Project 4: Blockchain Ledger

1. Overview

Project Release: Monday 04/16/18
Project Due: Tuesday 05/01/18

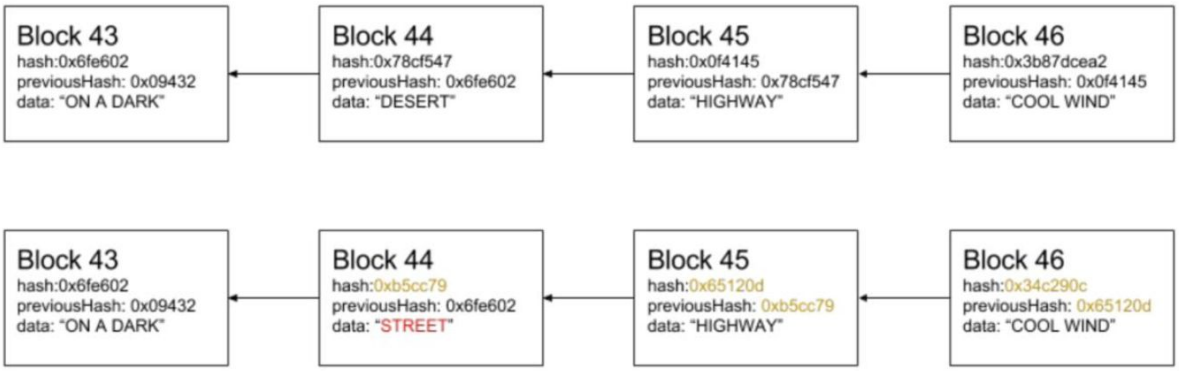
In this project, you will be introduced to the basis of blockchain, and implement a simplified version on your own. You can download the starter package for Go and also the handout through the course website.

2. Introduction

Cryptocurrency, for example, bitcoin, is a decentralized digital currency. Such a currency has the attach prevention mechanism to guarantee that the content inside the block cannot be changed and prevent network congestion caused by creating blocks maliciously. The attack prevention mechanism uses a cryptographic proof-of-work function that is designed to be computationally hard to execute. Clients compete to be the first to find a solution to the proof-of-work (introduced in Section 3) in order to get their signature attached to a sequence of transactions. If a client wins, they are rewarded with one coin. The work to solve proof-of-work is called mining. In this project, we spare you from the distributed environment details and you only need to implement the basic cryptocurrency functionalities. You're provided with a mining application that will award you coins if you successfully solve the difficult work and add a new block into the network. Each node, as a client, in the network needs to be able to solve the computing puzzles and communicate with other nodes in the network. A coin will be awarded automatically into the client's wallet once a block is mined and added successfully.

Your implementation should be able to create a new block and broadcast it to all the nodes in the network. A block can be added into the chain only when all the nodes in the network approve the new block. Also, when a new node is connected to the network, it needs to download the block chain from its peer.

For mining, you need to implement the following simplified algorithm: given a block, you need to calculate hash based on all the information contained in the block and the hash of the previous block. As a result, if someone wants to modify any block in the chain, the blocks following that block will be invalid. And the work to modify all the blocks following that block will be too much (like the figure below suggests).



So, what's the proof-of-work used in this project? In our project, the proof-of-work needs to calculate a hash with a prefix containing 0s. For example, if the difficulty is set to 10, the calculated hash should have a prefix containing 10 leading 0s. The figure below suggests the valid and invalid cases with difficulty set to 4.

Difficulty: 4

binaryHash: 0000100011110011.. VALID
hash: 08f3..

binaryHash: 0001011010111100.. NOT VALID
hash: 16bc..

3. The Code and General Guidelines

You need to modify **blockchain.go.** You will be provided with transport library similar to project 2 under the "labrpc" directory and find test_test.go to test your work under the working directory.

What is provided to you:

• We provide a transport layer that different nodes can communicate with each other to send the new block mined and also download block chain from other nodes upon start up.
• We provide test cases under the working directory

Your code should do the following:

• You will be given the functions that need to be implemented in addition to the helper methods that you may choose to write, which the test suite uses. After a block is mined, the node should broadcast this new block out to all the nodes on the network to acquire approval to add this block to the chain.
• For the first block in a blockchain network (which should be the first block of each copy of blockchain) is called genesis block, it has no previous block hash. Therefore, you can

directly hardcode the first block, and create this hard coded block when your node is started.

• In order to find a hash meeting the proof-of-work requirement, we need to calculate different hash based on the same block, which contradicts the consistency of hashing. As a result, we can add a nonce into our block to achieve our goal. You can brute-force enumerate all possible nonce until you mine the hash that meet the requirement. To calculate hash, you can use SHA256 in go.

• To add a new block mined by others, the validity of this block needs to be verified. For a new block, it has to have the index right after the last block in the chain. Also, the previous hash value should be the same to the hash value contained in the chain. And the hash value contained in the new block itself should be correctly calculated. If there are multiple new blocks broadcasted, we want to choose the block containing the earliest Timestamp.

It is possible that blockchain on different nodes will have different lengths in the distributed environment. Thus, when download the block chain from others, the blockchain with the longest length should be chosen. For blockchains with the same length, we want to choose the one containing the earliest timestamp in the last block.

4. Running and Testing

You can run the tests via following command -

```
go test -run 4
```

5. Hand in your work

You can submit a zip containing your solution along with the files we provided **in the same directory structure.** Also, please include a **group.txt** containing you and your partner's name and Andrew id. Make sure your code can be compiled and the tests can be run directly. Please document the directory structure for the GOPATH and GOROOT.