

# Lab4 Project Report

## Distributed Blockchain System

### Introduction

A simplified bitcoin blockchain project. Implemented several core functions of blockchain. The overall structure, mine a new block, consensus, bitcoin transaction(simplified), cryptography, verification and signature based on security library, decentralized p2p network, p2p websocket communication.

### Design RoadMap

1. Blockchain design
  - a. Mining algorithm
  - b. Transaction
  - c. Wallet
2. LabCoin design
  - a. Consensus
  - b. Blockchain cryptography
3. Transport design
  - a. HTTP server with api
  - b. P2P communication

### 1. Blockchain design

Blockchain design

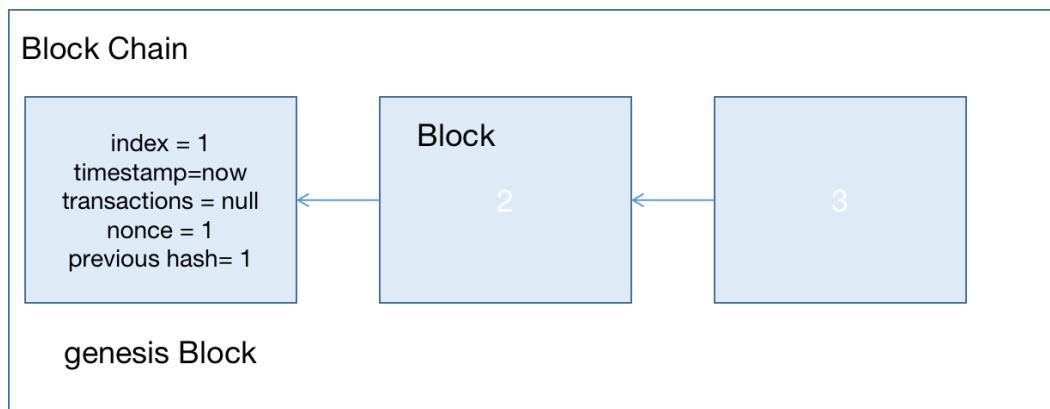


Fig 1. The blockchain diagram

## 1.1 Block Structure

```
/*
 *      //*****
 *      // ****|           index           |*****
 *      // ****|           hash            |*****
 *      // ****|           previousHash   |*****
 *      // ****|           timestamp      |*****
 *      // ****|           nonce          |*****
 *      // ****|           transactions   |*****
 *      // ****|           Transaction1->|*****
 *      // ****|           ->Transaction2->|*****
 *      // ****|           ->Transaction3 |*****
 *      // ****|
 */
public class Block {
    private final static int DIFFICULTY = 3;
    private int index;
    private String hash;
    private String previousHash;
    private long timestamp;
    private List<Transaction> transactions;
    private int nonce;
```

## 1.2 The genesis block

index starts from 1, timestamp is now(when it is created), transactions are null, nounce is 1, hash and previous hash are both 1.

```
Block genesisBlock = new Block(  
    1, System.currentTimeMillis(), new ArrayList<Transaction>(),  
    1, "1", "1");
```

### 1.3 Proof of work and the calculation of Block hash

Use SHA256 Hash in `java.security` packet, content to hash is:

```
Hash = SHA256(lastblockhash, transactions, nounce)
```

Calculate hash until the hash meets the DIFFICULTY requirements(starts with DIFFICULTY numbers of '0');

000            244356330        .....

difficulty

Fig 2 Difficulty illustration

Default difficulty is set to 3, can be edited in block-*DIFFICULTY*  
In this design, the block hash is calculated by

```
SHA256(index + transaction + previous hash + nonce)
```

Because a transaction is a list, we should parse it to a string. This project adopts json parse.

## 1.4 Transactions(Transaction, transactionInput, transactionOutput)

Transactions that occur on a blockchain must be validated by a number of the network's participants.

A transaction is only valid once the participants verify the transaction and a consensus is reached about its validity.

The transaction details are then recorded on the block and distributed to the network's participants.

The 'number' is encrypted by the owner's hash. New UTXO is created by mining or transaction. UTXO was stored in the entire blockchain. The sender will send from his own UTXO and encrypt it with the recipient's public key.

### 1.4.1 transactionInput and transactionOutput

When a block is mined, it will contain a base transaction(system reward), the base transaction contains only output and requires no signature.

When a peer wants to make a transaction, the transaction will find previous hash,

The sender can only send assets belonging to themselves(public key hash should match the sender's public key).

Transaction:

```
public class Transaction {  
    /**  
     * the globally unique ID, AKA UUID  
     */  
    private String id;  
    /**  
     * transaction input, indicate the input source of this transaction  
     */  
    private TransactionInput txIn;  
    /**  
     * transaction output, indicate the UTXO of this transaction  
     */  
    private TransactionOutput txOut;  
}
```

TransactionInput & TransactionOutput

```
public class TransactionInput {  
    /**  
     * the previous transaction's ID  
     */  
    private String txId;  
    /**  
     * the value of the input  
     */  
    private int value;  
    /**  
     * the signature of the sender  
     */  
    private String signature;  
    /**  
     * the public key of the receiver  
     */  
    private String publicKey;  
}
```

```
public class TransactionOutput {  
    /**  
     * the value of this UTXO  
     */  
    private int value;  
    /**  
     * the scripted public key of this receiver  
     */  
    private String publicKeyHash;  
}
```

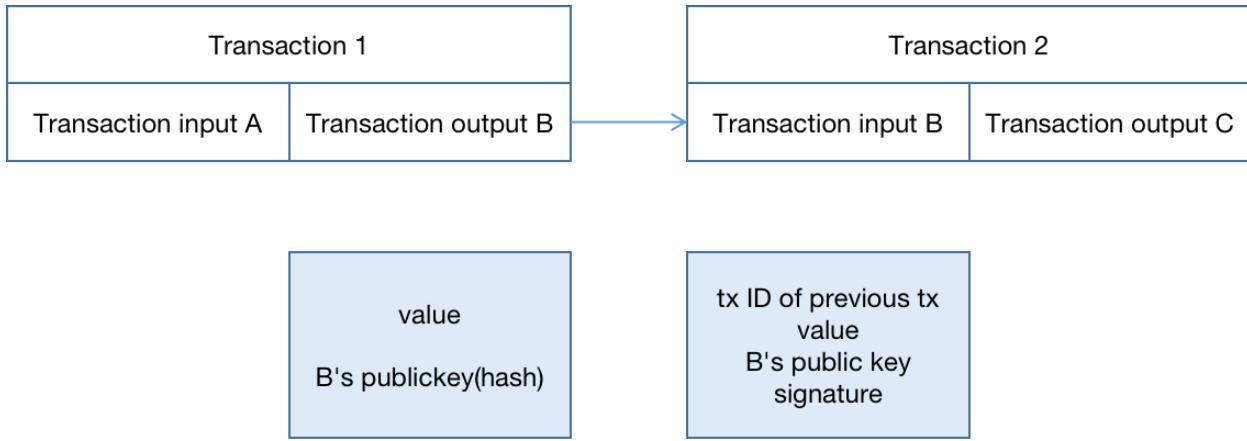


Fig 3. The structure of Transactions in a block

When a transaction is generated, it will be recorded locally as UTXO( unpacket transactions) until the next block is generated. During this process, the sender's UTXO will be consumed but the recipient hasn't received the amount of virtual asset.

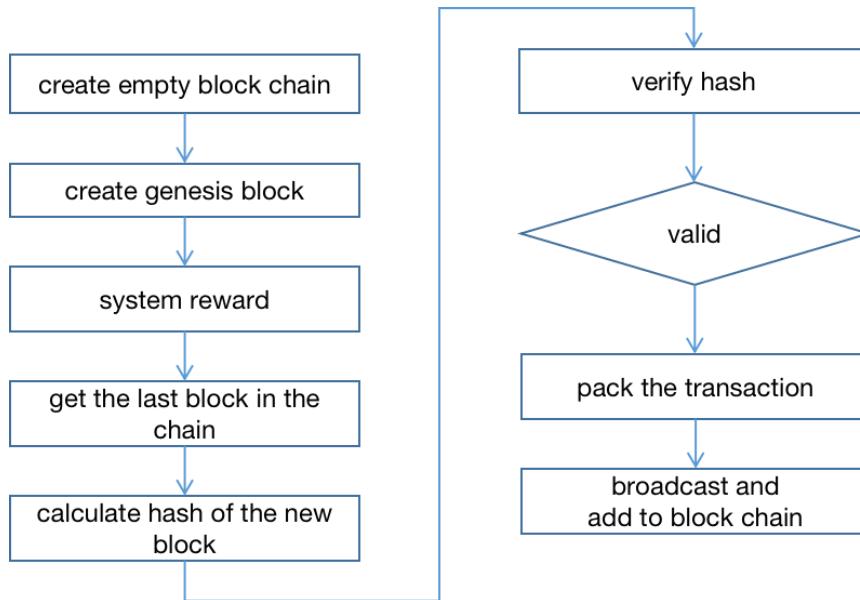


Fig 4. the work process

#### 1.4.2 Labcoin balance

The balance is calculated by UTXO.

When a user A wants to query for his balance, the system will iterate the UTXO list and find out all the outputs that \*belongs to A and calculates the sum of the virtual asset.

A UTXO belongs to A if the UTXO's public key (but in this design, publickeyhash) matches A's.

## 1.6 Labcoin wallet

Structure

```
public class Wallet {  
    private String publicKey;  
    private String privateKey;  
}
```

Wallet is a pair of ‘shorter name’ for public key.

Peers will have access to all the information of its own wallets and the public key information of others’ wallets.

The Wallet is anonymous, a user can see the details of wallets created by himself(public key, private key), but can only see the address and public key hash for others’ wallets.

When making a transaction, a user can use the address of the wallets as sender and recipient to process the transaction.

my wallet map

Address	Addr1	Addr2	Addr3	Addr4	...
Wallet	Wallet1 publickey1 privatekey1	Wallet2 publickey1 privatekey1	Wallet3 publickey1 privatekey1	Wallet4 publickey1 privatekey1	...

other's wallet map

Address	Addr1	Addr2	Addr3	Addr4	...
public key (hash)	publickeyhash1	publickeyhash1	publickeyhash1	publickeyhash1	...

## 2. Labcoin design

### 2.1 Consensus

If two peers mine a block successfully, keep the two blockchains locally until one is longer than the other one. The longer one will overwrite the other's local records until all achieve the same record.

When a peer wants to participate in the system, it will need to download all the information.  
When a peer has been disconnected for some time, it will need to update its local cache.

Compare the global blockchain and the local blockchain.

If the last block of the received blockchain has a higher index than the local blockchain:

- If the last one is exactly what the peer misses, directly add the block to local blockchain
- If the peer has missed more than one block, it should broadcast and ask all peers for the longest chain. Then pull down the longest chain.

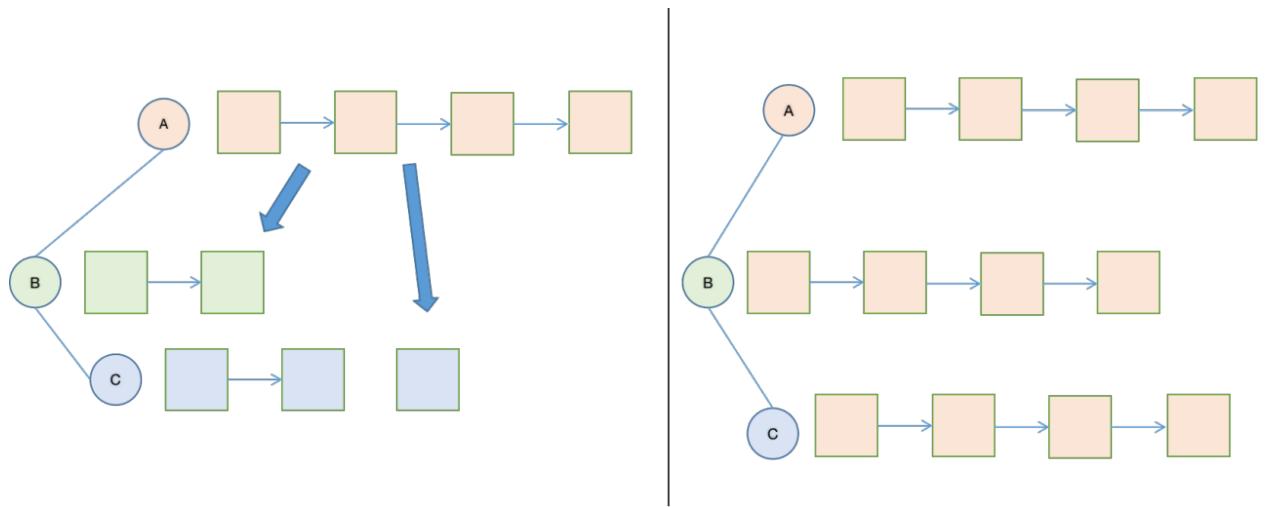


Fig 5. Consensus of the chain

### 2.2 Cryptography

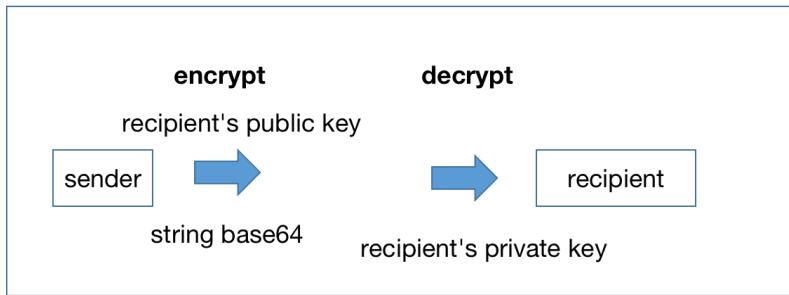
Asymmetric cryptography

Use RSA algorithm to generate key pair

Use MD5 to generate signature

Both in `java.security.*`

**Asymmetric cryptography: RSA**



**Sign: MD5**

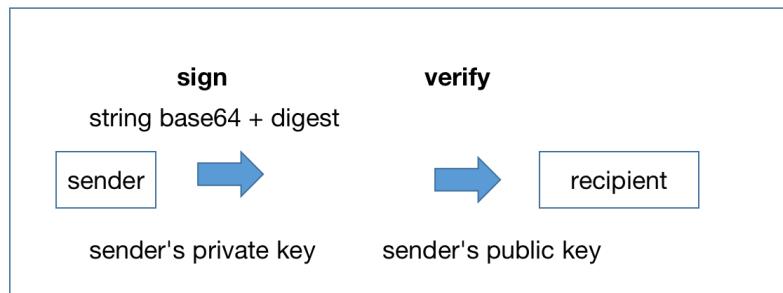


Fig 5. The encryption and sign

### 3. Transportation Layer:

#### 3.1 HTTP Server by jetty

1. client->server, request for data
  2. Server receives request, call server service, return data response to client
- Client receives the response, calls client service and deals with data.

HTTPSever api:

api	method	function	param	Broadcast or not
/chain	GET	get blockchain	/	no
/wallet/create	POST	create wallet	/	yes
/wallet/get	GET	Get Wallet get others wallet	address=<wallet address>	no

/wallet/get/othe r	GET	Get others' wallets	/	np
/wallet/get/bala nce	GET	Get balance of a specific wallet	address=<wallet address>	no
/mine	POST	Mine a block	address=<wallet address>	no
/transactions/ne w	POST	Propose a new transaction from sender to receiver	sender=<sender's wallet address> recipient=<recipient's wallet address> amount=<value>	Yes But needs to wait for another mine and get consensus of all others.
/transactions/ge t/packed	GET	Get packed transactions in this node	/	no
/transactions/ge t/unpacked	GET	Get unpacked transactions in this node	/	no
/transactions/ge t/all	GET	Get all transactions in this node	/	no
/peers	GET	Get peers connect to this node	/	no

### 3.2 P2P : P2P decentralized network

P2P means each node can be a client or server at the same time(but at different ports). In the project, the P2P design is adapted from easy chatroom with websocket.

P2P client will be able to call client services. P2P server will be a basic server handling messages from clients by calling functions in labcoin implementation.

There are some operations that will require broadcast.

1. When A wants to start a transaction, it should tell all the peers or others who will not be able to know about the transaction.
2. When A mined out a new block(faster than others), it should broadcast the block to all others. Other peers will decide whether to upload their own chain.
3. When A generates a new wallet, it should broadcast its address and public key to all peers.

3.

service	message	function
Get chain/ Blockchain broadcast	The json string of peer's chains	If the received block chain is newer, addBlock() if received a new last block. This is the normal add function  or replaceChain() if more than 1 is added or the previous block does not match. This will happen when a peer is disconnected for a while and reconnects to the p2p network.  Broadcast and ask all nodes for the chain if the received blockchain is genesis block. This will happen when a new peer joins the p2p network.
Deal with wallet request	The json string of others newly create wallet address and public key	Add to others' wallet map pool
Get all local wallets	The json string of get wallet query	Iterate my wallets and others' wallets, return a message containing all wallet lists
Generate all required queries as client	A json string carrying message type and message	Generate different service queries for other peers as a p2p client.
Peer communication	A json string carrying message type and message	websocket.send()

## 4. Project build:

### 4.1 how to build the project

mvn clean install

## 4.2 how to start a node

The peer node can be started with 1 or 2 or 3 arguments. The first argument is the port to start HTTP service, the second argument is the port to start p2p service, the third argument is the peer it wants to connect to.

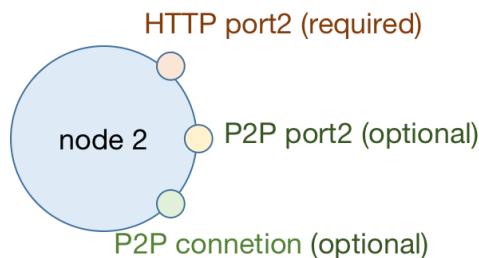


Fig 6. The three arguments of a peer node

The port can be freely chosen as long as it is unique. The peer websocket address is `ws://localhost:p2p_port_to_connect`

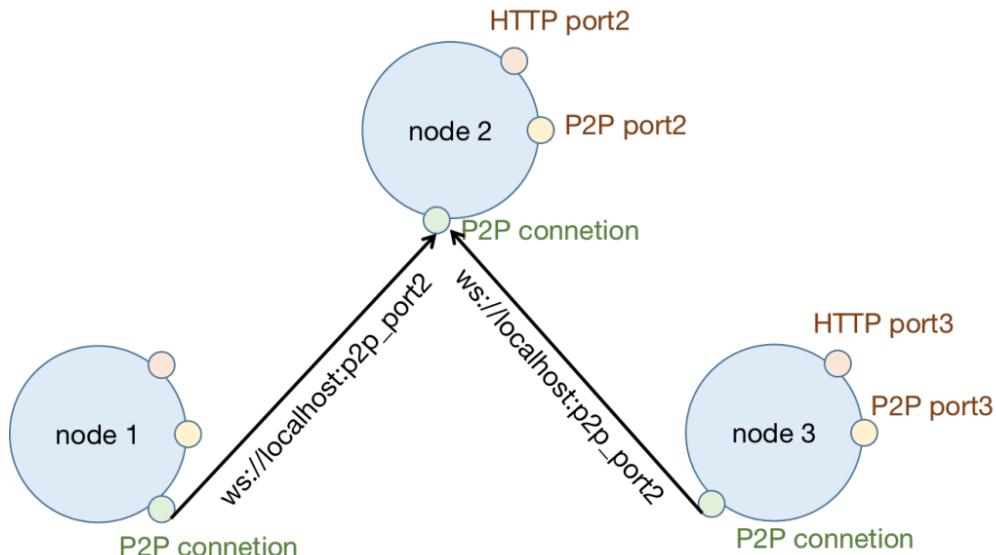


Fig 7. The illustration of connecting to p2p network

```
mvn clean install
```

```
cd target
```

**Node A:**

```
java -jar labBlockChain-1.0-SNAPSHOT.jar 1234 5678
```

**Node B (connect to A):**

```
java -jar labBlockChain-1.0-SNAPSHOT.jar 2345 6789
ws://localhost:5678
```

**Node C(connect to A) :**

```
java -jar labBlockChain-1.0-SNAPSHOT.jar 3456 7890
ws://localhost:5678
```

## 4.3 project api in terminal

### 4.3.1 Current BlockChain(GET)

```
curl http://localhost:1234/chain
```

### 4.3.2 Create Wallet(POST)

```
curl -i -X POST http://localhost:1234/wallet/create
```

### 4.3.3 Mine(POST)

```
curl --data "address=78415FA4D1BCFFF8AFBE5DD5C565C144"
http://localhost:1234/mine
```

### 4.3.4 Transaction(POST)

```
curl -H "Content-type:application/json" --data '{"sender": "9555D32727BD391C0A1B3FBE6C53FDBE", "recipient": "F784B5402BBCE003875E21C77550590A", "amount": 1}'
http://localhost:1234/transactions/new
```

### 4.3.5 Query for UXTOs(GET)

```
curl http://localhost:1234/transactions/get/unpacked
```

### 4.3.6 Query for all Transactions(GET)

```
curl http://localhost:1234/transactions/get/all
```

#### 4.3.7 Query for packed Transactions(GET)

```
curl http://localhost:1234/transactions/get/packed
```

#### 4.3.8 All wallets in this node(GET)

```
curl http://localhost:1234/wallet/get
```

#### 4.3.9 Other's Wallet Balance(GET)

```
curl "http://localhost:1234/wallet/get/other"
```

#### 4.3.10 Wallet Balance(GET)

```
curl  
"http://localhost:1234/wallet/get/balance?address=9555D32727BD39  
1C0A1B3FBE6C53FDBE"
```

#### 4.3.11 see socket peers connected(GET)

```
curl http://localhost:1234/peers
```

## 5. Test

### 5.1 Local Maven Test:

We implemented three local test cases which test the blockchain mining and transaction functionalities on a single machine. They are:

### 5.1.1 local Block OperationTest:

this testcase tests for mining a local block. We checked if the number of zeroes of the block hash matches the difficulty level. We also checked if the previous hash is indeed the previous block's hash.

### 5.1.2 stressBlockOperationTest:

this testcase is an extension of the previous one. It mines 100 blocks consecutively, and checks the validity of the appended block in every single iteration.

### 5.1.3 transaction API Test:

this testcase tests the functionality of making a transaction between wallets.

In our case, we define a dummy p2p network with only one client and one server. The client has two wallets, sender and receiver.

We will first mine 1 block in the sender wallet, and then send 1 LabCoin from sender to the receiver. When we get back the HTTP response, we will check if the transaction succeeds by asserting that the transaction is not a base transaction and the transaction amount is as expected.

## 5. 2 Demo Test:

1. Start A, A will create a genesis block
2. A create wallet A1, A2, both of these should have balance 0, should raise broadcast.
3. A1 mines a block, and should get a reward from the system, should raise broadcast.
4. Now start another peer B, B will create a genesis block
5. Create wallet B1, the balance should be 0, should raise broadcast.
6. A intend to make a transaction from A1 to B1, should raise broadcast

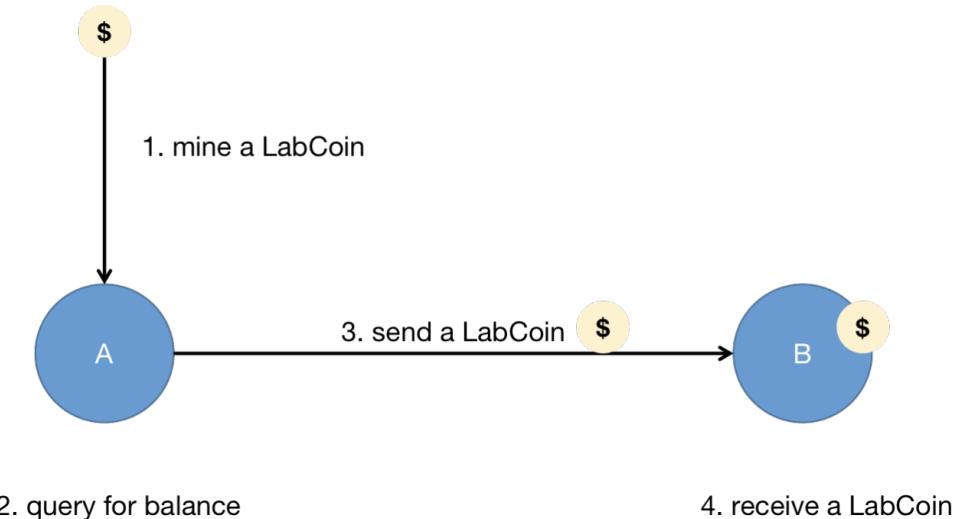


Fig 8. The brief process of demo

## 6. Points

### 6.1 Points Breakdown on Coding Tasks:

Local block mining/POW functionalities	20	finished
Broadcast, read, write, validation on blockchain	40	finished
Schema that handles concurrency issues on operations and helps reach the consensus on decisions	40	Finished and can reach consensus.  Limited concurrency situation with synchronization.
Transportation Layer	20	Finished by HTTP server
Transaction verification and real p2p communication scheme	Bonus: 20 (each 10, if bonus is allowed)	Finished simplified transaction(send 1 coin each time).  Finished by implementing

		real p2p
--	--	----------

## 6.2 Points Breakdown on Test cases:

Local test cases in maven Usage: mvn test (The test cases are developed when we are implementing blockchain operations on the local machine)	30 LocalOperationTest - 10 StressOperationTest - 10 TransactionAPITest - 10	Passed when run mvn clean install
Remote APIs Usage: demo on postman (All APIs except the consistency check)	50 (5 for each, 10 in total)	Need to be demoed
Consistency check Usage: demo on postman (after mining and transactions, do <code>curl http://localhost:port/check</code> for every single client)	10	

## 7. Conclusion

In this project, we designed and implemented a distributed blockchain system with a decentralized p2p network. Every user in our system, as a p2p client, is able to create a wallet, mine a block, make a transaction to another wallet, and connect to peers.

Our development process follows the test-driven development style. We realized that distributed systems are particularly hard to test on maven, so we first generate test cases that test local functionalities and modules. We also added a stress test on blockchain validity to make the local module robust enough. Then we begin developing the distributed part using Jetty and Java

Spring 2021, Lab 4 proposal  
14736 Di Lu(dilu), Yuan Gu(guy)

Servlet to develop APIs for distributed operations. We first wrote shell scripts to test distributed functionalities, but for the demo purposes, we choose postman due to its intuitive UI.

We have finished some challenging tasks like transactions. Our transaction is simplified and can only support 1 coin transaction each time. We can implement more complex transactions by modifying the structure of the transaction class. Change inputs and outputs to lists, change the current signature method and we can implement coin aggregate and exchange. Further, the list of transactions can be manipulated and structured by merkle tree, but we did not have time to try it.

The cryptography we use is RSA and MD5, the real cryptography is ECDSA. But as asymmetric cryptography, RSA is enough for our lab.

We simplified the consensus of the distributed peers in order to spare time for p2p design. We only considered the synchronized keyword. The p2p design is based on chatroom, with server and client dealing with structured messages.

## 8. APPENDIX: Screenshots of test and demo

### 8.1 Local test:

```
Terminal: step2_startpeer2.sh ✘ step1_startpeer1 ✘ step3_startpeer3.sh ✘ +  
fire-reports  
-----  
TESTS  
-----  
Running LabBlockChain.Block.TransactionTest  
Genesis Block: {"hash":"1","index":1,"nonce":1,"previousHash":"1","timestamp":162087177976,"transactions":[]}  
listening websocket LabBlockChain.basic.BlockChain.p2p port on: 5678  
2021-05-12 09:46:18.152:INFO:main: Logging initialized @572ms to org.eclipse.jetty.util.log.StderrLog  
listening LabBlockChain.http port on: 1234  
2021-05-12 09:46:18.254:INFO:oejs.Server:main: jetty-9.4.8.v20171121, build timestamp: 2017-11-21T16:27:37-05:  
00, git hash: 82b8fb23f757335bb3329d540ce37a2a2615f0a8  
2021-05-12 09:46:18.294:INFO:oejs.session:main: DefaultSessionIdManager workerName=node0  
2021-05-12 09:46:18.295:INFO:oejs.session:main: No SessionScavenger set, using defaults  
2021-05-12 09:46:18.297:INFO:oejs.session:main: Scavenging every 600000ms  
2021-05-12 09:46:18.303:INFO:oejsh.ContextHandler:main: Started o.e.j.s.ServletContextHandler@2a265ea9{/null,  
AVAILABLE}  
2021-05-12 09:46:18.310:INFO:oejs.AbstractConnector:main: Started ServerConnector@663c9e7a{HTTP/1.1,[http/1.1]  
}{0.0.0.0:1234}  
2021-05-12 09:46:18.311:INFO:oejs.Server:main: Started @731ms  
{AE7821144E167C3AAAAA18E17E411340=Wallet{publicKey='MIGfMA0GCSqGSIB3DQEBAQUAA4GNADCBiQKBgQCLiXmhBdFoZvYxuE/xZ/  
3Moxtt09iGb/nlcqpBPuyLIsVfv89z19xfdhFvCj+WdvKvH9LHD1AMg1jDyjFBpKStLvrpGJlGRYnoASp1B1Exu1ahmNIA9vLBJingWdYdPiiq  
Sx0pzBH3m0eFLioAwqqW3Mq2qnct6VLcd02gvFPQIDAQAB', privateKey='MIICdgIBADANBgkqhkiG9w0BAQEFAASCamAwggJcAgEAAoG  
BAIuJeaEF0Whm9jG4T/Fn/cyjG2072IZv+eVykE+7KUixV+/z3PX3F92EW8KP5Z28q8f2UcPUAyDWMPKMUGkpK0u+uKymUZFiegBKnuHURe7V  
qGY0gD2+UEmKeBZ1h0+KKpLHSnMEDjeY54UuKgDCqpbccyraqdy3pUtzN07aC8U9AgMBAAECgYB/pvXpdHwdqcvIsn4YGA2cjz8BEHXGj1wrJ1p  
QNDOj4wpzzPMJRForzsSmM9b2XL0prtZEzuLrf7vN6gRjzC6FkrSFegVoNXB4QnquGkAONjUAcfc8CmZqj5IVMNBJ0aQ02cwX+E/ALNwUpK93J+  
oTxv8NybvUviThFAIBsFu7wJ4QJBAMZf0QuS6fpTYDF0paJ/jWsNk/154nHk8xpBeicUtmwGa0jHJL1o0P4Ks10BnPvDS2n0HAByhOCxYzWk1M  
adxUCQQC0EjN/4o0UxLJdticSnRc7e/6JCuChYR4XbtN1oDSjWcS61tVVxw8bFQ08KvlfL/2Jz0k+/LzLry0NC4/oPJ+JAkA9Ls41yKEbmNyfi  
7UP0w5J1YzGyqwX5ZMjJ30oChbPhy3UJZEpjlnFomPn8D4RQbiJzRpuCU8/B0VQs9mKdw4LakASTJc3M8mzYJnPtCccWSJ3RQ6E1rolLD1mnDi  
ZB4LR5t17M0HSybvbQLN0d9mHttTJIC3D30NCv7aQUNB7BU4hAkEApGv1M2YBoGYU7SehLhkvgUhbCwSJxRAhwNcZLPgyXIrLo4fTC4Hk3yL2v  
V5wv7YkPcMQts4749CtSvdvmYNqjg=='}  
generateBaseTx txIn: {"intxId":"0","invalue": -1}  
generateBaseTx Wallet: {AE7821144E167C3AAAAA18E17E411340=Wallet{publicKey='MIGfMA0GCSqGSIB3DQEBAQUAA4GNADCBiQKB  
gQCLiXmhBdFoZvYxuE/xZ/3Moxtt09iGb/nlcqpBPuyLIsVfv89z19xfdhFvCj+WdvKvH9LHD1AMg1jDyjFBpKStLvrpGJlGRYnoASp1B1Exu1  
ahmNIA9vLBJingWdYdPiiqSx0pzBH3m0eFLioAwqqW3Mq2qnct6VLcd02gvFPQIDAQAB', privateKey='MIICdgIBADANBgkqhkiG9w0BA  
QEFAASCamAwggJcAgEAAoGBAIuJeaEF0Whm9jG4T/Fn/cyjG2072IZv+eVykE+7KUixV+/z3PX3F92EW8KP5Z28q8f2UcPUAyDWMPKMUGkpK0  
u+uKymUZFiegBKnuHURe7VqGY0gD2+UEmKeBZ1h0+KKpLHSnMEDjeY54UuKgDCqpbccyraqdy3pUtzN07aC8U9AgMBAAECgYB/pvXpdHwdqcv  
isn4YGA2cjz8BEHXGj1wrJ1pQNDOj4wpzzPMJRForzsSmM9b2XL0prtZEzuLrf7vN6gRjzC6FkrSFegVoNXB4QnquGkAONjUAcfc8CmZqj5IV  
MNBJ0aQ02cwX+E/ALNwUpK93J+oTxv8NybvUviThFAIBsFu7wJ4QJBAMZf0QuS6fpTYDF0paJ/jWsNk/154nHk8xpBeicUtmwGa0jHJL1o0P4  
Ks10BnPvDS2n0HAByhOCxYzWk1MadxUCQQC0EjN/4o0UxLJdticSnRc7e/6JCuChYR4XbtN1oDSjWcS61tVVxw8bFQ08KvlfL/2Jz0k+/Lz  
Lry0NC4/oPJ+JAkA9Ls41yKEbmNyfi7UP0w5J1YzGyqwX5ZMjJ30oChbPhy3UJZEpjlnFomPn8D4RQbiJzRpuCU8/B0VQs9mKdw4LakAST  
Jc3M8mzYJnPtCccWSJ3RQ6E1rolLD1mnDiZB4LR5t17M0HSybvbQLN0d9mHttTJIC3D30NCv7aQUNB7BU4hAkEApGv1M2YBoGYU7SehLhk  
vgUhbCwSJxRAhwNcZLPgyXIrLo4fTC4Hk3yL2vV5wv7YkPcMQts4749CtSvdvmYNqjg=='}  
generateBaseTx txOut: {"outvalue": 1, "outpublickeyhash": "15c156dbb90c020092955ae48cb01b123bb8d899fad4a945857704  
8a37d719b"}
```

Spring 2021, Lab 4 proposal  
14736 Di Lu(dilu), Yuan Gu(guy)

```
Terminal: step2_startpeer2.sh × step1_startpeer1 × step3_startpeer3.sh × +
```

```
ZlPgyXIrLo4fTC4Hk3yL2vV5wv7YkPcMQtts4749CtSvdvmYNqjg=='}}
```

```
generateBaseTx txOut: {"outvalue":1,"outpublicKeyHash":"15c156dbb90c020092955ae48cb01b123bb8d899fad4a9458577048a37d719b"}
```

```
Start Mining>>>>
```

```
>>>>>>>>>>>>>>>>>>>>
```

```
Mine Complete, correct calculateHash is: 000fb08bf4776871014aa832253797d2cca844a706a4c9e8843e9c88e8660f
```

```
time comsumption108ms
```

```
New Block: {"index":2,"hash":"000fb08bf4776871014aa832253797d2cca844a706a4c9e8843e9c88e8660f","previousHash": "1","timestamp":1620827178743,"transactions":[{"id": "7ddff71bca744002a55eb321c08f6f1b","txIn": {"txId": "0","value": -1}, "txOut": {"value": 1,"publicKeyHash": "15c156dbb90c020092955ae48cb01b123bb8d899fad4a9458577048a37d719b"}, "inputs": [], "outputs": []}],"nonce":4551}
```

```
{AE782114E167C3AAAAA18E17E411340=Wallet{publicKey='MIGfMA0GCSqGSIB3DQEBAQUAA4GNADCBiQKBgQCLiXmhBdFoZvYxuE/xZ/3Moxtt09iGb/nlcqpBPuyLIsVfv89z19xfdhFvCj+WdvKvH91HD1AMg1jDyjFBpKStLrvpGJLGRYnoASp1B1Exu1ahmNIA9vLBJingWdYdPiiqSx0pzBHY3m0eFl0AwqqW3Mq2qnct6VLczd02gvFPQIDAQAB', privateKey='MIICdgIBADANBgkqhkiG9w0BAQEFAASCamAwggJcAgEEAoGBAIuJeaEF0Whm9jG4T/Fn/cyjG2072IZv+eVYqkE+7KUixV+/z3Px3F92EW8KP5Z28qf2UcPUAyDWMPKMUcKpK0u+uKymUZFiegBKnuHURe7VqGY0gD2+UEmKeBZ1h0+KKpLHSnMEDjeY54UuKgDCqpbcyraqdy3pUtzN07aC8U9AgMBAEcgYB/pvXpdHwdqcvishn4YGA2cjz8BEHXGj1wrJ1pQNDoj4wpzzPMJRForzsSm9b2XL0prtZEZuLRF7vN6gRjzC6FkrSFegVoNXB4QnqoGkAOJnUAcfc8cmZqj5IVMNBJ0aQ02cwX+E/ALNwUpK93J+oTxV8NybvUiThFAIBsFu7wJ4QJBAMZf0QuS6fpTyDF0paJ/jWsNk/154nHk8pxBeicUtmwGa0jHJL1oP4Ks10BnPvDS2nOHAByH0CxYzWk1MadxUCQQCOEjn/4o0UxLJdticSnRc7e/6JCuChyR4XbtN1oDSjWcS61tVvxw8bfQ08KvlfL/2Jz0k+/Lzlr0NC4/oPJ+JAKa9Ls41yKEomNyfi7UP0w5J1YzGyqwX5ZMjJ30oChbPhy3UJZEpjlNFomPn8D4RQbijzRpuCU8/B0Vqs9mKdw41AkASTJc3M8mzYJnPtCccWSJ3RQ6E1rollD1mnDiZB4LR5t17M0HsybvbQLN0d9mHttTJIC3D30NCv7aQUNB7Bu4hAkEApGv1M2YBoGYU7SehLhvGUhbCwSJxRAhwNcZLPgyXIrLo4fTC4Hk3yL2vV5wv7YkPcMQtts4749CtSvdvmYNqjg=='}, 11D7D74B3319886D0FB9B3737B304537=Wallet{publicKey='MIGfMA0GCSqGSIB3DQEBAQUAA4GNADCBiQKBgQCH0ZTcgEb7YR9djzWqwcJQxX0MVV0HbJjtDLb9aEe07kHr9ewGzljIfURBF5ap2xzstVrzGSf2fVqGPFs160r40dU2Xs+NApSEL6Yd3J9CIgqPaUl561VoAhBNkcREeD4edYY4kCLaXXfNbFrz1oTFRzaKstMjvD2zkBPcc1QIDAQAB', privateKey='MIICeAIBADANBgkqhkiG9w0BAQEFAASCamIwgJeAgEAoGBAiFrlNyARvPthH12PNarBwldFFQxVU4ds0OK0Mtvi0r7TuQev17Ab0WMh9EFcXlqnbnH0xNWvMZJ/Z9wOY8WzXo6vg51TZe40CLIQvh3cn0IiCo9pSXnqVWgCEE2RxER4Ph51hjiQItppdd81sVHPWhMVHNqo2y0y08Pb0QE9wLVAgMBAEcgYACJRovvUrhkU+1V4ewcDUFPWz8VUcem7N0hP21KMf/gZ8GVUH6chmGKoj+kRJws3FYHibZRIIwUC/jBA/UAzfN/mZb/lakXtr0rMPWWdi7Pe0HWjfUUbmq/9zK8dmiXfUiH1JryaRfRJ9S0pXYA5peElxZWJJRChMHhG/AAeeBQJBA0c5VrLIMCDCKqmikaagymaWeyewc1nFvosK8TS6rj7dACH55Wc47+x2M5ojPlGieBGHM12R8V0CkDmQ0Px2e2QMCQQCWxZks0U5sfY8rZMmT0+1oyCowHZSoWqd00SRp8ppvQfQbGfKaL2+H0R+Z1yJzZcDAIcnw+ET7LFx/Olu+3/FHAKEAgu/4fwq6/bBmlgMyDdrVriFCyIJC/+X7Prs0ecfgysXcSv4jIKNrj+hIxG9iLA4otv+oHH2KIEZiYNERwtTwJBAIgVk1hFbFo/A0h0J/C1NT+0lmmA+734fL+9XkT7p0003m0Acgutjk3NtdmV1rm0hmHasLT/ef5r3+HMQKdXJ5UCQQCG23L8xLAT7JcvWK+Npe1AApMQckveww1qxYEi9SFm9VQnygvFjfmg1dN2CSmL/vLLbNR3tubcnszLMVPdTQd2'}
```

```
Transaction test complete!
```

```
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 1.576 sec
```

```
Running LabBlockChain.Block.BlockOperationTest
```

```
Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0 sec
```

```
Results :
```

```
Tests run: 3, Failures: 0, Errors: 0, Skipped: 0
```

```
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 1.576 sec
```

```
Running LabBlockChain.Block.BlockOperationTest
```

```
Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0 sec
```

```
Results :
```

```
Tests run: 3, Failures: 0, Errors: 0, Skipped: 0
```

## 8.2 remote test

### 8.2.1 Start node A, B

Start node A:

```
java -jar labBlockChain-1.0-SNAPSHOT.jar 1234 5678
```

Start node B: connect to A

```
java -jar labBlockChain-1.0-SNAPSHOT.jar 2345 6789 ws://localhost:5678
```

Let A get the current block chain: **GET http://localhost:1234/chain**

### 8.2.2 check genesis block

When A is created, the genesis block is created.

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'My Workspace' containing collections like 'labcoin\_peer 1234' and 'labcoin\_peer 2345 Copy 2'. The main area shows a 'GET' request to 'http://localhost:1234/chain'. The 'Body' tab displays the JSON response from the server:

```
1 Current BlockChain: [  
2   {  
3     "hash": "1",  
4     "index": 1,  
5     "nonce": 1,  
6     "previousHash": "1",  
7     "timestamp": 1620827423438,  
8     "transactions": []  
9   }  
10 ]
```

The status bar at the bottom indicates 'Status: 200 OK'.

### 8.2.3 check /peers

Let A GET peers: [Get http://localhost:1234/peers](http://localhost:1234/peers)

A is connected with B. The port is B's p2p working port (not the one we start p2p service, the listen port).

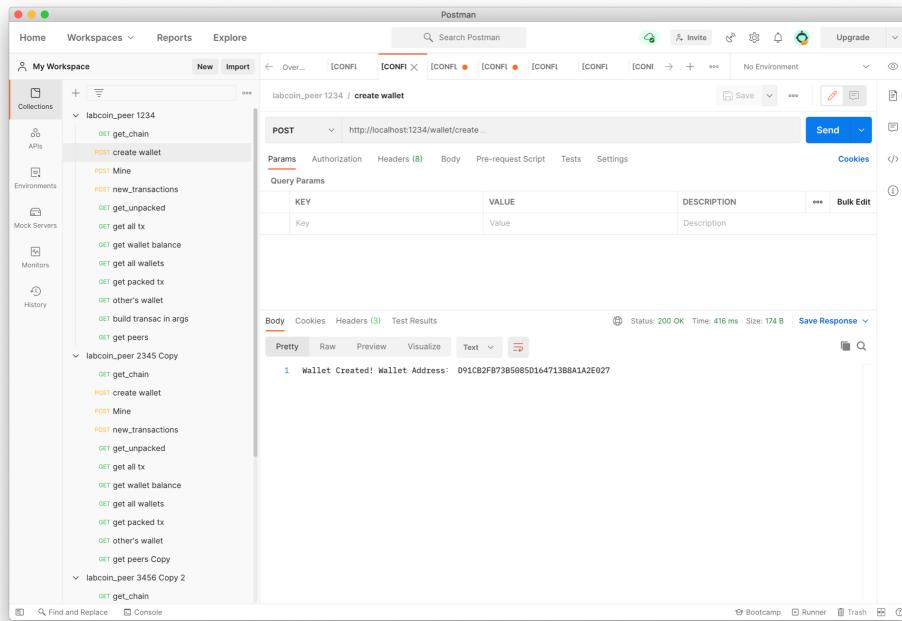
The screenshot shows the Postman application interface. On the left, there's a sidebar with 'My Workspace' containing several collections: 'labcoin\_peer 1234', 'labcoin\_peer 2345 Copy', and 'labcoin\_peer 3456 Copy 2'. The main area displays a 'labcoin\_peer 1234 / get peers' request. The 'Params' tab shows a single query parameter 'KEY' with value 'Value'. The 'Body' tab shows the response: 'Pretty' view displays '1. localhost:49719' and '2.'. The status bar at the bottom indicates 'Status: 200 OK'.

### 8.2.4 test wallet operation

Now let A create wallet A1

[POST http://localhost:1234/wallet/create](http://localhost:1234/wallet/create)

Spring 2021, Lab 4 proposal  
14736 Di Lu(dilu), Yuan Gu(guy)

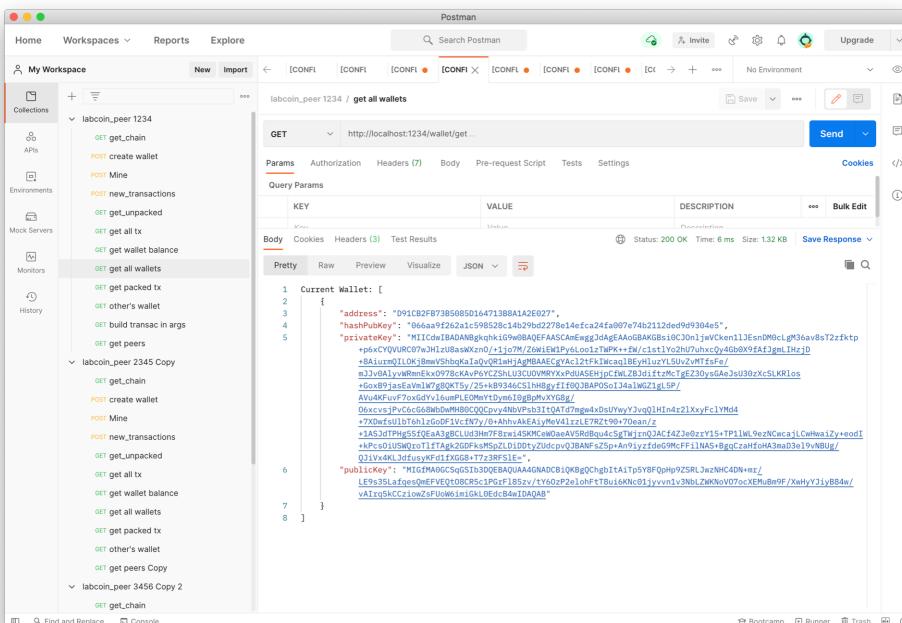


A wallet is created, address is D91CB2FB73B5085D164713B8A1A2E027

Now Get all wallets in this node:

GET http://localhost:1234/wallet/get

A can see the public key and private key of his wallet.



## 8.2.5 test mining and consistency

Let A Mine a block to the A1 wallet:

POST <http://localhost:1234/mine?address=D91CB2FB73B5085D164713B8A1A2E027>

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'My Workspace' containing collections like 'labcoin\_peer 1234' and 'labcoin\_peer 2345'. The main area shows a 'POST' request to 'http://localhost:1234/mine?address=D91CB2FB73B5085D164713B8A1A2E027'. The 'Params' tab has a single entry 'address' with value 'D91CB2FB73B5085D164713B8A1A2E027'. The 'Body' tab contains a JSON payload representing a new block:

```
1  New Block From Mining: {
2    "hash": "0803fb3301ef165d5193481fa0e6c73fb09e8cd11bbee91730413f6f97991ad0",
3    "index": 2,
4    "nonce": 5134,
5    "previousHash": "1",
6    "timestamp": 162082167102,
7    "transactions": [
8      {
9        "baseTx": true,
10       "id": "fbcc055511f142648169b26388f9da16",
11       "inputs": [],
12       "outputs": [],
13       "txIn": [
14         {
15           "txId": "0",
16           "value": -1
17         }
18       ],
19       "txOut": [
20         {
21           "publicKeyHash": "066aa9f262a1c598528c14b29bd2278e14efca24fa007e74b2112ded9d9384e5",
22           "value": 1
23         }
24     ]
25   }
```

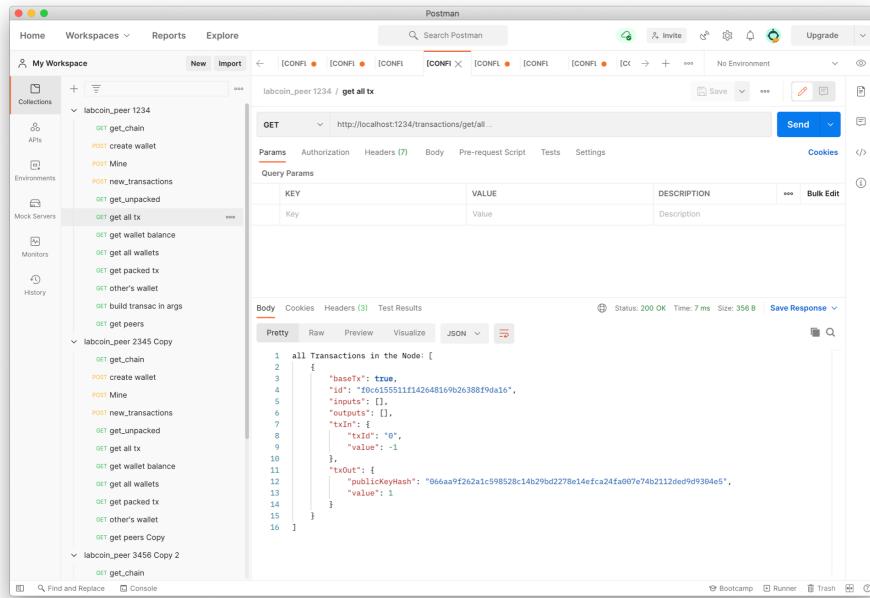
A block with base transaction (system reward) will be created.

Let A get the current all transactions:

GET <http://localhost:1234/transactions/get/all>

# Spring 2021, Lab 4 proposal

## 14736 Di Lu(dilu), Yuan Gu(guy)

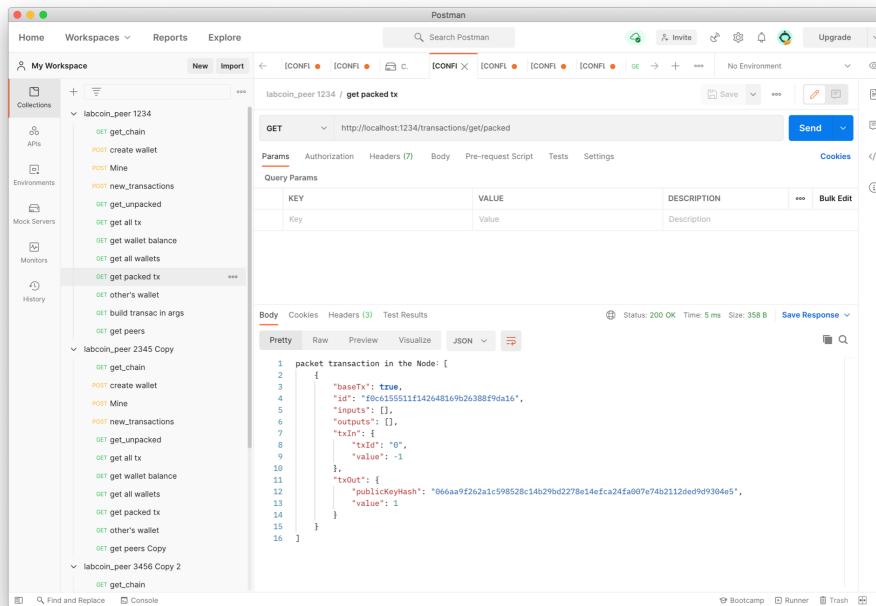


The screenshot shows the Postman application interface. On the left, there's a sidebar with 'My Workspace' containing collections like 'labcoin\_peer 1234' and 'labcoin\_peer 2345 Copy'. The main area shows a request for 'labcoin\_peer 1234 / get all tx'. The method is 'GET' and the URL is 'http://localhost:1234/transactions/get/all...'. The 'Body' tab shows a JSON response with a single transaction object:

```
1 all Transactions in the Node: [
2   {
3     "baseTx": true,
4     "id": "f0e015511ff142648169b26388f9da16",
5     "inputs": [],
6     "outputs": [],
7     "txIn": [
8       {
9         "txId": "0",
10        "value": -1
11      },
12      {
13        "publicKeyHash": "966aa9f262a1c598528c14b29bd2278e14efca24fa097e74b2112ded9d9304e5",
14        "value": 1
15      }
16    ]
}
```

Let A get the current packed transactions:

GET <http://localhost:1234/transactions/get/packed>



The screenshot shows the Postman application interface. The left sidebar is identical to the previous one. The main area shows a request for 'labcoin\_peer 1234 / get packed tx'. The method is 'GET' and the URL is 'http://localhost:1234/transactions/get/packed'. The 'Body' tab shows a JSON response with a single transaction object, identical to the one above:

```
1 packet transaction in the Node: [
2   {
3     "baseTx": true,
4     "id": "f0e015511ff142648169b26388f9da16",
5     "inputs": [],
6     "outputs": [],
7     "txIn": [
8       {
9         "txId": "0",
10        "value": -1
11      },
12      {
13        "publicKeyHash": "966aa9f262a1c598528c14b29bd2278e14efca24fa097e74b2112ded9d9304e5",
14        "value": 1
15      }
16    ]
}
```

They are the same, all transactions are packed. A have no UTXO now.

Current blockchain in A:

GET <http://localhost:1234/chain>

# Spring 2021, Lab 4 proposal

## 14736 Di Lu(dilu), Yuan Gu(guy)

```

1 Current BlockChain: [
2   {
3     "hash": "1",
4     "index": 1,
5     "nonce": 1,
6     "previousHash": "1",
7     "timestamp": 1620827423438,
8     "transactions": []
9   },
10  {
11    "hash": "0003fb381ef165d5193481fa0e6c73fb80e8cd11bbee91730413fe976991adb",
12    "index": 2,
13    "nonce": 5134,
14    "previousHash": "1",
15    "timestamp": 1620828167102,
16    "transactions": [
17      {
18        "baseTx": true,
19        "id": "fdcb15511f142648169b26388f9da16",
20        "inputs": [],
21        "outputs": [],
22        "txIn": [
23          {
24            "txId": "0",
25            "value": -1
26          }
27        ],
28        "txOut": [
29          {
30            "publicKeyHash": "066aa9f262a1c598528c14b29bd2278e14efca24fa007e74b2112ded9d9304e5",
31            "value": 1
32          }
33        ]
34      }
35    ]
36  }
37]

```

The blockchain is updated, the newly mined blockchain is successfully added.

```

1 Current BlockChain: [
2   {
3     "hash": "1",
4     "index": 1,
5     "nonce": 1,
6     "previousHash": "1",
7     "timestamp": 1620827556357,
8     "transactions": []
9   },
10  {
11    "hash": "0003fb381ef165d5193481fa0e6c73fb80e8cd11bbee91730413fe976991adb",
12    "index": 2,
13    "nonce": 5134,
14    "previousHash": "1",
15    "timestamp": 1620828167102,
16    "transactions": [
17      {
18        "baseTx": true,
19        "id": "fdcb15511f142648169b26388f9da16",
20        "inputs": [],
21        "outputs": [],
22        "txIn": [
23          {
24            "txId": "0",
25            "value": -1
26          }
27        ],
28        "txOut": [
29          {
30            "publicKeyHash": "066aa9f262a1c598528c14b29bd2278e14efca24fa007e74b2112ded9d9304e5",
31            "value": 1
32          }
33        ]
34      }
35    ]
36  }
37]

```

The previous hash matches the hash of the previous block(the genesis block). The new transaction ID is [f0c6155511f142648169b26388f9da16](#)

See A's wallet A1 balance now:

Spring 2021, Lab 4 proposal  
14736 Di Lu(dilu), Yuan Gu(guy)

GET

<http://localhost:1234/wallet/get/balance?address=D91CB2FB73B5085D164713B8A1A2E027>

Is 1 labcoin (from system reward).

The screenshot shows the Postman application interface. On the left, the 'Collections' sidebar is open, displaying three collections: 'labcoin\_peer 1234', 'labcoin\_peer 2345 Copy', and 'labcoin\_peer 3456 Copy 2'. The 'labcoin\_peer 1234' collection is expanded, showing various API endpoints such as 'get\_chain', 'create wallet', 'Mine', 'new\_transactions', 'get\_unpacked', 'get\_all\_tx', 'get\_wallet\_balance', 'get\_all\_wallets', 'get\_packed\_tx', 'get\_other's\_wallet', 'build\_transac\_in\_args', 'get\_peers', and 'get\_chain'. The 'get\_wallet\_balance' endpoint is selected. The main panel shows a 'GET' request to the URL <http://localhost:1234/wallet/get/balance?address=D91CB2FB73B5085D164713B8A1A2E027>. The 'Params' tab is active, with a single parameter 'address' set to 'D91CB2FB73B5085D164713B8A1A2E027'. The 'Body' tab shows the response body, which contains two lines of JSON: '1 wallet address is:D91CB2FB73B5085D164713B8A1A2E027' and '2 Balance in Wallet is:1 LabCoin'.

## 8.2.6 test transactions and consistency

Now let B create a wallet B1

# Spring 2021, Lab 4 proposal

## 14736 Di Lu(dilu), Yuan Gu(guy)

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'My Workspace' containing collections like 'labcoin\_peer 1234', 'labcoin\_peer 2345 Copy', and 'labcoin\_peer 3456 Copy 2'. The main area displays a POST request to 'http://localhost:2345/wallet/create'. The 'Params' tab shows a single parameter 'address' with value '533B8F11B2BE17C3B37FBFE32D6FCAE0'. The response status is 200 OK, and the body contains the message 'Wallet Created! Wallet Address: 533B8F11B2BE17C3B37FBFE32D6FCAE0'.

B's wallet balance now:

GET

<http://localhost:2345/wallet/get/balance?address=533B8F11B2BE17C3B37FBFE32D6FCAE0>

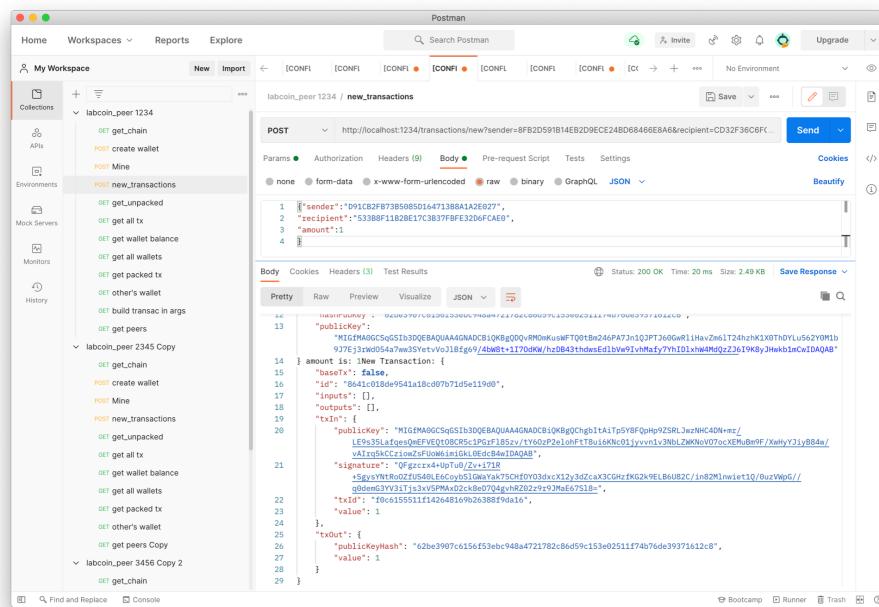
The screenshot shows the Postman application interface. On the left, there's a sidebar with 'My Workspace' containing collections like 'labcoin\_peer 1234', 'labcoin\_peer 2345 Copy', and 'labcoin\_peer 3456 Copy 2'. The main area displays a GET request to 'http://localhost:2345/wallet/get/balance?address=533B8F11B2BE17C3B37FBFE32D6FCAE0'. The 'Params' tab shows a single parameter 'address' with value '533B8F11B2BE17C3B37FBFE32D6FCAE0'. The response status is 200 OK, and the body contains the message '1. wallet address is:533B8F11B2BE17C3B37FBFE32D6FCAE0' and '2. Balance in Wallet is:0 LabCoin'.

Spring 2021, Lab 4 proposal  
14736 Di Lu(dilu), Yuan Gu(guy)

Now A's wallet1 proposed a transaction to B:  
`POST http://localhost:1234/transactions/new`

arguments:

```
{"sender":"D91CB2FB73B5085D164713B8A1A2E027",
 "recipient":"533B8F11B2BE17C3B37FBFE32D6FCAEO",
 "amount":1
}
```



The screenshot shows the Postman application interface. The left sidebar displays a workspace named 'My Workspace' with an environment named 'labcoin\_peer 1234'. Under this environment, there are several API endpoints listed under 'Collections': 'labcoin\_peer 1234' (containing 'GET get\_chain', 'POST create\_wallet', 'POST Mine', 'POST new\_transaction', 'GET get\_unpacked', 'GET get\_all\_tx', 'GET get\_wallet\_balance', 'GET get\_all\_wallets', 'GET get\_packed\_tx', 'GET other's\_wallet', 'GET build\_transact\_in\_arg', 'GET get\_peers'), 'labcoin\_peer 2345 Copy' (containing 'GET get\_chain', 'POST create\_wallet', 'POST Mine', 'POST new\_transactions', 'GET get\_unpacked', 'GET get\_all\_tx', 'GET get\_wallet\_balance', 'GET get\_all\_wallets', 'GET get\_packed\_tx', 'GET other's\_wallet', 'GET get\_peers Copy'), and 'labcoin\_peer 3456 Copy' (containing 'GET get\_chain'). The main panel shows a POST request to 'http://localhost:1234/transactions/new?sender=8FB2D591B14EB2D9ECE24BD6846E8A6&recipient=CD32F38C6FC...'. The 'Body' tab is selected, showing the JSON payload:

```
1 {"sender": "8FB2D591B14EB2D9ECE24BD6846E8A6",  
2 "recipient": "CD32F38C6FC...",  
3 "amount": 1,  
4 }
```

The response status is 200 OK, Time: 20 ms, Size: 2.49 KB.

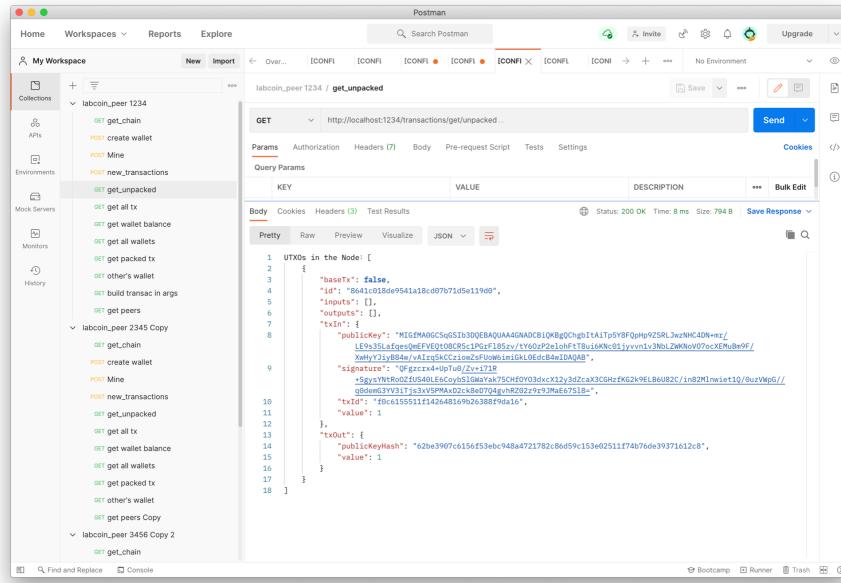
A new transaction is created locally.

The input transaction id is "txId": "[f0c6155511f142648169b26388f9da16](#)". Equals to the previous transaction ID. The value is 1, the output public key hash is B's public key hash.

A's current unpacked transaction:  
`GET http://localhost:1234/transactions/get/unpacked`

This transaction is not yet approved by all the nodes so is local.

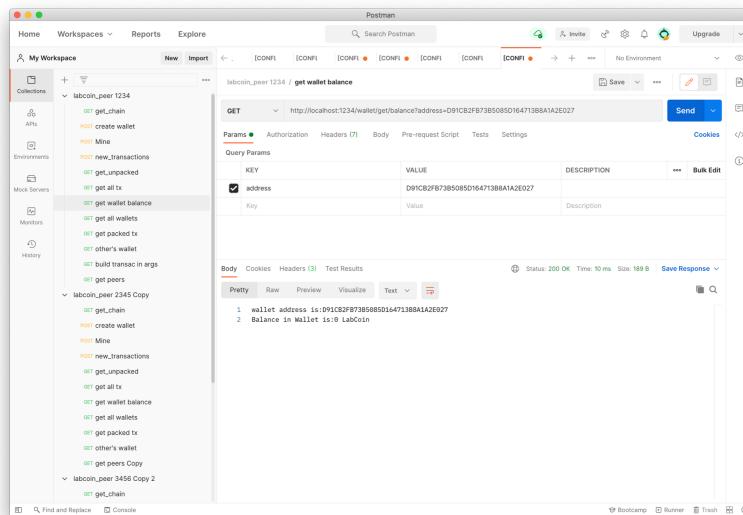
Spring 2021, Lab 4 proposal  
14736 Di Lu(dilu), Yuan Gu(guy)



A's current balance is reduced anyway to prevent repeatedly using the same coin.

GET

<http://localhost:1234/wallet/getBalance?address=D91CB2FB73B5085D164713B8A1A2E027>



B's current balance:

GET

<http://localhost:2345/wallet/getBalance?address=533B8F11B2BE17C3B37FBFE32D6FCACE0>

Still 0, again, the transaction is not agreed by now.

# Spring 2021, Lab 4 proposal

## 14736 Di Lu(dilu), Yuan Gu(guy)

The screenshot shows the Postman application interface. The left sidebar displays a collection named "labcoin\_peer 1234" containing various API endpoints such as "get\_chain", "POST Mine", "POST new\_transactions", and "GET get\_unpacked". The main workspace shows a GET request to "http://localhost:2345/wallet/get/balance?address=533B8F11B2BE17C3B37FBFE32D6FCAE0...". The "Params" tab is selected, showing a single parameter "address" with the value "533B8F11B2BE17C3B37FBFE32D6FCAE0". The "Body" tab shows the response body:

```

1 wallet address is:533B8F11B2BE17C3B37FBFE32D6FCAE0
2 Balance in Wallet is:0 LabCoin

```

Now if A mines a new block:

**POST http://localhost:1234/mine?address=D91CB2FB73B5085D164713B8A1A2E027**

The screenshot shows the Postman application interface. The left sidebar displays a collection named "labcoin\_peer 1234" containing various API endpoints. The main workspace shows a POST request to "http://localhost:1234/mine?address=D91CB2FB73B5085D164713B8A1A2E027". The "Params" tab is selected, showing a single parameter "address" with the value "D91CB2FB73B5085D164713B8A1A2E027". The "Body" tab shows the response body:

```

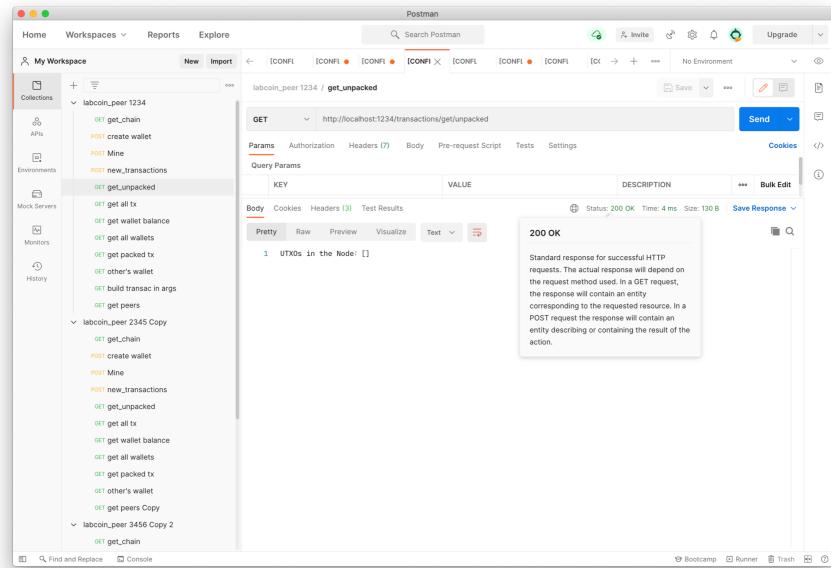
1 New Block from Mining: {
2   "hash": "00010120bd94c38d73a2efb3a2c94dfb27829dc57a4c3b49ea9805fbfd322e072",
3   "index": 3,
4   "nonce": 6693,
5   "previousHash": "0003fb381ef165d5193481fa0ee673fb80e8cd11bbee91730413f6f976991adb",
6   "timestamp": 162032919276,
7   "transactions": [
8     {
9       "baseTx": false,
10      "id": "8641c018de9541a18cd97b71d5e119d0",
11      "inputs": [],
12      "outputs": [],
13      "txIn": [
14        {
15          "publicKey": "MIGfMA0GCSqGSIb3DQEBAQAAQADCBiOKBg0ChgbItA1Tp5YBFQpHg9Z5RLjezNHc4DN+r_",
16          "L69s35Lsafes0eFPEV0B0CR5i1P9g1852v/Yt0d0p2e1ohFTBu16KNv01jyvnVn3hbl2WKhv07ocXEnuBe9F/",
17          "XmyfY1jB8dw+ATrqKCCz1oZsFuW61mGkL0Ed84nDAQdA",
18          "signature": "0FGccx4xUpTuJ2x+e1753",
19          "sSignature": "0FGccx4xUpTuJ2x+e1753",
20          "txOut": [
21            {
22              "publicKeyHash": "62be3987c6156f53ebc948a4721782c8d59c153e02511f74b76de39371612c8",
23              "value": 1
24            }
25          ]
26        }
27      ],
28      "txOut": [
29        {
30          "publicKeyHash": "62be3987c6156f53ebc948a4721782c8d59c153e02511f74b76de39371612c8",
31          "value": 1
32        }
33      ]
34    }
35  ]
36}

```

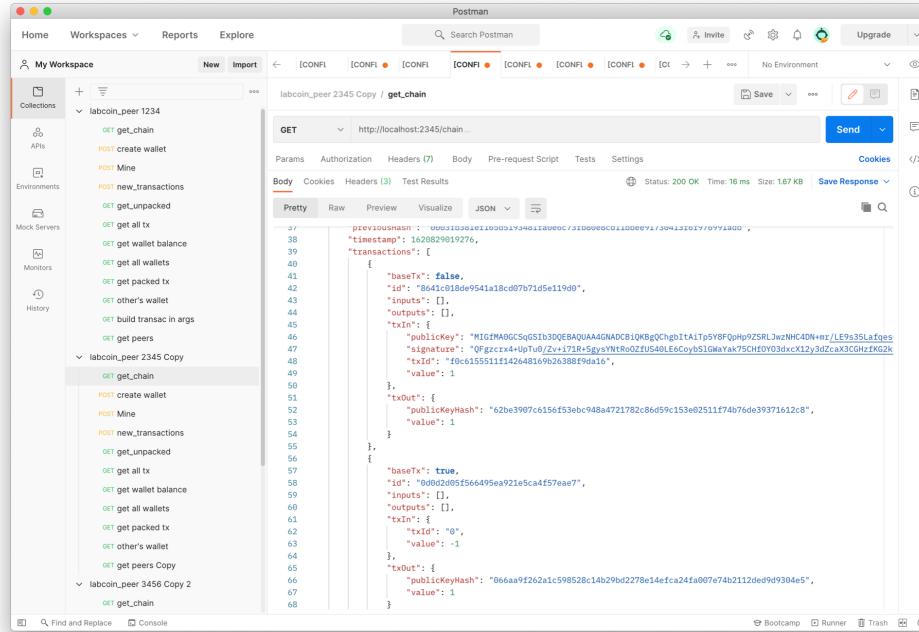
The transaction will be packed by the new block, A now have no unpacked transaction.  
**GET http://localhost:1234/transactions/get/unpacked**

# Spring 2021, Lab 4 proposal

14736 Di Lu(dilu), Yuan Gu(guy)



Now B's chain will be updated because A's chain is longer(2 blocks vs 1 block)  
 GET <http://localhost:2345/chain>



B's wallet B1 balance will be 1 now(the 1 from A), even B did not do any mining.  
 GET  
<http://localhost:2345/wallet/get/balance?address=533B8F11B2BE17C3B37FBFE32D6FCAE0>

# Spring 2021, Lab 4 proposal

## 14736 Di Lu(dilu), Yuan Gu(guy)

The screenshot shows the Postman interface with a collection named "labcoin\_peer 1234". A specific POST request titled "get wallet balance" is selected. The URL is set to `http://localhost:2345/wallet/get/balance?address=53388F11B2E17C3B37FBFE32D6FCAE0`. The "Body" tab shows the response content:

```

1 wallet address is:53388F11B2E17C3B37FBFE32D6FCAE0
2 Balance in Wallet is:1 LabCoin

```

B's UTXO is updated, he will check whether this UTXO is his when making a transaction.  
**GET** `http://localhost:2345/transactions/get/unpacked`

The screenshot shows the Postman interface with a collection named "labcoin\_peer 2345 Copy". A specific GET request titled "get\_unpacked" is selected. The URL is set to `http://localhost:2345/transactions/get/unpacked`. The "Body" tab shows the response content as a JSON array of UTXOs:

```

1 UTXOs in the Node: [
2   {
3     "baseTx": false,
4     "id": "8641c181de9541a18cd87b71dfe119e0",
5     "inputs": [],
6     "outputs": [],
7     "txId": {
8       "pubKey": "3E5E5A40C565E5153D9EB00A445A0C010B8C95C91A1F55YRjQhp9ZSLJwzNHc4DN+rZLr535aInoGEPfEV108C551Pn7f35cv1Yh0P21n1F7Tb1aRN01jyvvn1vZ0L2WKnU0V7uXfPhBp9f/XehY1jy8dwe/A1x1pk4tCz1oxZfUw0k0j14k1.06dc-B8x1DAQAB",
9       "signature": "0Fccxx4+lpTu/2x+1718+Sgym7HtR0zT540bEdcocy51m0wAyv75CHtV0sdx+ct1j3dzaxKGh2MK02k9EL8dUB2C/1x82M1nwxt1Q/u7uzWpG//obm0e7V71j3utPMwadz+ah07d4qz42q32ivv92M46751z",
10      "value": 1
11    },
12    "txOut": {
13      "publicKeyhash": "62b2e3907c156f53ebc948a4721782c8ed59c153e02511f74b76de39371612c8",
14      "value": 1
15    }
16  },
17  {
18    "baseTx": true,
19    "id": "1c10dd60b13c4cca84b6d8b89518ab",
20    "inputs": [],
21    "outputs": [],
22    "txId": "0"
23  }
24 ]

```

### 8.2.7 test new added peers and consensus

NOW Start a new node C, connect to B

Spring 2021, Lab 4 proposal  
14736 Di Lu(dilu), Yuan Gu(guy)

java -jar labBlockChain-1.0-SNAPSHOT.jar 3456 7890 ws://localhost:6789

B now has two peer connections:  
GET <http://localhost:2345/peers>

One is A's p2p listen port, One is C's p2p working port.

The screenshot shows the Postman application interface. On the left, there is a sidebar with 'My Workspace' containing several collections related to 'labcoin\_peer'. One collection is expanded, showing various API endpoints such as 'get\_peers', 'get\_chain', 'POST Mine', etc. In the main panel, a specific request is selected: 'labcoin\_peer 2345 Copy / get\_peers'. The method is set to 'GET' and the URL is 'http://localhost:2345/peers'. The response status is 200 OK, and the body contains the following JSON:

```
1 localhost:5678
2 localhost:56961
3
```

C is newly created, but it will receive the longest blockchain because its length is 1(the genesis block)

GET <http://localhost:3456/chain>.

The screenshot shows the Postman application interface. On the left, there is a sidebar with 'My Workspace' containing several collections related to 'labcoin\_peer'. One collection is expanded, showing various API endpoints such as 'get\_peers', 'get\_chain', 'POST Mine', etc. In the main panel, a specific request is selected: 'labcoin\_peer 3456 Copy 2 / get\_chain'. The method is set to 'GET' and the URL is 'http://localhost:3456/chain'. The response status is 200 OK, and the body contains the following JSON:

```
1 Current BlockChain: [
2   {
3     "hash": "1",
4     "index": 1,
5     "nonce": 1,
6     "previousHash": "1",
7     "timestamp": 16208817556357,
8     "transactions": []
9   },
10  {
11    "hash": "e9e93fb381ef10d5193481fa8e6c73fb80e8cd11bbe91730413f6f976991adb",
12    "index": 2,
13    "nonce": 5134,
14    "previousHash": "1",
15    "timestamp": 162088167102,
16    "transactions": [
17      {
18        "baseTx": true,
19        "id": "2b0c1595111142648169b26388f9da16",
20        "inputs": [],
21        "outputs": [
22          {
23            "txId": "0",
24            "value": 1
25          }
26        ],
27        "txOut": [
28          {
29            "publicKeyHash": "066aa9f262a1c598528c14b29bd2278e14efca24fa807e74b2112ded9f9304e5",
30            "value": 1
31          }
32        ]
33      }
34    ]
35  }
```

Spring 2021, Lab 4 proposal  
14736 Di Lu(dilu), Yuan Gu(guy)

C has the same UTXO as B from broadcast too. C will check whether this belongs to him if he wants to make a transaction.

GET http://localhost:3456/transactions/get/unpacked

The screenshot shows the Postman application interface with the following details:

- Header Bar:** Postman, Search Postman, Invite, Upgrade.
- Left Sidebar:** Home, Workspaces, Reports, Explore, My Workspace, Collections, API, Environments, Mock Servers, Monitors, History.
- Current Item:** labcoin\_peer\_3456 Copy 2, getUnpacked.
- Request Section:**
  - Method:** GET, URL: http://localhost:3456/transactions/get/unpacked.
  - Params:** Authorization, Headers (7), Body, Pre-request Script, Tests, Settings.
  - Query Params:** KEY, VALUE, DESCRIPTION.
  - Body:** Cookies, Headers (3), Test Results.
- Response Section:**
  - Status: 200 OK, Time: 11ms, Size: 1009 B.
  - Save Response, Bulk Edit, Cookies.
- Code View:** JSON response content.
- Bottom Navigation:** Find and Replace, Console, Bootcamp, Runner, Trash.

Let B create a transaction to C:

POST http://localhost:2345/transactions/new

```
{"sender":"533B8F11B2BE17C3B37FBFE32D6FCAE0",
 "recipient":"9BBE36B9E1559EA84B4FD12A9D11AD",
 "amount":1
}
```

The screenshot shows the Postman application interface. At the top, there are tabs for Home, Workspaces, Reports, and Explore. The main header includes a search bar for 'Postman' and various global settings like 'Invite', 'Help', and 'Upgrade'. Below the header, the left sidebar shows 'My Workspace' with sections for Collections, APIs, Environments, Mock Servers, and Monitors. A 'History' section is also present. The main workspace shows a tree view of API endpoints under 'labcoin\_peer 2345 Copy'. One endpoint, 'POST /new\_transactions', is selected and expanded, showing its details. The 'Body' tab is active, containing JSON data for sending a transaction. The response from the server is shown in the 'Test Result' tab, indicating a successful 200 OK status with a timestamp of 50 ms and a size of 2.49 KB. The response body contains a JSON object representing the new transaction.

```
1 "s": "senders": "533B8F1B2BE1783B7FBFEE32D6FCAE0",
2 "t": "recipient": "98BE536b9E1559E8A84B4F0FD12A9011AD",
3 "a": "amount": 1,
4 "c": "category": "transfer"
```

Body Cookies Headers (3) Test Result Status: 200 OK Time: 50 ms Size: 2.49 KB Save Response ▾

Pretty Raw Preview Visualize JSON

```
13 "publicKey": "MIGfMA0GCSqGSIb3DQEBAQAAQADQCBjQKbgC02PKXLMR1PeUvWk6z1SzQLeHtAD2C5Fc9e05SBK2P7LmRoghuIPu09pja1pJRL
14 "amount": 1, "category": "transfer", "recipient": "98BE536b9E1559E8A84B4F0FD12A9011AD", "senders": "533B8F1B2BE1783B7FBFEE32D6FCAE0", "signature": "h0qZew0kMwPrayv05GQ02hWxReyq0nEcFCYXykoEdcH30n1lRgR170NyHf79awsL50rzsBgf3564KwybteV0StpJatBuLr8
15 "txId": "11b18b3d8b7030e0080b10cycEq", "value": 1, "txOut": [
16 "id": "36a1330a0ed491981f72d614060d", "publicKeyHash": "c048784c0b3942fb9aed6fe58375ff991c20d5a3125dbb4c74b3b1a13df69",
17 "value": 1
18 ], "inputs": []
19 }, "txId": "11b18b3d8b7030e0080b10cycEq", "value": 1, "txOut": [
20 "id": "36a1330a0ed491981f72d614060d", "publicKeyHash": "c048784c0b3942fb9aed6fe58375ff991c20d5a3125dbb4c74b3b1a13df69",
21 "value": 1
22 ], "inputs": [
23 "id": "36a1330a0ed491981f72d614060d", "publicKeyHash": "c048784c0b3942fb9aed6fe58375ff991c20d5a3125dbb4c74b3b1a13df69",
24 "value": 1
25 ], "txId": "11b18b3d8b7030e0080b10cycEq", "value": 1, "txOut": [
26 "id": "36a1330a0ed491981f72d614060d", "publicKeyHash": "c048784c0b3942fb9aed6fe58375ff991c20d5a3125dbb4c74b3b1a13df69",
27 "value": 1
28 ]}
```

# Spring 2021, Lab 4 proposal

14736 Di Lu(dilu), Yuan Gu(guy)

Had someone mine a block:

POST <http://localhost:1234/mine?address=D91CB2FB73B5085D164713B8A1A2E027>

```

1 New Block from Mining: {
2   "hash": "00007ead5ad485cd140e58d7421f62b180d669327f4ccdf430019b6dd40166c",
3   "index": 4,
4   "nonce": "2852",
5   "previousHash": "000016248094c38073a2f1b3a2c94d1b27829dc57a4c3049ea9805f8bd323e072",
6   "timestamp": 1620823760652,
7   "transactions": [
8     {
9       "baseTx": "false",
10      "id": "d3edc933abaa4cd5918af72d6a14060d",
11      "inputs": [],
12      "outputs": [],
13      "txIn": {
14        "publicKey": "PwGfWmRfC5eGCJh300CBQUAA46NA0C3i0h@p00n+RfQ0t4h0x34/P4T7n1q3PTj600r11HavZm1T24hzkX0TH
15        DYL5d52Y0M19j37Ej3ned054ytw35vrtv318fg69/dsMht+17970kW
        hz0B43hdeEd1lv91vhnyf7h10l+nM0d273519KbJhwbhlnCwIDAB",
16        "signature": "NoQ2N50whm0gvcY3fFfOKYtdeUv0I
        lvtzxy2f8ocR9Vtcay050u2nxwYe0nCfYYKodcH9m11RwGR17QByhYf779aw5LS0rzv8Gp35akKeybtwQStp
        zRb6L0Q0u2ewh1b1b35d8yv03m081051cYeQ",
17        "txId": "861c91836e5541a3c09771d411980",
18        "value": 1
19      },
20      "txOut": {
21        "publicKeyHash": "c048704cf063942109aeed16f50375ff9916c2055a3125dbb4c74b301d13d169",
22        "value": 1
23    }
24  ]
25}

```

C will have the B->C transaction now

Get <http://localhost:3456/chain>

```

1 [
2   {
3     "hash": "00003fb381ef1e6d5193481fa0e6c73fb80e8cd11bbe91730413fe979991ad8",
4     "index": 2,
5     "nonce": "1",
6     "previousHash": "1",
7     "timestamp": 1620827556357,
8     "transactions": []
9   },
10   {
11     "hash": "00003fb381ef1e6d5193481fa0e6c73fb80e8cd11bbe91730413fe979991ad8",
12     "index": 2,
13     "nonce": "1",
14     "previousHash": "1",
15     "timestamp": 1620827556357,
16     "transactions": [
17       {
18         "baseTx": "true",
19         "id": "28c0551f142648169b26388f9da1e",
20         "inputs": [],
21         "outputs": [],
22         "txIn": {
23           "txId": "4",
24           "value": -1
25         },
26         "txOut": {
27           "publicKeyHash": "066aa97262a1c598528c14b29bd2278e14efca24fa007e74b2112de09e9304e5",
28           "value": 1
29         }
30       }
31     ]
32   }
33 ]

```

Get <http://localhost:3456/transactions/get/unpacked>

# Spring 2021, Lab 4 proposal

## 14736 Di Lu(dilu), Yuan Gu(guy)

The screenshot shows the Postman application interface. In the left sidebar, there's a collection named "labcoin\_peer 3456 Copy 2". A specific API endpoint "get wallet balance" is selected. The request method is "GET" and the URL is "http://localhost:3456/wallet/get/balance?address=98BE36B9E1559EA84B4FDFD12A9D11AD". In the "Params" tab, there is one parameter: "address" with the value "98BE36B9E1559EA84B4FDFD12A9D11AD". The response status is 200 OK, and the body contains the following JSON:

```
1. wallet address is:98BE36B9E1559EA84B4FDFD12A9D11AD
2. Balance in Wallet is:1 LabCoin
```

B has 0 coin now:

GET

<http://localhost:2345/wallet/get/balance?address=533B8F11B2BE17C3B37FBFE32D6FCAE0>

The screenshot shows the Postman application interface. In the left sidebar, there's a collection named "labcoin\_peer 2345 Copy". A specific API endpoint "get wallet balance" is selected. The request method is "GET" and the URL is "http://localhost:2345/wallet/get/balance?address=533B8F11B2BE17C3B37FBFE32D6FCAE0". In the "Params" tab, there is one parameter: "address" with the value "533B8F11B2BE17C3B37FBFE32D6FCAE0". The response status is 200 OK, and the body contains the following JSON:

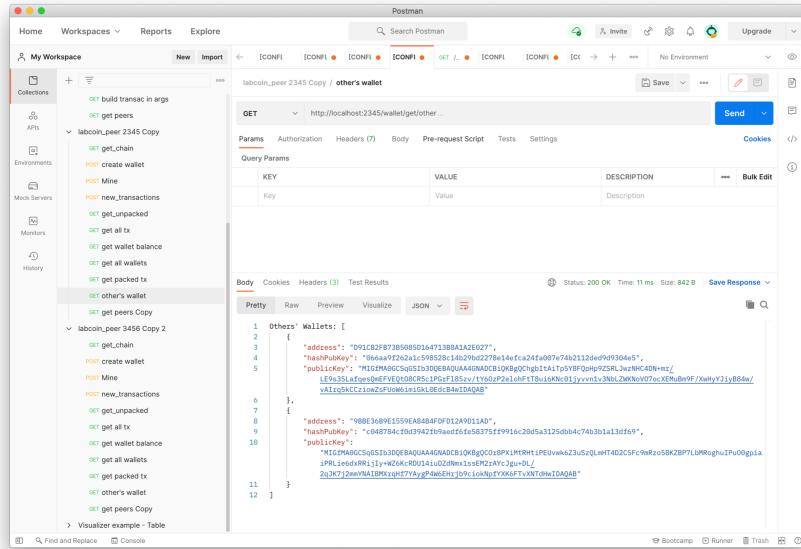
```
1. wallet address is:533B8F11B2BE17C3B37FBFE32D6FCAE0
2. Balance in Wallet is:0 LabCoin
```

If B now check other's wallet map, there will be 2 other wallet in script public key now.

GET <http://localhost:2345/wallet/get/other>

# Spring 2021, Lab 4 proposal

## 14736 Di Lu(dilu), Yuan Gu(guy)



The screenshot shows the Postman application interface. The left sidebar displays 'My Workspace' with collections like 'labcoin\_peer 2345 Copy' containing various API endpoints such as 'get\_peers', 'get\_chain', 'create\_wallet', etc. The main workspace shows a GET request to 'http://localhost:2345/wallet/get/other...'. The response status is 200 OK, time is 11 ms, and size is 842 B. The response body is a JSON object representing 'Others' Wallets' with two entries, each containing an address, hashPubKey, and publickey.

```
1 Others' Wallets: [
2   {
3     "address": "1P9QCBdF9T8B90605014d713B0A1A3E027",
4     "hashPubKey": "16d4aa9f23a21e59b23014a270a2278e14afca24fa007c74b21134e9f9304e6",
5     "publicKey": "MIGfMA0GCSqGSIb3DQEQUA4EcAOB1OgQChpdtA1tS9YBQnp72zRL2vHc4DN+ml_
6     L59u3L5L4oemqFVU70tBC8Cs1Pcf7185zyrYy6QfP2xloFt8u10Xh01jyvvn1v3N0L2wKoqv07ccXEmu8n9f/Xeby7jly884w/
7     vAfrs5kC7z1owZfUuW61x10LBEdcB8w4TQAB"
8   },
9   {
10     "address": "98BE3d9E1559EA8A848dFD12A011AD",
11     "hashPubKey": "c4d9784c10394219aefdf5e5375ff991c20d5a3125dbb4c74b3b1a13df69",
12     "publicKey": "MIGfMA0GCSqGSIb3DQEQUA4EcAOB1OgQChpdtA1tS9YBQnp72zRL2vHc4DN+ml_
13     iPR1eddxR81jLy+k26KcRDU141u0Z0NmxisEM2zAYcJgu+DL_
14     2qJK7j2mnNAIBXxqH77AygPd0EHj09;1ckNgfYXX6FTvXNTdHeIDQAB"
15   }
16 ]
```