| Sistemas Distribuídos **2011/2012**<br>Solução desenvolvida no projeto final<br>**Trabalho elaborado pelo grupo A0403** | **Grupo Segurança:**<br><br>João Loff – Nº 56960<br>Alexandre Almeida – Nº 64712<br>Tiago Aguiar – Nº 64870 | **Grupo Replicação:**<br><br>Hugo Sequeira – Nº 63925<br>Edgar Santos – Nº 64753<br>Viteche Ashvin – Nº 64878 |
|---|---|---|

## Introdução

Com os requisitos funcionais implementados, foi necessário abordar outros requisitos não funcionais, importantes ao funcionamento do projeto fiabilidade e segurança. Nesta fase foi necessário replicar o nosso sistema, através de um servidor de nomes e várias réplicas para cada servidor do operador. Foi definido o nosso modelo de faltas, que suporta falhas silenciosas e bizantinas. E foi definida em termos de segurança, os requisitos autenticidade e integridade no sistema.

Dentro do grupo de replicação e tolerância a falhas foi necessário desenhar um protocolo de tolera-se as falhas do modelo, o qual foi discutido e testado ao longo de várias iterações entre os elementos para que se conseguisse respeitar todos os requisitos pedidos. O grupo de segurança teve que adaptar os conceitos de segurança adquiridos nas aulas teóricas, também pesquisando mecanismos já existentes que podíamos usar, bibliotecas disponíveis e desenvolvimento de mecanismos novos. Neste campo criamos uma entidade de certificação e os nos nossos próprios certificados, inspirados nos certificados X509.

## Desenvolvimento

Baseado no protocolo quóruns referido nas aulas teóricas, foi necessário definir o nosso grau de replicação através de uma replicação ativa. Na nossa primeira análise, admitindo que em N réplicas temos uma falha bizantina (F=1), consideramos um grau de N=2F+1 ou N=4F+1 para a nossa solução.

Analisando cada caso, verificamos que utilizando um grau N=2F+1, iriamos obter um problema grave, pois no quórum de repostas aos pedidos, em caso de discordância nas duas réplicas do quórum, não íamos ter a certeza de qual a resposta certa, pois podemos não ter uma maioria. Além disso não garantimos que contactando a terceira réplica, teríamos a resposta certa, pois neste grau de replicação, apenas garantimos um servidor em comum em quaisquer duas diferentes operações.

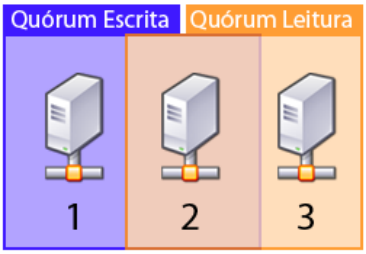


Figura 1. Dois diferentes quóruns em duas diferentes operações, no grau de replicação N=2F+1

Um exemplo bastante simples que demonstra este problema, é realizar uma escrita, em que apenas um dos servidores pode não receber o pedido, estando o pedido de escrita bastante atrasado (R3), não respondendo, e em seguida se efetuar-mos uma nova leitura podemos obter uma falha bizantina (R2), além de uma versão antiga (R3). Neste caso como referimos, não vamos conseguir decidir por consenso qual é a resposta correta, já que não há uma maioria de respostas dentro nem fora quórum com conteúdo e versão igual.

| | 1º Comando: Escrita (valor=10, versão=2) | | | 2º Comando: Leitura | | |
|---|---|---|---|---|---|---|
| Réplica ID | R1 | R2 | R3 - DELAYED | R1 | R2 - BIZ | R3 |
| Resposta | ACK | ACK | -- | 10 | 500 | 5 |
| Versão Respondida | 2 | 2 | -- | 2 | 3 | 1 |

Tabela 1. Exemplo de um problema grave no grau N=2F+1

Detectamos vários problemas neste grau de replicação. Por isso concluímos que seriam necessárias mais réplicas. Principalmente réplicas em comum entre quaisquer duas operações diferentes. Pensamos então no grau de replicação N=4F+1. Neste caso, conseguimos resolver todos os problemas detectados, obtendo assim três quóruns em comum em duas operações diferentes.

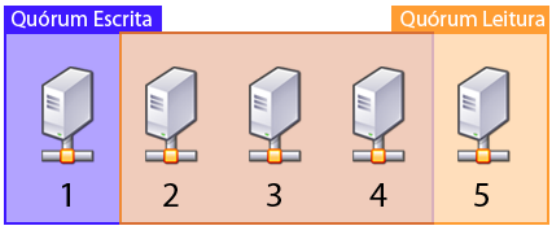


Figura 2. Dois diferentes quóruns em duas diferentes operações, no grau de replicação N=4F+1

Embora seja um bom grau de replicação, temos o problema de estamos a necessitar 5 réplicas para o nosso sistema. Tornando o sistema pouco optimizado, necessitando de muitos recursos. O grande desafio então foi conseguir obter uma solução num grau intermédio e conseguindo respeitar os requisitos enunciados.

### A Nossa Solução

Na nossa solução implementamos um grau de replicação N=3F+1, garantindo assim as seguintes importantíssimas propriedades:

1. Em quaisquer dois quóruns de operações diferentes existem sempre duas réplicas em comum nesses quóruns.
2. O cliente quando faz uma escrita só fica satisfeito com a escrita até que um quórum lhe responda.
3. Cada servidor/operador têm internamente uma versão, caracterizado por um inteiro.
4. Nas leituras, em caso de não haver consenso dentro do quórum, é sempre necessário questionar a réplica fora do quórum atual.

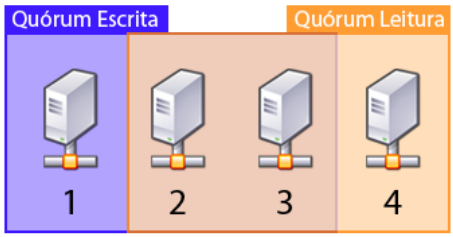


Figura 2. Dois diferentes quóruns em duas diferentes operações, no nosso grau de replicação, N=3F+1

### Escrita

Caso o cliente deseje fazer uma escrita no sistema, previamente, o nosso *Frontend* pergunta a todas as réplicas registadas as suas versões, esperando um quórum de respostas. Em seguida, incrementa a maior versão respondida. De seguida, o comando de escrita, que inclui a nova versão, é enviado assincronamente para todas as réplicas. Por fim, aguarda que um quórum de réplicas lhe confirme a escrita e analisa se o comando levantou uma exceção dentro do quórum, caso contrário, termina o comando com sucesso. Para confirmar a escrita, designámos que cada réplica devolve a nova versão ao *Frontend*.

### Leitura

Caso o cliente deseje fazer uma leitura no sistema, o *Frontend* comunica com todas as réplicas registadas, o pedido de leitura e aguarda por um quórum de respostas. Com o quórum de respostas dado, analisa se o comando levantou alguma exceção dentro do quórum, caso contrário, executa uma votação entre as respostas pela resposta correta. Se através da votação das respostas do quórum, não conseguiu chegar a um consenso pela resposta correta, surgem duas possibilidades a analisar para implementar a propriedade *4.*

Nesta altura, depois de esperarmos pelo quórum e votado, pode ter chegado ao nosso *Frontend* a resposta da réplica restante. Caso tenha chegado, vota-mos novamente com as todas as respostas, as de dentro do quórum mais a resposta que entretanto chegou. Aqui é garantido que se chegue a um consenso e exista uma resposta em maioria, pois é garantido pela propriedade *1.* e *2.* que mesmo que exista uma falha bizantina nesta leitura e uma resposta com versão antiga, vamos obter uma maioria pois temos dois servidores com a mesma versão e conteúdo.

Caso entretanto, não tenha recebido a resposta da restante réplica, comunica-se sincronamente com esta réplica e o seu resultado é o escolhido pois, novamente pela propriedade *1.* e *2.* garantimos que, nesta situação, este servidor esteve na operação passada e têm a versão correta.

Na nossa solução não aceitamos respostas atrasadas na rede, num pedido de leitura atual, pois para cada comando diferente, existe uma nova lista de resposta e uma nova lista de *handlers* assíncronos de *WebService*, portanto mesmo que chegue uma resposta atrasada a mesma não será considerada, pois será escrita numa lista antiga em memória, pendente de ser eliminada pelo *Garbage Collector* devido à referencia remota presente.

### Votações

Para qualquer votação, o número mínimo de ocorrências iguais de respostas necessárias para ser escolhida é o inteiro da metade do número de réplicas arredondado por cima: MinOcurrencias = RoundUp(Q/2).

### Excepções

Para verificarmos que ocorreu verdadeiramente uma exceção em qualquer pedido, têm de existir uma maioria de respostas a levantarem uma exceção, logo no nosso caso têm de ocorrer pelo menos três exceções para notificarmos a mesma ao cliente.

### Segurança

Para o caso em que uma falha bizantina responder em nome dos outros, implementamos uma entidade de certificação que emite certificados digitais que garantem a autenticidade e integridade nas mensagens transmitidas entre cliente e servidores[1]. Em seguida segue um exemplo, demonstrando o correto funcionamento da solução, onde as células a azul representam o quórum escolhido e a verde o quórum contactado para desempatar segundo a propriedade *4.*:

| | 1º Comando: Escrita (valor=10, versão=2) | | | | 2º Comando: Leitura | | | |
|---|---|---|---|---|---|---|---|---|
| Réplica ID | R1 | R2 | R3 - DELAYED | R4 | R1 | R2 - BIZ | R3 | R4 |
| Resposta | ACK | ACK | -- | ACK | 10 | 500 | 5 | 10 |
| Versão Respondida | 2 | 2 | -- | 2 | 2 | 3 | 1 | 2 |

Tabela 2. Exemplo do correto funcionamento da solução implementada.

[1] Coulouris, George F. Distributed Systems: Concepts and Design. 5th ed. Addison-Wesley, 2012. Pag. 495