Consensus Protocol

Zhijie Ren

January 31, 2017

1 Preliminaries

1.1 TrustChain

Here we consider a TrustChain-like [1] system, all nodes have their own Genesis block and create their own blockchain. The blocks only consist their own transactions. Each transaction is associated with signatures from both parts of the transaction.

1.2 Notations

We consider N nodes, with $\mathcal{C}_i, i \in \{1, 2, \ldots, N\}$ representing their blockchains. A blockchain $\mathcal{C}_i$ is defined as an ordered set $\{B_i(1), B_i(2), \ldots\}$ of blocks, in which block $B_i(k)$ belongs to either of the two types *Transaction Blocks* denoted by $\mathcal{T}$ and *Check Points (CP)* denoted by $\mathcal{P}$. We further denote the k-th appearances of the block that belongs to the type $\mathcal{T}$ and $\mathcal{P}$ by $T_i(k)$ and $C_i(k)$, respectively. In other words, we define $B_i(k) \equiv C_i(l)$ if $B_i(k) \in \mathcal{P}$ and $|\{B_i(j) \in \mathcal{P} : j \in [0, k]\}| = l$ and $B_i(k) \equiv T_i(l)$ is similarly defined. We denote $C_i(l) \leftarrow T_i(k), T_i(k) \leftarrow C_i(m) >$ if $b - a = \min_{a' < b, B_i(a') \in \mathcal{P}}(b - a')$ and $c - b = \min_{c' > b, B_i(c') \in \mathcal{P}}(c' - b)$ assuming that $C_i(l) \equiv B_i(a), T_i(k) \equiv B_i(b), C_i(m) \equiv B_i(c)$. Then, we call $C_i(l)$ and $C_i(m)$ the *previous CP* and the *next CP* of $T_i(k)$, respectively. We call the blockchain in between two neighboring CP's as a *piece* of the blockchain. A transaction from node i to j is represented as $tr(i \to j, s), i \neq j$, where s is the serial number. In this paper, we use the structure similar to Bitcoin [2] for the transactions. For each transaction $tr(i \to j, s), i \neq j$, a set of transactions $\mathcal{S}(tr(i \to j, s))$ should be provided as the *source* of this transaction. As a result, the currency is flowed from one transaction to another, and no actual "balance" is kept anywhere in our blockchain. A cryptographic hash function is denoted by $Y = H(X)$, in which Y is called the *hash result* of X. The public and private keys of node i are denoted by $K(i)$ and $k(i)$, respectively. A message M encrypted by a key $k(i)$ is represented as $E_{k_i}(M)$.

2 Consensus Process

Generally speaking, our protocol separate the consensus process from the validation of each individual transaction. This approach tremendously reduces the communication cost for the consensus process and even achieves an unbounded throughput.

Further, we address the problem of the "overhead of trust", which is the scenario for most of the consensus model. In traditional consensus problems or bitcoin-like systems, all transaction are agreed and validated by the whole community, which costs heavily and happens rarely in the real life scenario. Here, we introduce an alternative consensus model, in which individual transactions are not validated. However, any party is able to easily validate any transaction at any time, and no invalid transactions can be forged into a valid one. In other words, cheats will get caught.

In this section, we first introduce some definitions, then describe our consensus protocol and validation protocol, respectively.

2.1 Definitions

Definition 1 (Transaction Block (TB)) *A transaction block $T_i(k)$ consists of a hash result 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 2 (Transaction Index) *If a transaction is contained in the transaction message (the definition will follow) $t_i(k,m)$, then a vector of $[i,k,m]$ is called the* **index** *of this transaction. Note that since in our system a transaction is written in the blockchain of both the sender and the receiver, hence a transaction could have 2 indexes.*

Definition 3 (Transaction Message) *A transaction message $t_i(k,m)$ consists of the following:*

1. *The transaction, i.e., $tr(a \to b, s), a, b \in \{1, 2, \ldots, N\}, a = i$ and/or $b = i$.*
2. *The indexes of the sources of this transaction.*
3. *A hash of the previous CP, i.e., $H(C_i(l)), C_i(l) \leftarrow T_i(k)$.*
4. *A digital signature created by both parties of the transaction, which is the hash result of the previous three items encrypted with both private keys.*

Definition 4 (Check Point (CP)) *A check point consists of a hash result of the previous block and the result of the consensus process of the r-th round, denoted by $CON(r), t \in \{1, 2, \ldots\}$, which will be specified later, i.e., if $C_i(k) \equiv B_i(j)$, then $C_i(k)$ consists of $H(B_i(j-1))$ and $CON(r)$.*

2.2 Consensus Scheme

Our consensus scheme is used repetitively in *round*. Here, we assume the scheme has been executed in round $r-1$ and results in $CON(r-1)$. Then, we describe our scheme for node i in the r-th round. Note that the transactions are independent of our consensus schemes. Hence, we assume that the transactions are made with a rate R_i for node i and node i continuously writes them to the transaction blocks and appends them to its chain while the consensus scheme is executing.

1. After the reception of $CON(r-1)$, if a CP from node i is included in $CON(r-1)$, it generates a new CP, assume it is $C_i(k)$, and appends it to its chain.

2. If no new CP is generated, node i waits $\tau_{i,0}$ and broadcast its latest CP that does not reach consensus and its previous CP that reached consensus in $CON(r')$. Furthermore, a digital signature is associated with this message. We denote this message by $M' = [C_i(j'), C_i(k'), D']$.

3. If a new CP $C_i(k)$ is generated, it waits τ_i and broadcast this CP and its previous CP that reaches consensus in $CON(r-1)$. Furthermore, a digital signature is associated with this message. Let us denote this message by $M = [C_i(j), C_i(k), D]$.

4. Now the network will reach agreement on a set of pairs of CP's send out by the nodes. A scheme similar to the one used in [3] (will be specified later) is used to agree upon a set of CP's that will reach consensus in this round. The following messages will be excluded from this consensus process:

   (a) The messages with incorrect digital signature.

   (b) The messages M (M') with either $C_i(j)$ ($C_i(j')$) not included in any $CON(t)$ or $C_i(k)$ ($C_i(k')$) has already included in some $CON(t)$.

3 Validation Protocol

3.1 Definitions

First we consider the conditions that a transaction is valid. We first give our definition on the validation of a transaction.

Definition 5 (Valid Transaction) *A transaction $tr(i \to j, s)$ is* valid *if and only if the following conditions hold.*

1. *The transaction is contained in two messages $t_i(m, k)$ and $t_j(m', k')$.*

2. *These messages are correct in the sense that all information required for these messages are correct and properly signed.*

3. *Assume $T_i(k) \equiv B_i(l)$ and $T_i(k') \equiv B_j(l')$. Then, there exists $B_i(n)$ and $B_j(n')$ such that $B_i(n), B_j(n') \in \mathcal{P}$ and $B_i(n)$ and $B_j(n')$ are included in some $CON(r)$.*

4. *The hash results in blockchains $\mathcal{C}_i$ and $\mathcal{C}_j$ from the first block till the blocks $B_i(n)$ and $B_j(n')$, respectively, are correct.*

5. *The source transactions of $tr(i \to j, s)$ are valid.*

Then, we give a definition on the validation of pieces of a blockchain.

Definition 6 (Valid Piece of Blockchain) *The piece of blockchain $\mathcal{C}_i$ from $C_i(k)$ to $C_i(l)$ is said to be valid if one of the following conditions holds*

1. *There is no transactions in between $C_i(k)$ and $C_i(l)$ and the hash results in the blocks from $C_i(k)$ to $C_i(l)$ are all correct.*

2. *There are transactions in between $C_i(k)$ and $C_i(l)$ and all of them are valid.*

3.2 Validation Process

Any node i can validate a transaction $tr(u \to v, s)$ by sending a request to node u or v, the process of which is called a *validation process*. The validation process consists of four parts.

In the first part, node u send the transaction index of $tr(u \to v, s)$ to node i (we only consider the case for node u and skip the case for v due to similarity).

In the second part, if the pieces of blockchains which contains $tr(u \to v, s)$ are not obtained by i, i requests the following:

1. The piece of blockchain which contains $tr(u \to v, s)$.
2. A piece of blockchain from v which contains $tr(u \to v, s)$, which is signed by v.

Both items should be signed by u.

In the third part, node i request all the pieces of blockchains which are not yet validated by it, which might include:

1. All pieces of blockchain of u.
2. All pieces of blockchain of v, which are signed by v.
3. All pieces of other blockchains C_j, which contains a transaction $tr(j \to k, s')$ or $tr(l \to j, s')$ which is the source of $tr(u \to v, s)$, or recursively the source of the sources. These pieces should be signed by their creators.

In the fourth part, if node i has not received a required piece for this validation, it can request it from the owner of that piece, or any other node who might have that piece in its storage.

4 Proofs

To be added.

5 Requirements of the Protocol

5.1 Storage Requirement

For the validation process, all nodes are obligated to store all pieces of the blockchains which might be requested by the validator. In other words, node i needs to store a blockchain $\mathcal{C}_j$ with a signature of j if there exist a transaction $tr(j \to i, s)$ or $tr(i \to j, s)$, or a transaction $tr(j \to k, s)$ or $tr(l \to j, s)$ which is the source of $tr(i \to u, s')$ or $tr(v \to i, s')$, or recursively the source of the sources.

This requirement seems to be quite strong, especially a transaction has two copies in our scheme. However, comparing to most of the blockchain techniques which requires all transactions to be stored in all nodes, our scheme could possibly reduce the storage requirement by a huge amount depending on the network structure and the transactions. The storage requirement is at most at $o(|\mathcal{V}|)$, where $\mathcal{V}$ is the set of all transactions, which is at the same order as most of the traditional blockchain techniques.

5.2 Communication Requirements

For consensus process, it is clear that our scheme is very similar to the one used in [4]. However, our throughput is remarkably higher since no actual transactions are agreed during the consensus process. More precisely, the communication cost is reduced from $o(N^2|\mathcal{V}|)$ in most of the other consensus algorithms to $o(N^2)$ in our scheme. Moreover, since the time consumption of the consensus process is independent of the amount of transactions, we achieves an unbounded throughput. More precisely, although the delay still scales as a square function as N, the throughput will not be limited by N like the most of the other blockchains do.

In traditional blockchain techniques, all transactions in the blockchains are unforgeable, unforkable, and valid, which requires the global agreement on both of the unforgeability and validity of each transaction. In our case, the unforgeability is agreed globally, which prevents forking. However, the validity is checked locally. In other words, individual nodes are responsible for the validation of the transactions, and their validation is not trusted by others.

It seems to result in a flood of communications and wasted duplications of work since a transaction will be individually validated by multiple parties and the trust does not propagate. However, we have 4 good reasons to do that. First, not all transactions are interested by all parties. For example, a transaction $tr(i \to u, s')$ will only be interested to node i, j, and all parties that use this transaction as the source. A global agreement on the validity of this transactions will actually be a huge waste. Second, if all nodes keep tracks of the valid transactions, the overhead in validation process can be minimal. More precisely, by our validation process, if all other pieces required for the validation of a transaction is already validated, the communication will stop at the first step, which contains only a few bits. Also, this is not achieved at the cost of increasing storage requirement since by our structure, keeping tracks of the valid CP's, instead of all transactions, is enough. If all nodes keep tracks of their validations, the communication cost is actually at most $o(N^2|\mathcal{V}|)$, which is the case that all nodes want to validate all transactions. This is again at the same order of reaching global agreement. Third, the communications are most of the time in between two parties, which will reduce the latency due to less waiting and responding time. Also, since our validation is independent of the consensus process and the transactions, the latency in validation will not affect the throughput of our system. Fourth, in fact, this validation is similar to the real-life scenario. For example, a contract is signed with 2 parties only. There is no need for this contract to be agreed globally as long as it is legal, which in our scenario, cannot be forged and can be validated by everyone.

References

[1] J. A. Pouwelse, P. Otte, and M. de Vos, "TrustChain: A Sybil-resistant scalable blockchain," *Future Generation Computing Systems, SI: Cryptocurrency*, under review.

[2] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[3] Cachin C, Tessaro S. "Asynchronous verifiable information dispersal", *Reliable Distributed Systems, IEEE Symposium on. IEEE*, 2005: 191-201.

[4] Miller A, Xia Y, Croman K, et al. "The honey badger of BFT protocols", *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. ACM*, 2016: 31-42.