Relatório

***LINKS***

Para o desenvolvimento do projeto decidimos desenhar a nossa solução recorrendo ao uso de *Perfect Links* e *Best-Effort Broadcast*.

Escolhemos usar *Perfect Links* em primeiro lugar por garantirem que se um processo **pi** e um processo **pj** são corretos então qualquer mensagem **m** enviada por **pi** para **pj** é eventualmente entregue por **pj** (*validity*)**.** No caso do nosso projeto isto garante que se um dado processo normal recebe um PREPREPARE então este processo irá entregar um PREPARE e da mesma maneira se receber um PREPARE irá entregar um COMMIT.

O uso de *Perfect Links* também nos garante que nenhuma mensagem para um mesmo processo é entregue mais que uma vez (*no duplication*) e ainda que nenhuma mensagem é entregue sem ter sido enviada (*no creation*).

Na nossa abordagem de implementação dos *Perfect Links* o processo principal lança threads secundárias que estão encarregues de enviar as mensagens aos outros processos e esperar a receção de um *acknowledge*. Para evitar sobrecarregar o sistema, recorremos ao uso de um *timeout* que é o tempo que esperamos para reenviar a mensagem enquanto não for recebido o *acknowledge*.

Vale a pena mencionar que por cada vez que reenviamos a mensagem aumentamos o tempo deste *timeout*.

Optámos por não usar os *Authenticated Perfect Links* dado que não nos interessa garantir autenticidade. Esse fator já é garantido de outra forma como iremos explicar mais detalhadamente adiante.

***BROADCAST***

A opção do uso do *Best-Effort Broadcast* partiu desde logo por este tipo de protocolo nos dar várias garantias como: validade, não duplicação e não criação.

Para além disso, analisando o algoritmo *The Istanbul BFT Consensus Algorithm* garante-nos propriedades que nos permitem escolher este algoritmo.

O *Reliable Broadcast* também seria uma abordagem possível de se usar.

Em relação à propriedade BCB4 do algoritmo *Byzantine Consistent* Broadcast, é algo que não nos interessa garantir, uma vez que apenas pretendemos assegurar que cada processo correto entrega as suas próprias mensagens.

***TROCA DE MENSAGENS DO ALGORITMO***

Na nossa implementação existem dois tipos de troca de mensagens: cliente – servidor & servidor – servidor.

Vale a pena referir que dentro do leque de mensagens servidor – servidor existem 3 tipos das mesmas. As mensagens de PREPREPARE (enviadas pelo líder), as mensagens de PREPARE e ainda as mensagens de COMMIT.

Todas as mensagens trocadas entre cliente – servidor e as mensagens de PREPARE são assinadas digitalmente usando cifra publica/privada com o algoritmo RSA. A assinatura consiste então em cifrar com a chave privada o *hash* da informação valiosa a ser trocada. Graças a isto conseguimos garantir as propriedades de segurança: autenticidade, integridade e não repúdio.

Assinamos as mensagens de PREPARE com o objetivo de termos na nossa implementação uma base que permita facilmente implementar a mudança de líder do algoritmo Istambul BFT.

Por sua vez, de modo a garantir a integridade das restantes mensagens trocadas entre servidor – servidor (PREPREPARE e COMMIT) usamos um mecanismo de hmac em que fazemos o digest da informação valiosa com o algoritmo SHA3-256 e enviamos junto com o payload para o outro lado da comunicação. Do lado do recetor, este irá fazer também o digest da informação valiosa e em seguida irá compará-lo com o hmac recebido e se ambos coincidirem a integridade da mensagem não foi comprometida.

Cada processo (cliente/servidor) possui um par de chaves pública-privada sendo que a chave privada é de conhecimento exclusivo ao seu detentor e a chave pública conhecida por todos aqueles que vão comunicar com esse processo.

As chaves usadas para o bom funcionamento do protocolo encontram-se presentes na diretoria *SEC-Project/Secure-Messages/keys*.

***STRONG READS***

Para a implementação das *strong reads* apenas é retornado ao cliente os valores da conta correspondente ao estado das contas após a execução do último bloco decidido em consenso. Quando o cliente faz um pedido de BALANCE em modo strong, o seu pedido é enviado a todos os servidores, sendo que todos respondem com o valor que contêm e posto isto o cliente espera que exista um valor que esteja presente em f +1 das respostas.

***WEAK READS***

Para implementarmos as *weak reads* definimos que os servidores da blockchain têm um sistema que guarda *snapshots*. Na nossa implementação, um *snapshot* corresponde a um mapa válido (garantia de que este mapa já foi o mapa atual das contas numa instância passada) que contém o estado das contas, assim como a lista de assinaturas dos servidores que na mesma instância em que o *snapshot* correspondia ao estado das contas atual partilhavam esse mesmo estado.

Para a atualização dos *snapshots* cada réplica envia o seu estado atual das contas e a assinatura deste estado para todos os outros servidores. Posto isto, irá ficar à espera por mensagens dos outros servidores com um estado das contas dos mesmos e a respetiva assinatura. Após receber um pedido, cada réplica verifica se o estado das contas recebido é igual ao seu estado das contas atual e caso assim seja adiciona a assinatura do servidor que envio a mensagem à lista de assinaturas. Este processo repete-se até se atingir uma maioria de consenso.

Vale a pena mencionar que todos os *snapshots* são atualizados a cada X instâncias (no nosso caso 2).

Quando uma réplica recebe um pedido de *weak read*, envia o seu snapshot (estado das contas e lista de assinaturas) para o cliente de modo a que este consiga verificar que o estado das contas recebido foi numa dada instância o estado das contas atual na *blockchain*. Esta verificação é possível graças à lista de assinaturas.

***OUTROS TIPOS DE ATAQUE***

Descartamos mensagens iguais provenientes de processos com o mesmo port durante a mesma instância de consenso como forma de prevenir replay attacks.

Para evitar impersonation attacks na comunicação entre cliente e servidor usamos assinaturas digitais nos pedidos enviados entre os mesmos.

***CONCLUSÃO***

Concluindo, com o algoritmo implementado qualquer cliente que se ligue ao serviço da blockchain tem a garantia de que a sua transação é aceite podendo ou não ser realizada, dependendo do estado da blockchain.