PeerCoin

Austin Loza 9715590

March 18, 2018

Abstract

The main purpose is to be able to share small amounts of data from peer to peer without use of a server. On top of that, the blockchain should be capable of secure payments as well as capable of hosting some simple applications related to its previously stated purposes. Essentially, the person would pay a coin to get some data from a particular peer or a few coins along the way to get chunks from several peers, with the core of every request or application being a transaction of coins. To that end, there will be symmetric RSA encryption to secure the transaction between the two parties, and SHA-2 hashed "fingerprints" of past transactions to verify the chain. The "currency" aspect of this project is more like a fuel for receiving and isn't meant to be worth anything. The purpose of implementing a peer-to-peer file sharing network using a blockchain is to attempt to solve the problem of "leechers" in bittorrent.

Introduction

A blockchain is a distributed data structure made up of "blocks" that contain a hash of the previous block, a timestamp, and other data. This is usually implemented on a peer-to peer basis, where as transactions are made between peers, the chain increases in length. This allows every transaction to be verified and peers to be connected amongst themselves. Oftentimes this is used for secure and anonymous cryptocurrencies, where clients can exchange "coins" between anonymous addresses or create "contracts", as in the case of

Etherium, for compensation to be doled out when some task or set of tasks is completed.

Research Done

I have read the documentation for Bitcoin [1] and Etherium [2], as well as looked at some open-source implementations of cryptocurrencies on GitHub. Furthermore, I've looked into blogs [6] that explain how to build and generate peer-to-peer blockchains as well as papers outlining extensions to blockchains [3] [4]. Foundations for the use of blockchain as a transport layer for files and data have been laid by groups like Protocol Labs, the creators of Filecoin [8]. It's becoming ever clearer that the future of the internet will be found and possibly defined my blockchains and the unrealized potential they currently possess.

Revised Implementation

Blocks in the blockchain will contain a hash of their data, a block number, a parent's hash, a transaction count for the block, and a history of all transactions made. The data is sent alongside this, the "coins" effectively being the fuel for these file transfer transactions. When a user runs out of coins, they can no longer request files. Sharing files allows the user to acquire more coins so they can continue receiving data.

Results

I was able to send and receive data between nodes as well as, given a downloaded version of the blockchain, recognize all users on the network and share between them. The core of this project was a definite success. Files transfer properly, and when a user is out of coins, they can no longer share files. A blockchain file is made documenting the history of the network, down to the genesis block. Users can share and unshare files, getting coins for every file shared and coins if their shared file was requested. The project, as a proof-of-concept, works, though for it to be a viable method for file transmission will require significant refinement.

Future Work

If I were to continue on this project I would re-include RSA encryption for the data transfer as well as some cleanup of the transaction code to allow for inter-chain transaction passing. I would refine the inter-chain communications as well as add a time-based component to coin transfer (e.g timestamps that have some use). Furthermore, I would work on protecting the truthfulness of the chain, making sure it remains valid between transfers.

Comments

Deliverable C General Comments

I was able to implement file transfers between nodes, and while the details still require refinement, the project was successful. I had a few difficulties in setting up the GUI, so I decided against making one. A few things about the formatting and scope were changed due to time constraints. I did not encrypt the sent data with RSA encryption and I dropped timestamps as well. Currently, the project allows for data transfer alongside a blockchain, not entirely on top of it. However, I have reached my third milestone, being able to send data through the network using the blockchain as a throttle, however, lacking user addresses. There were a few challenges with this project, explained below in the Challenges section.

Challenges

There were many challenges I came across in the creation of this project, one being time pressures from having a very busy quarter. When writing the client/server code, I had difficulties implementing it in C++, so I wrote it in python and called it from the blockchain's system. This required learning the python C/C++ API, which took some time. I also had difficulties with assigning user addresses and for the current version I have usernames taking their place. Furthermore, I've had difficulty with blockchain history being transferred between nodes. It seems it would be easier with a centralized server hosting the chain history to be downloaded. The client can read the blockchain history and continue from there. However, at present the approach I have is not scalable to larger networks because of the lack of verification. Merging chains was also an issue I was unprepared to deal with. For example, when two sets of nodes interact between other nodes in their set and modify their set's chains, then two nodes (one from each set) try to communicate with vastly differing chain histories that may diverge anywhere along the path, there is no way in my program, at present, to merge these chains. I attempted to create validation functions for this purpose, but found it to be a difficult problem I didn't have the time to solve. Additionally, giving coins to every user that shared a portion of the requested file required creating a payment algorithm that would split requests up and would have required a massive overhaul of my client and server code as well as the blockchain interfaces.

Testing suggestions

NOTE: python2.7 is required

For testing, I recommend looking at coins.cpp . It's what I used to generate the genesis block, but also what I used to test inter-user communication given different nodes and access to a valid blockchain. Also look at README.md, which gives a general description of the functions I used from the blockchain and how to use them. It'll give a good idea of how to test the blockchain component. You can test by adding code to test.cpp and then "make test". Blocks.chain is the chain itself, saved every time the client terminates, though this has to be done manually for tests like coins.cpp. Blocks.chain is text and human-readable, look at it with cat. For data transfer, as long as ./gen (coins.cpp) has ben made and run, then you can make and use client.cpp to test the data transfer. It requires server.py running (if the sender and requester are in the same folder, the blockchain will update properly), which reads the files shared in peercoin.config . The client will assume the user is the one listed in the file and the server will only allow the files shared in the peercoin.config file to be sent.

Older Details

General Comments - From B

I was a lot busier than expected this quarter, preventing me from making as much progress and with as many updates on this project as I would have liked.

However, I will definitely be able to get this project to at least milestone three on my timeline below. Progress has been slow, but I'm still confident in my ability to finish the core of this project. Currently, RSA encryption and decryption with key generation is the only portion implemented. Hashing is currently in progress. Once the hasher has been completed (I insist on implementing the SHA algorithm myself), I will begin working on transactions and inter-node operations. By the time I've completed the second milestone, I believe the core of the project would be complete. The second milestone would contain a network simulation as a proof of concept for the networked implementation that would be created by the third milestone. The third milestone would be scaling things up into a real network, providing a method for these distributed nodes to connect to each other between different machines, plausibly even in different networks.

Deliverables

My deliverables will be source code and progress reports submitted as required as well as available on GitHub in a folder named "Reports"

(Link: https://github.com/SystemicCypher/PeerCoin)

Timeline

The first milestone is to have a functioning CLI program that encrypts, decrypts, and hashes in the blockchain's format.

The second would be a CLI program that can send a payment into another instance of the CLI program, that decrypts it. (e.g a second node in the blockchain)

The third would be implementing the ability to share data along with payment between the computers in the network, utilizing this software. (distributed nodes)

The final would be providing a GUI for the program and refining the interactions with the blockchain.

Further refinements will be made if time permits.

References

1. Nakamoto, Satoshi Bitcoin: A Peer-to-Peer Electronic Cash System

https://bitcoin.org/bitcoin.pdf

(Used to gain understanding of cryptocurrency/blockchain implementation and functions.)

2. Wood, Gavin ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER

   http://gavwood.com/Paper.pdf

   (Used to gain a better understanding of cryptocurrency/blockchain implementation and functions along with the Bitcoin paper. I've based my project a bit more on this specification over Bitcoin's.)

3. Greenspan, Gideon

   MultiChain Private Blockchain - White Paper

   https://www.multichain.com/download/MultiChain-White-Paper.pdf

4. Back, Corallo, Dashjr, et al.

   Enabling Blockchain Innovations with Pegged Sidechains

   https://www.blockstream.com/sidechains.pdf

5. Okupski, Krzysztof

   Bitcoin Developer Reference

   http://enetium.com/resources/Bitcoin.pdf

6. Ecomunsing, Build Your Own Blockchain: A Python Tutorial, Blog

   http://ecomunsing.com/build-your-own-blockchain

7. Cohen, Bram

   The BitTorrent Protocol Specification

   http://bittorrent.org/beps/bep_0003.html

8. Filecoin: A Decentralized Data Storage Network

   https://filecoin.io/filecoin.pdf