PROBECHAIN

Global Public Ledger Serving the Probe Ecology Builders

(V1.004 September 26, 2022)

Find us on Probe.Builders, Probe.Network or Github.com/Probechain

1 Introduction

1.1 Vision

Probechain is an important infrastructure of the PROBE.NETWORK. Committed to linking web2 and web3 to become a Global Public Ledger.
Probechain can support fair initial token distribution, high throughput and fast transaction execution and confirmation, and then supports an efficient distributed infrastructure for decentralized communication network, value exchange without trust and other complex Decentralized Applications (DAPPs).

1.2 Background

Bitcoin presented in 2008 is the first blockchain system that Powers a secure, decentralized, and consistent transaction ledger. Value in the system is generated through preset rules that everyone can participate in. Particitants can glean all transactions in it, and these transactions are cryptographically secured using Proof of Work (POW) consensus mechanism, thus preventing double spending. Its fairness, transparency and security have revolutionized society's perception about the implementation method of financial activities and made Bitcoin extremely popular in the last decade.

In late 2013, Ethereum proposed a more powerful network that can support DAPPs through smart contracts and a Quasi-Turing-complete Ethereum Virtual Machine (EVM). Instead of recording a single type of cryptocurrency, Ethereum can support various data-friendly services through recording related code and data. Ethereum extends the boundaries of blockchain technology, making it theoretically extremely powerful.

Though theoretically powerful, low transaction throughput of POW consensus based projects makes applications limited and expensive. Resent projects like EOS and TRON use Delegated Proof of Stake (DPOS) consensus to decrease network's communication burden and thus improve transaction throughput.

Besides DPOS based pro jects are efficient and transactions on them are relatively cheap, history also shows that they are secure. These projects are less popular, and one of the reasons is, we believe, their value distribution rules are not fair enough. Thus, Probechain introduce a hybrid consensus algorithm to satisfy both efficiency and fairness through combining POW, DPOS and Concise Byzantine Fault Tolerance (CBFT) Protocol. Also, it utilizes more efficient verifiable data structure which we call "Set Merkle Tree" to support faster database operations. Based on these innovative solutions, Probechain aims to providing a decentralized large-scale network and supporting lots of valuable applications such as decentralized communication network, issuance and circulation of digital assets and so on.

1.3 Features

1.3.1 Hybrid Consensus Algorithm

Probechain adopts a hybrid consensus algorithm that is a combination of POW and DPOS to balance efficiency and fairness. Most coinbase rewards are generated through POW mining, which makes sure that the initial distribution of PRO (official cryptocurrency of Probechain) is as fair as possible. Block generation frequency is determined by computing POWer of the network and POW mining difficulty. Probechain keeps the block generation speed close to constant in most of the time and slow down the speed in the case of poor network conditions through dynamically adjusting POW mining difficulty.

DPOS is adopted to reduce communication overhead thus improve transaction throughput. The consensus algorithm among DPOS nodes is a variant of Practical Byzantine Fault Tolerance (PBFT) protocol, which we

call Concise Byzantine Fault Tolerance (CBFT). CBFT algorithm takes blocks as the smallest unit to consensus and supposes detectable malicious faults do not occur among DPOS nodes. CBFT can further reduce communication overhead and average transaction confirmation time, see Section 4 for more details about Probechain's consensus algorithm.

1.3.2 Set Merkle-Tree Structure

Merkle Tree (MT) is a concise verifiable data structure, but it is not convenient for insert or delete operation. So Merkle-Patricia Trie (MPT) is used for faster database operation in most blockchain projects. Probechain uses a new data structure Set Merkle-Tree (SMT) to support faster database operation. SMT fully utilizes MT's advantages of fast calculation and convenient verification and modifies the structure to support fast insert and delation operation. Also, comparing to MPT, layered SMT can significantly reduce the number of hard disk reads and writes, thus improves database performance, see Section 3.3 for more details about SMT.

1.3.3 Direct Exchange Between Digital Assets and PRO

Asset trading (i.e., token exchange) is an important topic in blockchain. The realization of transactions can be achieved in both centralized ways (e.g., centralized exchanges) and decentralized ways (e.g., decentralized trading pools). The implementation of centralized ways has credit risks and violates the original intention of blockchain's decentralization, while trading pool based decentralized exchanges rely on algorithms to automatically price and execute transactions, which will inevitably have conflicts between arbitrage (i.e., impermanent loss) and slippage.

Probechain supports a direct exchange between PRO and a digital asset through an asset exchange transaction. This mechanism requires a centralized matchmaker. The matchmaker will first match parties to this transaction, and then collect their signatures, so that the transaction can be executed. In this process, parties to the transaction do not need to transfer their assets to the matchmaker, so they do not need to trust the matchmaker. See more details in Section 5.2.4.

1.3.4 Loss Reporting and Retrieval Mechanism

Probechain proposes a digital asset loss reporting and retrieval mechanism used in blockchain platforms. Traditionally, once an user of blockchain platforms losees the private key corresponding to his account, cryptocurrencies and other digital assets in the account will no longer be usable. In Probechain, when user's private key is lost, it is possible for him to transfer his cryptocurrencies and other digital assets to a newly generated account. Whether an account supports this mechanism or not is preset by its owner, and the owner can also determine the loss reporting period whenever it is allowed. See more details in Section 5.2.6.

1.3.5 Classified Account System

To facilitate the management of account data and realize specific functions (e.g., asset exchange, DPOS nodes voting and loss reporting and retrieval etc), Probechain classifies account into seven types: regular accounts, Probe Name System (PNS) accounts, digital asset accounts, contract accounts, voting accounts and loss report accounts respectively. Each type of account is organized in the form of Set Merkle-Tree. See Section 3.1 for more explanation about account types.

2 Architecture

Probechain adopts a 4-layer architecture, namely Storage Layer, Protocol Layer, Communication Layer and Application Layer. As shown in Figure 1, Probechain is a LevelDB-based public database composed of account information, application data and block data on the bottom. The database is updated according to compliant transactions and reaches an agreement through consensus mechanism. All transactions and consensus messages are spread into the network through a P2P communication protocol. Users can participate in Probe Ecology through various Application Programming Interfaces (APIs), such as wallet, block explore, mining software and other kinds of DAPPs.

3 Account

3.1 Types

There are six types of accounts in the Probechain, namely regular accounts, Probe Name System (PNS) accounts, contract accounts, digital asset accounts, voting accounts and loss report accounts respectively.

Account. An account is composed of address and data field. Address is the key of an account. For all six types of accounts, addresses are of the same format, which is a 25-byte code with the first byte represents the account type and the last 4 bytes derived from the first 21 bytes as verification code.

We use one byte 0x00, 0x01, 0x02, 0x03, 0x04, 0x05 at the begin of an address to represent regular account, PNS account, digital asset account, contract account, voting account and loss report account respectively. For a regular account, the following 20 bytes are derived through hashing a public key and extracting the last 20 bytes in an Elliptic Curve Digital Signature Algorithm (ECDSA); for a PNS account, the following 20 bytes can be obtained from hashing a readable name and extracting the last 20 bytes or directly assigned by the creator; for other account types, the following 20 bytes are obtained through hashing the initiator's account address and nonce and extracting the last 20 bytes. In the following context, we will use *address* to represent a 25-byte account address.

Regular accounts. Regular accounts are used to 1) record uses' balances of PRO and votes for candidate DPOS nodes and 2) initiate transactions. The data field of a regular account is composed of five parts: lossType, nonce, value, voteAccount and voteValue.
PNS accounts. PNS accounts are aliases in Probechain. They are also used to facilitate decentralized communication in Section 6. The data field of a PNS account is composed of three parts: type, owner and data.
Contract accounts. Contract accounts are used to support customized DAPPs and their data field is composed of value, codeHash, storageRoot, voteAccount and voteValue.
Digital asset accounts. Digital asset accounts are actually contract accounts, and we distinguish them for supporting better service of digital assets' issuance and circulation. The data field of a digital asset account is composed of value, codeHash and storageRoot.

It is necessary to record a mapping table between regular account address and the digital asset in a digital asset account. For Fungible Tokens, an *address* => *uint256* mapping "balances" can be used to track

of who owns how many tokens; for Non-Fungible Tokens, an uint => *address* mapping "owner" can be used to track the owner of a specific Non-Fungible Token. It is suggested for digital asset accounts to use "balance" or "owner" in order to support direct exchange between digital assets and PRO.

Voting Account. Voting accounts are used for users to compete to be DPOS nodes that are authorized to generate new blocks and get rewards. Their data field is composed of six parts: owner, info, pledgeValue, vote, validPeriod and state.

Loss report accounts. Loss report accounts record the information of regular accounts that are in the state of loss reporting or successfully retrieved recently to realize the function of loss reporting and retrieval in Section 5.2.6. Their data field is composed of five parts: state, infoDigest, lossAccount, newAccount and lastHeight.

3.2 Creation and Cancellation

In Probechain, an account needs to be created before being used. User needs to initiate a transaction in order to create an account.

When a new account is generated, a certain amount of PRO is consumed which is related to the type of the account. The PRO consumed will be returned to the user when the account is cancelled.

Creation and cancellation mechanism for accounts in Probechain is adopted for two reasons: 1) to save network storage, and 2) to match Probechain's verifiable data structure SMT in Section 3.3.

3.3 Authenticated Data Structure: Set Merkle-Tree

In Probechain, each type of accounts is structured as a new verifiable data structure, which we call Set Merkle-Tree (SMT). SMT is a variant of Merkle-Tree (MT) which can support fast operations for data insertion and deletion.

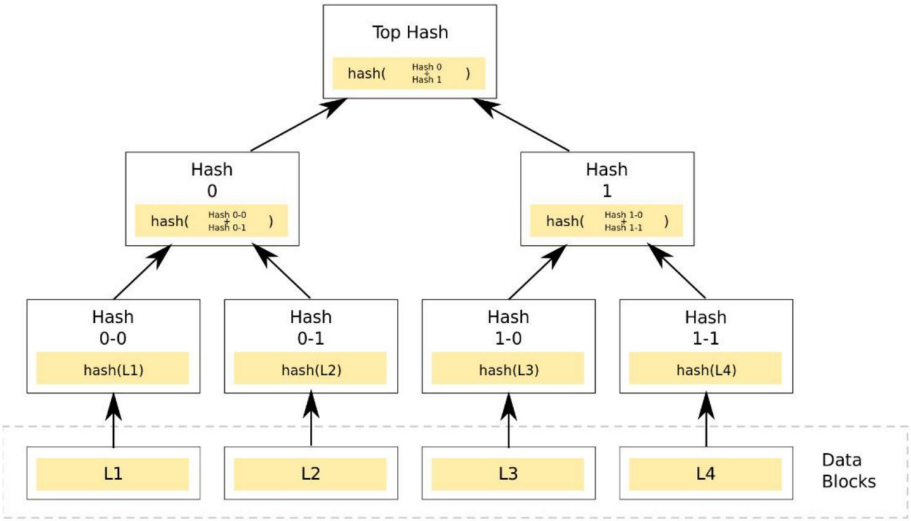
As show in Figure Below, a traditional MT collects elements and treat every element as basic unit to form a verifiable structured tree. It is efficient for element update while not efficient for element insertion or deletion. Since MT needs to reconstruct to a large scale whenever the number or positions of the corresponding elements are changed.

suppose L_i represents a set instead of an element, the it will support fast element insertion or deletion in a set. We first define a hash value of a set S as follows:

$$hash(S) = \sum_{e \in L} hash(e),$$

where the addition means binary addition (i.e., bitwise XOR).

The computation complexities for updating, inserting and deleting an element in SMT are the same and all efficient. Compared to the most popular data structure Merkle-Patricia Trie (MPT) used in blockchain systems up to date, SMT is more efficient in two aspects: 1) the hash computation is smaller for database operations; 2) layered structure makes it possible to keep all intermediate hash values in RAM for large database and significantly reduce the number of hard disk reads and writes.



Interpretation diagram for (Set) Merkle-Tree. If data block $L_i (i = 1, 2, 3, 4)$ represents an element, it is a traditional MT; if L_i represents a set of elements, then it is a SMT.

4 Probechain's Consensus

Proof of Work (POW) mechanism is fair for users to participate in blockchain network and profit from their participation, while Delegated Proof of Stake (DPOS) mechanism is a reasonable way to limit the number of participants in the consensus and thus reduce communication overhead. Combining POW and DPOS can balance fairness and network throughput: Most coinbase rewards can be earned by POW nodes and consensus is reached among DPOS nodes to ensure high throughput.

Probechain adopts a hybrid consensus protocol of POW and DPOS to balance its fairness and efficiency. POW mechanism provides the main method of initial distribution of PRO and POW mining difficult is used to adjust block generation speed. For DPOS nodes, Probechain adopts a Concise Byzantine Fault Tolerance (CBFT) protocol presented in Section 4.1 to reach consensus among them. Compared to Practical Byzantine Fault Tolerance protocol (PBFT), CBFT fully utilizes blockchain's structure and mortage of DPOS, resulting in reduced communication and better network liveness. CBFT can also largely reduces transactions' average confirmation time.

4.1 Concise Byzantine Fault Tolerance (CBFT) Protocol

Compared to the traditional form of state machine replication, blockchain system usually treats blocks as the smallest unit to reach agreement instead of requests (i.e., transactions in blockchain terminology). It is important for blockchain systems' efficiency, since once consensus can contain tens of thousands of transactions.

Notice that you need to pledge a lot of PRO to become a DPOS node in Probechain, it is reasonable to suppose DPOS nodes will not do easily detectable evil behaviors. Specially, posting contradictory opinions for the same block height is necessary an act of evil for the network, and it will be discovered inevitably. So, Probechain adopts the following Assumption 1 and key consensus rules of CBFT protocol.

Assumption 1 *Any DPOS node posts no more than one opinion at a specific block height. That is, a DPOS node may generate (also commit) or not generate a new block but cannot generate two different blocks at the same block height whenever it is his turn. Other DPOS nodes may commit a block (when a honest node receives a valid block) or reject it (when a honest node fails to receive a valid block) for a specific block height but can not do both.*

As in PBFT, CBFT suppose at most f nodes out of a total of $3f+1$ nodes are simultaneously faulty. For a specific height, we call the node responsible for generating a new block the Proposer and other nodes the Confirmers. A block height is called skipped if it is marked Rejected (including at least $f+1$ rejections) by a subsequent block, otherwise it is called a substantial height and the corresponding block is a substantial block. Under Assumption 1, CBFT consensus rules are straightforward:

1. Confirmers start a timer for each block height, and commit a valid block (generated by the Proposer) if received one within the scheduled time, otherwise reject this height. Both commitment and rejection are achieved through digital signatures.
2. Proposer can generate a new block based on a substantial block of a lower height with at least $f+1$ commitments and if the block heights are not consecutive, at least $f+1$ rejections are needed for each skipped block. A block is always committed by its Proposer.

We call a block stable if it receives at least $2f+1$ commitments or it is succeeded by a substantial block that receives at least $2f+1$ commitments. Under Assumption 1, transactions in a stable block are irreversible.

4.2 Hybrid Consensus Mechanism: POW + DPOS + CBFT

In this subsection, we show how to implement Probechain's hybrid consensus through combining POW, DPOS and CBFT. CBFT is the core of Probechain's hybrid consensus. DPOS is used to choose consensus nodes and POW answer and difficult is used to support signal to start timing and adjust block generation speed respectively. In order to describe the protocol, we use the following symbols:

- n, f: the number of DPOS nodes and upper limit of faulty nodes respectively, with $n = 3f + 1$.
- D: the fundamental POW difficulty which is periodically adjusted.
- COMMIT (REJECT): the action of commit (reject) a block (a block height) or the commitment (rejection) from DPOS nodes.
- $\#sig(h) \in [f + 1, 2f + 1]$: the number of consistent COMMIT (REJECTs) collected for a block of height h (block height h) in a subsequent block.
- For block height h, we define h' as the largest substantial height smaller than h.
- D_h: the difficulty of POW puzzle defined according to a block of substantial height h, where

$$D_h = D * \frac{3f + 1 - \#sig(h')}{f} \in [D, 2D],$$

  where h' denotes the largest substantial block height that is smaller than h.
- t: the time interval for Confirmers to wait before rejecting a block height.
- a(h): a POW answer defined by a block of substantial height h.

We describe Probechain's hybrid consensus beginning from a block with a substantial block height H as in Algotrithm 1.

At Step 1, once B_H is generated, the Proposer will commit it, broadcast the block header in the network and broadcast the block among DPOS nodes. The block header contains at least $f + 1$ (and no more than $2f + 1$) COMMITs for B_H and at least $f + 1$ (and no more than $2f + 1$) REJECTs for each height from H' + 1 to H - 1 (if there exists).

At Step 2, POW miner will try to solve POW puzzle once received the block header of B_H and verified its validity and POW difficulty is determined through Equation D_h above.

At Step 3, Confirmers of height H will verify the validity of B_H once receiving it and commits it if it is valid. Confirmers will also communicate B_H with other DPOS nodes if requested and broadcast their COMMITs in the network.

At Step 4, POW miner broadcasts the answer $a(H)$ in the network once succeeded.

At Step 5, a Confirmer for block height H+1 starts a timer waiting for block B_{H+1} if he have sighed COMMIT for B_H, received at least other f COMMITs for B_H and received a POW answer $a(H)$.

At Step 6, after received B_H, more than f COMMITs from other DPOS nodes for B_H and a valid POW answer $a(H)$, Proposer of block height H+1 generates block B_{H+1}.

At Step 7, every Confirmer for block height H+1 sighs COMMIT for B_{H+1} if he received it and verified its validity in time t; otherwise he signs REJECT for height H+1.

Later at Step 8, a Confirmer for height H+2 starts a timer waiting for block B_{H+2} on the condition that he reaches a local consensus about height H+1 and received another POW answer (a(H) if height H+1 rejected, otherwise a(H +1)).

At Step 9, Proposer of block height H + 2 generates B_{H+2} once received enough consistent opinions back up to the previous block of substantial height.

In case of faulty nodes, Proposers of block height H+1,H+2,... may fail to generate valid blocks. Then

Confirmers need to repeat Step 8 and Step 9 for higher block height until a valid block is generated. Under the assumption that faulty nodes are no more than f, the number of continuously skipped blocks will not exceed f.

Notice a block contains at least $f + 1$ consistent opinions about each block back up to the previous substantial block (e.g., of not skipped height) and there is a total of at most $3f + 1$ opinions for each height under Assumption 1, Probechain may accept two different state for a specific block height: confused state with at least $f + 1$ REJECTs and at least $f + 1$ COMMITs at the same time.

Then, neither state gets supports from $2f + 1$ DPOS nodes, which may be caused by bad network condition and we will increase the POW difficult in Equation D_h above to decrease block generation speed. In order to reach an agreement as quick as possible when Probechain forks, DPOS nodes are always suggested to choose the branch with the higher substantial block. Once a substantial block received $2f + 1$ commitments, it is irreversible.

Algorithm 1 Probechain's Hybrid Consensus

Input: A valid block B**H** with substantial height H, predetermined n DPOS nodes and their block generation sequence and fundamental POW difficulty D for this period

Output: A proceeding block

1. Producer of B**H** sigh COMMIT for B**H**, broadcasts the block's header and his COMMIT in Probechain and broadcasts the block only among DPOS nodes;
2. Miners compute
   $D_H = D * \frac{3f+1-\#sig(H'))}{f}$
   and compete to find an answer $a(H)$ defined by the header of B**H** ;
3. Every Confirmer of block height H verifies the validity of B**H** and once confirmed, he sighs COMMIT for B**H**, broadcasts the block among DPOS nodes and broadcasts his COMMIT in the network;
4. Once solving the POW puzzle, the node broadcasts POW answer a(H) in the network;
5. A Confirmer for block height H+1 starts a timer waiting for block B**H+1** and broadcast B**H** in the network if he have sighed COMMIT for B**H**, received at least f COMMITs for B**H** (excluding his own) and received a POW answer a(H);
6. After received B**H**, at least $f+1$ COMMITs for B**H** and a valid POW answer a(H), Proposer of block height H+1 generate block B**H+1**;
7. Confirmers for block height H+1 sigh COMMIT for B**H+1** if they received it and verified its validity in time t; otherwise they sign REJECT for height H+1;
8. Confirmers for height H+2 start a timer waiting for block B**H+2** under one of the conditions:
   - have signed COMMIT and received other f COMMITs for B**H+1** and received a POW answer a(H+1); or
   - have received $f+1$ REJECTs for height H+1 and another POW answer a′(H);
9. Proposer of block height H+2 can generate B**H+2** under one of the conditions:
   - based on B**H+1** if he have verified B**H+1** and received at least $f+1$ Confirmers' COMMITs for B**H+1** and a POW answer a(H +1); or
   - based on B**H** if he have verified B**H** and received at least f + 1 COMMITS for B**H** and at least $f+1$ REJECTs on height H+1 and two POW answers a(H) and a'(H);
10. Continue Step 8 and Step 9 with higher block height until a block is generated and accepted.

5 Block

Blockchain can be regarded as a public database whose initial state is set in the genesis block and it is operated in a sequence through instructions recorded in ordered blocks according to preset operating rules. In order to facilitate users to obtain data in the database and quickly verify its authenticity, block is usually structure as block header and block body. Block header contains a succinct state of the database, digest of transactions in the block, information indicating authorization of the block producer and other information about variable system parameters.

In this section, we introduce the content of a block header in Probechain and list novel transaction rules in Probechain.

5.1 Block

5.1.1 Block Header

In traditional POW blockchain, a block header is used to identify a particular block (and a particular database state) on an entire blockchain and is hashed repeatedly to create POW for mining rewards. A blockchain consists of a series of various blocks that are used to store information related to transactions that occur on a blockchain network. Each of the blocks contains a unique header, and each such block is identified by its block header hash individually. For Probechain, besides POW, a valid signature from DPOS nodes is also need and some variable parameters are all included in Probechain' block header.

In detail, a block header contains such data as follows:

- parentHash: the blockHash of hash value of the parent block.
- height: current block height.
- state: current state of the database.
- txHash: state root of transactions in the block.
- receiptHash: state root of transaction receipts in the block.
- timestamp: the unix timestamp for when the block was collated.
- difficulty: the POW difficulty of current block.
- gasLimit: the upper limit of gas for this block.
- gasUsed: gas used in this block.
- bloom: bloom filter of transaction logs for fast query.
- Proposer: address of the block generator for this block.
- historyState: Merkle root of history blockHashes for all past blocks for fast verification of light clients.
- POWAnswers: a list of POW answers for this block and skipped blocks.
- uncleAnswerHash: hash of POW answers for blocks with the most recent five heights. It is recorded to support compensatory rewards for POW answers not used to generate blocks.
- ConfirmSigHashes: digest of confirmation signatures from DPOS nodes for parentHash and skipped heights.
- ConfirmCounts: the list of the numbers of signatures collected for parentHash and skipped heights[2].
- signature: the signature of Proposer for the contents of the above fields.
- extra: record extra data from Proposer.
- mixDigest: used to define POW puzzle.

[2] We may append zero-knowledge proofs for nodes to verify the correctness of the signatures.

5.1.2 Block Body

Block body contains transactions and confirmation signatures for last substantial block and skipped heights if any.

5.2 Transaction

Transactions are signed messages originated by a regular account, transmitted by the Probechain network, and recorded on Probechain. In Probechain, we add a transaction type for different kinds of transactions in order to support more diverse functions instead of unifying them into a consistent format.

Transactions can be initiated by regular accounts with valid digital signatures or by a digital asset account and contract account through built-in functions. We only describe transactions initiated by ordinary accounts, and other conditions are similar except that signature, transaction serial number, gas price and gas limit are not needed.

5.2.1 Account Creation and Cancellation

As pointed in Section 3.2, an account needs to be created before being used. User needs to initiate a transaction in order to create an account. When a new account is generated, a certain amount of PRO is consumed which is related to the type of the account. The PRO consumed will be returned to the user when the account is cancelled. Account creation transaction is defined as:

$$(txType = 0x00,\ fromAddr,\ newAccount,\ nonce,\ data,\ gasPrice,\ gasLimit,\ sig)$$

For regular accounts, PNS accounts, voting accounts, and loss report accounts, account cancellation is implemented through direct valid signatures. The format is

$$(txType = 0xff,\ account,\ owner,\ beneficiary,\ nonce,\ gasPrice,\ gasLimit,\ sig)$$

where, account indicates the address of the account to be cancelled, and the creation consumption (and available balance if any) is released to specified beneficiary account.

The cancellation of digital asset accounts and contract accounts is realized by the built-in contract function and if there is on such a function, accounts can not be cancelled. We need to specify the beneficiary account in the cancellation function.

We also defined no-authorization required cancellation for voting accounts and loss report accounts when the current block height is much bigger than validPerid of a voting accounts or lastHeight of a loss report account. For such network-friendly transactions, there is no need to pay gas. The format is $(txType = 0xfe,\ account)$, and the beneficiary is automatically designated as the owner of a voting account or the newAccount of a loss report account.

5.2.2 Transfer Transaction

Transfer transactions refer to increasing the value of an account by reducing the value of another account by the same amount. The corresponding account needs to have value attribute (e.g., one of a ordinary account, an asset account, or a contract account). Transferring value from a regular account needs valid digital signature, and the format is defined as below:

$$(txType = 0x01,\ fromAddr,\ toAddr,\ nonce,\ value,\ gasPrice,\ gasLimit,\ sig)$$

5.2.3 Contract Call

A contract call refers to an account calling a contract account or a digital asset account, invoking the corresponding code and updating the stored data. A contract call can be initiated by a regular account as below:

$$(txType = 0x02, fromAddr, toAddr, nonce, data, gasPrice, gasLimit, signature)$$

5.2.4 Exchange Transaction

Exchange transaction refers to exchanging a digital asset with PRO through transaction messages signed by both parties. Here we suppose the digital asset amount in a digital asset account is recorded in a array of key-value pair with key is a regular account address representing its owner and value representing the number of the digital asset, then the format of an exchange transaction can be defined as:

$$(txType = 0x11, add_1, add_2, add_3, value_1, value_2, h, nonce, gasPrice, gasLimit, sig_2, sig_1)$$

which indicates that owner add_1 and add_2 would like to make a transaction: exchange add_1's $value_1$ and add_2 's $value_2$ digital asset defined in contract account add_3 only before block height h. The transaction is submitted by add_1 with his transaction *nonce* and signature sig_1. The transaction message also includes the digital signature of add_2 (i.e., sig_2) in order to ensure the legality of the transaction.

5.2.5 Vote

A voting account can be created as describe in Section 5.2.1. Other nodes can vote for this account and the owner can participate in competition to be a DPOS node.

Voting for an account. Users can vote for a voting account through a voting transaction, a regular account initiated voting account is defined as:

$$(txType = 0x21, from, voteAddr, value, nonce, gasPrice, gasLimit, sig)$$

At a certain time, each account can only vote for one voting account.

Become a candidate DPOS node. The owner of a voting account can apply to be a candidate DPOS node if received enough votes through a transaction:

$$(txType = 0x22, voteAddr, ownerAddr, nonce, data, gasPrice, gasLimit, sig')$$

or updating votes or data information in candidate list through

$$(txType = 0x23, voteAddr, ownerAddr, data, nonce, gasPrice, gasLimit, sig')$$

5.2.6 Loss Reporting and Retrieval

In Probechain, users can retrieve their PRO and other digital assets when they lost private keys if the corresponding regular account's lossType is set non-zero. Four steps procedures are used to realize this function.

Initiate a loss report for an account. First, user need to report a loss without revealing the actual account.:

$$(txType = 0x31, from, mark, infoDigest, nonce, gasPrice, gasLimit, sig')$$

where *mark* is the last 12 bits of lost account and info is actually a digest of the transaction message to be revealed.

Reveal the transaction message. After the loss reporting is recorded on blockchain, user can reveal the actual loss report information:

$$(txType = 0x32, lossAddr, oldAddr, newAddr, nonce, value, gasPrice, gasLimit, sig)$$

It is requested that the *infoDigest* of *lossAddr* is actually the Kaccak256 hash of the transaction.

Transfer the PRO in a lost account. Once lost report information's existence exceeds the time required for the lost account, user can transfer the PRO in lost account to the new account through a special transfer transaction.

$$(txType = 0x33, from, lossAddr, nonce, gasPrice, gasLimit, sig)$$

Update the address in digital asset accounts. Once a loss report transaction is succeed, we actually transfer the PRO in the lost account to another account. We can also transfer digital assets from the lost account to the new account.

$$(txType = 0x34, from, lossAddr, assetAddr, nonce, gasPrice, gasLimit, sig)$$

Reject a loss report. A malicious user may refuse revealing loss report information after initiating a loss report or report the loss of another user's account. So we need to cancel loss report information after initiating a loss report long enough without revealing the actual account or succeeding for enough time:

$$(txType = 0x3f, lossAddr)$$

and reject a loss report for an account without private key lost:

$$(txType = 0x3e, from, lossAddr, nonce, gasPrice, gasLimit, sig)$$

where the *from* address needs to be the lost report account address *lostAddr*.

6 The PNS and P2P Communication network

An introduction to Probe Name System (PNS). On the traditional internet, the Domain Name System (DNS) allows us to use human-readable names in browser while resolving those names to IP addresses or other identifiers. Similarly, on the Probechain blockchain, PNS solves the problem in a decentralized manner.

PNS is a fundamental DAPP itself, offering a decentralized name service. It supports name registration, management and cancellation of PNS accounts. The identifiers behind readable names may represent different meanings, such as an account address in Probechain or public information used for other DAPPs. We use the attribute "type" in a PNS account to distinguish different functionalities as show in Section 3.1.

Storage on Probechain: account addresses and name owners. PNS operates on account addresses instead of human-readable names: a readable name is converted to a PNS account address using hash algorithm and address generation algorithm. Specifically, the first byte of a PNS account address is fixed as $0x01$, the following 20 bytes are derived through hashing the readable name and extracting the last 20 bytes, and last 4 bytes are used as verification code. When referencing a PNS account, users can either use the account address directly or the readable name used to derive the account address instead.

There is an owner expressed as a regular account address for each PNS account. The owner can update information, type or the owner for a PNS account (through a valid digital signature).

Peer-to-Peer (P2P) communication network. There are two mainstream instant messaging architecture forms at present. One is the Client/Server (C/S) architecture, where the user needs to download and install the client software during use. The other is the Browser/Server (B/S) architecture, that is, this form of instant messaging software directly uses the Internet as a medium, and the client does not need to install any software.

Server-based instant messaging face the potential risk of leaking user data, especially under the premise that the government prohibits the provision of encrypted communication services. Traditionally, it is inconvenient for users to build a P2P communication network since their IP addresses change frequently. Using PNS as the startup engine, P2P instant messaging can be well realized and thus a huge P2P worldwide communication network can be constructed.

A P2P communication software can be used for users to communicate with each other without a server's aid through connection each other through IP addresses and application port numbers. The difficulty comes from how to obtain the other party's IP address. PNS gives a solution: users record their necessary information to connect each other in a PNS account and update it when needed. Combining PNS and a P2P communication software gives a method to construct a communication network without server. Also, the communication data can be encrypted through a key agreement procedure using data in PNS accounts before being transferred to further protect users' communication privacy.

7 Digital Asset

Digital Asset in Probechain. One major goal of Probechain is to efficiently realize the issuance and circulation of digital assets. Digital assets administered on blockchains are abstractions which symbolize virtual currency, or represent physical assets or financial assets in real life. Although essentially smart contracts, digital assets in Probechain are specially organized as digital asset accounts for better realization. Each digital asset account contains two parts: 1) a mapping table between account addresses (i.e., regular account addresses) and the corresponding asset account, 2) update rules for the mapping table.

Issuance and Circulation. As a token on Probechain, a digital asset has its own issuance and circulation mechanism. Compared with native currency in blockchain, which use POW or POS mechanism for issuance and signed transactions for circulation, the issuance and circulation of digital assets in Probechain are more flexible. Related operation rules are generally specified in contract code by the creator of the contract, such as digital asset's generation, destruction and transfer, etc.

To make trading of digital assets in Probechain more flexible and convenient, we also use exchange transactions in Section 5.2.4 to support directly exchanging between digital assets and PRO.

Contract Standards for Digital Asset. Digital asset contracts have lots of common attributes and functions, such as token generation, destruction, circulation, query and access control, etc. To standardize those common features, Ethereum developers have supported several token standards in Ethereum Improvement Proposal (EIP).

Among these standards , ERC-20 and ERC-721 are most widely used standards. The ERC- 20 introduces a standard for Fungible Tokens, in other words, they have a property that makes each token be exactly the same (in type and value) of another token. While the ERC-721 introduces a standard for Non-Fungible Token (NFT), that is, this type of token is unique and can have different value than another token from the same contract, maybe due to its age, rarity or even something else like its visual.

The Probe Virtual Machine (PVM) is modified from the Ethereum Virtual Machine (EVM), and can support all contract specifications on Ethereum network, including all token standards.

Bridge between digital assets and the real world. Transforming physical assets or financial assets into digital assets on blockchain can greatly facilitate the investors and financiers of the transaction, and increase the transparency of the assets' circulation. Two major challenges hinder large-scale implementation of digitizing assets on blockchain platform: 1) it is difficult to ensure the delivery of physical assets or the redemption of financial interests, 2) offering securities (especially equity) to the public is a regulated activity in most jurisdictions.

It is possible to develop off-chain standards in order to overcome the above difficulties. Probechain adds an attribute "description" for digital assets, which is the digest of the contract description, credit endorsements (e.g., audits issued by accounting agencies, credit ratings of rating agencies, and so on) or issuance authorization related to digital assets. It can better link digital assets in blockchain with physical assets or financial securities in the real world. In case the issuers of digital assets fail to fulfill their obligations, the holders of digital assets can take legal measures to exercise their rights or realize the corresponding benefits based on "description" of digital assets on Probechain and related documents as evidence.

8 Participate in Probe Ecology

The total amount of PRO will be **1 billion**, of which **93%** need to be generated with block, and **7%** are used to activate Probe Ecology.

Users can participate in Probe Ecology in a variety of ways, such as POW mining, generating blocks through DPOS election, service provider for matching transactions or historical data query and building personalized DAPPs.

Mining. People can earn most coinbase rewards of PRO through POW mining. Most PRO are generated through POW mining that anyone can participate in.

Block Generation. People can pledge their PRO and get votes through paying a certain cost and appropriate marketing to compete to become DPOS nodes. Then DPOS nodes can generate blocks cooperatively and earn transaction gas fees and part of coinbase rewords. DPOS nodes determine the security of Probechain and we always suppose the number of faulty nodes is less than $1/3$ the total number of DPOS nodes.

Service provider. To better realize applications in Probechain, people can provide some centralized services to improve efficiency and convenience of applications, such as transaction matching service for digital asset, support fast historical status query through running an archive node, assisting in realization of the initial handshake for point-to-point communication and auxiliary services for other DAPPs.

Deploying DAPPs. Probechain can support fast state transformation with high performance which is important for active DAPPs. Developers can build various projects through deploying smart contracts on Probechain and benefit from its efficiency.

Common users. Most users can interact with Probechain through simply sending transactions to transfer PRO, call smart contracts or vote for DPOS candidates. These users can run a light node client and query (verifiable) useful data from a full node.

9 Conclusion

Probechain is a scalable blockchain solution that has employed innovative methods for tackling challenges faced by legacy blockchain in the following aspects:

1) **It** is an important infrastructure of the PROBE.NETWORK. Committed to linking web2 and web3 to become a Global Public Ledger;

2) **It** balanced transaction throughput and fairness of initial PRO's distribution through adopting a hybrid consensus mechanism of combing POW and DPOS;

3) **It** largely reduced average transaction confirmation time through CBFT consensus algorithm among DPOS nodes;

4) **It** simplified database operations through new authenticated data structure SMT;

5) **It** reduced transaction cost between digital assets through supporting direct exchange between PRO and digital assets (e.g., tokens);

6) **It** proposed a digital asset loss reporting and retrieval mechanism used in blockchain platform.