User Documentation: Peer-to-Peer Blockchain Network using Python Socket Programming

Overview

This project provides a framework for creating a decentralized P2P network where nodes can connect, exchange data, and participate in blockchain activities such as mining, blockchain transfer and transaction broadcasting. The network consists of three types of nodes:

- Full Node: Fully participating node in the blockchain network, which store the blockchain in its entirety.
- Miner: Node responsible for creating new blocks through the mining process and maintaining a transaction pool.
- Light Node: Lightweight node that stores the hash of each block instead of the entire blockchain and can interact with the network for specific tasks like broadcasting its own transactions and requesting access to the complete blockchain from the full nodes.

Resilient Features

In our peer-to-peer network, we leverage socket programming to establish direct communication channels between nodes. Socket programming allows us to create connections over a network, enabling data exchange between devices.

With sockets, each node acts as both a client and a server, capable of initiating connections and accepting incoming connections. This bidirectional capability is essential for decentralized networks where all nodes have equal standing.

By utilizing sockets, we tap into the underlying infrastructure of the internet, leveraging its robustness and scalability. This approach enables us to build a network that can span across different devices and platforms, facilitating seamless communication regardless of the underlying technology stack.

In essence, socket programming serves as the backbone of our peer-to-peer network, enabling nodes to connect, communicate, and collaborate effectively in a decentralized manner.

This is a completely decentralized, peer to peer network and offers the following additional functionalities :

- Network Propogation and Broadcasting: Messages are efficiently broadcasted through the network without the need of establishing direct links between any two nodes.
- Spam Protection : Ensures that a node doesn't broadcast the same message repeatedly thereby flooding the network unnecesarily.

Dolev-Strong Protocol Implementation

Our network module is fortified with the robust **Dolev-Strong** protocol, ensuring seamless consensus among interconnected nodes within the peer-to-peer network. Spearheaded by the miner, this protocol orchestrates the dissemination of blocks while achieving unanimity among the network nodes.

The implementation employs a **bitmasking** technique to track the status of signatures associated with each message. In this approach, each node is authorized to set its corresponding bit to 1 upon integrating a

message into its pool. Subsequently, the node appends its signature to the message and propagates the message across the network.

Following a series of k rounds, if a solitary block emerges within the message pool of a node, it is deemed as the accepted block, signaling successful consensus attainment.

Various Node Types

Full Node

- Chain Management: A full node maintains a complete copy of the blockchain. It can add new blocks to the chain.
- Broadcast Transactions: Users can broadcast new transactions to the network using a full node. The node ensures that the transaction is propagated to other nodes for validation and inclusion in the blockchain.
- Network Connectivity: Full nodes can connect to other nodes in the network to synchronize blockchain data and exchange information.
- Display Chain: Provides functionality to display the complete blockchain stored by the node.
- Request Access to Blockchain: Full nodes can request access to the blockchain from other full nodes in the network. This feature allows full nodes to synchronize their local copy of the blockchain with other nodes in the network, ensuring consistency and integrity.
- Respond to Access Requests: Full nodes also respond to requests from other nodes or users for access to the blockchain data, facilitating decentralized access and distribution of blockchain information.

Miner

- Mining: The main functionality of a miner is to create new blocks by solving complex mathematical puzzles (Proof of Work). Miners continuously perform computations to find a valid block hash and add new blocks to the blockchain.
- Broadcast Transactions: Similar to full nodes, miners can broadcast new transactions to the network for validation and inclusion in the blockchain.
- Broadcast Blocks: After successfully mining a new block, miners broadcast the newly created block to the network for validation and propagation.
- Transaction Pool Management: Miners maintain a pool of unspent transactions waiting to be included in the next block. They can display the transaction pool and select transactions for inclusion in the next block.

Light Node

- Lightweight Operation: Light nodes operate with reduced resource requirements compared to full nodes. They store the hash of the blocks but do not store the entire blockchain and can still participate in network activities.
- Request Access: Similar to full nodes, light nodes can request access to the blockchain data from other nodes in the network.
- Broadcast Transactions : You can perform actions such as requesting access to the blockchain, broadcasting transactions, connecting to other nodes, stopping the node, displaying the blockchain.

How to Run

After cloning the repository in your local machine, navigate to the corresponding directory and run the following command :

- python blockchain.py

This prompts you to select the type of node you want to run: full node, miner, or light node, simply follow the instructions on the terminal to choose your node. Once you've selected the type of node, your machine's IP address will be fetched automatically. Follow the instructions displayed in the console to interact with the node.