



TÉCNICO
LISBOA

Sistemas Distribuídos 2016/2017

Relatório da 4º Parte do Projeto Tolerância a Faltas

Grupo A50

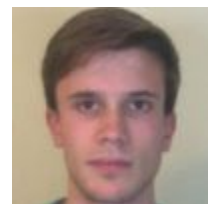
Repositório Git: <https://github.com/tecnico-distsys/A50-Komparator>



77459 - Henrique Lourenço

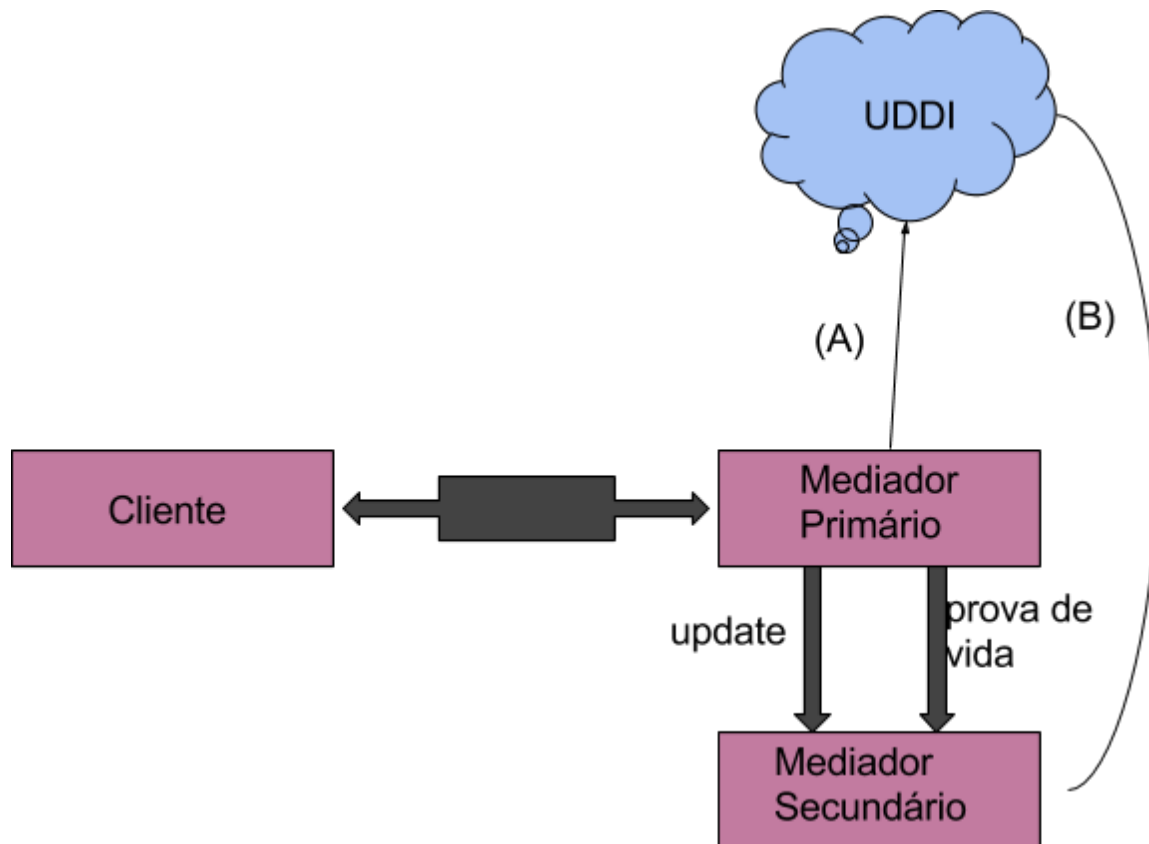


78215 - José Touret



78579 - Pedro Cruz

Figura da Solução



O cliente comunica sempre com o mediador primário. O mediador primário é responsável por enviar provas de vida ao mediador secundário - `imAlive()` - de modo a justificar a sua presença ativa, bem como enviar atualizações sempre que se altera estados do mediador - `updateShopHistory(...)` e `updateCart(...)`. Apenas o servidor primário se publica no UDDI - ligação (A). Após o mediador primário se tornar indisponível, vai parar de mandar provas de vida ao mediador secundário e desta maneira, o secundário publica no servidor de nomes UDDI - ligação B e torna-se no mediador primário que vai interagir com o cliente.

Descrição da Solução

- **Replicação**

O mediador secundário é criado da mesma maneira que se cria múltiplos fornecedores, através da pom. Foi criada a classe LifeProof no domínio de mediator-ws que faz lançamento de provas de vida do mediador primário para o secundário, através da operação unidirecional imAlive(). Esta operação foi acrescentada ao contrato WSDL do mediator-ws. Usa-se um TimerTask para executar a classe LifeProof de 5 em 5 segundos. Se o mediador secundário não receber o imAlive() do mediador primário dentro desses 5 segundos, o secundário torna-se no primário.

```
A50_Mediator
http://localhost:8072/mediator-ws/endpoint

I am the Secondary Server
class org.komparator.mediator.domain.LifeProof running...
Starting http://localhost:8072/mediator-ws/endpoint
Awaiting connections
Press enter to shutdown
class org.komparator.mediator.domain.LifeProof running...
[2017-05-19T16:04:57.630] intercepted INbound SOAP message:
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="
http://schemas.xmlsoap.org/soap/envelope/"><SOAP-ENV:Header/><S:Body><ns2:imAliv
e xmlns:ns2="http://ws.mediator.komparator.org/" /></S:Body></S:Envelope>
```

Figura 1 - Servidor do mediador primário envia provas de vida a servidor secundário de mediador ainda inexistente

```
A50_Mediator
http://localhost:8071/mediator-ws/endpoint

I am the Primary Server
class org.komparator.mediator.domain.LifeProof running...
Starting http://localhost:8071/mediator-ws/endpoint
Publishing 'A50_Mediator' to UDDI at http://a50:fjJay9Ii@uddi.sd.rnl.tecnico.uli
sboa.pt:9090/
Secondary Mediator is not available
Awaiting connections
Press enter to shutdown
class org.komparator.mediator.domain.LifeProof running...
Secondary Mediator is not available
```

Figura 2 - Servidor do mediador secundário recebe provas de vida a servidor do mediador primário

```
[2017-05-19T16:06:02.622] intercepted INbound SOAP message:
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="
http://schemas.xmlsoap.org/soap/envelope/"><SOAP-ENV:Header/><S:Body><ns2:imAliv
e xmlns:ns2="http://ws.mediator.komparator.org/" /></S:Body></S:Envelope>
class org.komparator.mediator.domain.LifeProof running...
class org.komparator.mediator.domain.LifeProof running...
Publishing 'A50_Mediator' to UDDI at http://a50:fjJay9Ii@uddi.sd.rnl.tecnico.uli
sboa.pt:9090/
Awaiting connections
Press enter to shutdown
```

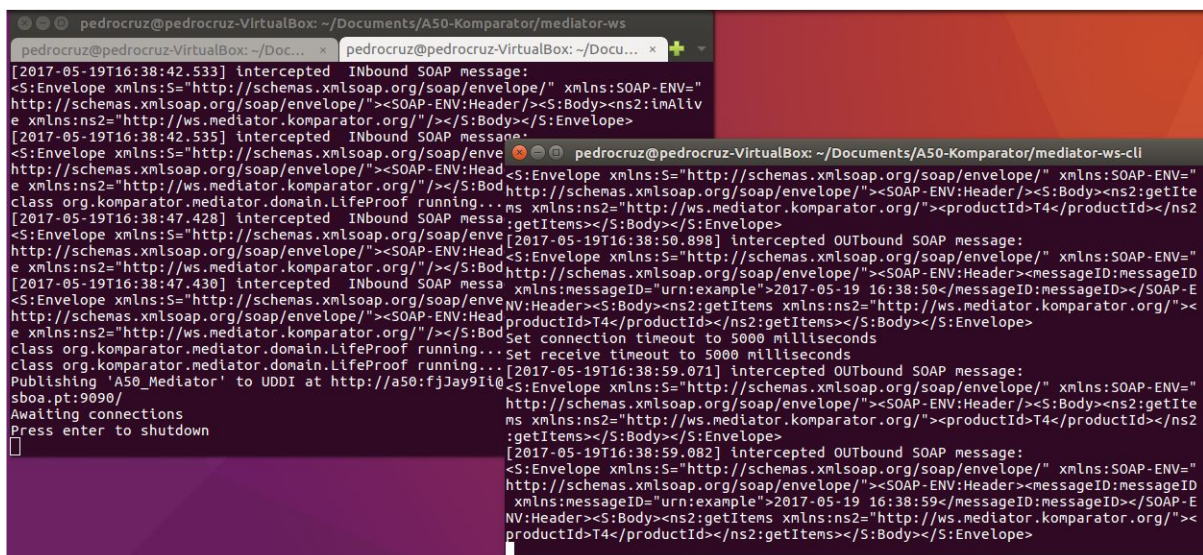
Figura 3 - Servidor do mediador secundário recebe a última prova de vida do mediador primário antes de este interromper a sua execução. Como não recebe outra prova de vida nos 5 segundos seguintes, o secundário torna-se no primário.

- **Timeouts e atualização de estado do servidor secundário**

Foi adicionado timeouts na classe MediatorClient, que permite tornar o cliente tolerante a faltas de quebra de ligação como no caso da paragem súbita do servidor primário. Os timeouts vão parar o fluxo de execução, dando tempo para que o mediador secundário assuma o papel do primário, e permitir que se volte ao fluxo de execução sem problemas. Adiciona-se também ao contrato WSDL dois métodos unidireccionais:

- ❖ updateShopHistory(ShoppingResultView shopResult)
- ❖ updateCart(CartView cart, String cartId)

Estes métodos têm o objetivo de sempre que se altere o estado do mapa de carts e da lista de shopHistory no servidor do mediador primário, ocorra também um update para o servidor secundário, de modo a este estar atualizado quando se tornar no servidor primário. Isto acontece nos métodos addToCart(...) e buyCart(...) do MediatorPortImpl.



The image shows two terminal windows side-by-side. The left window displays a series of intercepted SOAP messages, including an inbound message from the primary mediator and subsequent outbound messages to the secondary mediator. The right window shows the execution of the primary mediator, which eventually throws a 'Set receive timeout to 5000 milliseconds' error, indicating it has stopped responding. The timestamps in the logs show a sequence of events from 2017-05-19T16:38:42 to 2017-05-19T16:38:59.

Figura 4 - A meio da execução dos testes, interrompe-se a execução do servidor primário. Observa-se o tempo de timeout que é concedido (terminal da direita) enquanto o servidor secundário toma o papel do primário (terminal da esquerda)