

# 2ª Entrega

SISTEMAS DISTRIBUÍDOS  
2º SEMESTRE – 2017/2018



## Grupo 8

<https://github.com/tecnico-distsys/T08-SD18Proj.git>



André Fonseca  
84698



Diogo D'Andrade  
84709

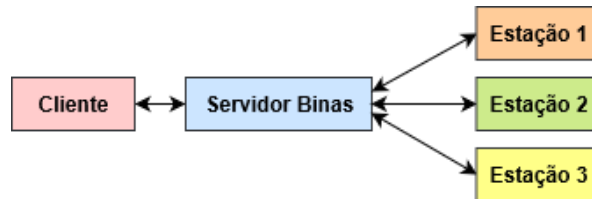


Leonor Loureiro  
84736

## MODELO DE FALTAS

- faltas silenciosas dos processos
- faltas silenciosas do canal
- no máximo, existe uma minoria de gestores de réplica que falha em simultâneo

## PROTOCOLO QUÓRUM CONSENSUS



### Cada cópia remota do registo do user guarda:

- saldo dos utilizadores
- respetiva tag, que identifica a versão
  - Incluindo a clientID na tag seria possível permitir escritas concorrentes nas stations por multiplos clientes, neste caso, multiplas threads do Binas. No entanto, no contexto do projeto, a frequência das operações de escrita não justifica esta implementação. Mecanismos de sincronização asseguram a consistência dos dados em operações de escrita paralela, sem a utilização da clientID.

**Quórum:**  $Q > 3/2 \rightarrow Q = 2$

Quórum de leitura e o quórum de escrita escolhidos são iguais uma vez que ao executarmos a fase de leitura na operação de atualização do saldo, a escolha de um quórum de escrita mais pequeno não trás nenhuma vantagem.

### LEITURA DO SALDO

#### CACHE HIT – BINAS TEM REGISTO DO UTILIZADOR

1. Binas retorna o valor local do saldo

#### CACHE MISS – BINAS NÃO TEM REGISTO DO UTILIZADOR

2. Binas invoca o método `getBalance` de todas as Estações.
3. Aguarda por  $Q$  respostas.
4. Seja  $v_{\max}$  o valor recebido correspondente à maior tag  $t_{\max}$ .

*Fase de Leitura*

#### A – Todos os valores recebidos têm a mesma tag

5. Binas retorna  $v_{\max}$

#### B – Recebe pelo menos um valor desatualizado relativamente aos outros

6. Binas invoca o método `setBalance( $v_{\max}$ ,  $t_{\max}$ )` de todas as Estações.
7. O Binas aguarda por  $Q$  acks.
8. O Binas retorna  $v_{\max}$ .

*Writeback*

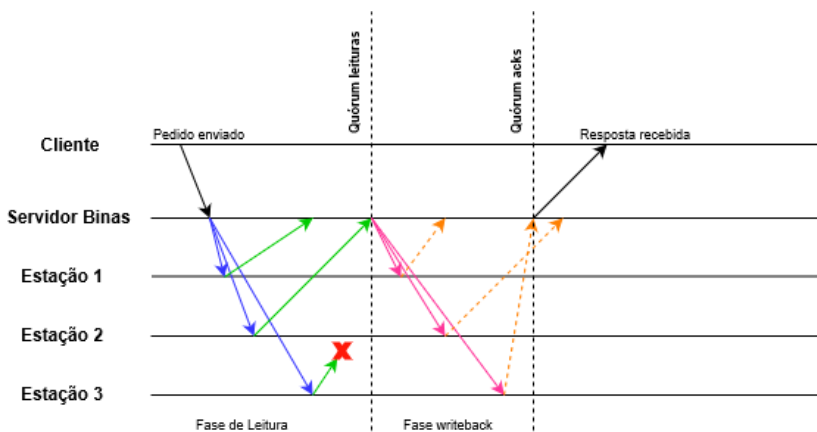


Figura 1 - Troca de mensagens `getBalance`

## ATUALIZAÇÃO DO SALDO

### CACHE HIT – BINAS TEM REGISTO DO UTILIZADOR

1.  $t_{\max}$  corresponde ao valor local da tag
2. O Binas invoca o método  $\text{setBalance}(v_{\text{new}}, t_{\max}+1)$  de todas as Estações
3. O Binas guarda por Q acks
4. O Binas retorna

Fase de Escrita

### CACHE MISS – BINAS NÃO TEM REGISTO DO UTILIZADOR

1. Binas invoca o método  $\text{getBalance}$  de todas as Estações
2. O Binas aguarda por Q respostas
3. Seja  $t_{\max}$  a maior tag dos valores recebidos
4. O Binas invoca o método  $\text{setBalance}(v_{\text{new}}, t_{\max}+1)$  de todas as Estações
5. O Binas guarda por Q acks
6. O Binas retorna

Fase de Leitura

Fase de Escrita

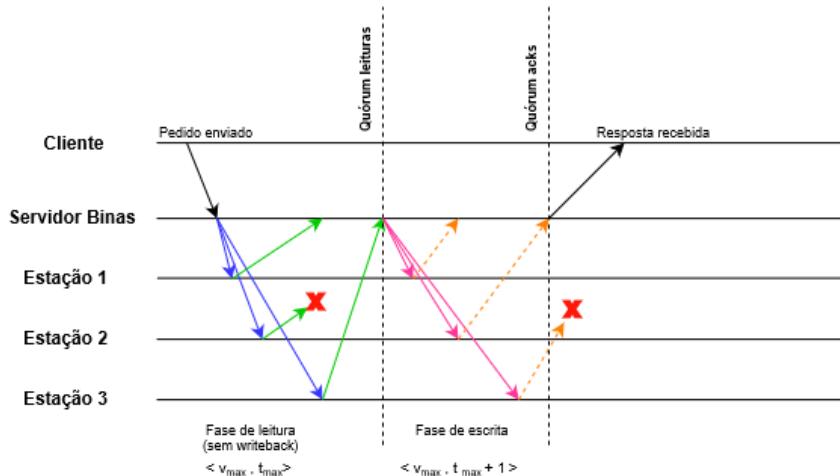


Figura 2 - Troca de mensagens  $\text{setBalance}$

## OTIMIZAÇÕES

### Fase de leitura:

- O Binas tem um sistema de cache, sendo que a operação de leitura somente precisa de usar o sistema replicado caso o Binas não tenha uma cópia local do registo do utilizador.
- Uma vez que o Binas é o único escritor do sistema replicado, esta otimização não cria inconsistências, relativamente aos valores do saldo e da tag.

### A fase de writeback:

- Garante a consistência no caso em que operação de leitura é feita concorrentemente com uma operação de escrita, uma vez que garante que se uma operação de leitura retorna o resultado de uma operação particular de escrita, então qualquer operação de leitura que começa após o fim da primeira leitura vê um resultado pelo menos tão recente.
- No caso em que uma operação de escrita falha a meio, completa a escrita.
- Na maioria dos casos, a fase de writeback não é executada uma vez um pedido de escrita é enviado para todas as estações e na ausência de falhas, todas as estações atualizam os seus registos, pelo que na fase de leitura todos retornam valores com a mesma tag.

## PROTOCOLO NAS ESTAÇÕES

- Em resposta à operação de leitura do saldo, a estação devolve o valor do saldo e a este associado.
- Em resposta à operação de atualização do saldo:
  - Caso o valor da tag seja superior ao armazenado na estação, então o valor do saldo é reescrito, juntamente com a tag.
  - Caso contrário, não é feita qualquer atualização.
  - Em ambos os casos é retornado um ack.

## MODELO DE INTERAÇÃO

- **Sistema assíncrono:** invocações remotas com *callback*.
  - *Polling* é mais eficiente quando se trata de um sistema dedicado, ou quando temos garantias de repostas muito rápidas. No entanto, estes casos não se verificam no contexto do nosso projeto, pelo que com o *callback* gastam-se menos recursos da máquina, o que justifica a complexidade extra da implementação.