




Grupo T07
SISTEMAS DISTRIBUÍDOS 2017/2018

Fotografia			
Nome	Filipa Marques	David Coimbra	Lúcia Lisboa
Número	57842	84708	84738

GitHub URL: <https://github.com/tecnico-distsys/T07-SD18Proj>

Tolerância a faltas

Aplicação protocolo *Quorum Consensus*:

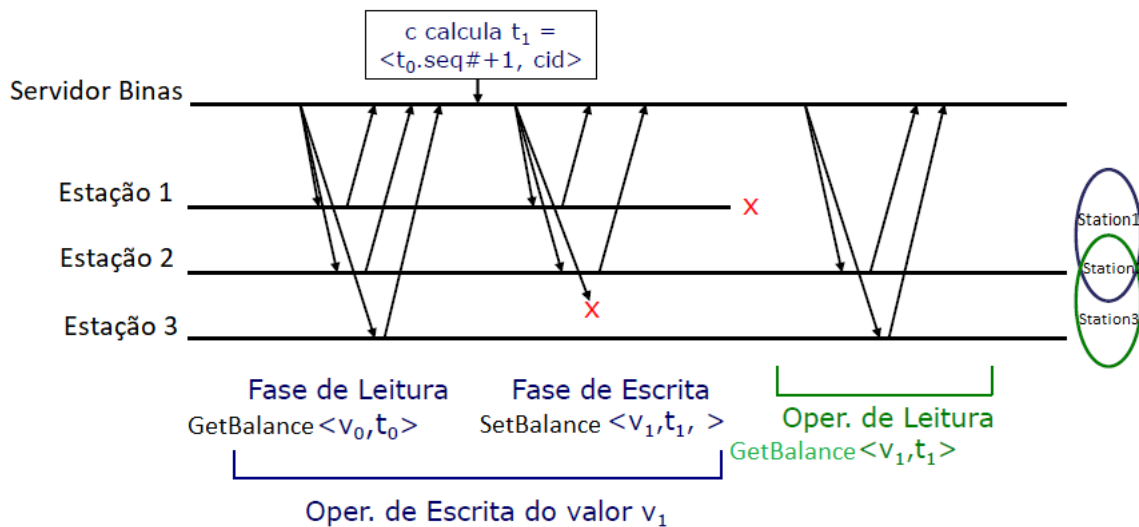


Figura 1 – Funcionamento base da solução de Quorum Consensus aplicada no projeto. A descrição da solução encontra-se no texto abaixo.

As operações de escrita ($setBalance$) do processo são sempre precedidas por uma operação de leitura; o servidor Binas começa por fazer um pedido de leitura ($getBalance$) a todas as réplicas, que devolvem o par valor-tag que têm guardado para o utilizador em questão. O servidor analisa Q respostas ($Q = \frac{1}{2}$ estações $+1$) para encontrar a maior tag e incrementa a parte sequencial da tag encontrada. De seguida, o servidor envia um pedido de escrita do novo valor e nova tag (máxima) a todas as estações. No entanto, o pedido de escrita feito à estação 3 perde-se e não chega à estação, sendo o *quorum* mantido através das respostas (*ack*) das estações 1 e 2. Após a operação de escrita terminar, a estação 1 pára de funcionar e é feito um novo pedido de leitura (por exemplo, uma operação $getCredit$) a todas as estações. As estações 2 e 3 respondem com os valores mais recentes guardados (E2 com o par $\langle t_1, v_1 \rangle$ e E3 com o par $\langle v_0, t_0 \rangle$). O servidor, ao analisar as respostas, seleciona a que tem a maior tag (valor devolvido pela estação 2), sendo essa a resposta enviada ao utilizador.

Implementou-se um modelo de chamadas assíncronas com *callback*, pois um sistema baseado em *polling* implicaria uma interrogação periódica de todas as réplicas a contactar. Num sistema com um elevado número de réplicas, despende-se demasiado processamento a lidar com as interrogações, ao passo que uma implementação com *callback* apenas tem de lidar com a interrupção gerada. Isto torna o *callback* uma solução mais robusta em comparação ao *polling*, apesar deste último ser mais simples de implementar.

Para lidar com eventuais falhas do servidor binas, quando é invocado algum utilizador que não se encontra registado no servidor Binas verifica-se se esse utilizador está registado nalguma das estações através da invocação do método $getBalance$ (uma leitura de quorum consensus). Em caso positivo, esse utilizador é novamente adicionado ao servidor binas, e o seu saldo recuperado (através do $\langle val, maxTag \rangle$ retornado pela invocação do $getBalance$). Esta solução permite não perder todos os dados dos utilizadores em caso de falha do binas, no entanto dados como se o utilizador possui uma Bina consigo serão perdidos pois não estão a ser guardados em gestores de réplicas nem existem mecanismos de tolerância de faltas aplicados a estes.

Como esta solução permite uma recuperação de dados suficientemente robusta, não se considerou necessário implementar outras variantes do protocolo. Assume-se que as réplicas têm fiabilidade igual entre si (tornando uma implementação de pesos variáveis redundante).

Troca de mensagens

Descrição das operações de leitura:

- No lado do servidor Binas:
 - Servidor envia, de forma assíncrona, um pedido (getBalance) a todas as estações;
 - Aguarda por um número Q respostas, sendo $Q = \frac{1}{2} \# \text{estações} + 1$
 - Por fim, retorna ao utilizador a resposta correspondente à maior tag recebida nas Q respostas
- No lado das estações:
 - Após receber um pedido getBalance pelo servidor Binas, responde com o par valor-tag correspondente ao email associado.

Descrição das operações de escrita:

- No lado do servidor Binas:
 - Executa leitura para obter maxTag
 - Envia, de forma assíncrona, um pedido de escrita (setBalance(val, maxTag+1)) a todos os gestores de réplica
 - Espera por Q acks
 - Retorna
- No lado das estações (ao receber setBalance(val, tag)):
 - Se a tag que recebe for mais recente, atualiza os seus valores de val e da tag
 - Responde com ack