For this program, my thinking is as follows:

1. Server keeps the blockchain instance. When a new client connects to the server, it sends client the last block in the blockchain.
2. Client builds a block based on the last block, and then mines it. This will generate a new block. Then client sends this block to server.
3. After server gets the block, it needs to check this block on the blockchain. Because there may be many clients doing the mining job simultaneously. When one client finishes the job and sends the block back to server first, server will only add this block to the blockchain and ignore all other clients' work. After checking, if "sPreHash" of this new block is same as the last block in the blockchain, which means this block is the first one which comes back, server will add this block to the blockchain. Otherwise, server will discard it.
4. After adding a new block, server will write it to a certain file (save.txt) in order to store it.

I use these techniques in this project:

1. I use boost::asio as the asynchronous internet connection. Every asynchronous operation starts a new asynchronous operation, keeping the service.run() busy. Main logic is in do/on_read/write functions.

2. I use boost::property_tree to transfer data as json between client and server. It makes transformation from object to string or string to object easier.
3. I use openssl/sha.h to generate hash value of a block.

I have submitted a bash file to build my code. But first, you need to make sure you have boost and openssl libs in /usr/local/lib and boost and openssl files in /usr/local/include. You need to open at least two terminals-one server and one client-to run this project, using commands “./server” and “./client”. Mining process will echo on the terminal, and blockchain will store in save.txt. Next time when you run the server, it will automatically read from save.txt.