Sistemas Distribuídos

2016/2017

Komparator

Relatório de Tolerância a Faltas

A57

Repositório

https://github.com/tecnico-distsys/A57-Komparator

		
64776	69836	72887
João Cardoso	Rita Ferreira	Ângela Ferreira
joaocardoso816@gmail.com	rita.c.ferreira@live.com	angela.ferreira@tecnico.ulisboa.pt
Professor do laboratório: Luís Sá Couto		

Solução para Tolerância a Faltas

Replicação

Para garantir o funcionamento contínuo e consistente do Mediator, mesmo em caso falta, foi necessário avaliar as possíveis faltas que poderiam ocorrer e qual a melhor abordagem para as tolerar. Em primeiro lugar, foi necessário replicar o Mediator. Esta tarefa implica mais do que criar um novo processo que aloje o Mediator, pois é necessário mantê-lo consistente com o estado do principal e tê-lo à escuta para assumir o seu lugar quando necessário.

O esquema da Fig. 1 representa a solução implementada:

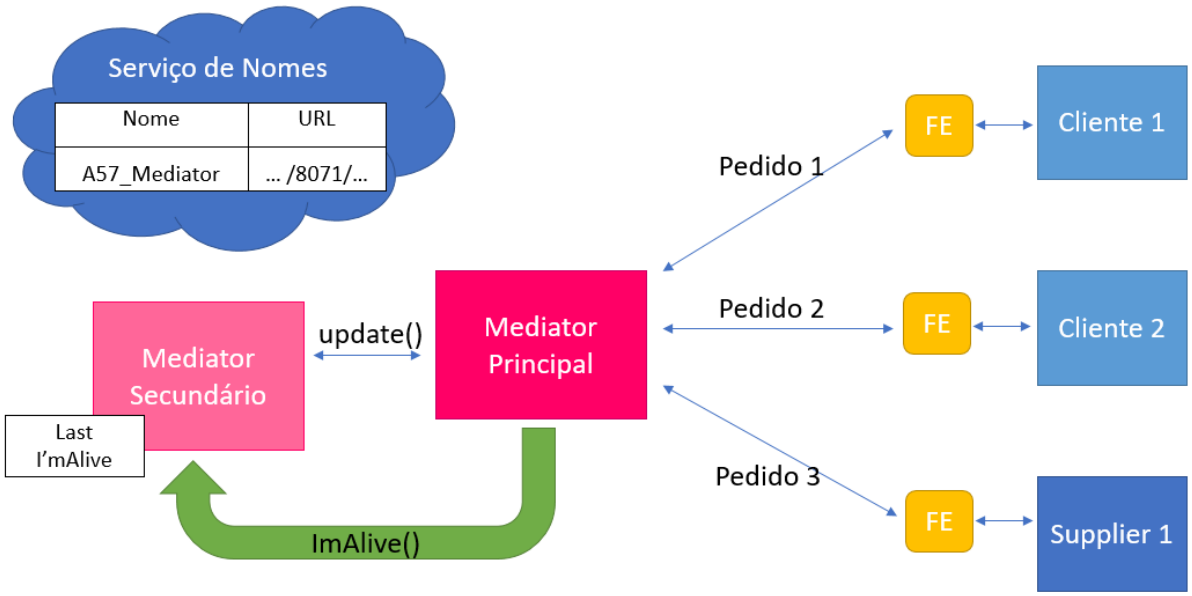

Figura 1 – Solução de Tolerância a Faltas

Na figura, existe um Mediator Principal que contacta com diferentes clientes do Mediator (que podem ser clientes em si ou Suppliers). Para além disso, também está em contacto com a sua réplica. Existem dois tipos de interação com a réplica:

1. **Update():** Estas mensagens são enviadas sempre que o estado interno do Mediator é alterado, por exemplo quando um Cliente realiza um pedido que modifique o conteúdo de um carrinho de compras, ou efetua uma compra. Servem para que o Mediator Secundário se mantenha atualizado para que, caso tenha que assumir o papel de principal, o Cliente poder continuar a realizar-lhe pedidos como se ainda se tratasse da mesma entidade.
2. **Prova de vida:** O Mediator Principal envia mensagens sequenciais, com um pequeno intervalo de tempo entre elas (5 segundos), à sua réplica, de modo a que esta saiba que ele continua ativo. A réplica limita-se a guardar o último "I'm Alive" recebido, para verificar o quão antigo ele é. É a partir da prova de vida, e da sua frescura, que a réplica sabe se deve assumir o papel de Mediator Principal ou não. No projeto, este timeout sem provas de vida é de 20 segundos.

No Serviço de Nomes, UDDI, apenas se liga o Mediator Principal, com o nome A57_Mediator e o cliente descobre-o apenas através do seu nome.

Quanto aos Clientes, estes comunicam apenas com o Mediator Principal, independentemente de este ser o original ou a réplica após assumir esse papel. Esta comunicação está encapsulada no Front End, para que a aplicação do Cliente não seja alterada quando o endereço do Mediator mudar. No projeto, o Front End não é uma entidade, mas sim um conjunto de alterações especificas no cliente e na aplicação do Mediator que permitem facilitar estas interações.

Quando eventualmente o Mediator falha, e a mensagem de I'm Alive recebida pela réplica ultrapassada um determinado período de tempo aceitável, a réplica assume o lugar de Mediator.

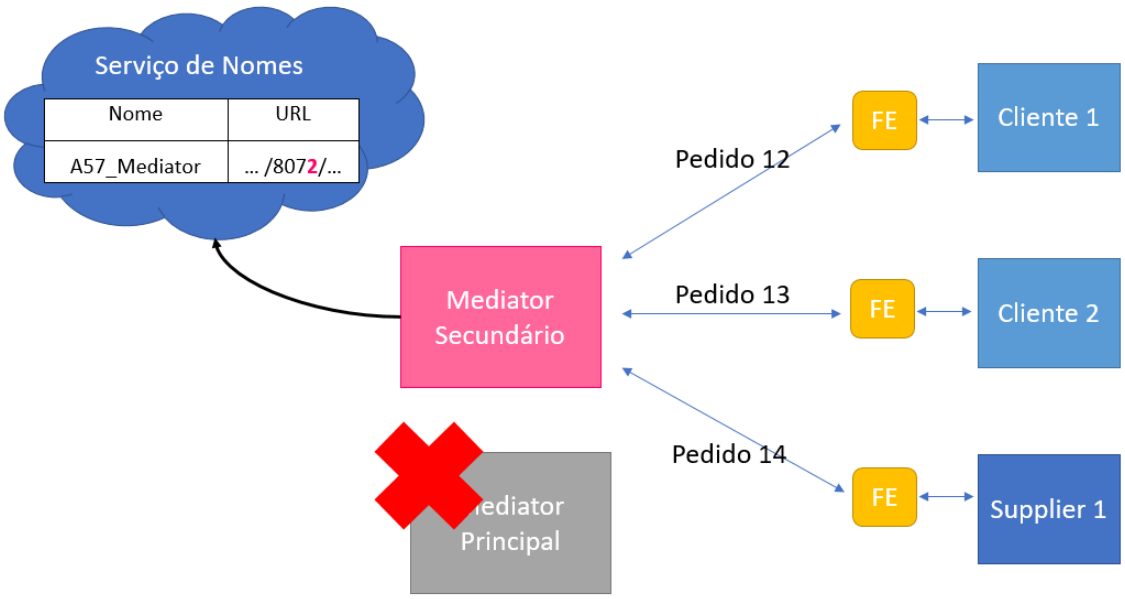

Figura 2 – A réplica (Mediator Secundário) assume o lugar de principal.

Para que isto aconteça, a réplica contacta o Serviço de Nomes para que o nome A57_Mediator corresponda ao URL da réplica, que nunca esteve registada no UDDI até então. A partir daí, os Clientes comunicam normalmente como se nada tivesse acontecido.

Esta solução permite tratar as faltas que são apresentadas na seguinte Fig. 3:

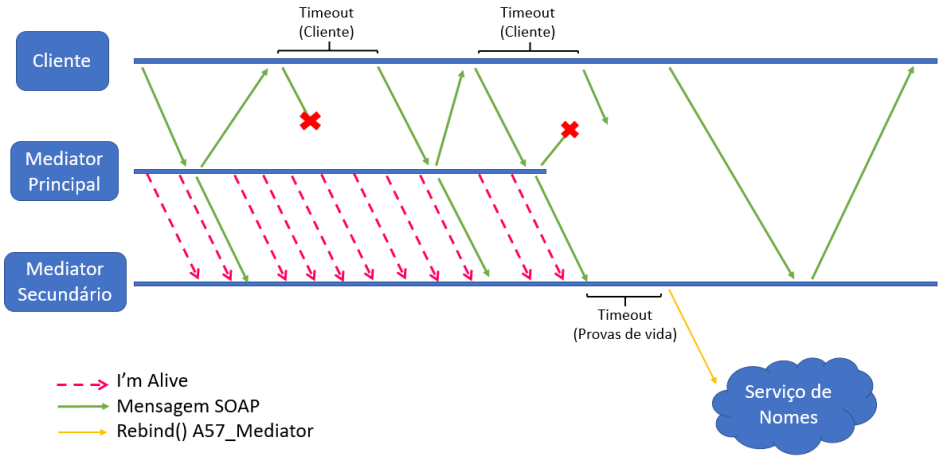

Figura 3 – Exemplo da Comunicação entre um Cliente, Mediator e Réplica e das diferentes falhas toleradas.

Semântica Exatamente-Uma-Vez

Para concretizar a semântica desejada, era necessário implementar um modo de garantir que um pedido fosse enviado, independentemente de falhas no canal, mas também que fosse executado uma e só uma vez. Para a primeira parte deste requisito, foi introduzido um timeout (de 5 segundos) no cliente para que, sempre que um pedido fosse enviado, e não obtivesse resposta, ele reenviaria este pedido.

A segunda parte do requisito não foi implementada por questões de tempo, mas seria possível resolver recorrendo, por exemplo, a uma HashMap que guardasse um id atómico gerado para esse pedido e como valor correspondente a resposta pretendida. Isto poderia ter sido implementado através de um Handler que, à semelhança dos da Segurança, capturava a mensagem SOAP com o pedido e adicionava ao header este Id.

Mensagens do Protocolo

A replicação ficou corretamente implementada graças a três métodos OneWay:

- ImAlive()
- updateCart()
- updateShopHistory()

O primeiro é invocado a cada 5 segundos pelas razões explicadas anteriormente. Os restantes são invocados quando a operação correspondente (addToCart e buyCart) são chamadas pelo cliente do mediator. Estes métodos têm a principal função de enviar à Réplica as alterações realizadas no Principal para que esta possa atualizar o seu estado.

Propagação de Alterações - Exemplo 1	

`Updating Secondary Status - addToCart() ...`	`Updating my status: addToCart()...`
	
	`Update finished!`

Explicação	
Estas mensagens são prints que aparecem quando é realizada uma chamada ao método addToCart() e o Mediator principal ainda está ativo. Sempre que um carrinho de compras é criado ou atualizado, o Mediator Principal faz uma chamada ao método OneWay `updateCart()` e envia uma mensagem à Réplica para que esta mantenha o seu estado atualizado, para assumir o papel de principal a qualquer altura, sem erros ou inconsistências.	
Prova de vida - Exemplo 2	
`I'm alive! Proving it!`	`LastAlive Proof: 2017-05-19T19:31:56.097`
Explicação	
As provas de vida são métodos OneWay enviados pelo Principal para que a Réplica saiba que este ainda está ativo. A mensagem à esquerda é é um print que surge quando um `ImAlive()` é enviado (a cada 5 segundos), e a mensagem da direita aparece quando a Réplica recebe uma Prova de Vida, servindo o Timestamp apenas para provar a sua frescura.	

Testes à Tolerância a Faltas

Para realizar testes à quarta parte do projeto, apenas é necessário terminar o Mediator Principal em variadas situações. Por exemplo, para testar se ele é corretamente substituído, basta desliga-lo, esperar um pouco, e enviar um ping() ou qualquer outra operação através de um cliente.

Para testar a consistência do estado da Réplica, é necessário chamar métodos que envolvam modificações ao estado, nomeadamente o addToCart() e o buyCart(), visto que estes trabalham com a lista de carrinhos de compras e as próprias compras efetuadas até ao momento.