Relatório - CSD Projeto

Daniel Batista

June 21, 2021

1 Introdução

Este projeto foi desenvolvido no âmbito da cadeira de Confiabilidade de Sistemas Distribuídos e teve como objetivo desenvolver um sistema de blockchain descentralizado e confiável tolerante a falhas bizantinas. A implementação consiste numa blockchain com consensos *proof of work* onde desde que não haja mais de f falhas em 3f+1 nós o sistema funciona corretamente.

Repositório do projeto: *https://github.com/dfmb99/CSD-Blockchain*

2 Especificação da implementação

2.1 Estrutura e operações dos nós

Os nós comunicam através do protocolo BFT-Smart. BFT-Smart é um protocolo tolerante a falhas bizantinas de replicação de máquinas de estado, é uma biblioteca de código aberto Java mantida pela unidade de pesquisa LaSIGE da Universidade de Lisboa.

Cada nó do sistema distribuído tem operações que os clientes podem chamar através de pedidos REST. Só depois da chamada da operação é que existe a comunicação de dados entre os nós para se atingir consenso. As operações que a blockchain suporta são:

- **transferCoins()** - broadcast de uma transação para o sistema
- **getBalanceOf()** - balanço de uma dada chave pública
- **getAllConfirmedTransactions()** - transações finalizadas no ledger
- **getTransactionsOf()** - transações finalizadas de uma dada chave pública
- **getLastBlock()** - informação do último bloco finalizado
- **getUnconfirmedTransactions()** - transações não finalizadas
- **mineBlock()** - broadcast de um bloco
- **totalCoins()** - total de moedas no ledger

2.2 Consensos e estrutura dos blocos

O algoritmo de consensos usado foi *proof of work* que é o algoritmo de consensos atualmente utilizado no protocolo da Bitcoin, a ideia geral de proof of work é que para que um dado bloco seja válido vai ser necessário que a hash do bloco contenha um x número de zeros iniciais o que só será possível testando uma série de valores para o nonce até que obtenhamos uma hash que seja válida. Sendo esta hash facilmente verificável mas dificil de gerar, torna-se um bom candidato a uma prova de que um dado trabalho computacional foi realizado. Cada bloco na implementação da blockchain tem 7 campos: *previous block hash*, *nonce*, *difficulty*, *height*, *timestamp* e *transaction data*, o que é equivalente ao protocolo da Bitcoin.

```
SHA256(previous_block_hash + nonce + difficulty + height + timestamp + transaction_data)
```

Dado que o *nonce* é o único campo que poderá ser alterado, é o campo que os miners que pretendem minerar blocos para receber as recompensas terão que ir alterando até descobrirem uma hash que tenho o mesmo número de zeros iniciais á dificuldade, e só assim é que as nodes que rodam o protocolo vão aceitar o bloco como válido.

2.3 Regras de consensus

As regras de consensus são as regras nas quais todos os nós corretos obecedem e para que o sistema distribuido seja correto e tolere falhas bizantinas não pode haver mais de f falhas em 3f + 1 nós. Cada nó tem que obedecer ás seguintes regras para o protocolo funcionar corretamente:

- Máximo de 100 transações por bloco
- Dificuldade é 5
- Máximo de recompensa por bloco é 10 moedas
- A hash de cada bloco tem de ser válida
- Todas as transações têm de ser válidas

A regra de ter um número máximo de transações serve para que não haja blocos muito grandes que permite que a validação e tranferência dos blocos entre os nós seja trivial. A dificulade ser 5 significa que para um bloco minerado ser considerado válido a hash do bloco tem de ter inicialmente pelo menos o valor da dificuldade (neste caso 5 zeros). E dado que na implementação da blockchain não existe nenhuma operação para obter moedas grátis, a única maneira de obter moedas inicialmente é fazendo a *proof of work* para minerar blocos, e inserindo uma transação no bloco onde o *sender* é null, o *receiver* uma chave pública da qual temos a chave privada e uma quantidade de moedas não superior a 10, caso este valor seja superior a 10 os nós não aceitam os blocos, porque não respeita uma das regras de consensus

2.4 Criptografia assimétrica

Na implementação da blockchain para ser possível que toda a informação seja pública mas ao mesmo tempo que haja o conceito de contas em que só o dono poderá eventualmente gastar as moedas a tecnologia de criptografia assimétrica tem que ser usada. A tecnologia de criptografia usada na implementação foi a *Elliptic Curve Digital Signature* por fornecer maior segurança com tamanhos de chaves menores, mais adequado para máquinas com baixa *bandwith*, baixo poder de computação, menos memória e possui implementações de hardware mais fáceis.

2.5 Transações

As transações são compostas por:

- sender
- receiver
- amount
- timestamp
- signature

O *sender* é a chave pública do endereço do qual se pretender enviar as moedas, o *receiver* a chave pública do enderenço de qual se pretende receber as moedas, *amount* a quantidade de moedas a enviar na transação, *timestamp* o timestamp de quando a transação foi enviada e a *signature* a assinatura feita pelo *sender* com a a sua chave privada juntamente com todos os campos da transação.

Para os nós do protocolo aceitarem uma dada transação têm que verificar todas as transações confirmadas e verificar que não há nenhuma *signature* igual á *signature* da transação que se pretende verificar num dado bloco, assim podemos garantir que não há *double spend* de moedas. Porque transações iguais por definição tem os campos iguais e portanto assinaturas tem que ser iguais, e caso isso acontece os nós verificam a existência de *double spend* e rejeitam o bloco.

2.6 Transações confirmadas / não confirmadas

Quando uma transação é enviada para um dos nós, o nó verifica se a transação é válida e guarda a transação em memória mas não fica registada em nenhum bloco (**transação não confirmada**), os *miners* são os que pedem aos nós as transações não confirmadas e fazem a proof of work para formar blocos que posteriormente são enviados aos nós que vão aceitar o bloco caso seja válido e respeita as regras de consensos mencionadas anteriormente, só depois de o bloco ter sido aceite pela maioria dos nós a transação é considerada finalizada (**transação confirmada**).

3 Problemas da implementação

3.1 Persistência dos dados

Na implementação atual, tanto as transações não confirmadas como os blocos já finalizados são guardados em memória, o que faz com que eventualmente dado que a memória dos nós é limitada, os nós *crasham* quando o tamanho do ledger é maior que a memória RAM do sistema. Uma solução para este problema é em vez de guardar os dados dos blocos finalizados em memória guardar os dados de forma persistente em disco e dado que os discos têm um tamanho muito superior á memória do sistema o ledger poderá crescer muito mais até haver problemas de armazenamento. Eventualmente problemas de armazenamento vão existir outra vez no futuro, mas pode-se simplesmente adicionar mais discos á máquina onde o nó está a rodar o que é muito mais eficiente e barato do que adicionar mais memória RAM.

3.2 Verificação de double spending

Como explicado anteriormente, para prevenir o *double spending* de transações cada nó terá que verificar que não há nenhuma transação com uma assinatura igual e para isso ser feito, cada nó terá que comparar as assinaturas de todas as transações uma a uma, o que cria um *bottleneck* muito grande dado que o número de transações totais no ledger é ilimitado e cresce linearmente com cada bloco finalizado. Eventualmente quando o ledger tiver um tamanho considerável de milhares de transações finalizadas, os tempos de verificação de cada transação vão ser muito altos o que vai prejudicar a performance do protocolo.

References

https://github.com/bft-smart/library
https://bitcoin.org/bitcoin.pdf
https://arxiv.org/ftp/arxiv/papers/1508/1508.00184.pdf