中山大学数据科学与计算机学院本科生实验报告

(2019 年秋季学期)

课程名称： 区块链原理与技术　　任课教师： 郑子彬

年级	2017	专业（方向）	软件工程
学号	17343160	姓名	赵孟阳
电话	13242886623	Email	1442196038@qq.com
开始日期	2019.11.14	完成日期	2019.12.13

一、 项目背景

本项目是区块链+供应链金融的一个方案，针对传统供应链金融中核心企业的信用无法在整个供应链中传递以及交易信息不透明化的缺点，进行新的方案设计。

在本方案中，实现了将供应链上的每一笔交易和应收账款单据上链，同时引入第三方可信机构来确认这些信息的交易，例如银行，物流公司等，确保交易和单据的真实性。同时，支持应收账款的转让，融资，清算等，让核心企业的信用可以传递到供应链的下游企业，减小中小企业的融资难度。

二、 方案设计

存储设计：

使用结构体 Bill 的可变长度数组保存账单，Bill 结构体中包括了账单涉及双方的地址，金额，合约自动支付时间，合约是否已经结束。其中账单的欠款方在进行账单转移后仍记录为原来的账单的欠款公司，从而使核心企业的信用可以传递，方便中小企业贷款。

数据流图：

```
public class Bill extends Contract {
        public RemoteCall<String> showAccount(BigInteger i);
        public RemoteCall<String> bank();
        public RemoteCall<Tuple5<BigInteger, BigInteger, String, String, Boolean>> Bills(BigInteger param0);
        public RemoteCall<TransactionReceipt> init();
        public RemoteCall<BigInteger> getBillNum();
        public RemoteCall<BigInteger> getBalanceOf(String tokenOwner);

        public RemoteCall<TransactionReceipt> applyForFunde(BigInteger _value, BigInteger _lastTime);
        public RemoteCall<TransactionReceipt> purchaseAndMakeBill(String _seller, BigInteger _value, BigInteger _lastTime);
        public RemoteCall<TransactionReceipt> transferBill(String _rootCompany, String _receiver, BigInteger _value);
        public RemoteCall<TransactionReceipt> autoPayForBill();

        public static Bill load(String contractAddress, Web3j web3j, Credentials credentials, ContractGasProvider contractGasProvider);
        public static RemoteCall<Bill> deploy(Web3j web3j, Credentials credentials, ContractGasProvider contractGasProvider);
}
```

核心功能介绍：

1. 银行监管：每个账户交易的发出方为自身，所以可以在账户交易前进行授权，保证交易的发出经过银行验证，交易发出方有效。

```
struct Trader{
        bool isPermitted;//在交易者发生购买或转让订单前必须先由银行进行授权
    }
    //账单的签订或转移需要银行给出权限
    function giveRightToTrader(address trader) private{
        require (msg.sender == bank,
            "Only bank can give right to trade"
        );
        require (traders[trader].isPermitted == false);
        traders[trader].isPermitted = true;
    }
```

2. 购买商品并签订账单（purchaseAndMakeBill）:操作者作为买家，与 seller 签订账单，经过 lastTimeh 后自动归还，并将交易 event 保存

```
function purchaseAndMakeBill(address _seller,uint _value,uint _lastTime)public//购买物品，签订账单
    {
        traders[msg.sender].isPermitted = true;
        require(
            traders[msg.sender].isPermitted == true,
            "Only getting right from bank can trade"
        );
        require(_seller != msg.sender, "Self-trader is disallowed.");
        require(_seller != address(0) && _value >= 0,"Not valid address or value");
        require(_lastTime > 0, "The lastTime need to bigger than zero");

        //签订账单
        Bills.push(Bill({
            value : _value,
            billEndTime : (now + _lastTime),///合约自动支付时间
            from : msg.sender,
            to : _seller,
            isOvered : false
        }));
        traders[msg.sender].isPermitted = false;
        emit Purchased(msg.sender,_seller,_value,(now + _lastTime));
    }
```

3. 应收账款的转让上链：指定需要转移账单的核心企业 rootCompany，查找 Bills 中是否存在此人的该账单，如果存在转让 value 的账额，并修改原账单的欠款金额，新账单的归还时间为旧的截止时间。转移账单 even 被记录。

```
function transferBill(address _rootCompany, address _receiver, uint _value)public//转移账单
    {
        traders[msg.sender].isPermitted = true;
        require(
            traders[msg.sender].isPermitted == true,
            "Only getting right from bank can trade"
        );
        require(_receiver != msg.sender, "Self-trader is disallowed.");
        require(_receiver != address(0),"Not valid address");
        require(_value > 0, "The value need to bigger than zero");

        bool isSuccess = false;
        //查找区块中是否存在来自rootCompany 欠 msg.sender的账单

        for(uint i = 0; i < Bills.length; i++){
            if(Bills[i].from == _rootCompany && Bills[i].to == msg.sender && Bills[i].isOvered == false){
                if(Bills[i].value >= _value){
                    Bills.push(Bill({
                        value : _value,
                        billEndTime : Bills[i].billEndTime,
                        from : _rootCompany,
                        to : _receiver,
                        isOvered : false
                    }));
                    Bills[i].value -= _value;
                    isSuccess = true;
                    traders[msg.sender].isPermitted = false;
                    emit BillTransformed(_rootCompany,msg.sender,_receiver,_value);
                    break;
                }
            }
        }
        require(isSuccess == true,"transfering Bill failed");
    }
```

4. 利用应收账款向银行融资上链：通过查找融资方拥有的所有被偿还账单的总偿还金额是否大于融资的金额，如果满足条件，银行向融资方借款，并签订融资应偿还的账单。融资 event 被保存。

```
function applyForFunde(uint _value,uint _lastTime)public//向银行申请融资,不能大于它持有的账单金额综合
    {
        require(bank != msg.sender, "Self-funde is disallowed.");
        require(_value > 0, "The value need to bigger than zero");
        require(_lastTime > 0, "The lastTime need to bigger than zero");
        bool isSuccess = false;
        uint totalValue = 0;
        for(uint i = 0; i < Bills.length; i++){
            if(Bills[i].to == msg.sender && Bills[i].isOvered == false){
                totalValue += Bills[i].value;
                if(totalValue >= _value){
                    Bills.push(Bill({
                        value : _value,
                        billEndTime : now + _lastTime,//合约自动支付时间
                        from : msg.sender,
                        to : bank,
                        isOvered : false
                    }));
                    transferFrom(bank,msg.sender,_value);
                    isSuccess = true;
                    emit FundeApplied(_value,(now+_lastTime));
                    break;
                }
            }
        }
        require(isSuccess == true,"applying for funde failed");
    }
```

5. 应收账款支付结算上链：启动账单自动偿还功能，超过截止日期的账单自动根据信息进行账户转账，并将账单改为结束状态。转账 event 被记录。

```
function autoPayForBill()public//支付应付账单
    {
        for(uint8 i = 0; i < Bills.length; i++){
            //条件
            require(now >= Bills[i].billEndTime && Bills[i].isOvered == false);
            require(Bills[i].value >= 0,"This Bills %d's value is not valid");
            //影响
            Bills[i].isOvered = true;
            //转账
            transferFrom(Bills[i].from,Bills[i].to,Bills[i].value);
        }
    }
}
function transferFrom(address from, address to, uint amount) private returns (bool success) {
        move(from, to, amount);
        emit Transfer(from, to, amount);
        return true;
    }
```

三、 功能测试

实验截图

1) 部署：

```
fisco-bcos@fiscobcos-VirtualBox:~/asset-app$ ./gradlew build

BUILD SUCCESSFUL in 2s
4 actionable tasks: 4 up-to-date
fisco-bcos@fiscobcos-VirtualBox:~/asset-app$ cd dist/
fisco-bcos@fiscobcos-VirtualBox:~/asset-app/dist$ bash bill_run.sh deploy
 deploy Bill success, contract address is 0xf5461613c2f7f4e0f7131ff9074f4e5f997c30da
```

获取银行 address:

```
fisco-bcos@fiscobcos-VirtualBox:~/asset-app/dist$ bash bill_run.sh bank
bank, address 0xea08bcbc6ce11de1148f36af8fff9e14bfae26b9
```

获取 accounts[1]和 accounts[2]

```
fisco-bcos@fiscobcos-VirtualBox:~/asset-app/dist$ bash bill_run.sh showAccount 1
show account 1, address 0xca35b7d915458ef540ade6068dfe2f44e8fa733c
fisco-bcos@fiscobcos-VirtualBox:~/asset-app/dist$ bash bill_run.sh showAccount 2
show account 2, address 0x14723a09acff6d2a60dcdf7aa4aff308fddc160c
```

获取账户余额:

```
fisco-bcos@fiscobcos-VirtualBox:~/asset-app/dist$ bash bill_run.sh getBalanceOf 0xca35b7d915458ef540ade6068dfe2f44e8fa733c
account 0xca35b7d915458ef540ade6068dfe2f44e8fa733c, balance 50000
```

购买:

```
fisco-bcos@fiscobcos-VirtualBox:~/asset-app/dist$ bash bill_run.sh purchaseAndMakeBill 0xca35b7d915458ef540ade6068dfe2f44e8fa733c 10000 60
purchase success
```

转移账单：

```
fisco-bcos@fiscobcos-VirtualBox:~/asset-app/dist$ bash bill_run.sh transferBill 0xca35b7d915458ef540ade6068dfe2f44e8fa733c 0x14723a09acff6d2a60dcdf7aa4aff308fddc160c 5000
transfer success
```

融资：

```
fisco-bcos@fiscobcos-VirtualBox:~/asset-app/dist$ bash bill_run.sh applyForFunde 4000 50
apply funde success
```

自动支付：

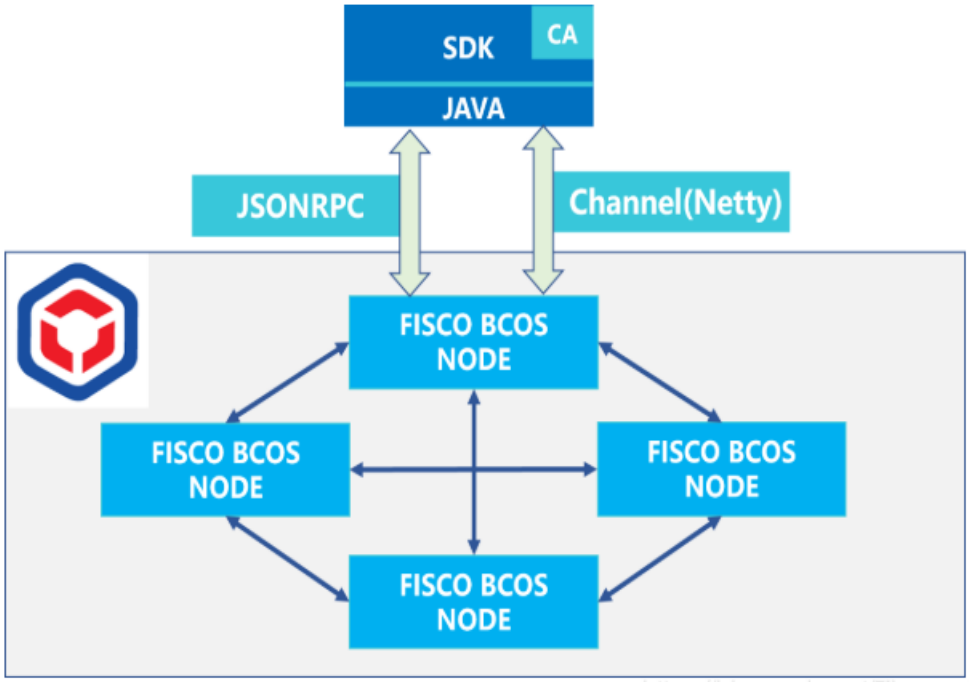
```
fisco-bcos@fiscobcos-VirtualBox:~/asset-app/dist$ bash bill_run.sh autoPayBill
autoPay success
```

界面展示

```
fisco-bcos@fiscobcos-VirtualBox:~/asset-app/dist$  bash bill_run.sh getBalanceOf 0xca35b7d915458ef540ade6068dfe2f44e8fa733c
account 0xca35b7d915458ef540ade6068dfe2f44e8fa733c, balance 50000
fisco-bcos@fiscobcos-VirtualBox:~/asset-app/dist$ bash bill_run.sh
 Usage :
   bash bill_run.sh deploy
   bash bill_run.sh init
   bash bill_run.sh getBalanceOf account
   bash bill_run.sh showAccount i
   bash bill_run.sh bank
   bash bill_run.sh getBillNum
   bash bill_run.sh autoPayBill
   bash bill_run.sh purchaseAndMakeBill sellerAddr value lastingTime
   bash bill_run.sh transferBill rootCompany seller value
   bash bill_run.sh applyForFunde value lastingTime


examples :
   bash bill_run.sh deploy
   bash bill_run.sh init
   bash bill_run.sh getBalanceOf 0xca35b7d915458ef540ade6068dfe2f44e8fa733c
   bash bill_run.sh showAccount 1
   bash bill_run.sh bank
   bash bill_run.sh getBillNum
   bash bill_run.sh autoPayBill
   bash bill_run.sh purchaseAndMakeBill 0xca35b7d915458ef540ade6068dfe2f44e8fa733c 10000 60
   bash bill_run.sh transferBill 0xca35b7d915458ef540ade6068dfe2f44e8fa733c 0x14723a09acff6d2a60dcdf7aa4aff308fddc160c 5000
   bash bill_run.sh applyForFunde 4000 50
fisco-bcos@fiscobcos-VirtualBox:~/asset-app/dist$
```

方案使用 javasdk 和 web3sdk 来实现服务端和链端的交互，从而构建区块链应用



接口设计：

```
public class Bill extends Contract {
        public RemoteCall<String> showAccount(BigInteger i);
        public RemoteCall<String> bank();
        public RemoteCall<Tuple5<BigInteger, BigInteger, String, String, Boolean>> Bills(BigInteger param0);
        public RemoteCall<TransactionReceipt> init();
        public RemoteCall<BigInteger> getBillNum();
        public RemoteCall<BigInteger> getBalanceOf(String tokenOwner);

        public RemoteCall<TransactionReceipt> applyForFunde(BigInteger _value, BigInteger _lastTime);
        public RemoteCall<TransactionReceipt> purchaseAndMakeBill(String _seller, BigInteger _value, BigInteger _lastTime);
        public RemoteCall<TransactionReceipt> transferBill(String _rootCompany, String _receiver, BigInteger _value);
        public RemoteCall<TransactionReceipt> autoPayForBill();

        public static Bill load(String contractAddress, Web3j web3j, Credentials credentials, ContractGasProvider contractGasProvider);
        public static RemoteCall<Bill> deploy(Web3j web3j, Credentials credentials, ContractGasProvider contractGasProvider);
}
```

四、 心得体会

本次实验对我来说有点艰难，前期 solidity 进行合约的编写在语言学习上刚开始有些不适应，在学习了相关的一些示例后有了些掌握，后面写起来比较轻松。刚开始使用的 Bill 是保存在公司内部信息中的，后面觉得每笔交易要保存两次，比较麻烦没有再这样做，改为了在合约中存储。

后面写接口和前段时，原来是打算用 html 和 node.js 来写的，html 比较简单，但是 node.js 学的不太熟就没有用，最后采用了技术文档中的 javasdk 来创建应用，调用功能时和在 fisco-bcos 中调用合约的情况比较符合，前端界面比较简陋，主要是时间不太够了，之后会考虑用 java 库来编写一个界面。

总之，这次通过实验，更进一步了解了 fisco-bcos 的优异环境和使用方式，对构建区块链应用有了一定的收获。