Implicit Consensus: Blockchain with Unbounded Throughput

Article · May 2017

CITATIONS
0

READS
84

4 authors, including:

Zhijie Ren
Delft University of Technology
11 PUBLICATIONS **17** CITATIONS
SEE PROFILE

Erkin Zekeriya
Delft University of Technology
66 PUBLICATIONS **1,196** CITATIONS
SEE PROFILE

Implicit Consensus: Blockchain with Unbounded Throughput

Zhijie Ren[1], Kelong Cong[2], Johan Pouwelse[2], and Zekeriya Erkin[1]

[1] Cyber Security Group, Dept. of Intelligent Systems
[2] Distributed Systems Group, Dept. of Software Engineering
Delft University of Technology, Mekelweg 4, Delft, the Netherlands
z.ren@tudelft.nl,

Abstract. Recently, the blockchain technique was put in the spotlight as it introduced a systematic approach for multiple parties to reach consensus without needing trust. However, the application of this technique in practice is severely restricted due to its limitations in throughput. In this paper, we propose a novel consensus model, namely the implicit consensus, with a distinctive blockchain-based distributed ledger in which each node holds its individual blockchain. In our system, the consensus is not on the transactions, but on a special type of blocks called Check Points that are used to validate individual transactions. Our system exploits the ideas of self-interest and spontaneous sharding and achieves unbounded throughput with the transaction reliability that equivalent to traditional Byzantine fault tolerance schemes.

Keywords: blockchain, distributed ledger, consensus algorithm, byzantine fault tolerance

1 Introduction

Blockchain, introduced and firstly applied in Bitcoin [1], is one of the hottest techniques in the information technology at this moment. Researchers see a huge potential in this technique and believe that it can be used as a systematic approach to replace trust. More precisely, in a network in which nodes do not trust each other, blockchain provides a way for the nodes to reach consensus and cooperate without a third party or a central authority. Typically, a blockchain technique is a distributed append-only database which consists of two components. First, as its name suggests, the database is an ordered sequences of blocks which are chained together. The newly generated data forms a new block and chains to the existing chain with a digest of the cryptographic hash function of the previous block. Second, a consensus algorithm is used for the network to agree on the new block that will be appended to the chain.

1.1 Problem Statement

One of the major problems of the blockchain technique is the scalability, which is caused by the limitation of the current consensus algorithms. Reaching consensus for a number of nodes that might be malicious, commonly known as

the Byzantine fault tolerance (BFT) problem [2], has been extensively studied for over 30 years. The problem could be described as the following. In an asynchronous network with malicious nodes, a BFT scheme should have the following properties.

- **Agreement:** If an honest node propose a vector v, then all honest nodes agree with v.
- **Correctness:** If an honest node decides v, then v must be proposed by at least one honest node.
- **Termination:** If an honest node propose v, then all honest node will eventually decide a vector.

In general, these properties are not feasible for a full-asynchronous network [3]. However, with some assumptions like weak synchronous or probabilistic agreement, these properties can be achieved with message complexity of $O(N^2)$ [4,5] when the number of the adversary is less than $\lfloor \frac{N-1}{3} \rfloor$.

1.2 State-of-the-Art

Towards the scalability of BFT. Many schemes have been proposed to increase the scalability of the traditional BFT schemes. Luu et al. divides the network into subgroups uniformly and runs BFT within the subgroup [6]. Other schemes like [7,8] opportunistically apply simpler schemes for good network scenarios, usually with message complexity of $O(N)$, and use "safe" schemes like PBFT [5] as backup. In networks with good connectivity and limited amount of malicious node, these approaches result in message complexity of $O(N)$ but risks even higher latency comparing to traditional BFT schemes in bad situations.

Increasing the throughput of POW. By introducing mining costs for messaging and incentives to compensate the costs, Bitcoin proposed a scalable solution for BFT that reduces the message complexity to $O(N)$ and functions in large networks with thousands of nodes. However, this scheme requires synchrony amongst the majority, which restricts the throughput of Bitcoin since the block size and frequency are then limited [9]. Scalable Bitcoin schemes like Bitcoin-NG and lightning network [10,11] separate the consensus from the contents of the block. As a result, the synchrony is not restricted by the block size and frequency. However, extra economical punishment methods are required since the costs and incentives of the traditional POW no longer gives protection to the validity of the contents.

Leader selection. The schemes of [1,10,11] can be seen as two parts: an economical rewarded leader election process and an economical guaranteed validation process. Schemes like Byzcoin, ALGORAND, hybrid consensus, and the sleepy model of consensus [8,12,13,14] keep the leader selection part. Then, instead of a sole leader, they select a group of leaders and the validation is guaranteed by BFT algorithms. This approach has some advantages over both POW and BFT. However, extra effort is needed to guarantee that the leader selection is unbiased, i.e., the malicious nodes cannot predict or control the selection results.

Sharding. The sharding idea proposed in Etheruem and Omniledger [15,16] has received a lot of attention recently. The basic idea of sharding is that the network is divided into shards and the intra-shard transactions are only agreed, validated, and recorded in the nodes of that shard. Then, some nodes are selected to validate and record the inter-shard transactions. In networks where the transactions pattern is rather isolated, sharding has a potential of achieving unbounded performance, i.e., less than $O(N)$ message complexity. The reason is that the transactions are not mandatory to be broadcast to the whole network. However, the choice of the shards for a network to achieve optimal performance remains a challenge.

1.3 Implicit Consensus

In this paper, a novel consensus model, namely the *implicit consensus*, is proposed, which achieves unbounded throughput for a distributed ledger type of blockchain system with equivalent reliability on validated transactions to traditional BFT based schemes. It has some similarities to some of the existing ideas like the side-chain ideas used in [11] and sharding [15,16]. However, compare to the existing schemes, the main innovations of our consensus model are the following.

- Replacing the *termination* property of BFT schemes by the *self-interest phenomenon*. In other words, for each transaction, our scheme does not guarantee consensus. However, because it is in the interest of the related parties, e.g., the issuer of this transactions, to convince the other nodes that this transaction is valid. Thus, nodes are encouraged to prove the validity and agreement of their transactions to as many nodes as they can and we guarantee that the malicious nodes cannot prevent them from doing that. It is arguably more close to real-life scenario. To distinguish this from traditional consensus in which each transaction has explicitly reached consensus (termination property), we call this *implicit consensus*. With implicit consensus, our scheme is scalable since the message complexity is reduced to $O(N)$.
- Spontaneous sharding. For each transaction, we prove that the other two properties of BFT, *agreement and correctness*, will hold as long as the transactions is locally validated by our validation scheme. This is done with collecting some faction of the total transactions of the network related to this transaction, called *proofs*. As a result, the network is spontaneously sharded since rational nodes will optimize their storage and transmission costs by only validating and recording the proofs of their transactions. There is no need for mechanisms that allocate the transaction set needed to be validated and recored for each node. Similar as sharding, our scheme achieves unbounded performance, i.e., the message complexity is less than $O(N)$ since transactions are not mandatory to be broadcast to the whole network.
- Uncompromised reliability. Although our scheme has its similarity to side-chain schemes like lightning network [11], we guarantee that the validated transactions on the the individual chains (can be seen as side-chains) are as

reliable as they are on the main chain. This is fundamentally different from side-chain approaches like lightning network, in which the reliability is not guaranteed by the main chain, but by the locked deposit.

1.4 Structure of Our System

We consider an asynchronous network with N nodes and $f \leq \lfloor \frac{N-1}{3} \rfloor$ adversaries. To achieve the implicit consensus, we propose a permissioned blockchain-based distributed ledger consisting of four layers: transactions, individual blockchains, the consensus scheme, and the validation scheme. The first layer of our system is **transactions**, which are defined in a similar fashion as Bitcoin. The second layer is **individual blockchains**. In our system each node has its own genesis block and blockchain, in which only transactions that related to the node itself are recorded. Besides the blocks that consists of transactions which are called *Transaction Blocks (TBs)*, a special type of blocks called *Check Point (CP)* is introduced. The CPs contain no transaction, but some already established consensus and a hash of the previous block. The third layer of our scheme is the **consensus scheme**, in which we plug in one of the existing Byzantine fault tolerance (BFT) schemes like [5,7,17,18] to reach consensus on the hashes of the CPs. One of the fundamental differences between our system and other blockchain systems is that the consensus is reached only on the hashes of the CPs instead of all transactions. As a result, if some CP reached consensus, the transactions that came before the CP are tamper-proof. However, this tells nothing about the validity of the transactions in these parts. Hence, in the fourth part, a **validation scheme** is used to validate individual transactions. The validation scheme is executed locally and only based on point-to-point communications. Since the CPs in the consensus have "sealed" the chains, the authenticity and integrity of the chains can be easily verified. We prove that although the validation scheme is run locally, the result is correct and consistent for all honest nodes, which suggests the agreement and correctness properties, i.e., implicit consensus.

1.5 Content of the Paper

Firstly, we introduce the four-layer system in Section 2. Then we show the necessary theorems and proofs in Section 3. The performance of our system is discussed with the focus on throughput and reliability in Section 4. In Section 5, we conclude our work.

2 Our System

2.1 Transactions

We consider a transaction based value-exchange blockchain system similar to Bitcoin, i.e., a distributed ledger. The s-th transaction from node i to node j is

denoted by $tr(i \to j, s)$. In this paper, a cryptographic hash function is denoted by $Y = H(X)$, in which Y is called the *digest* of X. Furthermore, we assume that there is a public-key infrastructure (PKI) such that each node holds a secret private key while the corresponding public key is known to all other nodes. A transaction $tr(i \to j, s)$ contains the following information.

- The sender and the receiver, i.e., i and j.
- A unique serial number s one-to-one mapped to this transaction $tr(i \to j, s)$.
- The indices of the sources of this transactions (see Definition 2). The sources are denoted by $\mathcal{S}(tr(i \to j, s))$, which is a set of previous transactions send to i. This is equivalent to the input of Bitcoin.
- The value of this transaction V_t and the remaining value V_r after this transaction. This is equivalent to the output of Bitcoin.
- A digital signature created by node i, which is the digest of the aforementioned items encrypted with the private key of i.

Here, we only consider two-party transactions. The transactions are only recorded in the chains of the related node. Hence, a transaction $tr(i \to j, s)$ is recorded in the chains of nodes i and j.

2.2 Individual Blockchains

We consider a permissioned network with N nodes and each node has its own blockchain. The blockchains are denoted by $\mathcal{B}_i, i \in \{1, 2, \ldots, N\}$ [3]. A blockchain $\mathcal{B}_i$ is then defined as an ordered set of blocks $\{B_i(1), B_i(2), \ldots\}$, in which each block contains a digest of its previous block, i.e., block $B_i(j), j > 1$ will contain $H(B_i(j-1))$. The genesis blocks (the first blocks in the chains) $B_i(1)$ are distinctive and contain the information about their unique identities and the initial balance of each node[4]. The initial balance can be seen as a transaction without sources and has an unique index (see Definition 2). Furthermore, this transaction is valid if the corresponding genesis block is included in some consensus. Otherwise, it is invalid (see Subsections 2.3 and 2.4).

There are two types of blocks in the chains: transaction blocks (TBs) and check points (CPs). We assume all genesis blocks are CPs. TBs are used to record the transactions and CPs are used for the consensus scheme. Now we introduce these two types of blocks.

Transaction Blocks As its name suggests, TB is used to record the transactions. We denote the k-th TB in $\mathcal{B}_i$ by $T_i(k)$. Then, if $T_i(k)$ is the j-th block in $\mathcal{B}_i$, we say that $T_i(k) \equiv B_i(j)$. A transaction block $T_i(k)$ consists of a digest of the previous block and M transaction messages $t_i(k, m)$, i.e., if $T_i(k) \equiv B_i(j)$, then $T_i(k)$ consists of $[H(B_i(j-1)), t_i(k, 1), t_i(k, 2), \ldots, t_i(k, M)]$.

[3] This definition is slightly naive since malicious nodes could have multiple versions of their blockchains. For the sake of easier comprehension, we use this definition here and will further address this problem in Section 3.

[4] Our system only focus on the reliable value exchange, thus we assume that there exists some pre-established agreement on the initial balance for the nodes.

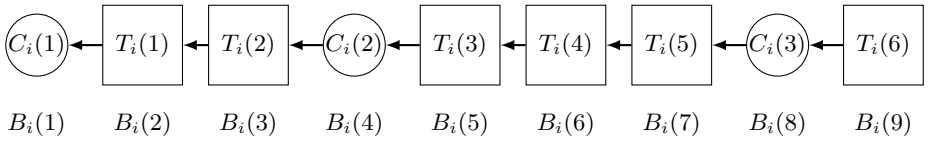

Fig. 1. Example of a blockchain of node i with six TBs and three CPs.

Definition 1 (Transaction Message). *A* **transaction message** $t_i(k, m)$ *is the m-th message in the k-th transaction block. It consists a transaction $tr(a \to b, s)$ where $a = i$ or $b = i$.*

Definition 2 (Transaction Index). *If a transaction $tr(a \to b, s)$ is in the transaction message $t_i(k, m)$, then a vector of $[i, k, m]$ are called the* **index** *of this transaction.*

Note that since a transaction is written in the chains of both the sender and the receiver, a valid transaction should have two indices. Also, the two transaction messages of a transaction are identical.

Check Points CPs are a special type of blocks which contain no transaction. Instead, they contain some established consensus. We use the consensus scheme to reach consensus on the digests of the CPs of this round. Similar to the TBs, we denote the k-th CP in $\mathcal{B}_i$ by $C_i(k)$. Then, if $C_i(k)$ is the j-th block in $\mathcal{B}_i$, we say that $C_i(k) \equiv B_i(j)$. The CPs are defined as follows.

Definition 3 (Check Point (CP)). *A* **check point** *consists of a digest of the previous block and the consensus established in the previous round (see Subsection 2.3).*

The relationship between blocks, TBs, and CPs is shown in Figure 1.

2.3 Consensus Scheme

Our consensus scheme is a *consensus process* used repetitively in *rounds*. In each round, the consensus process is used to reach consensus on consensus messages (CMs), which is defined as follows.

Definition 4 (Consensus Message (CM)). *A* **consensus message** *of node i in round r denoted by $M_i(r)$ consists of the following information.*

- *i and r.*
- *The digest of a CP $C_i(k)$ which has not been* **included** *in any consensus. We call a CP is* **included** *in some consensus if and only if the digest of this CP is in a CM and that CM has reached consensus.*
- *The position of the CP $C_i(k)$ and its previous CP $C_i(k-1)$ in the chain, i.e., two numbers j and j' that $C_i(k) \equiv B_i(j)$ and $C_i(k-1) \equiv B_i(j')$.*

– *A digital signature of i, which is the digest of the aforementioned items encrypted by the private key of i.*

The consensus process of round r starts when the consensus process of round $r-1$ is complete and the consensus result, denoted by $CON(r-1)$, is acknowledged by all honest nodes. Now, we describe the steps of the consensus process for node i of the r-th round.

– **Step 1:** After $CON(r-1)$ is obtained, if a CP from node i is included in $CON(r-1)$, it generates a new CP with $CON(r-1)$ and appends it to its chain.
– **Step 2:** It generates a new CM using its latest CP, and uses this as its input for the consensus process of this round.
– **Step 3:** A BFT algorithm is used to reach agreement on a set of input CMs of this round. The following CMs will be excluded from this consensus process by all honest nodes:
  • The CMs of incorrect rounds.
  • The CMs with incorrect digital signatures.
  • The CP included in the CM has already been included in some previous consensus.
  • The index of the previous CP that has been used by a CM which has already reached consensus. Also known as a "fork".
– **Step 4:** Output the result of this consensus process denoted by $CON(r)$, which is a vector consisting the CMs ordered by i.

For **Step 3**, any consensus algorithm that satisfies the agreement, termination, and correctness properties introduced in Subsection 1.1 can be used. Some of the choices are [7,17,18], which tolerant less than $\lfloor \frac{N-1}{3} \rfloor$ malicious nodes. Here, the *honest* and *malicious* nodes are defined as the following.

Definition 5 (Honest Node[5]). *An honest node is a node that creates correct CM messages and cooperates in the consensus process to reach consensus. Moreover, it always validates all of its transaction and only make transactions with correct information, sufficient balance, and validated sources which have not been used in previous transactions (see Subsection 2.4).*

Definition 6 (Malicious Node). *A malicious node can do anything to prevent consensus, creates any kind of transaction, and manipulates its chain, e.g., creates forks, to confuse honest nodes. Moreover, malicious nodes can collude. However, we assume that they cannot break the hash function or the asymmetric encryption.*

[5] Our definition of the honest node is stronger than that in some other literature, in which the honesty is round based, i.e., a node is considered honest if it does not conduct malicious behaviors in that round. However, this strong assumption is solely because that we would like to keep the core system as simple as possible. We will later show in Subsection 4.2 that the definition can be easily weaken to the round based honesty by simple mechanisms.

The consensus result $CON(r)$ is a vector consisting all the CMs that have reached consensus in this round. We denote the already established consensus till round r by $\mathcal{CON}(r) = \{CON(1), CON(2), \ldots, CON(r)\}$. By the properties of the BFT algorithm, $CON(r)$ is known and should be recorded in the blockchains of all honest nodes by the end of round r.

A CP included in $\mathcal{CON}(r)$ guarantees the tamper-proof property in the sense that the transactions previous to this CP are unforgeable. Here, we introduce the term *correct piece*.

Definition 7 (Correct Piece). *An ordered set of blocks* $\{B_i(j), \ldots, B_i(k)\}, k-j \geq 1$ *is called a* **piece** *if* $B_i(j) \equiv C_i(\ell), B_i(k) \equiv C_i(\ell+1)$. *This piece is* **correct** *if and only if:*

- $C_i(\ell)$ *and* $C_i(\ell+1)$ *are both included in* $\mathcal{CON}(r)$.
- *All digests in* $\{B_i(j), B_i(j+1), \ldots, B_i(k)\}$ *are correct.*

2.4 Validation Scheme

With the established consensus $\mathcal{CON}(r)$, the honest nodes can validate individual transactions without any knowledge of the transaction in advance and thus achieves the implicit consensus. In this subsection, we introduce our validation scheme. Note that there are two fundamental difference between our system and other blockchain systems. First, invalid transactions are allowed in our blockchains. Second, there is no globally agreed blockchain and each node might have different observations of the blockchains of the network. Hence, we will first give the definition of the valid transactions and invalid transactions in our system. Then, we show that our validation scheme allows the honest nodes to check the validity of the transactions.

Validity and Conditions for Validation In general, the validity of transactions should be a global and unambiguous property that is independent of the observation of the network by any specific node. In a ledger, a valid transaction should have correct format, sufficient balance, unspent sources, and be signed by the private key of the sender. Besides, each system has its own definition in the validity of the transactions, e.g., a valid transaction in Bitcoin should be in the longest chain for a sufficient long period of time. In our system, a valid transaction should have two messages in both chains of the senders and the receivers. Furthermore, it should also be included in the authentic chain. Hence, we have the following definition.

Definition 8 (Validity). *The validity conditions of a transaction* $tr(i \to j, s)$ *are the following.*

- **Two Messages:** *The transaction is written in exact two identical messages included in the chains of both sender and receiver, respectively.*
- **Correct Chains:** *The two messages are in correct pieces (Definition 7) and all pieces that are previous to these two pieces in the corresponding chains are also correct.*

- **Correct Messages:** *These messages are correct in the sense that all information are correct and signed with the private key of i*
- **Valid Sources:** *The transactions which used as source are valid.*
- **No Double Spending:** *The sources have not been used by other transactions.*
- **Sufficient Balance:** *The transaction value V_t plus the remaining value V_r equals to the sum of all the remaining values of all sources.*

A transaction is **valid** *if it satisfies all the validity conditions in the observation of any node in the network. Otherwise, it is an* **invalid** *transaction.*

Then, to achieve implicit consensus, a validation scheme should satisfy the following conditions.

- **Liveness:** All transactions can be verified by the validation scheme eventually, the result is either "validated"(verified as valid) or "falsificated" (detect as fraud).
- **Correctness:** All transactions validated by the honest nodes are valid. All transactions falsificated by the honest nodes are invalid.

Clearly, if a transaction satisfies the above conditions, it implies that the agreement and correctness properties are satisfied with our validation scheme, i.e., they are in implicit consensus.

Our validation scheme consists of two parts: proof collection and validation process. Now we introduce our validation scheme by considering the case that node u want to validate the transaction $tr(i \to j, s)$, denoted by a function $V_u(tr(i \to j, s))$.

Proof Collection The proof collection is a process that a node requests all necessary information that it needs to validate a transaction, which is called the *proofs* of this transaction.

Definition 9 (Proofs of a transaction). *The proofs of a transaction $tr(i \to j, s)$ consists of the following.*

- *All pieces of $\mathcal{B}_i$ from the first piece in the chain to the first piece which contains $tr(i \to j, s)$.*
- *All pieces of $\mathcal{B}_j$ from the first piece in the chain to the first piece which contains $tr(i \to j, s)$.*
- *For each source transaction of $tr(i \to j, s)$ or recursively the source of the sources until the initial balance in the genesis block, denoted by $tr(k \to l, s')$, all pieces of $\mathcal{B}_k$ signed by k and $\mathcal{B}_l$ signed by l from the first pieces to the ones containing $tr(k \to l, s')$.*

The proofs of a transaction are complete *if all the aforementioned items are collected. The proofs of a transaction are called* correct *if all the collected pieces are correct.*

To collect the proofs, three steps are taken by node u. All collected pieces are verified and the incorrect pieces are immediately discarded. Once the complete and correct proofs of the transaction are collected, the node terminates the proof collection and enters the validation process. If the complete proofs cannot be obtained within a certain time period, the transaction will be marked as "undecided". An undecided transaction could be validated in the future.

- **Step 1:** It requests the transaction indices of $tr(i \to j, s)$ from either node i or j.
- **Step 2:** It requests all the missing proofs from either node i or j.
- **Step 3:** It broadcasts the request of the missing proofs to the whole network.

All the nodes are required to keeps the proofs of all transactions related to themselves.

Validation Process

Definition 10 (Validation Process for a Transaction). *A validation process of a transaction $tr(i \to j, s)$ includes the verification of the following items.*

1. **Two Messages:** *The transaction with the serial number s has two and only two identical messages $t_i(m, k)$ and $t_j(n, \ell)$.*
2. **Correct Messages:** *All information in the messages is correct and signed with the private key of i.*
3. **No Double Spending:** *There are no forks for this transaction, i.e., there does not exist a validated transaction tr' written in message $t_i(m', k')$ with $(k' = k, m' < m)$ or $k' < k$ and the source transactions $\mathcal{S}(tr') \cap \mathcal{S}(tr(i \to j, s)) \neq \emptyset$.*
4. **Validated Sources:** *All the source transactions of $tr(i \to j, s)$ are validated.*
5. **Sufficient Balance:** *The transaction amount plus the remaining amount equals to the sum of the remaining amounts of all sources. All the amounts here are non-negative.*

A transaction that passes or failed the validation process is called a **validated transaction** *or a* **falsificated transaction**, *respectively.*

3 Correctness of the System

The correctness of our system is proved if the agreement, termination, and correctness conditions in Subsection 2.3 are satisfied for the consensus scheme and the liveness and correctness conditions in Subsection 2.4 are satisfied for the validation scheme. The consensus conditions are guaranteed by the BFT schemes. For proofs we refers to the original papers of these schemes [4,5,18]. Here, we prove the validation scheme satisfies the conditions of correctness and liveness. For the sake of space, all the proofs are included in the appendices.

3.1 Liveness

The liveness condition is crucial since in our system, a transaction is only authentic when it is validated. However, as can be observed from our validation scheme, the liveness condition is in general not feasible since we allow the transaction to be "undecided". Now, we give the following theorem and argue that our system is already reliable if we guarantee that the liveness condition holds for all transactions made by honest nodes.

Theorem 1 (Liveness of the Honest Nodes). *If i and j are both honest nodes, the outcome of the validation scheme for a transaction $tr(i \to j, s)$ should be either validated or falsificated before time t.*[6]

Note that Theorem 1 does not guarantee that all transactions are eventually validated or falsificated, i.e., some of the transactions made by malicious node violates the termination property, i.e., they cannot reach consensus to be falsificated. However, the affect to the liveness is very little since the invalid transactions have no impact on the functionality of this system, which is based on the validated transactions. Then, the validated transactions can be proved to be reliable and valid, which will be shown in the following subsection. However, unidentified invalid transactions could cause another problem, spamming, which will be addressed in Subsection 4.2.

3.2 Correctness

Correctness condition guarantees the validity of our validation scheme, i.e., the validation result of the honest nodes will be consistent with the validity of the transactions, which is a global and unambiguous property of the transaction.

Firstly, note that in our system there does not exist a globally agreed set of blockchains $\mathcal{B}_i, i \in \{1, 2, \ldots, N\}$, i.e., in different time, nodes might have different observations of the blockchain set $\mathcal{B}_i$ due to latency or intended forking by malicious nodes. However, all the versions obtained by the honest nodes must be aligned with the already established consensus $\mathcal{CON}(r)$. Hence, we define the view of the blockchains in round r as follows.

Definition 11 (View). *A* **view** *in consensus round r denoted by $I(r)$ is a set of blockchains $\mathcal{B}_i, i \in 1, 2, \ldots, N$ with $\mathcal{CON}(r)$ as its consensus results.*

Basically, a view is the observation of the network by the honest nodes. We now show that the position, order, and the content of the CPs are identical in all possible $I(r)$.

Lemma 1 (Consistency of the CPs). *If $B_i(k)$ and $B_i(\ell)$ are two blocks in the view $I(r)$, both of them are CPs included in the established consensus $\mathcal{CON}(r)$, and $B_i(k)$ is the previous CP of $B_i(\ell)$, then $B_i(k)$ is also the previous CP of $B_i(\ell)$ in any other view $I'(r)$. Moreover, $B_i(k)$ and $B_i(\ell)$ are identical to their counterpart in other views, respectively.*

[6] We show that they will only be validated in Theorem 2.

Then we will show that the CPs protect the consistency of the pieces of the chains, i.e., there cannot exist two distinctive pieces which start from the same CP or end by the same CP which are both correct.

Lemma 2 (Consistency of the Pieces). *If a piece of blockchain* $\mathcal{B} = \{B_i(k), B_i(k+1), \ldots, B_i(\ell)\}$ *in a view* $I(r)$ *is correct, then there does not exist another piece* $\mathcal{B}' = \{B_i(k), B_i(k+1), \ldots, B_i(\ell')\}$ *or* $\mathcal{B}' = \{B_i(k'), B_i(k'+1), \ldots, B_i(\ell)\}$ *in any view* $I'(r'), r' \geq r$ *that is correct.*

By Lemma 2, since the proofs of a transactions are simply a collection of pieces, we directly have the following theorem.

Theorem 2 (Consistency of the Proofs). *If* $\mathcal{P}$ *is the correct and complete proofs of a transaction* $tr(i \to j, s)$ *in a view* $I(r)$*, then there does not exist proofs* $\mathcal{P}' \neq \mathcal{P}$ *of the transaction* $tr(i \to j, s)$ *which are also complete and correct in any view* $I'(r'), r' \geq r$*.*

With the established lemmas and theorems, we prove the main theorem for the correctness of the validation scheme.

Theorem 3 (Correctness of the Validation Scheme). *Assume that* u *is an honest node. Then, if* $V_u(tr(i \to j, s)) =$ validated*, then* $tr(i \to j, s)$ *is valid. If* $V_u(tr(i \to j, s)) =$ falsificated*, then* $tr(i \to j, s)$ *is invalid.*

4 Performance

In this section, we compare the performance of our system to other blockchain systems with the focus on throughput and reliability.

4.1 Throughput

By design, the throughput of our system is independent of the consensus scheme since each node can create as many transactions as they could with no guarantee on validity. A fair throughput comparison should be between the rate of valid transactions in our system, i.e., the amount of total valid transactions made in our system per second, to the transactions rate of the other blockchain systems. The valid transaction rate is determined by the validation process, which is then determined by the amount of proofs that are required for each transaction. At first glance, the amount seems to be a lot since the proofs do not only contain all the related transactions, but also the chains of the nodes who make those transactions. However, the collection is incremental and we will show that the throughput is actually at least as good as some of the existing techniques.

First of all, if the transaction pattern of the network is isolated, e.g., exists subgroups of the network that only have transactions within the subgroup. In that case, the proofs of such transactions will only contain chains of the nodes in that subgroup. As a result, our scheme has the same advantage as sharding

[15,16] and achieves unbounded performance. Since specific sharding mechanism is not needed and the nodes simply optimize the storage and transmission costs spontaneously, we call it spontaneous sharding. A detailed analyze is shown in Appendix B. The same scenario holds for the micro transaction scenario allowed by lightning network. When two nodes transact many transactions with each other, the proofs are no more than the chain of each other. This is no more complicated than the validation in lightning network. However, note that our scheme achieves an uncompromised reliability. Hence, unlike lightning network which only functions for micro transactions, the application of our scheme is not economically restricted.

On the other hand, if the transaction pattern is more correlated in the sense that there is no isolated subgroup, our scheme still has less message complexity than $O(N)$ since not all transactions are required to be collected by each node. More specifically, node i do not need to store the chain of node j, if node j has never made a transaction that is used as the source (or recursively sources of the sources) of node i. As a result, the performance of our scheme is strictly unbounded except for the very extreme transaction patterns. Moreover, rational nodes will try to avoid that situation by choosing sources that minimize the amount of information to transmit. In other words, if the network is rational, it tends to be sharded with our scheme.

Note that although our throughput is unbounded, the latency still depends on the BFT algorithm, thus not scalable. More precisely, the consensus is reached on the CMs with a size of $O(N)$. As a result, the latency would be high in a large network. However, we can reduce the latency by using more scalable and efficient BFT schemes like [7,13,18] since our scheme is not restricted to a specific BFT algorithm.

4.2 Reliability

In has been proved that the reliability of validated transaction is the same as the traditional BFT schemes since the correctness and agreement hold for all honest nodes. Certainly, as discussed in Subsection 3.1, the price we pay is the termination, that is, some of the invalid transactions made by malicious node cannot be falsificated. The undecided transactions themselves do very little harm to the reliability since honest nodes will not use undecided transactions as sources thus this ambiguity will not propagate. However, it does give rooms to the malicious nodes to spam invalid transactions to overwhelm the honest nodes. This problem is similar to the DDoS (Distributed Denial-of-Service) attack which can be solved by some reputation/blacklist scheme. Actually, we believe that keeping the record of the invalid transactions is beneficial to the reliability of the system, since it provides the necessary information for honest nodes to identify malicious nodes and take actions.

Another problem is the degradation in the reliability if we loosen the constraint of the honest nodes in Definition 5 and allow honest nodes to be offline. This will harm the liveness condition since there is a chance that the proofs of valid transactions cannot be obtained when the nodes which have the proofs of

this transaction all go offline. However, this problem is actually solved by the logic behind our system and the "self interest" phenomenon, i.e., every node is responsible for its own transaction. In our system, a transaction is only valid if it is validated by other nodes. Hence, it is in the interest of at least one of the related parties to prove it to the other nodes. Furthermore, if a node wants to use a transaction as the source for its transaction, it not only needs to prove the validation of this transaction, but also needs to keep the proofs and show the proofs to the other related party. The validation scheme is also censorship-free, which suggests that any node that has validated a transaction can independently show the complete and correct proofs to other honest nodes for the validation. In other words, malicious nodes cannot prevent the termination property of valid transactions. As a result, the valid transactions are as reliable as the traditional BFT schemes since the agreement and correctness properties hold and the termination property is guaranteed by the self-interest of related parties.

5 Conclusions and Discussion

In this paper, we proposed a value-exchange blockchain system with a novel consensus model, namely implicit consensus. Our system achieves significant improvements in throughput and other important aspects comparing to other blockchains techniques. We hope that the following proposed concepts would shed a light on the future blockchain research.

- Termination condition of the BFT is not mandatory in a value exchange system such as a distributed ledger. It will always be the interest of some nodes to prove the validity and agreement of the transactions to the rest of the network. Hence, we do not need extra mechanism to force that. Scalability could then be achieved by leaving the termination condition aside.
- Side-chain transactions can be as reliable as they are on the main chain as long as the whole side-chains of all the sources are examined. This is not necessarily a heavy task since the amount of transactions that need to be validated is still only a fraction of the whole transactions set.
- In our system, rational nodes will try to optimize their storage and message transmission by only validate and keep the proofs for their own transactions, which is a spontaneous sharding. Moreover, if multiple sources are available for a transaction, they will choose the one that requires minimum data transmission. Then, the sharding is also self-optimized.

References

1. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. (2008)
2. Lamport, L., Shostak, R., Pease, M.: The byzantine generals problem. ACM Transactions on Programming Languages and Systems (TOPLAS) **4**(3) (1982) 382–401
3. Fischer, M.J., Lynch, N.A., Paterson, M.S.: Impossibility of distributed consensus with one faulty process. J. ACM **32**(2) (April 1985) 374–382

4. Bracha, G.: Asynchronous byzantine agreement protocols. Information and Computation **75**(2) (1987) 130–143
5. Castro, M., Liskov, B., et al.: Practical byzantine fault tolerance. In: OSDI. Volume 99. (1999) 173–186
6. Luu, L., Narayanan, V., Zheng, C., Baweja, K., Gilbert, S., Saxena, P.: A secure sharding protocol for open blockchains. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. CCS '16, New York, NY, USA, ACM (2016) 17–30
7. Guerraoui, R., Knežević, N., Quéma, V., Vukolić, M.: The next 700 BFT protocols. In: Proceedings of the 5th European conference on Computer systems, ACM (2010) 363–376
8. Kokoris-Kogias, E., Jovanovic, P., Gailly, N., Khoffi, I., Gasser, L., Ford, B.: Enhancing bitcoin security and performance with strong consistency via collective signing. CoRR **abs/1602.06997** (2016)
9. Croman, K., Decker, C., Eyal, I., Gencer, A.E., Juels, A., Kosba, A., Miller, A., Saxena, P., Shi, E., Sirer, E.G., et al.: On scaling decentralized blockchains. In: International Conference on Financial Cryptography and Data Security, Springer (2016) 106–125
10. Eyal, I., Gencer, A.E., Sirer, E.G., Van Renesse, R.: Bitcoin-NG: A scalable blockchain protocol. In: 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16), USENIX Association (2016) 45–59
11. Poon, J., Dryja, T.: The bitcoin lightning network: Scalable off-chain instant payments. Technical Report (draft) (2015)
12. Micali, S.: ALGORAND: the efficient and democratic ledger. CoRR **abs/1607.01341** (2016)
13. Pass, R., Shi, E.: Hybrid consensus: Efficient consensus in the permissionless model (2016)
14. Bentov, I., Pass, R., Shi, E.: The sleepy model of consensus. IACR Cryptology ePrint Archive **2016** (2016) 918
15. Buterin, V.: On sharding blockchains. Sharding FAQ (2017) Available at `https://github.com/ethereum/wiki/wiki/Sharding-FAQ`.
16. Kokoris-Kogias, E., Jovanovic, P., Gasser, L., Gailly, N., Ford, B.: Omniledger: A secure, scale-out, decentralized ledger.
17. Luu, L., Narayanan, V., Baweja, K., Zheng, C., Gilbert, S., Saxena, P.: SCP: A computationally-scalable byzantine consensus protocol for blockchains. IACR Cryptology ePrint Archive **2015** (2015) 1168
18. Miller, A., Xia, Y., Croman, K., Shi, E., Song, D.: The honey badger of BFT protocols. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, ACM (2016) 31–42

A Proofs

A.1 Proof of Theorem 1

Proof. By the definition of honest nodes, i and j will add the transaction messages to their chains. The messages will be included in a correct piece before some time t because of the conditions of the consensus scheme. Then, the correct and complete proofs of this transactions can be obtained by honest nodes since i and j are honest, which suggests that the outcome of the validation scheme will not be "undecided" and complete the proof.

A.2 Proof of Lemma 1

Proof. The proof follows from the definition of the CM and the consensus scheme. By the definition of the CM, the information of the position, order, and the digests of the content of the CPs are included in the CMs. Moreover, the CMs with incorrect information or the ones that attempts to create forks in CPs are discarded during the consensus process. As a result, the consensus $\mathcal{CON}(r)$ fixes the position, order, and the content of the CPs. Then, this lemma is established if the two views $I(r)$ and $I'(r)$ have the same consensus results.

A.3 Proof of Lemma 2

Proof. We prove this lemma by contradiction.

Assume there exists another correct piece of blockchain $\mathcal{B}' = \{B_i(k'), B_i(k'+1), \ldots, B_i(\ell')\}$, $k' \neq k$ or $\ell' \neq \ell$ that $\mathcal{B}' \neq \mathcal{B}$ in a view $I'(r')$. By the definitions of a correct piece, we know that $B_i(k), B_i(k'), B_i(\ell), B_i(\ell')$ are all CPs included in $\mathcal{CON}(r)$. Moreover, by Lemma 1, we have $\ell = \ell', B_i(\ell) = B_i(\ell')$ if $k = k'$ and $k = k', B_i(k) = B_i(k')$ if $\ell = \ell'$.

Then, since both $\mathcal{B}$ and $\mathcal{B}'$ are correct, all digests of all blocks in these pieces should be correct. Then, since $\mathcal{B} \neq \mathcal{B}'$, there must exists two blocks $B_i(n) \neq B'_i(n)$ such that $B_i(n+1) = B'_i(n+1)$, which suggests $H(B_i(n)) = H(B'_i(n))$. This contradicts the fact that the digests are collision free.

A.4 Proof of Theorem 3

Proof. We firstly proof the following statement: If $V_u(tr(i \to j, s)) =$ *validated* and all of its sources are valid, then $tr(i \to j, s)$ is valid. If $V_u(tr(i \to j, s)) =$ *falsificated* and none of its source are falsificated, then $tr(i \to j, s)$ is invalid. We prove this by contradiction.

Firstly, assume that there exist an invalid transaction $tr(i \to j, s)$ with valid sources and it is validated by an honest node u. Then, the correct and complete proofs of this transaction must have been collected by u. Furthermore, it must have passed the validation process. Then, since the steps in validation process (Definition 10) are precisely the validity conditions (Definition 8) except the **Correct Chains**, which has already been guaranteed by the proof collection.

By Definition 8, there exists an observation of this network in which all the validity conditions for this transactions are met, thus this transaction is valid. This contradict our assumption.

Then, we assume that there exists a valid transaction $tr(i \to j, s)$ with no falsificated source and it is falsificated by an honest node u. By the definition of the validation scheme, node u must have collected all the proofs for this transaction, which includes all the proofs for the sources. Hence all of its source are validated, thus are valid. Then, at least one of the items in the Definition 10 except the **Validated Sources** is violated, which suggests that the validity conditions (Definition 8) are not fulfilled. Then, by Theorem 2, the proofs are consistent. Hence, there does not exist an observation by an honest node in which all conditions are satisfied. By Definition 8, this transaction is invalid, which contradicts our assumption.

The theorem is thus proved by recursively using the proved statement on the transactions and their sources since the validity of the initial balance can be checked with $\mathcal{CON}(r)$.

B Analysis of the Throughput

Here we lower bound the rate of the valid transactions in our system. For the sake of easier comprehension, we assume that the transactions rate, communication capacity, and computation capacity are uniform for all nodes and all time and the adversaries do not spam invalid transactions.

We consider a subset of node in the network $\mathcal{G}, |\mathcal{G}| = g \leq N$ which only do transactions with the nodes in the subset. Assume that each chain grows with a rate of R messages/second and the duration of a consensus round is T. The amount of messages generated by this subset of nodes in a round is RgT, which can be divided into two parts: valid transactions and invalid transactions. Since the honest nodes only make transactions that they can validate, the amount of valid transactions is at least $R_v gT$ where R_v is the validation rate. The invalid transaction can only be made by adversaries. Since they do not spam, we have the amount of the invalid transactions equals to $R_a gT$ where $R_a = O(R_v)$. Then, we have $R = R_v + R_a = O(R_v)$.

Let us analyze the duration that a node needs to validate all transactions that it makes in a round. For the proof collection, it needs no more than all chains in $\mathcal{G}$, which requires data transmissions with no more than an amount of RgT messages since the proof collections is incremental, i.e., only the newly generated parts of the chains are needed. The proofs are collected based on point-to-point transmissions. Each node broadcasts its chain at a rate of C_{comm}/g, where C_{comm} is the communication capacity (message/second) of the nodes. The collection rate is then C_{comm} since nodes broadcast their chains simultaneously in different channels. Hence, we have the duration of proof collection $t_p \leq \frac{RgT}{C_{comm}}$.

For validation, in the worst case, all of these transactions need to be validated, which requires duration $t_v \leq \frac{RgT}{C_{comp}}$, where C_{comp} is the computation capacity (message/second). By basic queuing theory, we should have $t_p + t_v = T$.

Then, since honest nodes only make transactions that they can validate and the in all the RgT messages, the expected invalid message Since all validated transactions are valid (Theorem 3), combining all the inequalities above, we have a lower bound on the rate of the valid transactions for each node $R_v \geq \Omega(\frac{C}{g})$, where $C = \frac{C_{comm}C_{comp}}{C_{comm}+C_{comp}}$. Then, the throughput of this group is lower bounded by $\Omega(C)$ since a group has g nodes that can simultaneously make transactions.

This lower bound suggests that the throughput in any separate group of nodes in the network is completely independent of the rest of the network and only depends on the communication and computation capacity of the nodes in that group. This is an ideal property to have for a blockchain system since the throughput is no longer limited by the throughput of the consensus algorithm. In the best case that nodes are paired and only do transaction with each other, we achieve a throughput of $O(CN)$. In the worst case that all nodes make transactions with all other nodes, we achieve a throughput of $O(C)$.