir os fragmentos resultantes, de forma redundante, por vários servidores num sistema distribuído. A Fragmentação é realizada de forma a que não seja possível obter nenhuma informação de cada fragmento. A Dispersão é realizada de forma a que em cada servidor exista unicamente um subconjunto dos fragmentos e a que exista mais que um servidor com uma cópia do fragmento. Esta técnica providencia tolerância a intrusos: se um servidor for violado o intruso não irá conseguir obter nenhuma informação. Para além disso, se alterar ou apagar os fragmentos estes poderão ser recuperados através de outro servidor.

Este documento apresenta o desenho de um Arquivo CORBA baseado na técnica de FDR, que passaremos a designar por *Arquivo-FDR*. Este trabalho pretende concretizar um Arquivo que possa ser utilizado para guardar de modo seguro o estado de objectos persistentes. O desenho do Arquivo-FDR é baseado na composição de objectos CORBA que interagem exclusivamente através de invocações normalizadas. Assim, o Arquivo-FDR poderá ser utilizado sobre diferentes ORBs.

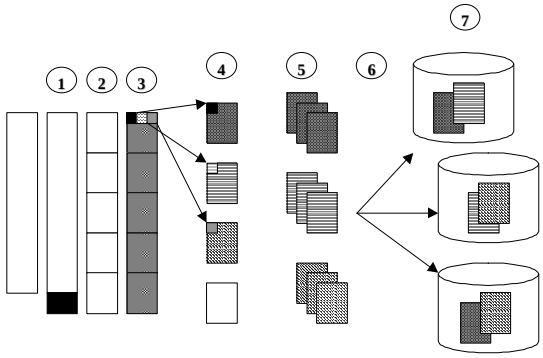
2. FRAGMENTAÇÃO E DISPERSÃO COM REDUNDÂNCIA.

Apresentamos uma breve descrição do modo como a técnica FDR pode ser aplicada para criar um arquivo seguro e tolerante a faltas. Esta descrição é baseada na concretização utilizada no sistema Delta-4 [5]. Mais adiante relevaremos as diferenças existentes entre o nosso desenho e o utilizado no sistema Delta-4.

A base desta técnica consiste em decompor o ficheiro em vários fragmentos. A operação de fragmentação terá que assegurar que, uma vez isolados, não revelam nenhuma informação útil. O tamanho dos fragmentos não depende do tamanho do ficheiro original. O ficheiro é dividido em páginas de tamanho fixo (adicionando "padding" na última página se necessário). Cada página é cifrada, assinada e distribuída por um número fixo de fragmentos. Este processo é ilustrado na Figura 1.

O nome de cada fragmento é obtido através de uma função criptográfica baseada num algoritmo não invertível derivado da chave de cifra, do nome do ficheiro, do índice da página e do índice do fragmento.

Após todos os fragmentos serem criados são enviados aleatoriamente para os servidores. A política de distribuição tem que garantir que os fragmentos são distribuídos por diferentes servidores e que são guardadas *R* copias do mesmo fragmento (para providenciar tolerância a faltas) . Este algoritmo deverá ter em conta o espaço disponível em cada servidor. Este serviço requer um servidor de segurança onde a chave de fragmentação será guardada.



1 – É adicionado, se necessário, "padding" ao ficheiro
2 – É dividido em páginas
3 - É cifrado
4 – São criados os fragmentos para cada página
5 – São efectuadas réplicas dos fragmentos
6 - Difusão dos fragmentos para os servidores
7 – Algoritmo de aceitação dos fragmentos

Figura 1 – Passos da técnica FDR

A técnica FDR complementa a segurança normalmente providenciada pela criptografia tornando-a mais robusta: um intruso que consegue apoderar-se de um servidor não tem acesso a todo o ficheiro. E mesmo que ele/ela conseguisse aceder a todos os N fragmentos teria de efectuar N! permutações dos fragmentos sobre os quais tentaria reconstruir o ficheiro em claro.

3. O DESENHO DO ARQUIVO-FDR

O desenho do Arquivo-FDR utiliza dois tipos de objectos: Fragmentadores e Armazéns. Os Fragmentadores são responsáveis pela fragmentação e pela sua distribuição sendo acedidos através do Serviço de Persistência de Dados ("Persistent Data Service", PDS). Os Armazéns são responsáveis por guardar os fragmentos.

Vários fragmentadores poderão coexistir no sistema. Assumimos que as comunicações entre o PDS e os Fragmentadores são seguras não sendo portanto necessário cifrar as comunicações entre eles. Tipicamente o Fragmentador será instanciado na maquina que corre o PDS sendo possível, no entanto, instanciar o Fragmentador noutra maquina desde que essa parte da rede seja considerada segura.

É ainda possível instanciar diferentes Fragmentadores por cada processo ou mesmo por cada objecto persistente. Esta política de configuração está fora do âmbito deste texto. Para aumentar a clareza da exposição, iremos considerar a associação de um único fragmentador dedicado a cada objecto.
Para providenciar a operação de guardar, os dados são fornecidos pelo PDS ao Fragmentador. O Fragmentador efectua a operação de fragmentar e estabelece o mapa de distribuição, i.e., selecciona os servidores onde será guardado cada fragmento. Cada fragmento será enviado por uma ordem aleatória do Fragmentador para o Armazém. A chave de fragmentação é guardada no servidor de segurança.
Para providenciar a operação de restauro, o Fragmentador recupera a chave de fragmentação do servidor de segurança e reconstrói o mapa de distribuição. Através desse mapa o fragmentador invoca um método do Armazém "restore-fragment" em cada servidor relevante de forma a conseguir obter uma cópia de cada fragmento. Os fragmentos são enviados para o Fragmentador por "call-backs". Para fornecer maior segurança a recuperação pode ser distribuída no tempo (i.e, cada Armazém atrasa o call-back por um espaço de tempo aleatório). O Fragmentador espera por todos os "call-backs" e reconstrói o estado do objecto. Se um ou mais fragmentos estiverem corrompidos (ou se um servidor falhar durante o processo) o Fragmentador invoca adicionais "restore-fragment" de forma a obter adicionais cópias dos fragmentos perdidos.
Uma aspecto fundamental do nosso desenho é que o Fragmentador e o Armazém interagem através de invocações normalizadas CORBA. Assim o Arquivo-FDR é independente das funcionalidades especificas de cada ORB.

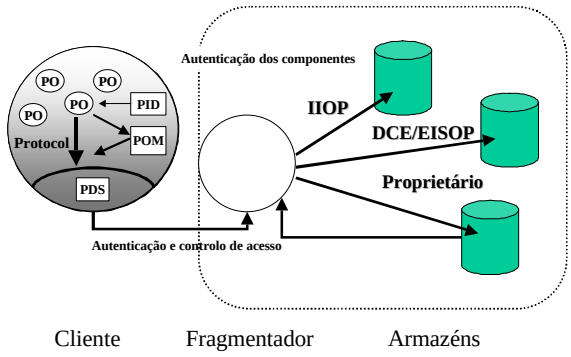


Figura 2 – Arquivo-FDR

Para adicionar tolerância a intrusos é possível configurar o sistema de modo a que o Fragmentador utilize diferentes protocolos inter-ORB para comunicar com cada Armazém individualmente como é ilustrado na Figura 2.

4. INTERACÇÃO COM O SERVIÇO DE PERSISTÊNCIA.

O Arquivo-FDR, como o nome indica, é uma concretização do nível de Arquivo (Datastore) da Especificação do Serviço de Persistência do CORBA. Duma forma simplificada, o arquivo aqui proposto oferece a interface para guardar o estado completo do objecto numa única invocação.

Por razões de desempenho, seria interessante que fosse possível actualizar só parte do estado do objecto sem ter que efectuar a operação completa de fragmentação e distribuição. Dado que o objecto é paginado, deveria ser possível reconstruir e recuperar unicamente as páginas afectadas. No entanto, como o nome dos fragmentos é obtido por uma função não invertível da referência do objecto, e não existe um índice de fragmentos explicito, não é claro que seja possível concretizar esta operação de modo eficiente sem alterar o algoritmo de atribuição de nomes aos fragmentos.

5. O ARQUIVO-FDR.

O serviço de persistência do Corba é especificado com a descrição de três interfaces basicos: Objecto Persistente ("Persistent Object", PO), Gestor de Objectos Persistentes ("Persistent Object Manager", POM), e o já mencionado Serviço de Persistência de Dados. Fundamentalmente estes interfaces contêm os mesmos métodos: *connect, disconnect, store, restore e delete.*
O PDS suporta uma colecção de pares de <Arquivo, Protocolo>. O Arquivo guarda os objectos persistentes e o Protocolo descreve o modo como o estado do objecto é transferido de e para o PO. Ambos, Arquivo e Protocolo, não são definidos na norma. Genericamente podemos descrever o serviço da seguinte forma: O PDS comunica com o PO através de um Protocolo e com o Arquivo através de uma interface especifica. O POM resolve dinamicamente a associação entre PO's e os diferentes PDS's. A identificação do objecto persistente no Arquivo é efectuado através do PID ("Persistent Identifier").
Assim, no nosso modelo, teremos as seguintes interfaces:

```
interface CosPersistencePID::PID {
   attribute string  Arquivo-FDR;
   string get_PIDString();
};

interface PID:  CosPersistencePID::PID {
  attribute string ID;
  void open();
  void store(in stream Sobj);
  void restore(in stream Sobj);
  void close ();
  void delete();
};

interface PIDFactory{
     PID create_unique_PID();
     PID create_PID_from_string(in string pid_string);
};

interface POProtocol {
     void save_state(in PID pid);
     void  load_state(in PID pid);
};
```

O POProtocol descreve o modo como o estado do objecto é transferido de e para o PO. No nosso modelo iremos considerar que terá de ser um protocolo que transforme o estado de um objecto numa "stream" que será fragmentada (pelo fragmentador) e distribuída pelos diversos armazéns. A interface, acima definida, apresenta um exemplo de como poderemos definir o protocolo. Também podemos utilizar o serviço de Exteriorização (Externalization) do Corba que fornece um mecanismo de de exportar e importar o estado de um objecto para uma "stream". A OMG está a definir outro

serviço, que podia também ser utilizado, que é o "object-by-value" que suportará a passagem por valor de objectos em invocações remotas.

```
interface PDS {
    PDS connect( in POProtocol object, in PID pid);
    void  disconnect(in POProtocol object, in PID pid);
    void store( in POProtocol  object, in PID pid);
    void restore(in POProtocol object, in PID pid);
    void delete(in POProtocol object, in PID pid);
};

interface Fragmentador{
  void store(in ID  id, in stream Sobj);
  void restore(in ID id, out stream Sobj);
  void delete(in ID id);
 void  receive-fragment(in FRAG fragment);   //call-back
};

interface Armazem{
      void store-fragment(in FRAG fragment);
      void delete-fragment(in FRAG fragment);
      void restore-fragment(in FRAG frgament);
}:
```

6. INTERACÇÃO COM O SERVIÇO DE SEGURANÇA.

O nosso objectivo é utilizar os mecanismos definidos na norma de Especificação do Serviço de Segurança CORBA para efectuar identificação, autenticação, autorização, controlo de acesso e outras funções de segurança. O serviço de segurança é responsável por autenticar o utilizador e obter informação sobre os seus acessos (direitos sobre o serviço e sobre os ficheiros).

As políticas de controlo de acesso podem ser definidas de forma a prevenir o acesso não autorizado ao Arquivo-FDR. Podem ser definidas a diferentes níveis: os principais que não têm acesso a nenhum serviço não deverão conseguir aceder ao Fragmentador; o Fragmentador deverá verificar se o utilizador tem acesso aos seus diferentes métodos (store/restore/delete) e, finalmente cada Armazém poderá impor restrições particulares sobre quais os principais que têm autorização de guardar fragmentos nesse servidor. A delegação de privilégios é utilizada para propagar o direitos do utilizador pela cadeia de objectos utilizada (PDS, Fragmentador, Armazém e Servidor de Segurança).

Em termos de segurança nas comunicações o Arquivo-FDR requer funções de autenticação (os Fragmentadores e os Armazéns deverão autenticar-se mutuamente), de protecção à integridade (simplifica o desenho) mas não requer confidencialidade pois esta é fornecida pela técnica FDR.

7. SEGURANÇA DO FDR.

Para implementar as funções descritas no parágrafo anterior os utilizadores serão registados no servidor de segurança de modo a proceder à sua autenticação quando pretendem utilizar o Arquivo-FDR. Para tal o servidor de segurança terá o registo dos utilizadores e estes serão agrupados por grupos para facilitar a gestão do controlo de acesso aos objectos persistentes a um conjunto de utilizadores.

7.1 Interacções

O controlo de acesso no sistema Arquivo-FDR é efectuado a dois níveis. Quais os Armazéns a que um utilizador poderá ter acesso para guardar os objectos persistentes e que permissões é que o utilizador tem sobre os objectos que estão arquivados.

A informação dos Armazéns a que um utilizador terá acesso será guardada juntamente com a sua identificação.

Quando um utilizador se autentica receberá a lista dos Armazéns pelos quais poderá distribuir os fragmentos.

O controlo de acesso aos objectos que estão arquivados é efectuado através de um registo com a informação referente à sua identificação (do objecto e não dos fragmentos), a sua chave de fragmentação, o dono do fragmento (quem o criou), do grupo de utilizadores que têm acesso a esse objecto e quais os acessos e da lista dos Armazéns onde estão guardados os fragmentos. Este registo é criado na operação de criação de um objecto persistente.

Quando o utilizador pretende aceder a um determinado objecto, as suas permissões são confirmadas por consulta do registo no servidor de segurança. Se o utilizador possuir autorização para o acesso pretendido, é-lhe enviada a chave de fragmentação e a lista dos armazéns a contactar.

Resta esclarecer o método utilizado para permitir aos armazéns verificarem a validade dos pedidos que lhes são entregues. Recordamons que os armazéns não conhecem a identificação dos objectos mas apenas a dos fragmentos não lhes sendo, por isso, possível consultar o servidor de segurança para confirmar a validade da operação.

A solução utilizada baseia-se no método descrito em [7]. Na operação de criação do nome do fragmento aplica-se uma função de hash ao nome do fragmento e ao nome do objecto, guardando-se o resultado no próprio fragmento. Por sua vez, em resposta a um pedido de um utilizador, o servidor de segurança envia uma senha aos armazéns contendo o resutado da aplicação da função de hash ao nome do objecto. O armazém, ao receber o pedido do utilizador, aplica a função de hash usando a informação contida na senha o nome do fragmento, comparando o resultado com o valor armazenado no próprio fragmento.

7.2 Concretização

Como toda a informação crítica se encontra no servidor de segurança, este deverá ser bastante seguro para tal será possível utilizar o modelo de servidores descrito em[7] em que o servidor de segurança é composto por vários servidores distribuídos geográficamente.

O cliente é registado em cada servidor e cada um fornece-lhe uma chave de identificação. Estas chaves devem ser guardadas em local seguro (ex: smartcard). O protocolo de autenticação é composto por 3 fases. Na primeira fase o cliente tenta autenticar-se em cada um dos servidores. Na segunda fase cada servidor envia a sua decisão para todos os outros servidores. Na terceira fase, de acordo com a maioria das decisões, o utilizador é autenticado e recebe de cada servidor uma chave de sessão. Esta chave é utilizada para autenticar os seus pedidos durante o processo de autorização.

A informação referente aos objectos persistentes é guardada numa estrutura de directórios que pode ser distribuída pelos vários servidores que compõem o servidor de segurança.

O protocolo de autorização é parecido com o de autenticação. O cliente solicita autorização ao servidor de segurança, cada servidor verifica localmente os direitos (que estão replicados em todos os servidores) e envia para todos os outros a sua decisão. Juntos decidem se o cliente tem autorização para efectuar a operação. A chave de cifra de cada objecto, usada para nomear os fragmentos, é distribuída pelos diversos servidores de modo a não ser revelada caso a segurança de um dos servidores seja comprometida. Se o acesso for autorizado é emitido um par de senhas com a informação de segurança (limitada no tempo) para o utilizador e para o conjunto dos armazéns.

8. PONTOS EM ABERTO.

Na arquitectura do Delta-4, cada fragmento era difundido por todos os servidores que decidiam localmente se aceitavam ou não o fragmento. Durante a operação de restauro a lista dos fragmentos era difundido pelos servidores e estes, caso possuíssem os fragmentos, devolviam-nos. Esta concretização assumia ainda que os servidores ou estavam operacionais ou não. Basicamente, o sistema descrito em [1] estava dependente da disponibilidade de primitivas de difusão eficientes [6].

No nosso desenho, lidamos com um sistema mais aberto, em que todas as invocações são ponto-a-ponto. No entanto, pretendemos tornar este serviço tolerante a quebras de rede e desconexões temporárias de servidores. Temos duas alternativas para extender o nosso trabalho nessa direcção.

A primeira alternativa, a que chamaremos distribuição estática, consiste em realizar a dispersão dos fragmentos com base num conjunto pré-definido e estático de armazéns. Disconexões temporárias de um dado Armazém são consideradas como uma falha do ponto de vista da operação de store/restore dos fragmentos. O grau de replicação dos fragmentos é configurado de modo a que se verifique uma elevada probabilidade da operação de store/restore salvaguardar ou obter o número desejado de cópias do fragmento.

A segunda alternativa, a que chamaremos distribuição dinâmica, consiste em os Fragmentadores efectuarem a distribuição de acordo com a disponibilidade dos Armazéns na altura da operação de salvaguarda. Esta distribuição torna mais fácil concretizar políticas de controlo de acesso a cada Armazém: antes de efectuar a distribuição o Fragmentador poderá certificar-se sobre a disponibilidade/autorização de guardar o fragmento. No entanto a distribuição dinâmica requer que o mapa de distribuição (a lista dos Armazéns utilizados) seja guardada no servidor de segurança, juntamente com a chave de fragmentação (para poder ser utilizado na altura da operação de restauro).

As disconecções temporárias é introduzem outro nível de complexidade. Dado que o nome dos fragmentos não revela nenhuma informação acerca da sua origem, não existe forma de distinguir diferentes "versões" do mesmo fragmento. A única forma de distinção é criar novos fragmentos, utilizando o número da versão (ou estampilha) como parâmetro da função de nomeação dos fragmentos. Sendo assim será necessário desenvolver um serviço de reciclagem automática para eliminar os fragmentos obsoletos dos Armazéns (note-se que nem todos os Armazéns que contém uma versão "antiga" de um fragmento poderão estar disponíveis quando a nova versão for criada).

9. CONCLUSÕES.

Neste documento foi apresentado o desenho de um arquivo distribuído, altamente seguro e tolerante a faltas, baseado no modelo CORBA. O Arquivo-FDR possui vários parâmetros de configuração que têm impacto na qualidade de serviço, no grau de segurança, no grau de tolerância a faltas e no domínio do desempenho (tamanho da página, método de cifra, nível de replicação, forma de distribuição, etc). Espera-se que o trabalho de concretização ajude a esclarecer a importância relativa de cada um destes parâmetros.

10. REFERÊNCIAS

[1] Jean-Charles Fabre, Yves Deswarte and Laurent Blain. *Fault-Tolerance and Security by Fragment-Redundancy-Scattering: An Overview*.

[2] J.C. Fabre, Y.Deswarte, B.Randell. Designing secure and reliable applications using fragmentation-redundancy-scattering: an object approach. In *1st European Dependable Computing Conference (EDCC-1)*, pages 21-38, Berlin, October 1994. LNCS 852.

[3] OMG. The Common Request Broker: Arquitecture and Specification, 1997.

[4] OMG. CORBAservices: Common Object Services Specification, 1997.

[5] D.Powell, editor. *Delta-4 – A Generic Arquitecture for Dependable Distributed Computing*. ESPRIT Research Reports. Springer Verlag, November 1991

[6] L.Rodrigues, P. Verissimo. XAMp: a Multi-primitive Group Communications Service. In *Proceedings of the 11th Symposium on Reliable Distributed Systems*, pages 112-121, Houston, Texas, October 1992. IEEE.

[7] Y. Deswarte, L.Blain, J.C. Fabre. Intrusion tolerance in distributed systems. In *IEEE Symposium on Research in Security an Privacy*, pages 110-121, Oakland (CA), USA, 1991.