Spring 5-17-2020

Implementing TontineCoin

Prashant Pardeshi
San Jose State University

Implementing TontineCoin

A Project

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Prashant Pardeshi

May 2020

The Designated Project Committee Approves the Project Titled

Implementing TontineCoin

by

Prashant Pardeshi

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSÉ STATE UNIVERSITY

May 2020

Dr. Thomas Austin	Department of Computer Science
Dr. Katerina Potika	Department of Computer Science
Dr. Robert Chun	Department of Computer Science

ABSTRACT

Implementing TontineCoin

by Prashant Pardeshi

One of the alternatives to proof-of-work (PoW) consensus protocols is proof-of-stake (PoS) protocols, which address its energy and cost related issues. But they suffer from the nothing-at-stake problem; validators (PoS miners) are bound to lose nothing if they support multiple blockchain forks. Tendermint, a PoS protocol, handles this problem by forcing validators to bond their stake and then seizing a cheater's stake when caught signing multiple competing blocks. The seized stake is then evenly distributed amongst the rest of validators. However, as the number of validators increases, the benefit in finding a cheater compared to the cost of monitoring validators reduces, weakening the system's defense against the problem. Previous work on TontineCoin addresses this problem by utilizing the concept of tontines. A tontine is an investment scheme in which each participant receives a portion of benefits based on their share. As the number of participants in a tontine decreases, individual benefit increases, which acts as a motivation for participants to eliminate each other. Utilizing this feature in TontineCoin ensures that validators (participants of a tontine) are highly motivated to monitor each other, thus strengthening the system against the nothing-at-stake problem. This project implements a prototype of Tendermint using the Spartan Gold codebase and develops TontineCoin based on it. This implementation is the first implementation of the protocol, and simulates and contrasts five different normal operations in both the Tendermint and TontineCoin models. It also simulates and discusses how a nothing-at-stake attack is handled in TontineCoin compared to Tendermint.

ACKNOWLEDGMENTS

I would like to convey my gratitude to my advisor Dr. Thomas Austin for his assistance and guidance during the course of this project. I would also like to express my appreciation towards the committee members Dr. Katerina Potika and Dr. Robert Chun, for their inputs and feedback on this project.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

Introduction

Bitcoin [1], often known as the first cryptocurrency, is a digital currency that is based on peer-to-peer technology to enable instant payments. It is not operated by any organization, any agency, or a bank. Invented in 2009 by an anonymous developer named Satoshi Nakamoto, it utilizes the blockchain technology to maintain transactions. *Blockchain* [2] is at the center of Bitcoin and handles the core mechanism for it. It is a ledger made of *blocks* that is maintained by each node in the blockchain network. A *block* in blockchain consists of valid transactions and other information and is linked to its previous block by storing a *cryptographic* hash of it. A node creates a block and broadcasts it in the network. If most of the nodes agree on that block, they append it at the end of their chain. This process of coming to an agreement is called *consensus*. Storing the cryptographic hash of the previous block provides resistance to modification of the data. It provides data integrity such that data in a block can only be altered when all the subsequent blocks are altered. Augmented with other technologies such as distributed consensus mechanism, digital signature, and cryptographic hash, blockchain powers various cryptocurrencies.

One of the most important distributed consensus mechanisms is *proof-of-work* (PoW) [3] [4]. It is the driving force behind Bitcoin. In this mechanism, a *prover* (a node which wants to prove something) can establish their claim to the *verifiers* (rest of the nodes) by demonstrating that they have spent a certain amount of computational effort. The *verifiers* can verify that the *prover* has spent the efforts by performing minimal computations. This process achieves the purpose of consensus as everyone agrees on something. There are different PoW schemes that use various algorithms such as SHA-256 [5], Scrypt [6], Blake-256 [7], etc. Many cryptocurrencies such as Bitcoin [1], Litecoin [8] [9] [10] [11], Ethereum [12] [13] [14], etc. are powered by PoW.

Although PoW is the most commonly used consensus mechanism in many cryptocurrencies, it suffers from a few disadvantages such as:

1. Energy wastage: Due to its heavy computational nature, PoW has led to enormous power consumption [15]. As per Bitcoin Energy Consumption Index [16], the Bitcoin network ranks 38 in terms of energy consumption compared with several countries.

2. Centralization: The increasing difficulty of the target value has led miners to form mining pools and replace their general-purpose processors with expensive special-purpose ASICs. Due to this, cryptocurrencies are not as decentralized as they should be.

3. Useless calculations: Calculations performed by the miners in PoW are complex, high energy-consuming, and of no use.

4. 51% attack: Although PoW makes it hard for miners to attack the system, it does not provide complete immunity. If miners come together to form 51% of the network's hashing power, they can control the network by preventing the blocks of the miners who have not joined them from getting accepted. They can produce a block based on an old block instead of a previously committed block. This will cause the blockchain to *fork*. A part of the network will follow the old blockchain containing the previous block while the rest of the network will follow the new blockchain containing the attacker's block. Since the majority of the network follows the new blockchain, the whole network will have to accept it. This way, the attacker can void the transactions that have already been validated in the previous block. If the attacker made some transfers in the previous block, they could again use that money since those transactions were

never committed. This attack is called a *double-spend* attack.

Proof-of-stake (PoS) [17] is another distributed consensus mechanism. It is a cost-effective and energy-efficient alternative to the PoW consensus model. Instead of using computational resources for mining, participants in this model use their wealth, i.e., their stake to gain a right to propose and verify blocks. A participant receives the ability to produce blocks equivalent to the amount of coins they staked or held. The more coins a participant owns, the more block producing power they have. However, the power is not solely based on the amount of coins. Instead, different cryptocurrencies use different methods based on the stake to select the next block producer. This prevents a participant(s) with the highest stake to control the network.

Peercoin [18] is the first cryptocurrency that utilized PoS protocol. Its consensus mechanism is based on the combination of both PoW and PoS models and uses the concept of *coin age*. A coin age is the product of a participant's coins and the duration for which they were held or not spent. For example, if a participant received 10 coins from another participant and held it for 50 days, that participant has accumulated 500 coin-days of coin age. Coin owner having high coin age has a higher probability of producing the next block. The process of generating a new valid block is called as *minting*. Peercoin introduced a new type of transaction named as *coinstake transaction* (based on the Bitcoin's *coinbase transaction)*, which is included by the block producer in their block. In this transaction, the block producer transfers themselves their coins, thus resetting or *consuming* the corresponding coin age. This prevents a participant with a high coin age to dominate minting. But, like Bitcoin, they also need to meet a *target* in order to mint a block. However, this target is not the same for all the participants. The more coin age a participant has, the easier it is to meet the target. A block producer generates hashes based on a stake modifier (defined by the network and recalculated after every 6 hours), their coins, and the current timestamp to compare it with the target. Since hashing is performed over a limited search space compared to Bitcoin's unlimited search space, a significant amount of energy is not consumed. Once a participant mints a block, they are eligible to produce the next block only after 30 days. A participant's probability to mint a block increases up till 90 days and remains the same after that. In case if there is a fork, the blockchain with the highest consumed coin age is selected by the network.

BlackCoin [19] [20] is the first cryptocurrency that utilized a pure PoS protocol. It is based on Peercoin's hybrid PoW/PoS protocol and addresses some of its issues. In Peercoin's model, a participant can keep on building their coin age, and once it is the highest, they can fork the blockchain and double-spend their coins. Also, the participants do not have to be online all the time. They can keep their node offline until they have accumulated enough coin age, after which they can mint blocks and again go offline. BlackCoin addressed these issues by removing the concept of coin age from their protocol. This protocol is purely based on the stake of a participant. The participants use their stake to verify new transactions and gain interest of 1% of the total staked coins yearly. The staked coins remain in their wallet. The participants have to *unlock* their wallet and be online to receive their stake rewards. To mint a block, a participant has to meet a target, which is also based on the coins they staked. By getting more participants to stake and be online, BlackCoin has reduced the probability of the 51% attack.

Although the above PoS based cryptocurrencies offered low operational cost and energy consumption, they suffered from *nothing-at-stake* [21] [22] problem. This problem is an assumption that if a network's blockchain forks, each participant in the network will support every fork. This is expected from the participants since they gain rewards for validating transactions. If they validate on all the chains, they increase their gains. Also, it does not cost anything to a participant to validate since, in a PoS network, participants have to stake to gain the validating right and not perform any expensive computations like in a PoW network. In contrast to this, if miners in a PoW network decide to mine on multiple chains simultaneously, they would have to split their hashing power and mine on each chain, which reduces their chances to mine a block.

Tendermint [23] is the first cryptocurrency that addressed nothing-at-stake problem by adapting a protocol proposed by Cynthia et al. [24], which allows a network to form a consensus in case of partial synchrony. Their protocol uses the PoS model and provides resilience up to one-third of dishonest participants in the network. In their protocol, a participant is required to *bond* their coins to gain the right to propose and verify blocks. A participant, also known as a *validator*, cannot use bonded coins until they are actively participating in the network. They receive their coins back after a certain number of blocks are created once they have *unbonded*. Tendermint handles the nothing-at-stake problem by penalizing a cheater by seizing their stake when caught signing multiple competing blocks. The seized stake is then evenly distributed amongst the rest of validators.

1.1 Problem

Although there are different variations of PoS available, Tendermint's consensus model is frequently used by other PoS systems. They handled the nothing-at-stake problem by punishing cheaters by seizing their stake and eliminating them from the network. A validator is cheating if they are validating blocks on multiple forks simultaneously. To catch a cheater, a genuine validator has to observe the network and report the cheating by submitting a type of transaction containing the cheater's signed competing blocks. Once the transaction is accepted by the network, the cheating validator's stake is seized and evenly distributed amongst the rest of validators.

While Tendermint's protocol protects the network from the nothing-at-stake problem, its defense weakens as the number of validators increases. As more validators join the network, the number of shares in a seized stake increases. This does not provide enough motivation to a genuine validator to monitor the network since the benefit of finding a cheater compared to the cost of monitoring reduced. This, in turn, also puts a limitation on the number of validators the system can support.

1.2 Proposed Solution

Previous work on TontineCoin [25] addresses this problem by utilizing the concept of a tontine. A tontine is an investment scheme where each participant invests into a common pool of money to raise a capital and receives a portion of benefits based on their share. A feature of tontines is that as the number of participants decreases, individual benefit increases, which acts as a motivation for participants to eliminate each other. Utilizing this feature in TontineCoin ensures that validators (participants of a tontine) are highly motivated to monitor each other, thus strengthening the system against the nothing-at-stake problem.

This project implements a prototype of Tendermint using the Spartan Gold codebase [26] and develops TontineCoin based on it. Spartan Gold is a simplified blockchain-based cryptocurrency written in JavaScript. This implementation is the first implementation of the protocol, and simulates and contrasts five different normal operations in both the Tendermint and TontineCoin models. These operations are initialization, the consensus process, transfer of coins among clients, bonding of clients, and unbonding of validators. It also simulates and discusses how a nothing-at-stake attack is handled in TontineCoin compared to Tendermint.

The remaining of the paper is organized as follows. Chapter 2 provides a background on PoW cryptocurrencies, Tendermint protocol, Tontine, TontineCoin model, and Spartan Gold code base. Chapters 3 and 4 describes the development of the Tendermint and TontineCoin prototypes, respectively. Chapter 5 shows and contrasts the normal operations in both the models. Chapter 6 shows and compares handling of a nothing-at-stake attack in both the models. Finally, chapter 6 presents the conclusion and future enhancements.

CHAPTER 2

Background

2.1 PoW cryptocurrencies

Bitcoin [1] is based on *Hashcash PoW* [27] scheme, which uses SHA-256 hash function. The participants in the Bitcoin's PoW network are called miners. They compete with each other *to produce a block* to receive a *block reward.* A block consists of valid transactions, the previous block's hash, a block version number, a timestamp, a target value, and a *nonce* value. The miner whose block is accepted by most of the network gets rewarded. To get their block to be accepted by other miners (verifiers), a miner (prover) must solve a problem that requires a lot of computational power. The problem is to produce a block whose cryptographic hash made by SHA-256 hash function is less than the *target* value. The target value is an extremely large 256-bit number that is common to all the miners. The difficulty for a miner to find a valid block depends on how the target value is set. A lower target value is more difficult to satisfy than a higher one. This problem is often defined as to find a hash that starts with a certain number of zero bits. Miners try by setting an integer value in the *nonce* field and incrementing it until they find a valid block. Once found, a miner announces the block to other miners. Upon receiving, miners check the block's validity. If it contains invalid transactions or its SHA-256 hash is not less than the target, it is discarded, and the receiver continues to find a valid block. Else, the receiver adds the block in their blockchain and begins to find the next block. This process of finding a valid block is called *mining.* The target value is recalculated after every 2,016 generated blocks so that every block takes ten minutes to mine on an average.

Litecoin [8] [9] [10] [11], a fork of Bitcoin, uses Scrypt algorithm in its PoW scheme. Similar to SHA-256, Scrypt is computationally intensive as it is required to produce large vectors of *pseudorandom* bits. But unlike SHA-256, Scrypt is also memory intensive. Miners need to store these vectors in Random Access Memory (RAM) so that they can be accessed whenever required. Since this type of mining requires more amount of memory than the processing power, it is called as memory-hard or memory-bound.

One of the purposes of Litecoin was to reduce the time required to produce a block which they achieved by lowering the difficulty of the target value. Litecoin's block takes 2.5 minutes on average to produce. This allows it to confirm transactions 4 times faster than Bitcoin. Similar to Bitcoin, the target value in Litecoin is recalculated after every 2,016 generated blocks.

Another purpose of Litecoin was to prevent the use of special-purpose hardware, known as *Application-Specific Integrated Circuits (ASICs)*, from mining. ASICs are the integrated circuits that perform a specific task rather than the general tasks. When used for mining, ASICs produce more hashes per second than CPUs or GPUs. The increasing target difficulty caused by the increasing number of miners in Bitcoin forced them to switch from CPUs to GPUs and then to ASICs for mining [28]. These caused the miners with general-purpose hardware to either join the miners having ASICs or purchase expensive ASIC themselves for mining. Due to Scrypt and its memory-intensive nature, Litecoin was initially successful in preventing ASICs mining. But soon later, ASICs were developed, which were Scrypt-capable. Even though Litecoin could not totally prevent ASICs mining, it hindered it as Scrypt-capable ASICs are more expensive and complicated to produce than SHA256-capable ASICs.

Ethereum [12] [13] [14], a cryptocurrency and a decentralized application platform, uses an algorithm named as *Ethash* in its PoW scheme. Similar to Bitcoin, miners in Ethereum need to check if a cryptographic hash is less than the *target* value. But instead of a block's hash, Ethereum's miners need to fetch random data from a dataset known as *Directed Acyclic Graph (DAG)*, hash it, and then compare it with the target. DAG is a large dataset that is generated from a common *pseudorandom cache* shared by all the miners. A miner requires to store the entire DAG in their memory so that they can fetch the data, compute the hash, compare it with the target, and repeat, in case the hash doesn't satisfy the target. Similar to Scrypt, Ethash is a memory-hard algorithm and offers resistance to ASICs. Ethereum's block takes 12 seconds on average to produce. Similar to Bitcoin, the target value in Ethereum is recalculated after every 2,016 generated blocks.

2.2 Tendermint

Tendermint [23] is a proof-of-stake consensus protocol in which the next block producer is selected based on the amount of coins they hold and a weighted round-robin algorithm [29]. Its design is similar to the design of the other blockchains. It comprises a peer-to-peer network made of nodes that communicate with each other to relay new information. Each node stores a copy of an ordered sequence of blocks known as a blockchain. Clients in the network have accounts which are identified by their public key or address. An account can hold some amount of coins. Transactions between clients can cause their amount of coins to change. A client can submit a transaction to the network to transfer coins from their account to others. By probing the transactions of an account from the blockchain, the amount of coins held by it can be determined.

Validator in a Tendermint network is analogous to a miner in a Bitcoin network with a difference that they follow PoS protocol instead of PoW. They are the clients that are allowed to produce or validate blocks in exchange for having their coins locked or *bonded*. Their right is equal to the amount of the coins they bond.

Tendermint protocol allows four types of transactions –

1. *Standard*: This transaction type lets a client transfer coins to other clients.

2. *Bond*: This transaction type lets a client have its coins locked in exchange for a right to become a validator. Their right is equal to the amount of the coins they bond.

3. *Unbonding*: This transaction type lets a client regain its bonded coins releasing its right to be a validator. The validator is free to use the bonded coins after a predetermined amount of time has passed post the submission of the unbonding transaction. This helps the network to identify a cheating validator before they regain their coins.

4. *Evidence*: This transaction type lets a client publish evidence against a cheating validator. If a validator validates or signs on two blocks at the same height, another validator can submit an evidence transaction, including two conflicting signatures. After the evidence transaction is accepted and committed in the blockchain, the bonded coins of the cheating validator are destroyed, and the validator is expelled from the validators set.

A block in the blockchain of Tendermint comprises of the following information:

1. Header: It contains the network name, the height of the blockchain, timestamp, previous block's hash, etc. A block's hash is derived from the hashes of the header, validation, and transactions of the block.

2. Validation for Block H-1 – These are the signatures of validators for the previous block.

3. Transactions: These are the current transactions to be committed in the blockchain.

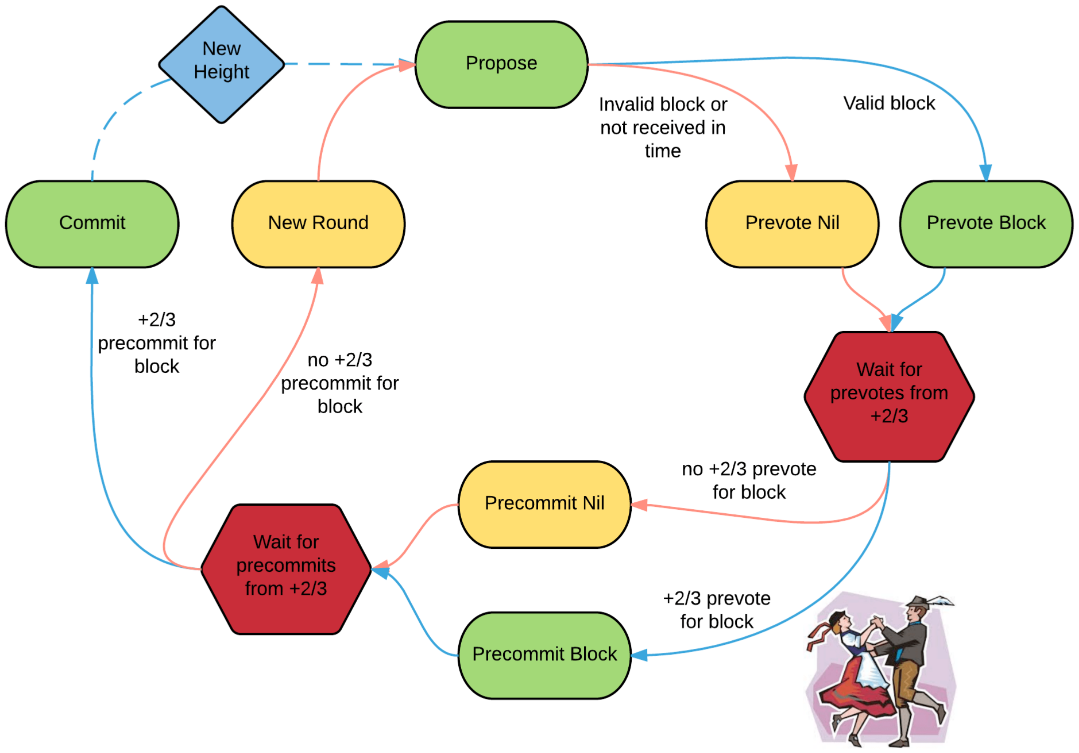

New Height
Propose
Invalid block or not received in time
Valid block
Commit
New Round
Prevote Nil
Prevote Block
+2/3 precommit for block
no +2/3 precommit for block
Wait for prevotes from +2/3
no +2/3 prevote for block
Precommit Nil
Wait for precommits from +2/3
+2/3 prevote for block
Precommit Block

Figure 1: An overview of the consensus process in Tendermint, adopted from [30]

Validators in Tendermint participate in a round-based consensus process, as shown in Figure 1, to produce the next block. This process involves signing votes for a block, and relaying them to others. In each round, a validator should broadcast three types of votes: a *prevote*, a *precommit*, and a *commit*. A vote comprises the height of the block, the round number, the type of vote, the block hash, and the signature. Each round is made up of three steps: *Propose*, *Prevote*, and *Precommit*; with two additional steps *Commit* and *NewHeight*.

The block proposer for a round is selected based on a weighted round-robin algorithm. Each validator maintains a priority queue of the *accumulated powers* of all validators. At the beginning of a round, each validator increases them by their associated validator's stake and selects the first validator from the queue to be the current proposer. Then, the proposer's accumulated power is decremented by the total stakes of all validators.

Once a proposer is selected, the consensus process follows the steps mentioned earlier and proceeds in the following manner:

1. Propose: At the beginning of this step, the designated proposer broadcasts their proposal (block) to their neighboring validators by signing it with their signature. Upon receiving, a validator then broadcasts it to their neighbors.

2. Prevote: At the beginning of this step, a validator sends out a signed prevote either for a block that they were locked onto from the previous round or the block from the current round. In case, if they have not received a block or the block they have is invalid, they send out a signed nil prevote.

3. Precommit: If a validator has received more than 2/3 of prevotes for a block, they lock onto that block and broadcast a signed precommit for that block. Else, if they have received more than 2/3 of nil prevotes for a block, they unlock onto that block. In case, if they have not received 2/3 of either of the votes, they do not sign or lock on a block.

   At the end of this step, if a validator has received more than 2/3 of precommits for a block, they move onto the commit step. Else, they continue to find the next proposer for the next round.

4. Precommit: At this step, a validator waits for two conditions to be satisfied to move on to the next round. They wait for the precommitted block to arrive if they have not still received it. Once the block is received, they send out a signed commit to other validators. They also wait till they receive 2/3 commits from other validators. Once both the conditions are satisfied, they move onto

the NewHeight step.

5. NewHeight: A validator stays at this step for some predetermined interval to gather remaining commits for the previous block at height H-1, given the validator is at height H. After this, each validator then determines the next block proposer.

While Tendermint's PoS protocol is better than Bitcoin's PoW protocol in ways such as reduced power consumption, better energy efficiency, and improved block production rate, it suffers from other issues.

One of the issues is an attack of denial-of-service (DoS) against proposers. Tendermint expects validators to setup their Sentry Node system to prevent this attack [31] [32]. This option is not embedded in Tendermint's system. If a validator opts-in, they have to take certain precautionary measures to maintain a node that is fully fault-tolerant. Or else, they are eliminated from the network after a duration of being offline.

Another issue is as the validators pool grows, the incentive to catch a cheater compared to the cost of monitoring validators reduces, giving a de-facto limitation on the number of validators the system can support.

2.3 Tontine

In his work on the history of tontines, McKeever [33] describes a *tontine* as an investment scheme in which each shareholder deposits their amount, and in return, they receive some benefit or profit while they are still alive. After a shareholder's death, their share is not transferred to their heir. Instead, it gets distributed among other shareholders after a small number of them are the only ones left alive.

The tontine scheme is supposedly invented by an Italian banker named Lorenzo de Tonti. In early 1650, he proposed the idea of a tontine to the king of France, Louis XIV to raise revenue for the state. The idea was that each participant would subscribe to the scheme by buying shares at 300 livres per share. The subscriber has to nominate a third party who will then receive an annual payment based on their age group and the interest gained on the initial raised capital. The share of a nominee would increase when another nominee in the same age group dies.

This idea of de Tonti never got off the ground. Later in 1670, the very first tontine was launched and operated by the city of Kampen, Holland.

France created its first national tontine in 1689, five years after the death of de Tonti. At the end of the first French tontine, a widow named Charlotte Barbier was the only subscriber remaining. She had received back 73,500 livres in return to her original investment of 300 livres only [34].

In the late seventeenth century, the Treasury Secretary of the U.S., Alexander Hamilton, put forward the idea of utilizing the tontine scheme to decrease the nation's debt [35]. Based on the tontine's version proposed by the British Prime Minister William in 1789, Hamilton's tontine had a different payout structure than the usual one. In Hamilton's tontine scheme, when the members pool reduced to 20 percent of the initial pool, the investor payments froze to the last beneficiaries. The surviving members would still receive their dividends, but it would not increase even if the other members die. Hamilton's tontine scheme was rejected by Congress.

Tontines have been used as a medium to raise the capital for projects such as the Richmond Bridge in London, the Tontine Hotel in Shropshire, and the Freemasons Hall in London. The first house of the New York Stock Exchange, the Tontine Coffee house, was built out of a tontine scheme and was the primary meeting place for traders to buy and sell stocks. Tontines are also associated with life insurance policies. Post mid-eighteenth century, the owner of Equitable Life Assurance Society, Henry Baldwin Hyde, used tontines as a medium to sell life insurance policies [36]. At its peak, Hyde's company had sold 2/3 of the United States' outstanding insurance policies. But these policies required policyholders to maintain monthly payments, which led them to lose their accumulated funds when a single installment was missed. Tontines were also used as the Christmas saving schemes by the communities of Rhostyllen, a village in the country of Wales [37].

Tontine offers profits to the individuals, which are the last surviving members in the scheme. This implies that to gain benefits, a member can either outlive other members or eliminate them by some means. Due to this nature, tontines are deemed as bad investment schemes. Some US states such as New York and Wisconsin passed laws that prohibited insurances that deferred dividends to more than five years [38].

However, this nature of tontine proves to be an advantage to solve Tendermint's "no monitoring" issue discussed in section 2.2. Using a tontine scheme in Tendermint, a validator remains motivated to catch a cheater as the gains are high compared to the cost of monitoring validators. This is because as the cheating validator gets eliminated, the share of the catching validator increases. Here elimination does not refer to the cheating member's death. It denotes their removal from the validators pool.

2.4 TontineCoin

TontineCoin [25] is a new PoS protocol based on the concept of a tontine. It extends Tendermint's PoS protocol to enforce a monitoring system to find out cheaters by offering increased gains on catching them. The payout for catching a cheater in the Tendermint system decreases as the validators pool expands. The cost of monitoring the system compared to the returns in-turn is high. TontineCoin's protocol keeps a validator strongly motivated to check for a cheating validator since it is in their benefit to do so. It is described as murder-based due to its similarity with the nature of tontine.

2.4.1 Tontine Formation

A tontine in TontineCoin is a group of clients interested in validating. Validators set in TontineCoin comprises of the tontines which are selected to join based on following two different approaches:

1. *Hybrid model*: In this model, clients group together to form *tontines*. However, tontine formation is not recorded on the main blockchain. Each tontine competes to find a PoW, which is better than the current one in the network and submits a bid whenever they find it. The tontine with the best proof at the end of the bidding process joins the validators pool.
2. *Pure PoS model*: In this model, instead of forming groups, clients individually submit a bond transaction in the network to become a validator. After receiving a fixed number of requests, a new tontine is created with those requesting clients. The new tontine joins the validators pool and stays active till the end of the tontine.

For any of the above models, the number of active tontines in the network is limited by a predefined number given by m. Each tontine is restricted to operate for a fixed duration, after which it is removed from the validators set. Also, once a tontine gets in the validators pool, its members function independently from their co-members.

2.4.2 Hybrid – The train model

Hybrid model is the combination of both PoW and PoS protocols. Clients band together to create tontines off-chain and compete with each other to find a proof that beats the current proof in the network. Once found, they submit a *bid* transaction. At the end of the bidding process, whichever tontine has the best bid joins the validators set. Members of the tontine then function independent of each other.

This model is described as the train model since after every predefined number of blocks given by N, the bidding process completes, and the selected tontine is added to the validators pool. It is similar to a scheduled train.

2.4.2.1 Hybrid Tontine Formation

Clients interested in validating must stake some coins and group together off-chain to form a tontine. The amount of staked coins in a tontine must be equal to a predefined amount given by S. The following details are included in a bid transaction by a tontine:

1. The block's hash that contains the details of the last selected tontine.
2. The stake of each member collectively making up to S coins.
3. The share of each member. A member's client id and their percentage of rewards together make their share in a tontine.
4. Nonce.

A member's percentage of rewards in a tontine depends on two factors, the amount of coins they staked and the amount of computing power they offered. TontineCoin's model does not specify how they affect in deriving the percentage. It is likely that a member having more staked coins and computing power than another member in the same tontine will have a higher share in returns.

2.4.2.2 Hybrid Tontine Selection

Once formed, a tontine competes with other tontines in the bidding process to find a PoW that beats the current proof on the blockchain. This current proof is referred to as the *current best bid*, and the tontine that submitted this bid is known as the *current bid leader*. A tontine submits a bid transaction to the network with all the details discussed above once it finds a better proof than the current best bid. A transaction fee is levied to restrict the tontines from submitting incorrect bid transactions. The selection process follows the steps mentioned below:

1. A predetermined member potentially a leader from the tontine submits a bid transaction in the network once they have a better PoW than the current bid leader's PoW.

2. After validating, the transaction is added in the current block by its producer.

3. If the submitted bid is better than the current best bid, the coins of current bid leader are unbonded, and the owner of the better bid becomes the new current bid leader.

4. Tontines continue to compete to find a PoW that beats the current bid leader's PoW until the selection block arrives. The selection block is the block at which the bidding process ends, and the tontine of the current bid leader is added to the validators pool.

5. The selected tontine begins functioning after a delay of a predefined number of blocks, and its coins remain bonded until it ceases to operate.

6. If the number of active tontines in the network is greater than m, the oldest tontine is removed from the validators set, and its coins are unbonded.

2.5 Spartan Gold

Spartan Gold [26] is a basic cryptocurrency built on a simplified version of a blockchain. It is created in JavaScript and runs on a Node.js platform. Internally, it uses a PoW consensus model to produce a block and is based on *UTXO (Unspent Transaction Output)* model instead of *Account/Balance* model to keep track of coins.

It simulates a cryptocurrency network in which transactions happen between clients and are validated and put into blocks by miners after mining. When the simulation starts, the working of the network is displayed on the command line in the form of logs. Spartan Gold comes with two pre-defined simulations, one which has a single miner and the other that has two miners.

Driver module is the entry point of the Spartan Gold system. It creates clients, miners and a *genesis* block and starts the simulation. The *genesis* block is the first block of miners' blockchain, which contains details of clients and miners and their initial funds. UTXOs and the balances of both the entities are displayed some time after the simulation starts to display the state of the system.

Transaction module allows a client to create transactions to transfer coins to other clients. A transaction is made up of *inputs*, *outputs* and an *id*. Each *input* in *inputs* contains a transaction ID of a transaction from which the coins should be picked, the index of an *output* within that transaction, the public key that matches the hash of the public key (address) of that output, and the signature that matches the signature on the output when checked with the public key. It is in the form *{txID, outputIndex, pubKey, sig}*. Each *output* in *outputs* contains a receiver's address and an amount to be transferred to them. It is in the form *{amount, address}*.

Wallet module allows a client to create a wallet in which they can store their coins and public/private keys associated with them. A coin contains a UTXO, its associated transaction ID, and the output index of that UTXO in that transaction. It is in the form *{output, txID, outputIndex}*.

Block module allows a miner to produce a block. A block is made up of transactions, the previous block's hash, a PoW target, the block's height, the time of it's creation, the UTXOs, and the *coinbase* transaction.

Clients are created using *Client* module. A client has a *wallet* to hold their *coins* and a *broadcast* method to broadcast messages to other clients. They can create and broadcast a transaction to transfer coins to other clients. They can receive the coins if they have the public key for the address.

Miners are clients, but they also produce blocks in the network. *Miner* module extends *Client* class and allows to create a miner. In addition to *coins* and a *broadcast* method, a miner has a *name* and maintains a blockchain. When the simulation begins, all miners start *mining*. Each miner creates a block based on the genesis block and begins searching for an integer i.e. a *proof* which when added to the block gives a hash value that is less than the block's *target*. If a miner has found a proof, they broadcast the block with the proof for other miners to validate and store in their blockchain and then proceed with *mining* the next block. If they haven't, after a predefined interval of time, they pause to listen to other miners for their proofs. After announcing, a miner adds a predefined number of coins as rewards to their wallet and proceeds with *mining* the next block. Miners also add transactions to their blocks as they are received and validated.

CHAPTER 3

Tendermint Prototype Development

This project implements a Tendermint's prototype based on the Spartan Gold code base in order to contrast its design with TontineCoin. The same components are utilized but are modified to incorporate Tendermint's protocol. Major changes are made in *Miner* module to replace Spartan Gold's PoW protocol with Tendermint's PoS protocol. *Miner* is renamed to *Validator* to keep the name consistent with the concept. Also, *Transaction* module is modified to support three more types of transactions - *bonding*, *unbonding*, and *evidence* - along with the existing type, *standard*, used to transfer coins. A new module named *Vote* is added to allow a validator to create votes for *prevote* and *commit*. This prototype of Tendermint follows *UTXO* model instead of *Account/Balance* model. Also, it does not have *precommit* stage as it is in the Tendermint's protocol.

3.1 *Driver*

Similar to its predecessor, *Driver* module acts as the entry point of the Tendermint system. It creates clients, validators and a genesis block and starts the simulation of Tendermint's protocol. In addition to a *REG* (*standard*) transaction that gives clients and validators their initial funds, the genesis block contains *BND* (*bonding*) transactions to bond the validators' stakes. This module also simulates a transfer of coins from a client to another using a *REG* transaction (described later), unbonding of a validator from the network by posting a *UND* transaction (described later), bonding of a client by posting a *BND* transaction (described later) with a stake, cheating of a validator by proposing a conflicting block, and seizing their stake and ejecting them by posting a *EVD* transaction by other validators. Also, after every predefined period of time, UTXOs and the balances of all the clients are displayed to show the current state of the system.

3.2 *Transaction*

Like its predecessor, *Transaction* module allows a client to create transactions. But along with the transfers, i.e., the *REG* transactions, a validator can create *BND*, *UND*, and *EVD* transactions. To incorporate this change, the transaction structure is modified to have two more fields described below:

1. *type*: A transaction's type which can be any of the following:
   (a) *REG*: The type for a transaction to transfer coins.
   (b) *BND*: The type for a transaction to stake coins and join the validators' network.
   (c) *UND*: The type for a transaction to receive staked coins and eject from the validators' network.
   (d) *EVD*: The type for a transaction to report a dishonest validator with evidence.
2. *data*: It contains the additional information required in a transaction. The following are the details of *data* for each of the transaction type:
   (a) In case of a *REG* transaction, it should contain receivers' names. This allows validators to maintain a *legder* of other validators' names and their coins.
   (b) In case of a *BND* transaction, it should contain bonding validator's name and public key. This allows validators to maintain a store of public keys of other validators.
   (c) In case of a regular *UND* transaction, it should be empty. In case of a seizing *UND* transaction, in which a validator posting it receives their

portion of the cheating validator's stake, *data* it should contain the *id* of the *EVD* transaction.

(d) In case of a *EVD* transaction, it should contain the cheating validator's name, their *BND* transaction and the conflicting blocks. *inputs* and *outputs* should be empty.

The structure of an *output* in *outputs* is also changed to consist *type* so that different types of UTXOs, i.e., *REG*, *BND*, and *UND* can be identified. Each output is of the form *{amount, address, type}*.

The validation for *REG*, *BND*, and regular *UND* transactions remain the same as it was in Spartan Gold. Additional validation is added for seizing *UND* transaction. In that case, the *id* of the *EVD* transaction is retrieved from *data* and *utxos* is checked if it contains it or not. In case it doesn't, the transaction is discarded. Also, the hash of the public key (address) and the signature from the *input* is not matched with the matching UTXO's address and signature. Both of these validations are skipped to allow the validator posting the seizing *UND* transaction to receive their share of coins. In case of a *EVD* transaction, currently, no validation is added. It can be a part of the revised implementation.

3.3 *Block*

Similar to its predecessor, *Block* module allows a validator to produce a block. Its structure is similar to that of Spartan Gold's except it does not contain a target and coinbase transaction and have two additional fields - a set of previous block *commit* votes and the block creator's signature.

3.4 *Validator*

Previously named as *Miner*, this module allows to create validators in the system. Similar to its predecessor, it extends *Client* class. A validator is initialized with a name, a Boolean value denoting whether they should cheat or not and another Boolean value denoting whether they should detect cheating or not. They also have methods to broadcast messages to other validators, add clients to the validators' network, remove validators from the validators' network and send messages to a specific validator.

A validator maintains other validators' stakes, accumulated powers, and the total amount of staked coins, to be utilized in the consensus process. Additionally, they maintain:

1. A legder of other validators' coins to check if a validator posting a *BND* transaction has the specified amount of coins.

2. Other validators' public keys to verify a block proposer's signature.

3. Other validators' *BND* transactions utilized to retrieve a validator's *BND* transaction in case they post a *UND* transaction.

4. A proposer and their proposal for the current round.

5. *prevotes* and *commits* for the current proposal received from the other validators.

6. *commits* for the previous proposal.

7. The transactions received from clients to be added to the blockchain.

A validator walks through the following stages in a round of consensus:

1. *findProposer*: They *unbond* validators, if there are any, and send the current state of the system to validators who have bonded to the validators' network at the end of the previous round. Then they proceed to find the current round's *proposer*, a validator whose accumulated power is maximum, and broadcast their name.

2. *receiveProposer*: Upon receiving a proposer's name, they check if all the received proposers' names are the same once they have received them from more than 2/3 of the total validators. If they are not the same, a message is displayed to denote this, and then the receiving validator moves on to the next round. If they are the same, the receiving validator sets its *proposer* field with the elected proposer's name and decrements its accumulated power by the total stakes.

3. *propose*: If the receiving validator themselves is the elected proposer, they create a proposal, i.e., a block and broadcast it. The transactions received from clients are added in the new block before broadcasting.

4. *receiveProposal*: Upon receiving a proposal, the validator validates it. If it is valid, it is set as *proposal*, and the validator moves on to prevote stage. If it is invalid, it is handled according to its invalidity (described later), and then the validator moves on to the next round.

5. *prevote*: The validator creates a signed *prevote* and broadcasts it. Upon receiving a prevote, the validator checks if it is signed by its owner. If it is not, an error message is displayed to show this. Else, the validator moves on to commit stage once they have received prevotes from more than 2/3 of the total validators.

6. *commit*: The validator creates a signed *commit* and broadcasts it. Upon receiving a commit, the validator checks if it is signed by its owner. If it is not, an error message is displayed to show this. Else, the validator moves on to commit the block and finalize the round once they have received commits from more than 2/3 of the total validators.

7. *finalize*: If the validator is set to cheat, they do not add the block in their blockchain after a predefined block height is reached. This causes them to

cheat by proposing a block based on an old block. Later, if they receive a block whose height is greater than the predefined block height, they broadcast a sync request to get synced with the current state of the system. Those transactions which are committed in the block are removed from transactions set maintained by the validator. A seizing *UND* transaction is created to receive part of the stakes of validators against whom EVD transactions are committed. Those validators are unbonded at the beginning of the next round. Validators whose BND transactions are committed are added to the validators' network. If the validator is proposer, they broadcast committed transactions to the clients and validators so that they can add their UTXOs to their wallets. Proposer also sends the current state of the system to validators who have requested for a sync. In the end, they broadcast a message to start the next round of consensus.

A validator bonds to the validators' network after their *BND* transaction is committed in the blockchain. *outputs* of this transaction contains an output specifying the validator's amount of stake, their self-address and type as *BND*. If there is a change amount after *inputs* are created, another output is added in *outputs* with that amount, the validator's self-address, and type as *REG*.

A validator unbonds from the validators' network after their *UND* transaction is committed in the blockchain. *inputs* of this transaction contains details to locate *outputs* of the validator's *BND* transaction and *outputs* contains details to receive them at their specified address.

A validator's stakes are seized after a *EVD* transaction against them is committed in the blockchain. Each validator (except the cheater) creates a seizing *UND* transaction having *inputs* containing details to locate *outputs* of the cheating validator's *BND* transaction. *outputs* contains details to receive them at the address specified by the validator. The *amount* in *outputs* is the equally divided share of that validator in cheating validator's stake.

A validator checks a block's validity when they receive it from its proposer. A block is invalid if its height is less than or equal to the validator's current block's height. The received block is further validated based on the following conditions:

1. If the block's height and signature are equal to the validator's current block's height and signature, it is a conflicting block in which case a *EVD* transaction is created using both the blocks and stored in transaction set for that validator to add in their next proposal.

2. If the block's previous block's hash does not match with the validator's current block's hash, it is invalid.

3. If the block's signature does not match with its proposer's signature, it is invalid.

If none of the above conditions are true, the block is valid.

3.5 *Vote*

This module allows a validator to create a vote. There are two types of vote – *prevote* and *commit*. A vote has a block's hash, a type, and a signature of its creator.

CHAPTER 4
TontineCoin Prototype Implementation

This chapter discusses the implementation details of the TontineCoin prototype. It is built based on the developed Tendermint prototype and utilizes the same components with the addition of a new component named *Tontine* that allows clients to create tontines and bid to become validators. *Validator* module is modified to provide support for *bid* transactions and to add the selected tontine in the validators' network when the selection block is reached. In this prototype, the last tontine to bid is selected instead of the tontine having the best PoW. Each validator is associated with a tontine ID and a share in its tontine. Each validator monitors its own tontine members for misbehavior. In case of a conflicting proposal, the cheater's tontine members create *EVD* transactions and broadcast it to the network, unlike Tendermint's validators, which keep *EVD* transactions to themselves to add in their next proposal. This reduces the time taken by the network to seize cheater's stake and eject them. After these transactions are committed, the cheater's tontine members create seizing *UND* transactions to receive the coins according to their shares in their tontine.

4.1 *Driver*

Similar to Tendermint, this module acts as the entry point of the TontineCoin system and simulates the same operations as Tendermint's. It first creates initial clients and validators with their respective attributes. A genesis tontine is created based on the initial validators, their shares, and stakes. A genesis block is created that contains a *REG* transaction, and the *BND* transactions except the *BND* transactions are retrieved from the genesis tontine that is included in *data* of the *BID* transaction (described later) posted by the leader of genesis tontine. It is also added to the genesis block.

To simulate bonding of clients, two more tontines are created based on new validators, their shares, and stakes. The leaders of each of these tontines submit their bid to join the validators' network. The last tontine to submit a bid is selected to be added to the validator's network when the selection block is reached. This prototype also simulates a cheating scenario similar to that of Tendermint.

4.2 *Transaction*

A new type of transaction is added in this module to allow new validators to create bid transactions. The type of this transaction is *BID*, and its *data* should include a tontine created using *Tontine* module. This transaction has empty *inputs* and *outputs.* Also, in case of a *BND* transaction, *data* should have a validator's *share* and their *tontine ID* along with their name and public key.

4.3 *Validator*

In addition to its previous fields, a validator also maintains three new fields – *share* to store their share in their tontine, *tontineID* to store the tontine's ID to which they belong, and *currentBidLeader* to store the current bid leader tontine. Along with a stake, a validator also maintains other validator's *share* and *tontineID.* Instead of the total amount of stakes, a validator maintains the sum of the shares of all validators.

A validator in TontineCoin walks through the same stages of consensus as they are in Tendermint with the following changes:

1. *findProposer*: Instead of a validator's stake, their share is added to their accumulated power, and the validator having maximum accumulated power is announced as the current round's proposer.

2. *receiveProposer*: Instead of total stakes, the elected proposer's accumulated power is decremented by total shares.

3. *finalize*: In addition to the operations performed in its predecessor, the validator updates the *currentBidLeader* with the tontine from the last *BID* transaction. When a predefined block height (selection block) is reached, the *BND* transactions from *currentBidLeader* are added to the transactions set.

Similar to Tendermint, in TontineCoin, a validator bonds, unbonds, and ejects from the validators' network after their associated transactions are committed in the blockchain. But in TontineCoin, after a validator is unbonded, other validators update *shares* of those validators whose *tontineID* match with the unbonding validator's *tontineID* as per the following formula:

$$n = o + ((o/(1 - c)) * c)$$

where n is the validator's new share, o is their old share, and c is the unbonding validator's share.

Also, instead of all validators, only those validators whose *tontineID* match with proposer's *tontineID*, check if the received block is a conflicting one or not. If yes, those validators create *EVD* transactions and broadcast it to the network. After these transactions are committed, they create and broadcast seizing *UND* transaction. A validator's share in the cheating validator's stake is given by the formula discussed above.

4.4 *Tontine*

This module allows a set of new validators to create a tontine. A tontine is made up of the following fields:

1. *nonce*: An integer value.

2. *members*: A validator's name with their share in the tontine, stake, and BND transaction.

3. *id*: A tontine ID.

It is assumed that the validators' total stake is equal to a predefined amount given by *MAX_ AMNT*, and the total share is *1*. Each validator's *share* is set to their share, and *tontineID* is set to newly created tontine's ID.

CHAPTER 5

Contrasting Normal Operations of TontineCoin and Tendermint

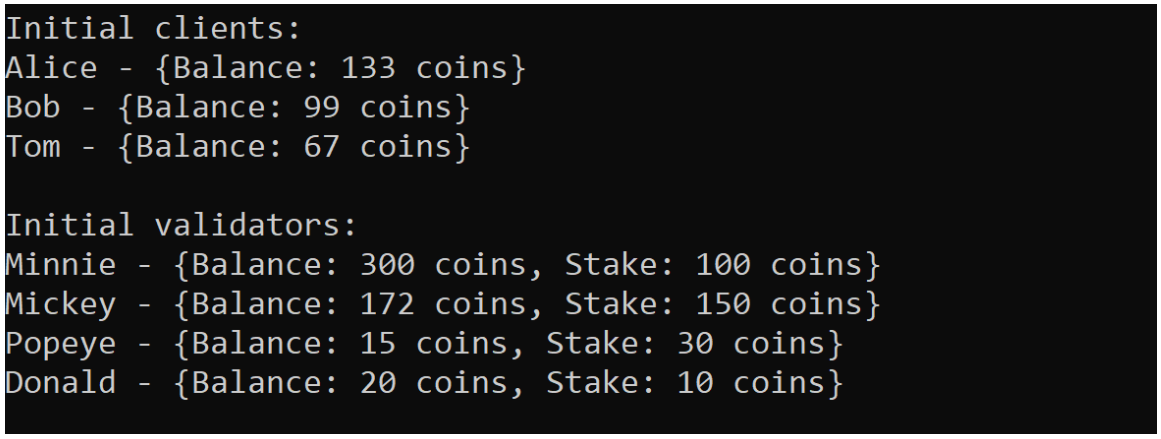
This chapter contrasts five different normal operations in both the Tendermint and TontineCoin models. These operations are initialization, the consensus process, transfer of coins among clients, bonding of clients, and unbonding of validators. Each system's *Driver* module is executed on Command Prompt on Windows 10 operating system installed with Node.js execution environment. Tendermint model is simulated with a network of 3 clients and 4 validators while TontineCoin with a network of 9 clients and 5 validators. The results are in the form of logs displayed on Command Prompt.

5.1 Initialization

Initialization operation creates initial clients with their coins in their wallets, initial validators with their stakes bonded in the system and a genesis block containing a *REG* and *BND* transactions. It is similar in both the models except validators of TontineCoin have shares in their tontines and their tontine IDs. Also, a genesis tontine is created in the TontineCoin system for initial validators to *bid* and *bond* to the system.

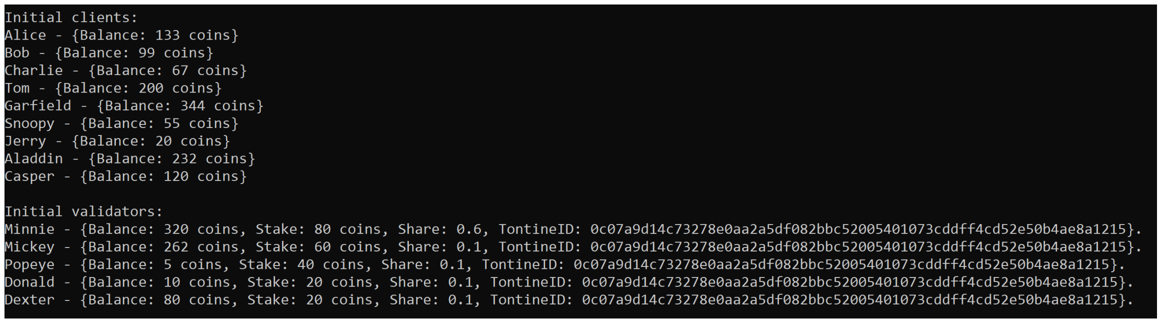
Figures 2 and 3 show initial balances of clients and validators in both the systems, respectively. Tendermint system is initialized with 3 clients and 4 validators while TontineCoin with 9 clients and 5 validators. In both the systems, the clients have their coins in their wallets and validators have their stakes bonded in their systems. Only validators of TontineCoin have shares in their tontines and their tontine IDs.

Figure 4 shows a genesis tontine of 5 validators, each of which has a share and a stake. The tontine has a nonce, members and an ID. Unlike other tontines, BND transactions are not included in a genesis tontine. They are created during the creation of the genesis block.



```
Initial clients:
Alice - {Balance: 133 coins}
Bob - {Balance: 99 coins}
Tom - {Balance: 67 coins}

Initial validators:
Minnie - {Balance: 300 coins, Stake: 100 coins}
Mickey - {Balance: 172 coins, Stake: 150 coins}
Popeye - {Balance: 15 coins, Stake: 30 coins}
Donald - {Balance: 20 coins, Stake: 10 coins}
```

Figure 2: Initial balances of 3 clients and 4 validators in Tendermint



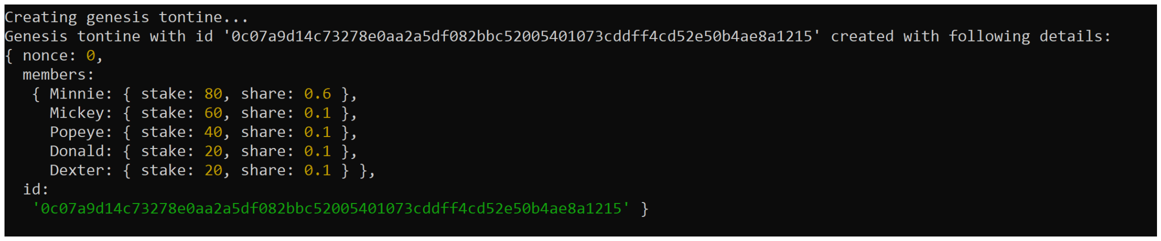
```
Initial clients:
Alice - {Balance: 133 coins}
Bob - {Balance: 99 coins}
Charlie - {Balance: 67 coins}
Tom - {Balance: 200 coins}
Garfield - {Balance: 344 coins}
Snoopy - {Balance: 55 coins}
Jerry - {Balance: 20 coins}
Aladdin - {Balance: 232 coins}
Casper - {Balance: 120 coins}

Initial validators:
Minnie - {Balance: 320 coins, Stake: 80 coins, Share: 0.6, TontineID: 0c07a9d14c73278e0aa2a5df082bbc52005401073cddff4cd52e50b4ae8a1215}.
Mickey - {Balance: 262 coins, Stake: 60 coins, Share: 0.1, TontineID: 0c07a9d14c73278e0aa2a5df082bbc52005401073cddff4cd52e50b4ae8a1215}.
Popeye - {Balance: 5 coins, Stake: 40 coins, Share: 0.1, TontineID: 0c07a9d14c73278e0aa2a5df082bbc52005401073cddff4cd52e50b4ae8a1215}.
Donald - {Balance: 10 coins, Stake: 20 coins, Share: 0.1, TontineID: 0c07a9d14c73278e0aa2a5df082bbc52005401073cddff4cd52e50b4ae8a1215}.
Dexter - {Balance: 80 coins, Stake: 20 coins, Share: 0.1, TontineID: 0c07a9d14c73278e0aa2a5df082bbc52005401073cddff4cd52e50b4ae8a1215}.
```

Figure 3: Initial balances of 9 clients and 5 validators in TontineCoin

Figures 5 and 6 show initial UTXOs held by validators in both the systems, respectively. Initial REG and BND outputs can be seen in these figures.

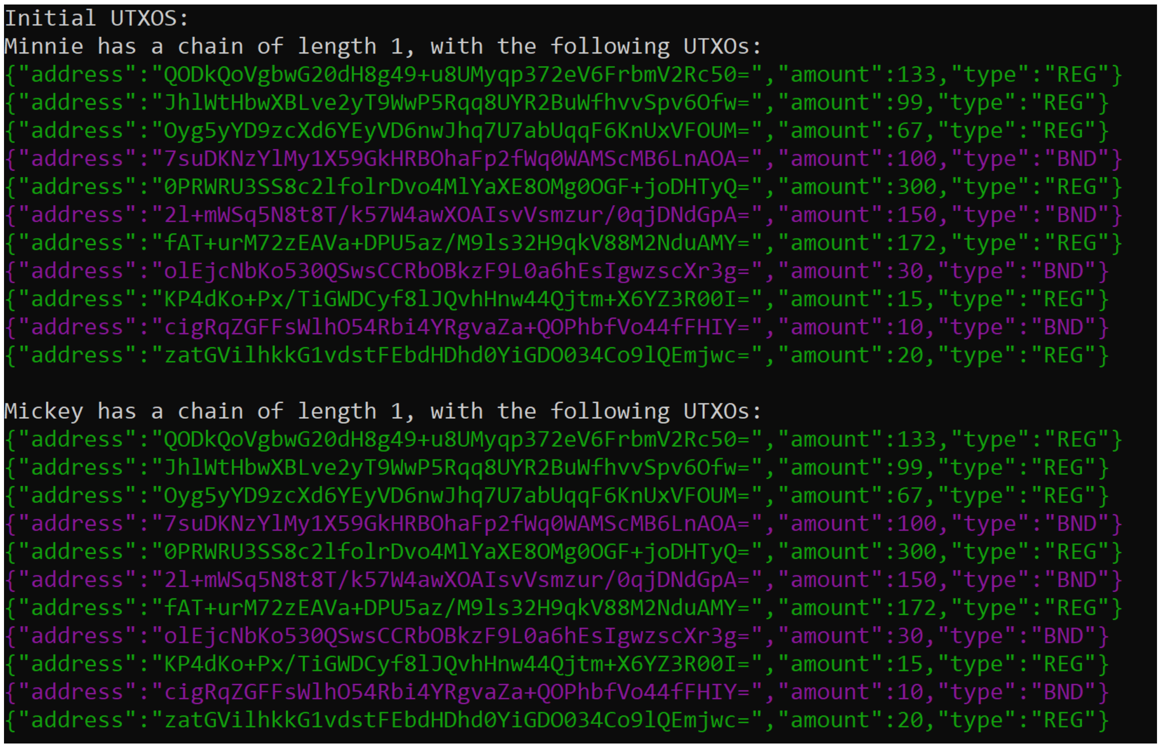
5.2 Consensus process

In both the systems, a round of consensus includes unbonding of validators, syncing of validators bonded in the last round, election of a proposer, creation and



```
Creating genesis tontine...
Genesis tontine with id '0c07a9d14c73278e0aa2a5df082bbc52005401073cddff4cd52e50b4ae8a1215' created with following details:
{ nonce: 0,
  members:
   { Minnie: { stake: 80, share: 0.6 },
     Mickey: { stake: 60, share: 0.1 },
     Popeye: { stake: 40, share: 0.1 },
     Donald: { stake: 20, share: 0.1 },
     Dexter: { stake: 20, share: 0.1 } },
  id:
   '0c07a9d14c73278e0aa2a5df082bbc52005401073cddff4cd52e50b4ae8a1215' }
```

Figure 4: A genesis tontine of 5 validators in TontineCoin



```
Initial UTXOS:
Minnie has a chain of length 1, with the following UTXOs:
{"address":"QODkQoVgbwG20dH8g49+u8UMyqp372eV6FrbmV2Rc50=","amount":133,"type":"REG"}
{"address":"JhlWtHbwXBLve2yT9WwP5Rqq8UYR2BuWfhvvSpv6Ofw=","amount":99,"type":"REG"}
{"address":"Oyg5yYD9zcXd6YEyVD6nwJhq7U7abUqqF6KnUxVFOUM=","amount":67,"type":"REG"}
{"address":"7suDKNzYlMy1X59GkHRBOhaFp2fWq0WAMScMB6LnAOA=","amount":100,"type":"BND"}
{"address":"0PRWRU3SS8c2lfolrDvo4MlYaXE8OMg0OGF+joDHTyQ=","amount":300,"type":"REG"}
{"address":"2l+mWSq5N8t8T/k57W4awXOAIsvVsmzur/0qjDNdGpA=","amount":150,"type":"BND"}
{"address":"fAT+urM72zEAVa+DPU5az/M9ls32H9qkV88M2NduAMY=","amount":172,"type":"REG"}
{"address":"olEjcNbKo530QSwsCCRbOBkzF9L0a6hEsIgwzscXr3g=","amount":30,"type":"BND"}
{"address":"KP4dKo+Px/TiGWDCyf8lJQvhHnw44Qjtm+X6YZ3R00I=","amount":15,"type":"REG"}
{"address":"cigRqZGFFsWlhO54Rbi4YRgvaZa+QOPhbfVo44fFHIY=","amount":10,"type":"BND"}
{"address":"zatGVilhkkG1vdstFEbdHDhd0YiGDO034Co9lQEmjwc=","amount":20,"type":"REG"}

Mickey has a chain of length 1, with the following UTXOs:
{"address":"QODkQoVgbwG20dH8g49+u8UMyqp372eV6FrbmV2Rc50=","amount":133,"type":"REG"}
{"address":"JhlWtHbwXBLve2yT9WwP5Rqq8UYR2BuWfhvvSpv6Ofw=","amount":99,"type":"REG"}
{"address":"Oyg5yYD9zcXd6YEyVD6nwJhq7U7abUqqF6KnUxVFOUM=","amount":67,"type":"REG"}
{"address":"7suDKNzYlMy1X59GkHRBOhaFp2fWq0WAMScMB6LnAOA=","amount":100,"type":"BND"}
{"address":"0PRWRU3SS8c2lfolrDvo4MlYaXE8OMg0OGF+joDHTyQ=","amount":300,"type":"REG"}
{"address":"2l+mWSq5N8t8T/k57W4awXOAIsvVsmzur/0qjDNdGpA=","amount":150,"type":"BND"}
{"address":"fAT+urM72zEAVa+DPU5az/M9ls32H9qkV88M2NduAMY=","amount":172,"type":"REG"}
{"address":"olEjcNbKo530QSwsCCRbOBkzF9L0a6hEsIgwzscXr3g=","amount":30,"type":"BND"}
{"address":"KP4dKo+Px/TiGWDCyf8lJQvhHnw44Qjtm+X6YZ3R00I=","amount":15,"type":"REG"}
{"address":"cigRqZGFFsWlhO54Rbi4YRgvaZa+QOPhbfVo44fFHIY=","amount":10,"type":"BND"}
{"address":"zatGVilhkkG1vdstFEbdHDhd0YiGDO034Co9lQEmjwc=","amount":20,"type":"REG"}
```

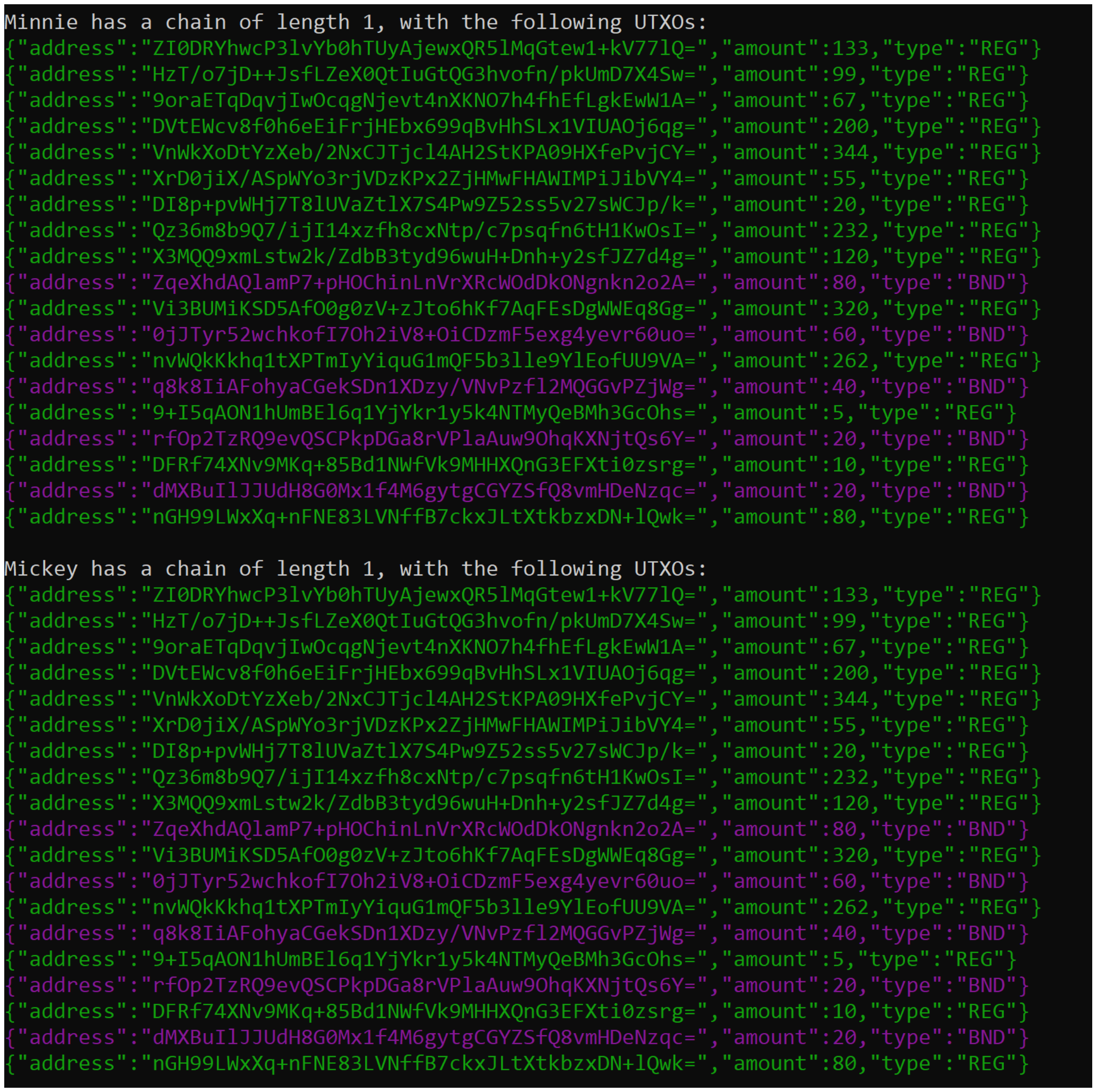
Figure 5: Initial balances of 3 clients and 4 validators in Tendermint

broadcast of a proposal by the elected proposer, its validation by other validators, prevote and commit stages, ejecting cheating validators, bonding validators, receiving outputs, removal of transactions already committed, and addition of the proposal in the blockchain. In addition to these operations, a validator in TontineCoin also supports tontines' BID transactions and selects *currentBidLeader's* validators to bond when the selection block is reached (discussed in 5.4.2).

Figures 7 and 8 show a round of consensus in both the systems, respectively. In figure 7, Mickey is elected as the proposer in the Tendermint system, while in Figure 8, Minnie is elected in the TontineCoin system.

5.3 Transfer of coins from a client to another

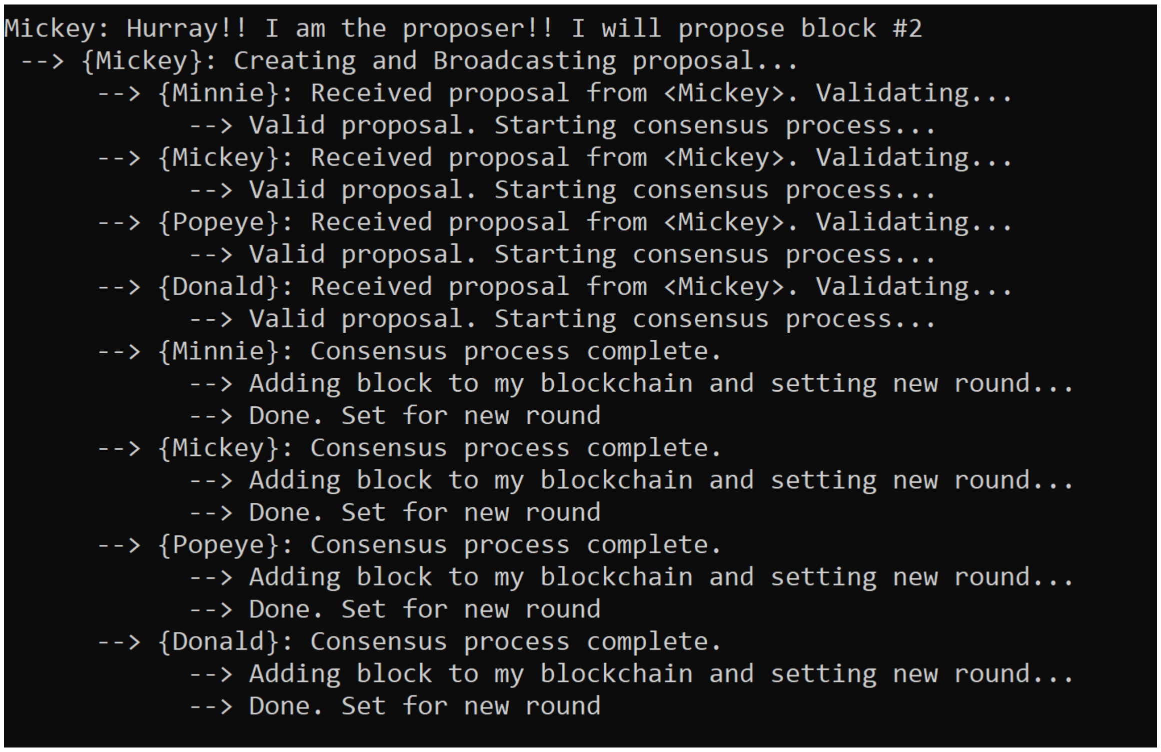
The process of transferring coins from a client to another is the same in both the systems. Figures 9, 10 and 11 show a transfer of 40 coins from Alice to Tom in



```
Minnie has a chain of length 1, with the following UTXOs:
{"address":"ZI0DRYhwcP3lvYb0hTUyAjewxQR5lMqGtew1+kV77lQ=","amount":133,"type":"REG"}
{"address":"HzT/o7jD++JsfLZeX0QtIuGtQG3hvofn/pkUmD7X4Sw=","amount":99,"type":"REG"}
{"address":"9oraETqDqvjIwOcqgNjevt4nXKNO7h4fhEfLgkEwW1A=","amount":67,"type":"REG"}
{"address":"DVtEWcv8f0h6eEiFrjHEbx699qBvHhSLx1VIUAOj6qg=","amount":200,"type":"REG"}
{"address":"VnWkXoDtYzXeb/2NxCJTjcl4AH2StKPA09HXfePvjCY=","amount":344,"type":"REG"}
{"address":"XrD0jiX/ASpWYo3rjVDzKPx2ZjHMwFHAWIMPiJibVY4=","amount":55,"type":"REG"}
{"address":"DI8p+pvWHj7T8lUVaZtlX7S4Pw9Z52ss5v27sWCJp/k=","amount":20,"type":"REG"}
{"address":"Qz36m8b9Q7/ijI14xzfh8cxNtp/c7psqfn6tH1KwOsI=","amount":232,"type":"REG"}
{"address":"X3MQQ9xmLstw2k/ZdbB3tyd96wuH+Dnh+y2sfJZ7d4g=","amount":120,"type":"REG"}
{"address":"ZqeXhdAQlamP7+pHOChinLnVrXRcWOdDkONgnkn2o2A=","amount":80,"type":"BND"}
{"address":"Vi3BUMiKSD5AfO0g0zV+zJto6hKf7AqFEsDgWWEq8Gg=","amount":320,"type":"REG"}
{"address":"0jJTyr52wchkofI7Oh2iV8+OiCDzmF5exg4yevr60uo=","amount":60,"type":"BND"}
{"address":"nvWQkKkhq1tXPTmIyYiquG1mQF5b3lle9YlEofUU9VA=","amount":262,"type":"REG"}
{"address":"q8k8IiAFohyaCGekSDn1XDzy/VNvPzfl2MQGGvPZjWg=","amount":40,"type":"BND"}
{"address":"9+I5qAON1hUmBEl6q1YjYkr1y5k4NTMyQeBMh3GcOhs=","amount":5,"type":"REG"}
{"address":"rfOp2TzRQ9evQSCPkpDGa8rVPlaAuw9OhqKXNjtQs6Y=","amount":20,"type":"BND"}
{"address":"DFRf74XNv9MKq+85Bd1NWfVk9MHHXQnG3EFXti0zsrg=","amount":10,"type":"REG"}
{"address":"dMXBuIlJJUdH8G0Mx1f4M6gytgCGYZSfQ8vmHDeNzqc=","amount":20,"type":"BND"}
{"address":"nGH99LWxXq+nFNE83LVNffB7ckxJLtXtkbzxDN+lQwk=","amount":80,"type":"REG"}

Mickey has a chain of length 1, with the following UTXOs:
{"address":"ZI0DRYhwcP3lvYb0hTUyAjewxQR5lMqGtew1+kV77lQ=","amount":133,"type":"REG"}
{"address":"HzT/o7jD++JsfLZeX0QtIuGtQG3hvofn/pkUmD7X4Sw=","amount":99,"type":"REG"}
{"address":"9oraETqDqvjIwOcqgNjevt4nXKNO7h4fhEfLgkEwW1A=","amount":67,"type":"REG"}
{"address":"DVtEWcv8f0h6eEiFrjHEbx699qBvHhSLx1VIUAOj6qg=","amount":200,"type":"REG"}
{"address":"VnWkXoDtYzXeb/2NxCJTjcl4AH2StKPA09HXfePvjCY=","amount":344,"type":"REG"}
{"address":"XrD0jiX/ASpWYo3rjVDzKPx2ZjHMwFHAWIMPiJibVY4=","amount":55,"type":"REG"}
{"address":"DI8p+pvWHj7T8lUVaZtlX7S4Pw9Z52ss5v27sWCJp/k=","amount":20,"type":"REG"}
{"address":"Qz36m8b9Q7/ijI14xzfh8cxNtp/c7psqfn6tH1KwOsI=","amount":232,"type":"REG"}
{"address":"X3MQQ9xmLstw2k/ZdbB3tyd96wuH+Dnh+y2sfJZ7d4g=","amount":120,"type":"REG"}
{"address":"ZqeXhdAQlamP7+pHOChinLnVrXRcWOdDkONgnkn2o2A=","amount":80,"type":"BND"}
{"address":"Vi3BUMiKSD5AfO0g0zV+zJto6hKf7AqFEsDgWWEq8Gg=","amount":320,"type":"REG"}
{"address":"0jJTyr52wchkofI7Oh2iV8+OiCDzmF5exg4yevr60uo=","amount":60,"type":"BND"}
{"address":"nvWQkKkhq1tXPTmIyYiquG1mQF5b3lle9YlEofUU9VA=","amount":262,"type":"REG"}
{"address":"q8k8IiAFohyaCGekSDn1XDzy/VNvPzfl2MQGGvPZjWg=","amount":40,"type":"BND"}
{"address":"9+I5qAON1hUmBEl6q1YjYkr1y5k4NTMyQeBMh3GcOhs=","amount":5,"type":"REG"}
{"address":"rfOp2TzRQ9evQSCPkpDGa8rVPlaAuw9OhqKXNjtQs6Y=","amount":20,"type":"BND"}
{"address":"DFRf74XNv9MKq+85Bd1NWfVk9MHHXQnG3EFXti0zsrg=","amount":10,"type":"REG"}
{"address":"dMXBuIlJJUdH8G0Mx1f4M6gytgCGYZSfQ8vmHDeNzqc=","amount":20,"type":"BND"}
{"address":"nGH99LWxXq+nFNE83LVNffB7ckxJLtXtkbzxDN+lQwk=","amount":80,"type":"REG"}
```

Figure 6: Initial balances of 9 clients and 5 validators in TontineCoin

Tendermint. Figure 9 shows Alice creating a REG transaction to transfer coins at an address provided by Tom. The same figure shows Popeye validating and adding Alice's REG transaction in his block. Other validators receive the block and confirm it as a valid one. Figure 10 shows UTXOs held by Minnie that includes Tom's UTXO and Alice's updated UTXO. Figure 11 shows their updated wallets. An extra coin is deducted from Alice's wallet as a transaction fee, which is actually not collected by



```
Mickey: Hurray!! I am the proposer!! I will propose block #2
 --> {Mickey}: Creating and Broadcasting proposal...
      --> {Minnie}: Received proposal from <Mickey>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Mickey}: Received proposal from <Mickey>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Popeye}: Received proposal from <Mickey>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Donald}: Received proposal from <Mickey>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Minnie}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round
      --> {Mickey}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round
      --> {Popeye}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round
      --> {Donald}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round
```

Figure 7: A round of consensus in Tendermint

any validator. This can be a rectified in the revised implementation.

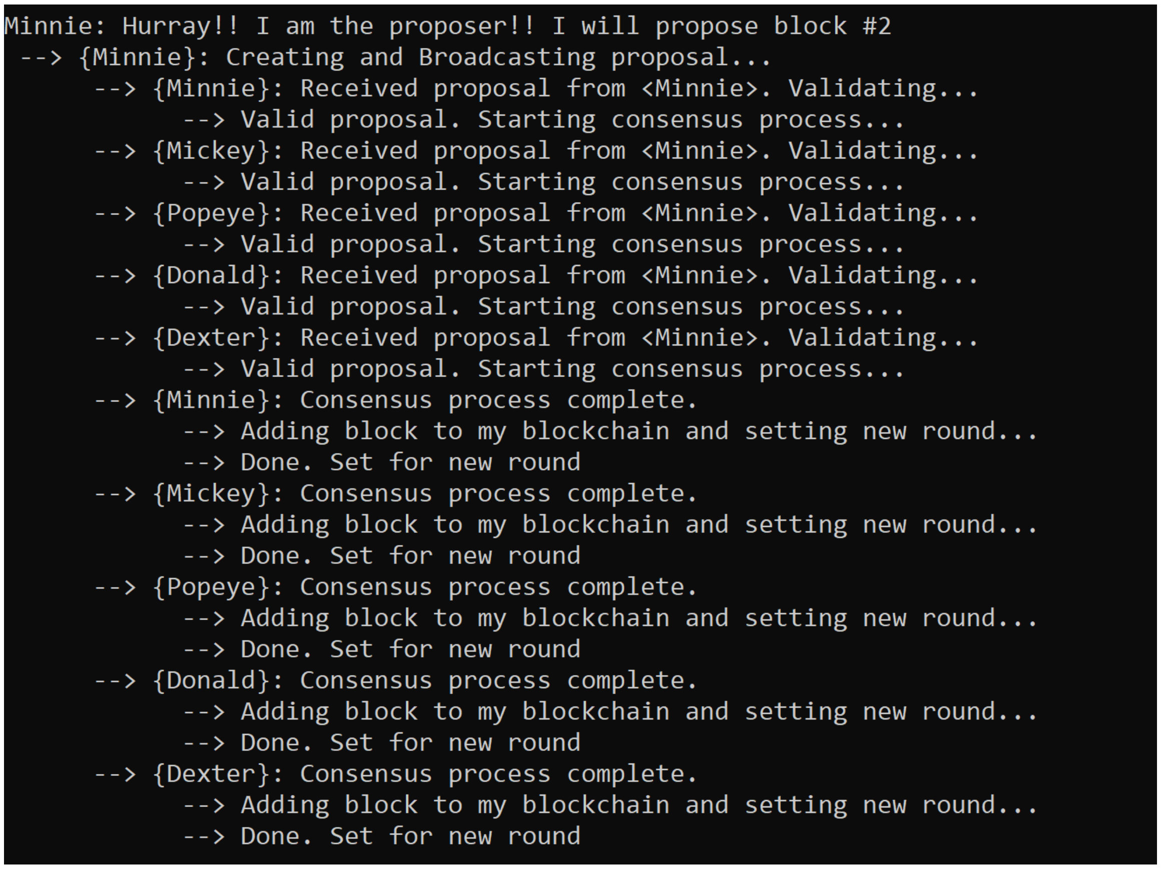
Figures 12, 13, and 14 show a transfer of 10 coins from Alice to Bob in TontineCoin in a similar way.

5.4 Bonding of a client to the validators' network

Bonding of a client to the validators' network in TontineCoin is different than that in Tendermint. Clients in Tendermint submit BND transaction to get in the validators' network while those in TontineCoin form tontines and submit BID transactions to get in.

5.4.1 Bonding of a client in Tendermint

Figure 15 shows Tom posting a BND transaction to be a validator. Figure 16 shows Donald adding Tom's BND transaction in his proposal. Figure 18 shows UTXOs held by Popeye consisting of Tom's UTXO. Figure 17 shows Tom as a proposer.

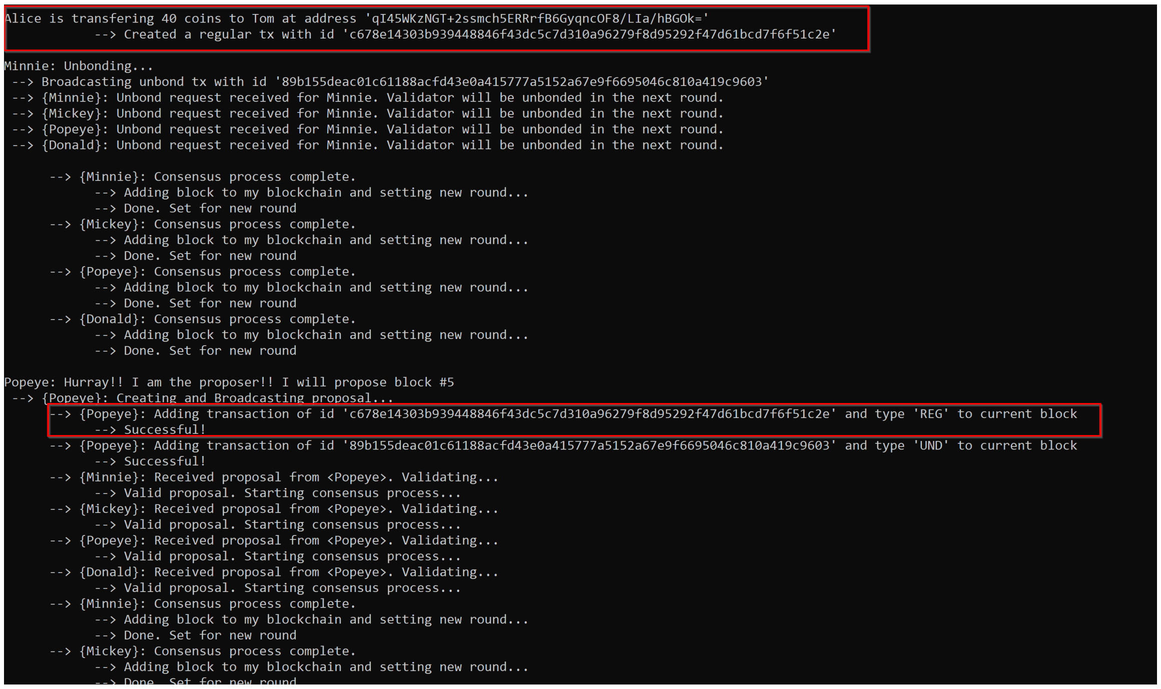


```
Minnie: Hurray!! I am the proposer!! I will propose block #2
 --> {Minnie}: Creating and Broadcasting proposal...
      --> {Minnie}: Received proposal from <Minnie>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Mickey}: Received proposal from <Minnie>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Popeye}: Received proposal from <Minnie>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Donald}: Received proposal from <Minnie>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Dexter}: Received proposal from <Minnie>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Minnie}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round
      --> {Mickey}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round
      --> {Popeye}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round
      --> {Donald}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round
      --> {Dexter}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round
```

Figure 8: A round of consensus in TontineCoin

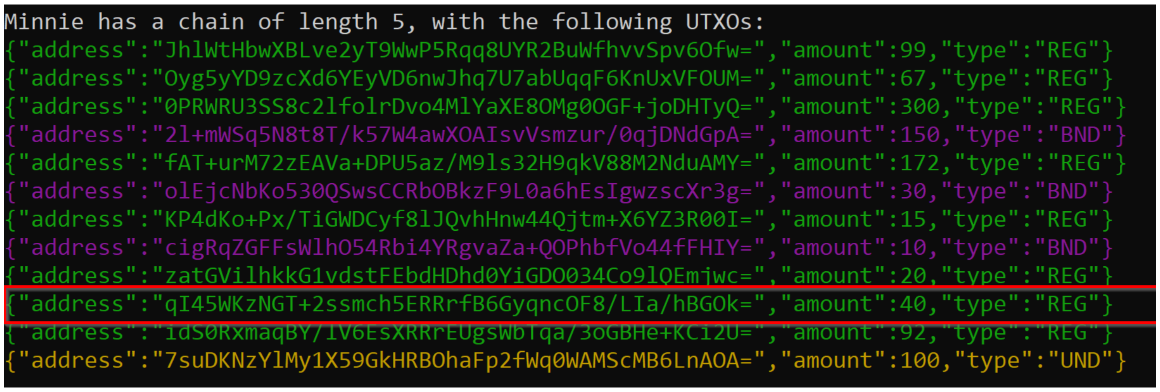
5.4.2 Bonding of a client in TontineCoin

Figure 19 shows Tom, Garfield, Snoopy, and Jerry creating a tontine and Aladdin and Casper creating another one. Tom and Aladdin submit BID transactions for their respective tontine. Figure 20 shows Dexter adding both the BID transactions in his proposal. Figure 21 shows selection of Aladdin's tontine by each of validators at the selection block height. Figure 22 shows bonding of Aladdin and Casper to the validators' network. Figure 23 shows Aladdin as a proposer. Figure 24 shows UTXOs held by Popeye consisting of Aladdin's and Casper's UTXOs. Aladdin's UTXO is changed because it is seized by his tontine members (cheating scenario discussed in 6.2).



```
Alice is transfering 40 coins to Tom at address 'qI45WKzNGT+2ssmch5ERRrfB6GyqncOF8/LIa/hBGOk='
            --> Created a regular tx with id 'c678e14303b939448846f43dc5c7d310a96279f8d95292f47d61bcd7f6f51c2e'

Minnie: Unbonding...
 --> Broadcasting unbond tx with id '89b155deac01c61188acfd43e0a415777a5152a67e9f6695046c810a419c9603'
 --> {Minnie}: Unbond request received for Minnie. Validator will be unbonded in the next round.
 --> {Mickey}: Unbond request received for Minnie. Validator will be unbonded in the next round.
 --> {Popeye}: Unbond request received for Minnie. Validator will be unbonded in the next round.
 --> {Donald}: Unbond request received for Minnie. Validator will be unbonded in the next round.

      --> {Minnie}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round
      --> {Mickey}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round
      --> {Popeye}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round
      --> {Donald}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round

Popeye: Hurray!! I am the proposer!! I will propose block #5
 --> {Popeye}: Creating and Broadcasting proposal...
      --> {Popeye}: Adding transaction of id 'c678e14303b939448846f43dc5c7d310a96279f8d95292f47d61bcd7f6f51c2e' and type 'REG' to current block
            --> Successful!
      --> {Popeye}: Adding transaction of id '89b155deac01c61188acfd43e0a415777a5152a67e9f6695046c810a419c9603' and type 'UND' to current block
            --> Successful!
      --> {Minnie}: Received proposal from <Popeye>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Mickey}: Received proposal from <Popeye>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Popeye}: Received proposal from <Popeye>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Donald}: Received proposal from <Popeye>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Minnie}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round
      --> {Mickey}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round
```

Figure 9: Creation and inclusion of REG transaction in Tendermint

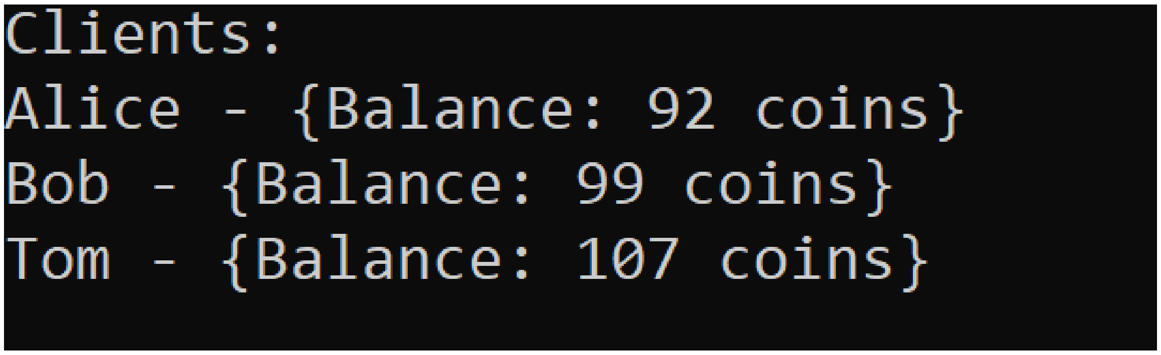


```
Minnie has a chain of length 5, with the following UTXOs:
{"address":"JhlWtHbwXBLve2yT9WwP5Rqq8UYR2BuWfhvvSpv6Ofw=","amount":99,"type":"REG"}
{"address":"Oyg5yYD9zcXd6YEyVD6nwJhq7U7abUqqF6KnUxVFOUM=","amount":67,"type":"REG"}
{"address":"0PRWRU3SS8c2lfolrDvo4MlYaXE8OMg0OGF+joDHTyQ=","amount":300,"type":"REG"}
{"address":"2l+mWSq5N8t8T/k57W4awXOAIsvVsmzur/0qjDNdGpA=","amount":150,"type":"BND"}
{"address":"fAT+urM72zEAVa+DPU5az/M9ls32H9qkV88M2NduAMY=","amount":172,"type":"REG"}
{"address":"olEjcNbKo530QSwsCCRbOBkzF9L0a6hEsIgwzscXr3g=","amount":30,"type":"BND"}
{"address":"KP4dKo+Px/TiGWDCyf8lJQvhHnw44Qjtm+X6YZ3R00I=","amount":15,"type":"REG"}
{"address":"cigRqZGFFsWlhO54Rbi4YRgvaZa+QOPhbfVo44fFHIY=","amount":10,"type":"BND"}
{"address":"zatGVilhkkG1vdstFEbdHDhd0YiGDO034Co9lQEmjwc=","amount":20,"type":"REG"}
{"address":"qI45WKzNGT+2ssmch5ERRrfB6GyqncOF8/LIa/hBGOk=","amount":40,"type":"REG"}
{"address":"idS0RxmaqBY/1V6EsXRRrEUgsWbTqa/3oGBHe+KCi2U=","amount":92,"type":"REG"}
{"address":"7suDKNzYlMy1X59GkHRBOhaFp2fWq0WAMScMB6LnAOA=","amount":100,"type":"UND"}
```

Figure 10: UTXOs after the REG transaction created in Figure 9 in Tendermint

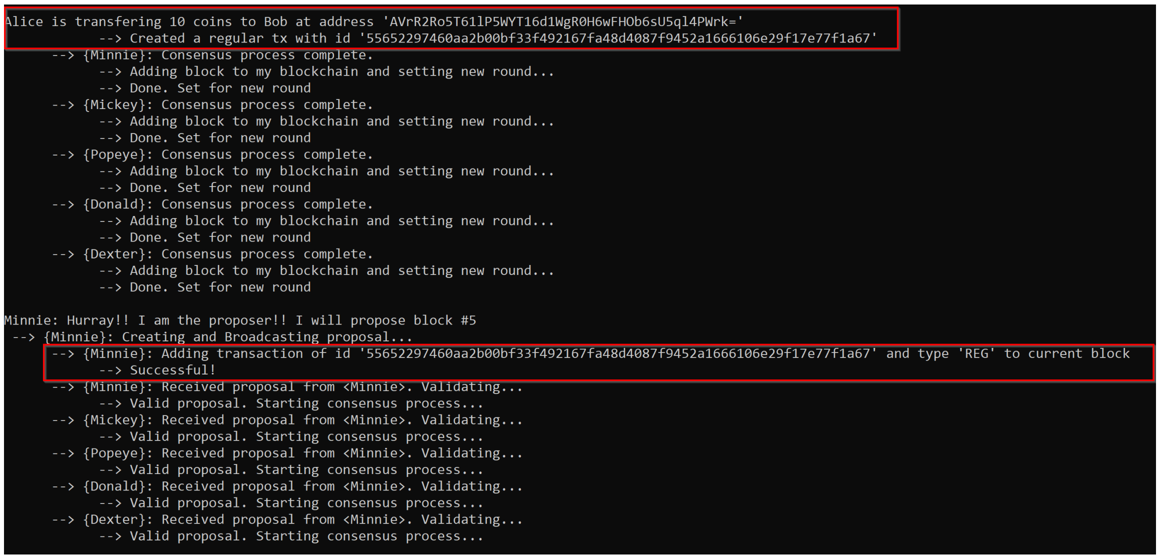
5.5 Unbonding of a validator from the validators' network

Unbonding of a validator from the validators' network in TontineCoin is similar to that of in Tendermint. Valdators in both the systems submit a UND transaction to receive their stake back and leave the network. However, in TontineCoin, those validators' shares are also updated who belonged to unbonded validator's tontine.



```
Clients:
Alice - {Balance: 92 coins}
Bob - {Balance: 99 coins}
Tom - {Balance: 107 coins}
```

Figure 11: Balances after the REG transaction created in Figure 9 in Tendermint



```
Alice is transfering 10 coins to Bob at address 'AVrR2Ro5T61lP5WYT16d1WgR0H6wFHOb6sU5ql4PWrk='
            --> Created a regular tx with id '55652297460aa2b00bf33f492167fa48d4087f9452a1666106e29f17e77f1a67'
      --> {Minnie}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round
      --> {Mickey}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round
      --> {Popeye}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round
      --> {Donald}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round
      --> {Dexter}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round

Minnie: Hurray!! I am the proposer!! I will propose block #5
 --> {Minnie}: Creating and Broadcasting proposal...
      --> {Minnie}: Adding transaction of id '55652297460aa2b00bf33f492167fa48d4087f9452a1666106e29f17e77f1a67' and type 'REG' to current block
            --> Successful!
      --> {Minnie}: Received proposal from <Minnie>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Mickey}: Received proposal from <Minnie>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Popeye}: Received proposal from <Minnie>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Donald}: Received proposal from <Minnie>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Dexter}: Received proposal from <Minnie>. Validating...
            --> Valid proposal. Starting consensus process...
```

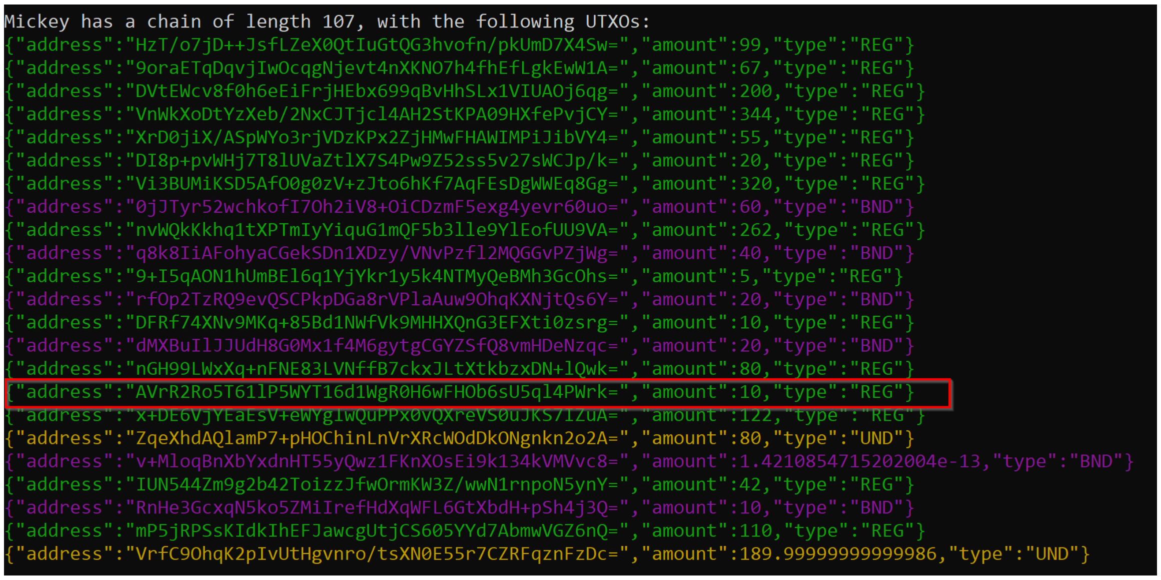
Figure 12: Creation and inclusion of REG transaction in TontineCoin

5.5.1 Unbonding of a validator in Tendermint

Figure 25 shows Minnie creating and broadcasting a UND transaction. Figure 26 shows Donald adding it in his proposal and Minnie unbonding after it is committed. Figure 27 shows a round of block creation after Minnie is unbonded. Figure 28 shows Mickey's UTXOs consisting of Minnie's unbonded UTXO.

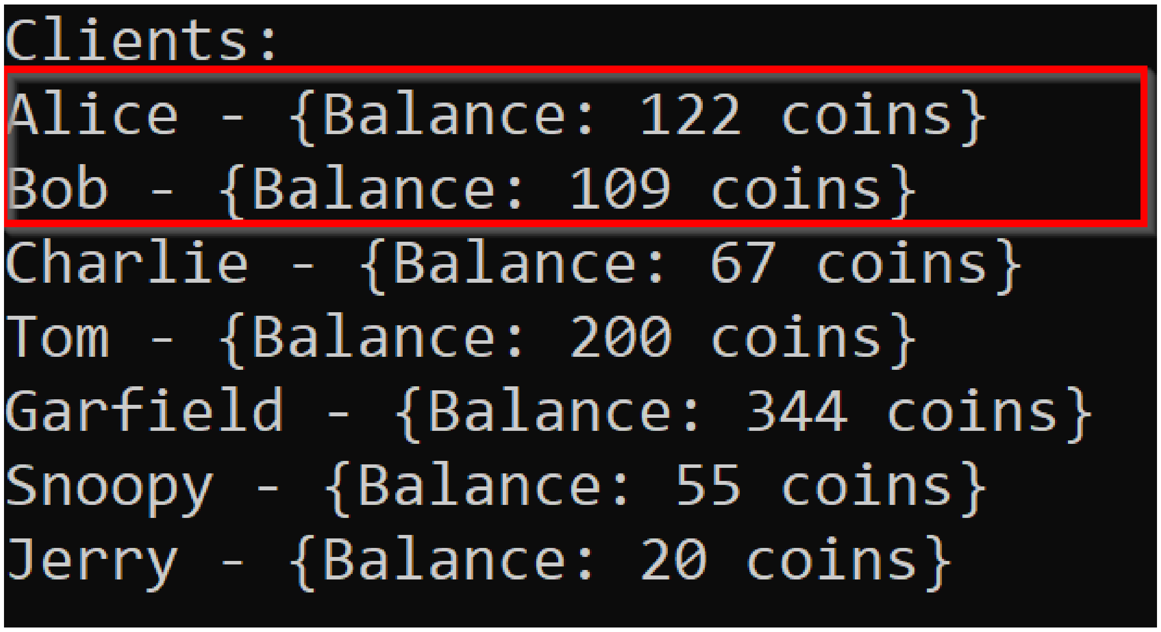
5.5.2 Unbonding of a validator in TontineCoin

Figure 29 shows Minnie, a validator of tontine shown in Figure 4, creating and broadcasting a UND transaction. Figure 30 shows Donald adding it in his proposal and



```
Mickey has a chain of length 107, with the following UTXOs:
{"address":"HzT/o7jD++JsfLZeX0QtIuGtQG3hvofn/pkUmD7X4Sw=","amount":99,"type":"REG"}
{"address":"9oraETqDqvjIwOcqgNjevt4nXKNO7h4fhEfLgkEwW1A=","amount":67,"type":"REG"}
{"address":"DVtEWcv8f0h6eEiFrjHEbx699qBvHhSLx1VIUAOj6qg=","amount":200,"type":"REG"}
{"address":"VnWkXoDtYzXeb/2NxCJTjcl4AH2StKPA09HXfePvjCY=","amount":344,"type":"REG"}
{"address":"XrD0jiX/ASpWYo3rjVDzKPx2ZjHMwFHAWIMPiJibVY4=","amount":55,"type":"REG"}
{"address":"DI8p+pvWHj7T8lUVaZtlX7S4Pw9Z52ss5v27sWCJp/k=","amount":20,"type":"REG"}
{"address":"Vi3BUMiKSD5AfO0g0zV+zJto6hKf7AqFEsDgWWEq8Gg=","amount":320,"type":"REG"}
{"address":"0jJTyr52wchkofI7Oh2iV8+OiCDzmF5exg4yevr60uo=","amount":60,"type":"BND"}
{"address":"nvWQkKkhq1tXPTmIyYiquG1mQF5b3lle9YlEofUU9VA=","amount":262,"type":"REG"}
{"address":"q8k8IiAFohyaCGekSDn1XDzy/VNvPzfl2MQGGvPZjWg=","amount":40,"type":"BND"}
{"address":"9+I5qAON1hUmBEl6q1YjYkr1y5k4NTMyQeBMh3GcOhs=","amount":5,"type":"REG"}
{"address":"rfOp2TzRQ9evQSCPkpDGa8rVPlaAuw9OhqKXNjtQs6Y=","amount":20,"type":"BND"}
{"address":"DFRf74XNv9MKq+85Bd1NWfVk9MHHXQnG3EFXti0zsrg=","amount":10,"type":"REG"}
{"address":"dMXBuIlJJUdH8G0Mx1f4M6gytgCGYZSfQ8vmHDeNzqc=","amount":20,"type":"BND"}
{"address":"nGH99LWxXq+nFNE83LVNffB7ckxJLtXtkbzxDN+lQwk=","amount":80,"type":"REG"}
{"address":"AVrR2Ro5T61lP5WYT16d1WgR0H6wFHOb6sU5ql4PWrk=","amount":10,"type":"REG"}
{"address":"x+DE6VjYEaEsV+eWYglWQuPPx0vQXreVS0uJKS7IZuA=","amount":122,"type":"REG"}
{"address":"ZqeXhdAQlamP7+pHOChinLnVrXRcWOdDkONgnkn2o2A=","amount":80,"type":"UND"}
{"address":"v+MloqBnXbYxdnHT55yQwz1FKnXOsEi9k134kVMVvc8=","amount":1.4210854715202004e-13,"type":"BND"}
{"address":"IUN544Zm9g2b42ToizzJfwOrmKW3Z/wwN1rnpoN5ynY=","amount":42,"type":"REG"}
{"address":"RnHe3GcxqN5ko5ZMiIrefHdXqWFL6GtXbdH+pSh4j3Q=","amount":10,"type":"BND"}
{"address":"mP5jRPSsKIdkIhEFJawcgUtjCS605YYd7AbmwVGZ6nQ=","amount":110,"type":"REG"}
{"address":"VrfC9OhqK2pIvUtHgvnro/tsXN0E55r7CZRFqznFzDc=","amount":189.99999999999986,"type":"UND"}
```

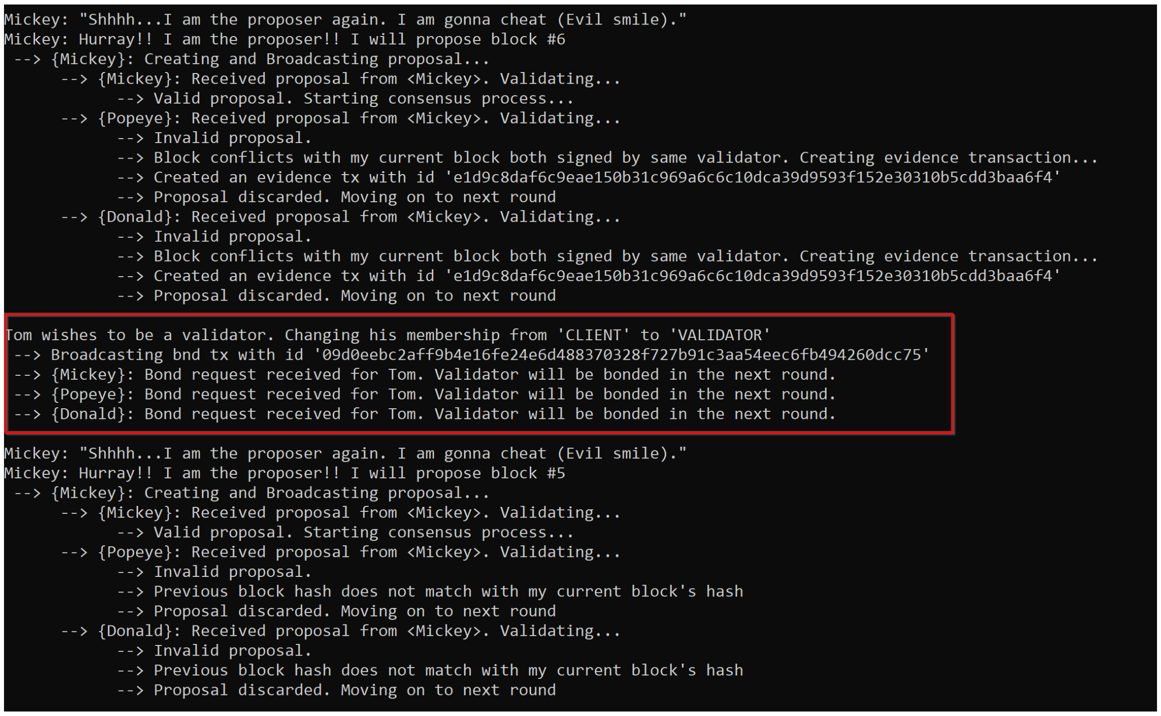
Figure 13: UTXOs after the REG transaction created in Figure 12 in TontineCoin



```
Clients:
Alice - {Balance: 122 coins}
Bob - {Balance: 109 coins}
Charlie - {Balance: 67 coins}
Tom - {Balance: 200 coins}
Garfield - {Balance: 344 coins}
Snoopy - {Balance: 55 coins}
Jerry - {Balance: 20 coins}
```

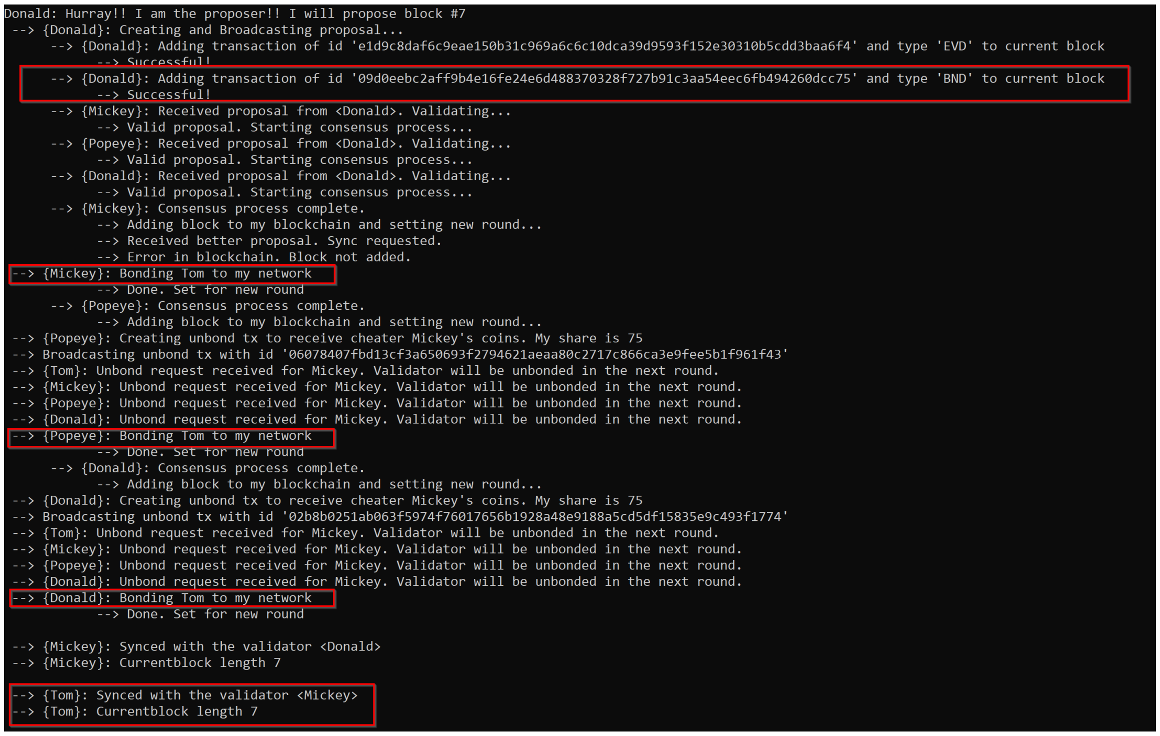
Figure 14: Balances after the REG transaction created in Figure 12 in TontineCoin

validators updating other validators' shares after Minnie unbonding. Since Mickey, Popeye, Donald, and Dexter belonged to Minnie's tontine, their shares are updated. Figure 31 shows a round of block creation after Minnie is unbonded. Figure 32 shows Mickey's UTXOs consisting of Minnie's unbonded UTXO.



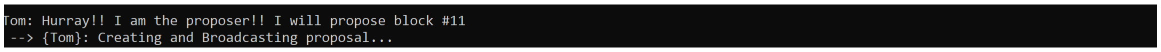
```
Mickey: "Shhhh...I am the proposer again. I am gonna cheat (Evil smile)."
Mickey: Hurray!! I am the proposer!! I will propose block #6
 --> {Mickey}: Creating and Broadcasting proposal...
      --> {Mickey}: Received proposal from <Mickey>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Popeye}: Received proposal from <Mickey>. Validating...
            --> Invalid proposal.
            --> Block conflicts with my current block both signed by same validator. Creating evidence transaction...
            --> Created an evidence tx with id 'e1d9c8daf6c9eae150b31c969a6c6c10dca39d9593f152e30310b5cdd3baa6f4'
            --> Proposal discarded. Moving on to next round
      --> {Donald}: Received proposal from <Mickey>. Validating...
            --> Invalid proposal.
            --> Block conflicts with my current block both signed by same validator. Creating evidence transaction...
            --> Created an evidence tx with id 'e1d9c8daf6c9eae150b31c969a6c6c10dca39d9593f152e30310b5cdd3baa6f4'
            --> Proposal discarded. Moving on to next round

Tom wishes to be a validator. Changing his membership from 'CLIENT' to 'VALIDATOR'
 --> Broadcasting bnd tx with id '09d0eebc2aff9b4e16fe24e6d488370328f727b91c3aa54eec6fb494260dcc75'
 --> {Mickey}: Bond request received for Tom. Validator will be bonded in the next round.
 --> {Popeye}: Bond request received for Tom. Validator will be bonded in the next round.
 --> {Donald}: Bond request received for Tom. Validator will be bonded in the next round.

Mickey: "Shhhh...I am the proposer again. I am gonna cheat (Evil smile)."
Mickey: Hurray!! I am the proposer!! I will propose block #5
 --> {Mickey}: Creating and Broadcasting proposal...
      --> {Mickey}: Received proposal from <Mickey>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Popeye}: Received proposal from <Mickey>. Validating...
            --> Invalid proposal.
            --> Previous block hash does not match with my current block's hash
            --> Proposal discarded. Moving on to next round
      --> {Donald}: Received proposal from <Mickey>. Validating...
            --> Invalid proposal.
            --> Previous block hash does not match with my current block's hash
            --> Proposal discarded. Moving on to next round
```

Figure 15: A client posting a BND transaction in Tendermint



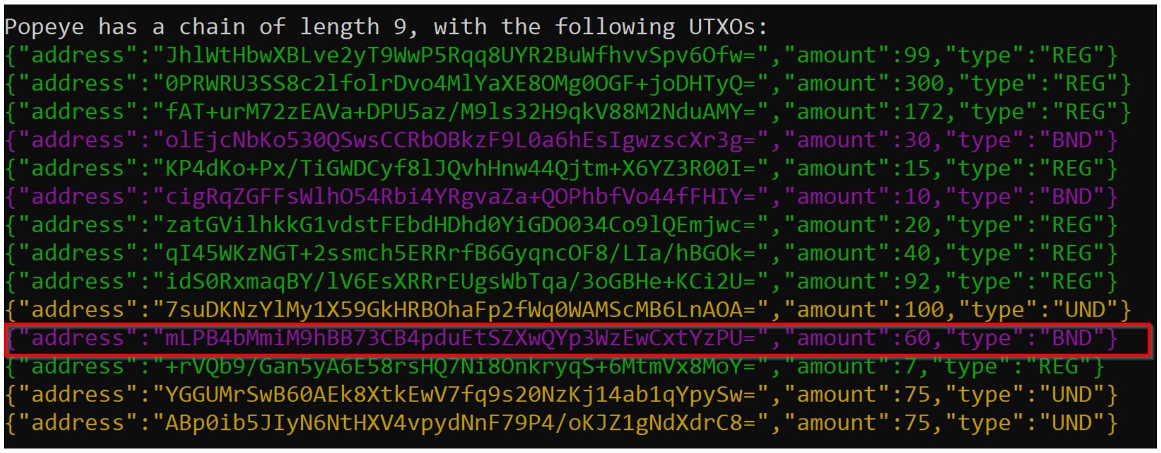
```
Donald: Hurray!! I am the proposer!! I will propose block #7
 --> {Donald}: Creating and Broadcasting proposal...
      --> {Donald}: Adding transaction of id 'e1d9c8daf6c9eae150b31c969a6c6c10dca39d9593f152e30310b5cdd3baa6f4' and type 'EVD' to current block
            --> Successful!
      --> {Donald}: Adding transaction of id '09d0eebc2aff9b4e16fe24e6d488370328f727b91c3aa54eec6fb494260dcc75' and type 'BND' to current block
            --> Successful!
      --> {Mickey}: Received proposal from <Donald>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Popeye}: Received proposal from <Donald>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Donald}: Received proposal from <Donald>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Mickey}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Received better proposal. Sync requested.
            --> Error in blockchain. Block not added.
 --> {Mickey}: Bonding Tom to my network
            --> Done. Set for new round
      --> {Popeye}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
 --> {Popeye}: Creating unbond tx to receive cheater Mickey's coins. My share is 75
 --> Broadcasting unbond tx with id '06078407fbd13cf3a650693f2794621aeaa80c2717c866ca3e9fee5b1f961f43'
 --> {Tom}: Unbond request received for Mickey. Validator will be unbonded in the next round.
 --> {Mickey}: Unbond request received for Mickey. Validator will be unbonded in the next round.
 --> {Popeye}: Unbond request received for Mickey. Validator will be unbonded in the next round.
 --> {Donald}: Unbond request received for Mickey. Validator will be unbonded in the next round.
 --> {Popeye}: Bonding Tom to my network
            --> Done. Set for new round
      --> {Donald}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
 --> {Donald}: Creating unbond tx to receive cheater Mickey's coins. My share is 75
 --> Broadcasting unbond tx with id '02b8b0251ab063f5974f76017656b1928a48e9188a5cd5df15835e9c493f1774'
 --> {Tom}: Unbond request received for Mickey. Validator will be unbonded in the next round.
 --> {Mickey}: Unbond request received for Mickey. Validator will be unbonded in the next round.
 --> {Popeye}: Unbond request received for Mickey. Validator will be unbonded in the next round.
 --> {Donald}: Unbond request received for Mickey. Validator will be unbonded in the next round.
 --> {Donald}: Bonding Tom to my network
            --> Done. Set for new round

 --> {Mickey}: Synced with the validator <Donald>
 --> {Mickey}: Currentblock length 7

 --> {Tom}: Synced with the validator <Mickey>
 --> {Tom}: Currentblock length 7
```

Figure 16: A validator adding the BND transaction created in Figure 15 in Tendermint



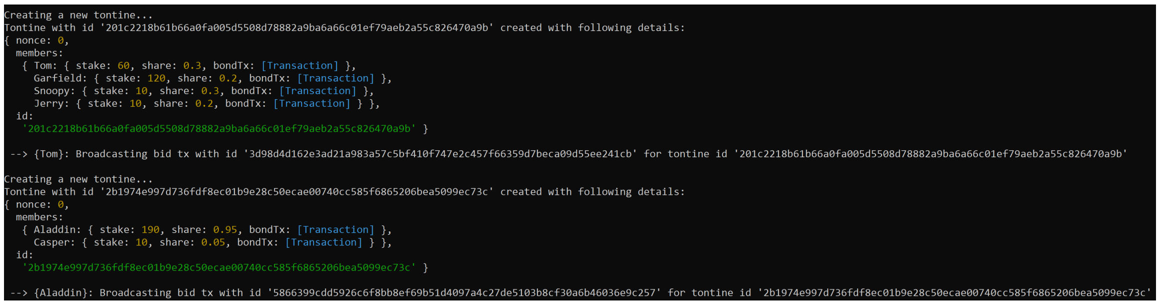
```
Tom: Hurray!! I am the proposer!! I will propose block #11
 --> {Tom}: Creating and Broadcasting proposal...
```

Figure 17: The client from Figure 15 as a proposer in Tendermint



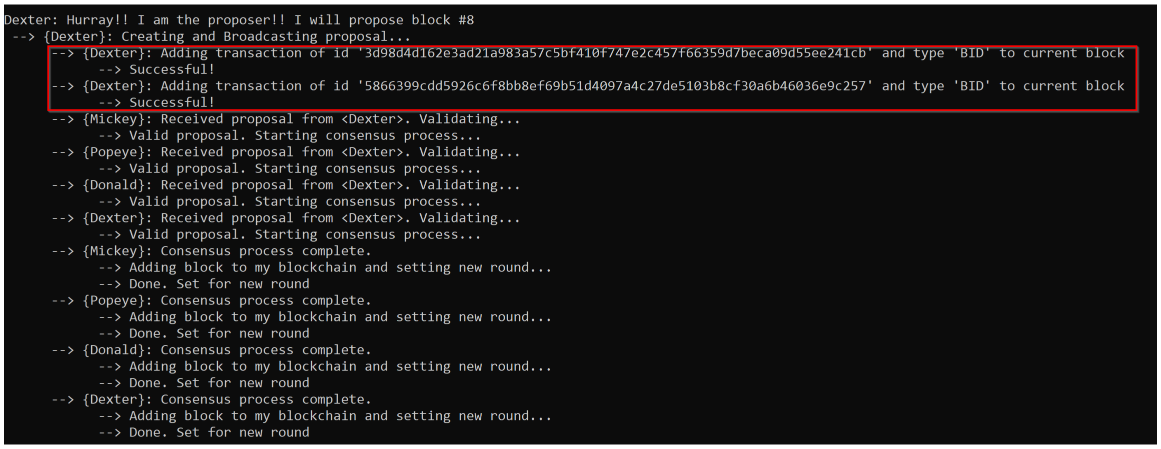
```
Popeye has a chain of length 9, with the following UTXOs:
{"address":"JhlWtHbwXBLve2yT9WwP5Rqq8UYR2BuWfhvvSpv6Ofw=","amount":99,"type":"REG"}
{"address":"0PRWRU3SS8c2lfolrDvo4MlYaXE8OMg0OGF+joDHTyQ=","amount":300,"type":"REG"}
{"address":"fAT+urM72zEAVa+DPU5az/M9ls32H9qkV88M2NduAMY=","amount":172,"type":"REG"}
{"address":"olEjcNbKo530QSwsCCRbOBkzF9L0a6hEsIgwzscXr3g=","amount":30,"type":"BND"}
{"address":"KP4dKo+Px/TiGWDCyf8lJQvhHnw44Qjtm+X6YZ3R00I=","amount":15,"type":"REG"}
{"address":"cigRqZGFFsWlhO54Rbi4YRgvaZa+QOPhbfVo44fFHIY=","amount":10,"type":"BND"}
{"address":"zatGVilhkkG1vdstFEbdHDhd0YiGDO034Co9lQEmjwc=","amount":20,"type":"REG"}
{"address":"qI45WKzNGT+2ssmch5ERRrfB6GyqncOF8/LIa/hBGOk=","amount":40,"type":"REG"}
{"address":"idS0RxmaqBY/lV6EsXRRrEUgsWbTqa/3oGBHe+KCi2U=","amount":92,"type":"REG"}
{"address":"7suDKNzYlMy1X59GkHRBOhaFp2fWq0WAMScMB6LnAOA=","amount":100,"type":"UND"}
{"address":"mLPB4bMmiM9hBB73CB4pduEtSZXwQYp3WzEwCxtYzPU=","amount":60,"type":"BND"}
{"address":"+rVQb9/Gan5yA6E58rsHQ7Ni8OnkryqS+6MtmVx8MoY=","amount":7,"type":"REG"}
{"address":"YGGUMrSwB60AEk8XtkEwV7fq9s20NzKj14ab1qYpySw=","amount":75,"type":"UND"}
{"address":"ABp0ib5JIyN6NtHXV4vpydNnF79P4/oKJZ1gNdXdrC8=","amount":75,"type":"UND"}
```

Figure 18: A validator's UTXOs after the BND transaction created in Figure 15 in Tendermint



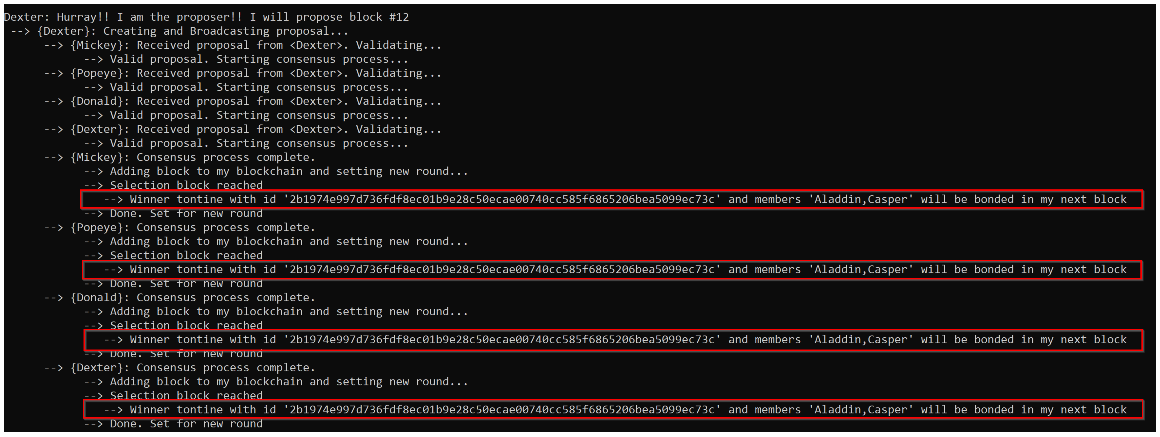
```
Creating a new tontine...
Tontine with id '201c2218b61b66a0fa005d5508d78882a9ba6a66c01ef79aeb2a55c826470a9b' created with following details:
{ nonce: 0,
  members:
   { Tom: { stake: 60, share: 0.3, bondTx: [Transaction] },
     Garfield: { stake: 120, share: 0.2, bondTx: [Transaction] },
     Snoopy: { stake: 10, share: 0.3, bondTx: [Transaction] },
     Jerry: { stake: 10, share: 0.2, bondTx: [Transaction] } },
  id:
   '201c2218b61b66a0fa005d5508d78882a9ba6a66c01ef79aeb2a55c826470a9b' }

 --> {Tom}: Broadcasting bid tx with id '3d98d4d162e3ad21a983a57c5bf410f747e2c457f66359d7beca09d55ee241cb' for tontine id '201c2218b61b66a0fa005d5508d78882a9ba6a66c01ef79aeb2a55c826470a9b'

Creating a new tontine...
Tontine with id '2b1974e997d736fdf8ec01b9e28c50ecae00740cc585f6865206bea5099ec73c' created with following details:
{ nonce: 0,
  members:
   { Aladdin: { stake: 190, share: 0.95, bondTx: [Transaction] },
     Casper: { stake: 10, share: 0.05, bondTx: [Transaction] } },
  id:
   '2b1974e997d736fdf8ec01b9e28c50ecae00740cc585f6865206bea5099ec73c' }

 --> {Aladdin}: Broadcasting bid tx with id '5866399cdd5926c6f8bb8ef69b51d4097a4c27de5103b8cf30a6b46036e9c257' for tontine id '2b1974e997d736fdf8ec01b9e28c50ecae00740cc585f6865206bea5099ec73c'
```

Figure 19: Creation of tontines and submission of BID transactions in TontineCoin



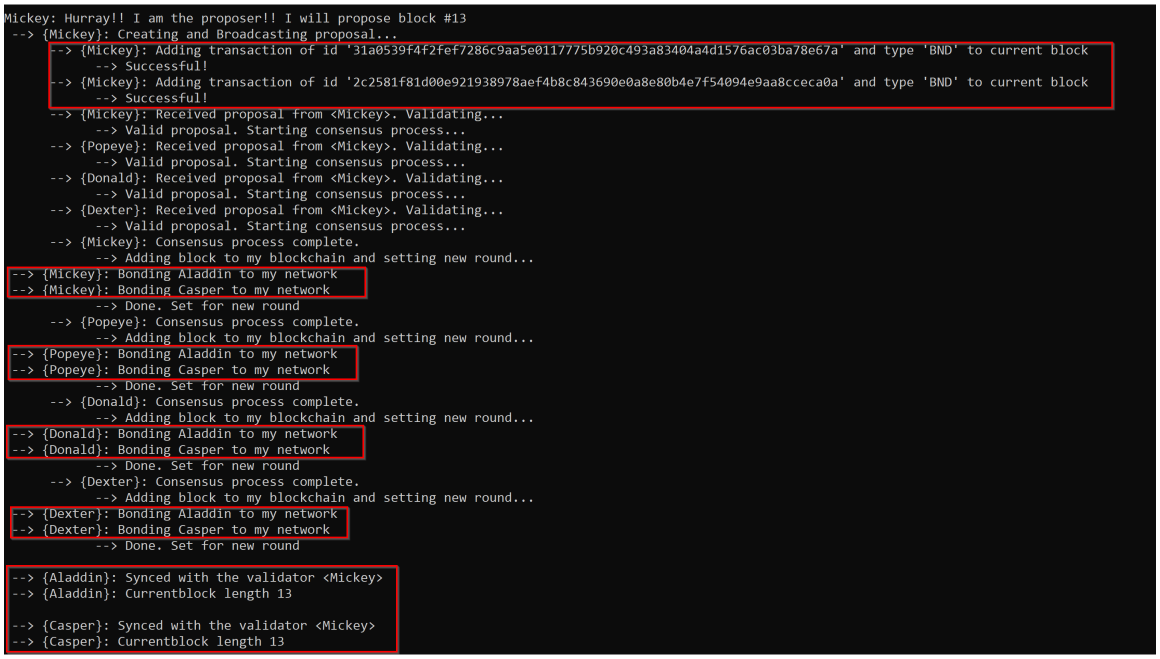
```
Dexter: Hurray!! I am the proposer!! I will propose block #8
 --> {Dexter}: Creating and Broadcasting proposal...
      --> {Dexter}: Adding transaction of id '3d98d4d162e3ad21a983a57c5bf410f747e2c457f66359d7beca09d55ee241cb' and type 'BID' to current block
            --> Successful!
      --> {Dexter}: Adding transaction of id '5866399cdd5926c6f8bb8ef69b51d4097a4c27de5103b8cf30a6b46036e9c257' and type 'BID' to current block
            --> Successful!
      --> {Mickey}: Received proposal from <Dexter>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Popeye}: Received proposal from <Dexter>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Donald}: Received proposal from <Dexter>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Dexter}: Received proposal from <Dexter>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Mickey}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round
      --> {Popeye}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round
      --> {Donald}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round
      --> {Dexter}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round
```

Figure 20: A validator adding the BID transactions created in Figure 19 in TontineCoin



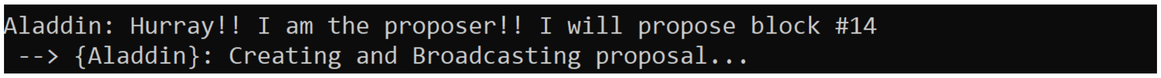
```
Dexter: Hurray!! I am the proposer!! I will propose block #12
 --> {Dexter}: Creating and Broadcasting proposal...
      --> {Mickey}: Received proposal from <Dexter>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Popeye}: Received proposal from <Dexter>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Donald}: Received proposal from <Dexter>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Dexter}: Received proposal from <Dexter>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Mickey}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Selection block reached
               --> Winner tontine with id '2b1974e997d736fdf8ec01b9e28c50ecae00740cc585f6865206bea5099ec73c' and members 'Aladdin,Casper' will be bonded in my next block
            --> Done. Set for new round
      --> {Popeye}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Selection block reached
               --> Winner tontine with id '2b1974e997d736fdf8ec01b9e28c50ecae00740cc585f6865206bea5099ec73c' and members 'Aladdin,Casper' will be bonded in my next block
            --> Done. Set for new round
      --> {Donald}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Selection block reached
               --> Winner tontine with id '2b1974e997d736fdf8ec01b9e28c50ecae00740cc585f6865206bea5099ec73c' and members 'Aladdin,Casper' will be bonded in my next block
            --> Done. Set for new round
      --> {Dexter}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Selection block reached
               --> Winner tontine with id '2b1974e997d736fdf8ec01b9e28c50ecae00740cc585f6865206bea5099ec73c' and members 'Aladdin,Casper' will be bonded in my next block
            --> Done. Set for new round
```

Figure 21: Selection of winner tontine from tontines created in Figure 19 in TontineCoin



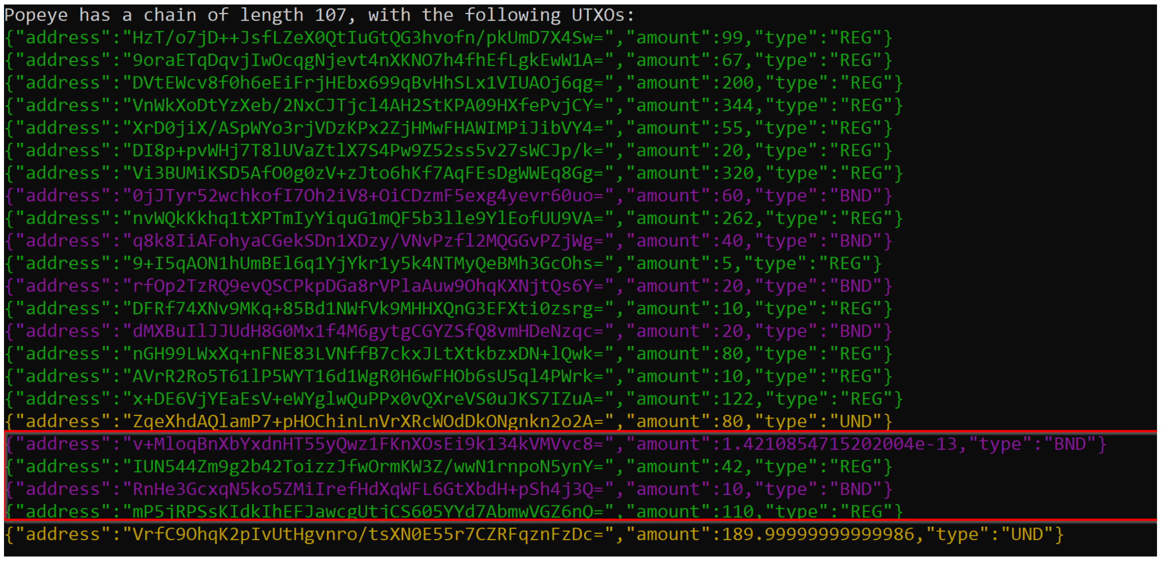
```
Mickey: Hurray!! I am the proposer!! I will propose block #13
 --> {Mickey}: Creating and Broadcasting proposal...
      --> {Mickey}: Adding transaction of id '31a0539f4f2fef7286c9aa5e0117775b920c493a83404a4d1576ac03ba78e67a' and type 'BND' to current block
            --> Successful!
      --> {Mickey}: Adding transaction of id '2c2581f81d00e921938978aef4b8c843690e0a8e80b4e7f54094e9aa8cceca0a' and type 'BND' to current block
            --> Successful!
      --> {Mickey}: Received proposal from <Mickey>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Popeye}: Received proposal from <Mickey>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Donald}: Received proposal from <Mickey>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Dexter}: Received proposal from <Mickey>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Mickey}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
 --> {Mickey}: Bonding Aladdin to my network
 --> {Mickey}: Bonding Casper to my network
            --> Done. Set for new round
      --> {Popeye}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
 --> {Popeye}: Bonding Aladdin to my network
 --> {Popeye}: Bonding Casper to my network
            --> Done. Set for new round
      --> {Donald}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
 --> {Donald}: Bonding Aladdin to my network
 --> {Donald}: Bonding Casper to my network
            --> Done. Set for new round
      --> {Dexter}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
 --> {Dexter}: Bonding Aladdin to my network
 --> {Dexter}: Bonding Casper to my network
            --> Done. Set for new round

 --> {Aladdin}: Synced with the validator <Mickey>
 --> {Aladdin}: Currentblock length 13

 --> {Casper}: Synced with the validator <Mickey>
 --> {Casper}: Currentblock length 13
```

Figure 22: A validator adding BND transactions of the tontine selected in Figure 21 in their proposal in TontineCoin



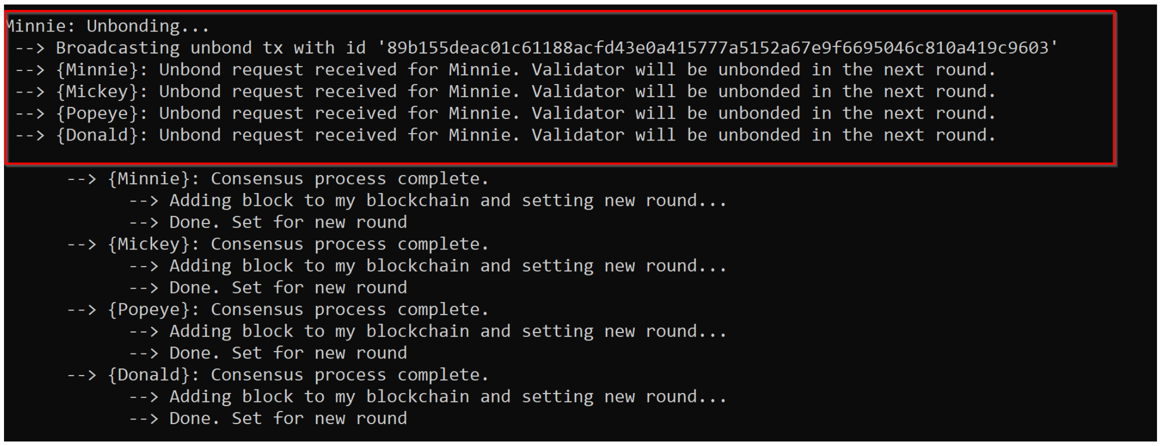
```
Aladdin: Hurray!! I am the proposer!! I will propose block #14
 --> {Aladdin}: Creating and Broadcasting proposal...
```

Figure 23: A validator from the tontine selected in Figure 21 proposing a block in TontineCoin



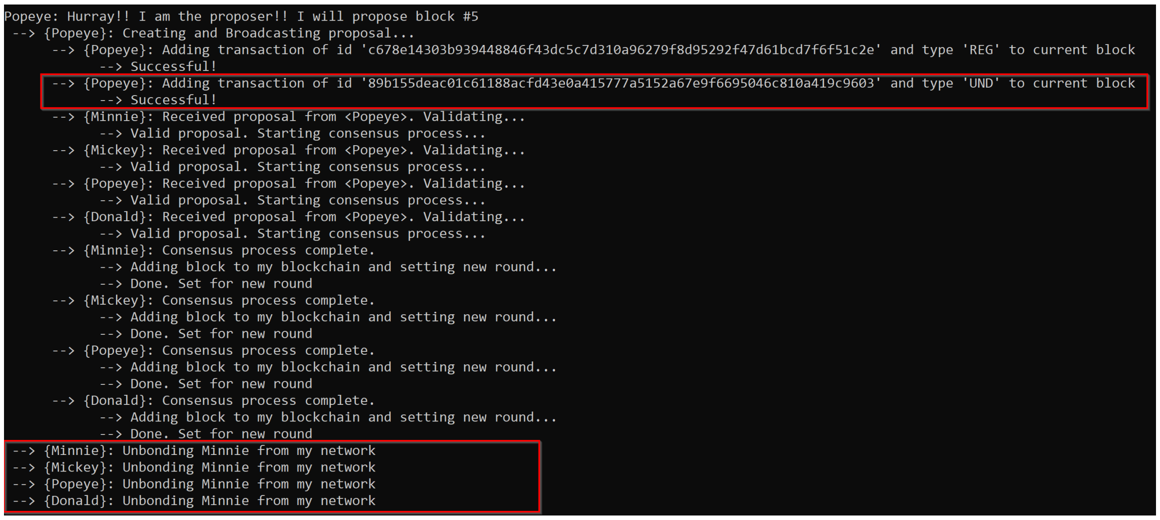
```
Popeye has a chain of length 107, with the following UTXOs:
{"address":"HzT/o7jD++JsfLZeX0QtIuGtQG3hvofn/pkUmD7X4Sw=","amount":99,"type":"REG"}
{"address":"9oraETqDqvjIwOcqgNjevt4nXKNO7h4fhEfLgkEwW1A=","amount":67,"type":"REG"}
{"address":"DVtEWcv8f0h6eEiFrjHEbx699qBvHhSLx1VIUAOj6qg=","amount":200,"type":"REG"}
{"address":"VnWkXoDtYzXeb/2NxCJTjcl4AH2StKPA09HXfePvjCY=","amount":344,"type":"REG"}
{"address":"XrD0jiX/ASpWYo3rjVDzKPx2ZjHMwFHAWIMPiJibVY4=","amount":55,"type":"REG"}
{"address":"DI8p+pvWHj7T8lUVaZtlX7S4Pw9Z52ss5v27sWCJp/k=","amount":20,"type":"REG"}
{"address":"Vi3BUMiKSD5AfO0g0zV+zJto6hKf7AqFEsDgWWEq8Gg=","amount":320,"type":"REG"}
{"address":"0jJTyr52wchkofI7Oh2iV8+OiCDzmF5exg4yevr60uo=","amount":60,"type":"BND"}
{"address":"nvWQkKkhq1tXPTmIyYiquG1mQF5b3lle9YlEofUU9VA=","amount":262,"type":"REG"}
{"address":"q8k8IiAFohyaCGekSDn1XDzy/VNvPzfl2MQGGvPZjWg=","amount":40,"type":"BND"}
{"address":"9+I5qAON1hUmBEl6q1YjYkr1y5k4NTMyQeBMh3GcOhs=","amount":5,"type":"REG"}
{"address":"rfOp2TzRQ9evQSCPkpDGa8rVPlaAuw9OhqKXNjtQs6Y=","amount":20,"type":"BND"}
{"address":"DFRf74XNv9MKq+85Bd1NWfVk9MHHXQnG3EFXti0zsrg=","amount":10,"type":"REG"}
{"address":"dMXBuIlJJUdH8G0Mx1f4M6gytgCGYZSfQ8vmHDeNzqc=","amount":20,"type":"BND"}
{"address":"nGH99LWxXq+nFNE83LVNffB7ckxJLtXtkbzxDN+lQwk=","amount":80,"type":"REG"}
{"address":"AVrR2Ro5T61lP5WYT16d1WgR0H6wFHOb6sU5ql4PWrk=","amount":10,"type":"REG"}
{"address":"x+DE6VjYEaEsV+eWYglwQuPPx0vQXreVS0uJKS7IZuA=","amount":122,"type":"REG"}
{"address":"ZqeXhdAQlamP7+pHOChinLnVrXRcWOdDkONgkn2o2A=","amount":80,"type":"UND"}
{"address":"v+MloqBnXbYxdnHT55yQwz1FKnXOsEi9k134kVMVvc8=","amount":1.4210854715202004e-13,"type":"BND"}
{"address":"IUN544Zm9g2b42ToizzJfwOrmKW3Z/wwN1rnpoN5ynY=","amount":42,"type":"REG"}
{"address":"RnHe3GcxqN5ko5ZMiIrefHdXqWFL6GtXbdH+pSh4j3Q=","amount":10,"type":"BND"}
{"address":"mP5jRPSsKIdkIhEFJawcgUtjCS605YYd7AbmwVGZ6nQ=","amount":110,"type":"REG"}
{"address":"VrfC9OhqK2pIvUtHgvnro/tsXN0E55r7CZRFqznFzDc=","amount":189.99999999999986,"type":"UND"}
```

Figure 24: A validator's UTXOs after the BND transaction created in Figure 21 in TontineCoin



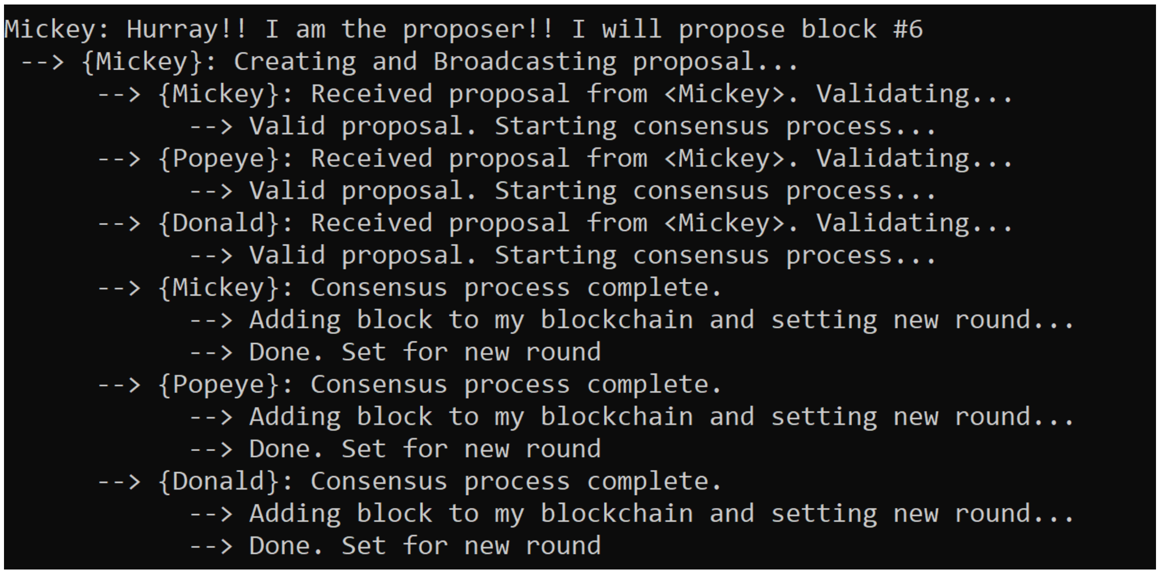
```
innie: Unbonding...
 --> Broadcasting unbond tx with id '89b155deac01c61188acfd43e0a415777a5152a67e9f6695046c810a419c9603'
 --> {Minnie}: Unbond request received for Minnie. Validator will be unbonded in the next round.
 --> {Mickey}: Unbond request received for Minnie. Validator will be unbonded in the next round.
 --> {Popeye}: Unbond request received for Minnie. Validator will be unbonded in the next round.
 --> {Donald}: Unbond request received for Minnie. Validator will be unbonded in the next round.

      --> {Minnie}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round
      --> {Mickey}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round
      --> {Popeye}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round
      --> {Donald}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round
```

Figure 25: A validator posting a UND transaction in Tendermint



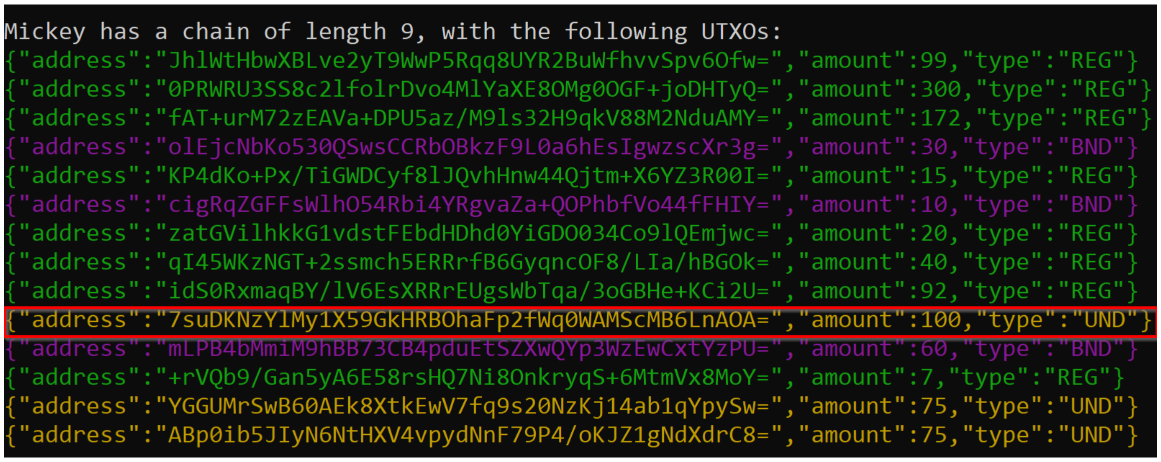
```
Popeye: Hurray!! I am the proposer!! I will propose block #5
 --> {Popeye}: Creating and Broadcasting proposal...
      --> {Popeye}: Adding transaction of id 'c678e14303b939448846f43dc5c7d310a96279f8d95292f47d61bcd7f6f51c2e' and type 'REG' to current block
            --> Successful!
      --> {Popeye}: Adding transaction of id '89b155deac01c61188acfd43e0a415777a5152a67e9f6695046c810a419c9603' and type 'UND' to current block
            --> Successful!
      --> {Minnie}: Received proposal from <Popeye>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Mickey}: Received proposal from <Popeye>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Popeye}: Received proposal from <Popeye>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Donald}: Received proposal from <Popeye>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Minnie}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round
      --> {Mickey}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round
      --> {Popeye}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round
      --> {Donald}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round
 --> {Minnie}: Unbonding Minnie from my network
 --> {Mickey}: Unbonding Minnie from my network
 --> {Popeye}: Unbonding Minnie from my network
 --> {Donald}: Unbonding Minnie from my network
```

Figure 26: A validator unbonding in Tendermint after their UND transaction is committed



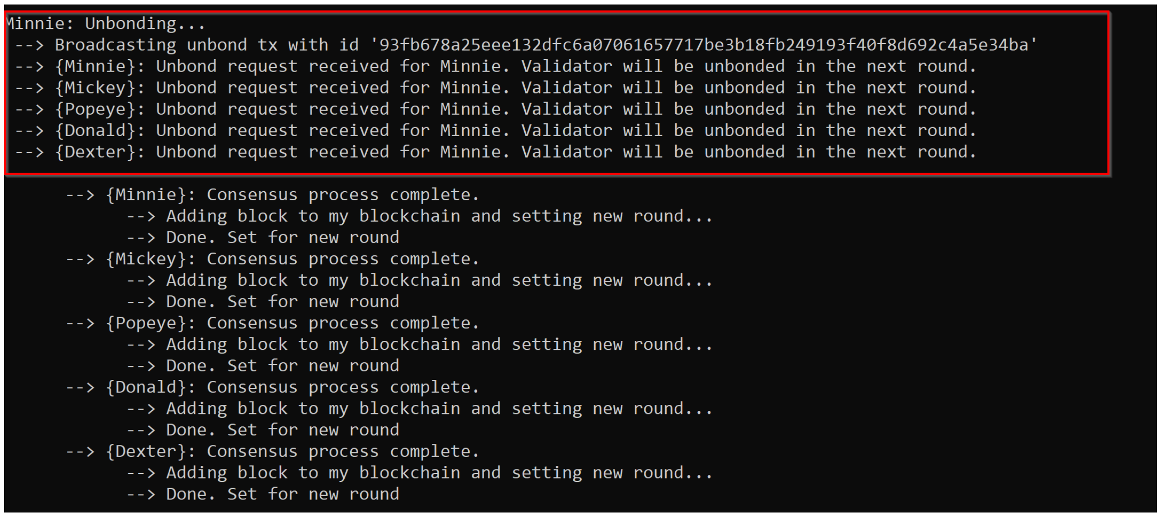
```
Mickey: Hurray!! I am the proposer!! I will propose block #6
 --> {Mickey}: Creating and Broadcasting proposal...
      --> {Mickey}: Received proposal from <Mickey>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Popeye}: Received proposal from <Mickey>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Donald}: Received proposal from <Mickey>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Mickey}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round
      --> {Popeye}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round
      --> {Donald}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round
```

Figure 27: A round of block creation after the validator from Figure 25 is unbonded in Tendermint



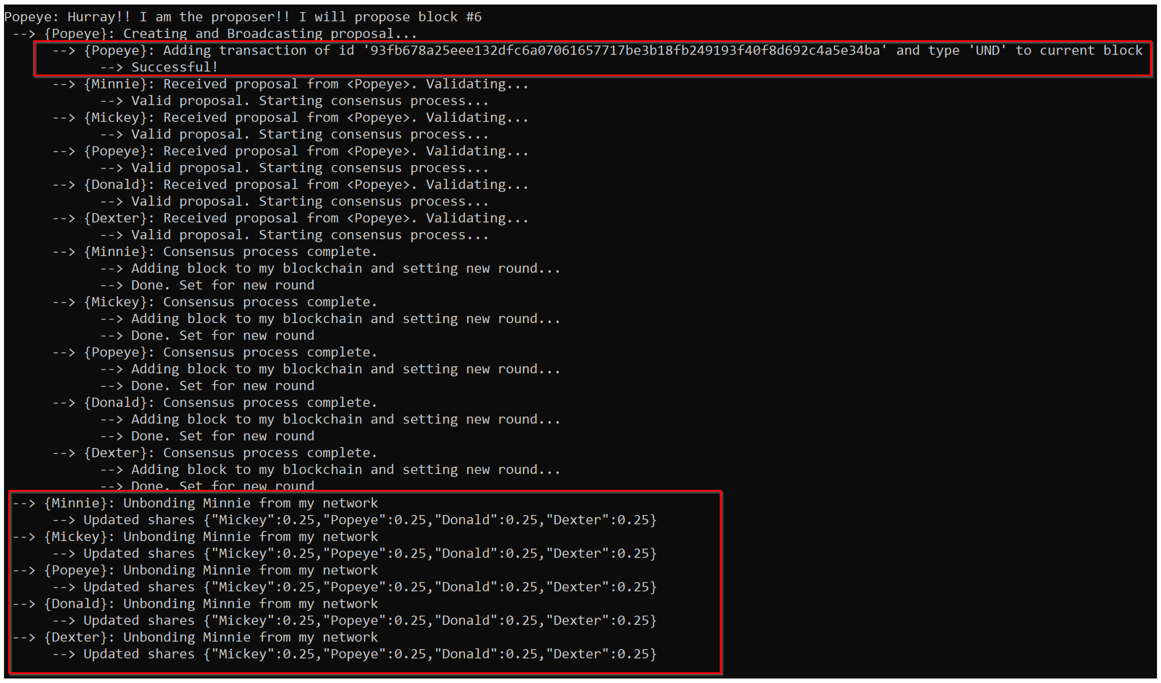
```
Mickey has a chain of length 9, with the following UTXOs:
{"address":"JhlWtHbwXBLve2yT9WwP5Rqq8UYR2BuWfhvvSpv6Ofw=","amount":99,"type":"REG"}
{"address":"0PRWRU3SS8c2lfolrDvo4MlYaXE8OMg0OGF+joDHTyQ=","amount":300,"type":"REG"}
{"address":"fAT+urM72zEAVa+DPU5az/M9ls32H9qkV88M2NduAMY=","amount":172,"type":"REG"}
{"address":"olEjcNbKo530QSwsCCRbOBkzF9L0a6hEsIgwzscXr3g=","amount":30,"type":"BND"}
{"address":"KP4dKo+Px/TiGWDCyf8lJQvhHnw44Qjtm+X6YZ3R00I=","amount":15,"type":"REG"}
{"address":"cigRqZGFFsWlhO54Rbi4YRgvaZa+QOPhbfVo44fFHIY=","amount":10,"type":"BND"}
{"address":"zatGVilhkkG1vdstFEbdHDhd0YiGDO034Co9lQEmjwc=","amount":20,"type":"REG"}
{"address":"qI45WKzNGT+2ssmch5ERRrfB6GyqncOF8/LIa/hBGOk=","amount":40,"type":"REG"}
{"address":"idS0RxmaqBY/lV6EsXRRrEUgsWbTqa/3oGBHe+KCi2U=","amount":92,"type":"REG"}
{"address":"7suDKNzYlMy1X59GkHRBOhaFp2fWq0WAMScMB6LnAOA=","amount":100,"type":"UND"}
{"address":"mLPB4bMmiM9hBB73CB4pduEtSZXwQYp3WzEwCxtYzPU=","amount":60,"type":"BND"}
{"address":"+rVQb9/Gan5yA6E58rsHQ7Ni8OnkryqS+6MtmVx8MoY=","amount":7,"type":"REG"}
{"address":"YGGUMrSwB60AEk8XtkEwV7fq9s20NzKj14ab1qYpySw=","amount":75,"type":"UND"}
{"address":"ABp0ib5JIyN6NtHXV4vpydNnF79P4/oKJZ1gNdXdrC8=","amount":75,"type":"UND"}
```

Figure 28: A validator's UTXOs displaying the UTXO of unbonded validator from Figure 25 in Tendermint



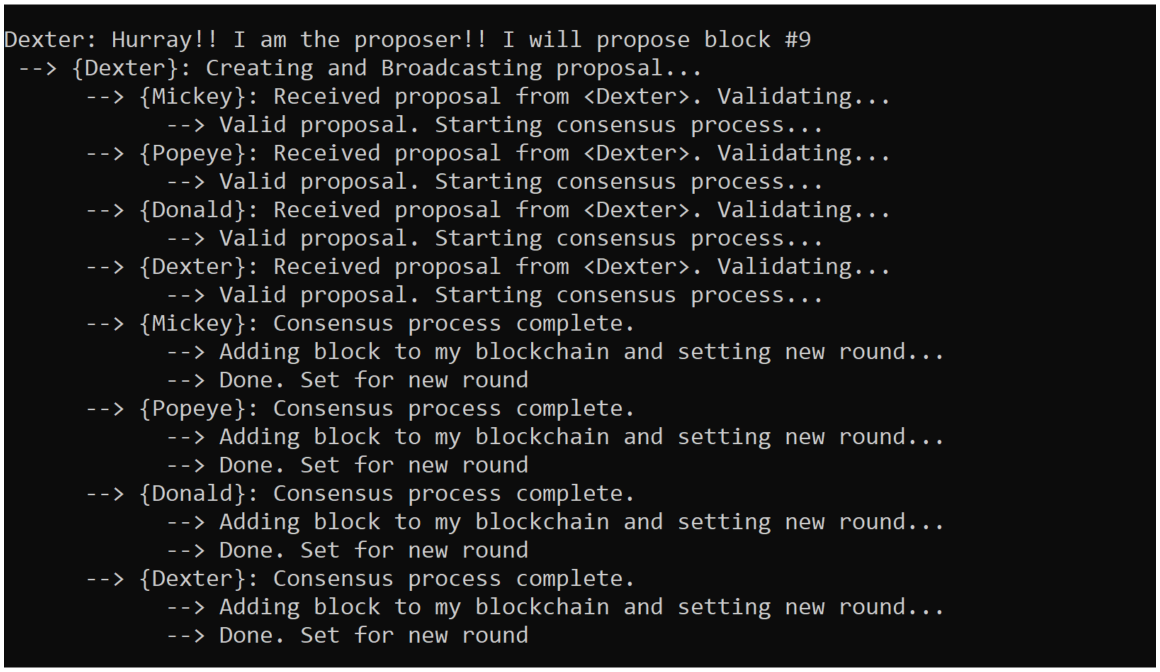
```
Minnie: Unbonding...
 --> Broadcasting unbond tx with id '93fb678a25eee132dfc6a07061657717be3b18fb249193f40f8d692c4a5e34ba'
 --> {Minnie}: Unbond request received for Minnie. Validator will be unbonded in the next round.
 --> {Mickey}: Unbond request received for Minnie. Validator will be unbonded in the next round.
 --> {Popeye}: Unbond request received for Minnie. Validator will be unbonded in the next round.
 --> {Donald}: Unbond request received for Minnie. Validator will be unbonded in the next round.
 --> {Dexter}: Unbond request received for Minnie. Validator will be unbonded in the next round.

      --> {Minnie}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round
      --> {Mickey}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round
      --> {Popeye}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round
      --> {Donald}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round
      --> {Dexter}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round
```

Figure 29: A validator posting a UND transaction in TontineCoin



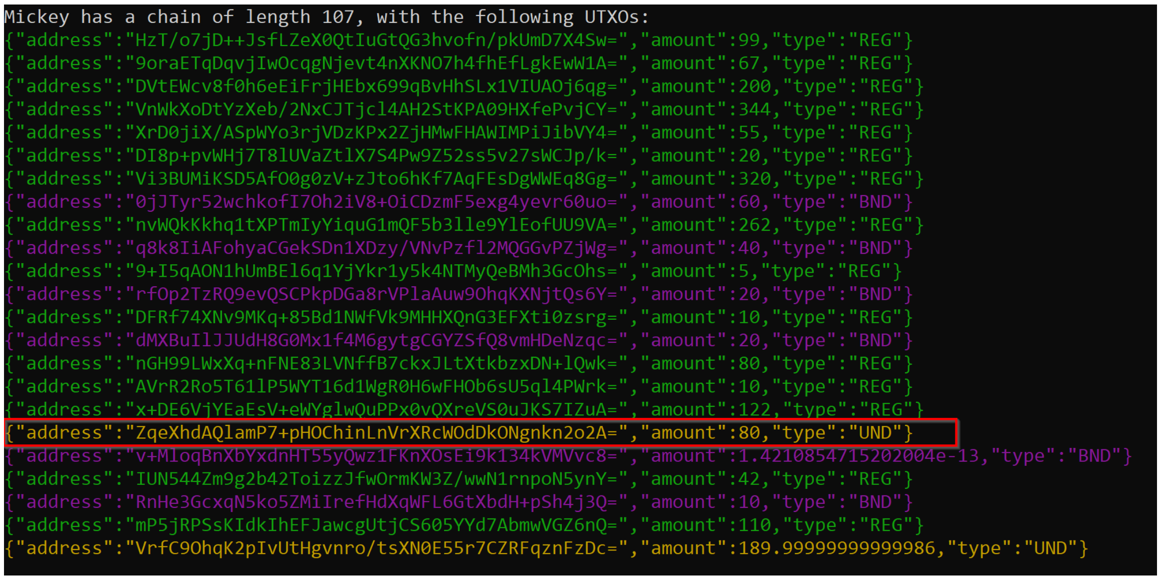
```
Popeye: Hurray!! I am the proposer!! I will propose block #6
 --> {Popeye}: Creating and Broadcasting proposal...
      --> {Popeye}: Adding transaction of id '93fb678a25eee132dfc6a07061657717be3b18fb249193f40f8d692c4a5e34ba' and type 'UND' to current block
            --> Successful!
      --> {Minnie}: Received proposal from <Popeye>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Mickey}: Received proposal from <Popeye>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Popeye}: Received proposal from <Popeye>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Donald}: Received proposal from <Popeye>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Dexter}: Received proposal from <Popeye>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Minnie}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round
      --> {Mickey}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round
      --> {Popeye}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round
      --> {Donald}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round
      --> {Dexter}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round
--> {Minnie}: Unbonding Minnie from my network
      --> Updated shares {"Mickey":0.25,"Popeye":0.25,"Donald":0.25,"Dexter":0.25}
--> {Mickey}: Unbonding Minnie from my network
      --> Updated shares {"Mickey":0.25,"Popeye":0.25,"Donald":0.25,"Dexter":0.25}
--> {Popeye}: Unbonding Minnie from my network
      --> Updated shares {"Mickey":0.25,"Popeye":0.25,"Donald":0.25,"Dexter":0.25}
--> {Donald}: Unbonding Minnie from my network
      --> Updated shares {"Mickey":0.25,"Popeye":0.25,"Donald":0.25,"Dexter":0.25}
--> {Dexter}: Unbonding Minnie from my network
      --> Updated shares {"Mickey":0.25,"Popeye":0.25,"Donald":0.25,"Dexter":0.25}
```

Figure 30: A validator unbonding in Tontinecoin after their UND transaction is committed



```
Dexter: Hurray!! I am the proposer!! I will propose block #9
 --> {Dexter}: Creating and Broadcasting proposal...
      --> {Mickey}: Received proposal from <Dexter>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Popeye}: Received proposal from <Dexter>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Donald}: Received proposal from <Dexter>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Dexter}: Received proposal from <Dexter>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Mickey}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round
      --> {Popeye}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round
      --> {Donald}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round
      --> {Dexter}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round
```

Figure 31: A round of block creation after the validator from Figure 29 is unbonded in TontineCoin



```
Mickey has a chain of length 107, with the following UTXOs:
{"address":"HzT/o7jD++JsfLZeX0QtIuGtQG3hvofn/pkUmD7X4Sw=","amount":99,"type":"REG"}
{"address":"9oraETqDqvjIwOcqgNjevt4nXKNO7h4fhEfLgkEwW1A=","amount":67,"type":"REG"}
{"address":"DVtEWcv8f0h6eEiFrjHEbx699qBvHhSLx1VIUAOj6qg=","amount":200,"type":"REG"}
{"address":"VnWkXoDtYzXeb/2NxCJTjcl4AH2StKPA09HXfePvjCY=","amount":344,"type":"REG"}
{"address":"XrD0jiX/ASpWYo3rjVDzKPx2ZjHMwFHAWIMPiJibVY4=","amount":55,"type":"REG"}
{"address":"DI8p+pvWHj7T8lUVaZtlX7S4Pw9Z52ss5v27sWCJp/k=","amount":20,"type":"REG"}
{"address":"Vi3BUMiKSD5AfO0g0zV+zJto6hKf7AqFEsDgWWEq8Gg=","amount":320,"type":"REG"}
{"address":"0jJTyr52wchkofI7Oh2iV8+OiCDzmF5exg4yevr60uo=","amount":60,"type":"BND"}
{"address":"nvWQkKkhq1tXPTmIyYiquG1mQF5b3lle9YlEofUU9VA=","amount":262,"type":"REG"}
{"address":"q8k8IiAFohyaCGekSDn1XDzy/VNvPzfl2MQGGvPZjWg=","amount":40,"type":"BND"}
{"address":"9+I5qAON1hUmBEl6q1YjYkr1y5k4NTMyQeBMh3GcOhs=","amount":5,"type":"REG"}
{"address":"rfOp2TzRQ9evQSCPkpDGa8rVPlaAuw9OhqKXNjtQs6Y=","amount":20,"type":"BND"}
{"address":"DFRf74XNv9MKq+85Bd1NWfVk9MHHXQnG3EFXti0zsrg=","amount":10,"type":"REG"}
{"address":"dMXBuIlJJUdH8G0Mx1f4M6gytgCGYZSfQ8vmHDeNzqc=","amount":20,"type":"BND"}
{"address":"nGH99LWxXq+nFNE83LVNffB7ckxJLtXtkbzxDN+lQwk=","amount":80,"type":"REG"}
{"address":"AVrR2Ro5T61lP5WYT16d1WgR0H6wFHOb6sU5ql4PWrk=","amount":10,"type":"REG"}
{"address":"x+DE6VjYEaEsV+eWYglwQuPPx0vQXreVS0uJKS7IZuA=","amount":122,"type":"REG"}
{"address":"ZqeXhdAQlamP7+pHOChinLnVrXRcWOdDkONgnkn2o2A=","amount":80,"type":"UND"}
{"address":"v+MloqBnXbYxdnHT55yQwz1FKnXOsEi9k134kVMVvc8=","amount":1.4210854715202004e-13,"type":"BND"}
{"address":"IUN544Zm9g2b42ToizzJfwOrmKW3Z/wwN1rnpoN5ynY=","amount":42,"type":"REG"}
{"address":"RnHe3GcxqN5ko5ZMiIrefHdXqWFL6GtXbdH+pSh4j3Q=","amount":10,"type":"BND"}
{"address":"mP5jRPSsKIdkIhEFJawcgUtjCS605YYd7AbmwVGZ6nQ=","amount":110,"type":"REG"}
{"address":"VrfC9OhqK2pIvUtHgvnro/tsXN0E55r7CZRFqznFzDc=","amount":189.99999999999986,"type":"UND"}
```

Figure 32: A validator's UTXOs displaying the UTXO of unbonded validator from Figure 29 in TontineCoin

CHAPTER 6

Handling of nothing-at-stake in TontineCoin and Tendermint

This chapter shows and discusses how a nothing-at-stake attack is handled in the TontineCoin model compared to the Tendermint model. A cheating scenario is simulated in both the systems in the same environment and network created in chapter 5. This scenario depicts a validator proposing a conflicting block and being ejected from the network.

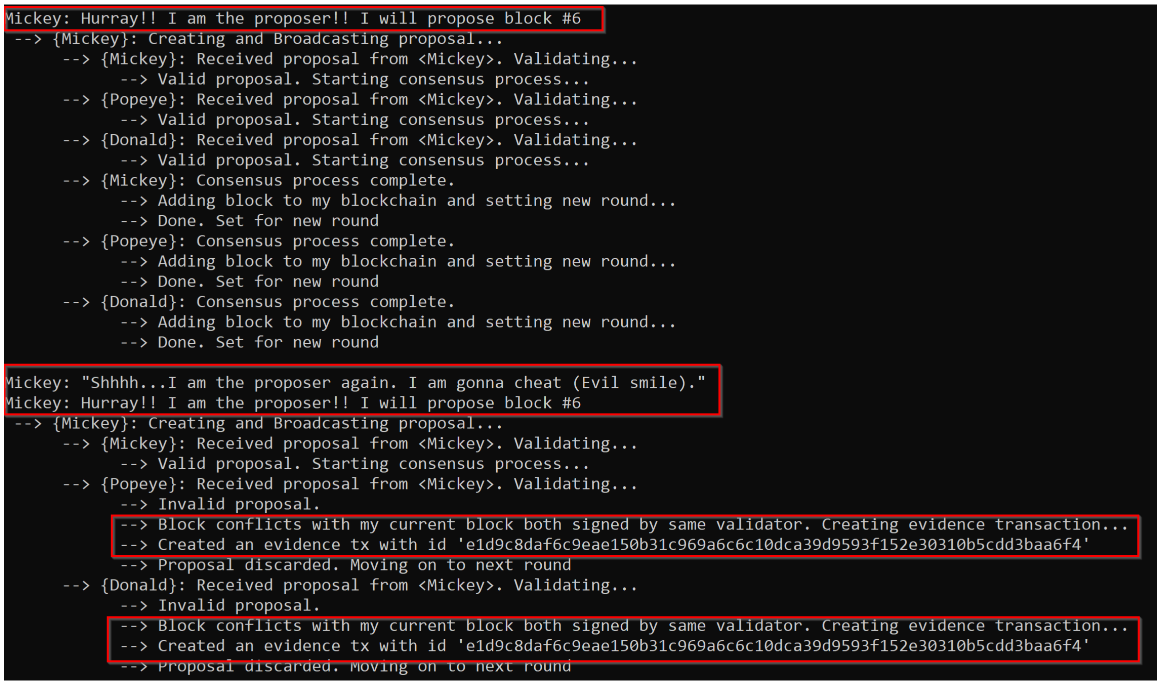
A validator in both the systems cheats when their faulty attribute is set, and they propose two blocks of the same height and with the same signature consecutively. In Tendermint, the cheating validator's stake is equally divided among the rest of validators. But in TontineCoin, it is divided among those validators who belong to the cheating validator's tontine. It is divided based on their share in their tontine.

6.1 Cheating scenario in Tendermint

Figure 33 shows Mickey proposing a conflicting block, and then Popeye and Donald creating the EVD transactions. Figure 34 shows Donald adding his EVD transaction in his proposal, and then Popeye and Donald creating the UND transactions. Figure 35 shows Tom adding them in his proposal, and then Mickey being unbonded from the network. Figure 36 shows Popeye's UTXOs consisting of the seized UTXOs. Figure 37 shows the updated wallets of validators after seizing the coins of Mickey.

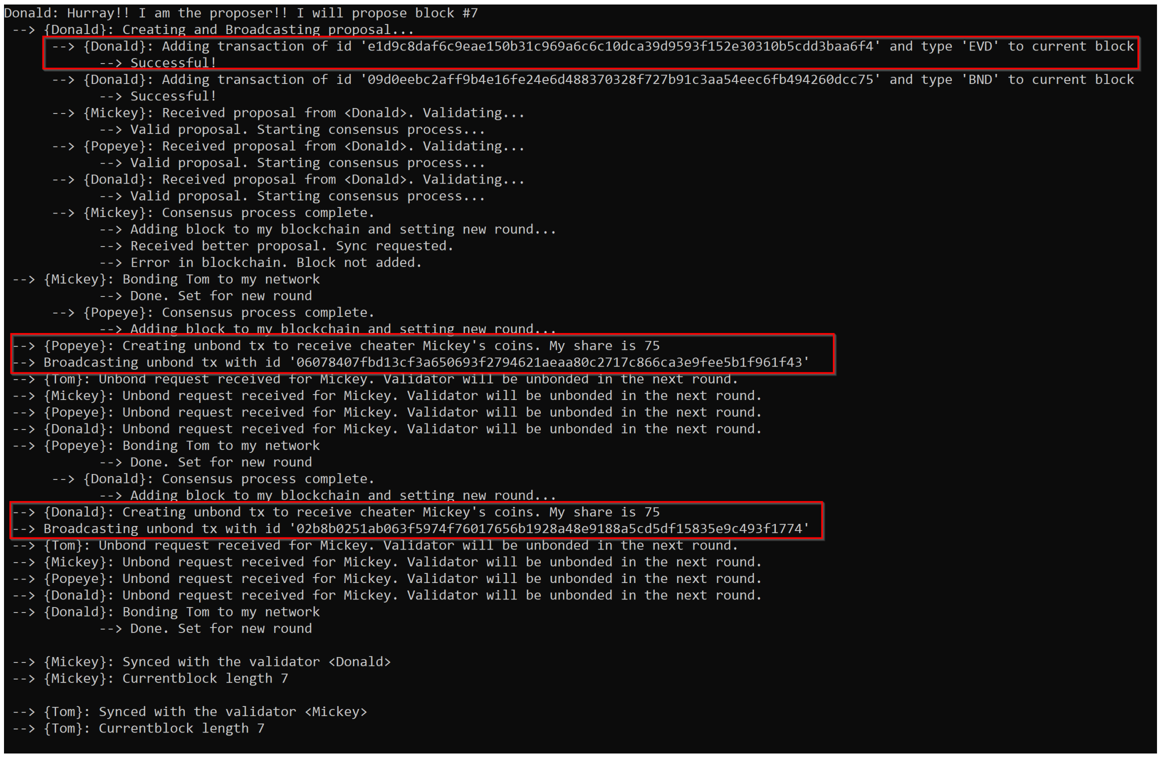
6.2 Cheating scenario in TontineCoin

Figure 38 shows Aladdin proposing a conflicting block and then Casper creating and broadcasting a EVD transaction. Figure 39 shows Popeye adding it in his proposal and then Casper creating a UND transaction. Figure 40 shows Donald adding it in his proposal, and then Aladdin being unbonded from the network. Figure 41 shows Popeye's UTXOs consisting of the seized UTXOs. Figure 42 shows the updated wallets of validators after seizing the coins of Aladdin.



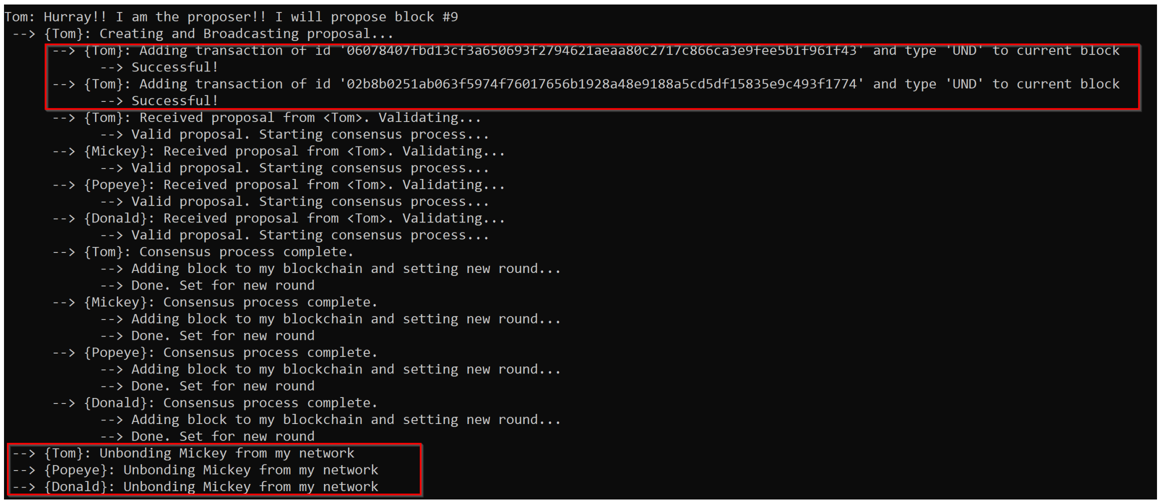
```
Mickey: Hurray!! I am the proposer!! I will propose block #6
 --> {Mickey}: Creating and Broadcasting proposal...
      --> {Mickey}: Received proposal from <Mickey>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Popeye}: Received proposal from <Mickey>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Donald}: Received proposal from <Mickey>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Mickey}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round
      --> {Popeye}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round
      --> {Donald}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round

Mickey: "Shhhh...I am the proposer again. I am gonna cheat (Evil smile)."
Mickey: Hurray!! I am the proposer!! I will propose block #6
 --> {Mickey}: Creating and Broadcasting proposal...
      --> {Mickey}: Received proposal from <Mickey>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Popeye}: Received proposal from <Mickey>. Validating...
            --> Invalid proposal.
            --> Block conflicts with my current block both signed by same validator. Creating evidence transaction...
            --> Created an evidence tx with id 'e1d9c8daf6c9eae150b31c969a6c6c10dca39d9593f152e30310b5cdd3baa6f4'
            --> Proposal discarded. Moving on to next round
      --> {Donald}: Received proposal from <Mickey>. Validating...
            --> Invalid proposal.
            --> Block conflicts with my current block both signed by same validator. Creating evidence transaction...
            --> Created an evidence tx with id 'e1d9c8daf6c9eae150b31c969a6c6c10dca39d9593f152e30310b5cdd3baa6f4'
            --> Proposal discarded. Moving on to next round
```

Figure 33: A cheater getting caught in Tendermint



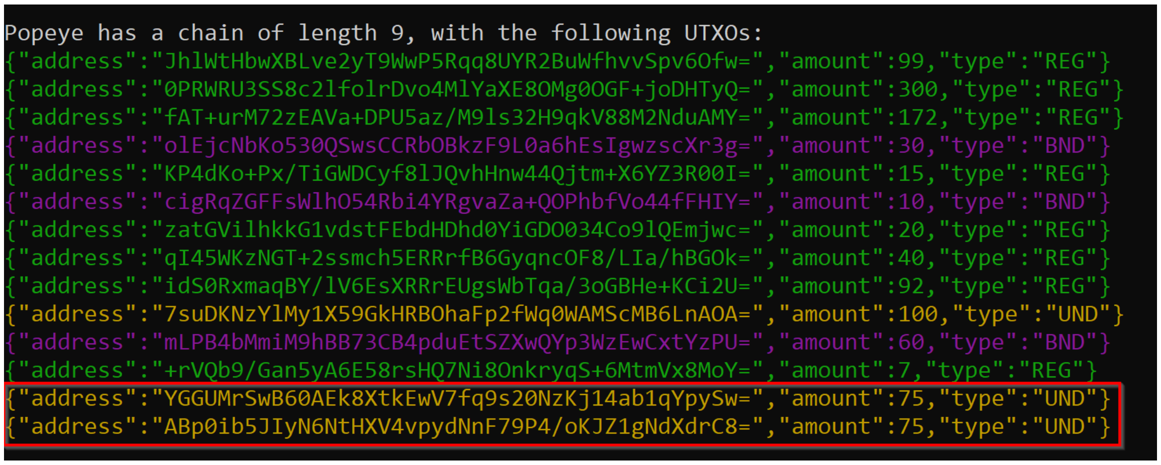
```
Donald: Hurray!! I am the proposer!! I will propose block #7
 --> {Donald}: Creating and Broadcasting proposal...
      --> {Donald}: Adding transaction of id 'e1d9c8daf6c9eae150b31c969a6c6c10dca39d9593f152e30310b5cdd3baa6f4' and type 'EVD' to current block
            --> Successful!
      --> {Donald}: Adding transaction of id '09d0eebc2aff9b4e16fe24e6d488370328f727b91c3aa54eec6fb494260dcc75' and type 'BND' to current block
            --> Successful!
      --> {Mickey}: Received proposal from <Donald>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Popeye}: Received proposal from <Donald>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Donald}: Received proposal from <Donald>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Mickey}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Received better proposal. Sync requested.
            --> Error in blockchain. Block not added.
 --> {Mickey}: Bonding Tom to my network
            --> Done. Set for new round
      --> {Popeye}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
 --> {Popeye}: Creating unbond tx to receive cheater Mickey's coins. My share is 75
 --> Broadcasting unbond tx with id '06078407fbd13cf3a650693f2794621aeaa80c2717c866ca3e9fee5b1f961f43'
 --> {Tom}: Unbond request received for Mickey. Validator will be unbonded in the next round.
 --> {Mickey}: Unbond request received for Mickey. Validator will be unbonded in the next round.
 --> {Popeye}: Unbond request received for Mickey. Validator will be unbonded in the next round.
 --> {Donald}: Unbond request received for Mickey. Validator will be unbonded in the next round.
 --> {Popeye}: Bonding Tom to my network
            --> Done. Set for new round
      --> {Donald}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
 --> {Donald}: Creating unbond tx to receive cheater Mickey's coins. My share is 75
 --> Broadcasting unbond tx with id '02b8b0251ab063f5974f76017656b1928a48e9188a5cd5df15835e9c493f1774'
 --> {Tom}: Unbond request received for Mickey. Validator will be unbonded in the next round.
 --> {Mickey}: Unbond request received for Mickey. Validator will be unbonded in the next round.
 --> {Popeye}: Unbond request received for Mickey. Validator will be unbonded in the next round.
 --> {Donald}: Unbond request received for Mickey. Validator will be unbonded in the next round.
 --> {Donald}: Bonding Tom to my network
            --> Done. Set for new round

 --> {Mickey}: Synced with the validator <Donald>
 --> {Mickey}: Currentblock length 7

 --> {Tom}: Synced with the validator <Mickey>
 --> {Tom}: Currentblock length 7
```

Figure 34: Inclusion of EVD transaction created in Figure 33 and UND transactions to seize cheater's stake in Tendermint



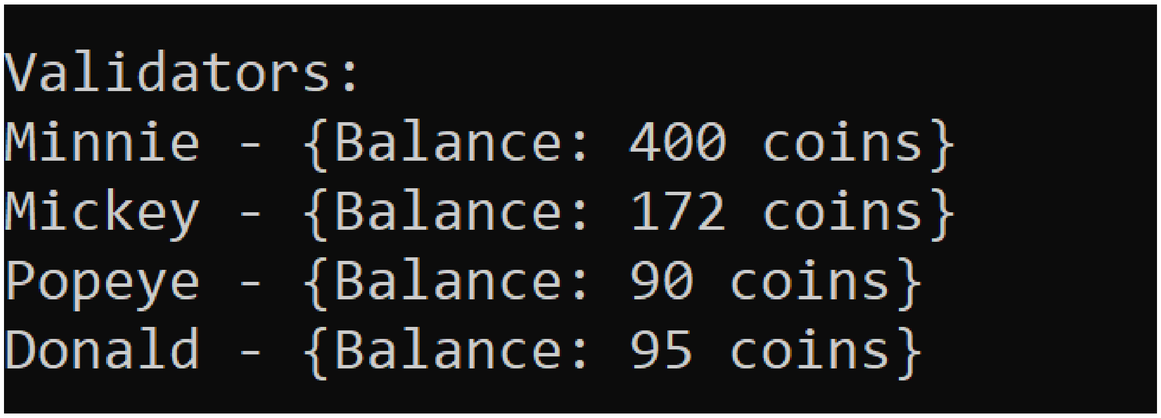
```
Tom: Hurray!! I am the proposer!! I will propose block #9
 --> {Tom}: Creating and Broadcasting proposal...
     --> {Tom}: Adding transaction of id '06078407fbd13cf3a650693f2794621aeaa80c2717c866ca3e9fee5b1f961f43' and type 'UND' to current block
           --> Successful!
     --> {Tom}: Adding transaction of id '02b8b0251ab063f5974f76017656b1928a48e9188a5cd5df15835e9c493f1774' and type 'UND' to current block
           --> Successful!
     --> {Tom}: Received proposal from <Tom>. Validating...
           --> Valid proposal. Starting consensus process...
     --> {Mickey}: Received proposal from <Tom>. Validating...
           --> Valid proposal. Starting consensus process...
     --> {Popeye}: Received proposal from <Tom>. Validating...
           --> Valid proposal. Starting consensus process...
     --> {Donald}: Received proposal from <Tom>. Validating...
           --> Valid proposal. Starting consensus process...
     --> {Tom}: Consensus process complete.
           --> Adding block to my blockchain and setting new round...
           --> Done. Set for new round
     --> {Mickey}: Consensus process complete.
           --> Adding block to my blockchain and setting new round...
           --> Done. Set for new round
     --> {Popeye}: Consensus process complete.
           --> Adding block to my blockchain and setting new round...
           --> Done. Set for new round
     --> {Donald}: Consensus process complete.
           --> Adding block to my blockchain and setting new round...
           --> Done. Set for new round
 --> {Tom}: Unbonding Mickey from my network
 --> {Popeye}: Unbonding Mickey from my network
 --> {Donald}: Unbonding Mickey from my network
```

Figure 35: Unbonding the cheater from Figure 33 in Tendermint



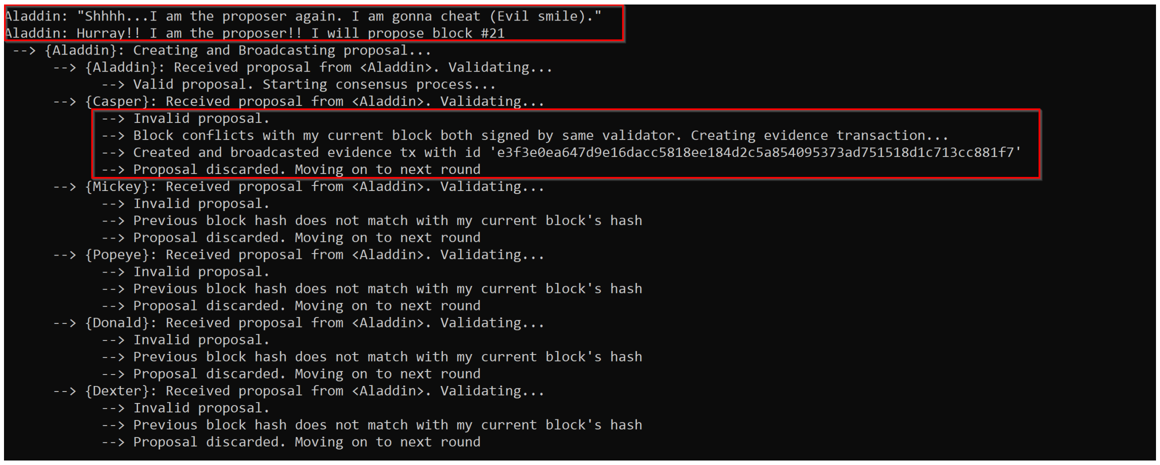
```
Popeye has a chain of length 9, with the following UTXOs:
{"address":"JhlWtHbwXBLve2yT9WwP5Rqq8UYR2BuWfhvvSpv6Ofw=","amount":99,"type":"REG"}
{"address":"0PRWRU3SS8c2lfolrDvo4MlYaXE8OMg0OGF+joDHTyQ=","amount":300,"type":"REG"}
{"address":"fAT+urM72zEAVa+DPU5az/M9ls32H9qkV88M2NduAMY=","amount":172,"type":"REG"}
{"address":"olEjcNbKo530QSwsCCRbOBkzF9L0a6hEsIgwzscXr3g=","amount":30,"type":"BND"}
{"address":"KP4dKo+Px/TiGWDCyf8lJQvhHnw44Qjtm+X6YZ3R00I=","amount":15,"type":"REG"}
{"address":"cigRqZGFFsWlhO54Rbi4YRgvaZa+QOPhbfVo44fFHIY=","amount":10,"type":"BND"}
{"address":"zatGVilhkkG1vdstFEbdHDhd0YiGDO034Co9lQEmjwc=","amount":20,"type":"REG"}
{"address":"qI45WKzNGT+2ssmch5ERRrfB6GyqncOF8/LIa/hBGOk=","amount":40,"type":"REG"}
{"address":"idS0RxmaqBY/lV6EsXRRrEUgsWbTqa/3oGBHe+KCi2U=","amount":92,"type":"REG"}
{"address":"7suDKNzYlMy1X59GkHRBOhaFp2fWq0WAMScMB6LnAOA=","amount":100,"type":"UND"}
{"address":"mLPB4bMmiM9hBB73CB4pduEtSZXwQYp3WzEwCxtYzPU=","amount":60,"type":"BND"}
{"address":"+rVQb9/Gan5yA6E58rsHQ7Ni8OnkryqS+6MtmVx8MoY=","amount":7,"type":"REG"}
{"address":"YGGUMrSwB60AEk8XtkEwV7fq9s20NzKj14ab1qYpySw=","amount":75,"type":"UND"}
{"address":"ABp0ib5JIyN6NtHXV4vpydNnF79P4/oKJZ1gNdXdrC8=","amount":75,"type":"UND"}
```

Figure 36: Seized UTXOs of the cheater from Figure 33 in Tendermint



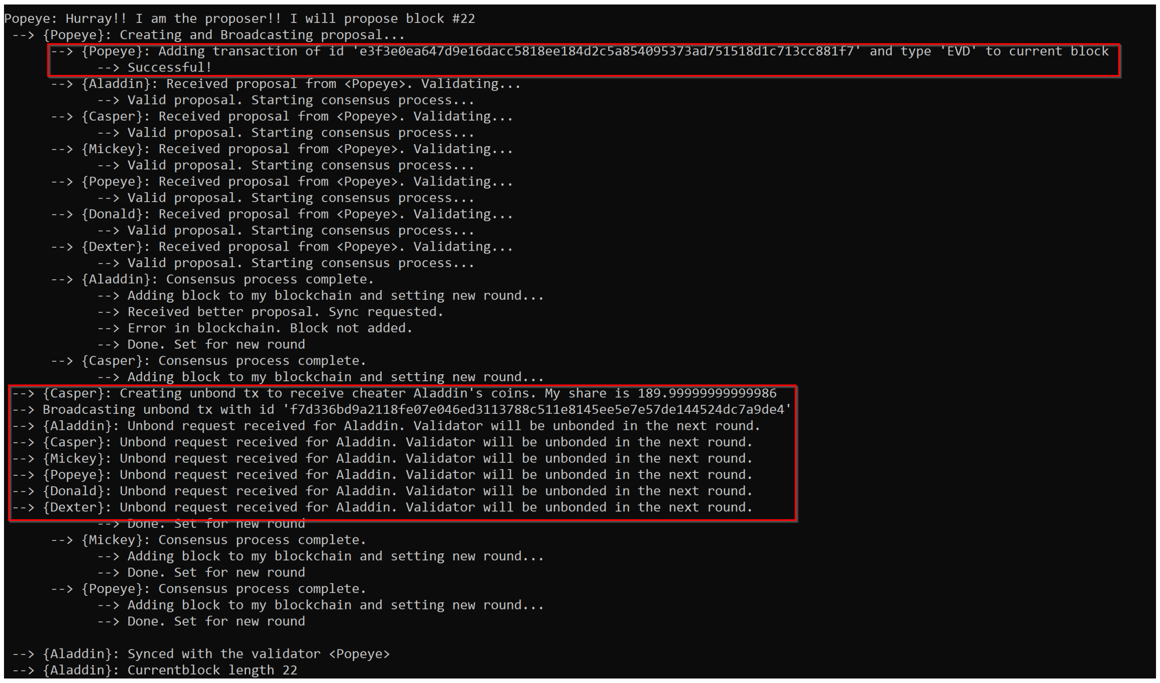
```
Validators:
Minnie - {Balance: 400 coins}
Mickey - {Balance: 172 coins}
Popeye - {Balance: 90 coins}
Donald - {Balance: 95 coins}
```

Figure 37: Updated wallets of validators after seizing the coins of the cheater from Figure 33 in Tendermint



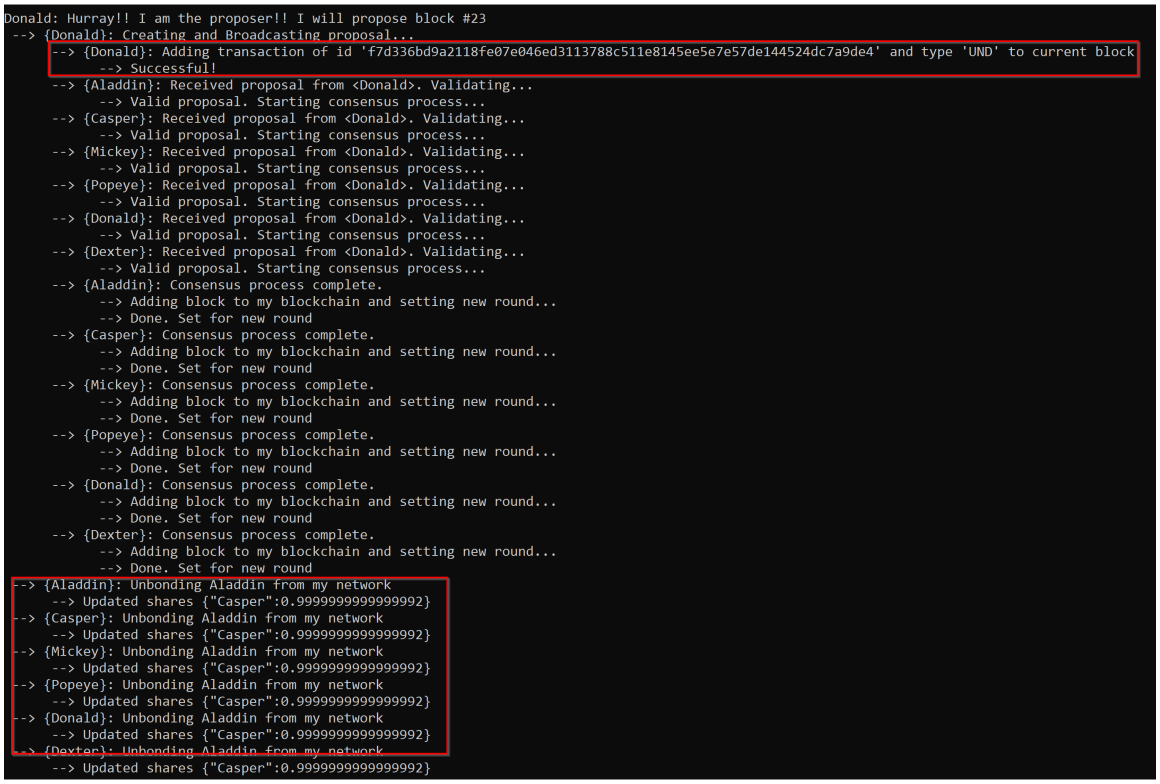
```
Aladdin: "Shhhh...I am the proposer again. I am gonna cheat (Evil smile)."
Aladdin: Hurray!! I am the proposer!! I will propose block #21
 --> {Aladdin}: Creating and Broadcasting proposal...
      --> {Aladdin}: Received proposal from <Aladdin>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Casper}: Received proposal from <Aladdin>. Validating...
            --> Invalid proposal.
            --> Block conflicts with my current block both signed by same validator. Creating evidence transaction...
            --> Created and broadcasted evidence tx with id 'e3f3e0ea647d9e16dacc5818ee184d2c5a854095373ad751518d1c713cc881f7'
            --> Proposal discarded. Moving on to next round
      --> {Mickey}: Received proposal from <Aladdin>. Validating...
            --> Invalid proposal.
            --> Previous block hash does not match with my current block's hash
            --> Proposal discarded. Moving on to next round
      --> {Popeye}: Received proposal from <Aladdin>. Validating...
            --> Invalid proposal.
            --> Previous block hash does not match with my current block's hash
            --> Proposal discarded. Moving on to next round
      --> {Donald}: Received proposal from <Aladdin>. Validating...
            --> Invalid proposal.
            --> Previous block hash does not match with my current block's hash
            --> Proposal discarded. Moving on to next round
      --> {Dexter}: Received proposal from <Aladdin>. Validating...
            --> Invalid proposal.
            --> Previous block hash does not match with my current block's hash
            --> Proposal discarded. Moving on to next round
```

Figure 38: A cheater getting caught in TontineCoin



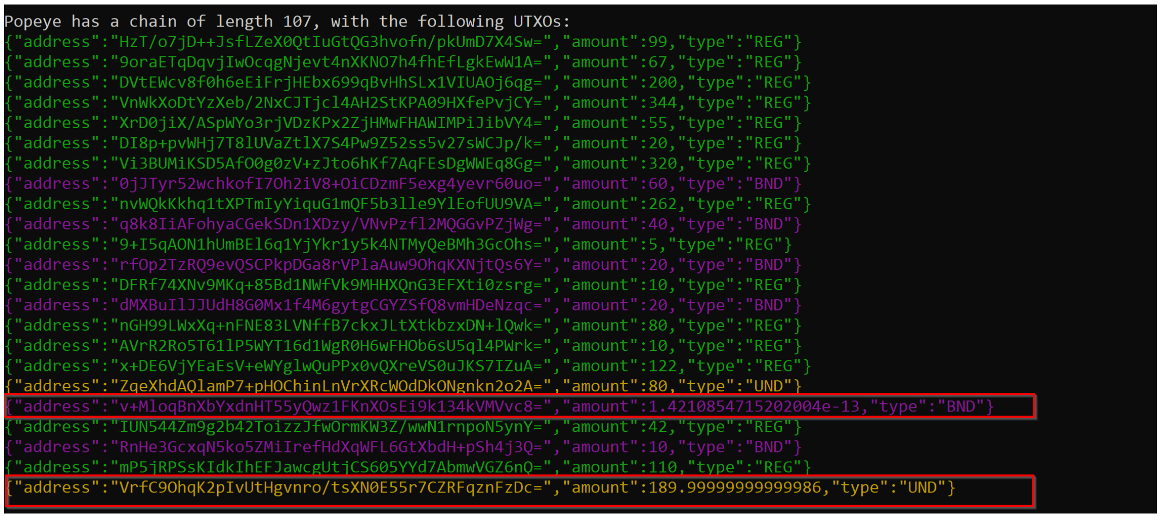
```
Popeye: Hurray!! I am the proposer!! I will propose block #22
 --> {Popeye}: Creating and Broadcasting proposal...
      --> {Popeye}: Adding transaction of id 'e3f3e0ea647d9e16dacc5818ee184d2c5a854095373ad751518d1c713cc881f7' and type 'EVD' to current block
            --> Successful!
      --> {Aladdin}: Received proposal from <Popeye>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Casper}: Received proposal from <Popeye>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Mickey}: Received proposal from <Popeye>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Popeye}: Received proposal from <Popeye>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Donald}: Received proposal from <Popeye>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Dexter}: Received proposal from <Popeye>. Validating...
            --> Valid proposal. Starting consensus process...
      --> {Aladdin}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Received better proposal. Sync requested.
            --> Error in blockchain. Block not added.
            --> Done. Set for new round
      --> {Casper}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
 --> {Casper}: Creating unbond tx to receive cheater Aladdin's coins. My share is 189.99999999999986
 --> Broadcasting unbond tx with id 'f7d336bd9a2118fe07e046ed3113788c511e8145ee5e7e57de144524dc7a9de4'
 --> {Aladdin}: Unbond request received for Aladdin. Validator will be unbonded in the next round.
 --> {Casper}: Unbond request received for Aladdin. Validator will be unbonded in the next round.
 --> {Mickey}: Unbond request received for Aladdin. Validator will be unbonded in the next round.
 --> {Popeye}: Unbond request received for Aladdin. Validator will be unbonded in the next round.
 --> {Donald}: Unbond request received for Aladdin. Validator will be unbonded in the next round.
 --> {Dexter}: Unbond request received for Aladdin. Validator will be unbonded in the next round.
            --> Done. Set for new round
      --> {Mickey}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round
      --> {Popeye}: Consensus process complete.
            --> Adding block to my blockchain and setting new round...
            --> Done. Set for new round

 --> {Aladdin}: Synced with the validator <Popeye>
 --> {Aladdin}: Currentblock length 22
```

Figure 39: Inclusion of EVD transaction created in Figure 38 and UND transaction to seize cheater's stake in TontineCoin



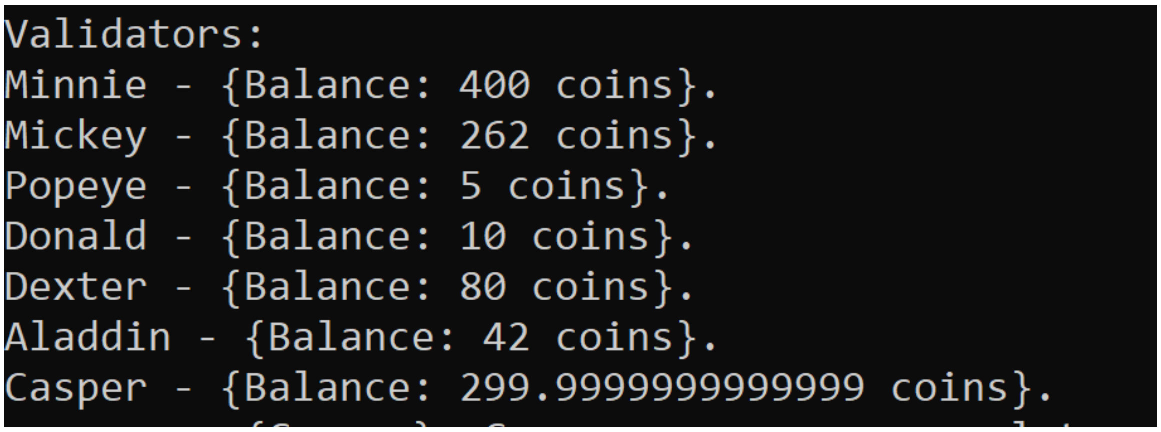
```
Donald: Hurray!! I am the proposer!! I will propose block #23
 --> {Donald}: Creating and Broadcasting proposal...
     --> {Donald}: Adding transaction of id 'f7d336bd9a2118fe07e046ed3113788c511e8145ee5e7e57de144524dc7a9de4' and type 'UND' to current block
           --> Successful!
     --> {Aladdin}: Received proposal from <Donald>. Validating...
           --> Valid proposal. Starting consensus process...
     --> {Casper}: Received proposal from <Donald>. Validating...
           --> Valid proposal. Starting consensus process...
     --> {Mickey}: Received proposal from <Donald>. Validating...
           --> Valid proposal. Starting consensus process...
     --> {Popeye}: Received proposal from <Donald>. Validating...
           --> Valid proposal. Starting consensus process...
     --> {Donald}: Received proposal from <Donald>. Validating...
           --> Valid proposal. Starting consensus process...
     --> {Dexter}: Received proposal from <Donald>. Validating...
           --> Valid proposal. Starting consensus process...
     --> {Aladdin}: Consensus process complete.
           --> Adding block to my blockchain and setting new round...
           --> Done. Set for new round
     --> {Casper}: Consensus process complete.
           --> Adding block to my blockchain and setting new round...
           --> Done. Set for new round
     --> {Mickey}: Consensus process complete.
           --> Adding block to my blockchain and setting new round...
           --> Done. Set for new round
     --> {Popeye}: Consensus process complete.
           --> Adding block to my blockchain and setting new round...
           --> Done. Set for new round
     --> {Donald}: Consensus process complete.
           --> Adding block to my blockchain and setting new round...
           --> Done. Set for new round
     --> {Dexter}: Consensus process complete.
           --> Adding block to my blockchain and setting new round...
           --> Done. Set for new round
 --> {Aladdin}: Unbonding Aladdin from my network
     --> Updated shares {"Casper":0.9999999999999992}
 --> {Casper}: Unbonding Aladdin from my network
     --> Updated shares {"Casper":0.9999999999999992}
 --> {Mickey}: Unbonding Aladdin from my network
     --> Updated shares {"Casper":0.9999999999999992}
 --> {Popeye}: Unbonding Aladdin from my network
     --> Updated shares {"Casper":0.9999999999999992}
 --> {Donald}: Unbonding Aladdin from my network
     --> Updated shares {"Casper":0.9999999999999992}
 --> {Dexter}: Unbonding Aladdin from my network
     --> Updated shares {"Casper":0.9999999999999992}
```

Figure 40: Unbonding the cheater from Figure 38 in TontineCoin



```
Popeye has a chain of length 107, with the following UTXOs:
{"address":"HzT/o7jD++JsfLZeX0QtIuGtQG3hvofn/pkUmD7X4Sw=","amount":99,"type":"REG"}
{"address":"9oraETqDqvjIwOcqgNjevt4nXKNO7h4fhEfLgkEwW1A=","amount":67,"type":"REG"}
{"address":"DVtEWcv8f0h6eEiFrjHEbx699qBvHhSLx1VIUAOj6qg=","amount":200,"type":"REG"}
{"address":"VnWkXoDtYzXeb/2NxCJTjcl4AH2StKPA09HXfePvjCY=","amount":344,"type":"REG"}
{"address":"XrD0jiX/ASpWYo3rjVDzKPx2ZjHMwFHAWIMPiJibVY4=","amount":55,"type":"REG"}
{"address":"DI8p+pvWHj7T8lUVaZtlX7S4Pw9Z52ss5v27sWCJp/k=","amount":20,"type":"REG"}
{"address":"Vi3BUMiKSD5AfO0g0zV+zJto6hKf7AqFEsDgWWEq8Gg=","amount":320,"type":"REG"}
{"address":"0jJTyr52wchkofI7Oh2iV8+OiCDzmF5exg4yevr60uo=","amount":60,"type":"BND"}
{"address":"nvWQkKkhq1tXPTmIyYiquG1mQF5b3lle9YlEofUU9VA=","amount":262,"type":"REG"}
{"address":"q8k8IiAFohyaCGekSDn1XDzy/VNvPzfl2MQGGvPZjWg=","amount":40,"type":"BND"}
{"address":"9+I5qAON1hUmBEl6q1YjYkr1y5k4NTMyQeBMh3GcOhs=","amount":5,"type":"REG"}
{"address":"rfOp2TzRQ9evQSCPkpDGa8rVPlaAuw9OhqKXNjtQs6Y=","amount":20,"type":"BND"}
{"address":"DFRf74XNv9MKq+85Bd1NWfVk9MHHXQnG3EFXti0zsrg=","amount":10,"type":"REG"}
{"address":"dMXBuIlJJUdH8G0Mx1f4M6gytgCGYZSfQ8vmHDeNzqc=","amount":20,"type":"BND"}
{"address":"nGH99LWxXq+nFNE83LVNffB7ckxJLtXtkbzxDN+lQwk=","amount":80,"type":"REG"}
{"address":"AVrR2Ro5T61lP5WYT16d1WgR0H6wFHOb6sU5ql4PWrk=","amount":10,"type":"REG"}
{"address":"x+DE6VjYEaEsV+eWYglwQuPPx0vQXreVS0uJKS7IZuA=","amount":122,"type":"REG"}
{"address":"ZqeXhdAQlamP7+pHOChinLnVrXRcWOdDkONgnkn2o2A=","amount":80,"type":"UND"}
{"address":"v+MloqBnXbYxdnHT55yQwz1FKnXOsEi9k134kVMVvc8=","amount":1.4210854715202004e-13,"type":"BND"}
{"address":"IUN544Zm9g2b42ToizzJfwOrmKW3Z/wwN1rnpoN5ynY=","amount":42,"type":"REG"}
{"address":"RnHe3GcxqN5ko5ZMiIrefHdXqWFL6GtXbdH+pSh4j3Q=","amount":10,"type":"BND"}
{"address":"mP5jRPSsKIdkIhEFJawcgUtjCS605YYd7AbmwVGZ6nQ=","amount":110,"type":"REG"}
{"address":"VrfC9OhqK2pIvUtHgvnro/tsXN0E55r7CZRFqznFzDc=","amount":189.99999999999986,"type":"UND"}
```

Figure 41: Seized UTXOs of the cheater from Figure 38 in TontineCoin



```
Validators:
Minnie - {Balance: 400 coins}.
Mickey - {Balance: 262 coins}.
Popeye - {Balance: 5 coins}.
Donald - {Balance: 10 coins}.
Dexter - {Balance: 80 coins}.
Aladdin - {Balance: 42 coins}.
Casper - {Balance: 299.9999999999999 coins}.
```

Figure 42: Updated wallets of validators after seizing the coins of the cheater from Figure 38 in TontineCoin

CHAPTER 7

Conclusion and Future Enhancements

This project has implemented a prototype of Tendermint and built a TontineCoin model based on it. It has simulated and contrasted five different normal operations in both these prototypes. These operations are initialization, the consensus process, transfer of coins among clients, bonding of clients, and unbonding of validators. The TontineCoin model differs from the Tendermint model in the execution of the above operations except for the transfer of coins. This project has also simulated and discussed how a nothing-at-stake attack is handled in TontineCoin compared to Tendermint. TontineCoin's approach of monitoring and ejecting dishonest validators differs from that of Tendermint's.

There are a few areas in which the TontineCoin model can be improved in future. A major enhancement would be to add support in it to validate TontineCoin's claim which ensures improved monitoring system provided the cost of monitoring is below its benefit. Also, currently, a block in the TontineCoin model does not include the current bid leader's PoW and hence is not compared with the bidding tontine's PoW. The last tontine to bid before the selection block reaches is selected. This can be improved in the future implementation. Additionally, since a cheating validator's stake in TontineCoin model is seized based on the associated tontine members' fractional shares, it is not completely seized. This can be seen in figure 41. The formula to calculate the new share can be modified to handle the precision issue in the future implementation.

The current implementation does not support a transaction fee in transactions. This is expected to avoid clients and validators from posting false transactions. The future implementation can be enhanced with this feature. Additionally, the current implementation is name-based instead of address-based. This means a validator stores other validator's information based on their name instead of their public key or address. This is not expected since a cryptocurrency is required to offer complete or partial anonymity. This can be rectified in a revised implementation.

LIST OF REFERENCES

[1] S. Nakamoto, ‘‘Bitcoin: A peer-to-peer electronic cash system,’’ 2008.

[2] Z. Zheng, S. Xie, H.-N. Dai, X. Chen, and H. Wang, ‘‘Blockchain challenges and opportunities: A survey,’’ *International Journal of Web and Grid Services*, vol. 14, no. 4, pp. 352--375, 2018.

[3] C. Dwork and M. Naor, ‘‘Pricing via processing or combatting junk mail,’’ in *Annual International Cryptology Conference.* Springer, 1992, pp. 139--147.

[4] M. Jakobsson and A. Juels, ‘‘Proofs of work and bread pudding protocols,’’ in *Proceedings of the IFIP TC6/TC11 Joint Working Conference on Secure Information Networks: Communications and Multimedia Security*, ser. CMS ’99. NLD: Kluwer, B.V., 1999, p. 258–272.

[5] H. Handschuh, *SHA Family (Secure Hash Algorithm).* Boston, MA: Springer US, 2005, pp. 565--567. [Online]. Available: https://doi.org/10.1007/0-387-23483-7_388

[6] C. Percival, ‘‘Stronger key derivation via sequential memory-hard functions.’’

[7] J.-P. Aumasson, S. Neves, Z. Wilcox-O’Hearn, and C. Winnerlein, ‘‘Blake2: Simpler, smaller, fast as md5,’’ in *Applied Cryptography and Network Security*, M. Jacobson, M. Locasto, P. Mohassel, and R. Safavi-Naini, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 119--135.

[8] ‘‘Litecoin - open source p2p digital currency,’’ https://litecoin.org, (Accessed on 01/03/2020).

[9] wedgethx. Youtube. ‘‘Charlie lee’s litecoin presentation at btc miami conference.’’ https://www.youtube.com/watch?v=Le5ByHtssnc. (Accessed on 01/03/2020).

[10] coblee, ‘‘[ann] litecoin - a lite version of bitcoin. launched!’’ https://bitcointalk.org/index.php?topic=47417.0, Oct 2011, (Accessed on 01/03/2020).

[11] B. Asolo, ‘‘Litecoin scrypt algorithm explained,’’ https://www.mycryptopedia.com/litecoin-scrypt-algorithm-explained/, Dec 2018, (Accessed on 01/03/2020).

[12] G. Wood *et al.*, ‘‘Ethereum: A secure decentralised generalised transaction ledger.’’

[13] B. Asolo, ‘‘Ethash explained,’’ https://www.mycryptopedia.com/ethash-explained/, Jan 2019, (Accessed on 01/06/2020).

[14] J. Ray, "Ethash," https://github.com/ethereum/wiki/wiki/Ethash, 2018, (Accessed on 01/06/2020).

[15] K. J. O'Dwyer and D. Malone, "Bitcoin mining and its energy footprint," 2014.

[16] "Bitcoin energy consumption index," https://digiconomist.net/bitcoin-energy-consumption, 2020, (Accessed on 04/06/2020).

[17] QuantumMechanic, "Proof of stake instead of proof of work," https://bitcointalk.org/index.php?topic=27787.0,, Jul 2011, (Accessed on 01/06/2020).

[18] S. King and S. Nadal, "Ppcoin: Peer-to-peer crypto-currency with proof-of-stake," *self-published paper, August*, vol. 19, 2012.

[19] P. Vasin, "Blackcoin's proof-of-stake protocol."

[20] "What is blackcoin?" https://www.americascardroom.eu/blackcoin-cryptocurrency/, (Accessed on 01/08/2020).

[21] W. Li, S. Andreina, J.-M. Bohli, and G. Karame, "Securing proof-of-stake blockchain protocols," in *Data Privacy Management, Cryptocurrencies and Blockchain Technology.* Springer, 2017, pp. 297--315.

[22] J. Martinez, "Understanding proof of stake: The nothing at stake theory," https://medium.com/coinmonks/understanding-proof-of-stake-the-nothing-at-stake-theory-1f0d71bc027, Jun 2018, (Accessed on 01/08/2020).

[23] J. Kwon, "Tendermint: Consensus without mining," *Draft v. 0.6, fall*, vol. 1, no. 11, 2014.

[24] C. Dwork, N. Lynch, and L. Stockmeyer, "Consensus in the presence of partial synchrony," *Journal of the ACM (JACM)*, vol. 35, no. 2, pp. 288--323, 1988.

[25] C. Pollett, T. Austin, K. Potika, and J. Rietz, "Tontinecoin: Murder-based proof-of-stake," 2020, unpublished.

[26] T. H. Austin, "Spartan gold," https://github.com/taustin/spartan-gold, 2018, (Accessed on 09/08/2019).

[27] A. Back, "Hashcash-a denial of service counter-measure," 2002.

[28] M. Taylor, "The evolution of bitcoin hardware," *Computer*, vol. 50, pp. 58--66, 01 2017.

[29] "Proposer selection procedure in tendermint," https://docs.tendermint.com/master/spec/reactors/consensus/proposer-selection.html, 2020, (Accessed on 10/18/2019).

[30] ‘‘What is tendermint?’’ https://docs.tendermint.com/master/introduction/what-is-tendermint.html, (Accessed on 10/31/2019).

[31] ‘‘Running in production,’’ https://tendermint.com/docs/tendermint-core/running-in-production.html#dos-exposure-and-mitigation, 2020, (Accessed on 10/25/2019).

[32] C. Unchained, ‘‘Tendermint explained — bringing bft-based pos to the public blockchain domain,’’ https://blog.cosmos.network/tendermint-explained-bringing-bft-based-pos-to-the-public-blockchain-domain-f22e274a0fdb, May 2018, (Accessed on 09/05/2019).

[33] K. McKeever, ‘‘A short history of tontines,’’ *Fordham J. Corp. & Fin. L.*, vol. 15, p. 491, 2009.

[34] G. Hirsch, ‘‘Tontines, tontine insurance, and commercial culture: Stevenson and osbourne’s the wrong box,’’ *Robert Louis Stevenson: Writer of Boundaries*, pp. 83--94, 01 2006.

[35] R. M. Jennings, D. F. Swanson, and A. P. Trout, ‘‘Alexander hamilton’s tontine proposal,’’ *The William and Mary Quarterly: A Magazine of Early American History*, pp. 107--115, 1988.

[36] M. A. Milevsky, *King William’s Tontine: why the retirement annuity of the future should resemble its past.* Cambridge University Press, 2015.

[37] ‘‘Rhostyllen: a history through pictures,’’ http://rhostyllen.info/misc.html, (Accessed on 12/10/2019).

[38] R. L. Ransom and R. Sutch, ‘‘Tontine insurance and the armstrong investigation: a case of stifled innovation, 1868--1905,’’ *The Journal of Economic History*, vol. 47, no. 2, pp. 379--390, 1987.