TÉCNICO LISBOA

Sistemas Distribuídos

2017/2018

Projeto, Parte 2
Tolerância a Faltas

RELATÓRIO

Grupo A30

Github: https://github.com/tecnico-distsys/A30-SD18Proj


81633 - João Henriques


82343 - Pedro Cunha


83434 - Beatriz Toscano

Realizamos a nossa segunda entrega utilizado a solução da primeira entrega apresentada pelo corpo docente da cadeira.

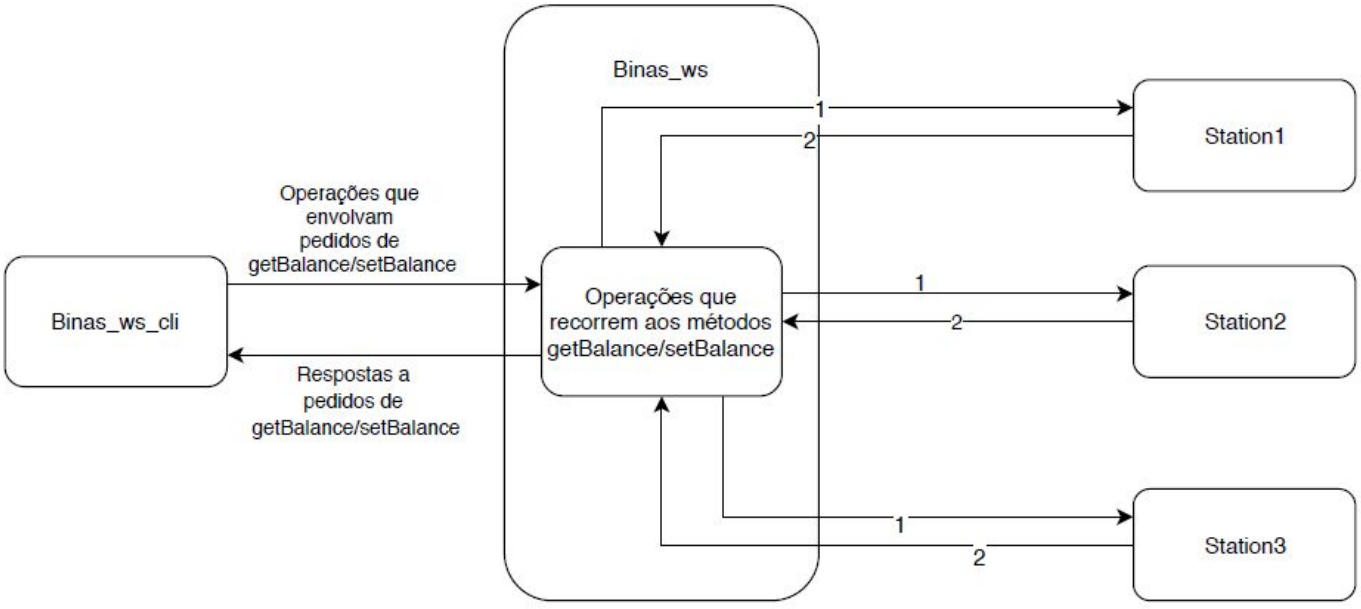
Começamos por definir as funções getBalance e setBalance, tendo nesta acrescentado um novo argumento (para além dos obrigatórios) do tipo *int* que serve para passagem da tag mais alta, e um novo tipo complexo BalanceView no WSDL. De seguida criamos a class Balance que contém um AtomicInteger para o balance e um AtomicInteger para a tag. O balance corresponde ao saldo da conta de um utilizador e a tag corresponde ao número de sequência de criação/alteração da instância da classe.

Em cada estação incluímos um ConcurrentHashMap que tem como chave o email do utilizador e como valor uma instância da classe Balance.

Ao utilizar o sistema de Quoruns, tendo em conta que existirão apenas três estações, garantimos que só poderá haver uma falha na leitura e escrita.

Para tratarmos das chamadas assíncronas, decidimos optar por invocação com Polling. Visto que o Callback usa uma flag para saber quando acabou a invocação, teríamos de criar uma flag para cada invocação. O Callback é geralmente usado, quando a resposta não altera o estado interno do programa ou então a resposta não é essencial para o restante funcionamento do programa. Visto que quando chamamos o getBalance é importante sabermos qual o maior *balance* e *tag*, decidimos usar Polling.

Figura da solução



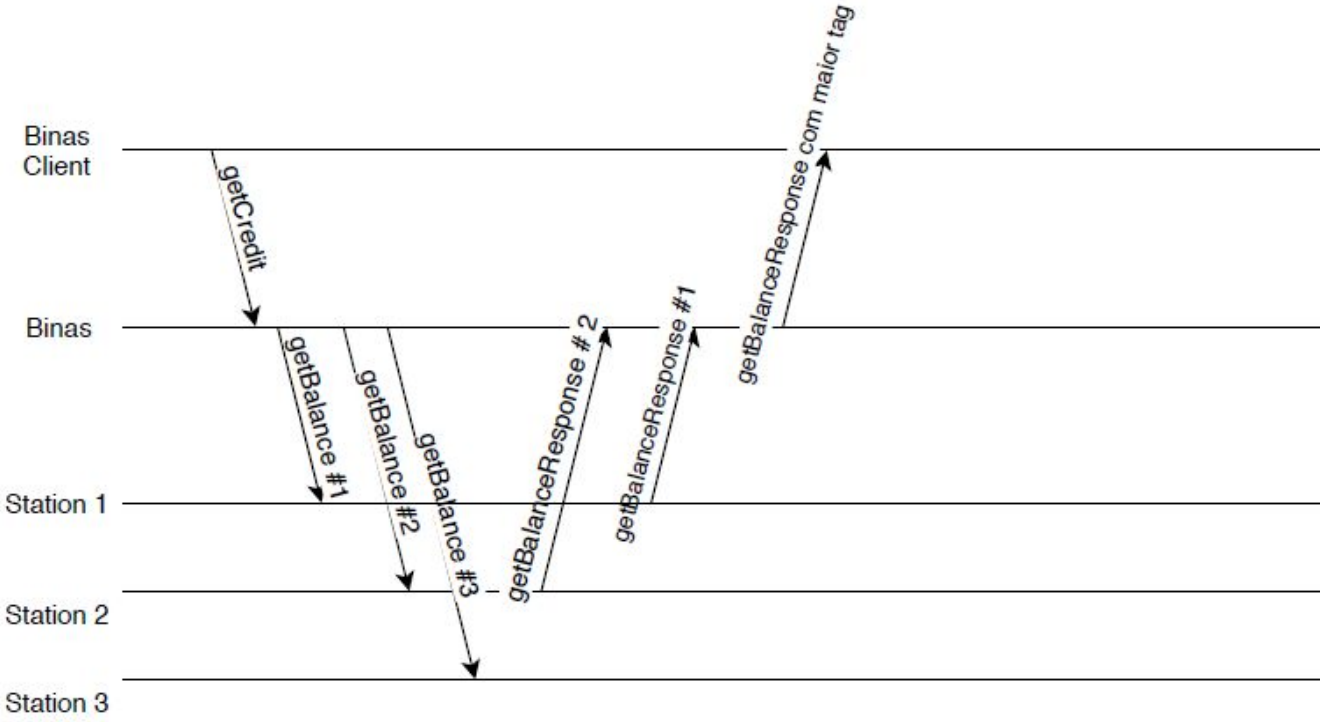
- 1 - Pedidos de getBalance/setBalance para um dado email na estação;
- 2 - Resposta aos pedidos para um dado email na estação;

Todos os pedidos de activateUser, rentBina, returnBina e getCredit efetuados por um Binas_client são tratados pelo servidor Binas de forma assíncrona através de

funções auxiliares getBalanceAux e setBalanceAux que chamam as funções getBalance e setBalance, recorrendo ao método de Polling, às 3 estações existentes.

Os pedidos (1) são efetuados às 3 estações, no entanto esperam apenas por duas respostas (2) devido ao QC, existe ainda um *delay* de 100 milissegundos, entre cada verificação do número de respostas. No caso de pedidos de getBalance são usados os valores da resposta de retorno que tenham a tag mais elevada, sendo que se as duas tags tiverem valores iguais, escolhemos a que lemos primeiro. Já nos pedidos de setBalance, não existe a necessidade de efetuar um retorno para o Binas_client, pelo que a resposta serve apenas como ACK.

Troca de mensagens do protocolo implementado



A imagem acima mostra o protocolo de troca de mensagens por nós implementado, nomeadamente a implementação da invocação do método getCredit por parte de um Binas_client.

Tendo as três Stations a correr, um Binas_sw e um Binas_Client, este último faz uma invocação o método getCredit do Binas_sw que por sua vez, efetua de forma assíncrona os pedidos de getBalance a todas as Stations, ficando de seguida a aguardar pela resposta de dois deles, verificando periodicamente, a cada 100 milissegundos, a sua chegada.

Após receber as duas primeiras respostas, independentemente das estações das quais são provenientes, compara a *tag* dos resultados de cada uma das

respostas, e devolve ao Binas_Client o *balance* correspondente à resposta com a *tag* maior.