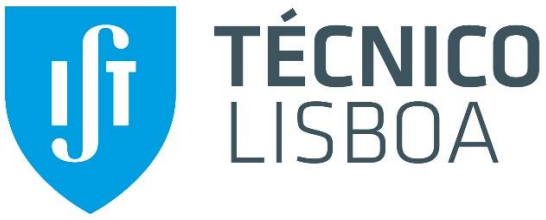
Sistemas Distribuídos



Relatório da 4ª Entrega do Projecto (P4)

Grupo A68



João Silveira
80789

Pedro Orvalho
81151

Rodrigo Mira
81271

https://github.com/tecnico-distsys/A68-Komparator

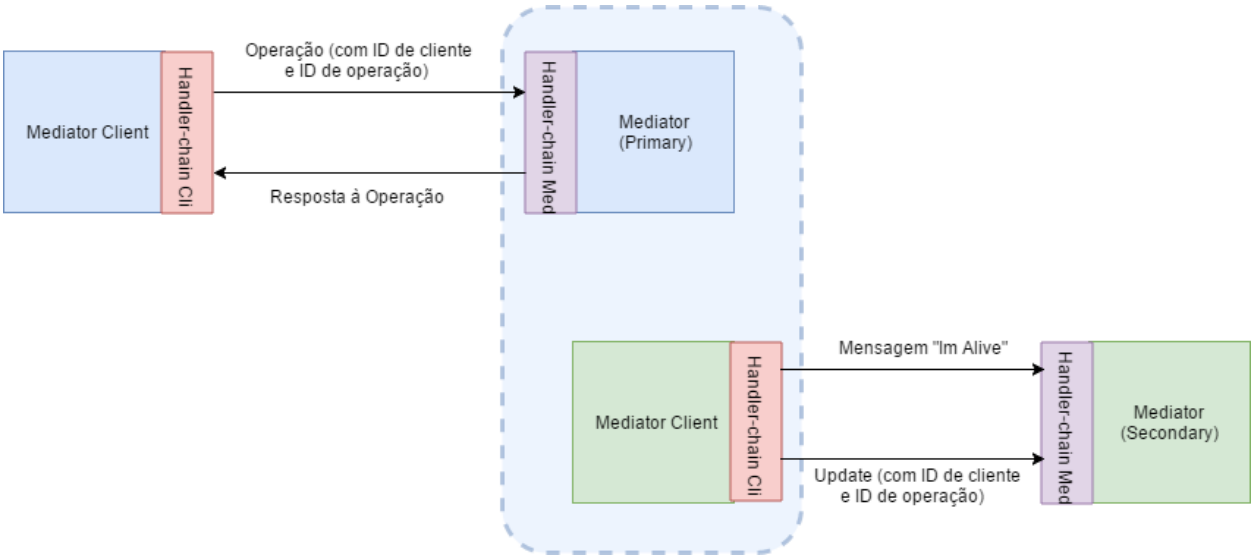


Figura 1 – Esquema da solução de tolerância a faltas (**Funcionamento normal**)

Replicação

O primeiro passo da replicação é a criação de um servidor secundário que corre paralelamente com o primário. A criação e funcionamento deste é semelhante com 2 principais diferenças: não se regista no UDDI e lida com o LifeProof e operações não idempotentes de forma diferente.

A fidelidade deste servidor ao primário é garantida através das operações updateCart, updateShopHistory e updateClear, chamados nas operações addToCart, buyCart e clear respectivamente. Estes updates são necessários apenas nestas operações pois estas efectuam alterações no estado do sistema que têm de ser propagadas para manter uma réplica actualizada. Estas operações de update limitam-se a alterar os atributos do servidor, pelo que não processam novamente toda a operação.

Substituição

O primeiro aspecto a realçar é a detecção da necessidade de substituição no caso do projecto. O servidor primário é responsável, através de uma classe “daemon” LifeProof.java, de enviar mensagens ImAlive (one-way) ao servidor secundário a cada 5 segundos, indicando que ainda está em funcionamento.

O servidor secundário é, por sua vez, responsável por verificar, cada 5 segundos, se recebeu a última mensagem de imAlive há mais de 7 segundos. Se isto se verificar, este assume que o servidor primário não está em funcionamento, e assume o seu papel, mudando o seu estado para primário (que altera o seu funcionamento como acima descrito) e publica-se no UDDI, para que possa ser acedido pelos clientes.

Semântica no-máximo-uma-vez

O objectivo de implementar esta semântica para as operações requisitadas pelo cliente é garantir que o cliente não requisita duas operações iguais quando apenas queria requisitar uma, devido ao time-out e reenvio da mesma. Isto poderia causar uma alteração errada ao estado do sistema na ocorrência da seguinte sequência de passos:

1. O cliente envia uma operação não idempotente para o servidor primário
2. O servidor primário regista e processa esta operação
3. O servidor primário envia o update consequente para o secundário, que o processa.
4. O funcionamento do servidor primário é interrompido antes de este enviar a resposta correspondente à operação ao cliente.
5. O servidor secundário deixa de receber mensagens imAlive e passa a ser o servidor primário
6. O cliente não recebe resposta, pelo que efectua time-out e reenvia a mesma operação para o novo servidor primário.
7. O servidor primário recebe esta operação e processa-a como se fosse nova, alterando o estado do sistema de forma errada.
8. O servidor responde ao cliente normalmente, logo o cliente não tem forma de entender o erro.

A solução deste tipo de situação envolve dois aspectos essenciais: garantir o descarte da operação duplicada e envio da resposta correcta ao cliente. A primeira questão é resolvida através do envio de ID's de cliente e operação. Estes são enviados nas operações através da adição dos mesmos na SOAP Message no IdHandler, e através dos argumentos das operações de update. Tanto o servidor primário como o secundário registam estes ID's, e rejeitam as operações/updates que tenham ID's já registados.

A questão da resposta é garantida através de um mapa no servidor que guarda a shoppingResult mais recente para cada cliente, recebida no update, que é usada para resposta se for requisitada uma operação "buyCart" duplicada (a operação "addToCart" não tem valor de retorno, logo não existe este problema).

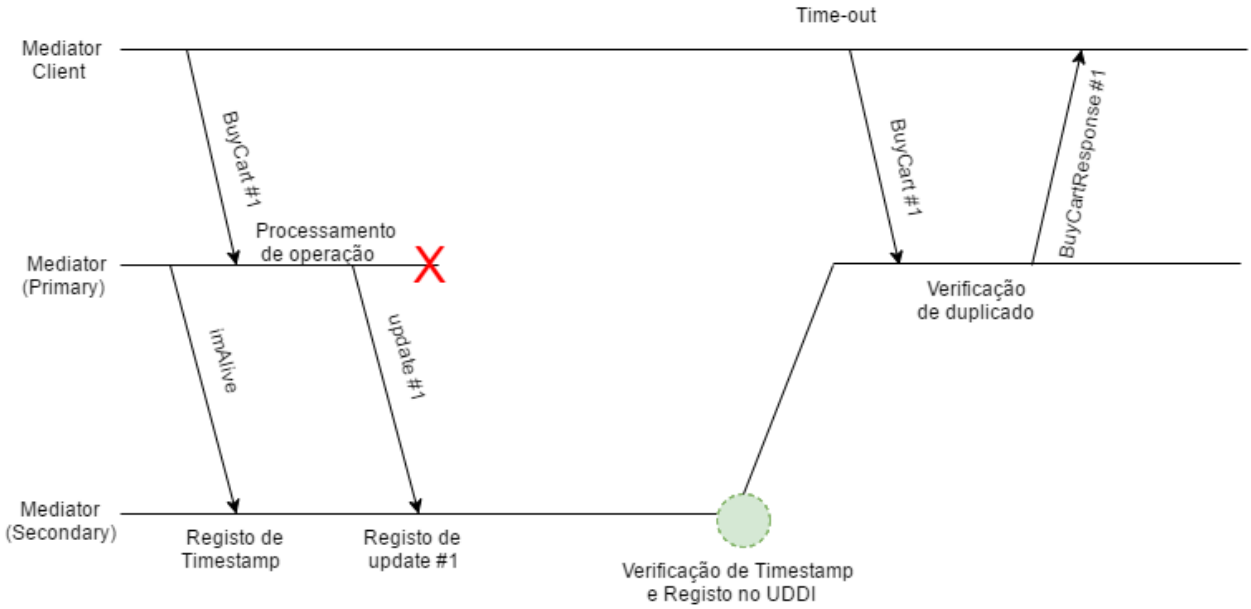


Figura 2 – Troca de mensagens com a solução implementada

Foram adicionados 2 testes (replacePrimary - nas classes BuyCartIT e AddToCartIT) que simulam a falha do mediador principal para que possam ser observadas facilmente as funcionalidades descritas acima.

Nota: Foi assumido que, tal como referido na aula teórica, um cliente não pode enviar um pedido antes de receber a resposta ao pedido anterior, pelo que não pode haver pedidos paralelos do mesmo cliente.