BurgerCoin:

An Ethereum blockchain based micro-economy

Final Report

CS467 Online Capstone Project
June 8, 2018

Pedro Torres-Mackie | Ted Sanjeevi | Andrew Lodge

Introduction

Blockchain technology has emerged as a contestant to revolutionize data. Computers process information, and this information is stored somewhere. This is true for your bank account (the running ledger of your financial transactions), your healthcare record (the running ledger of your medical events), your video game account (the running ledger of your gaming successes), and your DMV record (the running ledger of your interactions with the Department of Motor Vehicles). These are all data, about you, stored on somebody else's computer..

These are also all examples of "centralized" applications. There is a central repository for each of these records. Writing this project at Oregon State University in Corvallis, OR, USA, a hypothetical student will probably have the following centralized records:

- Wells Fargo Bank (or some other equivalent)
- Providence Hospitals
- Local Doctor's Office
- Oregon DMV
- US Treasury Department
- Xbox One
- Facebook

There are a few functions each of these central organizations do:

- Store our data
- Protect our data
- Charge a fee
- Disseminate our data for purposes ancillary to our own, but necessary for the functioning of the central organization. Think here about the hospital sending medical information to the insurance company or Facebook selling advertisements.

But what happens if we can't trust the central organization? What if they get so big that their fees become prohibitive? What if they sell our data to people we don't want to access it? What if we live in a dictatorship where the local economy is corrupt and a local farmer cannot make transactions without being robbed, but cannot live without making transactions?

Sociologically, blockchain is a new way of storing data and managing the trust that's necessary to exchange it. One of the most exciting things it's doing is establishing a way

for person's under the domination of controlling regimes to exchange value and currency in a way that is not under the watch and control of the corrupt.

Technically, a "block" is a set of transactions recorded during a short time interval. At the end of this time interval, this block is stored on each computer hosting the blockchain, and another block begins. This subsequent block, the record of transactions during the next time interval, is linked to the first, a third one begins, and the sequence continues. Previous blocks are immutable and cannot be changed. This sequential "chain" of linked, immutable blocks makes the blockchain.

One of the biggest security features of blockchain is that it's stored on every computer involved in mining operations. If there was one copy of the linked list of blocks (a centralized ledger), it's conceivable that a clever hacker could break in and adjust it. They could, for instance, erase the record of the transaction in which they paid you 5 Bitcoin. This will have the net effect of stealing 5 Bitcoin from you. The brilliance of blockchain is that the ledger (the sequence of blocks that is the history of all transactions) is stored on every mining computer. It is not conceivable that a hacker could erase the same transaction on all 500 computers in the time it takes to begin the next block.

Our BurgerCoin project is an exploration of blockchain technology from both a business and technological perspective. We wanted to see what you could do with blockchain and how you could do it. To do this, we created a cryptocurrency (BurgerCoin) on the Ethereum blockchain. We set up a mock economy in which 500 BurgerCoin is worth 1 burger, and the rewards for purchasing a burger is 100 BurgerCoin. This token economy is a microcosm of any other economy, and managing its technical implementation was the model we used to study blockchain's practical implementation..

Setup & Usage

A: Set up Metamask on Rinkeby

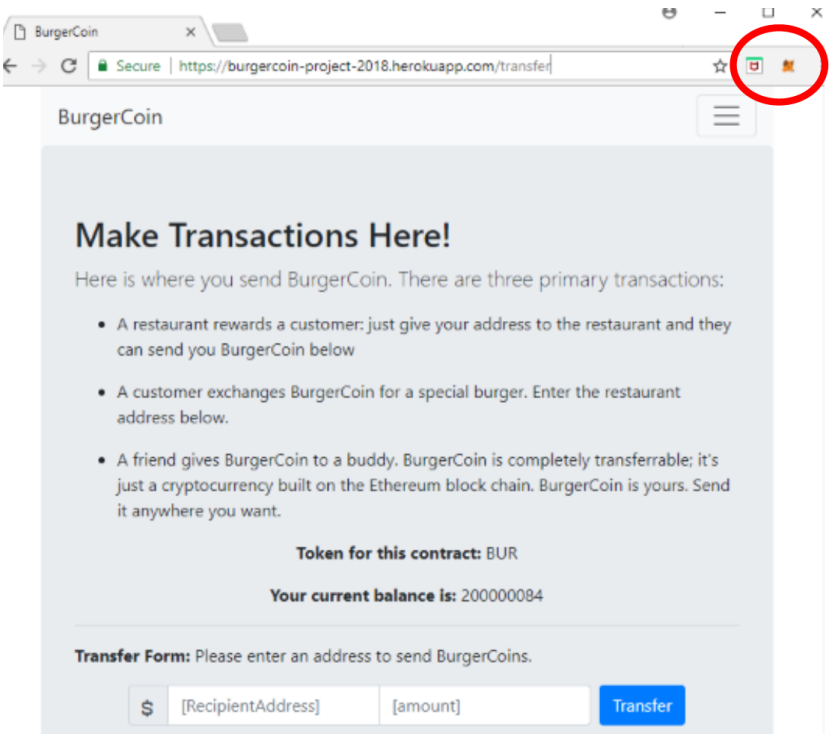
In order for a user to make transactions on the blockchain, he/she needs to have a wallet to store currency and a means of spending it. There are several Ethereum wallets available, and the one we recommend is MetaMask.

1. Go to https://metamask.io/
2. Click on the link to download the adapter for your browser of choice (it will detect your current browser)
3. Click to install MetaMask to your browser and give it permissions.

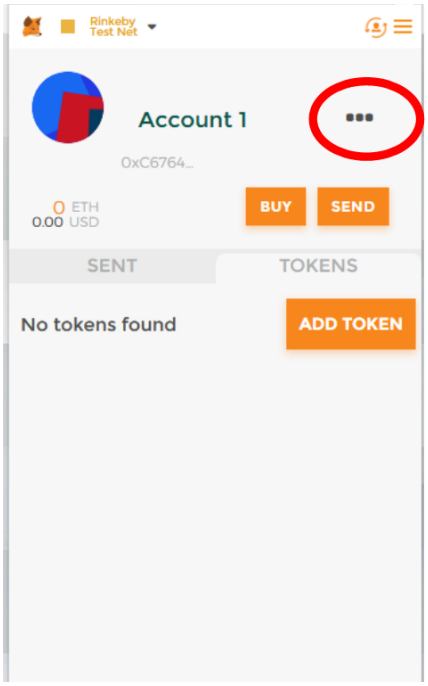
4. Select a password. You will be given a passphrase. Write this down somewhere secure. You account is only an address, password and passphrase, and balances. There is no password recovery. If somebody takes your password and cryptocurrency, there is no recourse. There is not going to be a link to reset your password.
5. Click to add tokens and add our project address, 0x5c87e1c456dcedfab139df81a7ca331bb0c14dda
6. At this point, you are watching your balance of BurgerCoin (BUR) and your balance will be 0
7. Finally, you are going to need your address, your “public key”. This is the identifier you will use for every transaction on the blockchain. Each transaction will appear to every node on the chain. All that will appear are the sender, receiver, and amount. In MetaMask, click the bold ellipses next to your account name, and select “View Account on Etherscan”. This will bring up a page with lots on interesting information types about your account. Since you just made your account, there won’t be much of it yet.

If you click elsewhere on the screen, your MetaMask interface will minimize. There are two very helpful links associated with your wallet to check the status of your transactions.

First, as shown in the figure below, you can get your wallet interface back with the fox icon on your browser.



A second very useful link is the ellipses next to your account name. This will take you to Etherscan where you can view the history of your transactions. During development this helped us isolate problems or transactions that may have been rejected. During use, it can be nice to check the status of a recent transaction. Blockchain transactions do take some time to process and it can be easy to wonder, "Is it working"?[1]



B: Get ether on Rinkeby with the ether faucet

In order to purchase BurgerCoin and execute transactions, you are going to need some Ether. We are hosting our school project on one of the test networks, so we are not dealing in actual currency, but rather test protocols. You can get test Ether through a "faucet". Go to https://faucet.rinkeby.io/ and follow the instructions. They will instruct you to post your address on a public social media post and copy the link to that post. At first, posting the address of your wallet publicly will feel strange. However, this is how blockchain works. Your public key is always visible on the blockchain when you make a transaction. It's your private key that makes things secure. This method does associate you with your address (public key), but this isn't a problem if you typically make transactions with people you know anyway.

C: Buy BUR with ether

(This step is optional; you can participate in the BurgerCoin economy without it.)

[1] Browsing on the Etherscan of your contracts, your tokens, and your tokens' contracts is actually a pretty good way to get a feel for what is happening with blockchain transactions.

With Ether in your wallet, you can now buy some BurgerCoin. Visit the "Join" tab of our web page, https://burgercoin-project-2018.herokuapp.com/join, and enter an amount of ether you'd like to spend on BurgerCoin. Currently, the amount of Ether has to be greater than 1. (1 Ether won't work, but 1.00000001 will).

When you submit this transaction, your MetaMask wallet will pop up and ask if you want to approve the transaction. Approve it, and then wait for the block to be mined for the transaction to go through. This often takes about a minute or two.

** Note ** you have to be signed into your wallet and have your wallet set to the Rinkeby test network for this to function properly. The transaction will not process if you aren't logged into your wallet or if it is pointing to a different network.

E: Transfer BUR to other addresses

This is the primary functionality of the smart contract. To send BurgerCoin to someone, head over to the transfer page https://burgercoin-project-2018.herokuapp.com/transfer . The first thing to do is enter the address (public key) of the person you want to send BUR to. Copy / paste is best here. Addresses are long and easy to get wrong. Then add the amount to send and submit. You'll be prompted to accept the transaction, and then it'll take about a minute or two to process.

To test this, you can use any external ethereum address as the recipient, and then check the etherscan.io page of the BurgerCoin contract to see that the transfer is recorded there. Our etherscan page is here:
https://rinkeby.etherscan.io/token/0x5c87e1c456dcedfab139df81a7ca331bb0c14dda

Also, here is an ethereum address you can use as a recipient to test transfers:
0x14ae8100Ea85a11bbb36578f83AB1b5C1cFDd61c

This transfer function is the same whether a patron is sending BUR to a restaurant or a restaurant is rewarding a patron with BUR.

Software Systems

Ethereum.

Ethereum is the blockchain platform we used to develop our project. It is the foundational technology on which our project is built. It can be thought of as the database for our project because it's where the data ledger is kept, but this data is distributed through many nodes and is immutable. In their own words:

Ethereum is a decentralized platform that runs smart contracts: applications that run exactly as programmed without any possibility of downtime, censorship, fraud or third-party interference.

These apps run on a custom built blockchain, an enormously powerful shared global infrastructure that can move value around and represent the ownership of property.[2]

Solidity.

Solidity is a contract oriented programming language. It writes the rules and parameters for the transactions that constitute blocks. It was initially designed to work in conjunction with the Ethereum blockchain. When it describes how to write smart contracts, the Ethereum Project's web site directs developers to Solidity. The relationship between Solidity and Ethereum is close enough that we considered describing them as one software system. There are, however, several contracting languages that are compatible with Ethereum, and Solidity is likewise compatible with several other blockchain platforms.[3]

Metamask.

"MetaMask is a bridge that allows you to visit the distributed web of tomorrow in your browser today. It allows you to run Ethereum dApps right in your browser without running a full Ethereum node."[4] Blockchain apps are inherently transactional. One of the main benefits of blockchain is security - I am in control of my own information even though it's stored on every node. Metamask is an adapter. It stores my address ("public key"), which on the blockchain is the thing associated with my currency. It stores my password ("private key"), which is required to complete transactions. It's an interface - when transactions are attempted on my behalf, it allows me to accept / reject them. From a user's standpoint, it is like a wallet. From a developer's standpoint, it is an interface between the browser and the smart contract.

Node.js. & Handlebars

We made extensive use of Node.js, Handlebars, and JavaScript to create the webapp and user interfaces. As detailed in the "Deviations from Original Plan" section, our app took a couple of turns as we discovered the best way to interact with the blockchain. During some the adolescent phase of our project, Handlebars' templating features were

[2] https://www.ethereum.org/
[3] https://en.wikipedia.org/wiki/Solidity
[4] https://metamask.io/

more helpful. The final implementation of our app could have been done with a simpler structure, but we had built that function into our app and leaving it there readies us for adding complexity later.

Heroku

Heroku is the webapp hosting platform we chose. Two of the notable benefits of this platform were linkage to Github and ability to host different file structures. Shortly after our midpoint report we decided to switch the basic website structure from a React.js app to a Node.js app. All we had to do was link the web site url to a different Github repository, click one “Deploy” button, and we were swapped. The service was both user friendly and robust.

Additional Tools, APIs, Libraries, etc…

Github

We made extensive use of Github during the semester. We made over 500 commits to the repository that ultimately drives our web page. The Gist feature of Github was predictably helpful at several turns, as was the ability to checkout each others’ repository for a current state of affairs.

Browserify

We needed to use the require() function to import ethjs libraries into the front end application. Since MetaMask and other web3 adapters make calls to the blockchain from the browser, some function calls have to be made from the client side of the app rather than the server side. Browserify is a good adapter to make these function calls (notable require()) available.

Iconic

Our webapp is not graphically intensive, but judicious use of icons makes the user interface more intuitive and obvious. The iconic library has the right amount of function for this.

Ethjs

Ethjs is a JavaScript library designed to work well with Ethereum. It is somewhat light weight and is designed to be async only. This specifically asynchronous functionality helped us solve one of the most time consuming hurdles of the project - how to make calls to the blockchain, wait for authorization, and wait for a response. Many other

approaches would either create errors in the web3 interactions or blow by the request for authorization by refreshing pages. The Ethjs library solved these elusive challenges.

Google Hangouts & Google Docs

The vast majority of group communication was via Hangouts. This was a simple yet versatile tool that let us communicate from our phones while about our lives and from our computers coding together. It was also nice to periodically be able to toss a picture or diagram into the chat and have it be visible to the others on both phones and computers. Google Docs was likewise an effective tool for writing reports together and sharing diagrams.

Teammate Contributions

This team worked well together and adopted an effective division of roles and responsibilities.

Roles

Pedro first conceived of the idea of BurgerCoin and was the architect of the project. At several points, his ideas gave direction to the course of our development. Ted adopted the role of technical problem solver. On several occasions, we were stuck with an annoying problem and he researched clever solutions to get over us over hurdles and moving forward. Andrew served as writer, working to implement the above solutions, create draft reports, and research and develop solutions to problems getting the web app to communicate with the blockchain.

Responsibilities

Pedro wrote the smart contract to create BurgerCoin, created the basic structure of the web page, drew the basic diagram for the back end data structure, and did significant research on how to use and apply blockchain technologies. Ted researched and wrote several problematic implementations. Notably, he discovered and wrote the implementations for our back end to send emails, our contracts to exchange tokens, and our app's ability to receive Ether in exchange for tokens. Andrew developed the centralized database and back end handler to store seeded accounts, developed much of the web app's interactions with both the blockchain and centralized database, and drafted initial reports.[5]

[5] Workload is reported here that does not appear in the final product since we intentionally discarded our email verification and centralized database.

In addition to these more distinct roles, each member of the group spent extensive time studying blockchain, how to use it, host it, implement it, and interact with it. Much of this research was doubled up. For example, we each had to learn how to create Ethereum sensitive accounts and manually interact with them. Much of this work was foundational and done during the earlier phases of development. There was lots of sharing and helping during these early times of parallel development.

Deviations from Original Plan

Adjustments from our original plan primarily revolved around the back end of our application. We had to discover which pieces should be “decentralized” and which pieces should be “centralized”. Smart contracts store information on the blockchain and are decentralized. Often, applications connect to information in a “centralized” database that is stored on either a central server or a few servers controlled by a central organization..

At the point of the initial project plan, we had not decided how much information would be stored on the blockchain vs how much information would be stored in a centralized database.

By the midpoint of the project we were working on a hybrid app. Some pieces of information would live on the blockchain, such as addresses, account balances, and transactions. Other pieces of information would live on a standard, centralized database. We were going to seed accounts with a predetermined number of tokens to get the BurgerCoin economy started. The centralized elements of this hybrid structure involved email verification to ensure potential users were valid as well as a database to store email addresses already associated with free tokens.

During the late phases of the project we realized that storing all the information on the blockchain was a more simple and elegant solution. Blockchain excels at security and we had begun to manage security through different, less-than-ideal means. Instead of seeding accounts with free tokens, we allowed users to purchase tokens with Ether. There would be no free tokens, and therefore no need for a database to record who had needed them. New users could just create an account and start receiving BUR payments a little at a time, or they could seed their own accounts in an Ethereum-for-BurgerCoin transaction. This new structure is illustrated below.

This blockchain-only data structure immediately seemed like the right solution. It was a bit sad to let go of portions of the project that we had spent lots of time on, but that at this point, these portions did not fit. We got rid of them. These hours are not self-evident in the finished project because we scrapped their fruits. Since, however, one of the goals of the project was to understand what blockchain can do and what its

merits are with respect to centralized applications, we feel like the progression of this time expenditure was consistent with our goals and helped us get a feel for where it fits.

Future Plans & Project Continuance

The next pieces of the app we plan to develop are a registry contract, an ERC721 token to represent special burgers, and a directory of participating restaurants.

Registry Contract

Currently, there is no distinction in the smart contract between a user and a restaurant. Both are abstractions that simply refer to a public key that is associated with a value (the quantity of tokens). A registry contract is a transaction - send a request to the smart contract and the smart contract will flag the user as a restaurant.

ERC721 Token

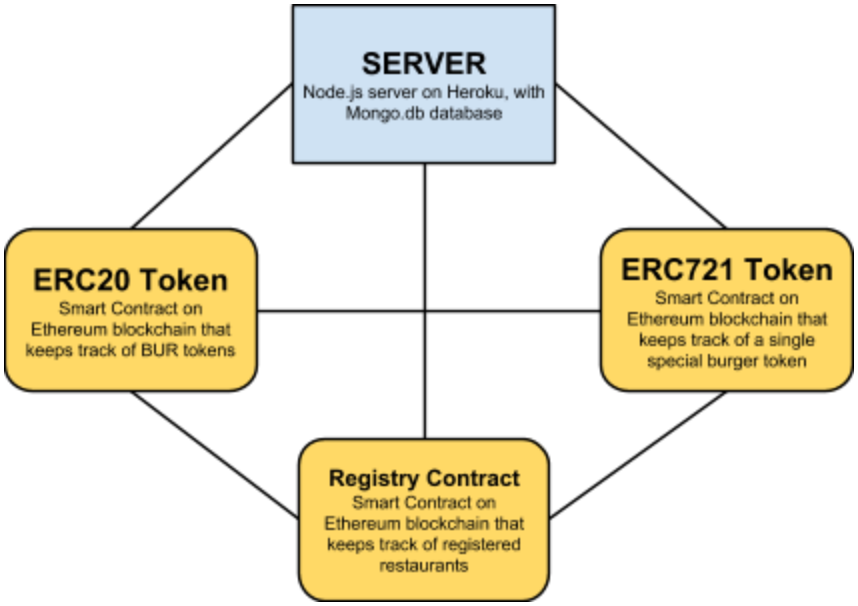
Once registered as a restaurant, a restaurant-user will be able to mint non-fungible tokens of a different variety (ERC721 instead of ERC20). Fungibility means interchangeability. A dollar is interchangeable with another dollar. A pet cat, however, is not interchangeable with a different pet cat.[6] BUR is interchangeable with other BUR. We will allow restaurants to mint a special token for each burger they offer, which will then become a record of earning a reward burger at a certain restaurant.

Restaurant Directory

While restaurants can certainly advertise that they are participating in the BurgerCoin ecosystem, it would be great to have a directory on our website of all restaurants that accept BUR and the special burgers they offer. This would be something we'd want to develop once our ecosystem is more mature, and there are more restaurants participating. This could also go hand in hand with both new contracts described above, as the information for participating restaurants, as well as their special burgers (represented by ERC721 Tokens) could all be stored on the blockchain as well.

With all these pieces implemented, our architecture would look like the following diagram:

[6] Currently (6-7-2018) electronic collectible cats (CryptoKitties) are one of the most common ERC271 non-fungible tokens.



Conclusion

This project has been an incredible introduction to the world of blockchain and decentralized application development. Launching a new token on the Ethereum network required a large research effort to educate ourselves on the nascent best practices around developing decentralized apps, the standards currently being proposed, and the important differences between architecting a traditional centralized client-server application, versus an application that runs on a decentralized blockchain like Ethereum. We strongly feel our team is coming out of this experience with a unique skill set and understanding around this new industry that will be an asset to us going forward.

While there have been many efforts made by the developers of Ethereum and others to make decentralized application development easier and more streamlined, and it is true that one could develop a basic coin and launch it in under an hour, creating a good working system on the blockchain that is easy to interact with is still very challenging. The main roadblock to this, is that the technology is still very early in its development, so most tools and development frameworks are buggy and in early version still. Moreover, blockchain apps are not readily supported by major browsers, which means the user experience ends up being clunky because we have to use a mix of plugins to achieve our goals.

When we started this project, we underestimated how much work would go into the blockchain development portion, and especially into making the blockchain back-end work with our centralized server and web front-end. Because of this, we had to make some significant changes to the architecture of our project, which meant letting go of code that we had spent weeks to develop. Despite this, we were able to learn a lot about many blockchain protocols,

and launch a few different versions of our BUR token until we landed on our current implementation in the Rinkeby Ethereum test network.

Developing BurgerCoin has been a challenging yet rewarding experience, and we've thoroughly enjoyed working together as a team, and learning about this exciting new technology. We also feel the concept of BurgerCoin is a great case study for a blockchain-based rewards program that could have real-world applications, so we hope to see similar concepts and systems brought to market in the future.