BasedCoin: A Proof-of-Work Cryptocurrency with Novel Contracts

1st Olliver Aikenhead
Dept. of Computer Science and Engineering
University of Nevada, Reno
Reno, United States

2nd Nicholas Alvarez
Dept. of Computer Science and Engineering
University of Nevada, Reno
Reno, United States

***Abstract*—One of the blockchain's most common implementations is with cryptocurrency. Proof-of-work and proof-of-stake are the most common verification methods. We seek to create a proof-of-work cryptocurrency that includes contracts. Existing cryptocurrencies, like Bitcoin and Ethereum, are large-scale implementations. Bitcoin is proof-of-work, and Ethereum is proof-of-stake with smart contracts. With a large amount of users and centralized coin exchanges, weak links can appear in the system, and with higher exchange rates for each currency comes a bigger a target for attacks. BasedCoin is a middle ground between the two cryptocurrencies. It attempts to fill a niche gap with a small-scale blockchain implementation. A Python application with a user interface for transactions and communication between other clients on the network is our solution. Contracts are implemented in the form of full Python programs, not functions of one. We find that the application is viable for users who want to host their own blockchain and use a small-scale cryptocurrency. The users may find a valuable application for BasedCoin, one which we cannot predict. In this way, the project itself could be used for other verification purposes, not solely as a cryptocurrency.**

***Index Terms*—peer-to-peer computing**

I. Introduction

Cryptocurrency is becoming increasingly popular. What started as a niche idea has become a multitude of "coins" and investment frenzies. There has been a shift from cryptocurrency as an anonymous way to send money towards a stock market-esque idea. In conjunction with the rise of popularity, third-party services have appeared to host wallets, act as exchanges, and make cryptocurrency more accessible. This comes with some associated risk, as the third-party controls each user's wallet. This is not necessarily malicious by itself, but trust must be placed in someone else's hands, and this can be abused. Third-party hacks are not uncommon, and due to blockchain's immutability, transfer of funds is irreversible [1].

There exists a gap in the market for a locally hosted, small-scale blockchain solution for sending money. One person in a friend group or company department could set up the application and begin to let others join. Verification would be done by one of many miners on the network. The system is meant to be low-stakes and provide a simple way for users to exchange currency without some of the risk posed by large third-party providers.

II. Description

A. Application Prototype

This application is built in a way that allows modularity and easy scaling with an increasing number of users. With a future integration into a database storing user credentials, the system would allow for multiple accounts. For the prototype, a set of two different user accounts were used for login purposes. The application implements a Proof-of-Work (PoW) verification process and includes contract functionality.

1) Login: The login page is simple and shows the username and password entry fields. A few security measures are in place. After four failed login attempts, the application will close. Assuming the user does input the correct password, only the password hash is sent for authentication purposes. If the username and hash match what is in the database (currently a static dictionary), the user is granted access to the application.

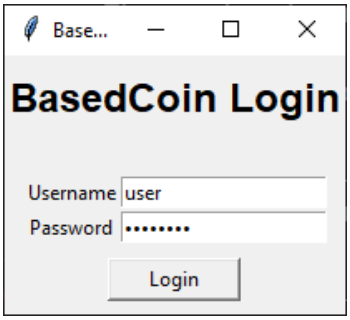

Fig. 1. The Login window.

2) Interface: The interface is the heart of the application, accessible once the user logs in. In the status bar, one can see which user has logged in.

a) Wallet: The Wallet tab shows the user's receiving address and their current balance. Currently, the address is randomly generated, to demonstrate what a user's wallet address may be. The initial amount of BasedCoin is also randomly generated between 1 and 100. As the user sends money or executes contracts, the listed amount will decrease accordingly. As they receive money, the amount will increase depending on the amount of BasedCoin they were sent. The Update button allows the user to check for any changes in their

account balance. Switching to the Wallet tab from somewhere else will also automatically refresh their wallet balance.

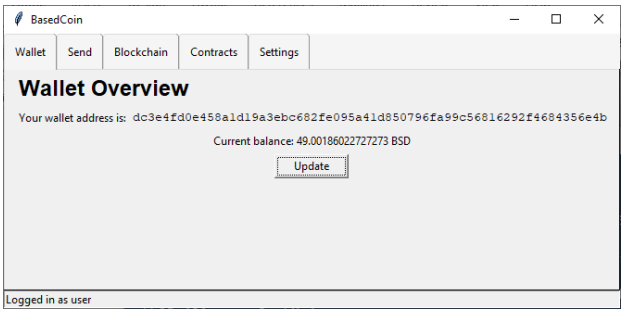

Fig. 2. The Wallet tab.

b) Send: The Send tab allows users to send currency to other users, based on their wallet address. They will specify to whom they are sending, the amount of BasedCoin to send, and if they choose, a message. The application checks if a receiving address has been entered. If it has, it then ensures the amount being sent is a number greater than zero. The information about the transaction is then sent out to the network. This also allows clients to go offline, return, and still receive the sent currency.

Upon sending some BasedCoin, the user is presented with information about the block that was just mined, such as the beginning of its hash and how many tries it took to find the correct nonce. The previous block's hash is then displayed in a subsequent window.

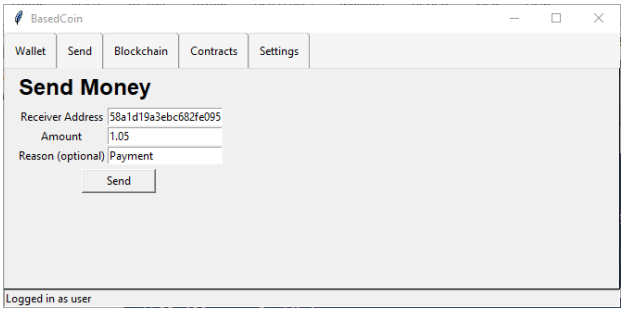

Fig. 3. The Send tab.

c) Blockchain: The Blockchain tab provides the user with an easy to navigate representation of the current blockchain. The most recent block number is listed, alongside a preview of its hash. The left and right arrow buttons allow the user to navigate through the chain, stopping at the genesis block and most recent block, respectively for the left and right sides. The Refresh button will check for any updates to the blockchain, as will switching to the tab from somewhere else in the application.

As User A sends money or executes contracts, User B can refresh the Blockchain page. The new blocks in the chain will then appear. This demonstrates the connectivity between multiple application clients.

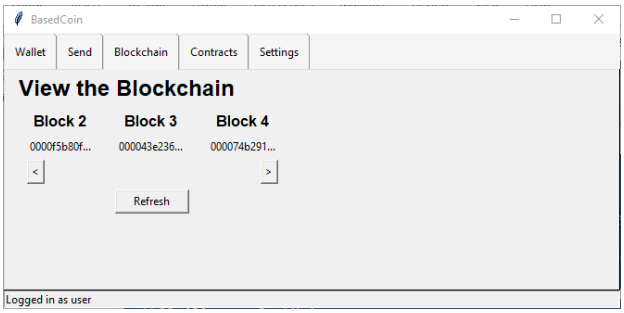

Fig. 4. The Blockchain tab.

d) Contracts: The most interesting aspect of the interface is the Contracts tab. From here, the user can choose their desired contract to execute. The contract's name is displayed on the page and clicking Apply will turn the filename bold. Finally, clicking Execute in the Control area will add another block to the chain and use a small amount of currency. The output of the contract is displayed on the right side of the window.

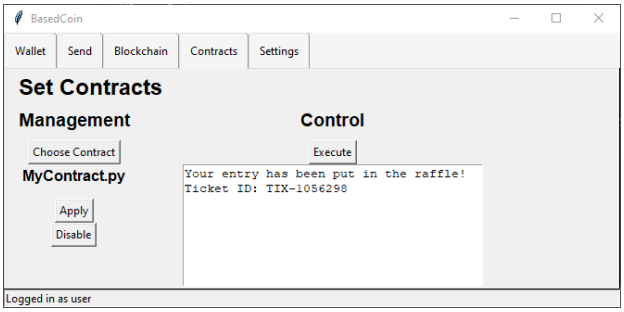

Fig. 5. The Contracts tab. The user has MyContract.py applied and has already executed it.

B. Differences from Existing Systems

BasedCoin was created to give users the freedom of running their own local blockchain. It is not intended to be used as a large-scale solution, as that goes against what the application was created for. However, it could be done if the user so desired. With the combination of the local blockchain and a small set of users, this blockchain implementation is more intimate and suited to the user's needs.

A key feature that sets BasedCoin apart from other cryptocurrencies is its implementation of contracts. The user can select Python programs as entire contracts, rather than using individual functions of a program (such as is the case with Ethereum). In this way, the user is provided more freedom with their contracts. Support for popup windows, such as a graph, are supported with BasedCoin contracts. This flexibility should prove as a key differentiator from similar services and provide novelty.

One key aspect of the design of the application is how it handles new blocks. Traditional cryptocurrencies have employed a Merkle tree to store bundles of transactions in a

space-efficient manner. Due to the intended small-scale implementations for this cryptocurrency, the decision was made to add each transaction to the chain individually. Instant feedback for the user was critical to development, as general use of the application would not warrant enough transactions in some time period t to meaningfully create a Merkle tree and add this to the chain.

C. Application Testbed

This application was written with Python and run with version 3.9.2 or later. It was equally tested on Windows 10 and MacOS Mojave or later. Required libraries are included in standard Python installations. The client connectivity is currently limited to applications running on a single machine, however with port forwarding and a modification to the source files, it can be done quite easily. For the application prototype, it was determined to be out-of-scope for the project. However, with the appropriate implementation, there is nothing that would prevent BasedCoin from being cross-platform.

III. User Manual

A. Setup

The BSD Server must be launched before any clients can start. This is so the application instances can communicate with one another. After the server is launched, which should require no extra user input, the BasedCoin clients may be ran. The following commands will start the server and client(s), respectively.

```
$ python3 BSD_Server.py   #Server
$ python3 Main.py         #Client(s)
```

B. Logging In

There are currently two username and password combinations that will grant access to the wallet portion of the application. Enter either pair in their respective fields and click "Log In."

- user pass
- user2 pass2
- user3 pass3

C. Using the Wallet

Navigation between pages can be done by clicking the respective tabs at the top of the application window. The Wallet tab only offers one user interaction in the Update button, which queries for account balance changes. This functionality is mirrored when the user navigates to the Wallet page. Otherwise, it is merely informational.

The Send Tab allows the user to send money to a desired recipient. Inputting the wallet address of another client will transfer funds to that client's account. The amount to send must be a number greater than zero, and it may not be more than the current wallet balance. Clicking Send will initiate the transaction.

The Blockchain tab provides insight into the active blockchain. Clicking the left or right arrow buttons will move backwards or forwards in the blocks. The Refresh button will check for any updates to the chain but navigating to the page will automatically refresh it as well.

Contracts can be selected in the Contracts tab. The Choose Contract button prompts the user for a Python file. Clicking Apply will make the selected contract active. It can then be executed with the Execute button. Output from the program is displayed in the text box on the page. Clicking Disable will not allow the contract to be executed.

The Settings tab is a placeholder for future settings, such as a server IP address. Right now, the user can toggle if the Settings page is blue or not.

References

[1] R. Browne, "More than $90 million in cryptocurrency stolen after a top Japanese exchange is hacked," CNBC, 20-Aug-2021. [Online]. Available: https://www.cnbc.com/2021/08/19/liquid-cryptocurrency-exchange-hack.html.