University of Southampton

COMP 6212 Computational Finance (2018-2019)
Report of Assignment 4

Blockchain and Smart contracts

Team members:
Siying ZHANG
30542731
Mark Distribution 33.3%

XinWANG
30171083
Mark Distribution 33.3%

Zhihan WANG
30005124
Mark Distribution 33.3%

Word count:
3129

1 Introduction

In this assignment, we are going to choose a blockchain infrastructure to manage a bonus scheme inside a company. We are going to use a private blockchain, named Hyperledger Fabric to achieve this task as it has many good properties, such as high security, high speed. the consensus is the process to guarantee the order of transaction, validate them and make a agreement on the state of transactions in order to avoid malicious attack and double-spending problem. We write a smart contract which can automatically execute contractual treaty to guarantee the transaction processing in the company available and secure.

2 Blockchain

2.1 General description

A blockchain is essentially a distributed database of records, or public ledger of all transactions or digital events that have been executed and shared among participating parties[6]. A blockchain acting like a layer on the Internet, operates on a Peer-to-Peer Network. It does not require trust among the participants and there is no need for a centralized institution as well.

A blockchain network is like assemblage of a series of blocks. Each block contains data about transaction information, a hash number of current block and a hash number of previous one. Both of the hash numbers are used for verification. The verification process helps to keep the network secure as a hash number is extremely hard for hackers or attackers to reverse. Therefore, the crypto-information is kept secure and cannot be changed secretly. Once a block is built in the chain, an algorithm, known as consensus begins to work in the blockchain protocol. This new block would be broadcasted and verified by a majority of the participants in the system without the the existence of centralised institution or a third party. And then every single transaction ever made is recorded in blockchain and would never be erased.

In the blockchain, two types of keys, say public key and private key, generated by a wallet are used. A public key is like an address and we can use it to transfer money to a specific person. At the same time, a private key gives its owner access to their digital assets[8]. Cryptocurrencies are brought to play in blockchain. It is token based and a tool wrote using code in the blockchain.

To be more precisely, a blockchain is a platform, in which business activities(transactions) happen any time. Therefore, three main parts in the heart of blockchain are smart contract(programmable transaction processing), ledger, state(i.e. ownership of an asset) respectively.

Bitcoin is the first platform and as the technology evolves, there are some other platforms available now[4] and designed for different industries.

2.2 Comparison

Five popular blockchains, Bitcoin, Ethereum, R3, Ripple, Hyperledger, are to be discussed in this sector, with general description, pros and cons and scenarios they can perform well. All of these would be reasons why we choose one specific blockchain to manage the bonus scheme in this assignment.

Firstly, I classify blockchains into two types, public ones and private/federated ones as there are two important points needed to be considered firstly when choosing a blockchain. They are accessibility and speed respectively. The details are shown in tabel 1.

Table 1: Public v.s. Private/federated blockchains

Feature	Public	Private/Federated
Accessibility	Open to read/write	Permissioned to read/write
Speed	Slow	Fast
Examples	Bitcoin, Ethereum	Hyperledger, R3, Ripple

Anyone can participate in the public blockchain to download data, run a new block and verify transactions without requirements for permission. In this case, costs spent on building or maintaining infrastructure are saved. However, a disadvantage is that the current business model is easily disrupted. Besides, most public blockchains are based on proof-of-work consensus algorithm which makes the speed in a very low level. It takes a large amont of time to mine and veify blocks. Therefore, for companies or groups

who have their own specific use case, a private or federated blockchain would be a good choice. However, private/federated blockchains also have their own shortcomings. With a central group or company, the decentralisation of the network is disturbed in some extent. The trust between participants and central groups is to be required.

As there are no permission requirements for public blockchains, the public ones can be used in any scenario, though might not perform very well. As referred in table 1, Bitcoins and Ethereum are two common public blockchains. We will not use them in this assignment as private ones are much better. Hence, the following discussion mainly focus on the private or federated blockchains. The comparison information is shown in table 2[12].

Table 2: Comparison among Private/federated blockchains

Feature	R3 corda	Ripple	Hyperledger
Field	Financial services sector	Banks	Different industry-specific projects
Smart contracts supported	Yes	No	Yes
Transaction cost	No	Yes	No
Cryptocurrency	No build-in cryptocurrency	XRP	No build-in cryptocurrency
Require mining	No	No	No

According to this table, all of them do not require mining. This is because unlike Bitcoin or Ethereumthe, consensus is not achieved by mining in these three blockchains. In addition, we can see that Hyperledger caters to different industries and allows for several business-related blockchain technologies[5], in contrast to others only designed for a specific field, like banks.

2.3 Hyperledger

Hyperledger is like an umbrella and supports five different frameworks[7], namely Fabric, Burrow, Sawtooth, Iroha, and Indy. The consensus algorithms used in these 5 frameworks are different as well. Fabric is tailored to the needs of business customers who must work within standardized enterprise trust systems[15]. Our task in this assignment is to choose a blockchain to manage a bonus scheme which happens inside a company. Based on the discussion above, the blockchain we are going to use here is Hyperledger Fabric.

3 Consensus

3.1 The Rationale

In terms of consensus, the architecture of blockchain system is distributed and the network of nodes manage all the transaction information in the ledger. In distributed system, the order of received transactions might be difference. Besides, the state of transactions might be difference due to network delay and computational power of host. Thus, the consensus algorithms can ensure the order of transactions and validate corresponding transactions in blocks in order to avoid malicious attack and fake consumption [1]. In other words, the process of consensus is to guarantee agreement on the order of transactions, validation and the update state of transactions between peers. In addition, two properties, safety and liveness, can guarantee the agreement between nodes. Safety is to ensure the order of inputs and correspond output, and update the state of transactions once the same set of transactions are received. And liveness is to ensure every validated transactions wii be received by nodes.

In Bitcoin, Proof-of-work algorithm is used to solve double-spending problems under consensus mechanism. However, different blockchain requirements can be satisfied, such as scalability, security and confidentiality and the consensus algorithm is completely difference in Hyperledger Fabric, compared with it in Bitcoin. In this assignment, Kafka algorithm is used. In addition, there are three phases of consensus in Hyperledger Fabric, that are endorsement, ordering and validation. In endorsement, nodes have different roles and are able to decide whether needs to support the transactions when they received proposals from client based on corresponding policy during the lifecycle of a transaction. Ordering phase is to make a agreement on the order of transaction to the global ledger for a set of ordered transaction within a period of time. And validation phase is to check endorsement policy and double-spending in order to validate the correctness of ordered transactions.

3.2 The pros and Cons

Distributed system is comprised by permissioned nodes, which provide the reliability for all nodes in ledger, in other words, there will not be malicious forgery of transactions information. Thus, it is not necessary for nodes to proof their works. Besides, transaction information is managed by Orderer Service Node (OSN) so that the order of transaction can be guaranteed and fork problems could be avoid [10]. The rationale of order service which is comprised by OSN is to use Kafka algorithm to implement consensus. Besides, the validated information of transaction will be sent to OSN via SDK in client, and then after validation, the information which is encapsulated as Kafka format will send to Kafka [11]. OSN will order all the transaction information after above steps. In addition, it is not necessary for all nodes to participant in consensus mechanism due to the architecture of consensus in Hyperledger. In other words, Kafka can guarantee the order of transaction and avoid lose when producers transmit information to consumers via Kafka. But, it rely on zookeeper

The other advantages are that finality of Kafka of happens in few second and it support crash fault tolerance while it cannot guarantee the system to make a agreement when nodes are malicious or faulty, compared with Byzantine Fault Tolerance (BFT) [14]. The limitation of permissioned voting-based approach is that it cannot satisfy a scenario between high speed and scalability.

3.3 Comparison

In terms of permissioned consensus approaches, the speed and finality in permissioned voting-based approach are as good as them in permissoned lottery-based. Despite the fact that delivery faults might be happen when a large number of untrusting orderers are employed, pluggable algorithm is provided by Hyperledger Fabric to address this problem. This algorithm is more flexible because specific algorithms can be used to satisfy the different requirements of application. Compared with Ethereum, it is a permissionless blockchain and nodes have the same role. In addition, the time of produce a block cannot guarantee, while bitcoin can ensure the time within 600 seconds and Ethereum can ensure the time about 15 seconds. Moreover, Proof-of-work is more flexible because each node is individual, that means it does not need other nodes, and nodes are able to decide whenever join the network, in other words, it is not necessary for all the nodes to join [13]. Thus, it cannot reach a good finality. However, the serviceability and flexibility in consensus algorithm in Hyperledger Fabric based on BFT are not as good as them in Proof-of-work algorithm because all nodes have to provide service online and the network will fault once over half of nodes are unusable. So the finality can be guaranteed.

In this scenario, there are many nodes are involves in, such as the staffs in the company and partner company which can be spend bonus, so Kafka algorithm is better than Redundant Byzantine Fault Tolerance (RBFT) which will spend more time to make consensus when the more nodes existed on the network [2]. In addition, it is not necessary for all nodes to participant to reach consensus due to the architecture of consensus in Hyperledger. However, RBFT does not allow, that means all the nodes must be connected.

4 Smart contract

4.1 Aim

The smart contract called chaincode. The every contract has a unique address[3]. In this scenario, the smart contracts are the program between company, employees and partners, which contains functions and states of the contracts. We assume the partners are in on the technology, the employees can pay tokens to partners directly.

The aim of the smart contract is to guarantee the company can reward employees with tokens based on their performance. The employees can choose one partners to pay tokens for merchandises they want. The certain amount of tokens can transfer from the the employees account to the partners account.

4.2 Actors and Roles

Externally Owned Accounts(EOA): The company, employees and partners are the externally owned accounts in this system. The externally owned accounts can own private keys and trigger smart contract [9]. The transaction can be generated between these accouts.

Miners: The miners verify the transactions between the externally owned accounts, for example,they verify the identifications of employees. After they finished verification, they can get rewards.

4.3 Functional Requirements

This system contains two smart contracts: the payroll contract and shopping contract.

The Payroll Contract records the time, cycle, and amount of tokens paid employees. This case is used of company and employees. the data is pairwise. The one is the company account, it should subject the amount of tokens paid to employees. Another one is the account of employee, it should add the amount of tokens the employee can get according to the performance of employee.

Functional requirements of Payroll Contract:

Function received: The company rewards the employees tokens based on their performance. The amount of tokens employees deserved should add to the account of employees. After that, the state of the contract should be RECEIVED.

Function continued: We need to confirm the people who want to take tokens out is the employee of the company. When we make sure the people are who they say they are, the state will become CONTINUED.

Function Interaction: The payroll contract need to interact with shopping contract. We need to design a contract to ensure a smooth handover. The smart contract can make sure the original contract has finished. When it finish the state will become FINISHED

The shopping contract records the amount of tokens and the product the employee chosen. This case is used of employees and partners. The data is pairwise. The one is the employees account, it should subject the amount of tokens paid for the product provided by the partner. Another one is the account of a certain partner, it should add the amount of tokens paid for the product.

In this use case the relationship between the company and employees is one-to-many. The figure 1 shows the process of the payroll contract and the tokens transfer from company to employees.

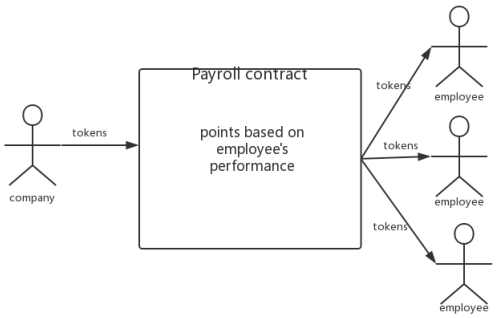

Fig. 1: Process of payroll contract.

Functional requirements of Shopping Contract:

Function checked: When employees chose the product, the smart contract should make sure the employee has enough tokens paying for the product. If the employee has enough money the state will become CHECKED. If the employee does not have enough tokens the state will be CANCELLED.

Function Tokenscanpay: When state is CHECKED, the employee is ready to pay. We need to make sure the company the employee want to sent tokens to the selected company, one of partners. After we checked, the state will become TOKENCANPAY.

Function paid: when employee pays for the product, the state sets to PAID and the tokens will transfer to the account of the partner which provide the product, and the ownership of the product will change from the partner to the employee.

In this use case the relationship between the employee and partners is one-to-many. The figure 2 shows the process of the shopping contract and the tokens transfer from employee to partners.

4.4 Non-Functional Requirements

Emergency stop

There are two situations can trigger the emergency stop.

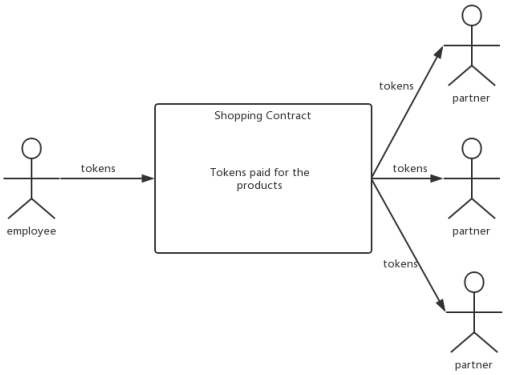

Fig. 2: Process of shopping contract.

Firstly, when owner of the contract want to stop the contract, the owner can sent requirement, we should make sure the requirement came from the owner, then the contract admin can stop this deployed contract.

Another one is that when bugs happened in the network, If the bug is not serious, it will not affect the transaction, the contract admin will trigger the emergency stop, all actions of the contract will be postponed. If the bug is serious, the funds should be return.

Rate limits The maximum points the employee can take out the points one time during a half year. If the frequency is more than a time, the second transaction will be canceled or require a special permission. The rate limiting can prevent substantial changes happen or a rapid drain on funds.

Security requirement–Delay

When a malicious activity happen, we need give time for the owner of the contract to response it. Combining with the emergency stop can prevent the money taken out by malicious users. In this system the owner of the contract can withdraw the contract within 4 weeks after the contract been set.

4.5 Limitations

1. The smart contract is new technology, the user maybe have not use is before, so the users need to spend time to understand it.
2. It difficult to make changes. In this system, we set the pending period is 28 days. After 28 days, it difficult to change. For example, if employee bought a product and that product doesn't work after a month, however the data is already written, it is technically difficult to change.

4.6 pseudocode with explanation

The smart contract to realize that company rewards the staff member additional points based on their performance. The smart contract written in Solidity is shown as follow.

```
contract Payroll {
    address employee;
    uint PointsReceived = 0;
    mapping (adress => unit) Amountpaid;



    function unpdatingPointsReceived () internal{
    require(msg.sender = employee);
    PointsReceived += msg.value
    }

    function () external payable{
```

```
    unpdatingPointsReceived ();

    }

    function isTransferFinish() public {
    ProcessEndTime = ContractSetTime + PendingTime;
    require( now >= ProcessEndTime ); //auction did not yet end    require(!ended);

    ended = true //changing conditions

    employee.transfer(Amountpaid);  //interacting with other contracts
    return true;
    }
}
```

Firstly, the smart contract initialize the constructor. When company sends the points to employees, the code is executed autonomously. The key words of payable is require to received points. Using PointsReceived counts the number of points in the contract. Using function *unpdatingPointsReceived*() to update the amount of points the employee has already owned, the total amount of points equal to the amount of points the employee already has add the amount of points transferred this time. The function *isTransferFinish*()*public* can be divided into three phases. The first phase is checking conditions. During the PendingTime, the owner of the contract can withdraw the contract. So the finish time of the contract is date of contract set plus PendingTime. The second phase is changing condition. The final phase is to make interaction with function that employee send points to partners for buying products.

This smart contract to realize that the staff can spend their bonus on products provided by partners.

```
contract Shopping {
    address employee;
    mapping (address => uint) public balances;

    function paid(adress partner, uint Amoutpaid){
    balances[msg.sender]  -= Amoutpaid;
    balances[partner]  += Amountpaid;
    }

    modifier costs(uint price) {
        if (msg.value <= price) {
            return false;
        }
}

    modifier TokensCanPay (){bool  employed = false ;
        for (uint i=0; i< employees.length; i++){
                if (employee[i] == msg.sender){
                        employed=true;
                }
        }
        require (employed);
        _;
    }

    functionisTransferFinish() public {
    ProcessEndTime = ContractSetTime + PendingTime;
    require( now >= ProcessEndTime )

    require( ended = true )

    return true;
    }
}
```

The $modifiercosts(uintprice)$ is used to check the points of the employees whether it is enough to pay the products they want to buy. If the points of the employee is more than points product needed, the transaction will go on. If points is less than needed the transaction will return false to stop it. The function $modifierTokensCanPay()$ can make sure the person who want to take out the tokens is the employee who own these tokens. After checking, if the person the people are who they say they are. The $functionpaid(adresspartner, uintAmoutpaid)$ which is inserted where the special symbol '_' will be executed. If the requirement can not be meet, this function will not be executed.

The function $functionpaid(adresspartner, uintAmoutpaid)$ can realize that tokens will be transferred from the employee's account to the partner's.

```
contract RateLimit {
  uint enabledAt = now;

  modifier enabledEvery(uint t) {
    require(now >= enabledAt, "Access is denied. Rate limit exceeded.");
    enabledAt = now + t;
    _;
  }
  function f(uint amount) public enabledEvery(0.5 year) {
     require(amount <= 10000);
  }
}
```

It can limits the frequency of requirements. In this scenario, we set the each employee just can take out of tokens twice a year. The maximum number of tokens they can take out is 10000. The contract can prevent contract owner take out large amount of tokens causing a rapid drain on funds.

```
bool private stopped = false;
address private owner;
modifier isAdmin() {
    if(msg.sender != owner) {
        throw;
    }
    _
}
function toggleContractActive() isAdmin public
{
}
modifier stopInEmergency { if (!stopped) _ }
modifier onlyInEmergency { if (stopped) _ }

function deposit() stopInEmergency public{}
{

function withdraw() onlyInEmergency public{}
```

The emergency stop also called circuit breaker. It can be trigger by the admin who is the trust third part. We need verify the identification of the admin by modifier $modifierisAdmin$. It also can be trigger by the rules written in code. There are two modifier in this code. The one is changing boolean by admin, the transaction in the will be suspended. Another one is allow people take out their money of bug is serious.

```
struct Withdrawal {
    uint AmountPaid;
    uint time;
}
mapping (address => uint) private balances;
mapping (address => Withdrawal) private requestedWithdrawals;
uint constant PendingTime = 28 days;

function requestWithdrawal() public {
```

```
    if (balances[msg.sender] > 0) {
        uint amountToWithdraw = balances[msg.sender];
        balances[msg.sender] = 0;

        requestedWithdrawals[msg.sender] = RequestedWithdrawal({
            amount: amountToWithdraw,
            time: now
        });
    }
}
```

This function can delay the contract finish time. In this scenario, we set the pending time is 28 days. During this time, any functions can be called. So if malicious activity happen the owner of contract has time to take out all tokens. We withdraw all tokens in the contract, the *AmountPaid* is the count of tokens in the contract.

5 Conclusion

We use Hyperledger Fabric blockchain to manage the bonus scheme using Kafka consensus algorithm, which improve speed and finality compared with standard proof of work algorithm and the RBFT in Hyperledger Indy. With the smart contract guarantee the transaction process, the bonus scheme is completely automatic and much safer and reliable.

References

1. Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y., et al.: Hyperledger fabric: a distributed operating system for permissioned blockchains. In: Proceedings of the Thirteenth EuroSys Conference. p. 30. ACM (2018)
2. Aublin, P.L., Mokhtar, S.B., Quéma, V.: Rbft: Redundant byzantine fault tolerance. In: 2013 IEEE 33rd International Conference on Distributed Computing Systems. pp. 297–306. IEEE (2013)
3. Azaria, A., Ekblaw, A., Vieira, T., Lippman, A.: Medrec: Using blockchain for medical data access and permission management. In: 2016 2nd International Conference on Open and Big Data (OBD). pp. 25–30. IEEE (2016)
4. Baliga, A.: Understanding blockchain consensus models. In: Persistent (2017)
5. Cachin, C.: Architecture of the hyperledger blockchain fabric. In: Workshop on distributed cryptocurrencies and consensus ledgers. vol. 310 (2016)
6. Crosby, M., Pattanayak, P., Verma, S., Kalyanaraman, V., et al.: Blockchain technology: Beyond bitcoin. Applied Innovation **2**(6-10), 71 (2016)
7. Dhillon, V., Metcalf, D., Hooper, M.: The hyperledger project. In: Blockchain enabled applications, pp. 139–149. Springer (2017)
8. Fay, T., Paniscotti, D.: Systems and methods of blockchain transaction recordation (Oct 6 2016), uS Patent App. 15/086,801
9. Karamitsos, I., Papadaki, M., Al Barghuthi, N.B.: Design of the blockchain smart contract: A use case for real estate. Journal of Information Security **9**(03), 177 (2018)
10. Sankar, L.S., Sindhu, M., Sethumadhavan, M.: Survey of consensus protocols on blockchain applications. In: 2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS). pp. 1–5. IEEE (2017)
11. Sousa, J., Bessani, A., Vukolic, M.: A byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform. In: 2018 48th annual IEEE/IFIP international conference on dependable systems and networks (DSN). pp. 51–58. IEEE (2018)
12. Valenta, M., Sandner, P.: Comparison of ethereum, hyperledger fabric and corda. [ebook] Frankfurt School, Blockchain Center (2017)
13. Vukolić, M.: The quest for scalable blockchain fabric: Proof-of-work vs. bft replication. In: International workshop on open problems in network security. pp. 112–125. Springer (2015)
14. Website:: Hyperledger architecture, volume 1 (2017), https://www.hyperledger.org/wp-content/uploads/2017/08/Hyperledger$_{A}rch_{W}G_{P}aper_{1C}onsensus.pdf$
15. Website:: The top 5 most popular blockchains: Pros and cons (2018), https://achievion.com/blog/top-5-popular-blockchains-pros-cons.html