DLC Lab Session 2: Simulating a Network and Wallets in Detail

Now that you have familiarized yourself with the Bitcoin Core software we will begin to use it to learn more about how blockchain networks work. Once you have some working knowledge for this (which will help make sense of the theory learned in lectures) we will move on to learning the practical considerations for keeping your digital currency safe.

Part A: Simulating A Network

You have installed Bitcoin Core. For this session it will be easier if you add the folder containing the different bitcoin programs (bitcoind, bitcoin-cli) to your $PATH. To see what folders are already automatically searched through when you command the terminal to run a program, type:

```
echo $PATH
```

Add the bitcoin/bin folder to your $PATH by typing:

```
export $PATH=$PATH:opt/bitcoin/bin
```

Now you should be able to run bitcoind, bitcoin-cli, or bitcoin-qt from anywhere.

We will simulate a 3-node regtest network. We will work from the terminal now, and not with the user interface. Start your first node with:

```
bitcoind -daemon -regtest
```

Here, 'daemon' will tell bitcoind to run silently (without printing all of its processes). Your first node will store its data in a hidden folder. To see it, type:

```
cd
ls -a
```

and look for '.bitcoin'. Inside you will see the configuration file, and block data for mainnet, testnet, and regtest. Now, to start more nodes, you will need to create new directories for each node's blockchain data and create a configuration file for each. Type:

```
cd
mkdir .alice
mkdir .bob
```

Alice and Bob will have two slightly different configuration files which define their ports and allow each other to connect. I'll explain how it works with Alice, and you can figure it out for Bob. Type:

```
nano .alice/bitcoin.conf
```

A text editor will open in the terminal. Copy and paste the following into it:

```
[regtest]
server=1 # tells Bitcoin-Qt and bitcoind to accept JSON-RPC commands
rpcuser=YOUR_USER
rpcpassword=YOUR_PASSWORD
rpcallowip=127.0.0.1
rpcport=9334 #Listen for RPC connections on this TCP port
port=18445
```

```
addnode=127.0.0.1:18446 # add bob's IP
addnode=127.0.0.1:18444 # add your original node's IP
```

Save by clicking ctrl and O, then hit enter. Exit nano text editor with ctrl and X. Now do the same for Bob's configuration file, but be careful to change his rpcport (to 9333), his port (to 18446) and make sure to add Alice's node IP (you just set her port to be 18445).

With these we can start two more nodes. Type:

```
cd
bitcoind -daemon -regtest -datadir=.alice
bitcoind -daemon -regtest -datadir=.bob
```

If everything worked correctly, your original node should have two new peers. To see, type:

```
bitcoin-cli getpeerinfo
```

Can you see three peers? Check the IP addresses ("addr") match those in the configuration file. Now you have simulated a network!

To run commands as Alice or Bob, it is easiest to set an *alias*. This way you don't need to type all of the options (e.g. your user, password, port, data directory) for every command. Type:

```
cd
alias alice-cli="bitcoin-cli -regtest -rpcuser=YOUR_USER -rpcpassword=YOUR_PASSWORD -rpcport=9334 -datadir=.alice"
alias bob-cli="bitcoin-cli -regtest -rpcuser=YOUR_USER -rpcpassword=YOUR_PASSWORD -rpcport=9333 -datadir=.bob"
```

Now investigate how three different nodes maintain their own wallets (their own set of private and public keys). Check your original node's balance and compare with Alice's and Bob's. You may need to generate blocks to mint some new coins. Hint:

```
alice-cli getbalance
bob-cli generatetoaddress <number of blocks> <bob's address>
```

Compare the blockchain information for each node (`getblockchaininfo`). Have all nodes agreed on the blockchain height? If you generate blocks with Alice, will Bob's blockchain automatically update? The three nodes should automatically maintain consensus, unless one of the nodes proposes an invalid transaction or block, then the others will refuse it.

Part B: Creating Transactions the easy way

In the first session you learned how to make "*coinbase"* transactions (there is one with each new block you mine). That's the transaction that mints new bitcoin. How about sending a transaction from your original node to Alice? The simplest way is to generate a new address for Alice and to use:

```
bitcoin-cli sendtoaddress <one of alice's addresses> <amount to send>
```

Check Alice's balance. Did she receive it? No? Why not? Check the transaction exists by copying the Transaction ID that was returned from above and type:

```
alice-cli gettransaction <Transaction ID>
```

You'll see all the transaction's details, including the amount you chose to send. So Alice has the transaction in memory. However, the transaction hasn't been processed yet (it has 0 confirmations).

If you do the same for Bob, you will notice he is also aware of the transaction. This means if he mines a block he can include that transaction in it and claim the fee *and* the newly minted bitcoin:

`bob-cli generatetoaddress 1 <one of bob's addresses>`

Check Alice's and Bob's balances again to see they have both increased. If you want to see your balance in a more detailed way, you can use `listunspent`. This will list all of the `unspent outputs' that the wallet is aware of and has the private keys to access. You can think of unspent outputs from transactions as coins. The wallet balance is the sum of all the unspent outputs controlled by a wallet.

`sendtoaddress` is the simple method for creating transactions. This way, you can't define the fee you are paying, you can't choose which coins you are spending, and you can't write any fancy scripts to program the rules (the contract) by which the bitcoin can be unlocked. For those of you who are interested in more advanced functionality, you can learn to *process raw transactions*.

Part C: Understanding Wallets

"A common misconception about bitcoin is that bitcoin wallets contain bitcoin. In fact, the wallet contains only keys. The "coins" are recorded in the blockchain on the bitcoin network. Users control the coins on the network by signing transactions with the keys in their wallets. In a sense, a bitcoin wallet is a *keychain*." - Mastering Bitcoin, Chapter 5

How your keys (public and private) are managed determines how safe your digital assets are. Wallets are key-management software that manage long key-chains for you. There are many types of wallet: Computer wallets (like Bitcoin Core), hardware wallets, mobile wallets, exchange-held wallets, etc. **Ask yourself:** *would you give your house keys to another person if you wanted to keep your house secure?* Well the same goes for bitcoin keys. If you hold bitcoin on an exchange, but you don't actually have the keys, then the exchange has ultimate control of your assets. If you keep your keys on your mobile, how sure are you that an attacker can't break in to your device? There are a million ways for online devices to be hacked, and there is no-one you can trust more than yourself to look after your own assets. The industry standard right now is to store your keys on an off-line hardware wallet, and to put this in your safe. But even this approach has limits.

Most wallets (keychains) are built from a single (very very large) random number. This number is converted into a 12 or 24 word *mnemonic* that is easier for humans to read and remember. With this mnemonic, the keychain can be recovered at any time. Typically, users will back up a few copies of this mnemonic in different locations. This secures them against accidental loss, or disaster (house burns down). However, it doesn't secure users very well against theft since a thief can also recreate the entire key-chain if they get access to a copy of the mnemonic.

The next step-up in digital asset security is called a multi-signature wallet. Unfortunately this is rather cumbersome to use. We will experiment with multi-signature wallets soon so you can learn the most secure methods for securing your digital assets. For now just know that there are some 'custody protocols' that define all the steps you need to take to look after your bitcoin, and tell you about possible threats and failure modes. These are basically a user-manual for the extra paranoid user!

https://glacierprotocol.org/

https://cerberus.clavestone.io/

https://www.smartcustody.com/2019-03-25-SmartCustody-_Simple_Self-Custody_Cold_Storage_Scenario_v1.0.0/