Ethereum blockchain Twitter

Brent Abad*
Cal Poly Pomona
Pomona, CA, USAA
baabad@cpp.edu

Andrew So*
Cal Poly Pomona
Pomona, CA, USA
acso@cpp.edu

Milush Yanev*
Cal Poly Pomona
Pomona, CA, USA
myanev@cpp.edu

***Abstract*—blockchain is a specific type of database that stores data in blocks that are chained together. As new data fills a block, a new block is created to chain in chronological order. The most common use of a blockchain is a transaction ledger. In today' s society where censorship and dishonesty is a problem on social media, a decentralized way for people to communicate with each other on the open web is an ideal way to keep people honest. In this paper, we explore the possibility with a prototype concept build of a Ethereum blockchain powered Twitter with Truffle and Ganache as the engine for our design. In our experimentation, we show that our design concept can mimic a black-chain that main goal is preserve message chains.**

***Index Terms*—blockchain, Decentralize, Ethereum, Social Media**

I. Introduction

Block chain technology has been talk of the town since the proliferation of bitcoin. Many more cryptocurrencies has come out to take advantage of the economical benefits of block chain.However, the technology can do more than cryptocurrencies as the internet landscape changes and people do more of their business online, users demands more security and privacy, as companies or government tend to use users' data for data mining for information or profit. Users today seek decentralization, as decentralization is seen as an answer from the hands of the entities that controls almost all of the internet. Many platform has emerge to enable developers to easily create block chain applications, some pf which are used for these project.

Ethereum. Ethereum is a block chain platform that provides a way to power decentralized applications. Ethereum uses ether to run its application and ether is the second popular cryptocurrency, next to bitcoin. Ethereum provides security for its block chain network by employing a proof of work system. Proof of work will allow block chain nodes to attach contract if they could solve certain problems, where in turn means the node has to spend resource, also since nodes also has copy of the block, it ca be verified by other nodes, therefore, attack will take a large amount of computing resource.

Twitter. Social media has seen a boom now more that ever, and as the industry raked in millions, many applications has popped up with many forms and schemes to attract users. Twitter is one social media application that allow users to post with only 280 characters. Twitter and other mainstream social media has still used centralized databases for their applications and because of this system, the system is susceptible to attack and censorship.

⋆ denotes equal contribution to this work.

Problem. The censorship has become one of the issue today on social media, as the medium is now used not only by teenagers but by political and world leaders. Because of the high profile users the issue of censorship has become a hot topic. The project aims to solve the problem using block chain technology, the security and the ability of Ethereum to provide decentralization can give the solution any regular social media needed.

II. Preliminaries

For any Ethereum application, certain programs are needed for testing and migration to the block chain for a successful application development.

A. Solidity

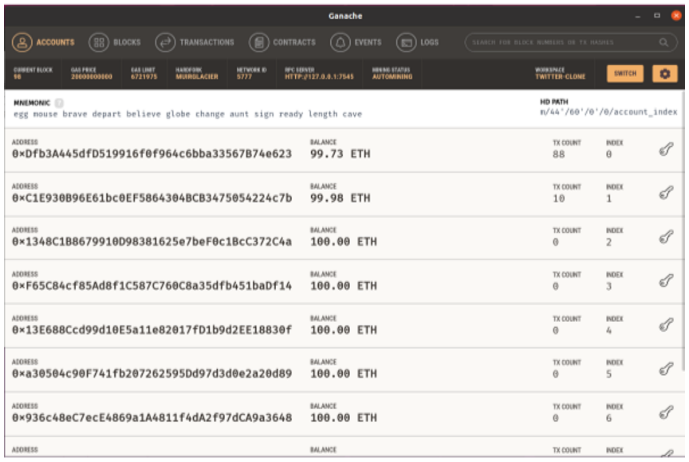
Solidity is an object-oriented, high-level language for implementing smart contracts. Smart contracts are programs which govern the behaviour of accounts within the Ethereum state. Solidity was influenced by C++, Python and JavaScript and is designed to target the Ethereum Virtual Machine (EVM) [1]. Solidity is used to write the smart contracts that will run in Ethereum platform. The smart contract are immutable codes that tells the block chain on how the application acts. The immutable nature of a smart contact ensures that, every contract that has been and will be migrated to the block chain are free from alterations in case an attack happens.

B. Truffle

A world class development environment, testing framework and asset pipeline for blockchains using the Ethereum Virtual Machine (EVM), aiming to make life as a developer easier [2]. Truffle manages compilation, testing and rapid deployment of smart contracts. Node Package Manager (NPM) handles the package management

C. Ganache

Ganache is a personal blockchain for rapid Ethereum and Corda distributed application development. You can use Ganache across the entire development cycle; enabling you to develop, deploy, and test your decentralized applications in a safe and deterministic environment [3]. Ganache create virtual users for the developers to use to act as user nodes. The virtual users are given ether for them to be able to interact with the



CURRENT BLOCK	GAS PRICE	GAS LIMIT	HARDFORK	NETWORK ID	RPC SERVER	MINING STATUS	WORKSPACE
98	20000000000	6721975	MUIRGLACIER	5777	HTTP://127.0.0.1:7545	AUTOMINING	TWITTER-CLONE

MNEMONIC
egg mouse brave depart believe globe change aunt sign ready length cave

HD PATH
m/44'/60'/0'/0/account_index

ADDRESS	BALANCE	TX COUNT	INDEX
0×Dfb3A445dfD519916f0f964c6bba33567B74e623	99.73 ETH	88	0
0×C1E930B96E61bc0EF5864304BCB3475054224c7b	99.98 ETH	10	1
0×1348C1B8679910D98381625e7beF0c1BcC372C4a	100.00 ETH	0	2
0×F65C84cf85Ad8f1C587C760C8a35dfb451baDf14	100.00 ETH	0	3
0×13E688Ccd99d10E5a11e82017fD1b9d2EE18830f	100.00 ETH	0	4
0×a30504c90F741fb207262595Dd97d3d0e2a20d89	100.00 ETH	0	5
0×936c48eC7ecE4869a1A4811f4dA2f97dCA9a3648	100.00 ETH	0	6

Fig. 1. Ganache Screenshot

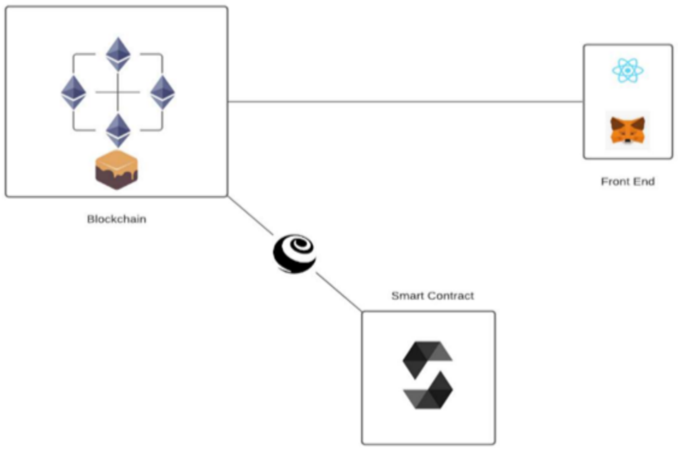


Fig. 2. System Architecture

block chain as shown in Figure 1. Although cryptocurrency is needed for users to use any application in Ethereum, ether is not necessary to install Metamask. Auto mining is a function of Ganache, although not use in the project, the auto mining function can be used to simulate an interaction between the users and the effect of miners in the application.

D. *Metamask*

Metamask is a browser add that enables access to the Ethereum block chain. Since the ethereum is another type of network, Metamask is needed for users to interact with Ethereum. Aside from being a gatekeeper, Metamask acts as an ether wallet for the user. Other Metamask functions are coin exchanges as multiple coins has flooded the market, Metamask could also facilitate coin swapping. The function could ease the coin swapping block chain users do as it cuts third party applications and do the process on Metamask.

III. System Overview

All three programs are used to construct an twitter-clone block chain application, by having the three component work together, we created a twitter clone application that let users post a message or tweet and be read by other members and also tipping where if another user like the post the other user could give the post's user some ether. In our Figure 2, we depict how each component is connected and interacts with one another.

A. *Front End*

The front end or user interface of the application is powered by React JS and Metamask. React will be used to create the user interface for the browser for the user to see the posts in the network. Metamask will be connected to the user's browser, once downloaded to enable connection to the network. The UI will facilitate the the user's action to the block chain and once the user has send the post or like a post, Metamask will deduct the ether from the users wallet and will continue to the next portion of the system as shown in Figure 3. The user interface could be change in case improvements has been added to the network.

B. *Block chain*

After the user has sent the information from the front end. Data will now go to the block chain, where in the case of the projects is Ganache, as it represents the nodes for the block chain. Ganache will deduct or add the necessary ether to the users. It is important to note that the Ethers are all available on the developers block chain and cannot be use outside the network.

C. *Smart Contract*

Smart contract is the building block of the block chain. Smart contract are deployed per post or like of a user and it carries the transaction that was done with the block chain. The process of migration and testing of smart contracts will be handled by truffle. For the testing program, the project uses java script to create code to test whether the smart contract is acting the way the developers desire. Rigorous testing of smart contract is essential for block chain development due to immutable nature of the contract.

IV. Implementation

In our implementation, we use Ganache with Truffle implement our server which interfaces with Metamask. With Truffle and Ganache, we host our Ethereum-based Twitter on a local host. First step is to create the Smart Contract solstice file. In order to do so we are using Truffle development environment. When the contract is created we have to run a local server, obtain the Ganache address and deploy it. A proper Migration must be executed upon deployment. Last step is the creation of a Metamask, addressing the right IP and Network ID, and add it to the browser extensions. Without a proper configuration there will be no way of getting and displaying the Ethereum wallet values. The Metamask is used as a virtual wallet. Last step is to connect everything together and test the User Interface for functionality. In order to pass, our service must be able to create a Post from multiple users, display their

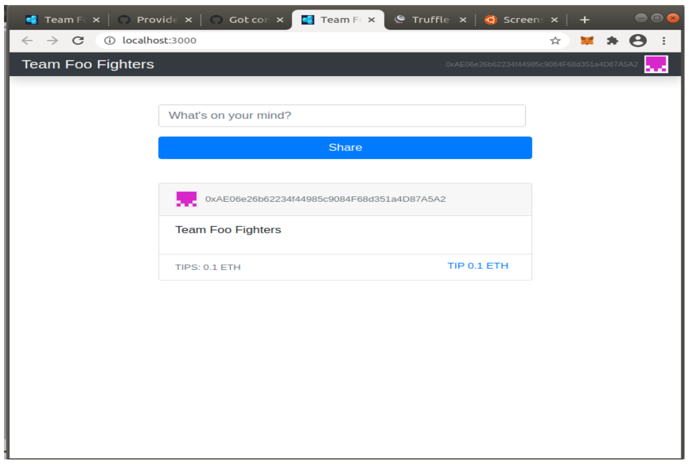


Fig. 3. User Interface Screenshot

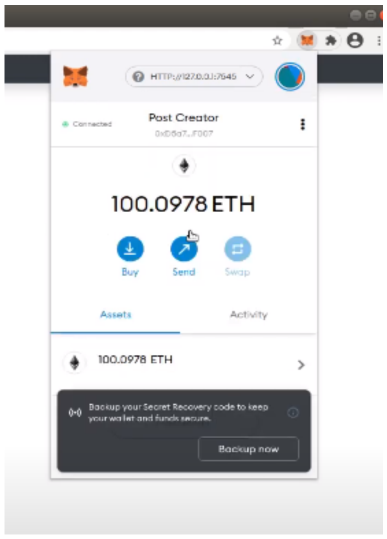


Fig. 4. Metamask Wallet

Ethereum Hash block address and be able to tip the creator a current post.

V. Experiments

In this section, our experiment demonstrates that our proof-of-concept work. We use a standard Ubuntu as the base conditions to run the Ethereum server for this paper. Important snips of code used listed in Appendix 5, 6, 7 and 8.We used references on YouTube to help developed our project from scratch.[1] [2]

A. Setup

For our PoC (Proof of Concept) blockchain server, we use Ganache with Truffle to run the Ethereum server and it's smart contracts. Each account is created on Ganache and is initialized with 100 ether to start off. We associate Metamask with our server to illustrate the contents of our blockchain code as shown in Figure 4. The Metamask wallet holds all the accounts used in the demonstration.[3] The user interface in Figure 3 will cost about 0.02 ether to post a message and a new smart contract onto the blockchain in our current setting. This value can be adjusted for future implementations to based off of different currencies.

The tipping feature is our implementation of 'likes' for a Tweet. The tipper will have their account deducted by 0.1 ether and have the value transferred to the original poster. Unfortunately, we never got to implementing the retweet feature as we would explain in our Security Analysis section.

B. Security Analysis

In the following, we explore the theoretical issues that we may face if we implemented this project on a larger scale: retweeting, proof of stake, Sybil attacks and other attacks.

Retweeting. For retweeting a post from the blockchain, we noticed that this will be an issue of how we setup the project as the re-tweeter would also be paying the gas tax to post the same exact message from the previous poster. This is an issue as the re-tweeter become the original poster thus making it hard to trace back the original poster. An idea was proposed to link the message of the previous blockchain to display on the new post. Another idea was to just only re-post the latest post for a re-tweet.

While our ideas could have prevent people from being censored, it limits us on how we store each message and how we read it. We never gotten to the point on how to handle the retweet issue due to technical difficulty and time constraints.

Proof of Stake. In a similar fashion to Ethereum's new standard of "Proof-Of-Stake", we make it so that users must pay ether to interact with our version of Twitter. Normally, most blockchains use proof of work to validate messages in a chain but with proof-of-stake, we only want to be sure that the user has ether on our social network to be able to interact with our posts. Free users would be able to see the posts but not post new messages or tip any messages.

The gas tax that is implemented in each new smart contract provides a way to limit fake accounts from trolling or spreading misinformation to other users. While, we understand this may limit user financially as they would have to pay with ether to post or to like certain messages. The benefits from doing it this way limits the amount of trolls that people would have the face as they would have to pay a certain amount ether to troll a person versus Twitter just allowing anyone to post for free on multiple accounts. This is also to make sure a impulsive user would take a second to think whether they want to post or like something as it cost money from their wallet.

In future implementations, we could make it so that malicious 'stake holders' or posters could loose all their ether in a similar fashion to what Ethereum 2.0 plans to implement in their new standards.

Sybil Attack Prevention. Sybil Attacks have been issue that plagued the online social networks for many years. A Sybil

[1]www.youtube.com/watch?v=nvw27RCTaEw
[2]www.youtube.com/watch?v=CgXQC4dbGUE
[3]www.youtube.com/watch?v=_cNI3jl82NE

attack is when a malicious actor uses multiple fake identities to create network To better handle OSN (Online Social network) attacks, we use the 'gas tax' to prove that the user is "valid" and "trustworthy" in the eyes of our system. We make this assumption that users are not gonna spend thousands of dollars to buy Ethereum to sign up multiple fake accounts.

While we don't have any features to prevent and to detect trusted accounts from being stolen, we are trusting the user to keep their own wallet credentials safe from malicious actors. The act of tipping a post creates transactions and builds up a network of trusted users. In our eyes, we are using money as a "GateKeeper" to keep our online trolls and malicious actors from messing with other users. [2]

Other Attacks. The only other valid attack concern is a 51 percent attack. This attack is when the attacker has taken control of the network by capturing 51 percent of the nodes. The attack is costly since interactions will cost ether and also the amount of computing power needed is very taxing for the hardware, making the attack not worth it.

VI. Related Works

We reviewed related works on block chains from three other applications: Blockchain-Enabled E-Voting, agri-food sector, and Healthcare.

A. Blockchain-Enabled E-Voting.

Securing election results is important for any democracy in any country. Blockchain enabled voting can help combat concerns in today's voting: voter access and voter fraud. [3] Each voter is given a wallet that contains the voter's credentials and is issued a single "vote coin". This allows each vote to counted properly and prevent "double spending" on votes as they are recorded into the blockchain. Like how we addressed it in our project, each poster has a wallet with a limited amount of ether to power each post.

B. Agri-food Sector.

Food security is vital to our everyday livelihood. It revolves on a complex system that distributes agricultural product throughout the world. The benefit of blockchains is that the technology's transparency and reliable applications help expand food tracking operations in today's supply chains. Such applications include applying unique digital identifiers onto food that is written onto a blockchain in which can prevent food waste and provide immutable records for transparency. Like in our project, we want to make it so that users can change their messages out of the blue without an form of records. In the author of this paper's perspective, Information and Communications Technology can be further implemented with a blockchain infrastructure to enable a good flexibility for applications in several sectors. [4]

C. Healthcare.

In healthcare, blockchains act like digital ledger that can better facilitate and secure data. blockchain have the potential to transform healthcare optimizing its business processes, lowering costs, improving patient outcomes, enhancing compliance, and enabling a secure way of sharing healthcare-related data.

The article brings up the point of when to use blockchain in certain scenarios. There is by no doubt that blockchain technologies can nourish healthcare ecosystem with data generation and sharing, from biomedical research in a lab with cell/tissue analysis to insurance payments when care is provided.

The authors further provide a point of view on how blockchains can improve their field of work but also serve as a warning as to how blockchains face challenges in how it is implemented to better serve the patient. [5]

VII. Conclusion

blockchain powered messages are one applications we can use to secure communications to a wider audience. To implement our goal, we had to remove the retweeting feature that is well-known on Twitter. In this work, we produced a prototype that uses Ethereum and Truffle as the engine to handle the blockchain and smart contracts. Although, we didn't manage to replicate all the features of Twitter, it still brings up the feasibility of a blockchain powered social media. According to how we designed our prototype, it helps to prevent censorship and Sybil attacks in certain applications. Overall, this project is a good proof of concept to show the effectiveness of block chain technologies with respect to security and the problem of censorship. The twitter clone has shown that posts or tweets are not erasable or changeable and the network is secured due to the use of block chain. Block chain can provide functionalities that conventional networks cannot give, and the success of Ethereum has only catapulted the use block chains and it can create new ecosystem of application that thrives in the block chain network.

References

[1] [Online]. Available: https://docs.soliditylang.org/en/v0.7.4/index.html

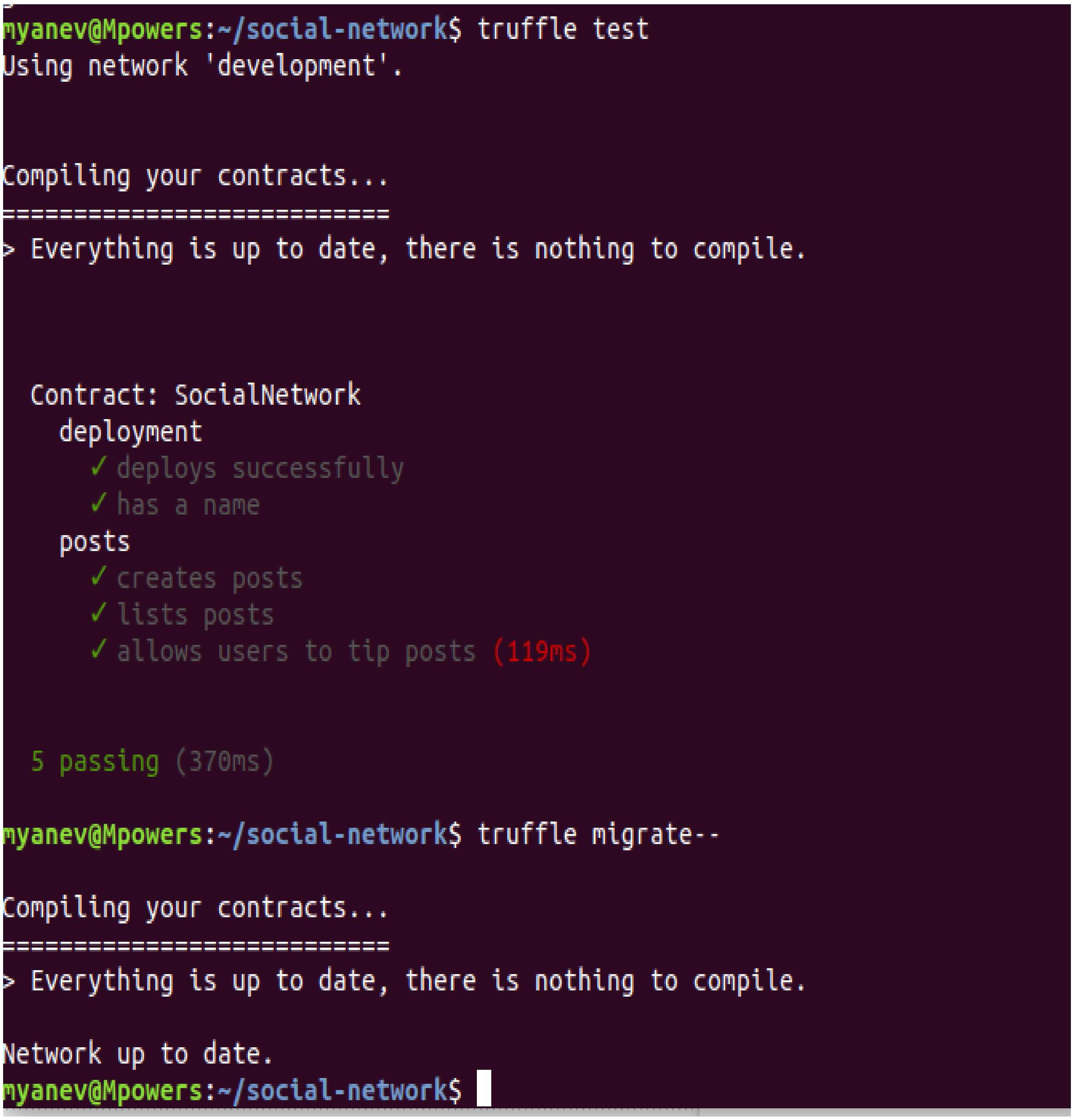
[2] N. M. Shekokar and K. B. Kansara, "Security against sybil attack in social network," in *2016 International Conference on Information Communication and Embedded Systems (ICICES)*, 2016, pp. 1–5.

[3] N. Kshetri and J. Voas, "Blockchain-enabled e-voting," *IEEE Software*, vol. 35, no. 4, pp. 95–99, 2018.

[4] C. C. F. P. L. R. Francesca Antonucci, Simone Figorilli and P. Menesatti, "'a review on blockchain applications in the agri-food sector," 2019. [Online]. Available: https://doi-org.proxy.library.cpp.edu/10.1002/jsfa.9912

[5] K. T. G. B. e. a. Mackey, T.K., "'fit-for-purpose?' – challenges and opportunities for applications of blockchain technology in the future of healthcare," 2019. [Online]. Available: https://doi.org/10.1186/s12916-019-1296-7

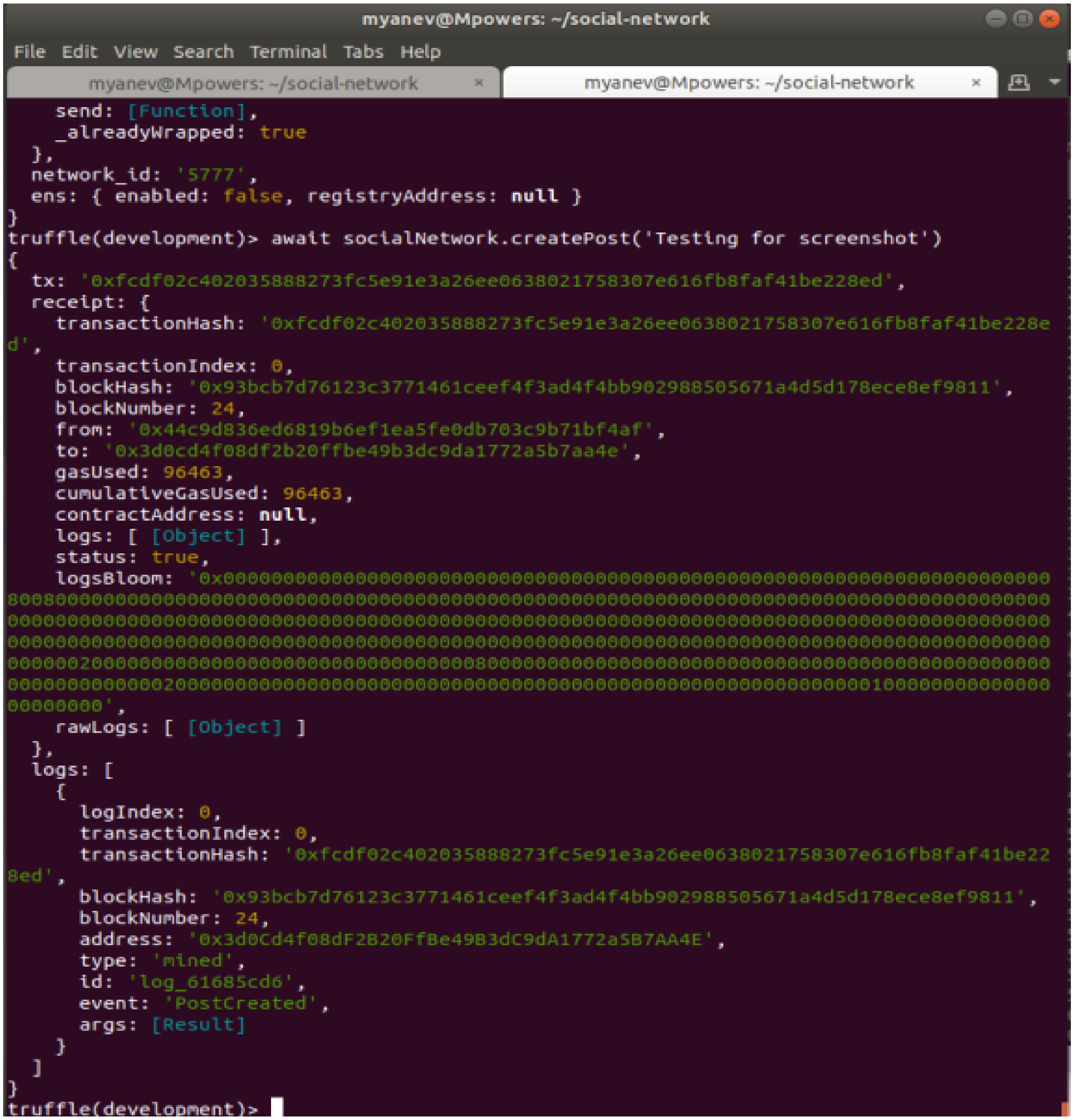
Appendix A
Migration Testing



```
myanev@Mpowers:~/social-network$ truffle test
Using network 'development'.


Compiling your contracts...
===========================
> Everything is up to date, there is nothing to compile.



  Contract: SocialNetwork
    deployment
      ✓ deploys successfully
      ✓ has a name
    posts
      ✓ creates posts
      ✓ lists posts
      ✓ allows users to tip posts (119ms)


  5 passing (370ms)

myanev@Mpowers:~/social-network$ truffle migrate--

Compiling your contracts...
===========================
> Everything is up to date, there is nothing to compile.

Network up to date.
myanev@Mpowers:~/social-network$
```

Fig. 5. Migration Testing

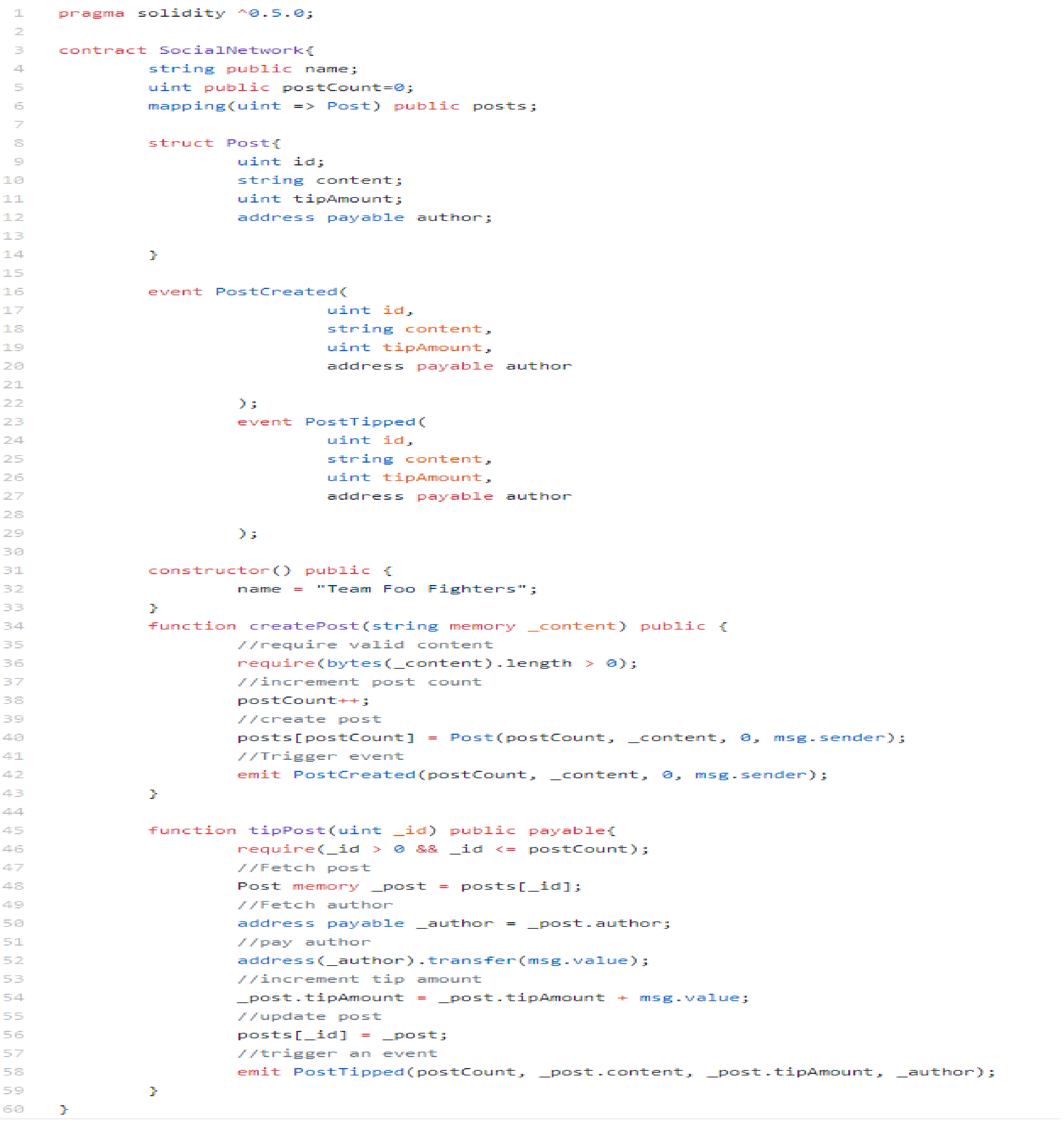
APPENDIX B
PROOF OF CONCEPT


myanev@Mpowers: ~/social-network
File Edit View Search Terminal Tabs Help
myanev@Mpowers: ~/social-network
myanev@Mpowers: ~/social-network

```
    send: [Function],
    _alreadyWrapped: true
  },
  network_id: '5777',
  ens: { enabled: false, registryAddress: null }
}
truffle(development)> await socialNetwork.createPost('Testing for screenshot')
{
  tx: '0xfcdf02c402035888273fc5e91e3a26ee0638021758307e616fb8faf41be228ed',
  receipt: {
    transactionHash: '0xfcdf02c402035888273fc5e91e3a26ee0638021758307e616fb8faf41be228e
d',
    transactionIndex: 0,
    blockHash: '0x93bcb7d76123c3771461ceef4f3ad4f4bb902988505671a4d5d178ece8ef9811',
    blockNumber: 24,
    from: '0x44c9d836ed6819b6ef1ea5fe0db703c9b71bf4af',
    to: '0x3d0cd4f08df2b20ffbe49b3dc9da1772a5b7aa4e',
    gasUsed: 96463,
    cumulativeGasUsed: 96463,
    contractAddress: null,
    logs: [ [Object] ],
    status: true,
    logsBloom: '0x000000000000000000000000000000000000000000000000000000000000000000000
800800000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000200000000000000000000000000000000800000000000000000000000000000000000000000000000
000000000000020000000000000000000000000000000000000000000000000000000000100000000000000
00000000',
    rawLogs: [ [Object] ]
  },
  logs: [
    {
      logIndex: 0,
      transactionIndex: 0,
      transactionHash: '0xfcdf02c402035888273fc5e91e3a26ee0638021758307e616fb8faf41be22
8ed',
      blockHash: '0x93bcb7d76123c3771461ceef4f3ad4f4bb902988505671a4d5d178ece8ef9811',
      blockNumber: 24,
      address: '0x3d0Cd4f08dF2B20FfBe49B3dC9dA1772a5B7AA4E',
      type: 'mined',
      id: 'log_61685cd6',
      event: 'PostCreated',
      args: [Result]
    }
  ]
}
truffle(development)>
```

Fig. 6. Proof of Concept that the project works

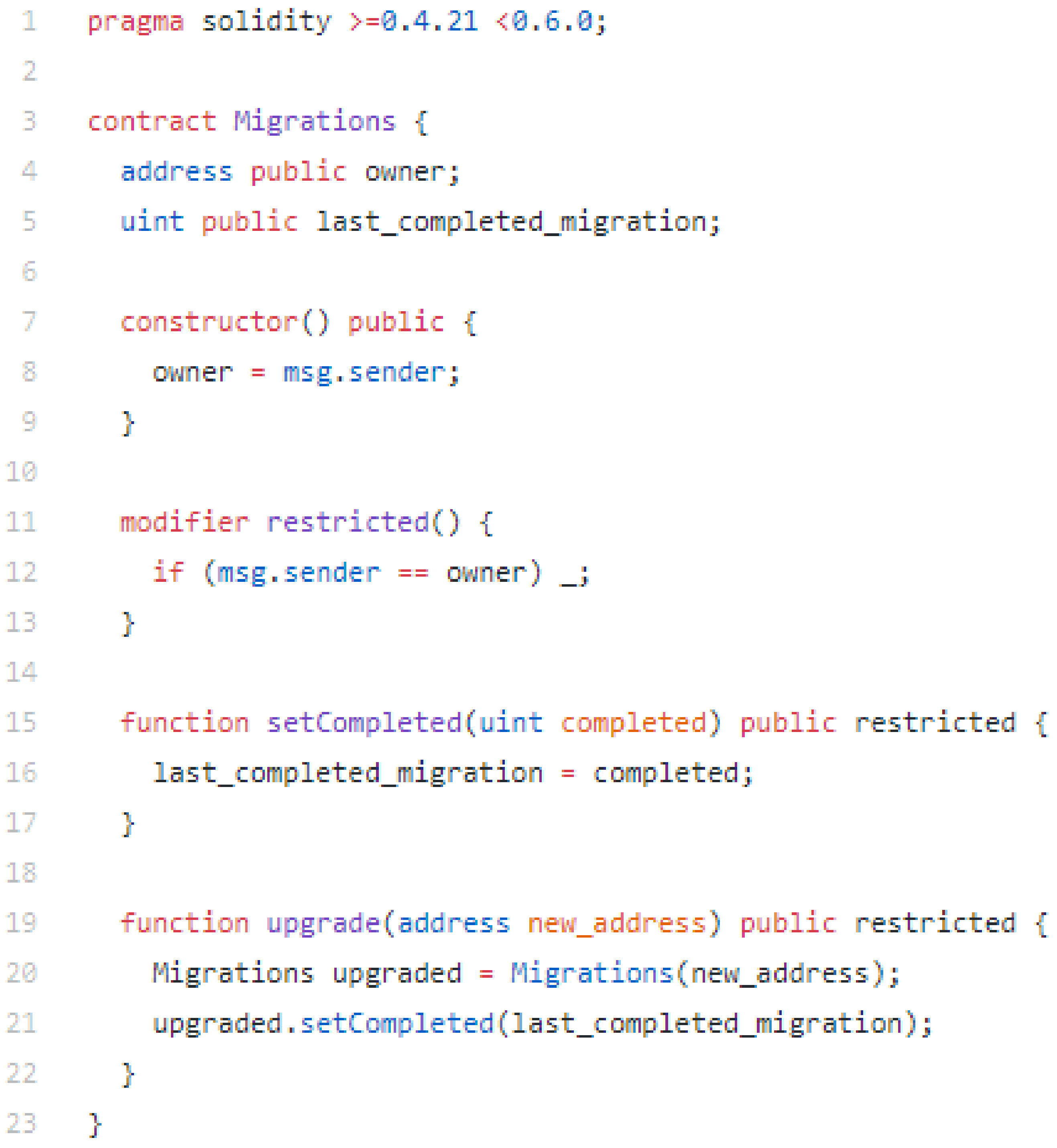
Appendix C
Contracts



```
pragma solidity ^0.5.0;

contract SocialNetwork{
        string public name;
        uint public postCount=0;
        mapping(uint => Post) public posts;

        struct Post{
                uint id;
                string content;
                uint tipAmount;
                address payable author;

        }

        event PostCreated(
                        uint id,
                        string content,
                        uint tipAmount,
                        address payable author

                );
                event PostTipped(
                        uint id,
                        string content,
                        uint tipAmount,
                        address payable author

                );

        constructor() public {
                name = "Team Foo Fighters";
        }
        function createPost(string memory _content) public {
                //require valid content
                require(bytes(_content).length > 0);
                //increment post count
                postCount++;
                //create post
                posts[postCount] = Post(postCount, _content, 0, msg.sender);
                //Trigger event
                emit PostCreated(postCount, _content, 0, msg.sender);
        }

        function tipPost(uint _id) public payable{
                require(_id > 0 && _id <= postCount);
                //Fetch post
                Post memory _post = posts[_id];
                //Fetch author
                address payable _author = _post.author;
                //pay author
                address(_author).transfer(msg.value);
                //increment tip amount
                _post.tipAmount = _post.tipAmount + msg.value;
                //update post
                posts[_id] = _post;
                //trigger an event
                emit PostTipped(postCount, _post.content, _post.tipAmount, _author);
        }
}
```

Fig. 7. Smart Contract Code Snip

Appendix D
Migration



```
pragma solidity >=0.4.21 <0.6.0;

contract Migrations {
  address public owner;
  uint public last_completed_migration;

  constructor() public {
    owner = msg.sender;
  }

  modifier restricted() {
    if (msg.sender == owner) _;
  }

  function setCompleted(uint completed) public restricted {
    last_completed_migration = completed;
  }

  function upgrade(address new_address) public restricted {
    Migrations upgraded = Migrations(new_address);
    upgraded.setCompleted(last_completed_migration);
  }
}
```

Fig. 8. Migration code snip