Visualization of Internal Blockchain Processes

Shao-Chun Ma
shao-chun.ma@rwth-aachen.de
Matriculation number: 373895

Supervisor: Prof. Dr. Thomas Rose
Advisor: Thomas Osterland

Chair for Computer Science 5
Information Systems
RWTH Aachen

This thesis is submitted for the degree of
M.Sc. Software Systems Engineering

Aachen, Germany
June 19, 2018

Abstract

In a blockchain system, multiple nodes distribute transactions and blocks simultaneously. Therefore, the behaviors of nodes and the mining processes are complex and difficult to be identified. In this thesis, we present a visualization tool which examines the influences of the network delay and mining strategies in a blockchain system which employs proof-of-work as the consensus protocol. Our approach is based on the simulation of a blockchain system, which is based on a multi-agent system. The data sent between the nodes are monitored and recorded by the watchdog, and then they are sent to the visualizer. As a result, the visualizer can provide a visualization of internal blockchain processes in real-time. We provide scenarios to demonstrate the potential applications of the visualization tool. Consequently, it proves that the visualization tool is suitable for helping researchers to analyze the mining processes step by step correctly.

Keywords: *blockchain; visualization; mining strategy; network delay; proof-of-work*

Declaration

I, Shao-Chun Ma, declare that the whole work is completed by myself without the help of the third parties, except the advice from the supervisor, Prof. Dr. Thomas Rose, and the advisor, Thomas Osterland.

This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Acknowledgements

First, I would like to appreciate the useful guides and suggestions from the supervisor, Prof. Dr. Thomas Rose, and the advisor, Thomas Osterland. Additionally, Prof. Wolfgang Prinz also helped me to improve the weakness of this thesis. Last but not the least, I owe the success of this thesis to my parents, Li-Ju Chen and Yuan-Chun Ma, and my wife, Yu-Ming Chien. They supported me through the difficulties that I encountered during the work.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

The blockchain technology was first proposed in Satoshi Nakamoto's paper, *Bitcoin: A Peer-to-Peer Electronic Cash System* [1], in 2008. It created a new type of digital currency, called *cryptocurrency*. For example, Bitcoin [2] and Ethereum [3] are two of the well-known cryptocurrencies. Cryptocurrencies have a significant characteristic that distinguishes them from traditional currencies: they are decentralized. This means that there is no authority in cryptocurrencies. The single authority in conventional currencies is necessary as it prevents the problem of double-spending. Therefore, the validity of transactions are important in cryptocurrencies, and it is fulfilled by consensus protocols which are based on cryptography techniques. [4]

The major process of the blockchain systems starts with the creation of a transaction. This transaction is distributed over the blockchain network. In this state, the transaction is not-mined and therefore not persistent. Every node in the network has its own transaction pools that contain the not-mined transactions.

In a blockchain system with the proof-of-work protocol, the nodes maintain the blockchain data structure by themselves. They compete against each other in extending the blockchain by creating new blocks that persistently store transactions from the transaction pools. The addition of new blocks provides effort, since it is necessary to solve a cryptographic puzzle with the characteristic that it is hard to solve, but easy to verify given a correct solution. The solving process is called mining. A correct solution is distributed over the network and verified by the remaining network participants. In the case that the verification succeeds, the block is added to the blockchain, and the containing transactions are removed from the transaction pools.

1.1 Motivation

There are two factors that play important roles in the mining processes: the *network delay* and the different *mining strategies* of miners. While a transaction is published through the blockchain network, each node receives it at a different time due to characteristics of peer-to-peer networks. Therefore, each miner has different pending

transactions in the transaction pool at the same time. Moreover, each miner mines a block according to the individual mining strategy simultaneously. The blocks generated by miners are different from each other, and they are also published through the unstable peer-to-peer network. Consequently, it is possible that the set of nodes partitions into different groups that work on different instances of the blockchain. These blockchains are maintained simultaneously until one blockchain becomes longer. The longest blockchain in the network is assumed to be the correct one, which is resolved by the consensus protocol.

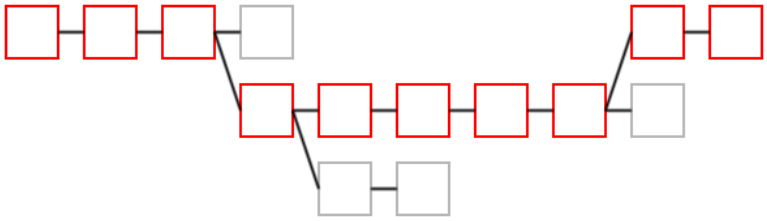


Figure 1.1: Red blocks are considered to be the longest blockchain.

For a simple example, the red blocks are the longest blockchain in Figure 1.1. There are forks on the path because of the past competitions.

In a real blockchain system, for example, there are about ten thousand of nodes in Bitcoin in April 2018 from an online statistical result [1]. For such wide and massive blockchain networks, it is hard to analyze the individual behavior of each node, e.g., the influences of the mining strategy to the mining processes under the unstable peer-to-peer network. Moreover, the mining processes of a blockchain system described earlier are dynamic and complex. Therefore, to focus on the research of the mining processes in a blockchain system, we decide to provide a visualization application to display the influences of the network delay and the mining strategy on the mining processes step by step.

1.2 Methodology

Figure 1.2 explains the basic idea of the solution. The *simulation* of the blockchain system is based on a multi-agent system, and the *watchdog* monitors and records the mining activities in the blockchain system. Thus, when the transactions and the blocks are published and received by the nodes, the watchdog sends the required data to the *visualizer*, which is responsible for visualizing all the mining events that happened in the blockchain system.

The application is able to visualize the steps of the mining processes of each node in real-time. By setting the parameters of the network delay and the mining strategy, researchers can analyze and observe the mining processes with predefined conditions in the blockchain system. Additionally, in order to replay the same result of the blockchain processes, it is possible to provide a configuration file that defines all the

[1] `https://bitnodes.earn.com/`

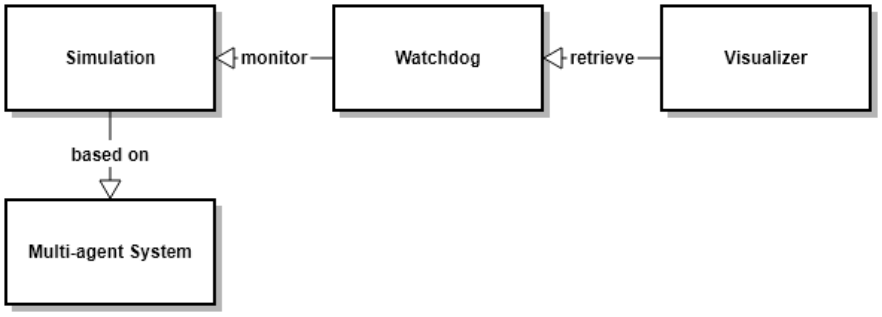


Figure 1.2: The method.

properties of nodes and the parameters of the network delay and the mining strategy in the blockchain system.

1.3 Contribution

There are 8 online visualization applications, Bitbonkers, Bitnodes, BitcoinCity, Blockseer, DailyBlockchain, Elliptic, Interaqt, and Live Globe, that are summarized by Tri A. Sundara et al. [5]. Additionally, two applications, Blockchain and Etherscan, use tabular methods to represent transactions and blocks. However, these visualization applications only display the static information on Bitcoin network such as the values and the timestamps, and thus they cannot satisfy the goal which helps users to understand the dynamic mining processes.

There are collections of previous literature that provide visualization tools to analyze the patterns of transactions on Bitcoin network. These visualization tools [6, 7, 8, 9] use statistical methods such as line charts, bar charts, and graph analysis to visualize the relationships and patterns of transactions and blocks. The big data visualization methods are applied [10] to find patterns of blocks. The flows of bitcoins can be tracked in the time series analysis [11]. The real identification of an address can also be tracked in the analysis [6]. These visualization tools are implemented for analysis, though our solution focuses on visualizing the actual mining processes.

Amitai Porat et al. [12] proposed a method to analyze the potential applications of proof-of-work based blockchain networks. They considered the influences of network latency to the blockchain system. Nevertheless, our solution also considered the influences of mining strategy which are not presented in their work.

Comparing to other visualization applications and tools for blockchain systems, our solution provides the visualization of the dynamic mining processes. Furthermore, the mining processes are visualized step by step while the miners are publishing blocks through the network. Consequently, our solution is suitable for researching and understanding the complex mining processes with the factors of network delay and mining strategy.

1.4 Thesis Structure

This thesis is composed of the following chapters.

- **Chapter 2**
  The review of the previous literature and the contribution of our approach are provided in this chapter.
- **Chapter 3**
  The assumptions and definitions of the blockchain system that the visualization is based on are given in this chapter.
- **Chapter 4**
  This chapter contains the architecture of the visualization application and the components that are implemented.
- **Chapter 5**
  In this chapter, the introduction to the usage of the visualization application is provided.
- **Chapter 6**
  Three scenarios that can be achieved by the visualization application are proposed in this chapter.
- **Chapter 7**
  At the end of this thesis, the conclusion and the future work are discussed.

Chapter 2

Related Works

Since the blockchain technology is proposed, different methods for visualization and analysis are explored. As Bitcoin is the first and main blockchain network in the world and the data can be accessed publicly, much literature conducts researches based on it. Generally, there are three categories of related works for the visualization of a blockchain system.

- **Visualization of Blockchain Information**
  There are a lot of online visualization applications which can visualize the static information of transactions and blocks on a blockchain system. They display the information in detail and usually serve as the base for analysis.
- **Analysis of Bitcoin Transactions**
  As Bitcoin becomes a popular peer-to-peer network recently, most of the recent literature focuses on the analysis of the relationships between transactions and blocks on Bitcoin network. They try to recognize special patterns to gain economic experience and prevent criminal activities.
- **Analysis of Consensus Protocols**
  An analysis of the consensus protocols on the blockchain networks is an interesting topic as it can reveal the potential applications on blockchain platforms.

2.1 Visualization of Blockchain Information

There are some online tools and applications that provide visualization of the static status and information of transactions and blocks on Bitcoin network. Tri A. Sundara et al. [5] summarized 8 visualization applications and compared their visualization methods and technologies. We shortly review these applications here, so the main difference between our application and them can be identified clearly.

- **Bitbonkers** [13]
  Bitbonkers rendered a live 3D animation of transactions and blocks. Balls represent transactions, and they keep dropping down to the plate from the top. On the other hand, cubics represent blocks. The sizes of balls are different according to their total values. (Figure 2.1)

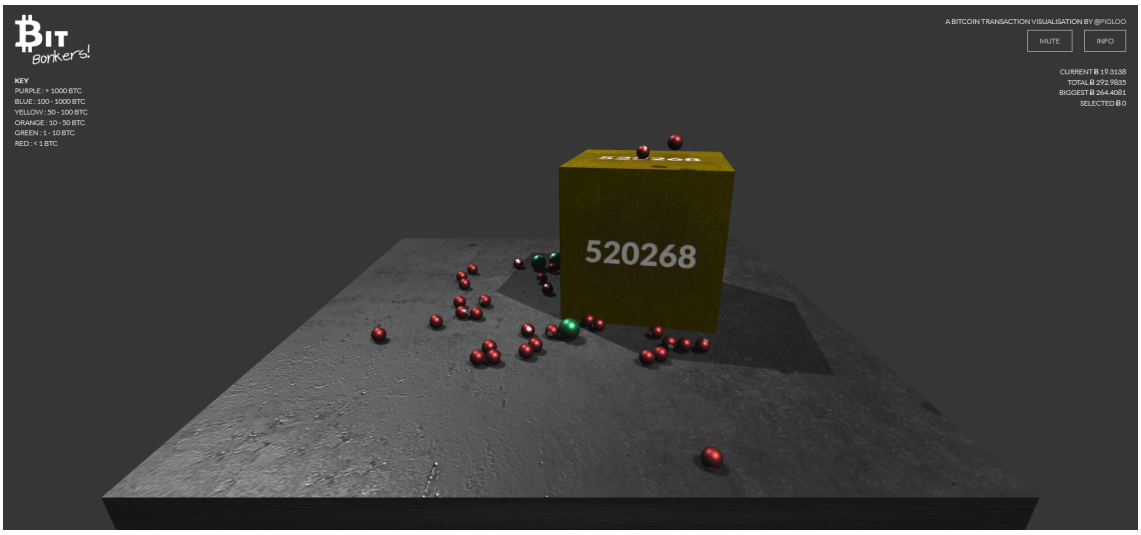


Figure 2.1: Bitbonkers.

- **Bitnodes** [14]
  Bitnodes focused on the visualization of the distribution of nodes on Bitcoin network. The density of nodes indicates the most active areas on Bitcoin network in the world.(Figure 2.2)

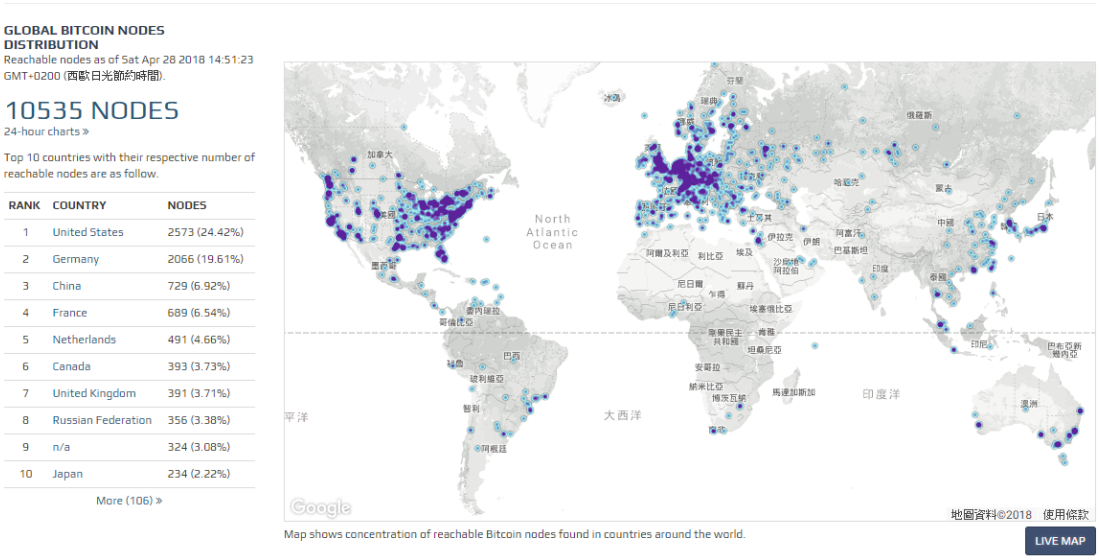


Figure 2.2: Bitnodes.

- **BitcoinCity** [15]
  BitcoinCity built a small city as the visualization of Bitcoin. It used a road to chain the blocks, and the houses with different heights on both sides of the road represented transactions with different values. (Figure 2.3)

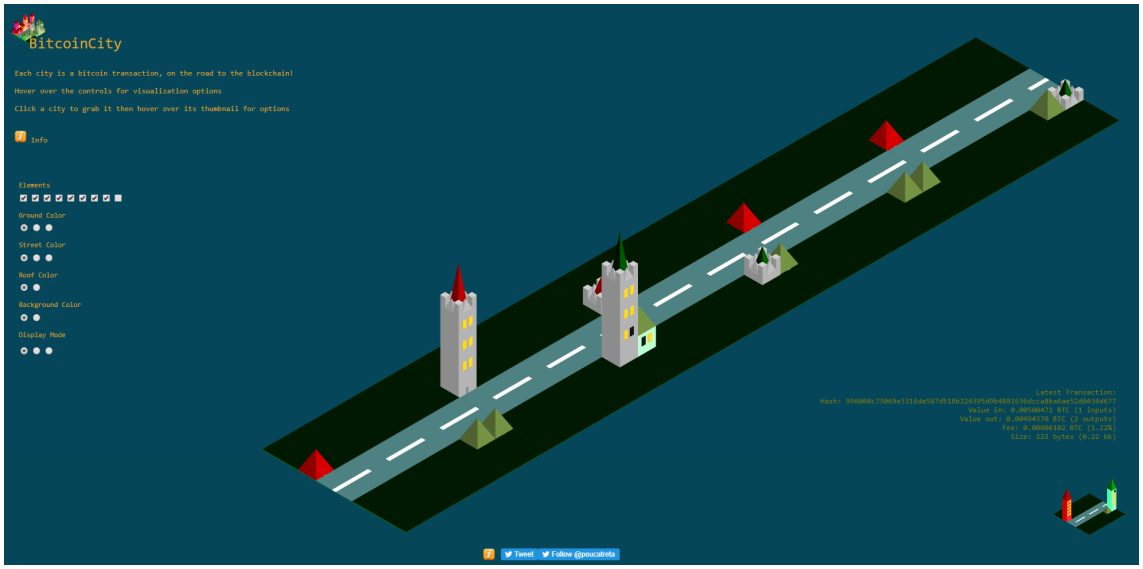


Figure 2.3: BitcoinCity.

- **Blockseer** [16]
  Blockseer located a mined transaction and visualized the outputs of the transaction. For example, the screenshot shows there are nine outputs of the transaction which in at the root of the tree structure. That is, the transaction is divided into nine parts which are delivered to different addresses respectively. (Figure 2.4)

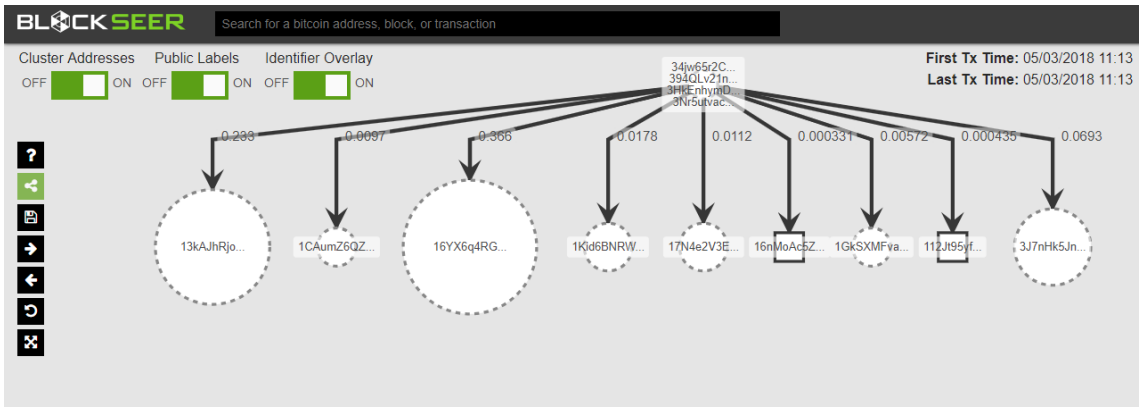


Figure 2.4: Blockseer.

- **DailyBlockchain** [17]
  DailyBlockchain displayed the mined transactions as the biological cells. The size of the cell are determined by the number of outputs of the transaction. The cells are generated from the center and spread toward every direction. (Figure 2.5)

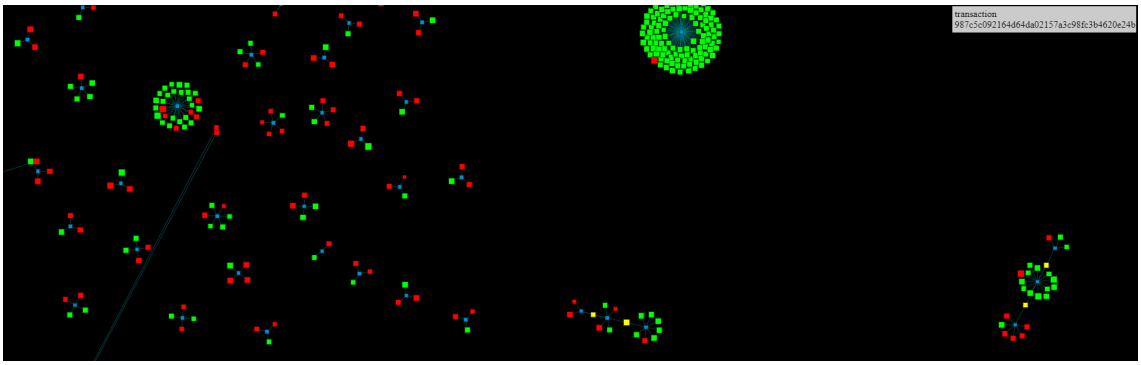


Figure 2.5: DailyBlockchain.

- **Elliptic** [18]
  Elliptic visualized the whole history of Bitcoin like the Big Bang of the universe. However, the oldest blocks are placed at the center of the universe, which is opposite to the Big Bang theory in the reality. The blocks are divided by the dotted circles according to their birth years. (Figure 2.6)

Figure 2.6: Elliptic.

- **Interaqt** [19]
  Interaqt gathered the mined transactions which are represented in circles into a big circle. The screen is empty at first, the more and more transactions will join into the big circle as time passes. (Figure 2.7)

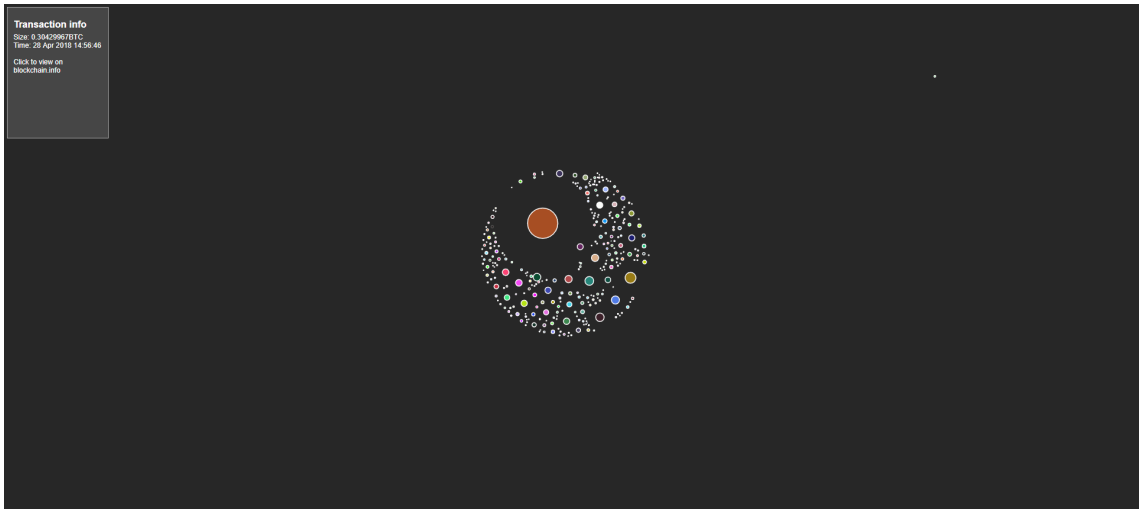


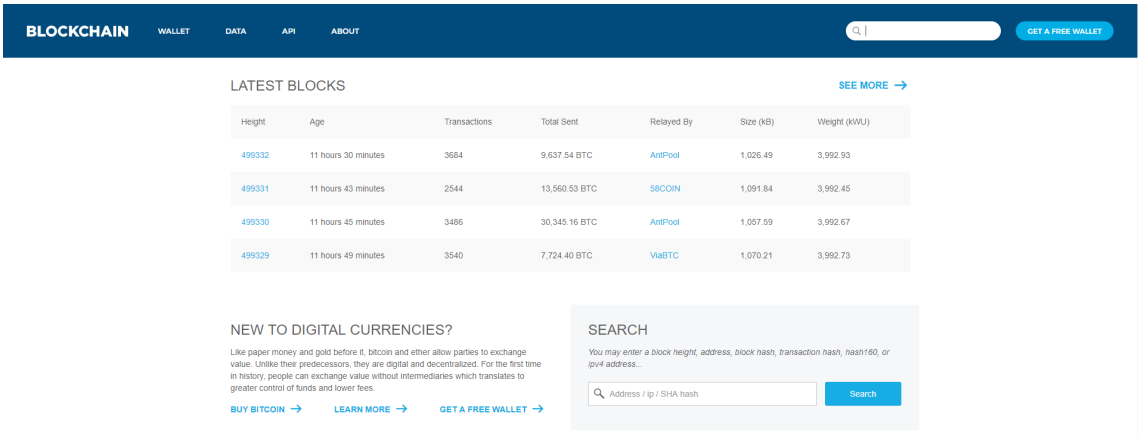
Figure 2.7: Interaqt.

- **Live Globe** [20]
  Live Globe combined the geographies of blocks and the earth together in the visualization. There are two blocks that are placed at the position of New York City in the screenshot. However, the color of the blocks are also green, so it is difficult to distinguish them. (Figure 2.8)

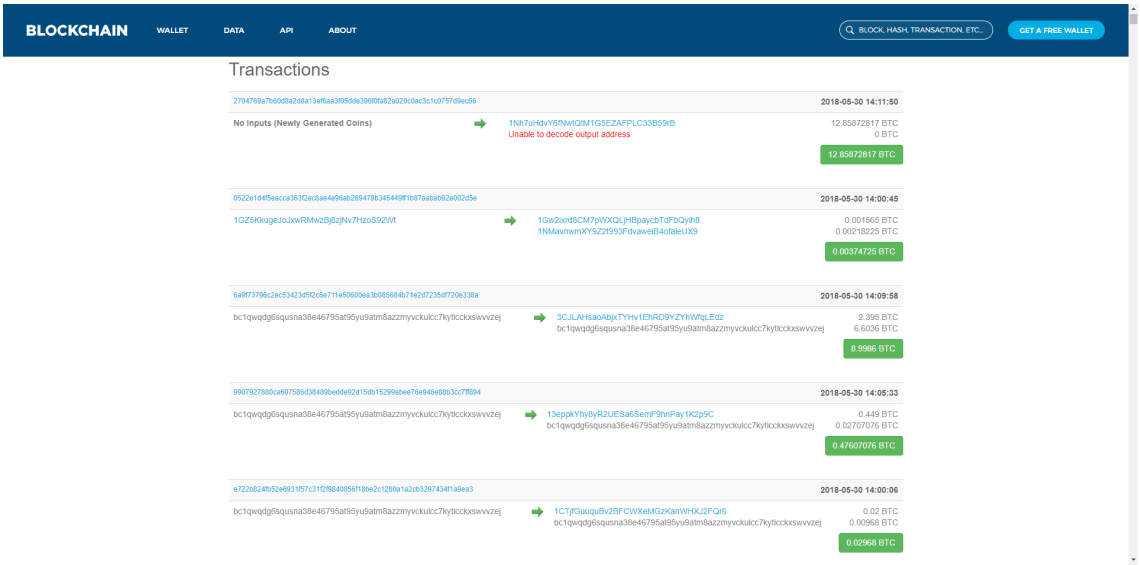
Figure 2.8: Live Globe.

In addition to the 2D and 3D animations of the visualization of Bitcoin, there are applications which use tabular methods as the visualization of blockchains.

- **Blockchain** [21]
  The tabular (Figure 2.9a) contains the height, the timestamp, the mined transactions, the total values, the size, etc. of mined blocks on Bitcoin network. The details information of a block can be viewed by selecting it, and then the list of included transactions with addresses, inputs, and outputs (Figure 2.9b) is displayed.



(a) Blocks by Blockchain.



(b) Transactions by Blockchain

Figure 2.9: Blockchain.

- **Etherscan** [22]
  Etherscan which is for Ethereum network is similar to Blockchain, but it places the information of blocks and transactions on the same page. The top right part displays the line chart of the marketing value of Ethereum. The list of blocks is placed on the bottom left of the page, and the list of transactions is at

the bottom right part. The details can also be checked by selecting individual blocks or transactions. (Figure 2.10)

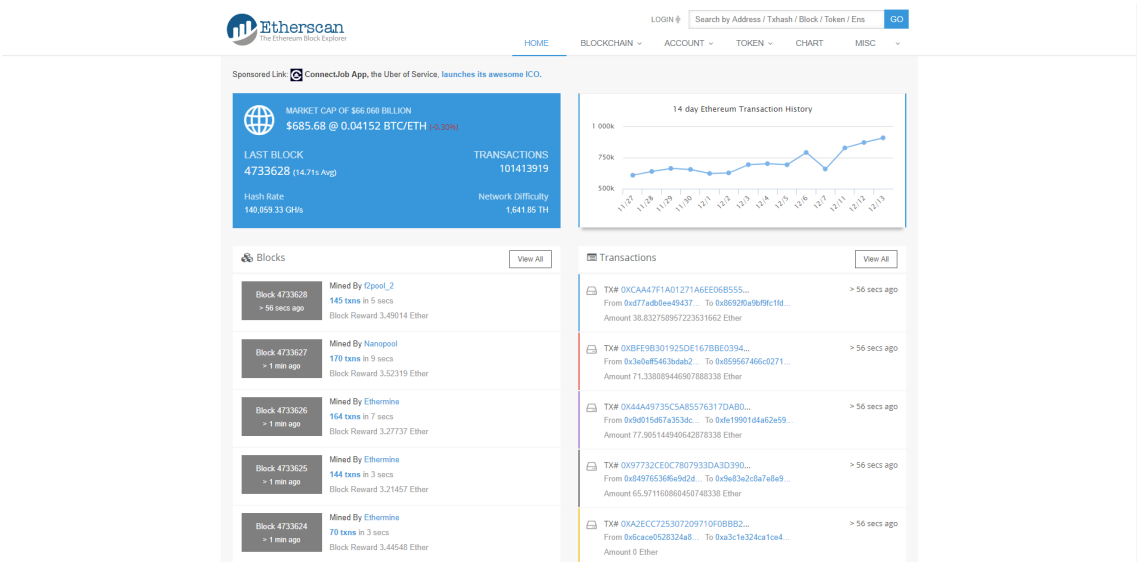


Figure 2.10: Etherscan.

In short conclusion, although the above tools and applications showed visualization methods of blockchains, their goals are to provide the static information and statistics instead of the dynamic mining processes that happen in a blockchain system. It is difficult to gain valuable understandings about the complex and dynamic mining processes in these applications.

2.2 Analysis of Bitcoin Transactions

As Bitcoin is the most popular and famous blockchain network currently, it is worth to explore the patterns of transactions and blocks to prevent malicious abuse of blockchain services. The nodes and transactions are anonymous on blockchain networks, so the recognition of patterns can be challengeable. Here we present some analysis methods and visualization technologies that can identify the relationships between nodes and transactions in time series on Bitcoin network.

- **BitConeView: Visualization of Flows in the Bitcoin Transaction Graph** [11]
  Giuseppe Di Battista et al. provided a visualization tool called *BitConeView* that analyzed the flows of bitcoins in the transactions by timestamps. It can be used to analyze the patterns of the flows of bitcoins and locate tainted transactions. Figure 2.11 [11] shows that the flows of bitcoins are visualized in time series, and the lifetime of a bitcoin can be tracked in the connected bars which are blocks.

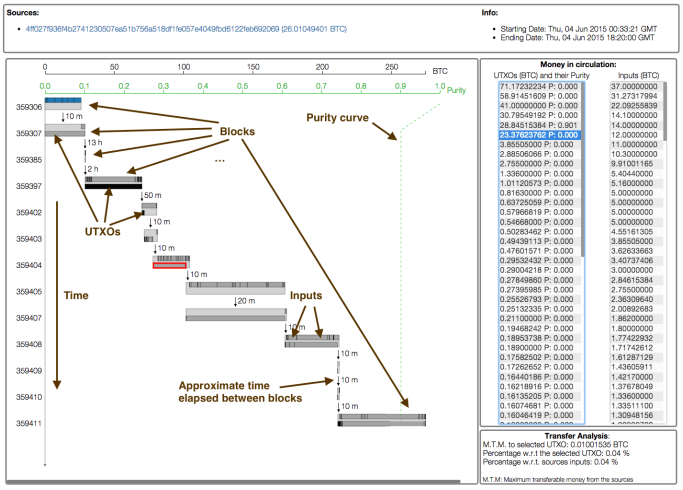


Figure 2.11: BitConeView.

- **Blockchain Explorer: An Analytical Process and Investigation Environment for Bitcoin** [6]
  To prevent the criminal activities and illegal behaviors that abuse the services of Bitcoin, Hiroki Kuzuno et al. proposed an analyzing system which managed the data and statistics of blockchains for the usages of law enforcement investigation and training. Figure 2.12 [6] shows the analysis methods in the visualization. The line chart on the left side represents the balance of an address, and the bar charts on the right side represent the number of transactions in hours or weeks.

Figure 2.12: Analysis methods by Blockchain Explorer.

Figure 2.13 [6] displays the activities of an address in calendars. The bright squares indicate the time when the address published transactions.

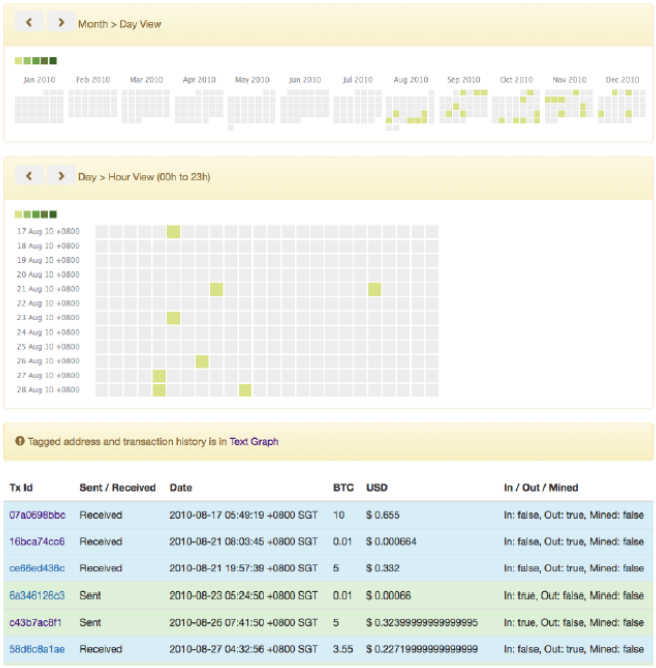


Figure 2.13: Activities of an address by Blockchain Explorer.

Figure 2.14 [6] maps the inputs and outputs of a transaction into tree structures. As a result, the flows of bitcoins can be tracked by the input and output nodes.

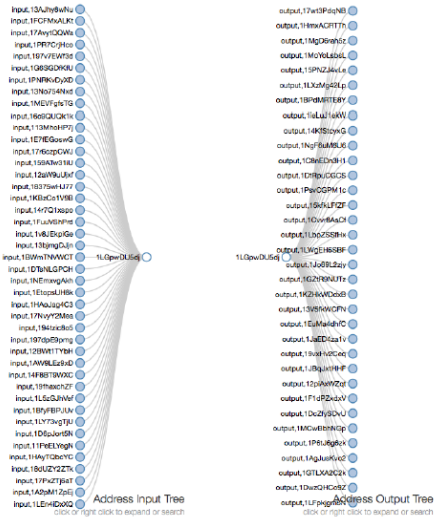


Figure 2.14: Mappings of inputs and outputs in a transaction by Blockchain Explorer.

With above visualization, the activities of an address and the mappings to the real identification can be explored.

- **Visualizing Dynamic Bitcoin Transaction Patterns** [10]
  Dan McGinn et al. focused on visualizing transactions of Bitcoin from the top-down viewpoint. The visualization of the large scale of data provided domain experts and the general public a tool to analyze and identify the transaction patterns with big data visualization methods. Figure 2.15 [10] shows the visualization result. The blocks are connected by lines if they contain related transactions. Thus, different patterns can be identified in the graphs.

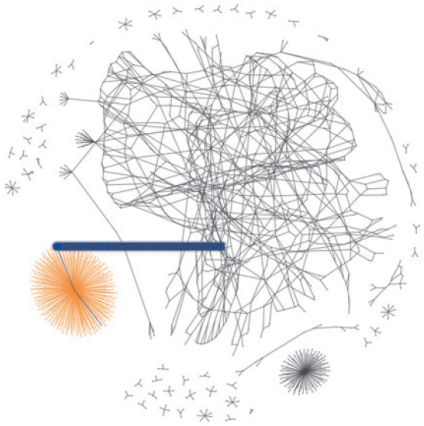


Figure 2.15: Visualization by Dan McGinn et al.

- **Bitcoin Visualization** [7]
  The visual analytics tool that was proposed by Loïs Saublet enabled economists to analyze the metrics and actors on Bitcoin network and provided good user experience by working together with end users. As in Figure 2.16 [7] shows, a period of Bitcoin marketing values can be displayed by a line chart, and economists can examine it.

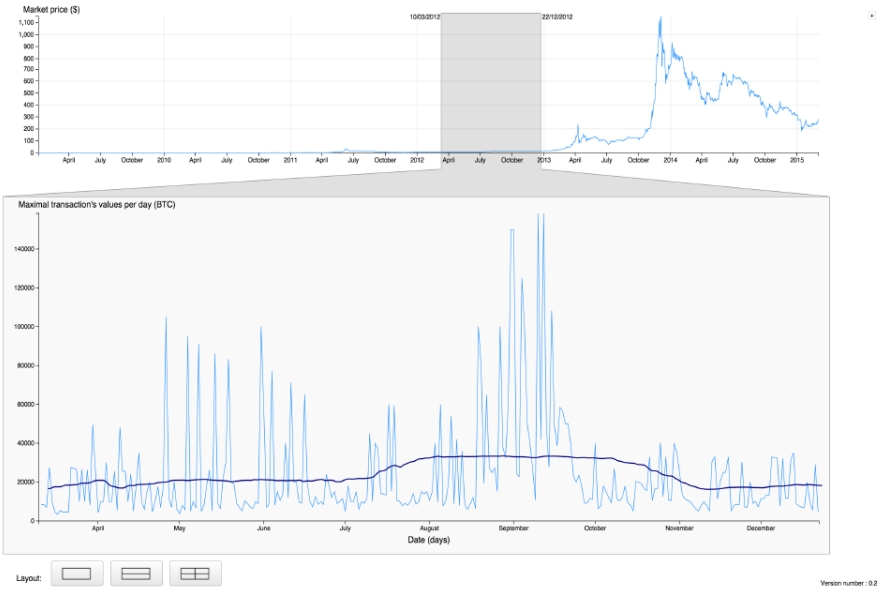


Figure 2.16: Visualization by Loïs Saublet.

- **Bitcoin Transaction Graph Analysis** [8]
  Michael Fleder et al. proposed a graph-analysis framework that could analyze the relationships between public addresses and transactions. It can be used to match candidate Bitcoin transactions with the transactions in the reality. As a result, it demonstrated that the transactions are not entirely anonymous on Bitcoin network. Figure 2.17 [8] shows the transaction graph which gathers the data in a whole day. The relationships of transactions can be identified via the connections. The thickness of the lines are the values, so several patterns such as large volume transactions can be analyzed.

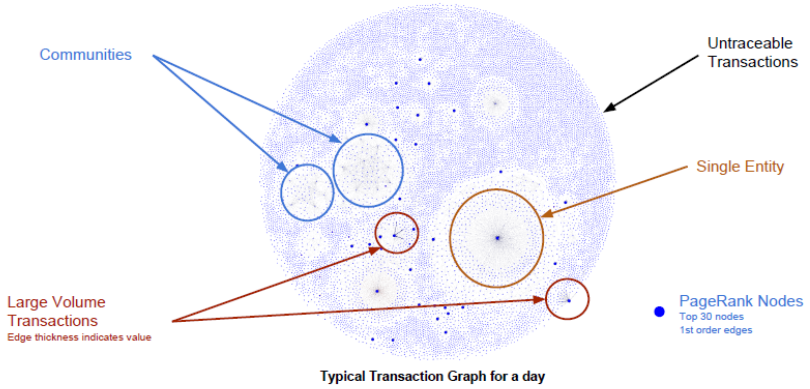


Figure 2.17: Visualization of the transaction graph.

- **Exploring the Bitcoin Network** [9]
  The method proposed by Annika Baumann et al. employed graph mining algorithms to analyze the relationship of network usage and exchange rate. It served as the basis for the analysis of the anonymity and economic relationships on Bitcoin network. They used statistical bar charts or line charts for visualization as Figure 2.18 [9] shows.

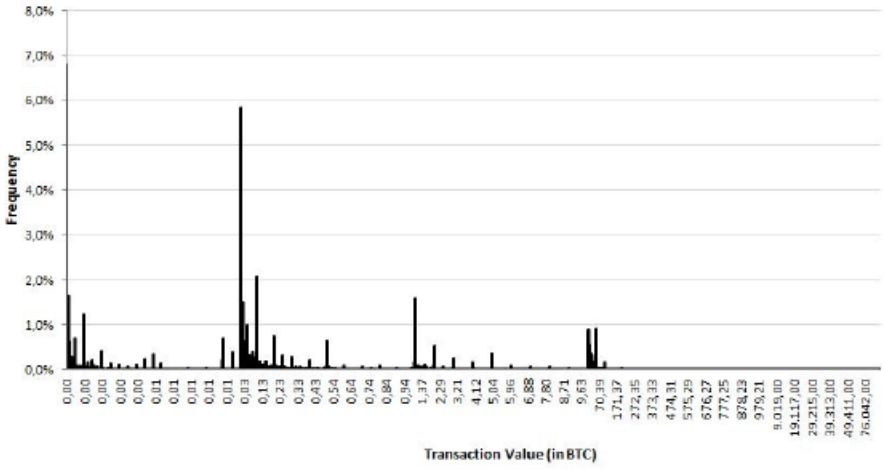


Figure 2.18: Visualization by Annika Baumann et al.

In summary, the goals of the above tools and methods are to analyze the patterns and relationships of transactions and blocks. That is, they try to recognize patterns to prevent criminal activities and obtain economic inspiration. Therefore, the visualization results of the above literature are more like statistical results for the analysis instead of the mining processes.

2.3 Analysis of Consensus Protocols

To analyze the network conditions of proof-of-work based blockchain system, Amitai Porat et al. [12] provided an application that was based on Ethereum platform. It simulates the asynchronous mining activities and network latency, and provided a real-time analysis of the blockchain system. They presented the potential applications of the blockchain technology through the analysis.

Figure 2.19 [12] shows the visualization by Amitai Porat et al. The visualization is very simple, as the blocks are chained by lines. Therefore, the miners of the blocks cannot be identified in the visualization.

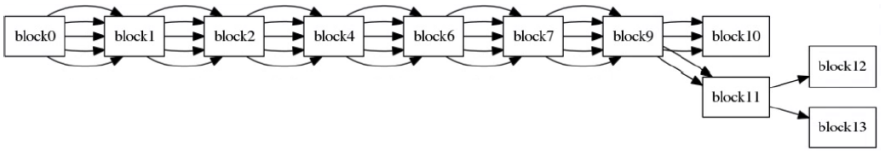


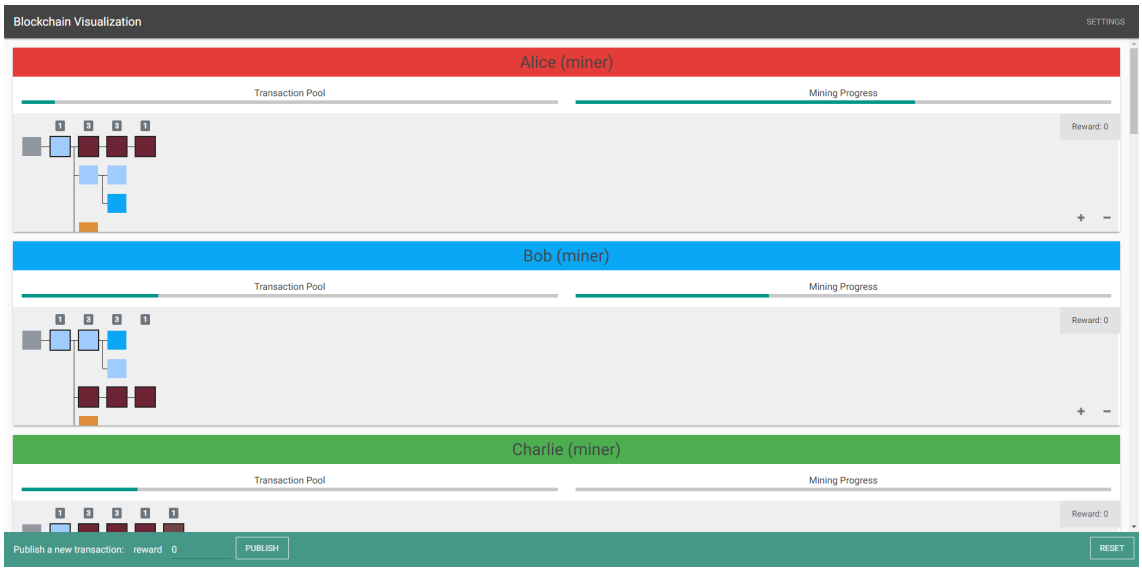
Figure 2.19: Visualization by Amitai Porat et al.

The main difference to our work is that we assume that miners have individual mining strategies which are parameterized, e.g., they select different sets of transactions from transaction pools. Therefore, there are miners who are able to mine blocks faster than the others under the same computing power, but they may suffer lower mining rewards. The combination of different mining strategies and the network delay makes the visualization of the blockchain system dynamic and undetermined. Thus our solution is suitable for researchers to find the influences of different factors on the blockchain system. Moreover, their work focused on finding the potential applications of Ethereum platform. On the other hand, our work emphasizes on visualizing the continuous mining processes while miners have different mining strategies and suffer from the network delay.

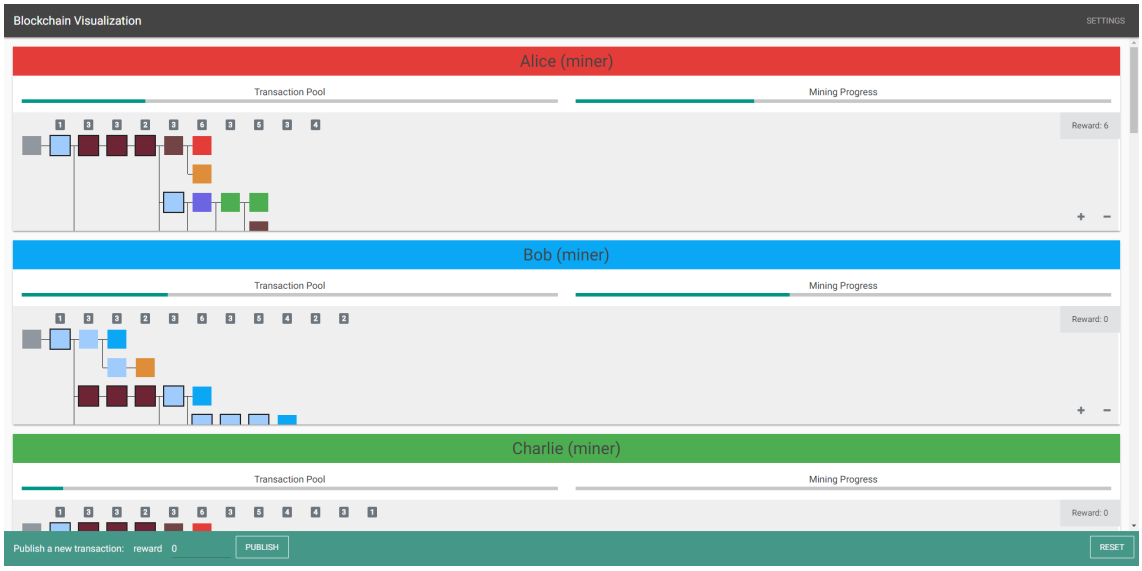
2.4 Our Approach

To focus on the dynamic mining activities, our tool provides a real-time visualization of the mining processes that cannot be achieved by the above visualization tools and applications. Through our visualization, users can observe the mining processes step by step. Moreover, our visualization is independent to specific blockchain platforms,

e.g., Bitcoin and Ethereum, because we built a simple simulation of the blockchain system ourselves. Therefore, we can concentrate on the mining processes which are the most important activities in a proof-of-work based blockchain system.



(a) Mining Step 1



(b) Mining Step 2

Figure 2.20: Steps of Mining.

As we can see in Figure 2.20, the tree structure of the blockchains keeps growing while the mining activities continue. The basic visualization ideas of our project are inspired by a former project called "Kooperative Music Box" [23] which was developed by Fraunhofer Blockchain-Labor. It used shapes and colors to distinguish different blocks and transactions which were generated by different nodes. As a result, in the visualization application, the blocks are represented as squares, and they are chained by lines to form tree structures, and the colorful blocks are generated by different miners.

Our project aims to visualize all the dynamic mining processes that could happen in a blockchain system, with the factors of the network delay and the different mining

strategies. This distinguishes our work to all the related works mentioned before. Consequently, our work is worthwhile as a contribution and a trial to the blockchain research area.

Chapter 3

Blockchain System

Before continuing to the later part, it is worth to define the blockchain system that we refer to in the visualization application. Because the research of blockchain technology is still in progress, there are different methods and approaches which are used to construct different blockchain systems. In this chapter, we introduce the definitions of the blockchain system that serves as the basis of the visualization application. It contains data types, different nodes, network delay, mining strategy, and the consensus protocol.

3.1 Data Types

There are only two types of data structures that are used in the blockchain system, *transactions* and *blocks*. These are published and received by nodes continuously in the same blockchain network. A transaction contains information such as the trade information of cryptocurrencies or the information of an item. A block contains a number of transactions that are mined by a miner, and blocks are chained together to form a tree structure that represents the blockchain data structure. The transactions that are included in a block are considered to be confirmed if the block is in the longest blockchain.

For transactions (Table 3.1), the properties contain the *ID*, the *type*, the *timestamp*, the *reward*, and the *privilege*. The *ID* represents the identifier of the transaction in a blockchain system. Here we use 8 random characters and digits to simplify the representation. The *type* indicates that the data structure is a transaction. The *timestamp* is the time when the transaction is created.

The *reward* and the *privilege* are related to the mining strategy. The *value* of a transaction is the sum of the reward and the privilege, as the equation 3.1 states.

$$value = reward + privilege \tag{3.1}$$

Transaction	
Properties	*Description*
ID	the identifier of the transaction.
type	is always "transaction".
timestamp	the time when the transaction is created.
reward	the number of rewards that the miner will receive.
privilege	to prevent the starvation of the transaction.

Table 3.1: Properties of Transactions.

Miners can set the *minimum value of transactions* that are qualified to be mined. The reward is the number of money that will be assigned to the miner as a motivation because miners spend computing power to solve puzzles in a proof-of-work based blockchain system. The privilege is set to 0 when a transaction is created, and it is added by a specific number if the transaction is not selected to be mined each time during the mining activities. However, the increased value does not change the reward that the miner will receive. That is, the privilege only prevents the transaction from starvation, i.e., the transaction will not have a chance to be mined because of its low reward.

To explain the value of transactions clearly, suppose that a transaction with the reward of 5 was generated and published, and a miner, Alice, received this transaction. At this time, Alice set the privilege of this transaction to 0. Now the total value of this transaction is

$$\begin{aligned} reward &= 5 \\ privilege &= 0 \\ value = reward + privilege &= 5 + 0 = 5 \end{aligned}$$

Assume that Alice decides to mine a block, but she does not select this transaction as one of the candidates because Alice expects that the *minimum value of transactions* should be 6. Therefore, Alice adds the privilege of this transaction by a specific number (1 in this example). Now the value of this transaction is

$$\begin{aligned} reward &= 5 \\ privilege &= 1 \\ value = reward + privilege &= 5 + 1 = 6 \end{aligned}$$

Thus, when Alice decides to mine a block next time, she will select this transaction as one of the candidates because the value of this transaction is not less than 6. It

demonstrates that the privilege prevents this transaction from starvation, but Alice still gets the reward of 5 by mining this transaction. We design this mechanism because a transaction should not be ignored forever in the visualization tool, but in the real world, the one who published the transaction with low reward should increase the reward by themselves.

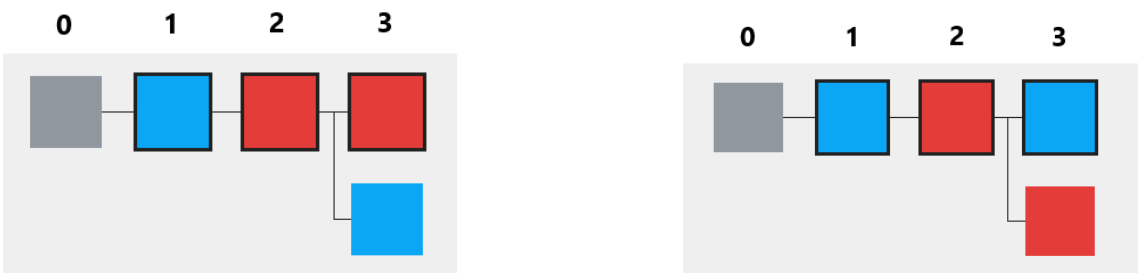
Block	
Properties	*Description*
ID	the identifier of the block.
type	is always "block".
timestamp	the time when the block is created.
miner	the public address of the miner.
previous	the hash value of the previous block.
layer	indicates the position of the block in the blockchain.
color	the color of the block.
transactions	an array of transactions.

Table 3.2: Properties of Blocks.

For blocks (Table 3.2), the properties contain the *ID*, the *type*, the *timestamp*, the *miner*, the *previous*, the *layer*, the *color*, and the contained *transactions*. The *ID* represents the identifier of the block in a blockchain system. Here we also use 8 random characters and digits as the hash value. The *type* indicates that the data structure is a block. The *timestamp* is the time when the block is created. The *miner* represents the public address of the miner who mined this block, and it is 8 random characters and digits. The *previous* contains the ID of the previous block that is chained in the same blockchain. The *block* contains an array of transactions, and the total reward of the block is the sum of these transactions.

The *layer* and *color* are useful for the visualization of the blockchains. The layer defines the position of a block in a blockchain and ensures that the results of the blockchain databases between different nodes are the same. The different colors distinguish the miner of the blocks in the visualization. The blocks with the same color mean that these blocks are from the same miner.

To illustrate the positions of blocks in the visualization, suppose that there are two miners, Alice and Bob. Alice's color is red, and Bob's color is blue. According to Figure 3.1, Alice and Bob mined two blocks individually. The grey block represents the genesis block. The red blocks are from Alice, and the blue ones are from Bob. The numbers of layers of the blocks are indicated on the top of the figures. In each layer, the numbers of different colors of blocks are the same. However, the vertical positions of the blocks are not always the same. It is because that each miner received the same block at a different time. In this example, Alice and Bob



(a) Alice's blockchain (b) Bob's blockchain

Figure 3.1: Visualization of Blocks.

put their own blocks on the top of the layer because they received their own blocks earlier. As a result, the layer guarantees that the blockchain databases are equal between every node, but remains the difference of received time.

3.2 Nodes

In the blockchain system, there are three types of nodes that communicate with each other.

- **Transaction Generator**
  The transaction generator is unique in the blockchain system. It is responsible for generating and publishing transactions to miners.
- **Miner**
  Miners are the most important nodes in the visualization because they mine and publish blocks according to their individual mining strategies. Each miner has her own transaction pool which contains all the pending transactions. Because the transaction generator publishes the transactions through the unstable network, each miner has different sets of pending transactions at the same time.
- **Nonminer**
  Nonminers only receive blocks from miners and publishes blocks to their neighbors.

The blockchain data structures of different nodes are not always the same since the blockchain system is started. It is because of the unstable peer-to-peer network between different nodes, and the correct and real-time visualization of the different blockchain data structures is the main feature of our application. For example, Figure 3.1 shows the difference between Alice's and Bob's blockchain data structures.

The relationship between different types of nodes is shown in Figure 3.2. In the beginning, the transaction generator publishes transactions to the miners. When a miner mines a block, she will publish the block through the blockchain network. Other miners and nonminers will again publish the received block to their neighbors

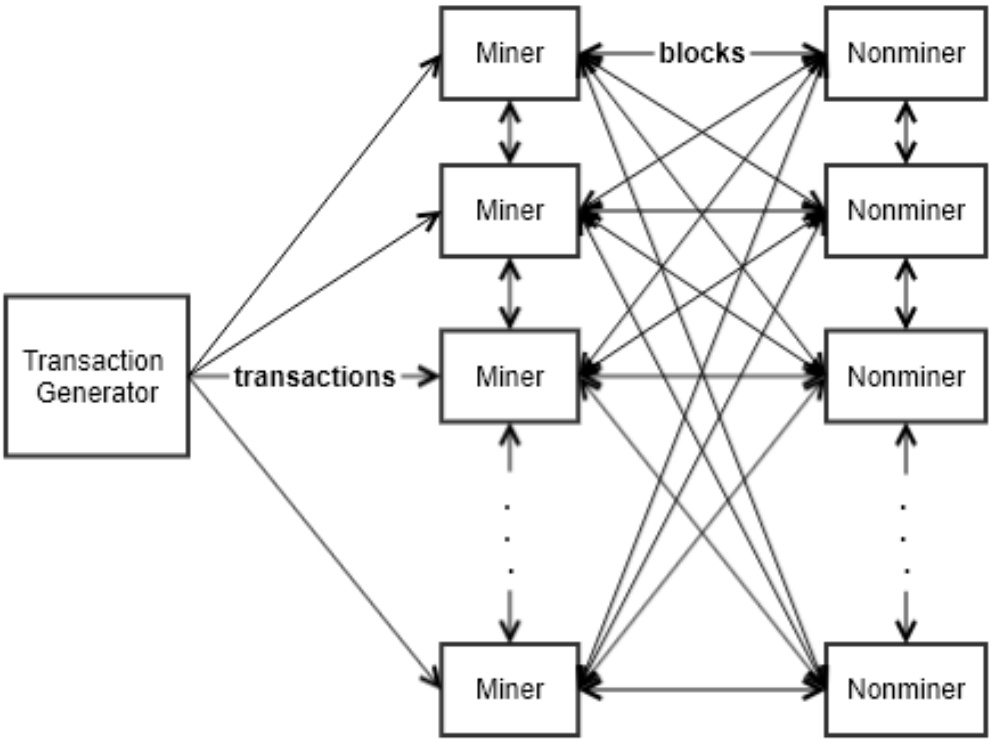


Figure 3.2: Relationships of Nodes.

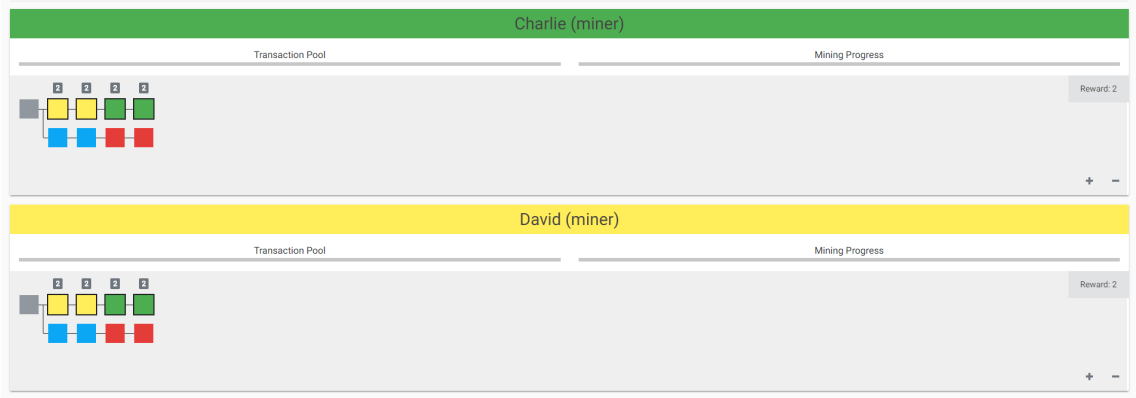
and they validate it after receiving. If a miner or a nonminer receives duplicate blocks, then she will discard the block to prevent sending and receiving the same block repeatedly.

3.3 Unstable Network

The network between each node is fragile due to the characteristic of peer-to-peer networks. Therefore, the publishments of transactions and blocks have a period of delay. Because of that, forks will happen in the blockchain visualization while several miners are mining simultaneously. Moreover, nodes could be partitioned into different groups and compete with other groups.

For example, in Figure 3.3, there are four miners, Alice (red color), Bob (blue color), Charlie (green color), and David (yellow color). They are partitioned into two groups, i.e., Alice and Bob in group 1 and Charlie and Bob in group 2, because the connections of different groups have high network delay, and the networks of the miners in the same group are much faster. Therefore, the miners in the same group consider that their own blockchain is the longest because they share information with other members in the same group more easily than those in the other groups. The phenomenon that Alice and Bob put their blocks on the top and the same for Charlie and David is displayed in the visualization correctly.

(a) Group 1



(b) Group 2

Figure 3.3: Groups of Miners.

3.4 Mining Strategy

Every miner has different mining strategy. There are four parameters that influence the mining strategy and mining activities.

- **Mining Time**
  A miner needs a period to mine a block by solving the puzzles. Thus, this parameter reflects the computing power of miners and the difficulty of puzzles.
- **Minimum Value of Transactions**
  This parameter is used to select a set of candidate transactions that are considered to be mined into a block. It is the threshold of the values of transactions which are qualified to be selected when the miner decides to mine a block.
- **Number of Mined Transactions**
  This parameter is the size of a block, i.e., the number of transactions that a block must contain. The sizes of blocks are always fixed for the same miner.
- **Maximum Number of Pending Transactions**
  Miners store pending transactions in the transaction pools. This parameter prevents that a miner responds too slowly to the blockchain system when the minimum value of transactions is set by a large number.

The four parameters make the mining behaviors of the miners different from each other. The influences of the mining strategy to the blockchain system under specific environment can be identified clearly through the visualization. More details of how the mining strategy is applied in the mining activities can be found in Section 4.6.

3.5 Consensus Protocol

The consensus protocol that is applied in the blockchain system is proof-of-work. Under proof-of-work protocol, miners solve cryptographic puzzles with their computing power. By comparing the blockchain data structures between each node, the longest blockchain can be identified easily. Miners always switch to the longest blockchain immediately to ensure that they are working on the correct fork of blockchains. Additionally, miners only get rewards by adding blocks to the longest blockchain.

The parameters of the mining strategy are designed for proof-of-work protocol. Therefore, the parameters will be entirely different if the applied consensus protocol is changed to other ones such as proof-of-stake in the future.

Chapter 4

Implementation

In this chapter, the details of implementation are discussed. It starts with the overview and the architecture of the visualization application, and is followed by describing three important component in the application: the simulator, the watchdog, and the visualizer. In the next section, we provide a UML class diagram and several tables to describe the functionalities of the classes. Finally, the important algorithms about the mining processes are proposed to clarify the operation of mining activities.

4.1 Architecture

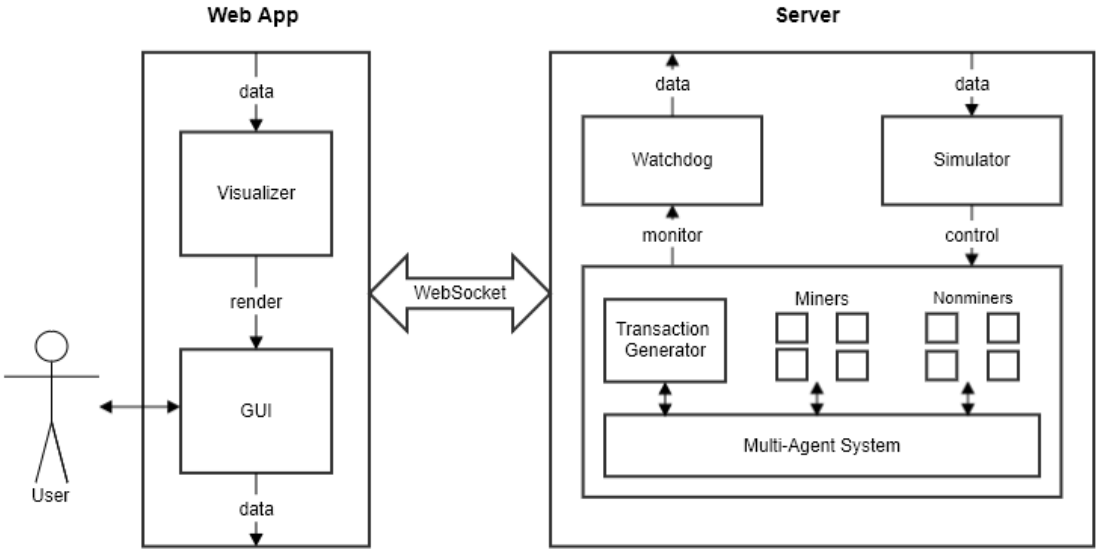


Figure 4.1: Architecture.

Figure 4.1 shows the whole architecture of the visualization application. The architecture contains a web app and a server, which are connected with the WebSocket [24]. The WebSocket technology makes the real-time communications between the

web app and the server possible, as the visualization of the real-time mining processes is necessary. The user can interact with the web application through the graphical user interface.

The server maintains the blockchain system. It is composed of three components.

- **Blockchain System**
  The blockchain system is based on a multi-agent system. It contains a transaction generator, multiple miners, and multiple nonminers. The multi-agent system provides communications for these nodes.
- **Simulator**
  The simulator is responsible for receiving data from the client side and controlling the blockchain system.
- **Watchdog**
  When the blockchain data structures change in the blockchain system, the watchdog will catch these changes, and send them to the web app.

Since the nodes on the blockchain network can be regarded as agents on a multi-agent system, we decided to construct the blockchain system with a multi-agent system framework. Here, We choose a web-based multi-agent system framework called Eve [25]. The main features of Eve are stated below.

- **Robustness**
  The agents are decoupled and can be maintained independently. Thus, the whole system is still active even an agent fails.
- **Flexibility**
  It is flexible to add or remove agents.
- **Reduced complexity**
  The complexity of the distributed software design is lower.
- **Scalability**
  The limitation of the number of agents does not exist, so the scalability can grow rapidly.

In addition, Eve provides reliable communications with JSON-RPC protocols [26]. Therefore, agents can send and understand the JSON format data between each other, and they can be maintained independently in anywhere, e.g., on the cloud or desktops. Eve is an open source project which is implemented in JavaScript, and it is easy to learn. As a result, Eve allows us to construct a distributed blockchain system without worrying about the technical problems such as asynchronous and locking problems.

The server is built in Node.js with Express framework because they are based on the same programming language of Eve. During the runtime, a transaction generator,

miners, and nonminers can be instantiated and destroyed at any time thanks to the flexibility of Eve.

The web application is responsible for the visualization and the interactions with users. There are two components in the web application.

- **Visualizer**
  The visualizer is responsible for rendering the visualization of blockchains when it is notified by the watchdog. It receives the data of transactions and blocks from the server and renders them appropriately on the HTML documents.
- **Graphical User Interface**
  The GUI provides an index page and a settings page. The index page is for the visualization part, and the settings page displays all the defined parameters in the blockchain system. More details can be referred to Section 5.1.

The visualization is based on Three.js, a JavaScript framework for rendering 2D and 3D graphics. Three.js uses WebGL as the renderer, and the updating performance is 60 frames per second. The reason to choose Three.js as the visualization framework is that it is a famous open source project, and the community is active. Moreover, Three.js provides basic shapes, e.g., squares and lines, and handles the updating of frames. Therefore, we can focus on optimize the user experience of the visualization without considering the implementation of WebGL.

4.2 Simulator

The simulator is responsible for controlling the blockchain system and handling all the requests from the client. It is unique in the whole system, so it follows the singleton pattern. The role of the simulator is like a "door" as it is the only gate to communicate with the blockchain system.

After the blockchain system starts, the simulator initializes the status of the blockchain system, i.e., it instantiates a unique transaction generator, multiple miners, and multiple nonminers according to the user configuration. The waiting list of transactions in the transaction generator and the blockchain data structures of miners and nonminers are also initialized by the simulator.

When the user wants to change the parameters, i.e., the mining strategy and network delay, or the visualizer needs the information of transaction pools and blockchains, the simulator will interact with the blockchain system by setting the parameters or retrieving the required data. The data are sent through the WebSocket in real-time.

The simulator is an important component because it hides the details of the implementation of the blockchain system and provides a unique and robust interface for the requests from the client. As a result, it decouples the relationship between the web application and the blockchain system. The detail of the simulator can be referred to 4.5.

4.3 Watchdog

The watchdog is responsible for monitoring the behaviors of nodes and notifying the visualizer when the data should be updated. It is unique in the whole system, so it also follows the singleton pattern.

The watchdog watches three kinds of data in the blockchain system.

- blockchain data structures
- status of transaction pools
- status of mining

While a miner receives a transaction or a block, a nonminer receives a block, or a miner is solving the puzzle, the watchdog will notify the visualizer in real-time through the WebSocket. Therefore, it is guaranteed that the visualization of the data is synchronized with the blockchain system all the time.

With the help of the watchdog, the activities that happened in the blockchain system is recorded completely. Hence, it ensures the correctness of the visualization application. The detail of the watchdog can be referred to 4.5.

4.4 Visualizer

The visualizer is responsible for rendering the data that are sent from the watchdog. It is the main part of the visualization application because it renders the real-time visualization.

Three items are updated by the visualizer continuously, which are marked in Figure 4.2.

1. status bars for transaction pools
2. status bars for mining
3. blockchain data structures
4. total reward

The status bars for transaction pools and the status bars for mining are auxiliary tools to help the user understand the events that are actually happening in the mining activities. The status bar for the transaction pool becomes longer if the number of pending transactions is growing and vice versa. The status bar for mining is usually empty, except that the miner is solving the puzzle. It increases each second

Figure 4.2: Interface of the visualizer.

when the mining activity continues. Unlike the real blockchain networks, the time of solving the puzzles is predictable in our visualization application, and it is defined in the parameter of mining strategy. The main area of the visualization is reserved for the blockchain data structures. Squares represent blocks, and they are chained by lines. Different colors of the squares represent the different sources of the blocks. The growth of the blockchain data structures is dynamic and real-time as the mining activities are in progress. The total reward of the miner increases when she adds the block to the longest blockchain.

The visualizer expresses fantastic blockchain visualization that is able to attract the user. In addition, it explains the steps of mining processes understandably, even for the viewers who are new to blockchain technology. The detail of the visualizer can be referred to 4.5.

4.5 UML Diagram

Figure 4.3 demonstrates the class diagram of the visualization application. The blockchain system is based on the multi-agent system framework Eve. Abstract class `AbstractNode` inherits from class `Agent` in Eve, and class `TransactionGenerator`, class `Miner`, and class `Nonminer` inherit from it. The blockchain system is controlled by class `Simulator`, and class `Watchdog` monitors it and notifies class `Visualizer` when it is necessary to update. Class `GUI` provides the user interface and is able to interact with `Simulator`. Finally, a helper class `Hash` provides simulation functionalities of the hash computation. The following tables contain all the classes that compose the visualization application.

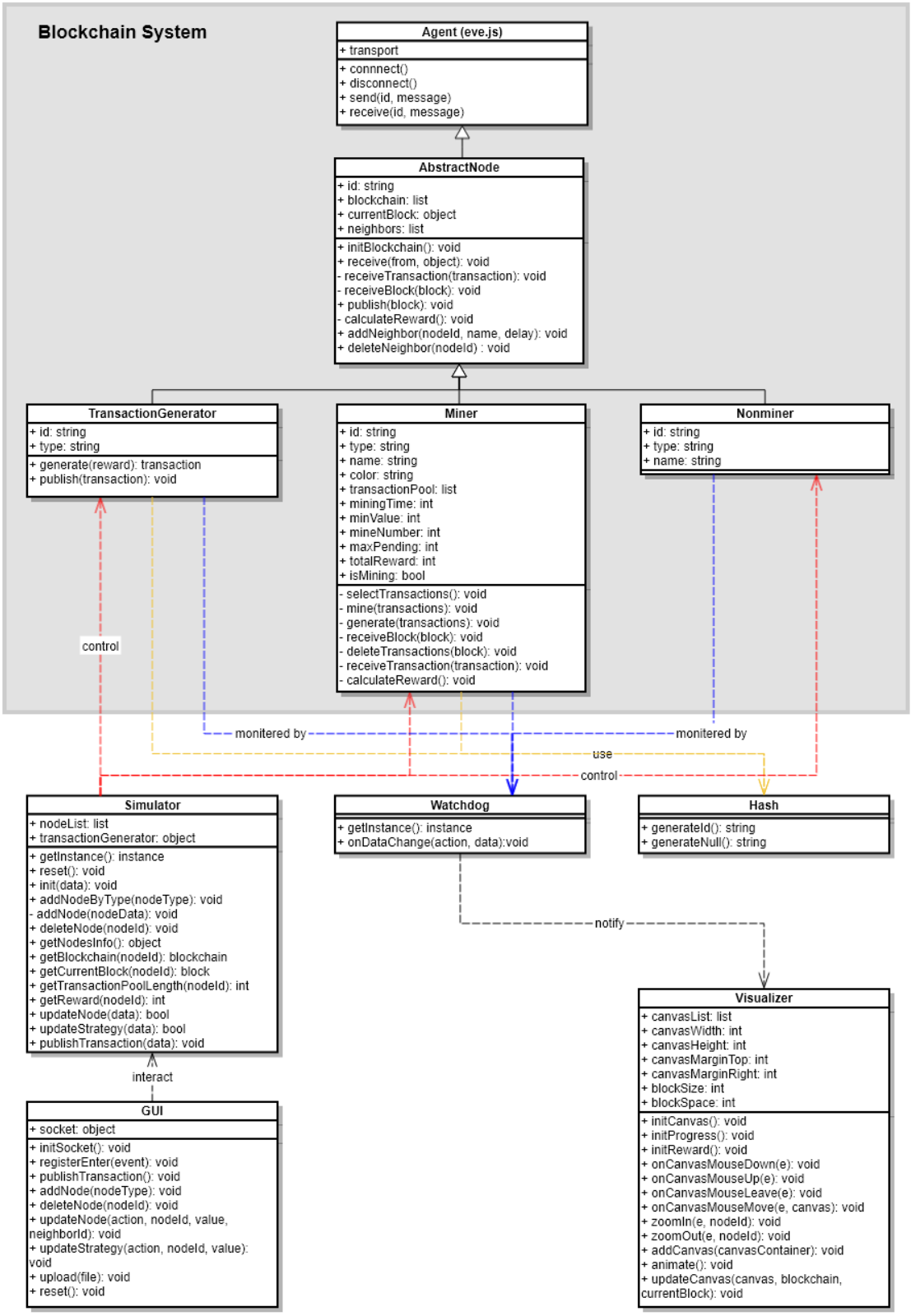


Figure 4.3: Class Diagram.

- **Agent** (Table 4.1)
  `Agent` is from the multi-agent system framework, Eve. It establishes the communication channel for the agents. In the implementation, the channel is opened with HTTP protocols. It makes sending and receiving transactions and blocks very simple.

Agent (eve.js)	
Properties	*Description*
transport	communication channels.
Methods	*Description*
connect	connect with a transport.
disconnect	disconnect from a transport.
send	send messages to a transport.
receive	receive messages from a transport.

Table 4.1: Class `Agent`

- **AbstractNode** (Table 4.2)
  It defines the common properties and methods of its children. `AbstractNode` is an abstract class, so it is not allowed to instantiate instances from it. In contrast, we can create nodes from the children of `AbstractNode`, i.e., `TransactionGenerator`, `Miner`, and `Nonminer`

AbstractNode	
Properties	*Description*
id	unique in the multi-agent system.
blockchain	blockchain data structures.
currentBlock	the id of the block which is on the top of the longest blockchain.
neighbors	reachable nodes.
Methods	*Description*
initBlockchain	initialize blockchain data structures.
receive	inherit from `eve.js`.
receiveTransaction	define actions of receiving transactions.
receiveBlock	define actions of receiving blocks.
publish	publish transactions or blocks.
calculateReward	calculate the total mining rewards.
addNeighbor	add a new node as a neighbor.
deleteNeighbor	delete a node from neighbors.

Table 4.2: Class `AbstractNode`

- **TransactionGenerator** (Table 4.3)
  It is responsible for generating and publishing transactions. There should be only one transaction generator in the blockchain system. Furthermore, the neighbors of the transaction generator only contain the miners.

TransactionGenerator	
Properties	*Description*
id	unique in the multi-agent system.
type	is always "generator".
Methods	*Description*
generate	generate a transaction.
publish	publish a transaction.

Table 4.3: Class `TransactionGenerator`

- **Miner** (Table 4.4)
  `Miner` is the main role in the blockchain system. It generates and publishes blocks through the blockchain network. Therefore, it controls all the mining processes in the blockchain system.

Miner	
Properties	*Description*
id	unique in the multi-agent system.
type	is always "miner".
name	a simple nickname.
color	the color of the generated blocks.
transactionPool	contain pending transactions.
miningTime	computing power.
minValue	minimum value of transactions.
mineNumber	size of blocks.
maxPending	maximum pending transactions.
totalReward	total number of rewards.
isMining	whether the miner is mining.
Methods	*Description*
selectTransactions	select pending transactions as candidates.
mine	solve the puzzles.
generate	generate a block instance.
receiveBlock	inherit from `AbstractNode`.
deleteTransactions	delete mined transactions from the transaction pool.
receiveTransaction	inherit from `AbstractNode`.
calculateReward	inherit from `AbstractNode`.

Table 4.4: Class `Miner`

- **Nonminer** (Table 4.5)
  `Nonminer` represents common users who do not solve complex mathematical puzzles in the blockchain system. The only behavior of nonminers is to receive and publish blocks. Consequently, the miners and the nonminers connect with each other.

Nonminer	
Properties	*Description*
id	unique in the multi-agent system.
type	is always “nonminer”.
name	a simple nickname.

Table 4.5: Class `Nonminer`

- **Simulator** (Table 4.6)
  `Simulator` is the only master of the blockchain system. It is able to add and update nodes, and retrieve information from the blockchain system.

Simulator	
Properties	*Description*
nodeList	a list of all nodes.
transactionGenerator	reference to the transaction generator.
Methods	*Description*
getInstance	get the instance of the simulator.
reset	reset the blockchain system.
init	initialize the blockchain system.
addNodeByType	add a new node with defined type.
addNode	add a new node.
deleteNode	delete a node.
getNodesInfo	get the information of a node.
getBlockchain	get the blockchain data structure of a node.
getCurrentBlock	get the information of the current block of a node.
getTransactionPool-Length	get the level of capacity of the transaction pool of a node.
getReward	get the number of rewards of a node.
updateNode	update the information of a node.
updateStrategy	update the parameters of the mining strategy of a node.
publishTransaction	publish a transaction through the transaction generator.

Table 4.6: Class `Simulator`

- **Watchdog** (Table 4.7)
  `Watchdog` monitors the blockchain system. When a mining activity starts, it keeps notifying the visualizer until the end.

Watchdog	
Methods	*Description*
getInstance	get the instance of the watchdog.
onDataChange	monitoring the data.

Table 4.7: Class `Watchdog`

- **Visualizer** (Table 4.8)
  `Visualizer` uses 3D graphical library to render the blockchain visualization. It maintains all miners' and nonminers' canvases and the mouse events which enable users to drag, zoom in, and zoom out the canvases.

Visualizer	
Properties	*Description*
canvasList	a list of all canvases.
canvasWidth	the width of each canvas.
canvasHeight	the height of each canvas.
canvasMarginTop	the top margin of each canvas.
canvasMarginRight	the right margin of each canvas.
blockSize	the size of blocks.
blockSpace	the spaces between blocks.
Methods	*Description*
initCanvas	initialize all canvases.
initProgress	initialize progress bars.
initReward	initialize the numbers of rewards.
onCanvasMouseDown	mouse event.
onCanvasMouseUp	mouse event.
onCanvasMouseLeave	mouse event.
onCanvasMouseMove	mouse event.
zoomIn	zoom in with mouse.
zoomOut	zoom out with mouse.
addCanvas	add a new canvas.
animate	update by each frame.
updateCanvas	update a canvas.

Table 4.8: Class `Visualizer`

- **GUI** (Table 4.9)
  The graphical user interface is responsible for the interactions with users, including uploading files, updating the blockchain system, publishing transactions, and resetting the blockchain system.

GUI	
Properties	*Description*
socket	the instance of WebSocket.
Methods	*Description*
initSocket	initialize WebSocket.
registerEnter	register enter keyboard.
publishTransaction	publish a transaction.
addNode	add a new node.
deleteNode	delete a node.
updateNode	update a node.
updateStrategy	update the parameters of the mining strategy of a node.
upload	upload a configuration file.
reset	reset the blockchain system.

Table 4.9: Class `GUI`

- **Hash** (Table 4.10)
  This class generates a sequence of random numbers to simulate the hash functions. It avoids putting lots of computing resources to calculate hash values.

Hash	
Methods	*Description*
generateId	generate a hash value for id.
generateNull	generate a hash value only with zeros.

Table 4.10: Class `Hash`

4.6 Algorithms

The visualization focuses on the mining processes, i.e., generating and publishing blocks. Hence, the algorithm about the mining processes is the key parts of the entire system. The algorithm can be divided into three parts.

- Select Candidate Transactions
- Mining
- Add Block to Blockchain

Before a miner starts to mine a block, she needs to select a set of candidate transactions for the transaction pool. Thus, these candidate transactions serve as the content of the mined block. After receiving a block, the node will check the layer of the new block, and switch to the new block if the blockchain becomes the longest.

Algorithm 1 states the procedure of selecting a set of candidate transactions. Assume the pending transactions in the transaction pool are sorted according to their values. There are two cases at the beginning of the algorithm. *Maximum number of pending transactions* is defined in the parameters of the mining strategy.

1. **the number of pending transactions in the transaction pool $\geq$ maximum number of pending transactions**
   Because the mining strategy does not allow appending more pending transactions into the transaction pool, the miner must select a set of pending transactions. *Minimum value of transactions*, a parameter of mining strategy, is ignored here. Consequently, the miner will continue selecting the transaction with the highest value in the transaction pool, until the number of candidate transactions is enough for mining a block.
2. **the number of pending transactions in the transaction pool $<$ maximum number of pending transactions**
   In this case, the miner selects transactions which values are higher than the *minimum value of transactions*.

In both cases, if a transaction is not selected as a candidate transaction, then the privilege of the transaction will increase. Finally, if the number of selected candidate transactions is equal to the *number of mined transactions*, then the miner will start to mine a block. Otherwise, the miner will put the candidate transactions back into the transaction pool and wait later.

Algorithm 1 Select Candidate Transactions

```
1: procedure SELECT TRANSACTIONS
2:     candidateTransactions := ∅
3:
4:     if transactionPool.length ≥ maximum number of pending transactions then
5:         for all pending transaction do
6:             if candidateTransactions.length <
7:             number of transactions to be mined then
8:                 candidateTransactions ∪ this transaction
9:                 remove this transaction from transaction pool
10:            else
11:                add the privilege of this transaction by 1
12:            end if
13:        end for
14:    else if transactionPool.length ≥ number of transactions to be mined then
15:        for all pending transaction do
16:            value ← reward + privilege
17:            if value > minimum value of transactions AND
18:             candidateTransactions.length < number of transactions to be mined
   then
19:                candidateTransactions ∪ this transaction
20:                remove this transaction from transaction pool
21:            else
22:                add the privilege of this transaction
23:            end if
24:        end for
25:    end if
26:
27:    if candidateTransactions.length = number of mined transactions then
28:        MINE(candidateTransactions)
29:    else
30:        add candidateTransactions to pending transactions
31:    end if
32:
33:    notify watchdog
34: end procedure
```

Regarding the mining algorithm (Algorithm 2), it simulates the process of solving a puzzle in the proof-of-work based blockchain system. After determining the candidate transactions, the miner starts a timer until it meets the duration of time that should be spent on mining, i.e., the miner solves the puzzle successfully. Thus, the miner is ready for generating and publishing the block.

Algorithm 2 Mining

```
1:  procedure MINE(candidateTransactions)
2:      count ← 0
3:      isMining ← true
4:
5:      repeat
6:          count += 1
7:          notify watchdog
8:      until count ≥ mining time
9:
10:     isMining ← false
11:     block ← GENERATE(candidateTransactions)
12:     RECEIVEBLOCK(block)
13:     notify watchdog
14: end procedure
```

Receiving a new block triggers the consensus protocol to resolve the longest blockchain (Algorithm 3). There are two cases that will happen when receiving a new block.

1. **the ID of the previous block of the new block is empty**
   It happens because the new block was mined by the miner herself. Therefore, the miner puts the new block on the top of the current blockchain, i.e., the miner adds the new block after the current block.

2. **the ID of the previous block of the new block is not empty**
   The consensus protocol is very simple because we do not consider malicious nodes in the blockchain system due to simulation. If the received block is at a higher layer, i.e., the blockchain becomes longer, then the node will switch the current block to the received block to ensures that she is working on the correct blockchain. Moreover, the transactions of the received block are deleted from the transaction pools because these transactions are not pending anymore.

Last, the node publishes the received block through the blockchain network to make sure that every neighbor has the same blockchain data structure. In addition, the miner calculates and updates her total rewards when the blockchain becomes longer.

Algorithm 3 Add Block to Blockchain

```
1:  procedure RECEIVE BLOCKS(block)
2:      if block.previous = none then
3:          block.previous ← currentBlock.id
4:          block.layer ← currentBlock.layer + 1
5:          currentBlock ← block
6:          delete the mined transactions from transaction pool
7:      else if currentBlock.layer < block.layer then
8:          currentBlock ← block
9:          delete the mined transactions from transaction pool
10:     end if
11:
12:     add block to blockchain
13:     publish block to neighbors
14:     calculate the total rewards
15:     notify watchdog
16: end procedure
```

Chapter 5

Application

In this chapter, we introduce the usage of the visualization application. It begins with a simple example to demonstrate the useful features of the visualization application. Finally, the explanation of the configuration files is provided.

5.1 Flow

The general flow of the visualization application is guided here. In the beginning, the blockchain system is empty, and the user can either upload a configuration file or add nodes manually (Figure 5.1). The introduction starts with creating nodes and publishing transactions manually, as it is clearer to begin with a simple example.

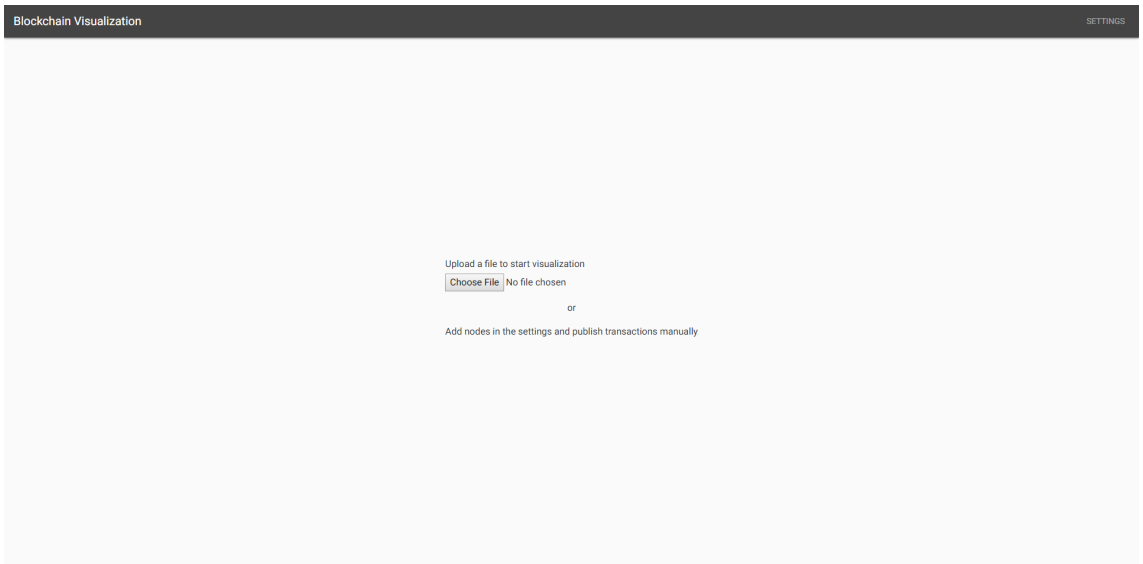


Figure 5.1: Start of the Application.

By clicking the top right button in the navigation bar, the user will be redirected to the settings page, as Figure 5.2 shows.

The settings page contains all the nodes that exist on the blockchain network. How-

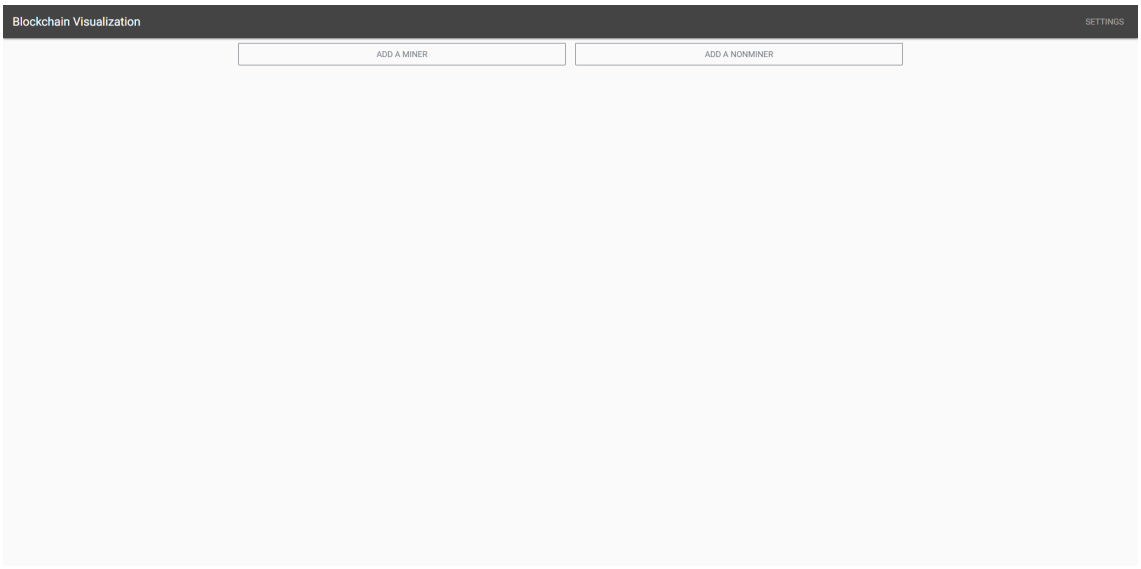


Figure 5.2: Settings Page.

ever, the blockchain system is empty currently, so the settings page is empty. To add a node, the user can click "ADD A MINER" button or "ADD A NONMINER" button. In this example, we add three miners and two nonminers.

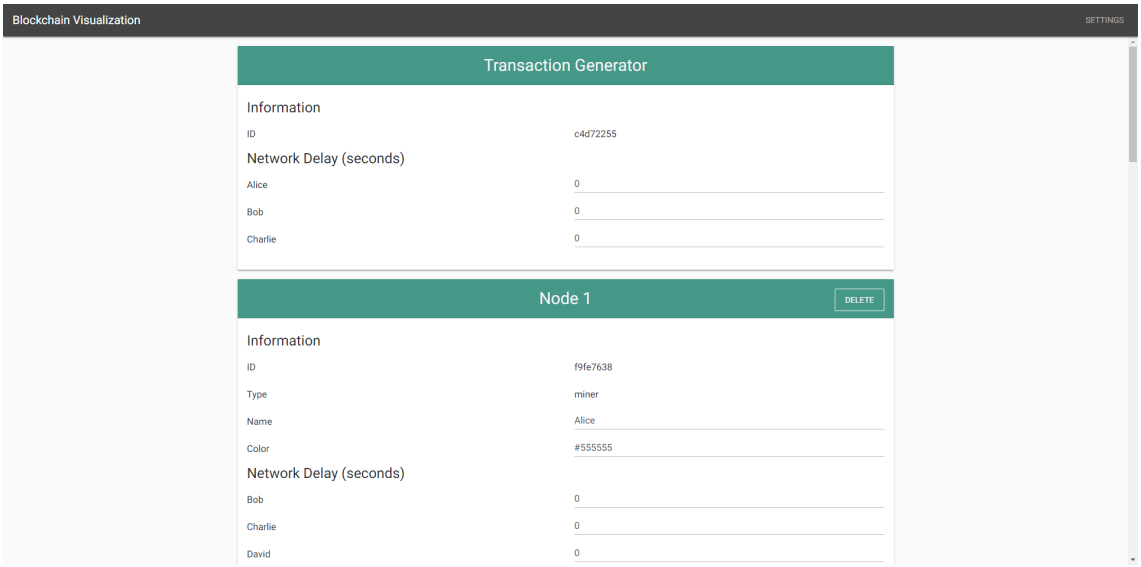


Figure 5.3: Settings of Transaction Generator.

After creating nodes, the settings page contains three miners and two nonminers now. Additionally, the transaction generator is added to the blockchain system automatically. In Figure 5.3, the properties of the transaction generator are displayed in the most top block. The properties contain the ID and the network delay to its neighbors.

By scrolling down the settings page, the miners that exist in the blockchain system are displayed first. The parameters of the miners can be divided into three parts, as Figure 5.4 shows.

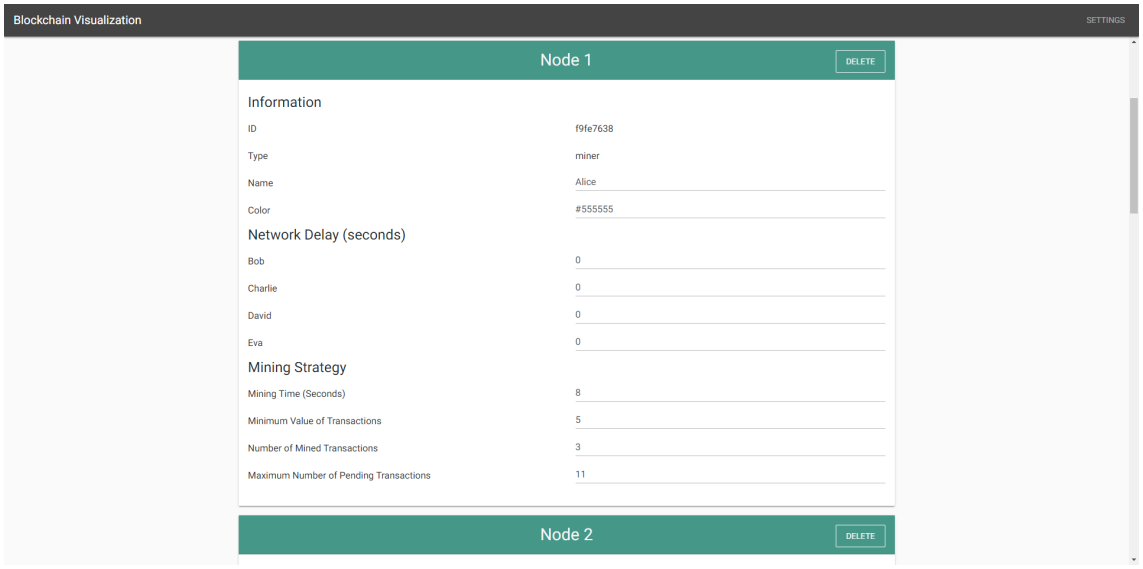


Figure 5.4: Settings of Miner.

- **Information**
  The properties of the miner, including the ID, the name, and the represented color.
- **Network Delay**
  It defines the periods of delay to the neighbors.
- **Mining Strategy**
  This part contains the four parameters that are used for the mining strategy.

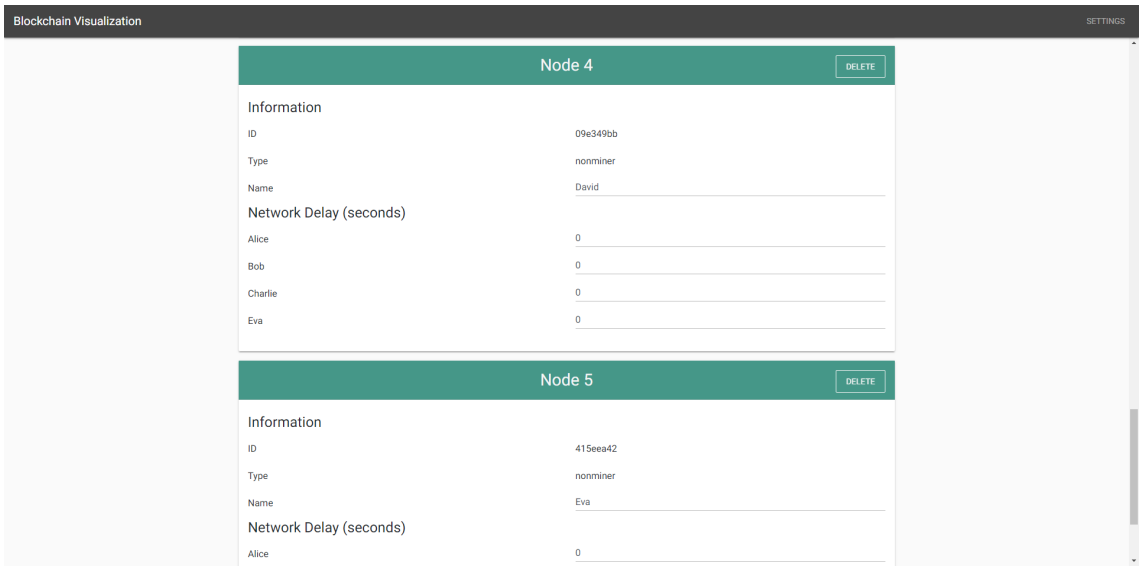


Figure 5.5: Settings of Nonminer.

The last part of the settings page is for the nonminers. Because nonminers do not mine a block by themselves, the represented color and the parameters of mining strategy are missing here, as Figure 5.5 shows.

The user can give miners and nonminers nicknames such as Alice, Bob, etc., to make the relationship between the nodes more understandable. The nicknames may not be unique in the blockchain system for the reason that the nodes are recognized by their unique IDs. Moreover, the parameters are updated automatically while the user is typing. All the parameters are generated randomly at first, so it is better to check and set each parameter manually.

In this example, we set Alice (red color), Bob (blue color), and Charlie (green color) as miners, and David and Eva as nonminers. For the mining strategy, we set the following three parameters to the same number for all miners.

- Mining Time (Seconds): 1
- Number of Mined Transactions: 1
- Maximum Number of Pending Transactions: 10

For the parameters of *minimum value of transactions*, we set 1 for Alice, 5 for Bob, and 9 for Charlie. Therefore, Alice is expected to mine blocks faster than Bob and Charlie. For the network delay, we set all the parameters to 1 second.

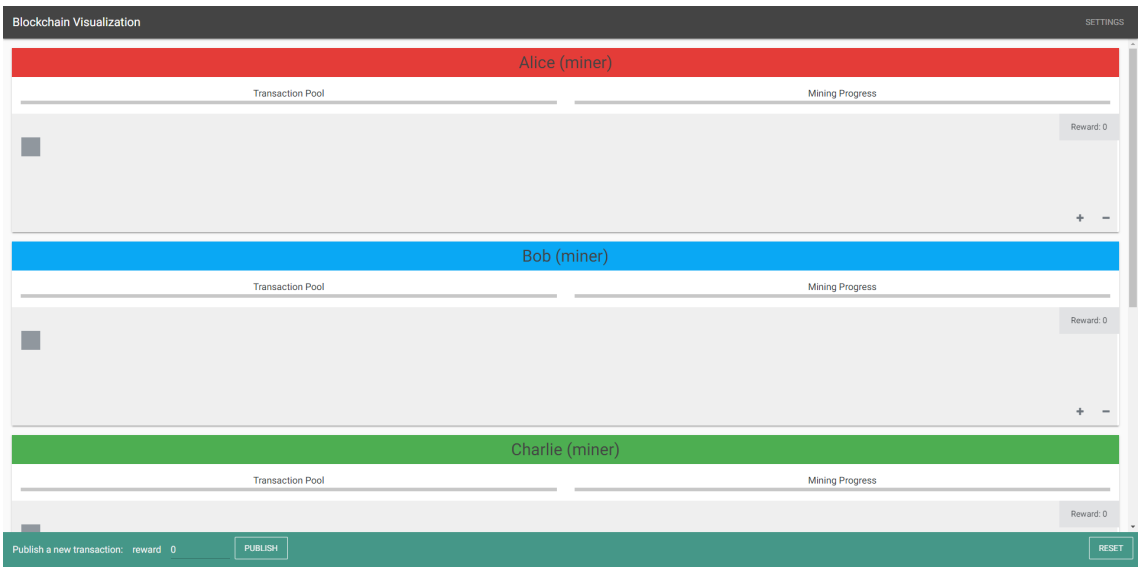


Figure 5.6: Initial status.

After finishing the configuration, the user can see the initial status of the blockchain system like Figure 5.6. Each row represents the visualization of a miner or nonminer. A miner has a colorful head and two progress bars which display the status of the transaction pool and the mining activity. At the bottom of each row, it is the main area for visualizing the blockchain data structure. In the beginning, the grey block represents the genesis block.

The user can publish a transaction at the bottom area of the screen. By entering a number of the reward, a transaction will be generated by the transaction generator and published. In the first step, we publish a transaction which reward is 1. Hence,

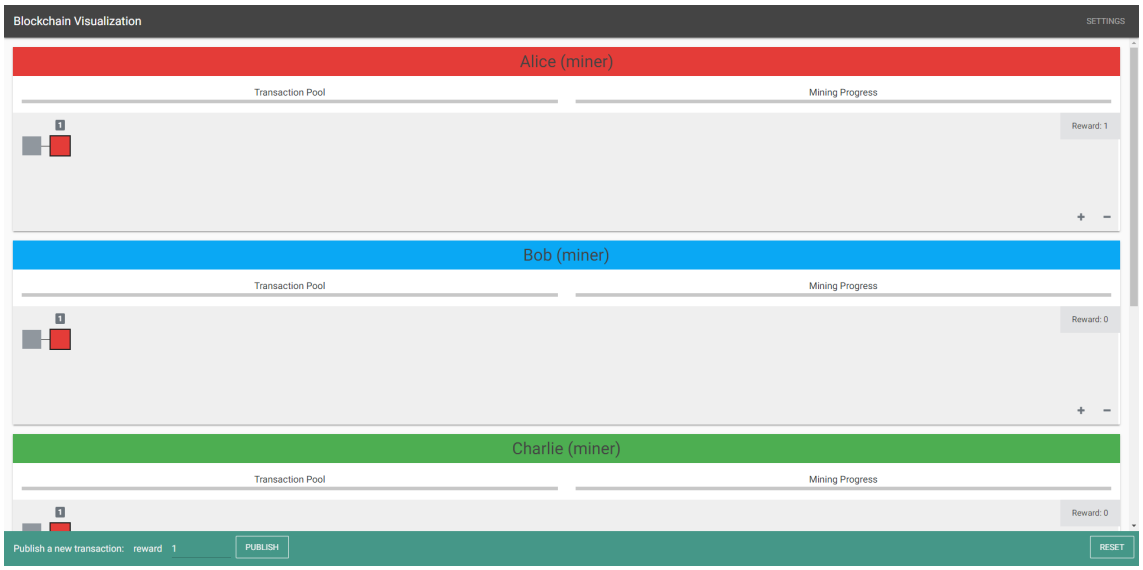


Figure 5.7: Alice mined a block.

it is expected that only Alice will mine a block because the *minimum value of the transactions* is 1 for Alice, and the necessary *number of mined transactions* is also 1. The result is showed in Figure 5.7. Moreover, now Alice's total reward is 1 because she added a block to the longest blockchain successfully.

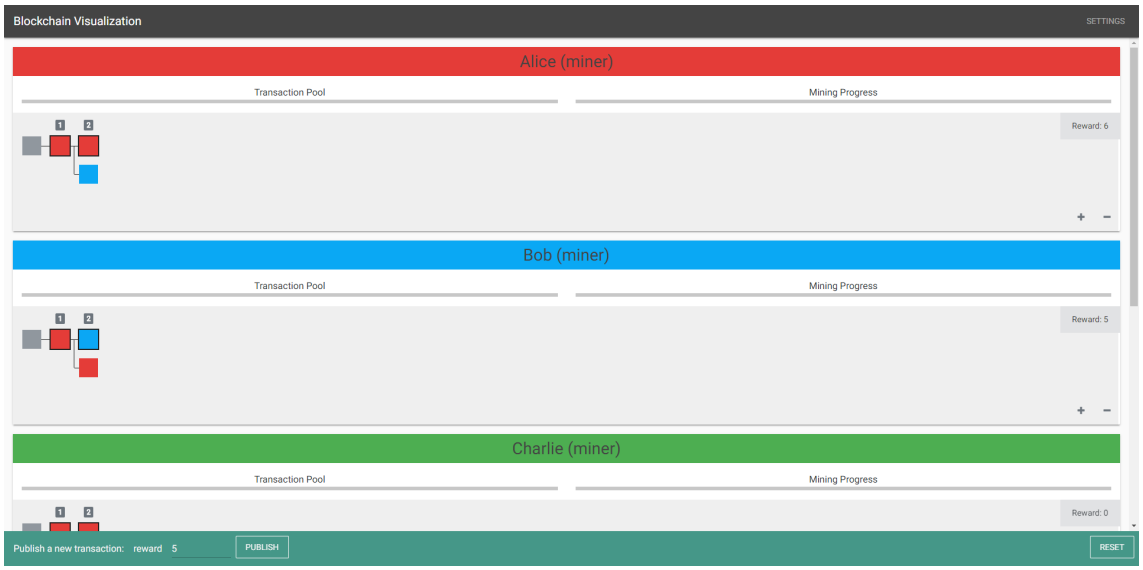


Figure 5.8: Alice and Bob mined a block simultaneously.

Figure 5.8 demonstrates that Alice and Bob competed with each other by adding a block at the same time. Because we published a transaction with the reward of 5, Alice and Bob both decided to mine a block according to their mining strategies, i.e, the reward of the transaction is not less than their *minimum value of the transactions*. As a result, the fork happened for the reason of simultaneous mining activities.

In the next step, Figure 5.9 shows the influence of the mining strategy to the mining processes. We changed the parameters of the mining strategy at first. Thus, Alice's

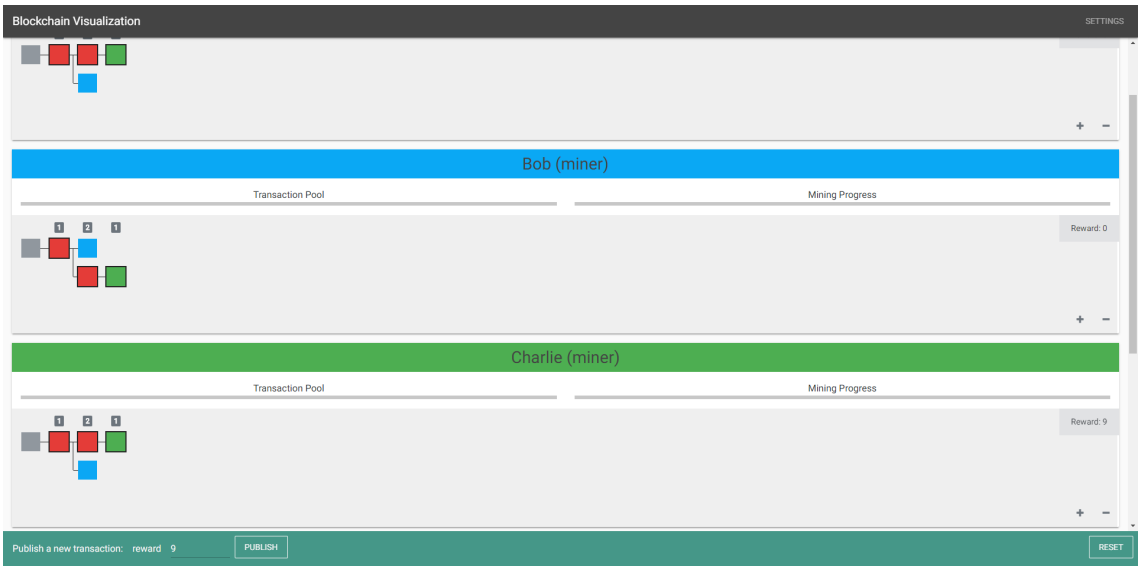


Figure 5.9: Charlie mined a block.

and Bob's parameters of *number of mined transactions* were both changed to 2. After that, a transaction with the reward of 9 was published. The result was that Charlie mined a block for the reason that it satisfied Charlie's mining strategy, i.e, the *number of mined transactions* is only 1 for Charlie. On the other hand, Alice and Bob did not mine a block because of the insufficient number of transactions.

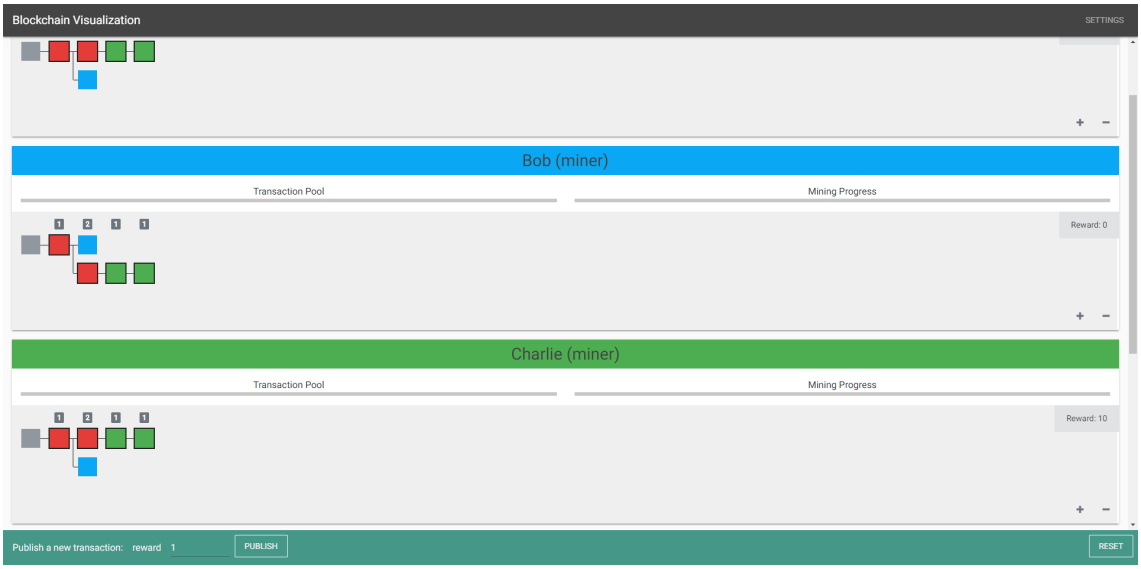


Figure 5.10: Charlie mined another block.

Figure 5.10 shows the process of selecting candidate transactions. We published a transaction with the reward of 1. The result was that Charlie mined a block after about 8 seconds due to the process of selecting candidate transactions. Normally, a miner starts to select candidate transactions in every second, and the privileges of transactions that are not selected are added by 1 each time. Consequently, the value of the published transaction was becoming 9 after 8 seconds, and it satisfied Charlie's mining strategy, i.e., the *minimum value of the transactions*.

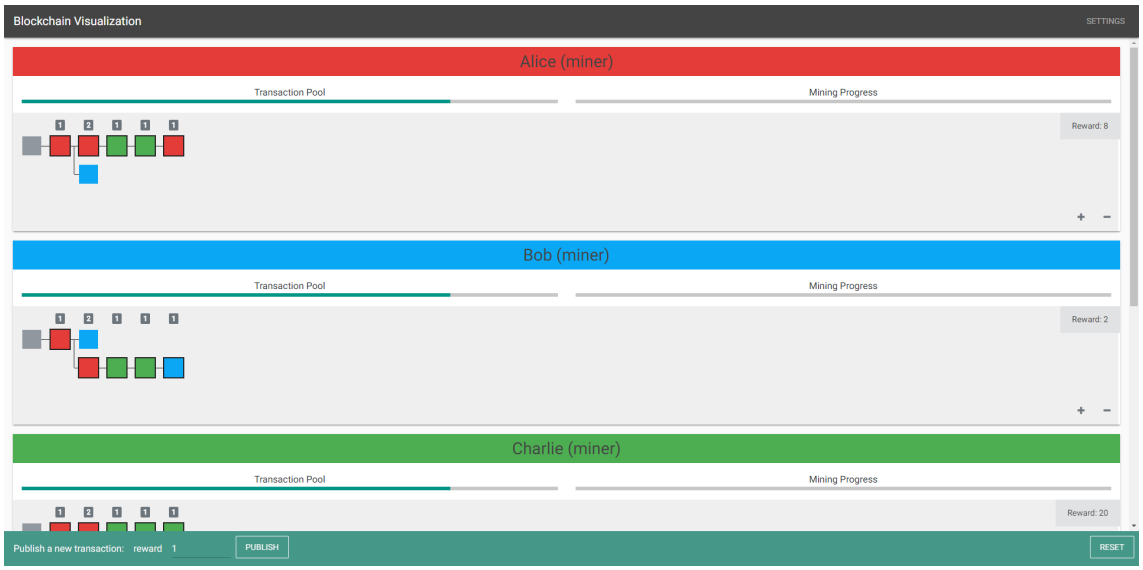


Figure 5.11: Overflow of transaction pools.

Finally, Figure 5.11 shows the overflow of the transaction pools. First, the parameters of *minimum value of the transactions* were all set to 15 for the three miners, and then we published multiple transactions with the reward of 1. Because of a large number of transactions with low reward is published, the transaction pools became full in a short time, and it was displayed in the progress bars for transaction pools. As mentioned in the mining algorithm, all the three miners are forced to mine a block due to the overflow of the transaction pools, i.e., the number of pending transactions is larger than 10 in this example. The result can be checked in Figure 5.12, and it is the same as the expectation that all miners mine blocks simultaneously.

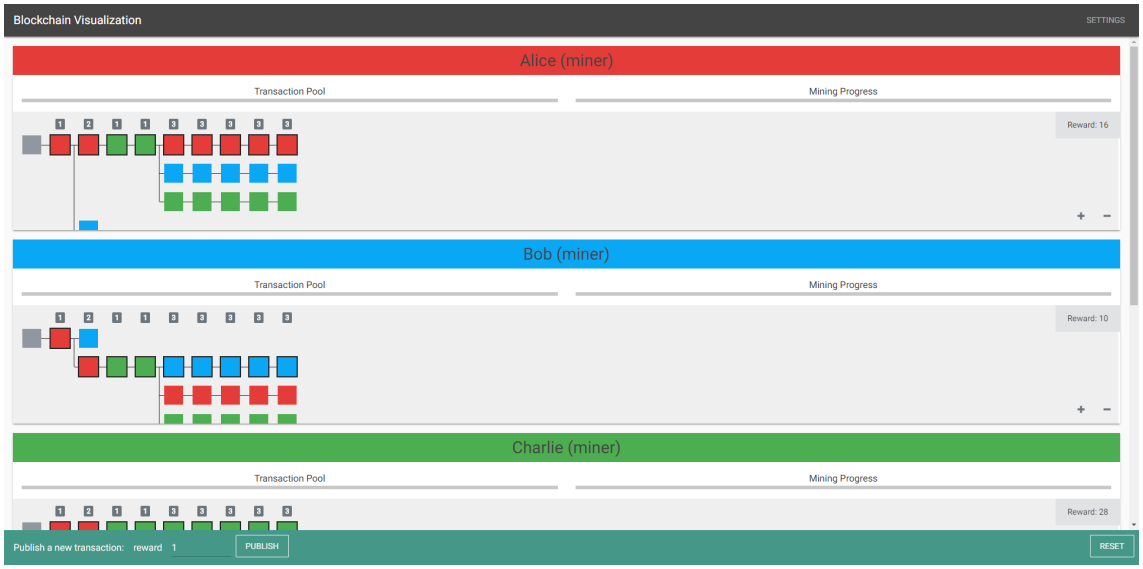


Figure 5.12: Alice, Bob, and Charlie were forced to mine.

During the mining processes, the following auxiliary information in the visualization is also helpful and dynamic.

- **Progress bars for the transaction pool**
  It displays the level of capacity of the transaction pool.
- **Progress bars for the mining status**
  It presents the mining activities in real-time, i.e., how much time is remaining for solving a puzzle.
- **Number of total rewards**
  It displays the calculation of the total reward that belongs to the miner.
- **Number of forks**
  On the top of each fork, there is a number that represents the number of forks.

Additionally, the user can drag the visualization area to move the blockchain data structure, and zoom in and out by clicking the plus and minus buttons on the bottom right side.

5.2 Configuration Files

To replay the same visualization of blockchain processes, we can define the configuration of the blockchain system in a file and upload it to the application as in Figure 5.1. The configuration file is composed of the following three parts.

- the properties and the parameters of the mining strategy of nodes
- the network delay between each node
- the transactions that will be published through the network

We provide a sample configuration file in Listing B.1. The file is in JSON format, and the file is used to initialize the blockchain system in the beginning.

First, for nodes, it is important to define a unique transaction generator at the beginning of the file, and then it is followed by miners and nonminers. The required properties for the transaction generator, miners, and nonminers are provided as the following.

1. **Transaction Generator**
    - id
    - type
2. **Miner**
    - id
    - type
    - color
    - name
    - miningTime
    - minValue
    - mineNumber
    - maxPending
3. **Nonminer**
    - id
    - type
    - name

Thd *id* of nodes can be a string with arbitrary length, and here we use 8 characters and digits to represent the identifier. The *type* distinguishes different nodes, and the valid values are *generator*, *miner*, and *nonminer*. The *name* of miners and nonminers are nicknames for them. Miners also have unique *colors* to differentiate blocks in the visualization. The parameters of *miningTime*, *minValue*, *mineNumber*, and *maxPending* are from the mining strategy, which are *mining time*, *minimum value of transactions*, *number of mined transactions*, *minimum number of pending transactions* respectively.

The second part of the configuration file is a list of delay between each node. Each node has a block that begins with the *id*, and then it is followed by a list of the neighbors and the amount of delay. As mentioned before, the transaction generator only has miners as neighbors, while miners and nonminers connect with each other. Moreover, the transaction generator is not in the neighbors of miners and nonminers because it is meaningless to send blocks to the transaction generator.

In the last part, it contains a list of transactions with the number of rewards and the delay after the starting of the blockchain system, i.e., we can arrange the transactions to be published in a sequence.

Chapter 6

Scenarios

In this chapter, we explore two different scenarios that can be achieved by the visualization application. The first demonstrates the influences of the mining strategy, and the second scenario also adds the influences of network delay to the visualization. In addition, the presentation of orphaned blocks and forks proves that the visualization application handles the mining processes correctly. In the last scenario, we explore the limitation of the visualization application.

6.1 Scenario I: A Fast Miner

In the first scenario, there is a fast miner who mines blocks much quicker than the other miners due to the mining strategy. With the low network delay, the fast miner can publish the blocks quickly. Therefore, none of the other miners can compete with the fast miner, i.e., only the fast miner can add blocks to the blockchain.

In this experiment, there are three miners, Alice (red color), Bob (blue color), and Charlie (green color), who compete with each other in the blockchain system. The configuration of this experiment is provided in Listing B.2.

As the configuration file shows, Alice's mining strategy is the fastest because the parameters of *mining time*, *minimum value of transactions*, and *number of mined transactions* are all set to 1. On the contrary, Bob's and Charlie's parameters are both higher than Alice's parameters. The parameters of network delay are all set to 1. Hence, the blockchain network is stable and fast and it only takes 1 second to publish the blocks to neighbors. For transactions, we defined a sequence of transactions which interval is 1 second.

Figure 6.1 demonstrates the results of the above configuration. Although some transactions have large number of rewards, Alice still dominated the blockchain system because of her fast mining strategy. As a result, all the 10 transactions were mined by Alice, even if Bob tried to mine a block at the seventh layer.

In summary, this scenario shows the competition between the miners. As our expec-

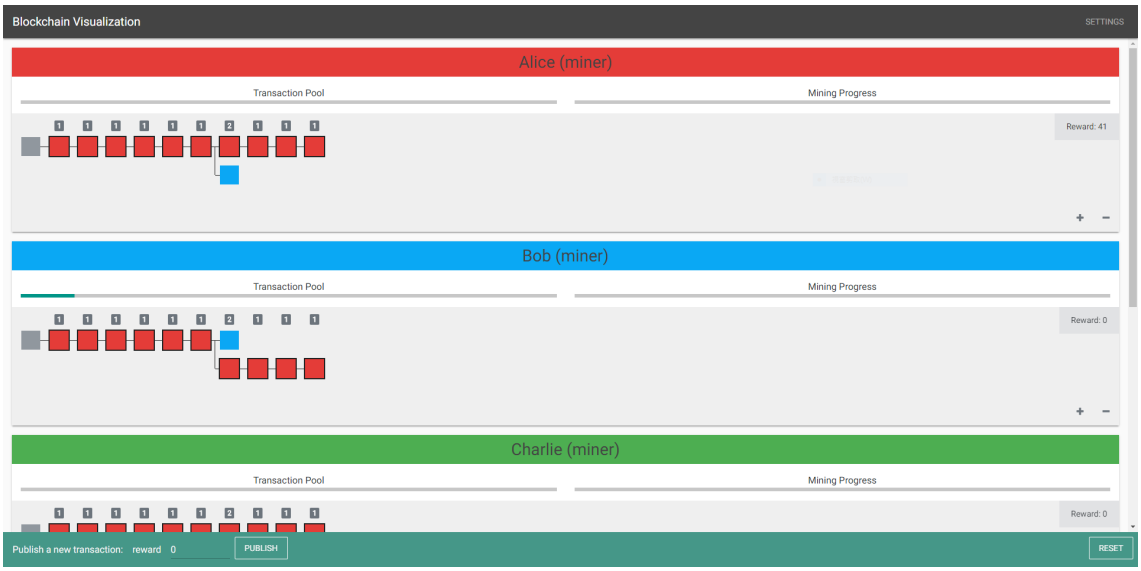


Figure 6.1: Alice was a fast miner.

tation, the mining strategy with the lowest *mining time*, *minimum value of transactions*, and *number of mined transactions* is the fastest one. Additionally, it shows that the orphaned blocks were handled by the visualization application correctly, as the fast miner still worked on the correct blockchain data structure.

6.2 Scenario II: Multiple Forks

In the second scenario, there are three miners who have the similar computing power and mining strategies, and compete with each other for a long time. Because the spread of blocks has high delay, miners add their own blocks to the blockchain much more quickly than others do. Thus, each miner considers her own blockchain is the longest one, and the competition continues. Listing B.3 is the configuration file for this scenario.

In the configuration file, Alice's, Bob's and Charlie's parameters of the mining strategy are set to the same value. Therefore, it is expected that the three miners will always mine at the same time. The parameters of the network delay are higher than the parameters of *mining time* to simulate the bad network conditions between three miners. The sequence of transactions is the same as Listing B.2.

In Figure 6.2, Alice, Bob, and Charlie compete with each other for a long time because the duration of mining time is shorter than the duration of publishing blocks. The miners are divided into three groups which work on different blockchain data structures because every miner considered that their own blockchain is the longest. It can be checked that the mining processes visualized the mining processes according to the above description, as Figure 6.3 shows.

In conclusion, it demonstrates that the visualization application is able to handle the forks correctly. Furthermore, the mining processes of the competition are also

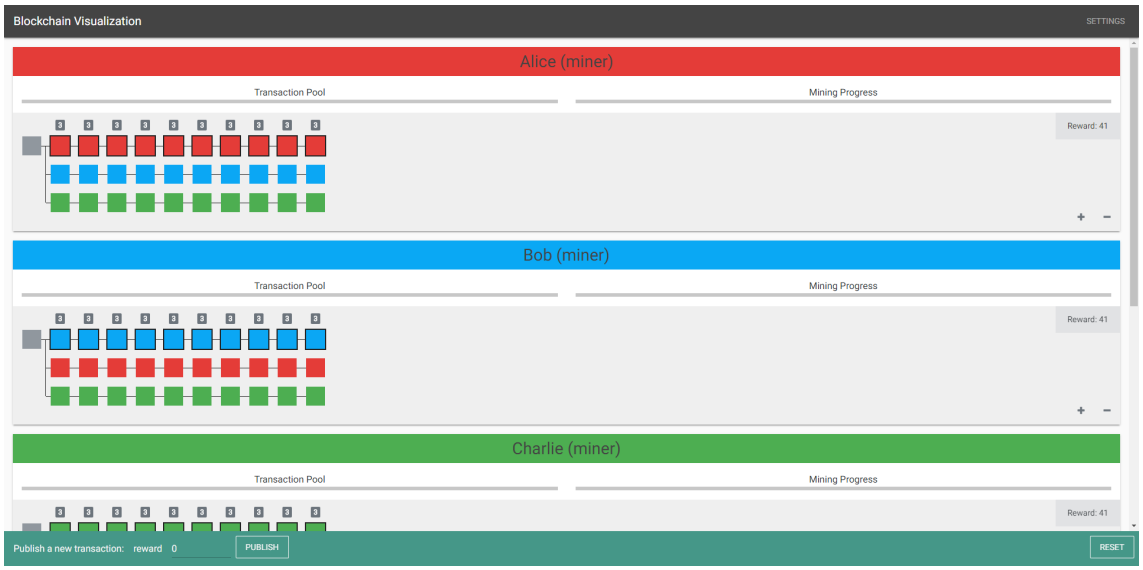


Figure 6.2: Alice, Bob, and Charlie competed with each other.

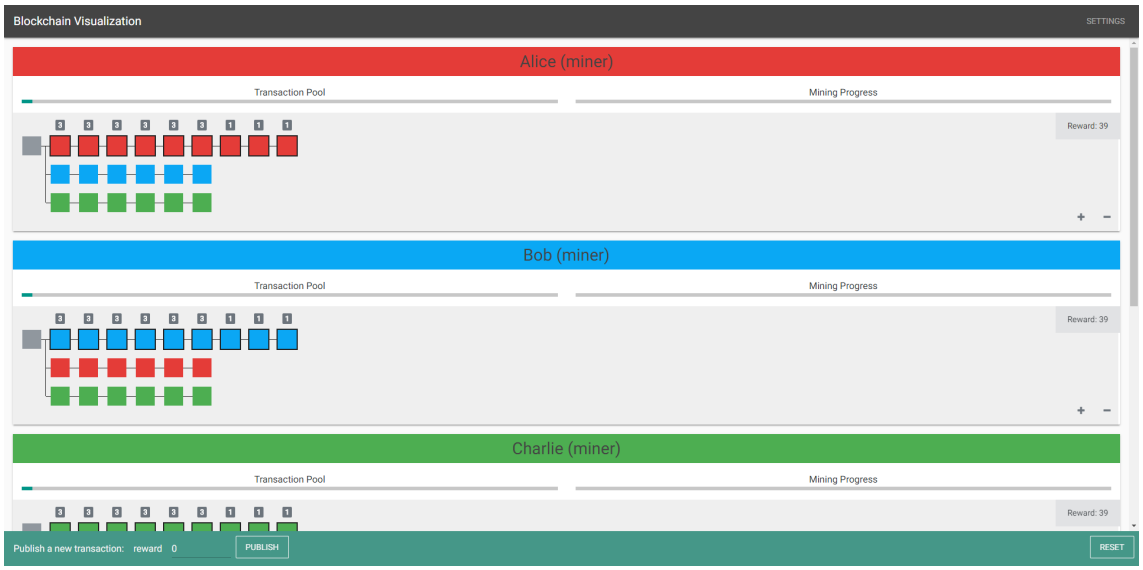


Figure 6.3: Mining processes of scenario II.

correct. However, there is a weak point in the visualization application. The forks cannot be merged in the current implementation because the parameters of the mining strategy are fixed during the visualization. Thus, the scenario of multiple forks in the visualization represents the case of the hard forks in the real blockchain networks. It should be improved to handle the case of the soft forks in the future.

6.3 Limitation of Nodes and Transactions

Through experiments, we found that the maximum number of nodes that can be handled by the application is 15 because of the limitation of the visualization engine, i.e., the Three.js library. The visualization engine cannot create too many canvases due to the performance issue. Hence, the sum of the number of miners and nonminers

is limited to 15. On the other hand, the number of transactions that can be handled should be unlimited. We tested it by publishing 100 transactions, and the application still operates correctly. Figure 6.4 shows the result of this experiment.

Figure 6.4: Result of the limitation experiment.

Chapter 7

Conclusion

7.1 Summary

This thesis proposes the first trial to visualize the complex and dynamic mining processes in the blockchain system. The visualization is based on the simulation of the blockchain system which is powered by the multi-agent system. With the watchdog which monitors the data in the blockchain system, the visualizer can update the visualization in real-time after it receives the data from the watchdog. The visualization of blockchain processes clearly indicates the mining activities step by step.

The visualization application helps researchers to conduct experiments and focus on the mining activities which are influenced by the network delay and the mining strategy in the blockchain system. By providing a configuration file which defines all the properties and parameters in the blockchain system, the same mining processes can be replayed by the visualization application. With the discussions of the scenarios, it proves the potential applications of the visualization application.

The process of this thesis begins with designing the visualization. To build a simple blockchain visualization, we decided to use colors to represent the source of the block. We did not consider different sizes of blocks, which are proved as a good visualization in other visualization applications such as Bitcoincity and Bitbonkers because it can help users to understand the rewards from mining. For transactions, we visualize the status of transaction pools instead of visualizing the details of transactions. It was a good decision because it does not obscure our focus on the blockchain visualization. The design style is inspired by material design [27], a light yet beautiful design which can satisfy the requirement of simple visualization design.

After that, we need to decide the base of visualization, i.e., the blockchain system. The selection of the multi-agent system rather than using blockchain platforms such as Ethereum is a right choice in the implementation. Because we need different mining strategies for each miner, it cannot be achieved in these blockchain platforms. As a result, the only way is to implement a simulation of a blockchain system by ourselves.

Finally, we decided that the factors of visualization are network delay and mining strategies. Network delay is simulated by the scheduling functionalities of the program, which is a reasonable way. Mining strategies do not exist in the real blockchain systems, so we decided the parameters by ourselves. We emphasized the computing power and the selection of transactions in the mining activities. The weakness of this model is that the attacks cannot be identified in the visualization. Designing a blockchain visualization which can prevent attacks will be the next step.

7.2 Future Work

In the future, we plan to do the following improvements to the visualization application.

- **Add different consensus protocols**
  The consensus protocol can be extended to include other algorithms, such as proof-of-stake. Therefore, researchers can switch between different consensus protocols to compare different mining behaviors.
- **Add malicious nodes**
  To simulate the attacks on the blockchain system, malicious nodes should be included in the application. Thus, we need to improve the algorithm of consensus protocols to validate and detect malicious nodes.
- **Dynamic nodes**
  In a peer-to-peer network, nodes can jump in and out at any time. Ensuring the dynamic creations and deletions while the visualization is running could be a good feature.
- **Merge of forks**
  To solve the merge problem in the second scenario, the naive solution is to provide a randomness to the parameters in the blockchain system. For example, the duration of mining time should be randomness between a range because solving a puzzle cannot be predicted in the real blockchain system.
- **Confirmation of blocks**
  The longest blockchain sometimes changes in the visualization. It can be improved to implement the confirmation of blocks. For example, a block is confirmed if it is 6 blocks deep and on the longest blockchain in Bitcoin.
- **Different sizes of blocks**
  In the visualization, the sizes of blocks are the same even if their values are different. Many visualization tools such as Bitcoincity and Bitbonkers distinguish different sizes of blocks. Therefore, it is a good feature to let the users get more information about the reward.

Bibliography

[1] S. Nakamoto. (2008) Bitcoin: A peer-to-peer electronic cash system. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[2] S. Nakamoto and M. Malmi. Bitcoin. [Online]. Available: https://bitcoin.org/en/

[3] E. Foundation. Ethereum project. [Online]. Available: https://www.ethereum.org/

[4] A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder, *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction.* Princeton University Press, 2016.

[5] T. A. Sundara, I. Gaputra, and S. Aulia, "Study on blockchain visualization," Sept 2017.

[6] H. Kuzuno and C. Karam, "Blockchain explorer: An analytical process and investigation environment for bitcoin," in *2017 APWG Symposium on Electronic Crime Research (eCrime)*, April 2017, pp. 9–16.

[7] L. Saublet, "Bitcoin visualization," Master's thesis, Université Paris-Sud, August 2015.

[8] M. Fleder, M. S. Kester, and S. Pillai, "Bitcoin transaction graph analysis," Feb 2015.

[9] A. Baumann, B. Fabian, and M. Lischke, "Exploring the bitcoin network," in *WEBIST 2014 - Proceedings of the 10th International Conference on Web Information Systems and Technologies*, vol. 1, 04 2014.

[10] D. McGinn, D. Birch, D. Akroyd, M. Molina-Solana, Y. Guo, and W. J. Knottenbelt, "Visualizing dynamic bitcoin transaction patterns," vol. 4, no. 2, 2016.

[11] G. D. Battista, V. D. Donato, M. Patrignani, M. Pizzonia, V. Roselli, and R. Tamassia, "Bitconeview: visualization of flows in the bitcoin transaction graph," in *2015 IEEE Symposium on Visualization for Cyber Security (VizSec)*, Oct 2015, pp. 1–8.

[12] A. Porat, A. Pratap, P. Shah, and V. Adkar, "Blockchain consensus: An analysis of proof-of-work and its applications," Stanford University, Tech. Rep.

[13] Bitbonkers. [Online]. Available: https://bitbonkers.com/

[14] Bitnodes. [Online]. Available: https://bitnodes.earn.com/

[15] Bitcoincity. [Online]. Available: http://bitcoincity.info/

[16] Blockseer. [Online]. Available: https://www.blockseer.com/

[17] Dailyblockchain. [Online]. Available: https://dailyblockchain.github.io/

[18] Elliptic. [Online]. Available: https://info.elliptic.co/hubfs/big-bang/bigbang-v1.html

[19] Interaqt. [Online]. Available: http://bitcoin.interaqt.nl/

[20] Live globe. [Online]. Available: https://blocks.wizb.it/

[21] Blockchain. [Online]. Available: https://blockchain.info/

[22] Etherscan. [Online]. Available: https://etherscan.io/

[23] F. FIT. Music box. [Online]. Available: https://www.fit.fraunhofer.de/de/fb/cscw/blockchain/radiobox-blockchain.html

[24] Websocket. [Online]. Available: https://tools.ietf.org/html/rfc6455

[25] Almende. Eve. [Online]. Available: http://eve.almende.com/

[26] Json-rpc. [Online]. Available: http://www.jsonrpc.org/specification

[27] Material design. [Online]. Available: https://material.io/

Appendices

Appendix A

Source Codes

The source codes are maintained under the following link:

`https://scm.fit.fraunhofer.de/scm/git/BlockchainVisualizer`

Appendix B

Configuration Files

B.1 A Sample of Configuration Files

```
{
    "nodes": [
        // transaction generator
        {
            "id": "6f71279b",
            "type": "generator"
        },
        // miner (Alice)
        {
            "id": "f1e50b4f",
            "type": "miner",
            "color": "#e53935",
            "name": "Alice",
            "miningTime": 1,
            "minValue": 1,
            "mineNumber": 1,
            "maxPending": 10
        },
        // nonminer (David)
        {
            "id": "480b945f",
            "type": "nonminer",
            "name": "David"
        },
    ],
    "delays": [
        // transaction generator
        {
            "id": "6f71279b",
            "neighbors": [
                {
                    "id": "f1e50b4f",
                    "seconds": 1
                }
            ]
        },
        // miner (Alice)
        {
```

```
                "id": "f1e50b4f",
                "neighbors": [
                    {
                    {
                        "id": "480b945f",
                        "seconds": 1
                    }
                ]
            },
            // nonminer (David)
            {
                "id": "480b945f",
                "neighbors": [
                    {
                        "id": "f1e50b4f",
                        "seconds": 1
                    }
                ]
            }
        ],
        "transactions": [
            {
                "reward": 2,
                "delay": 0
            }
        ]
    }
```

Listing B.1: Sample of Configuration File

B.2 Scenario I

```
    {
        "nodes": [
            // transaction generator
            {
                "id": "6f71279b",
                "type": "generator"
            },
            // miner (Alice)
            {
                "id": "f1e50b4f",
                "type": "miner",
                "color": "#e53935",
                "name": "Alice",
                "miningTime": 1,
                "minValue": 1,
                "mineNumber": 1,
                "maxPending": 10
            },
            // miner (Bob)
            {
                "id": "2d30b6fd",
                "type": "miner",
                "color": "#03a9f4",
                "name": "Bob",
                "miningTime": 5,
```

```
                "minValue": 4,
                "mineNumber": 3,
                "maxPending": 10
            },
            // miner (Charlie)
            {
                "id": "a6b877ab",
                "type": "miner",
                "color": "#4caf50",
                "name": "Charlie",
                "miningTime": 6,
                "minValue": 7,
                "mineNumber": 2,
                "maxPending": 10
            }
        ],
        "delays": [
            // transaction generator
            {
                "id": "6f71279b",
                "neighbors": [
                    {
                        "id": "f1e50b4f",
                        "seconds": 1
                    },
                    {
                        "id": "2d30b6fd",
                        "seconds": 1
                    },
                    {
                        "id": "a6b877ab",
                        "seconds": 1
                    }
                ]
            },
            // miner (Alice)
            {
                "id": "f1e50b4f",
                "neighbors": [
                    {
                        "id": "2d30b6fd",
                        "seconds": 1
                    },
                    {
                        "id": "a6b877ab",
                        "seconds": 1
                    }
                ]
            },
            // miner (Bob)
            {
                "id": "2d30b6fd",
                "neighbors": [
                    {
                        "id": "f1e50b4f",
                        "seconds": 1
                    },
                    {
```

```
                        "id": "a6b877ab",
                        "seconds": 1
                    }
                ]
            },
            // miner (Charlie)
            {
                "id": "a6b877ab",
                "neighbors": [
                    {
                        "id": "f1e50b4f",
                        "seconds": 1
                    },
                    {
                        "id": "2d30b6fd",
                        "seconds": 1
                    }
                ]
            }
        ],
        "transactions": [
            {
                "reward": 2,
                "delay": 0
            },
            {
                "reward": 3,
                "delay": 1
            },
            {
                "reward": 9,
                "delay": 2
            },
            {
                "reward": 4,
                "delay": 3
            },
            {
                "reward": 7,
                "delay": 4
            },
            {
                "reward": 5,
                "delay": 5
            },
            {
                "reward": 1,
                "delay": 6
            },
            {
                "reward": 5,
                "delay": 7
            },
            {
                "reward": 3,
                "delay": 8
            },
            {
```

```
142                 "reward": 2,
143                 "delay": 9
144             }
145         ]
146     }
```

Listing B.2: Scenario 1

B.3 Scenario II

```
1     {
2         "nodes": [
3             // transaction generator
4             {
5                 "id": "6f71279b",
6                 "type": "generator"
7             },
8             // miner (Alice)
9             {
10                "id": "f1e50b4f",
11                "type": "miner",
12                "color": "#e53935",
13                "name": "Alice",
14                "miningTime": 1,
15                "minValue": 1,
16                "mineNumber": 1,
17                "maxPending": 10
18            },
19            // miner (Bob)
20            {
21                "id": "2d30b6fd",
22                "type": "miner",
23                "color": "#03a9f4",
24                "name": "Bob",
25                "miningTime": 1,
26                "minValue": 1,
27                "mineNumber": 1,
28                "maxPending": 10
29            },
30            // miner (Charlie)
31            {
32                "id": "a6b877ab",
33                "type": "miner",
34                "color": "#4caf50",
35                "name": "Charlie",
36                "miningTime": 1,
37                "minValue": 1,
38                "mineNumber": 1,
39                "maxPending": 10
40            }
41        ],
42        "delays": [
43            // transaction generator
44            {
45                "id": "6f71279b",
46                "neighbors": [{
47                        "id": "f1e50b4f",
```

```
                        "seconds": 1
                    },
                    {
                        "id": "2d30b6fd",
                        "seconds": 1
                    },
                    {
                        "id": "a6b877ab",
                        "seconds": 1
                    }
                ]
            },
            // miner (Alice)
            {
                "id": "f1e50b4f",
                "neighbors": [{
                        "id": "2d30b6fd",
                        "seconds": 3
                    },
                    {
                        "id": "a6b877ab",
                        "seconds": 3
                    }
                ]
            },
            // miner (Bob)
            {
                "id": "2d30b6fd",
                "neighbors": [{
                        "id": "f1e50b4f",
                        "seconds": 3
                    },
                    {
                        "id": "a6b877ab",
                        "seconds": 3
                    }
                ]
            },
            // miner (Charlie)
            {
                "id": "a6b877ab",
                "neighbors": [{
                        "id": "f1e50b4f",
                        "seconds": 3
                    },
                    {
                        "id": "2d30b6fd",
                        "seconds": 3
                    }
                ]
            }
        ],
        "transactions": [
            {
                "reward": 2,
                "delay": 0
            },
            {
```

```
                "reward": 3,
                "delay": 1
            },
            {
                "reward": 9,
                "delay": 2
            },
            {
                "reward": 4,
                "delay": 3
            },
            {
                "reward": 7,
                "delay": 4
            },
            {
                "reward": 5,
                "delay": 5
            },
            {
                "reward": 1,
                "delay": 6
            },
            {
                "reward": 5,
                "delay": 7
            },
            {
                "reward": 3,
                "delay": 8
            },
            {
                "reward": 2,
                "delay": 9
            }
        ]
    }
```

Listing B.3: Scenario 2