TÉCNICO LISBOA

Relatório – Tolerância a Faltas
Sistemas Distribuídos 2016/17

Grupo A42:

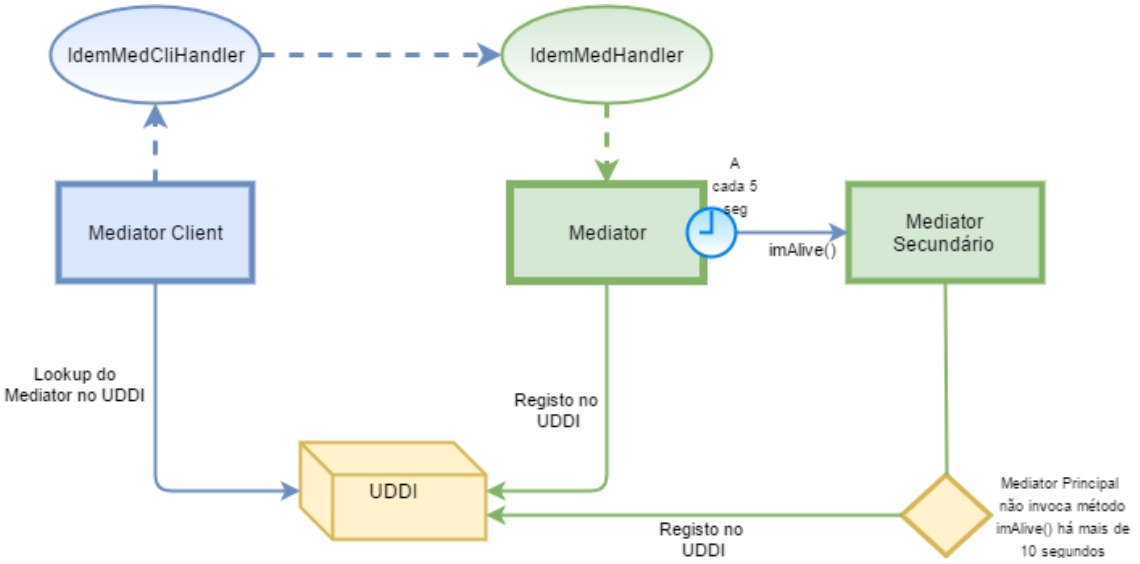
Github: https://github.com/tecnico-distsys/A42-Komparator.git

81172
Carolina Xavier

81186
Stéphane Duarte

81328
Inês Leite

Tolerância a faltas



ImAlive() - O Mediator secundário obtém uma prova de vida do Mediator principal através deste método, guardando o timestamp do instante em que o método é executado.

LifeProof - Esta classe estende a classe TimerTask do Java. Achamos que esta classe era a mais adequada por se tratar de um simples método temporizado. O método run existente nesta classe chama o método imAlive(), de 5 em 5 segundos, no servidor através do cliente. Caso o mediator principal não invoque o método imAlive() há mais de 10 segundos o mediator secundário substitui-o no UDDI com o mesmo endereço.

IdemMedCliHandler – Adiciona um id, gerado aleatoriamente, a cada um dos pedidos enviados pelo cliente. Mensagens com pedidos repetidos mantêm no header o mesmo id.

IdemMedHandler - Obtém o id do do pedido e verifica se o pedido é ou não repetido. Assim se consegue garantir a idempotência das operações.

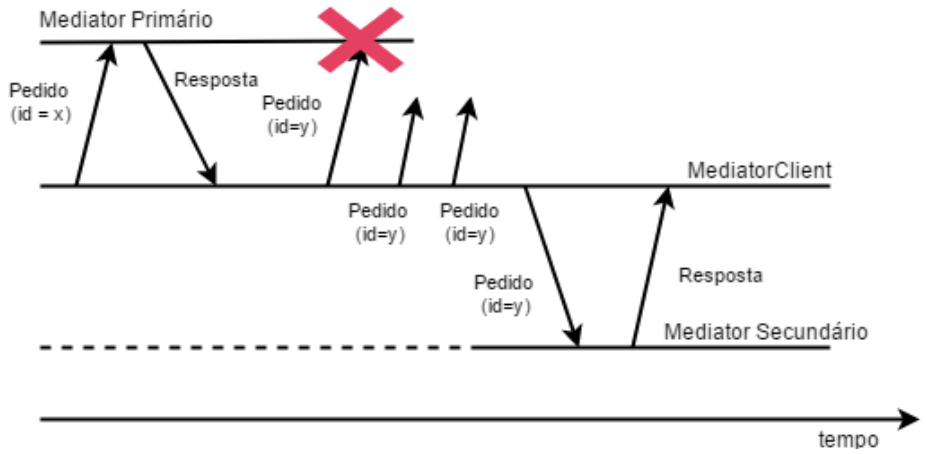
Atualização de estado do mediador secundário

Criámos duas funções (updateHistory() e updateCarts()) para que o secundário possa ter informação sobre todas as compras e carros já existentes. Para isso, enquanto é o Mediator primário que se encontra registado no UDDI sempre que é feita uma actualização nas listas _history e _listcarts (métodos buyCart, addCart e clear) esta actualização é também feita pelo Mediator secundário.

Troca de mensagens do protocolo:

Esta técnica permite garantir o Front-end cliente com semântica no-max-1-vez.

As mensagens passam todas pelos handlers, cuja explicação encontra-se detalhada em cima. Cada identificador representa uma mensagem SOAP diferente.



Na imagem podemos ver um exemplo de troca de mensagens entre os mediators primário e secundário e o MediatorClient. Inicialmente é feito um pedido por parte do MediatorClient ao Mediator que se encontra registado no UDDI (Mediator primário) e este responde-lhe com uma mensagem. Seguidamente o MediatorClient faz um novo pedido ainda ao Mediator primário, com um id diferente do primeiro pedido, mas antes que o Mediator consiga gerar uma resposta é terminado (deixa de estar publicado no UDDI). Depois de passar um *timeout* e o MediatorClient não obter uma resposta, este envia novamente o pedido (mensagens com o mesmo id do pedido anterior). Vai fazendo isto até que o Mediator secundário é publicado no UDDI e aí o MediatorClient passa-lhe a enviar os pedidos. Assim, o MediatorClient envia-lhe novamente o mesmo pedido (mensagem com o mesmo id) e obtém resposta do Mediator secundário.

Para garantir que não são realizadas operações duplicadas tanto o Mediator primário como o secundário guardam um *map* com todos os ids das mensagens do MediatorClient já recebidas. Enquanto o Mediator primário não é terminado e o secundário encontra-se como backup, sempre que e feita uma atualização deste *map* pelo primário é também feita pelo secundário.