A Scale-out Blockchain for Value Transfer with Spontaneous Sharding

2 authors:

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

A Scale-out Blockchain for Value Transfer with Spontaneous Sharding

Zhijie Ren*, Kelong Cong†, Taico V. Aerts*, Bart. A. P. de Jonge*, Alejandro F. Morais* and Zekeriya Erkin*
*Faculty of Electrical Engineering, Mathematics, and Computer Science
Delft University of Technology, Van Mourik Broekmanweg 6, Delft, the Netherlands, 2628XE
Email: z.ren@tudelft.nl
†Ecole Polytechnique Fédérale de Lausanne (EPFL), CH-1015 Lausanne, Switzerland
Email: kelong.cong@epfl.ch

Abstract—**Bitcoin, as well as many of its successors, require the whole transaction record to be reliably acquired by all nodes to prevent double-spending. Recently, many blockchains have been proposed to achieve scale-out throughput by letting nodes only acquire a fraction of the whole transaction set. However, these schemes, e.g., sharding and off-chain techniques, suffer from a degradation in decentralization or the capacity of fault tolerance.**

In this paper, we show that the complete set of transactions is not a necessity for the prevention of double-spending if the properties of value transfers is fully explored. In other words, we show that a value-transfer ledger like Bitcoin has the potential to scale-out by its nature without sacrificing security or decentralization. Firstly, we give a formal definition for the value-transfer ledger and its distinct features from a generic database. Then, we introduce an off-chain based scheme with a shared main chain for consensus and an individual chain for each node for recording transactions. A locally executable validation scheme is proposed with uncompromising validity and consistency. A beneficial consequence of our design is that nodes will spontaneously try to reduce their transmission cost by only providing the transactions needed to show that their transactions are double-spending-proof. As a result, the network is sharded as each node only acquires part of the transaction record and a scale-out throughput could be achieved, which we call "spontaneous sharding".

I. Introduction

Blockchain technology, made popular by Bitcoin [1], can be described as an append-only database maintained by distributed nodes instead of central authorities. One of the most well-known applications of blockchain technology is cryptocurrency, in which the blockchain is in the form of a distributed ledger, i.e., the data is transactions which are records of value transfers, called transactions, between nodes. The most crucial part of a distributed ledger for value transfer is the prevention of double-spending, which is achieved by consensus algorithms that guarantee all honest nodes in the network keep a consistent ledger of all valid transactions. The consensus algorithm can be divided into two categories, the Nakamoto-like consensus algorithms such as Proof-of-Work (POW) [1] or Proof-of-Stake (POS) [2], [3] and Byzantine fault tolerance (BFT) consensus algorithms such as PBFT [4]. For distributed ledger type of blockchain, most of the consensus algorithms effectively achieve the following conditions.

- **Agreement (Consistency):** Two honest nodes should not have disagreement on the validity of a transaction.
- **Validity (Correctness):** Invalid transactions cannot be validated by honest nodes.
- **Termination (Liveness):** All transactions will be eventually known by all honest nodes.

Strictly speaking, the above conditions are not achievable in asynchronous networks [5], [6]. However, by slightly compromising either asynchronous [4] or deterministic conditions for termination [7], [8], the above-mentioned conditions can be achieved in practical asynchronous network. Blockchains with both Nakamoto-like consensus [9] and BFT consensus can have scalable throughput, i.e., the communication cost per transaction (CCPT) is restricted to $O(N)$, where N is the number of nodes in the network. Various consensus algorithms could achieve consensus with $O(N)$ complexity.

- **Improved BFT algorithms:** Traditional BFT algorithms like [4], [7], [8] have $O(N^2)$ CCPT. However, many recent BFT algorithms like [10], [11], [12], [13] achieve $O(N)$ CCPT by either packing up transactions or opportunistically running a much simpler scheme with traditional schemes as the back-up for the worst scenario.
- **Nakamoto-like consensus:** The POW scheme in Bitcoin introduced a game theoretical aspect to this problem. Then, instead of restricting the number of faulty nodes, an assumption is put on the rationality of nodes in the network. However, some early POW or POS based schemes have limitation in the transaction rate to meet the synchronous requirements [14]. With this problem solved in novel algorithms like [3], [15], [16], [17], $O(N)$ CCPT is feasible in Nakamoto-like consensus.

A. Scalability of Blockchain

The most crucial problem in a decentralized value-transfer system is double-spending, which could be prevented when all nodes have a consistent record of all transactions. Then, $O(N)$ CCPT is required for all transactions. Blockchains with $O(N)$ CCPT are commonly referred as "scalable" blockchains since their throughput will not decrease (or increase) with the number of nodes and the computation and communication capacities in the network.

1) Scale-out Blockchain Solutions: Recently, several solutions have been proposed to achieve $o(N)$ CCPT, sometimes referred as "scale-out" throughput as the throughput will increase as N grows, by reducing the number of validators and recordkeepers for each transaction. In other words, the termination property is compromised, i.e., a transaction is not necessarily known to or validated by the whole network, but a part of it. Here, we introduce three types of such schemes.

- **Off-chain Solutions:** This type of approach are mostly associated with some existing blockchain systems as the main chain. Each node holds their transactions locally, sometimes referred as "off-chain", and only sends a description or the eventual outcome of these transactions to the "main chain", referred as "on-chain". Since there is no guarantee on the validity of the "off-chain" transactions, either validation nodes are introduced to validate and endorse these transactions [18], [19] or economical deposit should be provided for the transactions [20], [21]. In both cases, the validity condition is compromised due to centralization or the economical constraint.
- **Directed Acyclic Graph (DAG) Solutions:** In another type of approach, we call DAG solutions, the transactions are not structured in a chain, but in a DAG [22], [23], [24]. The validity is dependent on the (directly or indirectly) outgoing edges of the transaction, which represents the nodes that have validated it. A scale-out throughput can be achieved if the acquirement of the complete graph is not obligated for all nodes [1]. Then, the validity of the transactions is compromised due to its dependency on the validators.
- **Sharding Solutions:** Recently, sharding solutions, which artificially divide the network, have been widely studied and discussed [25], [26], [27], [28]. They include schemes that fairly and randomly divide the network into small shards with vanishing probability of any shard having an overwhelming number of adversaries. Hence, the BFT consensus algorithm is run only within the shards and the CCPT is then $O(g^2)$ ($O(g)$ if scalable BFT algorithms are used) where g is the size of the shard. However, the validity condition is also compromised in the sense that the sharding is only feasible when the ratio of adversaries in the network is small. Moreover, according to our knowledge none of the existing sharding schemes proved $g = o(N)$, which is the condition for scale-out throughput.

2) Problem Statement: It seems to be infeasible to achieve scale-out performance with the same level of security or decentralization as Bitcoin or blockchains using classical BFT algorithms. This does not come as a surprise since intuitively, double-spending can only be prevented with global consensus. This problem severely hampers the mainstream adoption of blockchain system since the security and trustworthiness of the blockchain system grows with the size of the network and the decentralization level. As a result, a trilemma is formed among throughput, security, and decentralization as also stated in [25], [27]. However, at the meantime, traditional BFT algorithms could reach consensus on any type of message, which is redundant for many blockchain systems since messages in Bitcoin and cryptocurrencies are "transactions" which represent value transfers.

This leads to the research questions considered by this paper:

- What is the key functionality/features of the value-transfer blockchains?
- Can we use these features to design a scale-out blockchain system to achieve the functionalities of value transfer without sacrificing reliability or decentralization?

B. Overview of Our Solution

Our solution gives an answer to the above questions. By exploring the features of value transfers which have not yet been used by other blockchain systems, we propose a blockchain system with a very simple structure to achieve scale-out throughput.

1) Value-Transfer Ledgers: Most of the aforementioned blockchain systems are decentralized solutions for value transfers and focus on reaching BFT consensus on transactions to prevent double-spending. However, traditional BFT consensus algorithms are generic and achieve BFT consensus regardless of the message type. In Bitcoin and other blockchain systems using Nakamoto-like consensus [9], some realistic interpretation of transactions is used and the notion of rational behavior is introduced: rational issuers of transactions are interested in proving the validity of their transactions and keeping synchronized with other nodes. As a result, they either take effort themselves by mining or hire other nodes by paying transaction fee to submit their transactions to a blockchain which reaches BFT consensus. In this paper, we take one step further to formally define the features for value transfers in the Value-Transfer Ledgers (VTL) model, i.e.,

- rational senders of the transactions should take effort to prove the authenticity of the transaction to the receiver;
- rational receivers should check the authenticity of a transaction while receiving it;
- a rational receiver will not care about the authenticity of other transactions unless they have an impact on their received transactions.

With these features, we propose a system that minimizes the redundancy of reaching BFT consensus on the transactions as if they are generic data and allows secure and reliable value transfers in a full decentralized fashion.

2) Our System: Our system has an off-chain structure, which contains individual chains for nodes to record their own transactions and a main chain for the consensus of the abstracts of their chains, i.e., provides a shared global state. Further, a locally executable validation function is proposed to have correct and consistent validation results upon all transactions. Besides a validation function, the crucial part of a valid validation scheme is that all honest nodes should also have a consistent observation of the transactions. In our system, we employ the aforementioned features of value transfers

[1]In fact, in [22], [23], [24], the complete graph is required to prevent double-spending for their applications. Hence, they are not scale-out schemes, although a DAG scheme designed similarly could scale-out for some other applications.

to achieved an alternation: instead of letting all nodes have consistent observation on all transactions, we guarantee that all nodes *that want to know the validity of a transactions* will have consistent observation on all transactions that *have impact on the validity of that transaction*. We also prove that this alternation is enough to have a valid system for value transfers.

3) Spontaneous Sharding: Moreover, the most innovative result in this paper is *spontaneous sharding*, which is a natural and direct consequence of using our system for value transfers. Generally speaking, in value-transfer systems, the values are passed from one node to another. In our system, for each piece of value, a proof is associated with it and the size of the proof grows with the number of nodes that it has been passed to. Then, since the sender could choose the source of his fund for the transactions, e.g., in Bitcoin, a node could choose from several of his unspent transaction outputs, rational nodes will choose the pieces of value with the minimum size of proof for the sake of the transmission cost. As a result, nodes will tend to cycle the value in small shards rather than the whole network. In other words, the network is sharded by the nature of the system without sacrificing either security or decentralization.

C. Main Contributions

The main contributions of this paper are the following.

- We formally define the VTL and distinguish it from other types of ledgers and databases.[2]
- We propose an off-chain based blockchain system that prevents double-spending without sacrificing either security or decentralization. In particular, our consensus algorithm achieves uncompromised agreement and validity conditions of the BFT consensus in VTL model.
- We prove that our system is a valid VTL system. In other words, although our system do not guarantee BFT for generic types of data, we guarantee that if all nodes have interest in their values in the system and behave rationally, the valid transactions in our system are double-spending-proof.
- It is shown that the CCPT of our system is upper bounded by $O(N)$, which suggests scalable throughput. Moreover, we show that our system could achieve scale-out throughput via spontaneous sharding in several scenarios.

D. Content of This Paper

This paper is organized as follows. In Section II, we formally introduce the VTL model and assumptions. In Section III, we introduce our system and prove the correctness of this system in VTL model. We analyze the performance of our scheme and introduce the concept of spontaneous sharding which results in scale-out throughput in Section IV. In Section V, we conclude our paper with possible topics for further exploration.

[2]A slightly similar idea have been raised in [29] without a formal defined model or details for feasible schemes.

II. Model

In this paper, we emphasize on our novelties and contributions by showing that our system provides the minimum functionalities for value transfers. These functionalities can be used as building blocks for more generic VTL systems. Hence, some other elements are simplified to the most comprehensive level, e.g.,

- We consider every node holding some initial value. The mining of new coins is not considered.
- Transactions are defined similarly to Bitcoin, namely the Unspent Transaction Output as input (UTXO) structure. We assume a transaction has only one sender and one receiver.
- We consider a weak asynchronous network with $f \leq \lfloor \frac{N-1}{3} \rfloor$ Byzantine adversaries, just as the one used in [4], so that PBFT can be straightforwardly applied. Note that this assumption is solely made for easy comprehension of our system. The same framework proposed in our system can be plugged into any permissioned or permissionless blockchain or consensus algorithm that achieves global BFT consensus on all transactions.
- We assume that there exists an unbreakable hash function $Y = H(X)$ and a digital signature scheme $Y = Sig_i(X)$ based on the public-private key pairs where node i is the signer.

A. Network Model

We consider a weak asynchronous network of N nodes in which the message delay does not increase indefinitely as described in [4]. Each node holds some initial value that could be transacted with others. We assume that there are $f \leq \lfloor \frac{N-1}{3} \rfloor$ Byzantine adversaries and we have the following definitions for honest nodes and adversaries.

Definition 1 (Honest nodes and Adversaries)**. Honest nodes** will follow the schemes of the system. **Adversaries** can behave arbitrarily.

The network is assumed to be permissioned, i.e., the nodes are known to each other by their identities $n \in \{1, 2, \dots, N\}$. We also assume that there exists a public key infrastructure (PKI) and nodes can link between the identity and the public key of each node. Moreover, we introduce the "chain" as a data structure that consists of an ordered sequence of blocks. Each block consists of multiple transactions and a hash digest of the previous block, except for the first block, namely the genesis block.

B. VTL Model

Inspired by Bitcoin, most of the blockchain systems mimic value transfer systems, e.g., currency, in a decentralized fashion. The problem could be described as the following: Each node holds some positive value that they could transfer to others via transactions. A transaction is only valid if it is authorized by the owner of the value and the value cannot be double-spent. In other words, for any "value" in the network, it has three properties:

1) **Ownership:** Value has an owner. Only the owner of the value can authorize to transfer his value.
2) **Fluidity:** Any transfer can be completed in finite time.
3) **Validity:** The value cannot be created or duplicated.

A value transfer system can be in many forms, e.g., the account-based ledgers, which is widely used in banking system and many other accounting systems. Bitcoin, as well as many other blockchain systems, use a ledger with UTXO structure, which is very suitable for decentralized systems. Here, we introduce the UTXO structure.

1) UTXO: Firstly, in UTXO a transaction is an authorized piece of information that transfers the value from one node to another. In this paper, we use the following definition for a transaction, which is a slight variation of the traditional UTXO structure used in Bitcoin.

Definition 2 (Transaction)**.** A transaction tx_i is a five-tuple: $tx_i = \langle \text{Source}_i, s_i, d_i, a_i, r_i \rangle$ where Source_i is the set of transactions which are used as the source, s_i is the sender, d_i is the receiver, a_i is the transacted value, and r_i is the remaining value.

In Bitcoin, transactions are usually referred to the ones on the longest chain, which also suggest that they are *valid* transactions. However, in some other systems like [30], [31], the concept of a valid transaction is ambiguous since invalid transactions can also exist on the chain. As a result, a deterministic and consistent rule should be applied for all nodes to determine the valid transactions. Here, we define the valid transaction in UTXO as the following.

Definition 3 (Validity of a Transaction)**.** A transaction $tx_i = \langle \text{Source}_i, s_i, d_i, a_i, r_i \rangle$ is valid if and only if the following conditions hold.

- **Confirmed and authorized:** tx_i, as well as some witness indicating that tx_i is authorized by s_i, e.g., a digital signature of tx_i signed by s_i, are on a tamper-proof ledger.
- **Valid sources:** All transactions in $tx_j \in \text{Source}_i$ are valid.
- **Value equality:** The original value equals to the sum of the transacted value and the remaining value, i.e., $\sum_{tx_j \in \text{Source}_i} r_j = a_i + r_i$.
- **No double-spending:** Assuming $tx_i = t_{u,k,l}$, for any $tx_j \in \text{Source}_i$, there does not exist a valid transaction $tx_{i'} = t_{u,k',l'}, tx_j \in \text{Source}_{i'}$ such that $k' < k$ or $k = k, l' < l$.

Then, we define the unspent transaction in UTXO.

Definition 4 (Unspent Transaction)**.** A transaction tx_i is an unspent transaction if there is no other transaction tx_j in the ledger which is valid and $tx_i \in \text{Source}_j$.

Clearly, in UTXO structure, the value exists in the form of unspent transactions. The value is always transferred from one unspent transaction to another unspent transaction, instead of transferring from one account to another account as the account-based ledger structure.

2) Properties of VTL Model: Traditional blockchain systems prevent double-spending by reaching the classical BFT consensus on all transactions. The most straightforward approach is to treat transactions as bit strings and use classical BFT algorithms [4], [7], [8] or improved BFT algorithms [10], [11], [13] to reach consensus. However, this approach misses the notion of "value" behind the transactions and discards the differences between a transaction and general data. These differences will be uncovered if the original notion of value is focused. Here, we pick up the idea behind the "rational nodes" and "transaction" notions in Nakamoto-like consensus and add more real-world interpretations to these two notions in value transfers.

Firstly, value has an owner and the receiver of an unspent transaction is the owner of that value until it is spent again. The owner would take full initiative and responsibility of proving the existence and the authenticity of the value to any node upon request. If he fails to do so, it will be considered as against his own interest. For instance, if the value is considered as money, the holder of the money is motivated to prove the money is real when he uses it for purchase. A failure in proving will cause the purchase to fail, which is against his own interest.

Secondly, the concern of the nodes is the authenticity of the value they owned instead of the transaction records. Hence, nodes will check the past transaction records only if the records have impact on the authenticity of the value that they concern. Otherwise, nodes have no interest and will not care about the validity of a past transaction.

As a result, we make three assumptions in VTL model. Throughout this paper, we use the term "node u is curious about transaction tx_i" to represent that node would like to check the validity of transaction tx_i.

Assumption 1 (History Disinterest)**.** A node u is curious about a spent transaction tx_i only when it is curious about an unspent transaction tx_j and the validity of tx_i is required to check the validity of tx_j.

Assumption 2 (Rational Receiving)**.** A node u is curious about transaction tx_i if it is the receiver of tx_i and does not know the validity of it.

Assumption 3 (Rational Owner)**.** If node u is the receiver of a valid and unspent transaction tx_i, it will provide the validity proof of tx_i to any node once it is requested.

In practice, it is not rational for a node to validate an unspent transaction if it is not the receiver since validation is resource consuming. Hence, we have an alternative version for Assumption 2 to minimize the cost in a resource-limited network.

Assumption 4 (Rational and Cost-saving Receiving)**.** A node u is curious about transaction tx_i **if and only if** it is the receiver of tx_i and does not know the validity of it.

This assumption will not affect the correctness of our scheme. It will be applied in the performance analysis for simplicity.

3) Valid VTL System: Then, we define a valid VTL system with a structure of UTXO.

Definition 5 (Valid VTL System)**.** A system with UTXO structure is called a valid VTL system if it satisfies the following conditions under Assumption 1-3:

1) **Ownership:** If an honest node receives a valid transaction, then he can make one valid transaction using it as a source. Meanwhile, no other node can make a valid transaction using it as a source.
2) **Fluidity:** A valid transaction will be considered as valid by the receiver in finite time if both the sender and the receiver are honest.
3) **Validity:** Invalid transactions will not be considered as valid by honest nodes.

Remark 1 (Relationship Between BFT Consensus and Valid VTL System)*.* Clearly, the Validity condition of valid VTL systems is exactly the Validity condition in the BFT consensus. Then, the Fluidity condition is guaranteed by all three BFT consensus conditions and the Ownership condition is guaranteed by the Agreement condition and the way that UTXO structure is designed. Hence, BFT consensus on the transactions is a sufficient condition for a valid VTL system. However, later we will show that it is not a necessary condition for VTL since the BFT consensus with a weakened Termination condition is also sufficient for a valid VTL system.

III. Our System

Our system consists of three parts: individual chains for transactions, a main chain for a global shared state, and a validation scheme for validation of the transactions. In this section, we first introduce these three parts of our system and give important theorems of the system. Then, we prove that our system is a valid VTL system as well as prove that our system actually only compromises the Termination condition of the BFT consensus.

A. Individual Chains

Each node generates an individual chain to record their own transactions in a first-in-first-out fashion. An individual chain of node u is an ordered set of blocks $\{B_{u,1}, B_{u,2}, \dots, \}$ and a block is an ordered set $B_{u,k} = \{H(B_{u,k-1}), t_{u,k,1}, t_{u,k,2}, \dots\}$, where $t_{u,k,l}$ is a transaction sent by node u with valid sources, value equality, and no double-spending as defined in Definition 3. In our system, we assume that there is an initial value assigned to each node in the same fashion as a transaction with no source. The sender and receiver of this transaction are both the node itself.

The size of a block can be arbitrary. Periodically, nodes send *Abstracts* to the *main chain* (will be introduced in the next paragraph). The abstract is defined as the following.

Definition 6 (Abstract)**.** An abstract of block $B_{u,k}$, denoted by $A_{u,k}$, is a four-tuple: $A_{u,k} = \langle u, k, H(B_{u,k}), Sig_u(u||k||H(B_{u,k}))\rangle$.

B. Main Chain

The main chain uses PBFT as its consensus algorithm and the blocks consist of *Abstracts* signed by the corresponding nodes. We assume that the abstracts of all genesis blocks are on the main chain. Since it has been proved that the PBFT can reach BFT consensus on messages in our network model [4], we simply see the main chain as a reliable and secure primitive and all abstracts included on the main chain reaching the BFT consensus. Honest nodes will send abstracts of their newest blocks to the main chain when they observe that their previous abstracts are on-chain.

C. Confirmation

The transactions on individual chains are arbitrary in the sense that they are neither tamper-proof nor signed. The transactions will be tamper-proof and signed if an abstract of a block that comes after it is contained in the main chain, which we call confirmed transactions. Here, we give the formal definitions of a confirmed transaction and a confirmed block.

Definition 7 (Confirmation)**.** A block $B_{u,k}$ is confirmed if

- an abstract of the block or a block after it, i.e., $A_{u,k'}, k' \geq k$, is on the main chain;
- for all abstracts of node u, denoted by $A_{u,l}$, that are on the main chain and $l \leq k'$, $A_{u,l}$ is compliant to their corresponding blocks.

A transaction $tx_i = \langle \text{Source}_i, s_i, d_i, a_i, r_i \rangle$ is a confirmed transaction if $tx_i \in B_{u,k}, s_i = u$, and $B_{u,k}$ is confirmed. We call $B_{u,k}$ and tx_i are confirmed by $A_{u,k'}$.

The confirmation of a transaction suggests that it is tamper-proof as if it is on-chain, which is shown in the following theorem.

Theorem 1 (Confirmed Transactions)**.** *If $tx_i = t_{u,k,l}$ is a transaction confirmed by abstract $A_{u,k'}, k' \geq k$, then there does not exist a chain of confirmed blocks $\{B'_{u,1}, B'_{u,2}, \dots, B'_{u,k'}\}$ such that $t'_{u,k,l} \neq tx_i$ and all hashes are correct.*

The formal proof of this theorem is given in Appendix A. By Theorem 1, when a transaction is confirmed, the position and content of it cannot be changed. Furthermore, it is also signed since the sender of the transaction is the same as the signer of the abstract and the abstract contains an unforgeable signature of the sender. Note that a confirmed transaction here is not the same as confirmed transaction in other blockchain systems like Bitcoin, as they are not yet validated.

D. Validation Scheme

Our validation scheme consists of two parts: a proof collection process that allows any node that is curious about a transaction to reliably and efficiently collect the proof of it; a validation function that deterministically decide whether a transaction is valid or not depending on the collected proof.

1) Proof Collection: First we define the validity proof of a transaction tx_i.

Definition 8 (Validity Proof). Assuming that the sender $s_i = u$ for transaction tx_i, $tx_i \in B_{u,k}$, and there exists an abstract $A_{u,k'}, k' \geq k$ in the main chain, a validity proof $\mathcal{P}(tx_i)$ is the union of a set of all blocks before and including $B_{u,k'}$ and the proofs of all transactions in Source_i, i.e., $\mathcal{P}(tx_i) = \{B_{u,k''} | k'' \leq k'\} \cup \{B_{v,l} | B_{v,l} \in \mathcal{P}(tx_j), tx_j \in \text{Source}_i\}$.

By Definition 8, a validity proof of $tx_i \in B_{u,k}$ includes the chain of u from the genesis block to a block $B_{u,k'}, k' \geq k$ which has an abstract in the main chain. Moreover, it also includes the chains of the sources of this transaction, and recursively the sources of the sources until the genesis block.

In the following lemma, we show that the proofs of valid transactions can always be collected by nodes who are curious about them in the VTL model.

Lemma 1 (Feasibility of the Proof Collection). *If a node u is curious about a valid transaction tx_i, then it can always identify a node v such that it would provide the proof of tx_i.*

Proof. If tx_i is an unspent transaction, this lemma directly follows from Assumption 3 since the receiver of tx_i will provide it. If tx_i is a spent transaction, then by Assumption 1, u will only be curious about tx_i if u is curious about an unspent transaction tx_j and the validity of tx_i is required for the validity of tx_j. By Definition 8, we have $\mathcal{P}(tx_i) \subset \mathcal{P}(tx_j)$. Hence, by Assumption 3, u can collect the proof of tx_j from the receiver of tx_j. □

By Lemma 1, the proof of a transaction tx_i can always be collected reliably and efficiently. By reliably, we mean that by Lemma 1, the proof can always be collected in the VTL model without any risk of disconnections. By efficiently, we mean that the collection is a simple point-to-point communication without the need of a reliable broadcast scheme like [8] to tolerant malicious behaviors.

However, although by our model the receiver of an unspent transaction is motivated to provide the correct proof, the requester of the proof will not accept it as a proof without his own verification. We give the Proof Verification Algorithm $\mathsf{Ver}(\mathcal{P}(tx_i))$ in Algorithm 2 in Appendix B. If $\mathsf{Ver}(\mathcal{P}(tx_i)) =$ pass, it suggests that $\mathcal{P}(tx_i)$ is indeed a validity proof for transaction tx_i since the algorithm is a direct translation from the definition of the validity proof.

2) Validation Function: The deterministic Validation Function is given in Algorithm 1.

The correctness of the validation function is given in the following theorem.

Theorem 2. $\mathsf{V}(tx_i, \mathcal{P}(tx_i)) = \mathsf{valid}$ *if and only if tx_i is valid.*

This theorem holds since the validation function is a straightforward translation of the definition of the validity. The detailed proof is given in Appendix C.

Algorithm 1 Validation Function $\mathsf{V}(tx_i, \mathcal{P}(tx_i)), tx_i = \langle \text{Source}_i, s_i, d_i, a_i, r_i \rangle \in B_{u,k}$

```
#Validity Proof Check
if $\mathsf{Ver}(\mathcal{P}(tx_i)) \neq$ pass then return unknown
#Equality Check
if $\sum$(all remaining values from $\text{Source}_i$) $\neq a_i + r_i$ then return unknown
#Double-Spending Check
for $B_{u,m}, m = [1 : k]$ do
    for All transactions $tx_j$ in $B_{u,m}$ do
        if $\text{Source}_j \cap \text{Source}_i \neq \emptyset$ and $tx_i \neq tx_j$ then return unknown
#Source Check
for all transactions $tx_j$ in $\text{Source}_i$ do
    if $\mathsf{V}(tx_j, \mathcal{P}(tx_j)) \neq$ valid then return unknown
return valid
```

E. BFT Satisfactory

Here, we show that our system satisfies the agreement and validity condition of BFT with a compromised termination condition for all valid transactions.

Theorem 3 (BFT Satisfactory). *Our system satisfies the following conditions in VTL model. Here, we use the term "node u validates a transaction tx_i" to represent that node u runs a validation function on tx_i with the result* valid.

- **Agreement (Consistency):** *If an honest node validated a transaction, then, if another honest node is curious about this transaction, it will also validate it.*
- **Validity (Correctness):** *If a transaction can be validated by an honest node, then at least one honest node that is curious about it can validate it.*
- **Termination (Liveness):** *If a transaction is proposed by an honest node, then it can be validated in finite time.*

Proof.

- **Agreement:** If a transaction tx_i is validated by an honest node, i.e., $\mathsf{V}(tx_i, \mathcal{P}(tx_i)) = \mathsf{valid}$, then, by Theorem 2, tx_i is valid. By Lemma 1, if another node is curious about tx_i, the proof can be collected. Then, since the validation function is deterministic, another curious honest node will also run the validation function and the result will be valid.
- **Validity:** Firstly, by Theorem 2, a validated transaction is equivalent to a valid transaction. Then, by Lemma 1 its proof can be collected by an honest curious node and by Theorem 2 it will be validated.
- **Termination:** If a transaction is proposed by an honest node, by the definition of the honest node, it should have valid sources, value equality, and no double-spending as suggested in Subsection III-A. Then, by the BFT satisfactory of the PBFT scheme we used for the main chain, this transaction will eventually be confirmed and meets all requirement of a valid transaction. Then, by Theorem 2, it can be validated.

□

F. Validity of the System

Now we show that our system is a valid VTL system by showing all three conditions in Definition 5 are guaranteed in the VTL model.

Theorem 4 (Validity of Our System)**.** *A system described in this section is a valid VTL system.*

Proof.

- **Ownership:** By the Validity condition of Theorem 3, honest nodes can make valid transactions with valid received transactions as sources. Meanwhile, if another node makes a transaction with sources that he is not the receiver, by Theorem 1, the proof will be considered as incorrect and the validation will fail.
- **Fluidity:** This condition is guaranteed by combining all three conditions in Theorem 3.
- **Validity:** This condition directly follows from the Validity Condition in Theorem 3.

□

The insight of Theorem refth:vtl showing our system as a valid VTL system is that our system does not guarantee BFT for generic data and cannot be used in applications where Assumption 1-3 do not hold. For example, valid transactions cannot be validated by any other nodes if the sender refuses to offer the proof to any other nodes. However, this scenario is basically denying the value of the sender himself and thus should not happen in the VTL model with UTXO structure if the nodes are rational. On the other hand, if the transactions are in the form of debits instead credits and the receiver is the interested party, our system could guarantee neither the BFT condition nor the conditions for a valid VTL system.

IV. Performance Analysis and Spontaneous Sharding

In this section, we will give explanations for the scale-out claim that we made for the throughput. First we show that the throughput of our system is scalable even in the worst case and will naturally scale out if the transaction pattern is already sharded. Then, we explain why and how our system could spontaneously shard. We give examples with theoretical and simulative analysis to show that the feasibility of spontaneous sharding as well as the scale-out throughput.

A. Communication Cost Per Transaction

In our system, the main chain is using PBFT with $O(N^2)$ message complexity. However, the number of transactions associated with one abstract in the main chain are arbitrary and independent of the main chain. As a result, the communication cost of the main chain can be made into a negligible term in CCPT if we choose the number of transactions associated with one abstract to be $\omega(N^2)$. The duration of the consensus process still plays an important role in the latency of our system. However, note that the PBFT-based scheme is used only for easy comprehension and can be easily replaced by other scalable and low latency blockchain systems to improve the latency.

Then, we make a few assumptions for easier analysis of the CCPT. We assume that rational nodes will not care about invalid transactions and thus will not try to re-collect the proof of a transaction if it is failed for a number of times. In other words, malicious nodes cannot spam invalid proofs to jam the network. For the simplicity in analysis, we only consider the resource limited network with Property 4. Then, in our system, the CCPT can be represented by p/T, where p is the total communication cost of all proofs and T the total number of transactions made by the whole network.

In general, the proof of a transaction tx_i includes the chains of the sender, the senders of all sources of this transaction, and the senders of, recursively, the sources of the sources. In most blockchain systems, the storage is traded for validation efficiency, i.e., the validated transactions and their proofs are stored and the proofs of new transactions are collected incrementally. Then, in the worst case when the proof of any transaction includes the chains of all nodes, if the storage is not limited, all nodes simply need to be updated with all transactions in the network. Each transaction will be communicated by $O(1)$ message per each node due to Property 3 and Lemma 1 since a point-to-point based collection is sufficient to guarantee reliability and there is no need for BFT reliable broadcast schemes. Then, for each node, the communication cost for all its proofs is $O(1)T$. For the whole network, $p = O(1)NT$ and the CCPT is thus $O(N)$.

A better case would be that the transaction pattern is separated into shards and the nodes only make intra-shard transactions. In that case, the proof of any transaction contains the chains of only the nodes in their shards and the CCPT is $O(g)$, where g is the size of the shard. As a result, our system achieves scale-out throughput.

B. Spontaneous Sharding

Here, we consider a more interesting case that the transaction pattern is not separated into small shards and show that our system could still achieve scale-out throughput if all nodes behave rationally. We call this spontaneous sharding. The idea behind the spontaneous sharding is simple: rational nodes will try to minimize their transmission and storage costs by minimizing the proof size of each transaction as well as the number of recorded transactions. Then, the minimum cost of either transmitting or storing the transaction is actually the cost of proving the authenticity of the value in the transaction, which depends on the number of nodes that this value has been passed through.

More precisely, let us focus on a piece of value originated from a genesis block. For each time that it is transferred to a node, a confirmed chain of that node is included in the proof of the transaction of that value. Meanwhile, if the value is used together with other sources to make transaction, then these pieces of value are combined as well as their proofs. Hence, rational node will always avoid combining values for the sake of transmission and storage costs. Besides, the cost can be reduced by trying to make each piece of value only cycling in a small shard of the network, which makes the proof of the value only contain the chains of the nodes in that shard. Let us denote P_i for the set of chains that are included in the proofs of all values owned by node i. By the analysis made in Subsection IV-A, there is minimum overhead for proof collection in our system and the communication cost for each transaction per node is $O(1)$ messages. Hence, the CCPT in our system can be calculated as $O(g)$, where $g = E[|P_i|]$. This is called spontaneous sharding since the throughput increased as if the network has been naturally sharded.

Here, we give a more detailed analysis with graph theory. Let us consider the transaction pattern as a weighted directed graph $G(V, \vec{E})$, where the vertices $v \in V$ represent the nodes in the network. Then, instead of actually transactions, the edges $e = (u, v, w), e \in \vec{E}$ represent the transaction channels and their capacities, i.e., the existence of transactions between the sender u and receiver v, and the rate for the transactions (amount per second) denoted by w. Then, we assume a stable and sustainable value-transfer network, where all nodes have equal amount of income and outcome in a long term. Let us denote the sets for inbound edges and outbound edges of node i as I_i and O_i, respectively, i.e., $I_i = \{e \in \vec{E} : e = (u, i, w)\}$, $O_i = \{e \in \vec{E} : e = (i, u, w)\}$.

When a piece of value sent by node i via edge $e = (i, u, w) \in O_i$ to node u, this amount of value w should return to node i through a path. The nodes on the path form a node set, denoted by $\mathcal{N}_e \subseteq V$. In our system, the value will then return with all the chains of nodes in $\mathcal{N}_e$. The same holds for the values received from the edges in I_i as each piece of these values will eventually return to the corresponding inbound neighbor of node i. As a result, we have $P_i = \{v \in \mathcal{N}_e : e \in I_i \cup O_i\}$.

There are two ways to optimize P_i and achieve spontaneous sharding: local optimization and global optimization.

A local optimization can straightforwardly be done by the following: according to the information about the chains that the receiver already has, the sender will choose from all its unspent transactions for the ones with the least amount of required proofs. For example, node 1 has transacted with a receiver node 2 who has already acquired the chains of $\{3, 4, 5, 6\}$ in this round. Then, if node 1 has this information, it will prefer to use the unspent transactions with proofs that consist of the chains from $\{3, 4, 5, 6\}$ for transactions to node 2 so that it does not need to send proofs anymore. In Appendix D, a naive smart transacting algorithm is given to achieve local optimization.

A global effort could be made by letting all nodes broadcast their acquired chains each consensus round. This effort is spontaneous and beneficial to the nodes themselves, so a reliable broadcasting scheme is not necessary. An additional $O(N^2 g)$ communication cost is required each round, which adds at most $O(g)$ to the CCPT. With the global information, some optimization schemes could be run locally as references for the source selection when nodes send transactions. The purpose of the global optimization is to route all values with the same sink through paths that include the minimum number of nodes. A feasible global optimization scheme is a non-trivial problem that we leave for future research.

However, the performance of the sharding and the eventually throughput depends heavily on the network model and the transaction pattern. Here, we give theoretical analysis showing the possibility of our system to scale-out in some large random networks. Then, we use simulation to show spontaneous sharding is feasible even in small networks.

Remark 2 (Tragedy of the Commons). It seems that the spontaneous sharding would only happen if all nodes perform rationally and cooperate, which will fall into the pitfall of tragedy of the commons [32] if some nodes with high capacity do not optimize their proof sizes. However, this system is not identical to the tragedy of the common scenario since spontaneous sharding could also happen locally so that transmission cost is a private resource rather than public resource. In other words, a group of resource limited nodes can optimize their transactions locally and reduces their transmission cost without needing global cooperation.

C. Examples

Firstly, we show that our system with a global optimization scheme will scale out in large random networks.

Example 1. We consider a random directed weighted graph constructed as the Erdős-Rényi model with N nodes, M edges, connectivity $p = \frac{M}{N(N-1)}$, and p is larger than $\frac{\ln N}{N}$ so that the network is fully connected. We define f as the "weight factor", which is the average number of transaction channels required for a piece of value, i.e., $f = E[\lceil \frac{w}{E[w]} \rceil]_w$. This value is 1 in unweighted graph and is $O(1)$ if w follows Poisson distribution.

For Example 1, let us consider the average size of P_i. Firstly, if our system is globally optimized, we will have

$$\begin{aligned} E[|P_i|] &= E[|\{v \in \mathcal{N}_e : e \in I_i \cup O_i\}| && (1) \\ &\leq E[\sum_{e \in I_i \cup O_i} |\mathcal{N}_e|]. && (2) \end{aligned}$$

Hence we focus on one path in which the value from an outbound edge e from node i flows back to i. The average path length is denoted by l. Then, due to the limited capacity of the edges, this value requires $\sim fl$ edges to be delivered. Hence, there are $\sim fl$ nodes in the set $\mathcal{N}_e$. Combining this with (2) we have

$$\begin{aligned} & E[\sum_{e \in I_i \cup O_i} |\mathcal{N}_e|] && (3) \\ & \sim \; 2cfl && (4) \\ & \sim \; 2f \cdot c \frac{\ln N}{\ln pN} && (5) \\ & \sim \; 2f \cdot c \frac{\ln N}{\ln c}, && (6) \end{aligned}$$

where c is the average inbound (outbound) connectivity, which equals to $M/N = p(N-1)$. Here, we have (4) since the average number of elements in $I_i \cup O_i$ is $2c$. Then, (5) follows from the average path length $l \sim \frac{\ln N}{\ln pN}$ from the random graph. Combining (6) with (2) and observing that $2f \cdot c\frac{\ln N}{\ln c}$ is dominated by N when $c = o(\frac{N}{f \ln N})$, we have the following condition: if conditions

$$\begin{cases} p > \frac{\ln N}{N} \\ p = o(\frac{1}{f \ln N}) \end{cases} \tag{7}$$

hold, our system could scale out if a global optimization scheme is used for spontaneous sharding, i.e., $E[|P_i|] = o(N)$. Here, f will be $O(1)$ if the transaction rates between nodes follow Poisson distribution. In that case, for instance, if $p = O(\frac{\ln N}{N})$ which suggests $c = O(\ln N)$, each node will only need to acquire on average $O(\frac{\ln N \cdot \ln N}{\ln \ln N})$ chains.

Then, we give an example showing that even without global optimizations, a naive local optimization scheme could already result in the reduction of acquired chains in a small network. In order to show that the spontaneous sharding could be achieved in the worst case, we artificially construct an extreme network.

Example 2. We consider N nodes $\{1, 2, \ldots, N\}$ placed in a ring and each node transacts to the next c nodes on its right and receives from the next c nodes on its left. Each node is given an initial amount of value and will uniformly at randomly make transactions to the c nodes. The frequency and the amount of the transaction follow Poisson and uniform distributions, respectively.

We run a simulation with our system for $N = 10, 15, 20, 25$ nodes with difference connectivity c. We simulate the communication between nodes with Netty[3] and the main chain with Tendermint [33]. We apply a naive smart transacting algorithm in which the sender simply checks his own transaction records for the information about the P_i of his receivers and choose the sources accordingly. The implementation details and the source code of our system and the simulation can be found in https://github.com/blockchain-lab/ScaleOutDistributedLedger. Fig. 1 shows the scenarios of sharding in stable states. It can be observed that when c is small, we have $g < N$, meaning spontaneous sharding is achieved. However, when c is equal to or larger than $3, 4, 4, 6$ for $10, 15, 20, 25$ nodes, respectively, all nodes would acquire all chains. Then the throughput of our system is no better than other scalable blockchain systems. It is due to the naiveness of the algorithm that we use for local optimization, e.g., the current algorithm does not avoid combining sources with different proofs into a single transactions and does not distinguish between chains received only once and chains being continuously updated. With better optimization algorithms, we believe that smaller g can be achieved for larger c.

V. Conclusion and Future Work

In this paper, we proposed a novel blockchain system for the most considered type of distributed ledgers which we called VTL model. In VTL model, we assume that nodes are rational

[3]http://netty.io/

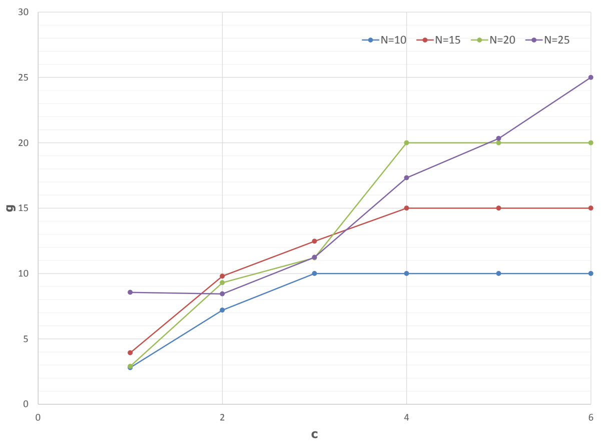

Fig. 1. Average number of chains acquired by nodes for various connectivity in stable states.

and will be motivated to prove their possessed value. Our system has a very simple and fully decentralized structure that does not introduce any node serving as "validator". Our system achieves uncompromised agreement and validity conditions and could scale out by spontaneous sharding without sacrificing security or decentralization. However, as the focus of this paper is put on the formal theoretical introduction of VTL and the off-chain and proof-based framework, many refinements in practical perspective are left for future research.

- **Checkpoints to improve storage efficiency:** As sharding is a spontaneous and gradual process that might requires a initial phase, it might happened that nodes are required to record the whole transaction set until sharding starts. Then, the storage cost per transaction will not scale-out and cost new nodes quite heavily to join. This problem could be mitigated by introducing checkpoints in the main chain, which verifies the validity of certain values so that later on the proofs of these values do not have to date back to the genesis blocks. However, this do requires the newcomers to trust the old nodes who verified these values.
- **Private channels for low latency payments:** Private off-chain channels like [20] is complicated in traditional blockchains since the notion of value is hinged to the on-chain ledger. However, as in our system the value is off-chain by nature, private channels for low latency micro-payments can be easily designed.
- **Supportive to conditional payments/smart contracts:** We conjecture that conditional payments and smart contracts can also be supported by this system with modified data structure and validation scheme as long as each transaction includes some value transferred to at least one of the receivers. Such system will simultaneously achieve sharding on both communication/storage resources and computation resources.
- **Real-world Implementation:** We conjecture that our system will also scale-out in more practical networks

models [34], [35] or real-life transactions patterns [36], [37]. Moreover, we believe that for most of the cryptocurrencies nowadays, our system will be very beneficial in throughput since most of them are very "trader-centric". Then, for most users who only transact with traders, their transmission and storage cost can be significantly saved if the traders apply local optimization.

- **Discrimination and hidden forks:** As the proof size of transaction can be very different, it might cause issues of discrimination and hidden forks, e.g., values with huge proof sizes are refused by some nodes in the network, effectively causing a fork of the chain. This problem can be partially solved if checkpoints are used. However, we do not necessarily see this as a problem and argue that this is no more dangerous than forks in traditional blockchains.

Acknowledgment

This work was supported by NWO Grant 439.16.614 Blockchain and Logistics Innovation.

The authors would also like to thank Chiel Bruin and Karol Jurasinski from Delft University of Technology for the help in implementing the scheme and the anonymous reviewers for the insightful comments.

References

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

[2] S. King and S. Nadal, "PPCoin: Peer-to-peer crypto-currency with proof-of-stake," 2012. [Online]. Available: https://peercoin.net/assets/paper/peercoin-paper.pdf

[3] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proceedings of the 26th Symposium on Operating Systems Principles*. ACM, 2017, pp. 51–68.

[4] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *OSDI*, vol. 99, 1999, pp. 173–186.

[5] M. J. Fischer, N. A. Lynch, and M. S. Paterson, "Impossibility of distributed consensus with one faulty process," *J. ACM*, vol. 32, no. 2, pp. 374–382, Apr. 1985.

[6] S. Gilbert and N. Lynch, "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services," *SIGACT News*, vol. 33, no. 2, pp. 51–59, Jun. 2002.

[7] M. Ben-Or, B. Kelmer, and T. Rabin, "Asynchronous secure computations with optimal resilience," in *Proceedings of the thirteenth annual ACM symposium on Principles of distributed computing*. ACM, 1994, pp. 183–192.

[8] G. Bracha, "Asynchronous byzantine agreement protocols," *Information and Computation*, vol. 75, no. 2, pp. 130–143, 1987.

[9] J. Garay, A. Kiayias, and N. Leonardos, *The Bitcoin Backbone Protocol: Analysis and Applications*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 281–310.

[10] C. Cachin and S. Tessaro, "Asynchronous verifiable information dispersal," in *Reliable Distributed Systems, 2005. SRDS 2005. 24th IEEE Symposium on*. IEEE, 2005, pp. 191–201.

[11] A. Miller, Y. Xia, K. Croman, E. Shi, and D. Song, "The honey badger of BFT protocols," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 31–42.

[12] R. Guerraoui, N. Knežević, V. Quéma, and M. Vukolić, "The next 700 BFT protocols," in *Proceedings of the 5th European conference on Computer systems*. ACM, 2010, pp. 363–376.

[13] R. Kotla, L. Alvisi, M. Dahlin, A. Clement, and E. Wong, "Zyzzyva: speculative byzantine fault tolerance," in *ACM SIGOPS Operating Systems Review*, vol. 41, no. 6. ACM, 2007, pp. 45–58.

[14] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, E. G. Sirer *et al.*, "On scaling decentralized blockchains," in *International Conference on Financial Cryptography and Data Security*. Springer, 2016, pp. 106–125.

[15] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse, "Bitcoin-NG: A scalable blockchain protocol," in *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*. USENIX Association, 2016, pp. 45–59.

[16] R. Pass and E. Shi, "Hybrid consensus: Efficient consensus in the permissionless model," *IACR Cryptology ePrint Archive*, 2016. [Online]. Available: http://eprint.iacr.org/2016/917.pdf

[17] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *Annual International Cryptology Conference*. Springer, 2017, pp. 357–388.

[18] S. D. Lerner, "Rsk: Bitcoin powered smart contracts," 2015. [Online]. Available: https://uploads.strikinglycdn.com/files/90847694-70f0-4668-ba7f-dd0c6b0b00a1/RootstockWhitePaperv9-Overview.pdf

[19] G. Wood, "Polkadot: Vision for a heterogeneous multi-chain framework," 2016. [Online]. Available: http://www.the-blockchain.com/docs/Gavin%20Wood%20-%20Polkadot%20-%20%20Vision%20For%20A%20Heterogeneous%20Multi-chain%20Framework.pdf

[20] J. Poon and T. Dryja, "The bitcoin lightning network: Scalable off-chain instant payments," *Technical Report (draft)*, 2015. [Online]. Available: https://lightning.network/lightning-network-paper.pdf

[21] J. Poon and V. Buterin, "Plasma: Scalable autonomous smart contracts," 2017. [Online]. Available: https://plasma.io/plasma.pdf

[22] S. Popov, "The tangle," 2014. [Online]. Available: https://iota.org/IOTA_Whitepaper.pdf

[23] A. Churyumov, "Byteball: A decentralized system for storage and transfer of value," 2016. [Online]. Available: https://byteball.org/Byteball.pdf

[24] L. Baird, "The swirld hashgraph consensus algorithm: Fair, fast, byzantine fault tolerance," 2016. [Online]. Available: http://www.swirlds.com/downloads/SWIRLDS-TR-2016-01.pdf

[25] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, and B. Ford, "Omniledger: A secure, scale-out, decentralized ledger," *IACR Cryptology ePrint Archive*. [Online]. Available: https://eprint.iacr.org/2017/406.pdf

[26] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A secure sharding protocol for open blockchains," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: ACM, 2016, pp. 17–30.

[27] V. Buterin, "On sharding blockchains," *Sharding FAQ*, 2017. [Online]. Available: https://github.com/ethereum/wiki/wiki/Sharding-FAQ

[28] M. Al-Bassam, A. Sonnino, S. Bano, D. Hrycyszyn, and G. Danezis, "Chainspace: A sharded smart contracts platform," *CoRR*, vol. abs/1708.03778, 2017. [Online]. Available: http://arxiv.org/abs/1708.03778

[29] P. Todd, "Email: [bitcoin-development] tree-chains preliminary summary," 2014. [Online]. Available: https://www.mail-archive.com/bitcoin-development@lists.sourceforge.net/msg04388.html

[30] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, 2014. [Online]. Available: http://gavwood.com/paper.pdf

[31] Y. L. Yonatan Sompolinsky and A. Zohar, "Serialization of proof-of-work events: Confirming transactions via recursive elections," *IACR Cryptology ePrint Archive*. [Online]. Available: https://eprint.iacr.org/2016/1159.pdf

[32] G. Hardin, "The tragedy of the commons," *Journal of Natural Resources Policy Research*, vol. 1, no. 3, pp. 243–253, 2009.

[33] J. Kwon, "Tendermint: Consensus without mining," 2014. [Online]. Available: https://tendermint.com/static/docs/tendermint.pdf

[34] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," *Reviews of modern physics*, vol. 74, no. 1, p. 47, 2002.

[35] D. J. Watts and S. H. Strogatz, "Collective dynamics of small-worldnetworks," *nature*, vol. 393, no. 6684, p. 440, 1998.

[36] M. A. Serrano and M. Boguñá, "Topology of the world trade web," *Phys. Rev. E*, vol. 68, p. 015101, Jul 2003.

[37] K. Soramki, M. L. Bech, J. Arnold, R. J. Glass, and W. E. Beyeler, "The topology of interbank payment flows," *Physica A: Statistical Mechanics and its Applications*, vol. 379, no. 1, pp. 317 – 333, 2007.

Appendix A
Proof of Theorem 1

Proof. We proof this theorem by contradiction. If there exists a chain $\{B'_{u,1}, B'_{u,2}, \ldots, B'_{u,k'}\}$ which consists of confirmed block and $t'_{u,k,l} \neq tx_i$. First of all, confirmed blocks suggest that all abstracts of $\{B'_{u,1}, B'_{u,2}, \ldots, B'_{u,k'}\}$ is on the main chain. Then, since $t_{u,k,l} = tx_i$ is confirmed by an abstract $A_{u,k'}$, all abstract of the chain $\{B_{u,1}, B_{u,2}, \ldots, B_{u,k'}\}$ are also on the main chain. Moreover, since the abstracts are signed and the digital signatures are assumed to be unbreakable, the chains $\{B'_{u,1}, B'_{u,2}, \ldots, B'_{u,k'}\}$ and $\{B_{u,1}, B_{u,2}, \ldots, B_{u,k'}\}$ will have the same set of abstracts of node u on the main chain, which includes $A_{u,k'}$ as the abstract of both $B'_{u,k'}$ and $B_{u,k'}$. Since the hash function is assumed to be unbreakable, we have $B'_{u,k'} = B_{u,k'}$. Then, if $t'_{u,k,l} \neq tx_i$, then we will have $B'_{u,k} \neq B_{u,k}$ and $B'_{u,k'} = B_{u,k'}, k' \geq k$, which contradicts our assumption of the unbreakable hash function. □

Appendix B
Proof Verification Algorithm

Here we give the Proof Verification Algorithm as Algorithm 2.

Algorithm 2 Proof Verification Algorithm $\mathsf{Ver}(\mathcal{P}(tx_i)), tx_i \in B_{u,k}$

```
#Verify the chain including this transaction
count ← 0
absmark ← 0
for $B_{s_i,m}, m = 1 : \max$ do        ▷ Check the integrity of the chain
    if $tx_i \in B_{s_i,m}$ then count + +
    if $m \neq 1$ and first element in $B_{s_i,m} \neq H(B_{s_i,m-1})$ then return fail
    if $A_{s_i,m}$ is included in the main chain then
        absmark ← $m$
        if $H(B_{s_i,m}) \notin A_{s,m}$ or $Sig_u(u, k, H(B_{u,k}))$ is not correct then return fail
if absmark $< k$ then return fail        ▷ Check the confirmation
if count $\neq 1$ then return fail        ▷ Check the existence of the transaction
#Verify the chains of the sources
for all $tx_j \in \mathrm{Source}_i$ do
    if Ver($tx_j$) $\neq$ pass then return fail
return pass
```

Appendix C
Proof of Theorem 2

Proof. We first prove that if $\mathsf{V}(tx_i, \mathcal{P}(tx_i)) = \mathsf{valid}$ then tx_i is valid. It directly follows from the four checks in Algorithm 1 since they are exactly the conditions in Definition 3.

We then show that if $\mathsf{V}(tx_i, \mathcal{P}(tx_i)) \neq \mathsf{valid}$ then tx_i is not valid. To prove this, we first prove the statement "if $\mathsf{V}(tx_i, \mathcal{P}(tx_i)) \neq \mathsf{valid}$ and $\forall tx_j \in \mathrm{Source}_i, \mathsf{V}(tx_j, \mathcal{P}(tx_j)) =$ valid, then tx_i is not valid."

We prove this statement by contradiction. Assuming that there exists a transaction tx_k such that $\mathsf{V}(tx_k, \mathcal{P}(tx_k)) \neq \mathsf{valid}$ but for all $tx_j \in \mathrm{Source}_k, \mathsf{V}(tx_j, \mathcal{P}(tx_j)) = \mathsf{valid}$, and tx_k is valid.

By our algorithm, at least one of the four checks other than the "**Source Check**" is failed. If the step "**Proof Check**" fails, it suggests that a proof $\mathcal{P}(tx_i)$ does not exist, which contradicts the assumption that tx_i is valid. If the step "**Equality Check**" fails, it contradicts the **Value equality** condition of valid transaction. If the step "**Double-Spending Check**" fails, it contradicts the **No double spending** condition of valid transaction.

We then prove that if $\mathsf{V}(tx_i, \mathcal{P}(tx_i)) \neq \mathsf{valid}$ then tx_i is not valid by contradiction. If this does not hold, then there must exist a transaction that violates the statement proved above. This transaction might be tx_i, the source of tx_i, or recursively one in the sources of the sources. □

Appendix D
Smart Transacting Algorithms

Here we give a naive smart transacting algorithms $\mathrm{Source}_i = \mathsf{ST}(d_i, a_i, \mathcal{C}_u)$ in Algorithm 3 for rational nodes, where node u intends to send an amount of a_i to node d_i in transaction tx_i and $\mathcal{C}_u$ is a collection of all transactions and proofs recorded in node u. Here, we assume that nodes have sufficiently large computation capability so that the computation cost is insignificant comparing to the communication cost.

Algorithm 3 Non-interactive Smart Transacting Algorithm $\mathrm{Source}_i = \mathsf{ST}(d_i, a_i, \mathcal{C}_u)$

```
#Step 1: Check for all unspent transactions
UT ← all unspent $tx_i$ that are in $\mathcal{C}$.
#Step 2: Determine the chains that $d$ already has according to $\mathcal{C}$
Collected ← $\emptyset$
for each $tx_i$ in $\mathcal{C}$ and $d_i = d$ do
    chains$_i$ ← $\{v | \mathcal{B}_v \in \mathcal{P}(tx_i) \cap \mathcal{C}_u\}$        ▷ All chains in the proof of $tx_i$ according to $\mathcal{C}_u$
    Collected ← Collected ∪ chains($i$)
#Step 3: Find the sources which has the least amount of chains to send
for all Source$_n$ ⊂ UT such that the sum amount no less than $a_i$ do
    Proof$_n$ ← union of all $\mathcal{P}(tx_i), tx_i \in$ Source$_n$
    NChains$_n$ ← $\{v | \mathcal{B}_v \in$ Proof$_n\}$
    ToCollect$_n$ ← NChains$_n$/Collected
return Source$_l$ where ToCollect$_l = \min(|$ToCollect$_n|)$
```

Note that Algorithm 3 is a non-interactive algorithm. The choice of the sources is much easier in an interactive fashion, in which the receiver simply tells the sender the chains that he already has once per round. Then, the second step in Algorithm 3 can be omitted. The cost of this communication is $O(gc)$, where g is average number of chains that a node acquires and c is the average number of transacting targets of each node. This cost is no larger than $O(gN)$ and adds no more than $O(g)$ cost to the CCPT for the same reason that we have for global optimization in Subsection IV-B. Both interactive and non-interactive algorithms will result in spontaneous sharding. For a stable network with sufficient transactions been made by each node, either interactive or non-interactive schemes will have similar performance since the transaction pattern is fixed and each node should already have enough prior knowledge for the chains that each receiver has.