Tera Hypernet Whitepaper

Contents

Abstract

With blockchain technology growing in popularity and adoption become more widespread, more applications have utilized the benefits of blockchain and decentralized ledger technology than ever before. Unfortunately, no fundamental progress has been made in a consensus algorithm that could engender trust and widespread use beyond niche industries. The current blockchain technology is compromised by consensus algorithms and there is a huge waste of computing power and other resources due to inefficiencies within their respective systems.

Tera Hypernet introduces a brand-new consensus algorithm, which solves the aforementioned predicament that has troubled blockchain technology development for a long time and stunted the progress of blockchain technology for years, bringing about two new changes. Firstly, it improves the processing ability of the blockchain system. Secondly, it combines powerful and huge computing resources while carrying blockchain applications to build a hyper-computing network in order to help blockchain technology further evolve. We will inherit all the advantages of existing technology, and improve the existing business of blockchain to open up a brand new world that could only be imagined before.

Back to Probability Right

The new era comes with new features of infrastructure. When it comes to blockchain, after more than a decade of development, and the underlying consensus algorithms are constantly evolving to adapt to the features of the basic network facilities of the new era. The traditional blockchain system based on proof-of-work also has its underlying technical limits. The probability right of each node is determined by the amount of computing power in an oversimplified way, and hence creating a monopoly of facts.

The limits of technology lie not in technical limitations, but in the inferences based on rigorous mathematics. If we want to lead blockchain into a new era, we need to break the shackles of “chain” and “block”, return to the origin of bit, and reconstruct “coding and decoding” in the bit probability cloud to create a new consensus mechanism. We shall also break ourselves from the narrow competition in hashing power, and make innovation in the mechanism at the source of access. We have adopted the method of probabilistic election, which is followed by bookkeeping, and hope to find a better solution between efficiency, probability and computing power to adapt to the features of network facilities in the current era.

We adopt the core algorithm of probability negotiation, and use a proof-of-probability consensus mechanism to tackle the root of problems. It boosts efficiency with both security and scalability. Verification as needed is also a method to release more computing power and bandwidth. The method of probability game is used to prove as needed that it could improve the efficiency of the network on the basis of zero-knowledge.

Crypto-Identity and Tera Code Activation

If the mining machine is regarded as an individual Crypto-Identity, its value is beyond a mere competition of computing power. “Time, computing power, storage, bandwidth, witness, pledge” all have their value, and corresponding returns should be rewarded in the system. A system representing machine rights will eventually appear. If we could explore the potential value of resources like” time, computing power, storage, bandwidth, witness, pledge” and let it solve the problems of speed, bandwidth, application, and security in the network, we will be able to make good use of everything eventually. In this way, the value of HyperNet can be interpreted as tens of thousands or millions of Crypto-Identities, whose potential resource value is fully mobilized.

How can we ensure that each Crypto-Identity has unique identity rights and can effectively participate in the blockchain network system based on the Proof-of- Probability(POP). Initially, we will provide an open source Tera code generation program, and add a TPM module to each mining machine to ensure physical uniqueness. At the same time, the mining node with the Tera code needs to register its public key on the block via a smart contract. Each block can only activate one Tera code, and the bookkeeping node has the right to choose the best bid among the node applications to be activated. And when the node is activated on the chain, it also has the same approval rights and becomes a Crypto-Identity with independent identity and equal power.

Proof-of-Probability Consensus Mechanism

The evolution of the world starts from the bottom. The same rule applies to science and technology as well as financial markets. Root problems in the current world remain unfixed. The root problem of traditional currencies lies on the basis of its function-trust.

Numerous technical elites began their exploration of decentralized trustworthy mechanisms as early as decades ago. From Chaumian blinding to David's B-money and Adam Back's hash cash, the solution to the core problem is to find an achievable decentralized consensus mechanism to solve the uniqueness and reliability of the information recorded in the ledger, so as to establish a decentralized network of trust machine.

The blockchain technology based on proof-of-work could be simply understood that at first, different nodes of the entire network do the bookkeeping, and then hashing power is used to compete for the legitimacy of bookkeeping jointly. This method has been proved to be feasible for the collaboration of non-large-scale system nodes. Nonetheless, it also reveals some flaws. For example, the limit of speed makes it difficult to meet large-scale commercial applications, and the computing power of the entire network node will generate a large amount of repetition and waste. At the same time, as the system nodes increase regularly, the system and communication performance will decrease. The performance, stability and scalability of the system are facing huge challenges.

We aim to improve it via reaching consensus by higher efficiency, and then use a probability negotiation algorithm to choose a batch of credible nodes, followed by bookkeeping the selected nodes. The results shall be verified as needed to reduce the load of the entire network. The computing power is liberated to jointly work at completing the tasks of the application layer, and finally generate a hyper-computing network in the new era.

The efficiency, security and scalability of the system can be well balanced in the network. Once the underlying consensus mechanism is innovated, we could free up computing power, bandwidth and storage to enable the system to adapt to large-scale applications.

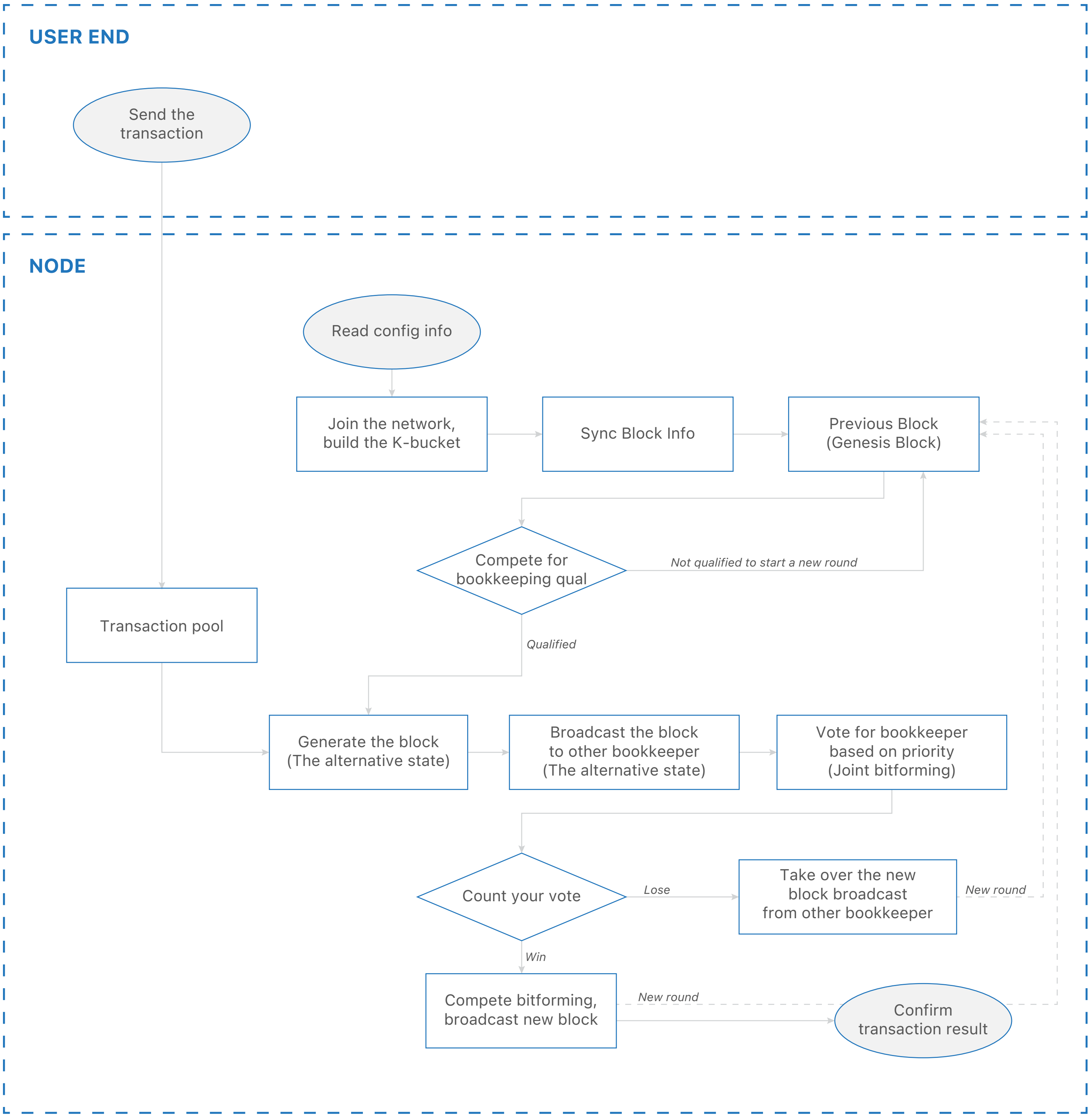
USER END
Send the transaction
NODE
Read config info
Join the network, build the K-bucket
Sync Block Info
Previous Block (Genesis Block)
Compete for bookkeeping qual
Not qualified to start a new round
Qualified
Transaction pool
Generate the block (The alternative state)
Broadcast the block to other bookkeeper (The alternative state)
Vote for bookkeeper based on priority (Joint bitforming)
Count your vote
Lose
Take over the new block broadcast from other bookkeeper
New round
Win
Compete bitforming, broadcast new block
New round
Confirm transaction result

Fig.1 Flow Chart of Proof-of-Probability Consensus Mechanism

Probability Negotiation

We adopt pre-screening on the probability of the nodes of the entire network that obtain the bookkeeping rights. Instead of simply relying on computing power for competition, a batch of candidate nodes are randomly negotiated to average the block generation probability of the nodes. This is the core of our probability negotiation algorithm.

To break it down, each node uses its private key to sign with the hash of the previous block and round value to generate encrypted data related to the node and the block. And then get the hash of encrypted data. Each node shall judge the

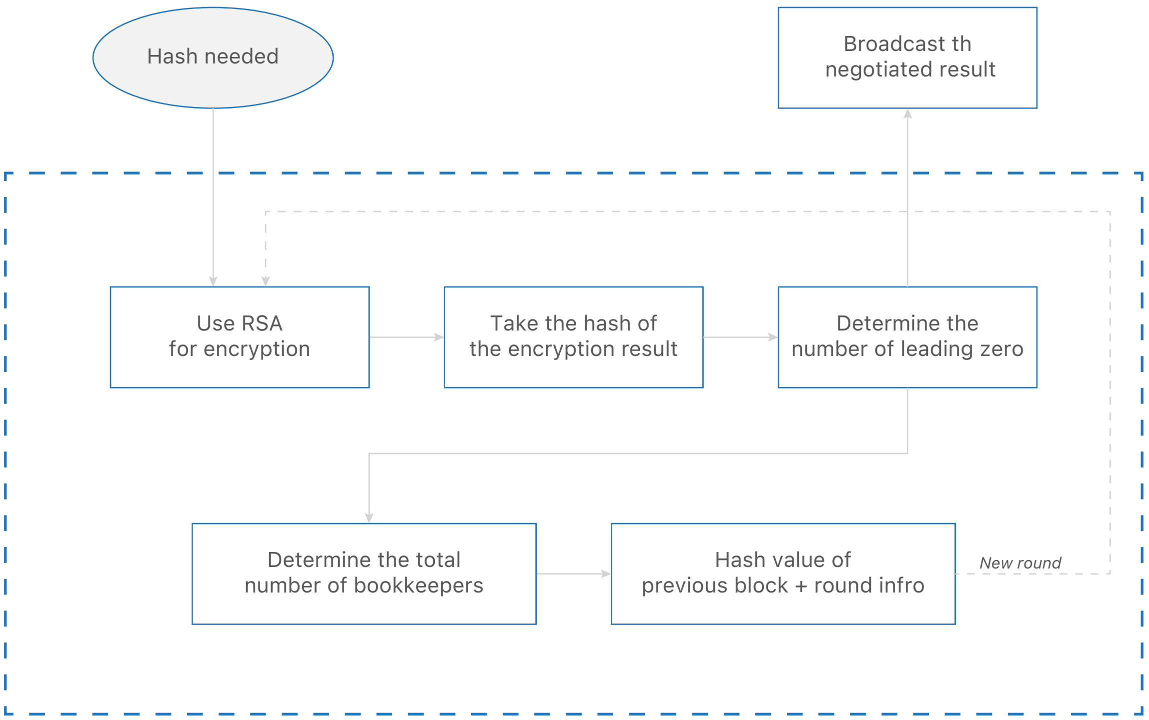
Hash needed
Broadcast th negotiated result
Use RSA for encryption
Take the hash of the encryption result
Determine the number of leading zero
Determine the total number of bookkeepers
Hash value of previous block + round infro
New round

Fig.2 Flow Chart of Probability Negotiation Algorithm

number of leading zero of the hash. The number of leading zero of the hash obtained by the encrypted data of each node is different and are distributed randomly. According to the total number of nodes in the entire network, the system will specify the number of leading zero to filter a batch of nodes as a potential bookkeeping node.

Round value: starting from 1, each node linearly increases the round value to calculate until the final calculation result meets the number of leading zero, unless there is a result produced by probability negotiation in the network.

A node with its hash of encrypted data meets the required number of leading zero will broadcast on its own, announcing that it has become the selected node for bookkeeping. To be honest, other nodes are easily used to verify the authenticity of such qualification because the public key information of the node's corresponding identity is already on the chain. The encrypted data broadcasted by a node can be verified by other nodes.

Bitforming

After probability negotiation, a batch of candidate bookkeeping nodes will be generated in the entire network according to the priority of the round value (the number of candidate nodes varies according to the size and efficiency of the network node). Each candidate node in this batch will broadcast itself a candidate for bookkeeping qualification to the entire network. Each node can verify whether it is cheating.

Each candidate bookkeeping node is a candidate and also a voting node in the voting committee. Meanwhile, the voting committee consists of each candidate bookkeeping node in the batch. Each bookkeeping node has the right vote for the bookkeeping node with the highest round priority in the batch, and finally, select the bookkeeping node that finally generates the block by a majority of votes. Because each candidate bookkeeping node broadcasts candidate bookkeeping qualification to the entire network, all nodes in the entire network maintain a current voting committee list.

After each candidate bookkeeping node broadcasts its candidate bookkeeping qualification, it starts to collect transactions into a block. After the bookkeeping is completed, the candidate bookkeeping node will use its own private key to sign the hash of its own block to be shortlisted to generate encrypted data, and then broadcast it to each voting committee node (that is, all candidate bookkeeping nodes in this batch)

Each voting committee node will take the hash of the received encrypted data of each candidate bookkeeping node within a specified time, calculate the number of leading zero of each hash, and select the one with the biggest number of leading zero(including the judgment of the encrypted data of its own block to be shortlisted), and then sign the hash of the selected candidate bookkeeping node’s block with its private key and send it back to that candidate bookkeeping node, this process is called a vote.

Each candidate bookkeeping node will count the total number of votes it receives. The total number of candidate bookkeeping nodes (that is, voting committee nodes) in the current batch is certain, so each candidate bookkeeping node can calculate its own vote rate. The node that obtains a majority of votes within the specified time becomes the final bookkeeping node. The result of voting is transparent and public in the entire network. Each vote uses the private key signature of each voting node, and its public key information is available in the entire network, so the voting result can be easily verified by all the nodes. The candidate bookkeeping node with the biggest number of leading zero becomes the bookkeeping node, and its generated block to be shortlisted will be packaged with the received voting result data and broadcast to all nodes.

Probability Right of Block Generation

Use the private key of a single device to encrypt the current block to see if the result of the encryption meets the characteristic value. There can be countless devices competing for the generation of this block. It avoids the emergence of mining machines as speculators because a node gets only one chance to negotiate a probability, and it does not depend on its specific computing power. It depends on its identity information. There is no significant difference between being doing 100 operations in 1 second and doing one operation because of the priority of probability negotiation with low round value that probability negotiation with high round value.

On-demand Verification

Blockchain can, and should, be verified independently. That is what we are here to do.

Our nodes ensure everything is completed on the correct chain (which is legal and accepted by the blockchain community). A 'legal' blockchain is one that is verified by nodes and miners. Invalid chains are rejected, and the chains with the most support will be the strongest. Nodes will therefore be able to ascertain which chain is the strongest in a way that is fast, efficient, and streamlined.

Verifying the block is beneficial, and there is another deeper reason. Suppose a strong participant tries to tramper with the protocol, and it is supported by most nodes and miners. If no one else verifies the chain, then it is susceptible to attacks. Everyone's client end will accept the new chain by default, and when anyone sees what's happening, it will be coordinated by objectors to reject. However, if ordinary users are verifying, then the coordination problem falls on the other side: whoever tries to change the agreement has the responsibility to convince the user to download the software patch to accept the change of agreement.

A. Basic structure verification

This mainly pertains to whether the data structure or information on the blockchain has been altered, including KYC information, voting information and so on. This prevents falsification by calculating the hash of the structure and comparing it with the hash stored in the structure.

B. Transaction information verification

Transactions are the most basic form of blockchain usage, and trust is the basis of everything. An asymmetric encryption algorithm is built into our system. Unlocking this requires both a public and a private key, acting as a pair. If the public key is used to encrypt data, only the corresponding private key can be used to decrypt; if the private key is used to encrypt the data, only the public key can be used for verification.

The transaction verification here mainly refers to the verification of the correctness of the transaction information. It prevents overdraft and double spending by double-checking the account information and increasing the transaction count in the account.

C. Block generation logic verification

This verifies block's bookkeeper credentials and voting information incorporating the block's generation logic to ensure that the XXX node of the block is consistent with the expected result, while preventing and shielding bad nodes from randomly generating blocks or deliberately generating bad blocks.

This part is the core function of Tera Hypernet. It will follow the entire block generation process and verify the correctness of each step of the logic in real time This ensures the platform runs and operates smoothly.

Transaction

The transaction refers to the transfer of certain assets from one account to another. The main part of the blockchain is the transaction. The main information stored in the block is the transaction list.

A. Transaction process

It takes a process from issuing to confirming for a transaction to happen. This process requires the participation of all nodes. It comes to fruition by writing the transaction to the block and uploading it to the chain.

There are several steps involved in the transaction flow:

- Send transaction request;
- Verify the legality of the transaction;
- Add the transaction to the local transaction pool;
- Broadcast the transaction;
- Other nodes receive the transaction and join the transaction pool;
- Select transactions and add them to the transaction list of the block as well as package the block according to certain rules during the packaging stage;
- Broadcast and confirm the block;
- Confirm the packaged transaction according to certain rules;

B. Dealing with nodes

After receiving the transaction, the node will add the transaction to the transaction poo. It then divides it into local transactions and general transactions. The transactions that meet the criteria are added to the transaction pool. The ones that do not are discarded.

Local transactions refer to transactions issued by this node. Local transactions have a higher priority. They will be given priority when packaging blocks and will be stored on disk to prevent transaction loss.

The node transaction pool has a size limit. If the transaction pool is full, a number of transactions will be deleted. When the block is packaged, the preferentially packaged transaction will be packaged into the block.

C. Storage

Transactions are not stored separately in LeveDB, but are stored in the body of the block. When storing different types of key-value pairs in LeveDB, different prefixes are added to the keywords to distinguish them. In order to be able to query the data, the index information of each transaction is also stored in the database. The index information contains the block index and block hash for locating the body of the block, and also includes the index position in the body of the block.

The storage of transaction receipts is similar to that of transacting. The difference is that the transaction receipt is stored separately in LevelDB, prefixed with r. In addition, considering that the transaction receipt and the transaction have a one-to-one correspondence, the index information can also be used to quickly locate the location of the transaction receipt to speed up the search of the transaction receipt.

K-bucket

The Tera Hypernet is implemented based on the Kademlia protocol. Kademlia is a protocol algorithm implemented through a distributed hash table. It was designed by Petar and David for a decentralized P2P computer network. Kademlia stipulates the structure of the network and the way of information exchange through node query. KAD chooses to use a recursive algorithm in node search. All nodes participating in the communication form a virtual network, and these nodes are identified by a set of numbers (or called node ID). The node ID is not only used for identification, but also for value positioning.

Each node maintains a routing table that consists of multiple lists. The number of lists is composed of Node ID bits like Node ID 160bits. When comparing with other node IDs, the number of bits with different numbers is 0~ 159 Kind, which means that there are 160 lists and it is stipulated that a maximum of k information is stored in each list. Hence it is also called K bucket.

Since the range of the coverage distance of each K bucket increases exponentially, this results in more information for nodes close to itself, and less information for nodes far away from itself, thus ensuring that the routing query process is convergent. Because the interval is divided exponentially, it has been proved that for a Kad network with N nodes, logN steps at most are required to query, and the target node can be accurately located.

A. Building K-bucket

Once the Tera node goes online, the initial work of K-bucket establishment work begins. Nodes access the network through their own stored seed nodes or active node information obtained from the network. It uses the Kademlia protocol to obtain more node information through the access node. It then adds this node information to the K bucket. Then use this node information as seeds and repeat this work until enough node information is obtained to complete the K-bucket node data.

B. Updates and maintenance

When node 'X' receives a PRC message, the IP address of sender 'Y' is used to update the corresponding K bucket. The specific steps are as follows:

1. Calculate the distance between yourself and the sender: d(x,y), note: x and y are ID values, not IP addresses.

2. Select the corresponding K bucket to perform the update operation through the distance d.

3. If the IP address of y already exists in the K bucket. Move the corresponding item to the end of the K bucket.

4. If y's IP address is not recorded in the K bucket:

    4.1. If the record items of the K bucket are less than k, the information of y is directly inserted into the end of the queue.

    4.2. If the record items of the K bucket are greater than k, then select the record item in the header for RPC_PING operation.

        4.2.1. If z does not respond, remove the information of z from the K bucket, and insert the information of y into the end of the queue.

        4.2.2. If z has a response, move the information of z to the end of the queue and ignore the information of y.

The K-bucket update mechanism effectively implements a strategy of updating the recently seen nodes, unless the online nodes have not been removed from the K-bucket. In other words, nodes that have been online for a long time are more likely to remain in the K-bucket list. This is based on: a node that has been online for a longer time is more worthy of our trust, because it is likely to remain online within the next hour. The performance will be greater than that of the latest node we visited, which corresponds to the stability of the Kad network and reduces network maintenance costs.

By keeping the nodes that have been online for a long time in the K bucket, Kad significantly increases the probability that the nodes in the K bucket are still online in the next time period, which corresponds to the stability of the Kad network and reduces network maintenance costs (no need to build nodes frequently Routing table) brings great benefits. Another advantage of this mechanism is that it can prevent DOS attacks to a certain extent, because Kad will update the K bucket information only when the old node fails, which avoids flooding routing information through the addition of new nodes.
In order to prevent K buckets from aging, all K buckets that have no update operations within a certain period of time will randomly select some nodes from their K buckets to perform the RPC_PING operation. This enables Kad to alleviate the traffic bottlenecks and allows it to respond to node failures quickly.

LevelDB

All data of Tera HyperNet is stored in LevelDB. LevelDB is Google's open source persistent KV stand-alone database. In fact, LevelDB is not a full-featured database. It is an interface to operate the database. LevelDB applies the LSM (Log Structured Merge) strategy, lsm_tree delays and batch index changes. It efficiently transfers the updates to disk, reducing overheads.

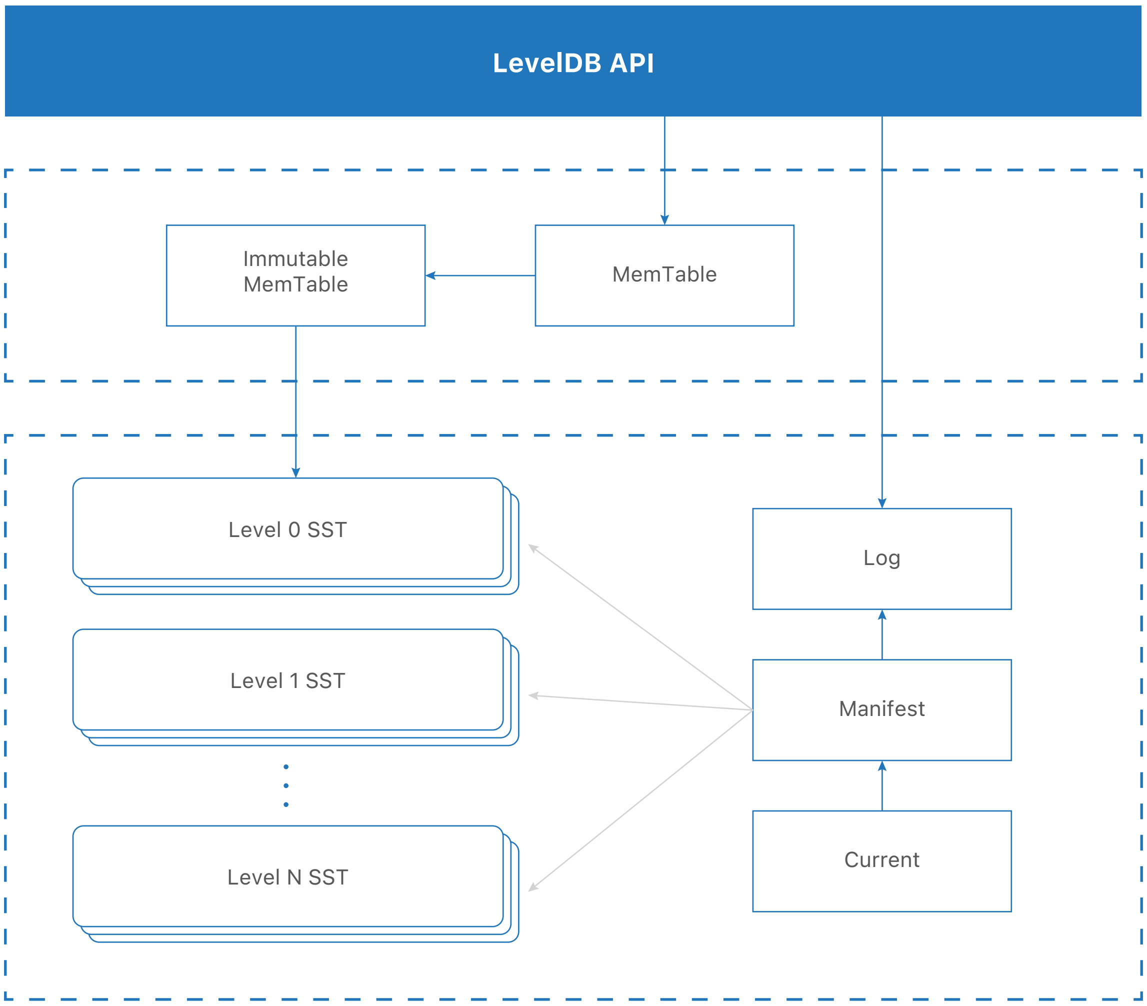

Fig.3 Whole Structure of LevelDB

LevelDB has the following features:

- The keys and values are arbitrary byte arrays;
- Data is stored in the order of keys;
- A customized comparator could be used to define the sorting methods;
- The basic operations are Put(key, value), Get(key), Delete(key);
- Multiple changes can be made in an atomic batch;
- Support Snapshot. Users can establish a consistent view;
- It can be iterated forward and backward;
- Support snappy compressed data;
- Operating system related operations are abstracted, and these interfaces can be customized across platforms;
- Only one process can access the database at the same time, but it supports multi-threaded access.

HyperNet

Finance is based on one core principle: trust. How that trust is denominated is based upon different systems. During the time of the gold standard, gold was the store of value that trust is based around. Within a large and growing community, Bitcoin is that store of value, but it is not without its faults. Bitcoin is energy-intensive and much of it is held in the hands of the few through years of monopolization and market manipulation, breaking away from its original intention of Satoshi Nakamoto, which was to provide a new way of doing finance, without governments, without central banks.

In the new era, the old way is being uprooted and new operational modes are being established. With the Internet of Things in full swing, a new way needs to be established. That is what we are here to do.

Our system is based on the bottom consensus algorithm. At its core, it is a negotiation solution based around probability. This uses a decentralized approach to decompress the entire network system to prevent the entire network from competing within each other. In this way, the excess computing power, bandwidth and storage can be fully utilized. All these forces can therefore come together to form a new type of HyperNet and become a new network in a new “generating relationship”.

The trust of the entire network is the foundation of everything. To make everything seamless, however, not everything needs to be verified by all nodes. We use on-demand verification, done by stakeholders. This optimizes the verification efficiency of the network and provides a new solution to the industry.

Tera HyperNet has the following core advantages:

1. The new proof-of-probability consensus mechanism solves the problem of network efficiency.
2. There are unlimited nodes.
3. Each device is a node and they have equal rights.

With the expansion of scale and the formation of a hyper-computing network, it will become an intermediary between the Internet of Everything and the Internet of Trust, a shared distributed computing infrastructure.

As mentioned above, the Tera Hypernet system is a revolution in the basic technology of blockchain. This change inherits the advantages of existing blockchain technology and better meets the demand for various applications. This change can allow more nodes to participate in the Tera Hypernet system, and provide the distributed computing and storage tasks required by the blockchain. It leaves room for storage and computing and brings it all together into a HyperNet.

Each member of the Tera HyperNet has strong storage, computing and network capabilities aligned to our working mechanism. Tera HyperNet also provides an incentive mechanism. This will provide the scale and computing power beyond the traditional cloud services operated by Google and Amazon, which act as one entity. In effect, we create an open computing and trading world.

References

[1] L. A. ZADEH. “Probability Measures of Fuzzy Events”. https://core.ac.uk/download/pdf/82591812.pdf

[2] Jim Pitman “Exchangeable and partially exchangeable random partitionss,” .https://www.stat.berkeley.edu/~aldous/206-Exch/Papers/pitman95a.pdf ,Received: 5 May 1992

[3] A. Back, “Hashcash - a denial of service counter-measure,” http://www.hashcash.org/papers/hashcash.pdf, 2002.

[4] C. Schnorr, “Effificient signature generation by smart cards,” J. Cryptology, vol. 4, no. 3, pp. 161–174, 1991.

[5] “Bitcoin“ http://bitcoin.org/bitcoin.pdf,” 2008.

[6] “Ethereum” https://github.com/ethereum/wiki/wiki/White-Paper, 2014.

[7] S. Gilbert and N. Lynch, “Brewer’s Conjecture and the Feasibility of Consistent Available Partition-Tolerant Web Services,” in In ACMSIGACT News, 2002, p. 2002.

[8] H. Krawczyk. Simple forward-secure signatures from any signature scheme. In ACM CCS 00, pages 108– 115. ACM Press, Nov. 2000.

[9] G. Wood, “Ethereum: A secure decentralised generalised transaction ledger,” http://gavwood.com/paper.pdf, 2014.

[10] Jing Chen Sergey Gorbunov Silvio Micali Georgios Vlachos “ALGORAND AGREEMENT Super Fast and Partition Resilient Byzantine Agreement” https://algorandcom.cdn.prismic.io/algorandcom%2F218ddd09-8d6f-42f7-9db9-5cfbc0aedbe5_algorand_agreement.pdf