Licenciatura em Engenharia informática e de Computadores — Alameda
Unidade Curricular de Sistemas Distribuídos — Ano Letivo 2017/2018

Relatório da 2º Entrega do Projeto (P2)

Tolerância a Faltas

Grupo A46

Hélio Domingos
83473

Miguel Regouga
83530

João Pina
85080

Repositório GitHub:

https://github.com/tecnico-distsys/A46-SD18Proj

Tolerância a Faltas

Para garantir a disponibilidade e a fiabilidade do sistema tendo em conta a relevância de informação que o servidor binas guarda, nomeadamente os saldos dos utilizadores, é implementado no projeto uma arquitetura de replicação ativa sobre este, que mantém o sistema disponível e funcional caso exista uma falha no Binas.

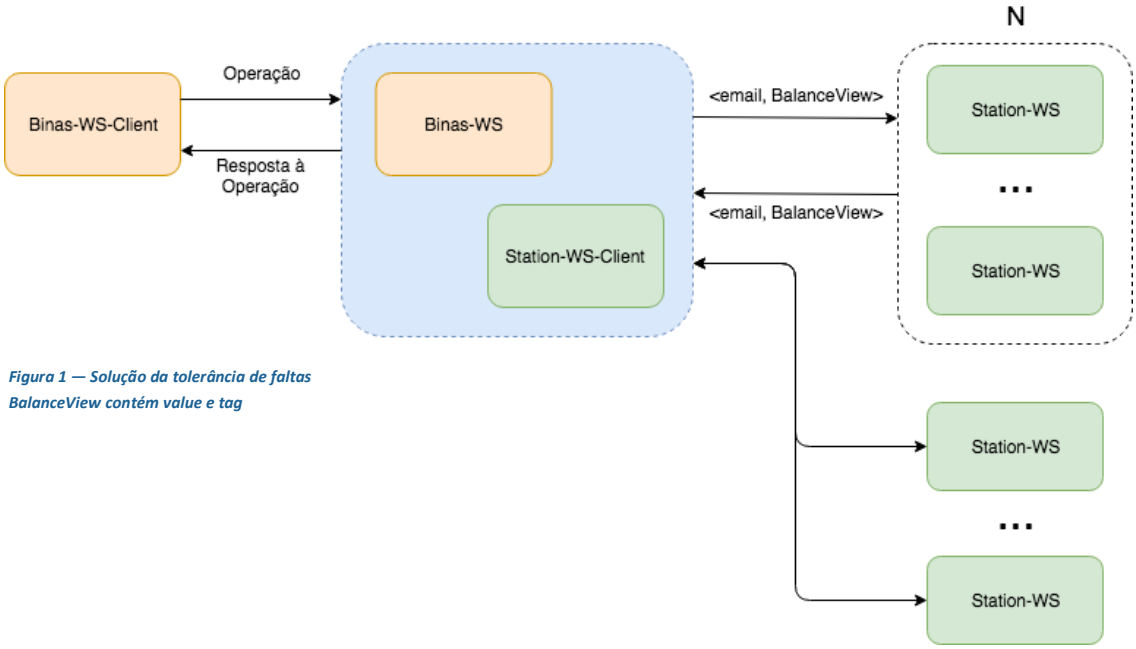

Figura 1 — Solução da tolerância de faltas
BalanceView contém value e tag

Replicação Ativa

O funcionamento base da replicação ativa garante que todos os servidores recebem pela mesma ordem os pedidos dos clientes. Assim, o primeiro passo para permitir esta funcionalidade é a replicação dos saldos dos utilizadores pelos N servidores de estação inicialmente existentes no sistema (representados na imagem com a letra **N**). Os servidores posteriormente criados não irão atuar como réplicas. Os **N** servidores passam a guardar o saldo de todos os utilizadores, que se encontra associado a uma tag.

Quorum Consensus

Para suportar a replicação ativa do projeto, é utilizado uma variante do protocolo Quorum Consensus; este protocolo incorpora um conjunto de subconjuntos de réplicas, tal que quaisquer dois subconjuntos se intersetam. Como dito anteriormente, cada réplica guarda o valor do objeto — que, no nosso caso, trata-se do saldo dos utilizadores — e o respetivo *tag*.

Esta tag será um inteiro que será utilizado como "versão" do saldo do utilizador. Ainda assim, o servidor Binas é *multi-threaded*, logo é possível que duas ou mais *threads* do Binas decidam ler ou escrever

concorrentemente no sistema replicado. Nesse cenário, ambas as *threads* comportam-se como clientes concorrentes. Contudo, as funções que gerem o *getBalance* e o *setBalance* (*write* e *read* do *BinasManager*) são *synchronized* para impedir que duas *threads* consigam aceder aos dados simultaneamente. Isto permite que a tag não inclua o *client-id*, ao contrário do previsto pelo protocolo Quorum Consensus, visto que apesar de o Binas ser *multi-threaded*, ele comportar-se-à como um único cliente. A não existência deste *client-id* torna o processo mais eficiente, uma vez que são efetuadas menos verificações.

As leituras previstas pelo protocolo são efetuadas pelo servidor principal, que lança uma chamada à função *read* a todos os gestores da réplica, que respondem com <email, balanceView>. O servidor aguarda por Q (*quorum*, calculado pela divisão inteira de N por 2 +1) respostas, escolhendo sempre o valor que tem a *tag* maior (mesmo que exista uma maioria de réplicas com outra *tag*).

N a fase de escrita o servidor executa uma leitura para obter o valor que tem a maior *tag*, estabelece uma nova *tag* com o seu valor incrementado em um, e envia uma chamada à função *write* a todos os gestores de réplica. O servidor espera então por Q *acknowledges* (ACK).

Operações Assíncronas

Para resolver as situações em que o cliente não pretenda ficar bloqueado à espera da resposta do servidor, é possível fazê-lo através da invocação de operações assíncronas. Para que isto seja efetuado, é possível utilizar dois métodos — *polling* ou *call-back*. No nosso caso, foi escolhido utilizar a primeira, dado que a complexidade extra que a segunda iria acrescentar não justificaria a sua implementação, uma vez que estamos perante um ambiente de poucas estações e clientes.

Troca de Mensagens

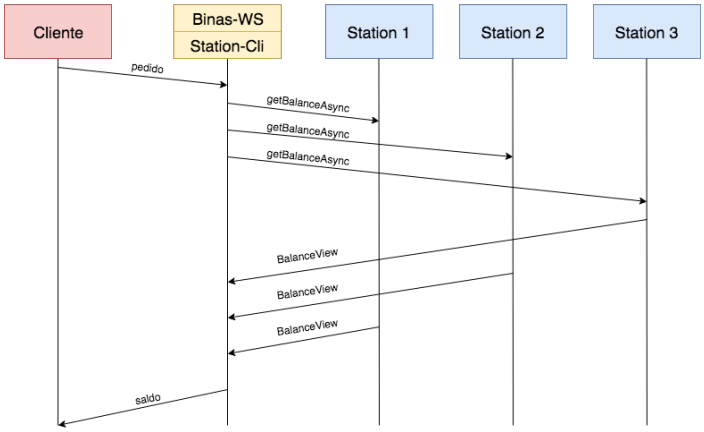

Figura 2 — Troca de mensagens quando é chamada a função de leitura (getBalance)

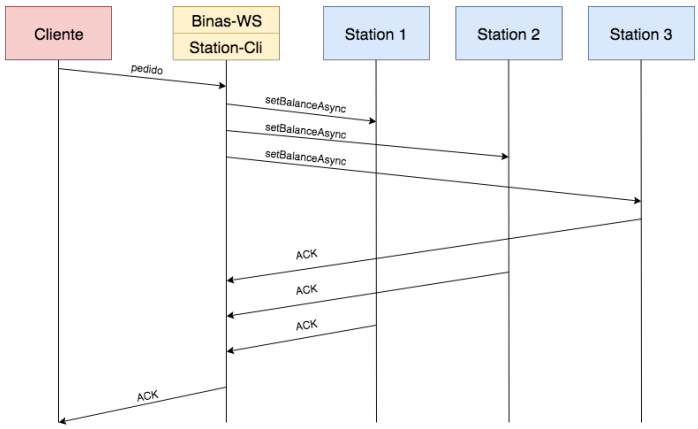

Figura 3 — Troca de mensagens quando é chamada a função de escrita (setBalance)

O cliente pede a leitura do saldo ao servidor, que por sua vez pede a cada estação a leitura do saldo. É então devolvido o saldo (guardado numa balance view) para o servidor, que o retoma para o cliente.