Implicit Consensus: Blockchain with Unbounded Throughput

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

1 **Cyber Security Group**
Department of Intelligent Systems

2 **Distributed Systems Group**
Department of Software Engineering
Delft University of Technology
Mekelweg 4, Delft, The Netherlands
z.ren@tudelft.nl

Abstract

(This paper has been submitted to DISC 2017 on May 8th, 2017.)

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, reliability, storage requirement, and privacy. 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 agreement is not on the transactions, but on a special type of blocks called Check Points that are used to validate individual transactions. Our system achieves superior performance over all existing blockchain techniques in multiple aspects with equivalent reliability. Most remarkably, our system achieves unbounded throughput which is by far the best throughput achieved by any blockchain technique.

1998 ACM Subject Classification C.2.4 Distributed Systems

Keywords and phrases Blockchain, Distributed Ledger, Consensus Algorithm, Byzantine Fault Tolerance

Digital Object Identifier 10.4230/LIPIcs.xxx.yyy.p

1 Introduction

Blockchain, introduced and firstly applied in Bitcoin [15], 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 trade-off between the throughput and the scalability, which is caused by this limitation of the current consensus algorithms. Reaching consensus for distributed nodes in an unsafe network, commonly known as the Byzantine fault tolerance (BFT) problem [10], has been extensively studied for over 30 years. Traditional approaches allow honest nodes to reach consensus if the number of the adversaries is less than 1/3 of the total population [1, 3]. However, these algorithms requires at least $O(N^2)$ communications for a network with N nodes, thus do not scale well into large networks with thousands of nodes. Bitcoin and its proof-of-work (POW) scheme opened up a new horizon on this problem. In Bitcoin, incentives are introduced to put cost on malicious behaviors, which limits the capability of the adversaries. It thus allows reaching consensus in a network with thousands of nodes, as long as the computational power of the malicious nodes is less than 1/4 of the total computational capability [6]. However, POW based public blockchains like Bitcoin and Ethereum [19] have rather low throughputs, e.g., Bitcoin can have at most seven transactions per second. This throughput is incomparable to the throughput of thousands of transactions per second achieved by traditional BFT algorithms [4]. Currently, there does not exist a blockchain system which achieves high throughput in a network with thousands of nodes due to the limitation in the consensus algorithm.

1.2 State-of-the-Art

For POW, many approaches have been proposed to improve the throughput of POW based blockchains, e.g., [5, 18], which all have costs in security and reliability. In general POW based schemes has fundamental limitations in throughput to achieve an acceptable level of security given the current network infrastructure [4]. Furthermore, POW schemes are very expensive to deploy in the sense that it consumes a huge amount of energy to be even close to the Bitcoin level of security.

On the other hand, the BFT algorithms still suffer from the the scalability issue which restricts it to a small network (maximum around a hundred nodes) [3, 8, 11, 14]. The solution is using BFT scheme on either a small set of nodes permissioned in advance or on a large network with a certain hierarchical structure. The former case has been widely considered in practical consortium chains like [2, 9], which achieves remarkably good performance. In the latter case, using BFT on a large network is unfavorable since BFT algorithms do not scale well. Hence, traditional BFT schemes are combined with group partition schemes which guarantee the ratio of the adversaries in each level of the structure should not exceed 1/3, e.g., [12, 16]. Besides BFT consensus and Bitcoin consensus which consider global consensus, blockchains like [13, 17] consider partial consensus which improves the throughput at the costs of reliability and security.

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 level of reliability on validated transactions comparing to traditional BFT based schemes. We argue that compared to the classical consensus model, the implicit consensus is closer to real-life scenario. Let us consider the differences in making a contract with classical blockchains and in real life. With classical blockchains, the contract is revealed to all nodes and the validity is checked by all nodes. As a result, the validity of this contract is undeniable and unforgeable. On the other hand, in real life, the contract is only revealed to the related parties. The authority, e.g., court system, are not necessarily aware of the existence and the details of this contract. However, if the related parties have a disagreement upon a certain contract, they can provide proofs to the authority and ask for aid. Then, the honest parties can reach agreement upon the authority's judge. In other words, the consensus is not on individual contract, but on the fact that the authenticity and validity of the contract can be checked by the authority, and they should agree with the authority.

Now, we give the definition of the implicit consensus. Firstly, let us first consider that some type of consensus (BFT consensus or Bitcoin consensus), expressed as the *Explicit Consensus*, is reached upon a transaction. It suggests the following.

- **Explicit Consensus:** All honest nodes have agreed that the transaction is valid.

This type of consensus can actually be reformulated into two parts.

- **Consensus 1:** Each honest node has agreed that the transaction is valid.
- **Consensus 2:** Each honest node has agreed that all other honest nodes have also agreed that the transaction is valid.

However, a third consensus is hidden in here, which we call the *Implicit Consensus*.

- **Implicit Consensus:** All honest nodes have agreed that the validity of any transaction is verifiable, unambiguous, and unforgeable.

Clearly, the implicit consensus is equivalent to the **Consensus 2** under the condition of **Consensus 1**. This is exactly the consensus in the real-life scenario with authority replaced by blockchains, i.e., all honest parties agree that the authority will make correct and fair judgment without necessarily knowing the validity in advance. In other words, in the real-life scenario, the standalone implicit consensus is enough for the reliability of the system.

1.4 Structure of Our System

We consider an asynchronous network with N nodes and $f \leq \lfloor N/3 \rfloor$ 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 itself are recorded. Besides the blocks that consists of transactions which are called *Transaction Blocks (TBs)*, another type of special blocks called *Check Points (CPs)* are 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 [3, 8, 12, 14] 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 parts, 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 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 in the aspects of throughput, reliability, and storage requirement in Section 4. In Section 5, we conclude our work and give recommendations for the future 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 $\mathcal{B}_i, i \in \{1, 2, \dots, N\}$ [1]. A blockchain $\mathcal{B}_i$ is then defined as an ordered set of blocks $\{B_i(1), B_i(2), \dots\}$, in which each block contains a digest of its previous block, i.e., block $B_i(j), j > 1$ will contains $H(B_i(j-1))$. 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[2]. 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.

[1] 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.

[2] 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.

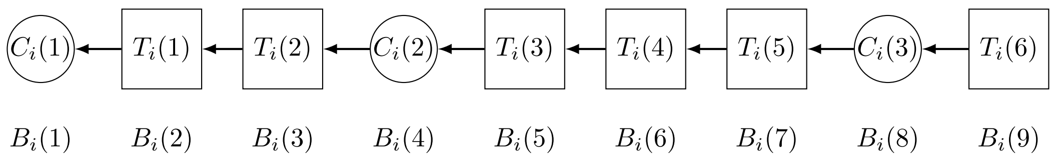

Figure 1 Example of a blockchain of node i with six TBs and three CPs.

2.2.1 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 $[H(B_i(j-1)), t_i(k, 1), t_i(k, 2), \ldots, t_i(k, M)]$.

▶ **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.

2.2.2 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 previous 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 following conditions can be used.

- **Agreement:** If an honest node sends a CM, then all honest nodes agree with this CM in the consensus.
- **Termination:** If all honest nodes send their CMs, then they reaches some consensus eventually.
- **Correctness:** If a CM is in the consensus, then it must have been send by some node.

Some of the choices are [8, 12, 14], which tolerant less than $\lfloor \frac{N}{3} \rfloor$ malicious nodes. Here, the *honest* and *malicious* nodes are defined as the following.

▶ **Definition 5** (Honest Node[3]). 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.

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*.

[3] 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.

▶ **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.

2.4.1 Validity and Conditions for Validation

In general, the validity of a transactions should be a global and unambiguous property that 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 all of the above conditions, it implies that the validity is verifiable, unforgeable, and consistent with our validation scheme, i.e., the 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 $tr(i \rightarrow j, s)$, denoted by a function $V_u(tr(i \rightarrow j, s))$.

2.4.2 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 $tr(i \rightarrow j, s)$ consists of the following.

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

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 $tr(i \rightarrow j, s)$ 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.

2.4.3 Validation Process

▶ **Definition 10** (Validation Process for a Transaction). A validation process of a transaction $tr(i \rightarrow j, s)$ 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 $\mathcal{S}(tr') \cap \mathcal{S}(tr(i \rightarrow j, s)) \neq \emptyset$.
4. **Validated Sources:** All the source transactions of $tr(i \rightarrow j, s)$ 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 [1, 3, 14]. Here, we prove the validation scheme satisfies the conditions of correctness and liveness.

3.1 Liveness (of the Honest Nodes)

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 11** (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.*[4]

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. ◀

Note that Theorem 11 does not guarantee that all transactions are eventually validated or falsificated, i.e., some of the transactions made by malicious node cannot 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 $\mathcal{B}_i, i \in \{1, 2, \dots, N\}$, 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 12** (View). A **view** in consensus round r denoted by $I(r)$ is a set of blockchains $\mathcal{B}_i, i \in 1, 2, \dots, N$ with $\mathcal{CON}(r)$ as its consensus results.

[4] We show that they will only be validated in Theorem 15.

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 13** (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.*

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. ◀

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

Proof. We prove this lemma by contradiction.

Assume there exists another correct piece of blockchain $\mathcal{B}' = \{B_i(k'), B_i(k'+1), \dots, B_i(\ell')\}$, $k' \neq k$ 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 13, 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. ◀

By Lemma 14, since the proofs of a transactions are simply a collection of pieces, we directly have the following theorem.

▶ **Theorem 15** (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 16** (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.*

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 $tr(i \rightarrow j, s)$ 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 15, 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 $\mathcal{CON}(r)$. ◀

4 Performance

In this section, we compare the performance of our system to other blockchain systems in the aspects of throughput, reliability, and storage requirement.

4.1 Throughput

By design, the throughput of our system is independent of the consensus scheme since the creation of the transactions and TBs are completely independent of the consensus round. In other words, each node can create as much transactions as they could with no guarantee on validity. Hence, a fair throughput comparison should be between the rate of the 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. 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 16), 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 $O(C)$.

Since this throughput is significantly different from the *scalable* throughput achieved and stated in other works [4, 12, 14], we claim that an “unbounded” throughput is achieved by our system, In most literature, the term “scalable” is used to compare with the “unscalable” throughput of classical BFT algorithms, which have communication costs of at least $O(N^2)$ per transaction. Then, a scalable blockchain usually suggests a blockchain with a consensus algorithm with a communication cost of $O(N)$. Hence, the transaction rate of a scalable blockchain system is upper bounded by $O(C)$ even if the transactions are very uniformly distributed. So far as we know, our scheme is the first and only blockchain system which achieves unbounded performance, i.e., the transaction rate in between $O(C)$ and $O(CN)$.

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 [8, 12, 14, 16] since our scheme is not restricted to a specific BFT algorithm.

4.2 Reliability

In many existing blockchain systems, the throughput improvement is achieved by sacrificing either the decentralization [2, 9] or the reliability [5, 13]. In our system, as discussed in Subsection 1.3, going from explicit consensus to implicit consensus does not compromise on either the decentralization or the reliability of the network. Firstly, it is clear that our system is completely decentralized. Then, it has been shown that a transaction is validated by an honest node, it is as reliable as a valid transactions in classical blockchain systems with standard reliability assumptions. That is to say, a malicious node cannot convince an honest node that an invalid transaction is valid unless it controls more than $\lfloor N/3 \rfloor$ nodes and/or it breaks the hash function or the asymmetric encryption.

Certainly, as discussed in Subsection 3.1, the price we pay is that 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 that we will briefly discuss in Subsection 5.2. 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 the honest nodes to identify the 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. As a result, the proofs of a transaction will also propagate with the transaction itself. In other words, the more important the proofs are, the more copies there will be in the network.

4.3 Storage Requirement

Storage requirement is another important aspect of the blockchain. Many blockchain systems require all nodes to store the whole chain by design, which would cause trouble in some storage limited applications. Most of the current blockchain systems allow *lightweight nodes* (the node which only store transactions that related to itself) at a cost of reliability since traditionally, the reliability depends on all nodes knowing and validating all transactions. In our system, on the other hand, all nodes are already semi-lightweight nodes by design since all nodes are only required to store the transactions that related to it self and all the proofs that support these transactions. The nodes are not completely lightweight since the proofs require all the blocks that came before the related transactions. However, the storage requirement is still significantly low comparing to other blockchain systems like Bitcoin, in which all transactions of the whole network are required to be stored in each node.

5 Conclusions and Future Works

In this paper, we proposed a value-exchange blockchain system with a novel consensus model, namely implicit consensus. There are three main difference from our system to other blockchain systems.

- Each node has its own blockchain.
- The consensus is not on individual transactions, but on some special blocks called *Check Points*.
- Not all transactions on the chains are valid.

Our system achieves significant improvements in throughput and other important aspects comparing to all other blockchains techniques.

Besides the benefits in performance, our system is also very flexible due to its distinctive structure. Many extensions and mechanisms can be added to achieve additional functionalities for various applications without the notorious risk of hard-forking (the splitting of the network caused by an incompatible rule updated by part of the network). The reason is that the consensus is only on the CPs. Hence, a change in the content will only result in failures in validations but not failures in reaching consensus. We end by listing a few of the extensions which we think are the most interesting ones for future research.

5.1 Generalization

In our system we made the assumption of the two-party transaction for easier comprehension and presentation. In fact, allowing multi-party transaction will not result in any significant change to our system. Similarly, the agreements other than value exchange could also be compatible with our system, e.g., smart contracts. However, the validation scheme should be redesigned so that there are sufficient interest for the nodes to validate the chains of the other nodes.

5.2 Reputation Scheme

Besides valid transactions, our system keeps the traces of the invalid transactions as well. This fundamental difference gives room for reputation or blacklist schemes. By collecting the information like the quantity, the type, and the patterns of the invalid transactions of individual nodes, the nodes with malicious intentions can be detected and the reputation of each node can be computed.

5.3 Privacy Enhancement

The privacy issue is one of the biggest concerns for the application of blockchain in the industry since the reliability depends on the transparency and accessibility of the data in classical blockchain systems. The structure of our system makes it easier to design a privacy enhanced validation scheme. More precisely, since all validations are based on point-to-point communications, advanced cryptographic techniques like Zero-knowledge Proof [7] can be used to hide all information in the collected proofs while guarantees the correctness of the validation.

5.4 Centralization

Centralization used to be undesirable in the context of blockchain. However, many blockchain systems like [2, 9] introduce a certain level of centralization to provide efficiency, flexibility, and privacy. Nowadays, the level of centralization is considered by many researchers as a trade-off to reliability and security, and is highly application dependent. The structure of our system supports various levels of centralization, which could meet the requirements of various applications. The reason is that each node can decide their own strategy in validation. As a result, groups with various levels of centralization can appear in our system simultaneously. For example, let us assume that there is a subgroup of nodes in the network that only allow a certain node to collect their chains and trust its validation result. Then, this subgroup becomes a centralized system with no impact on the rest of the network, which might still be decentralized.

5.5 Merging of Multiple Networks

Our system is essentially permissioned. However, unlike other permissioned blockchains, new nodes can easily join the network by simply adding another consensus scheme with another set of CPs without making any change on the existing chains.

References

1 Gabriel Bracha. Asynchronous byzantine agreement protocols. *Information and Computation*, 75(2):130–143, 1987.

2 Christian Cachin. Architecture of the hyperledger blockchain fabric. In *Workshop on Distributed Cryptocurrencies and Consensus Ledgers*, 2016. URL: `https://www.zurich.ibm.com/dccl/papers/cachin_dccl.pdf`.

3 Miguel Castro, Barbara Liskov, et al. Practical byzantine fault tolerance. In *OSDI*, volume 99, pages 173–186, 1999.

4 Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gün Sirer, et al. On scaling decentralized blockchains. In *International Conference on Financial Cryptography and Data Security*, pages 106–125. Springer, 2016.

5 Ittay Eyal, Adem Efe Gencer, Emin Gun Sirer, and Robbert Van Renesse. Bitcoin-NG: A scalable blockchain protocol. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, pages 45–59. USENIX Association, 2016.

6 Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *International Conference on Financial Cryptography and Data Security*, pages 436–454. Springer, 2014.

7 Oded Goldreich. *Foundations of cryptography: volume 2, basic applications.* Cambridge university press, 2009.

8 Rachid Guerraoui, Nikola Knežević, Vivien Quéma, and Marko Vukolić. The next 700 BFT protocols. In *Proceedings of the 5th European conference on Computer systems*, pages 363–376. ACM, 2010.

9 Mike Hearn. Corda: A distributed ledger. 2016. URL: `http://block.academy/researches/corda-technical-whitepaper.pdf`.

10 Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401, 1982.

11 Shengyun Liu, Paolo Viotti, Christian Cachin, Vivien Quéma, and Marko Vukolic. XFT: Practical fault tolerance beyond crashes. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, pages 485–500, 2016. URL: `http://dl.acm.org/citation.cfm?id=3026877.3026915`.

12 Loi Luu, Viswesh Narayanan, Kunal Baweja, Chaodong Zheng, Seth Gilbert, and Prateek Saxena. SCP: A computationally-scalable byzantine consensus protocol for blockchains. *IACR Cryptology ePrint Archive*, 2015:1168, 2015.

13 David Mazieres. The stellar consensus protocol: A federated model for internet-level consensus. *Stellar Development Foundation*, 2015.

14 Andrew Miller, Yu Xia, Kyle Croman, Elaine Shi, and Dawn Song. The honey badger of BFT protocols. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 31–42. ACM, 2016.

15 Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008. URL: `https://bitcoin.org/bitcoin.pdf`.

16 Rafael Pass and Elaine Shi. Hybrid consensus: Efficient consensus in the permissionless model, 2016. URL: `http://eprint.iacr.org/2016/917.pdf`.

17 Serguei Popov. The tangle. 2014. URL: `https://iota.org/IOTA_Whitepaper.pdf`.

18 Yonatan Sompolinsky and Aviv Zohar. Secure high-rate transaction processing in bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 507–527. Springer, 2015.

19 Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper*, 151, 2014. URL: `http://gavwood.com/paper.pdf`.