Peer to Peer Blockchain

Ankur Singh, Mahesh Meena, Prashant Kumar
160129, 160375, 160500

Indian Institute of Technology, Kanpur

1 Introduction

Blockchain is stirring a technological revolution that could forever change how we conduct our affairs online. Many expect it to disrupt practically all industries, providing a platform that is secure enough to foster trust and confidence even without a controlling authority to oversee the system.

In 2008, a white paper prepared by a person with the pseudonym Satoshi Nakamoto described the details of a "Peer to Peer Electronic Cash System" that eventually launched as Bitcoin. The system that was presented eliminates any trusted financial institution, such as a central bank. Instead, all the transactions will be recorded and validated by all nodes on a peer-to-peer network.

In a Peer-to-Peer architecture, one doesn't need a central server to maintain the current state of a blockchain. There isn't a central authority like a bank that maintains the state. Instead, all the nodes in the Bitcoin network that wish to maintain a copy of the Bitcoin blockchain update their copies of the blockchain to include your transaction. That way, as long as 51 of the nodes in the network "agree" on the state of the blockchain, it maintains its fidelity.

Inspired by the above idea, in this work of ours, we will program a blockchain so that it uses a Peer-to-Peer architecture instead of a central server. For this purpose, we will employ the P2P library written in Go called go-libp2p. We will make use of hashing to maintain the integrity of the blockchain.

2 Goals

- Create our blockchain
- Use hashing to maintain integrity of the blockchain
- Use a peer to peer network instead of a central server for our blockchain

Week 2

Ankur Singh, Mahesh Meena, Prashant Kumar
160129, 160375, 160500

Indian Institute of Technology, Kanpur

1 Technology

We start coding the blockchain in Go and use the following libraries initially:

- Spew, used for visualization of blockchain to aid in debugging
- Mux, used for writing web handlers
- Godotenv, a port of the Ruby dotenv project which loads env vars from a .env file

2 Modelling Data

We define the structure of each of our blocks that will make up the blockchain. Each Block contains data that will be written to the blockchain. The contents of the Block are:

- **Index** is the position of the data record in the blockchain
- **Timestamp** is automatically determined and is the time the data is written
- **Data** of the blockchain
- **Hash** is an identifier representing this data record
- **PrevHash** is an identifier of the previous record in the chain

We use hashes to identify and keep the blocks in the right order. By ensuring the PrevHash in each Block is identical to Hash in the previous Block we know the proper order of the blocks that make up the chain.

3 Hashing and Generating New Blocks

Hashing data is done for 2 main reasons:

- **To save space**: Hashes are derived from all the data that is on the block. It's much more efficient to hash that data into a single SHA256 string or hash the hashes than to copy all the data in preceding blocks over and over again.

- **Preserve integrity of the blockchain**: By storing previous hashes, we're able to ensure the blocks in the blockchain are in the right order. If a malicious party were to come in and try to manipulate the data, the hashes would change quickly and the chain would "break", and everyone would know to not trust that malicious chain.