IOT DATA BLOCK CHAIN

Construct the Internet of Things Data Ecosystem

Table of Contents

Chapter I IOT Database Blockchain·Internet of Everything

IOT Database Blockchain fuses the Internet of Everything theory of Internet of Things and blockchain technology, dedicated to creating an interconnected big data platform. Traditional operating platform of Internet of Things (IoT) adopts a centralized technology. Either data collection or operation is based on the prerequisite that any party of IoT works for due diligence. Trust mechanism develops slowly, which seriously hinders the development of IoT. Focusing on the problems in the process of IoT, IOT Database Blockchain puts emphasis on problems that blockchain technology is used in IoT industry, such as data acquisition, data concurrency and data coexistence, combining blockchain-IoT-big data to build a new generation of Internet of Things architecture.

IOT Database Blockchain stores the data of IoT participants in each link and realizes the transformation of assets. At the same time, IOT Database Blockchain builds a technology-based trust system based on the characteristics of the blockchain. In IOT Database Blockchain, due to its non-defective modification and traceability, participants have absolute trust in the fair mechanism of the transaction, which ensures the normal operation of the transaction.

LDBC (Logistics data block chain) is the basic token of IOT Database Blockchain. LDBC facilitates the capitalization of commodity circulation data in IoT through IOT Database Blockchain and subverts the acquisition of traditional data. Based on basic circulation data, IOT Database Blockchain derives other smart contracts and ultimately contributes to high-availability big data of IoT.

Chapter II IOT Database Blockchain Technology

1.1 IOT Database Blockchain

The purpose of IOT Database Blockchain is to integrate and enhance the technology based on concepts of on-chain meta-protocol, scripts, smart contracts, cross-chain collaboration and distributed storage, enabling developers to create a data presentation and data storage platform, which is arbitrary consensus-based, scalable, standardized, fully-featured and easy to develop and collaborate. By creating ultimate abstract base layer - blockchain with built-in Turing complete programming language - anyone can create contracts and decentralize applications, and set up freely defined ownership rules, trading methods and state transitions function.

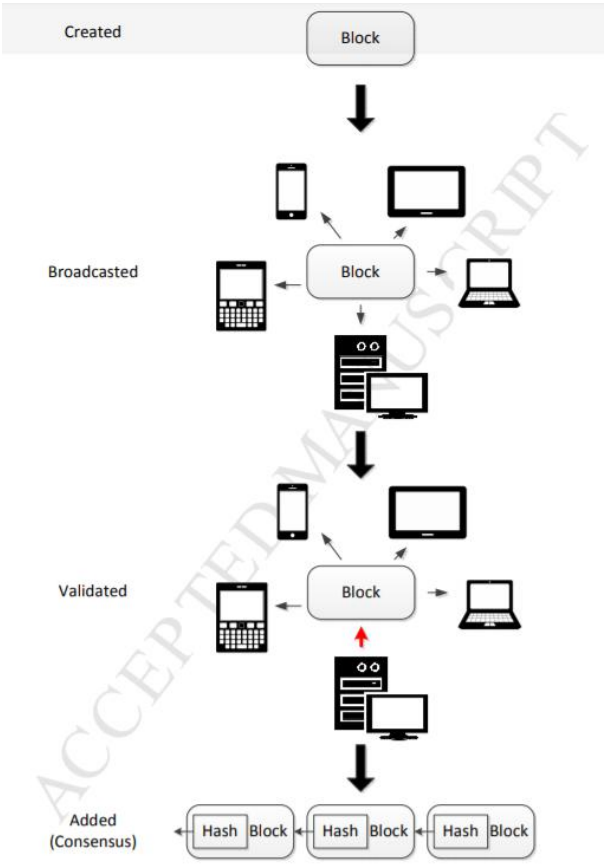

Fig.1 A typical blockchain work flow

1.2 IOT Database Blockchain Account

In the account system, the state is composed of objects called "accounts" (each account with a 20-byte address) and a state of transferring value and information between two accounts. LDBC account contains four parts:

• Random number, counter used to determine that per transaction can only be processed once

• The current balance of accounts

• The contract code of accounts

• Storage of accounts (default is empty)

LDBC is the main cryptographic fuel within IOT Database Blockchain, which is used to pay transaction costs. In general, there are two types of accounts: all external accounts (controlled by private keys) and contract accounts (controlled by contract codes). All external accounts have no code, and people can send messages from an external account by creating and signing a transaction. Each time the contract account receives a message, the code inside the contract will be activated, allowing it to read and write internal stores, and send other messages or create contracts.

1.3 News and Transactions

First, messages can be created by external entities or contracts, whereas bitcoin transactions can only be created externally. Second, messages can optionally contain data. Third, if the recipient of messages is a contract account, you can choose to respond, which means that IOT Database Blockchain message also contains the concept of functions.

The “transaction” in IOT Database Blockchain refers to signature packet that stores messages sent from an external account. The transaction contains the recipient of messages, the signature used to confirm the sender, the balance of currency accounts, the data to be sent and two values called STARTGAS and GASPRICE. In order to prevent the exponential explosion and infinite loop of the code, each transaction needs to limit the calculation steps caused by the executing code, including the initial message and all messages that are caused by the execution. STARTGAS is a limitation, and GASPRICE is costs required to pay the miners for each calculation step. If during the executing transaction, "the fuel is used up," all status changes will be restored to their original status, but the transaction fees already paid are not recoverable. If fuel is still left when the executing transaction is suspended, the fuel will be returned to the sender. The creation contract has a separate transaction type and corresponding message type; contract's address is calculated based on the account's random number and the hash of transaction data.

An important consequence of the message mechanism is the "primary citizen" property of IOT Database Blockchain - contracts have the same rights as external accounts, including the right to send messages and create other contracts. This allows the contract to

serve multiple different roles at the same time. For example, users can make a member of a decentralized organization (a contract) become a mediation account (another contract), providing intermediation services for an eccentric individual who uses customized quantum-based Lambert signature (the third contract) and an entity who uses an account that is secured by five private keys (the fourth contract). IOT Database Blockchain does not need to care what type of account each party of the contract is.

1.4 IOT Database Blockchain State Transition Function

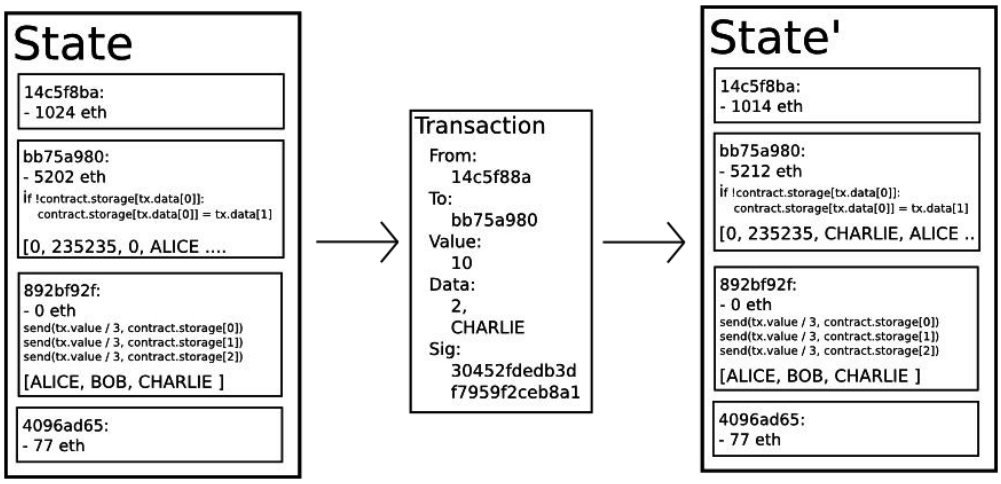

Fig. 2 State Transition Function

IOT Database Blockchain state transition function: APPLY(S,TX) -> S', can be defined as follows:

1 Check if the format of the transaction is correct (that is, with correct value), if the signature is valid and if the random number matches the random number of the sender's account. If not, the error is returned.

2 Calculation transaction fee: fee = STARTGAS * GASPRICE, determine the sender's address from the signature. Subtract the transaction fee from the sender's account and increase the sender's random number. If the account balance is insufficient, the error is returned.

3 Set the initial value GAS = STARTGAS, subtract a certain amount of fuel value according to bytes in the transaction.

4 Transfer the value from the sender's account to the recipient's account. If the receiving account does not exist yet, create the account. If the receiving account is a

contract, run contract code until the code or the fuel runs out.

5 If the sender's account does not have enough money or code execution runs out of fuel, the value transfer fails and the original status is restored, transaction fee still needs to be paid and added to the miner's account.

6 Otherwise, all remaining fuel is returned to the sender, and the consumed fuel is sent to the miners as transaction costs.

For example, suppose the contract's code is as follows:

if !contract.storage[msg.data[0]]:

contract.storage[msg.data[0]] = msg.data[1]

It should be noted that in reality contract code is written with underlying virtual machine code. Assume that contract memory is initially empty, with a value of 10 coins and 2000 fuels, whose price is 0.001 coins. And after the transaction whose two data field values are [2, 'CHARLIE'] [3] is sent, the processing of state transition function is as follows:

1 Check if the transaction is valid and the format is correct.

2 Check the transaction sender has at least 2000*0.001=2 coins. If so, subtract 2 coins from the sender's account.

3 Initially set gas=2000. We assumes that the transaction length is 170 bytes and the cost per byte is 5, minus 850, so there is 1150 left.

4 Subtract 10 coins from the sender account and add 10 coins to the contract account.

5 Run the code. In this contract, it’s simple to run the code: it checks if the index 2 in contract memory has been used. If it is not used, set its value as CHARLIE. Assuming that this consumes 187 units of fuels, the remaining fuel is 1150 – 187 = 963.

6. Add 963*0.001=0.963 coins to the sender's account to return to the final status.

If there is no contract to receive the transaction, then all transaction costs are equal to GASPRICE multiplied by the length of transaction bytes, and transaction data is independent of transaction costs. In addition, it should be noted that the message initiated by the contract can allocate fuel limits to calculations they generate. If the sub-calculated fuel runs out, it will only return to the state at the time the message was sent. Therefore, like a transaction, a contract can also protect its computing resources by setting strict limits to the subcalculations it generates.

1.5 Code Execution

The code of IOT Database Blockchain contract is written in a low-level stack-based

bytecode language, which is referred to as "IOT Database Blockchain virtual machine code". The code consists of a series of bytes, each byte representing an operation. In general, code execution is an infinite loop. Once program counter is incremented by one (the initial value is zero), the operation is executed once until code execution is completed or an error, STOP or RETURN instruction is encountered. The operation can access three kinds of space for storing data:

●Stack, a last-in first-out data store, 32-byte values can be pushed into or out of the stack.

●Memory, infinitely extendable byte queue.

●The long-term storage of the contract, the storage of a secret key/value, where the secret key and the value are all 32 bytes in size. Unlike the stack and memory that are reset at the end of the calculation, the stored content will be maintained for a long period.

The code can access the value, the sender and the data in the received message just like accessing block chain data. The code can also return the data's byte queue as output.

The formal execution model of virtual machine code is surprisingly simple. When IOT Database Block chain virtual machine is running, its complete calculation state can be defined by the tuple (block_state, transaction, message, code, memory, stack, pc, gas), where block_state is the global state that contains all account balance and storage. During each round of execution, the current instruction is found by calling the pc (program counter) bytes of the code, and each instruction defines how to affect the tuple itself. For example, ADD pops two elements and inserts their sum to the stack, subtracting gas (fuel) by one and adding pc by one. SSTORE pops the top two elements and inserts the second element to the first contract storage location defined by each element, subtracting the gas value by up to 200 and adding pc by one. Although there are many ways to optimize by just-in-time compilation, the basic implementation of IOT Database Block chain can be implemented in hundreds of lines of code.

1.6 Blockchain and Mining

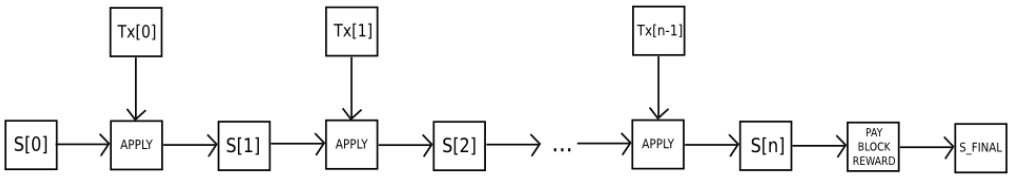

Fig.3 Map of Blockchain and Mining

Although there are some differences, the blockchain of IOT Database Block chain is similar to the block chain of Bitcoin in many aspects. The difference in their block chain architectures is that the block of IOT Database Block chain contains not only transaction records and recent status, but also block number and difficulty values. The block verification algorithm in IOT Database Block chain is as follows:

1 Check if the previous block referenced by the block exists and is valid.

2 Check if the timestamp of the block is larger than the previous referenced block and is less than 2 minutes.

3 Check if block number, difficulty values, transaction roots, ter-roots and fuel limits (many of the underlying concepts specific to Ethereum) are valid.

4 Check whether the workload certification of the block is valid.

5 Assign S[0] to the STATE_ROOT of the previous block.

6 Assign TX to the transaction list of the block. There are n transactions in total. For i belonging to 0...n-1, a state transition S[i+1] = APPLY(S[i], TX[i]) is performed. If any of conversion errors occur, or the gas used to execute the program here exceeds GASLIMIT, an error is returned.

7 Use S[n] to assign S_FINAL and pay a block bonus to the miner.

8 Check if S-FINAL is the same as STATE_ROOT. If the same, the block is valid. Otherwise, the block is invalid.

The confirmation efficiency of IOT Database Block chain is far beyond Bitcoin. The reason is that the state is stored in the tree structure, and only a small part of the tree structure needs to be changed for each additional block. Therefore, in general, most of the tree structures of two adjacent blocks should be the same, so storing data once can be referenced twice by using a pointer (ie, a subtree hash). A tree structure called "Patricia Tree" can achieve this, including the modification of Merkel tree, which not only allows to change nodes, but also insert and delete nodes.

In general, a perfect example of IOT Database Block chain is a self-enforced reward for solving the problem of sharing logistics information. Through original annular calculate, data sharing is made into a standardized consensus, improving the recognition and consensus rewards of block chains.

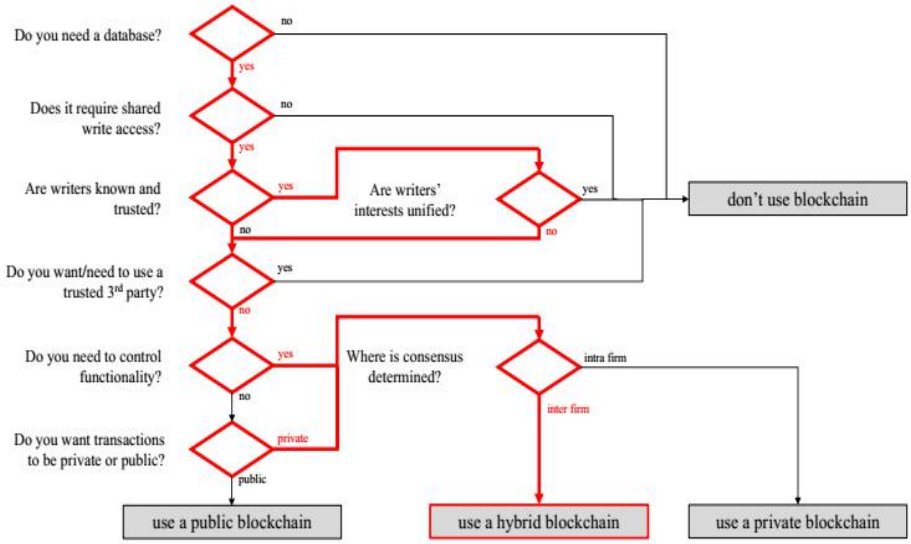

Fig. 4 The decision taken on what kind of blockchain to use

1.7 Improved Ghost Protocol

The motivation behind the Ghost Protocol is that the current fast-acknowledged block chain is plagued by low security because of the high block rate of the block; because the block needs to spend a certain amount of time (set to t) spreading to the entire network. If miner A has dug a block and then miner B happens to dig another block before A's block spreads to B, miner B's block will be obsolete and will not contribute to network security. In addition, there is also a centralization problem: if A is a mine that has 30% computor power of the total net while B has 10% power, A will face the risk that 70% of the time will produce invalid blocks while B will face the risk that 90% of the time will produce invalid blocks. Therefore, if invalidation rate is high, A will be more efficient simply because of the higher share of computing power. Combining these two factors, quickliy produced block chains are likely to result that a mine pool possesses the computing power share that can actually control mining. The power share of the process. By including waste blocks when calculating which chain is the "longest" one, Ghost Protocol solves the first problem of degrading network security; that is, in addition to the parent blocks and earlier ancestral blocks, descendant blocks after the invalidation of ancestral blocks are also added to calculate which block has the maximum workload confirmation to support it. IOT Database Block chain pays 87.5% of the rewards for the waste blocks that contribute to new block confirmation as "uncle block". The "nephew block" that is included in the calculation will receive 12.5% of the rewards. However, the transaction fee is not given to uncle blocks.

IOT Database Blockchain implements a ghost protocol of simplified version that only goes down to the fifth level. It is featured that waste blocks can only be incorporated into calculation by from the second generation to the fifth generation descendant blocks of the parent as uncle blocks, rather than more distant descendant blocks (such as the sixth generation of parent blocks, or the third generation descendant blocks of grandfather blocks). There are several reasons for this. First, unconditional ghost protocol will give excess complexity to the calculation of which uncle block of given blocks is legitimate. Second, unconditional ghost protocol with compensation used by IOT Database Block chain deprives miners of incentives to mine on the chain rather than on an open attacker's chain. Finally, the calculation shows that the five-level ghost protocol with incentives achieves a 95% or more efficiency even when the block-out time is 15 seconds, while the benefit of a 25%-powered miner from centralization is less than 3%.

Because each transaction posted to the block chain consumes the costs of download and verification, there is a need for a regulatory mechanism that includes transaction fees to guard against spamming transactions. The default method used by Bitcoin is purely voluntary transaction fees, which relys on miners as gatekeepers and sets a dynamic minimum fee. Because this method is "market-based", allowing miners and transaction senders to determine prices based on supply and demand, this method is successfully accepted in Bitcoin community. However, the problem with this logic is that transaction processing is not a market; although it is intriguing to intuitively interpret transaction processing as a service provided by the miner to the sender, the fact is that a miner's transaction requires each node processing of the network, so the largest part of transaction processing costs are borne by a third party rather than the miner who decides whether to include the transaction. As a result, a construction site tragedy is very likely to happen.

However, when given a special and less precise simplified assumption, loopholes in this market-based mechanism miraculously eliminate its influence. The argument is as follows. Assumptions:

1 A trade brings k steps and provides the reward kR to any miner who has included the trade, where R is set by the transaction publisher, and k and R are (approximately) visible to miners in advance.

2 The cost that per node handles each step is C (ie, the efficiency of all nodes is consistent).

3 There are N mining nodes, each of which has the same computing power (ie, 1/N of the total network power).

4 There is no full node without mining.

When the expected reward is greater than the cost, miners are willing to mine. In this way, because the miner has 1/N chance to process the next block, the expected benefit is kR/N, and the miner's processing cost is simply kC. In the case when kR/N > kC, ie R > NC, miners are willing to include transactions. It is noted that R is the cost per step provided by the sender of the transaction, which is the lower limit the miner will benefit from processing the transaction. NC is the cost of processing an operation across the entire network. Therefore, miners only have the motivation to include those transactions that benefit more than costs.

However, there are several important deviations from these assumptions and actual situation:

1. Because additional verification time delays the broadcast of the block, and thus it increases the chance that the block becomes a waste block, miners handling the transaction pay higher costs than other verification nodes.

2. There are full nodes without mining.

3. The distribution of computing power in practice may end up being extremely uneven.

4. There are real speculators, politicians and madmen who are committed to destroying the Internet. They can cleverly set up contracts so that their costs are much lower than other verification nodes.

The first point above drives miners to include fewer transactions and the second point increases NC; therefore, the impact of these two points at least partially offset each other. The third and fourth points are main problems; as a solution, we simply establish a floating upper limit: no block can contain more operands than BLK_LIMIT_FACTOR times long-term exponential moving average. Specifically:

blk.oplimit = floor((blk.parent.oplimit * (EMAFACTOR − 1) + floor(parent.opcount * BLK_LIMIT_FACTOR)) /EMA_FACTOR)

BLK_LIMIT_FACTOR and EMA_FACTOR are constants which are temporarily set as 65536 and 1.5, but they may be adjusted after further analysis.

1.8 Computation and Turing Complete

It needs to be emphasized that IOT Database Block chain virtual machine is Turing complete; it means that virtual machine code can implement any imaginable calculation,

including infinite loops. There are two ways to implement loops in IOT Database Block chain virtual machine code. First, JUMP instruction can cause the program to jump back somewhere, and JUMPI instruction that allow conditional statements like while x < 27: x = x * 2 to implement conditional jumps. Second, the contract can call other contracts, with the potential to achieve loops through recursion. It naturally leads to a question: Can malicious users be forced to shut down by forcing miners and all nodes into an infinite loop? This problem arises because of an outage problem in computer science: there is no way to know in a general sense whether a given program can finish running in a limited amount of time.

Our solution solves the problem by setting the maximum number of calculated steps to run for each transaction. If it is exceeded, the calculation is reinstated, but it still has to pay. The news works in the same way. To show the motivation behind this scenario, please consider the following example:

An attacker creates a contract that runs infinite loops and then sends a transaction that activates loops to the miner, who will process the transaction and run infinite loops until the fuel runs out. Even if the fuel is exhausted and the transaction stops halfway, the transaction is still correct (back to where it was) and the miner still earns the cost of each step from the attacker.

An attacker creates a very long infinite loop intention that forces the miner to calculate for a long time, so that several blocks have been generated before the end of the calculation and the miner could not collect the transaction to earn fees. However, the attacker needs to issue a STARTGAS value to limit the number of executable steps, so the miner will know in advance that the calculation will take too many steps.

An attacker sees a contract containing a contract such as send(A,contract.storage[A]); contract.storage[A] = 0, and then sends a transaction that is only enough to perform the first step but not the second step. The transaction (that is, withdrawal without reducing account balance). Contract author does not need to worry about defending similar attacks, because all changes will be reverted if execution stops halfway.

A financial contract works by extracting the median of nine private data publishers to minimize risks. An attacker takes over one of data providers and then designs the variable address calling mechanism as a changeable data provider to run an endless loop, so as to persuade any attempt to request funds from this contract to be suspended as the fuel runs out. However, the contract can set fuel limits in the message to prevent such problems.

The replacement for Turing complete is Turing incomplete, where JUMP and JUMPI instructions do not exist. And in each contract, only one copy is allowed to exist in the calling stack at a given time. In such a system, the above-mentioned fee system and the uncertainty of the efficiency of solutions surrounding us may not be necessary, because the cost of executing a contract will be determined by its size. In addition, Turing incomplete is even not a big restriction. In all the contract examples we have envisioned so far, only one needs to be cycled, and even this cycle can be replaced by the repetition of 26 single-line code segments. Taking into account the serious troubles and limited benefits brought by Turing complete, why not simply use a Turing incomplete language? The fact is that Turing incomplete is far from a concise solution. why? Please consider the following contract:

C0: call(C1); call(C1);

C1: call(C2); call(C2);

C2: call(C3); call(C3);

…

C49: call(C50); call(C50);

C50: (run one step of a program and record the change in storage)

Now, send a transaction like this to A. We have a contract that takes 250 steps in 51 transactions. The miner may try to maintain a maximum executable step for each contract and calculate possible execution steps for contracts that call others for recursion, so as to detect such logic bombs in advance, but this would make it impossible for miners to create contracts for other contracts (because the creation and execution of the above 26 contracts can be easily put into a single contract). Another problem is that the address field of a message is a variable, so in general it may not even be possible to know in advance which other contract the contract will call. Thus, we finally have a surprising conclusion: the management of Turing complete is amazingly easy, while in the absence of the same control, the management of Turing incomplete is surprisingly difficult - then why not make the agreement Turing complete?

1.1 1.9 Decentralization of Mining

The purpose of IOT Database Block chain is to use a mining algorithm based on a function that randomly generates a unique hash for every 1000 random numbers, with a sufficiently wide computational domain, to remove the advantages of dedicated hardware. Each individual user can use their private laptop or desktop machine to complete a certain

amount of mining activity almost for free, but more mining will require them to pay for electricity and hardware costs after 100% CPU usage. ASIC mining companies need to pay for electricity and hardware from the first hash. Therefore, if centralized revenue can be kept below (E + H) / E, ordinary miners still have room to survive even if ASICs are manufactured. In addition, we plan to design the mining algorithm to mine that requires access to the entire block chain, forcing the miners to store completed block chains or at least be able to verify each transaction. This removes the need for centralized mine pool; while mine pool can still play a random role in smoothing income distribution, this function can be done equally well for a P2P pool without centralized control. In this way, even if most of ordinary users still prefer to select light clients, it can also help to prevent centralization by increasing the number of full nodes in the network.

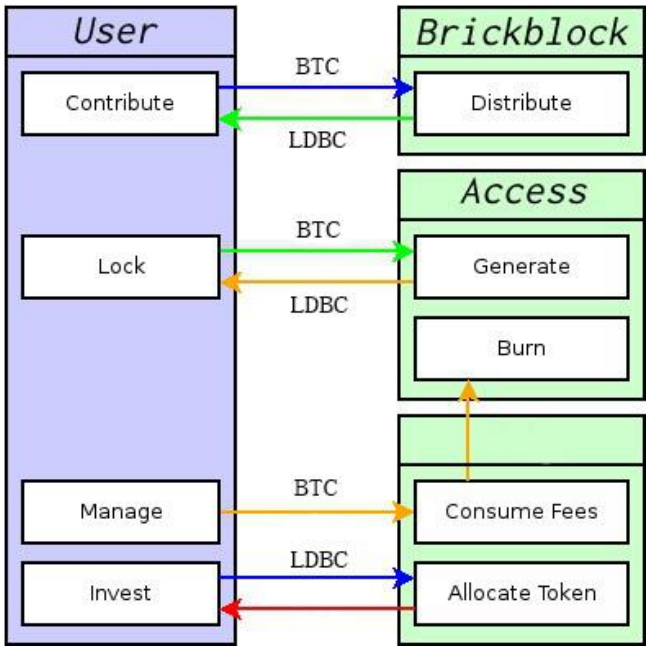

Fig. 5 Different types of tokens and their interactions

1.10 Extensibility

Extensibility is often the focus of attention. Like bitcoin, IOT DATABASE CLOCKCHAIN also suffers from the fact that each transaction requires every node in the network to deal with this dilemma. The current block chain size of Bitcoin is about 20GB, which grows at a rate of 1MB per hour. If Bitcoin network processes Visa-class 2000tps transactions, it will grow at 1MB every three seconds (1GB per hour, 8TB per year). IOT Database Block chain may experience similar or even worse growth patterns, because there will be many applications above the chain, rather than Bitcoin as a simple currency.

However, the fact that full nodes in the chain only need store state, rather than complete block chain history has improved the situation.

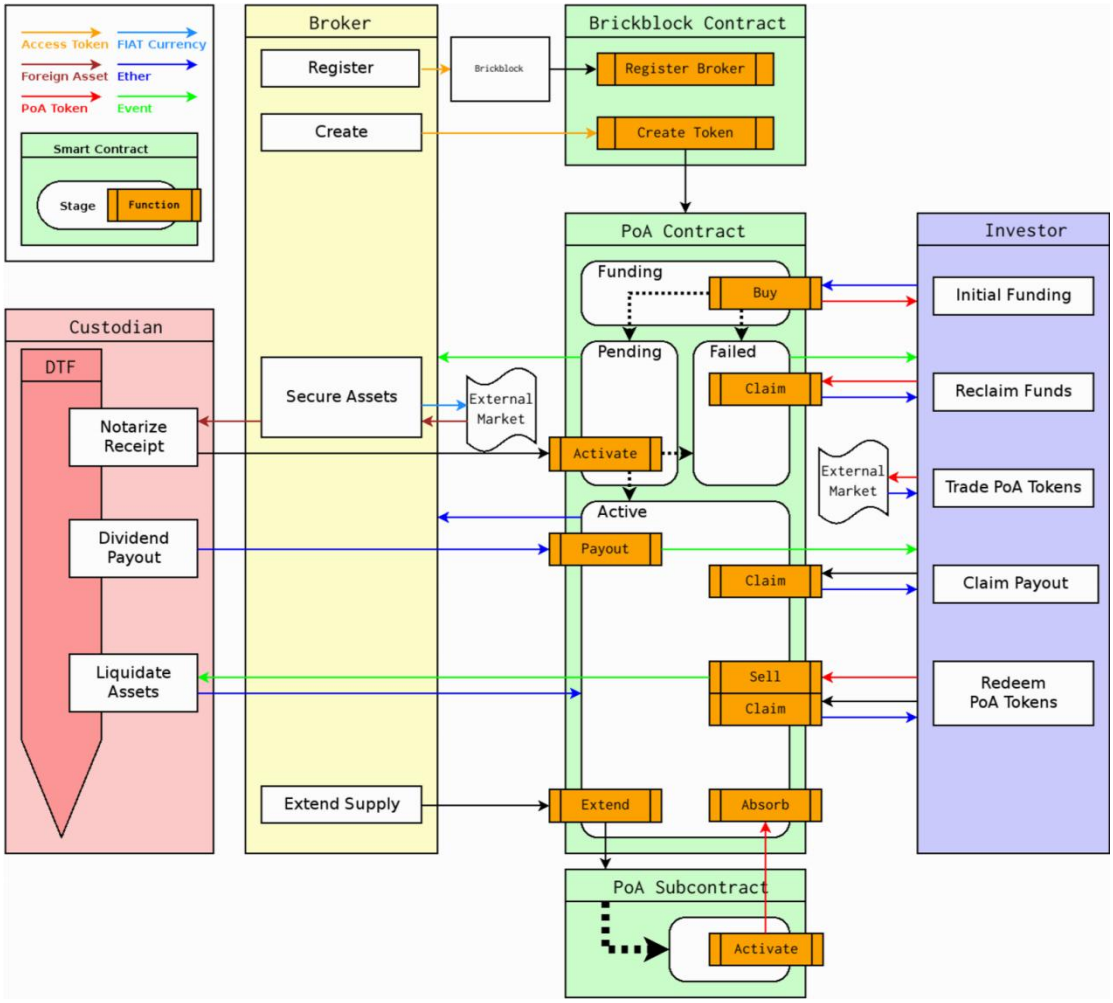

Fig.6 Cycle of Token

Large block chains face the risk of centralization. If the block chain size increases to 100TB, the likely scenario is that only a very small number of large merchants will run full nodes, while regular users use light SPV nodes. It raises concerns about the risk of fraud at full node partnership (for example, change block rewards and give themselves BTC). Light nodes will have no way to detect this fraud immediately. Of course, there may be at least one honest full node, and fraudulent information may leak through channels like Reddit a few hours later, but it's too late. Despite users make effort to repeal the blocks that have already been generated, they will all encounter huge infeasible coordination issues of the same scale as the launch of a successful 51% attack. IOT Database Blockchain will use two additional strategies to address this issue. First, because of the block chain-based mining algorithm, at least each miner is forced to become a full node, which guarantees a

certain number of full nodes. Second, more importantly, after dealing with each transaction, we will include the root of an intermediate state tree into the block chain. Even if the block verification is centralized, as long as there is an honest verification node, centralization can be avoided by a verification protocol. If a miner issues an incorrect block, the block is either in the wrong format or the state S[n] is wrong. Since S[0] is correct, there must be the first error state S[i], but S[i-1] is correct, verification node will provide the index i, together with a subset of the Patricia tree nodes required by the processing of APPLY (S[i-1], TX[i]) -> S[i]. These nodes will be mandated to perform the calculation to see if the resulting S[i] is consistent with previously provided values. In addition, it's more complicated that malicious miners issue incomplete blocks to attack, resulting in insufficient information to determine whether the block is correct. The solution is a question-response protocol: verification node questions target transaction index, and the light node receiving the information untrusts the corresponding block until another miner or verifier provides a subset of Patricia nodes as correct evidence.

1.11 Review

The above contract mechanism for decentralized application enables any one person to establish a command line application (basically speaking) through global network consensus on a virtual machine, which can change the accessible state of the entire network as its "hard disk". However, for most people, the lack of adequate user-friendliness of the command line interface used as a transaction delivery mechanism makes decentralization an attractive alternative. Finally, a complete "decentralized application" should include underlying business logic components and the upper graphical user interface components. IOT Database Block chain client is designed as a web browser, but it includes support for "LDBC" Javascript API. It can be used by specific web pages seen by the client to interact with the chain. From the perspective of "traditional" web pages, these web pages have completely static content, because block chain and other decentralized protocols will completely replace the server to handle user-initiated requests. Finally, decentralized protocols are hopeful to use LDBC to store web pages in some way.

1.12 Distribution Rules of IOT Database Block chain

IOT Database Block chain network contains its own IOT DATABASE currency. LDBC plays a dual role, providing major liquidity for various data transactions, and more importantly, it provides a mechanism for paying transaction costs.

The distribution model is as follows:

●Through sales activities, computing power allocation and data sharing, IOT Database Blockchain has generated approximately 9 billion LDBCs in 10 years and pre-drilled 5.5 billion LDBCs to be fully used to pay for the wages and rewards of developers and researchers, as well as projects investing in the ecosystem.

●2.7 billion LDBCs will be allocated to early contributors of project development and private placements.

●2.3 billion LDBCs will be included in the foundation account as a data contribution award, which will be released after the ring computing standard goes online in 2019.

●500 million LDBCs will be used to reward developers and community initial construction as a foundation.

●350 million LDBCs have been mined each year since the time they are on line, and a total of 3.5 billion LDBCs have been dug in 10 years.

Dismantling and decomposing linear linear growth model reduces the risk of wealth's excessive concentration in Bitcoin, and gives people living in the present and future a fair chance to acquire money while maintaining incentives to acquire and hold IOT DATABASE currency. Because in the long run, "money supply growth rate" tends to zero. As time goes by, the loss of currency due to carelessness and death will always happen. Assuming that the loss of the currency hss a fixed ratio of annual money supply, the final total circulation will stabilize at a value which equals to annual currency issue divided by annual money supply (for example, when the supply reaches 11x, 0.11x is excavated each year and 0.11x is lost, reaching an equilibrium). In addition to the linear issuance method, the growth rate of supply of LDBC which just likes Bitcoin tends to be zero in the long term.

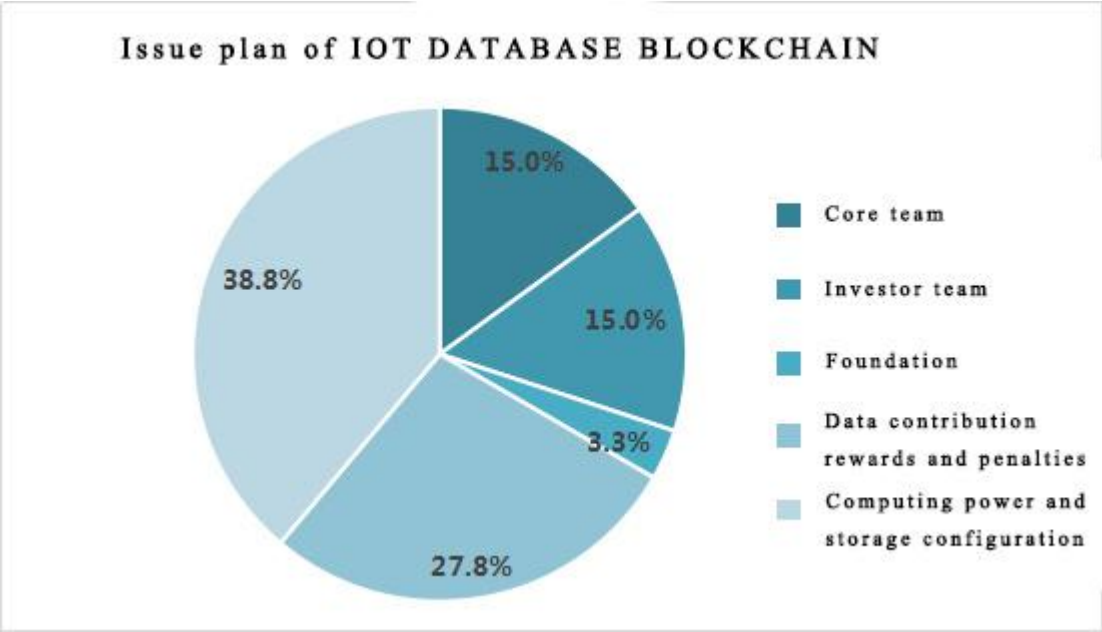

Fig. 7 Issue plan of IOT Database Block chain

The number of tokens: 9 billion

15% - U.S. Cloud Block chain Network Technology Co., Ltd., Asia Pacific Core Team of IOT Database Block chain

15% - Investor team

8.9% - Foundation

27.8% - Data contribution reward 20%

38.9% - for computing power and storage configuration

1.13 Conclusion

IOT Database Block chain protocol centers around decentralized data to share decentralized storage, decentralized computing and protocols and decentralized applications established by dozens of similar concepts. It has the potential to fundamentally improve the efficiency of the logistics industry and provide strong support for other P2P protocols by initially adding economic layers. In the end, there will be a large number of applications that have nothing to do with money.

The concept of arbitrary state transitions implemented by IOT Database Block chain protocol provides a platform with unique potential; unlike closed protocols designed for single purposes such as data storage or finance, it is open-ended in design. We believe that as a base layer, it is extremely suitable to serve an extremely large number of logistics industry and non-industry agreements that will emerge in the coming years.

Chapter III Team profile

Yanomiyabi	The inventor of circular computing, an expert on the application Blockchain, has been involved in the underlying development of several Japanese exchanges.
Stephen Temple	IT prodigy and holder of multiple blockchain patents.
Cheong Ben	Singapore expert on the application of Internet of Things, Japanese, the initiator of the theory of IOT Database Blockchain, Ph. D. in the Singapore University of Technology and Design.
Haitao Feng	The former Baidu blockchain underlying technology expert, with years of underlying development experience.
Xiaoming Peng	The former CMSWL CEO, the logistics data expert.

Chapter IV Consultants

Rachel Wilson	Professor, data security expert, early member of the Bitcoin community.
Michael Graetzel	Professor, leading figure in light nanotechnology industry.
Shinichi Mochizuki	Professor, Professor of Kyoto University, mathematician, leading figure in Anabelian geometry.

Chapter V Disclaimer

This document is only for the purpose of conveying information, which does not constitute relevant opinions on the sale and purchase of this item. The above information or analysis does not constitute an investment decision. This document does not constitute any investment advice, investment intentions or teaching investment.

This document does not constitute or understand to behaviors that provide any securities or any contract or commitment in form.

Relevant intended users clearly understand the risks of this project. Once investors participate in the investment, it indicates that they understand and accept the risk of the project and are willing to personally bear all the corresponding results.

The operating team does not bear any direct or indirect losses caused by participating in the project.