STRATOS

STRATOS WHITEPAPER

Published in 2021

Stratos is a large-scale distributed infrastructure network. Stratos provides three-in-one resources which include computation, storage and network traffic. Stratos uses Proof-of-Traffic algorithms to reward infrastructure participants and contributors in accordance with network traffic. At the same time, Stratos uses its own blockchain to measure the usage of computing resources (computation, storage and network traffic), and uses Practical Byzantine Fault-Tolerant consensus algorithm (PBFT) to integrate the network Proof-of-Traffic consensus algorithm deeply, providing settlement services and related financial payment services for network providers and network users in an efficient, fair and transparent manner.

Stratos is a decentralized data mesh for blockchain, it defines the next generation of decentralized computing services that can serve Dapp, allowing smart contract developers such as Uniswap, AAVE and NFT Dapps to use decentralized infrastructure to deploy, store and execute their code and data, so the community developers can publish the system without complicated network deployment. The developers can fully focus on the product business logic without worrying about the underlying infrastructure, making development and operations more effective and reducing development costs.

Stratos aims to build a distributed network ecosystem with infrastructure providers, middleware developers, and Dapp developers for common development, sharing value, and efficient commitment to provide credible, reliable, and low-cost applications for the prosperity of blockchain applications. Stratos will offer an indestructible environment for all the Dapps and no one will take your application down.

TABLE OF INDEX

CHAPTER 1

CURRENT SITUATION AND THOUGHT

1. Current situation and thought

1.1. Thoughts triggered by Bitcoin, Ethereum smart contract and blockchain technology

In 2008, Satoshi Nakamoto introduced a method which is implemented entirely through peer-to-peer technology Electronic cash system in his paper <Bitcoin: A Peer-to-Peer Electronic Cash System> [1]. It enables online payments to be directly initiated by one party and paid to the other party without any financial institution in the middle. Bitcoin created a precedent for a decentralized encrypted digital currency, becoming the first fully distributed digital encrypted currency that can complete transactions without any intermediary. To date, the total market value of Bitcoin has exceeded $290 billion. People usually call the digital cryptocurrency system represented by Bitcoin the blockchain technology version 1.0.

Between 2013 and 2014, Vitalik Buterin proposed the concept of Ethereum. Ethereum is an open source, public blockchain platform with a smart contract function. Ethereum provides an internal Turing complete scripting language, which allows users to build and process peer-to-peer smart contracts or transaction types by using its dedicated encrypted digital currency Ethereum (ETH). Up to now, the total market value of Ethereum has exceeded 48 billion U.S. dollars. People usually refer to the combination of digital encryption currency represented by Ethereum and smart contracts as version 2.0 of blockchain technology.

Blockchain technology is a self-referential data structure used to store a large amount of transaction information. Each record is linked in an orderly manner from back to front. It is open and transparent, cannot be tampered with, and is easy to trace. The blockchain itself uses computer network technology to compulsorily process many asynchronously executed nodes in different ways of synchronous information processing to achieve the consensus of all nodes in the system. This process does not require any centralized node to provide services or credit endorsement. In short, it greatly reduces the cost of establishing credit among all nodes in the system. Once this kind of point-to-point credit verification is applied in a real business environment, it will greatly eliminate the cost of trust between various nodes in the value chain, so the transaction information which is transferred between nodes can be valued differently without third-party verification or endorsement.

Since human society began to enter the stage of commodity economy, all commercial activities are inseparable from the effective flow of three types of values, namely information flow, logistics and value flow. Any behavior that effectively improves efficiency or reduces costs in the liquidity of any type of value can create a huge incremental trading market for the commodity economy. The birth of advanced transportation tools, such as ships, trains and airplanes, have greatly improved the efficiency of logistics, reduced logistics costs, and created the possibility of commodity circulation in the global market. The inventions of the telegraph, telephone, fax machine and the Internet have greatly improved the efficiency of information flow and reduced the cost of information flow, so that commodity information can reach a wider range of consumer groups and meet or even give birth to transaction needs. The emergence of banknotes, checks, credit cards, and electronic payment methods have greatly improved the efficiency of value transfer, greatly reduced costs, and greatly accelerated the capital turnover of all parties, so that less capital can be used to meet the needs of larger-scale transactions.

The essence of blockchain technology is to combine information flow and value flow into one, and at the same time, make it possible to complete point-to-point direct transaction confirmation without any third-party providing credit endorsement or intermediary services, which makes the value chain in any commercial field. The transactions between all participants (that is, participants in all transaction behaviors) can be automatically executed based on the smart contract developed between them, and the execution process is open and transparent, cannot be tampered with and can be traced. This greatly eliminates the cost of trust between transaction participants, and simplifies the multiple, multi-layer, asynchronous transaction process in the value chain into a one-time automatic transaction process in which all participants participate. Such a transaction method will counteract all participants in the value chain, and promote them to establish a consensus mechanism from the perspective of maximizing the benefits of the overall value chain, thereby greatly changing the production relationship between the participants, from the original distrust and even opposition to each other. The mode of infringing interests is transformed into a mode of establishing collective consensus, transferring and sharing interests, and further liberating the productivity of all parties in the overall value chain.

Although the various application ideas and scenario designs of blockchain technology are blooming, all attempts to implement these ideas and scenarios into applications face many common problems:

1. A completely decentralized design concept cannot coexist with security and high efficiency at the same time **(“the impossible triangle paradox”)**. Therefore, the consensus mechanism of STRATOS and the design of the network topology must achieve a compromise and balance between the three.

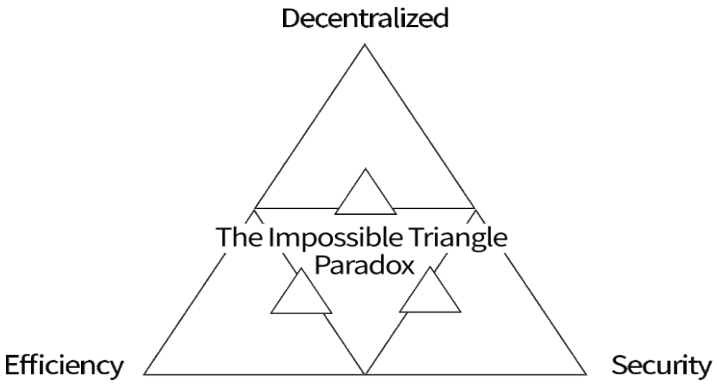

FIGURE 1. The impossible triangle paradox

2. At present, with the exception of a very small number of companies, the application and development of blockchain all encounters a lack of infrastructure and middleware, and the bottleneck of throughput, making the possibility of designing complex smart contracts almost non-existent. This prevents the possibility of using blockchain technology to solve various demand scenarios in the real world.

3. Blockchain, as an emerging technology, lacks regulatory laws and benign institutional design, combined with the financial attributes of blockchain technology itself, the cost of evildoing by blockchain practitioners has become very low, which has led to various financial scams and air currency projects. Deepening the public's negative impression of blockchain technology, making the popularization and implementation of blockchain technology encounter greater obstacles.

1.2. Decentralized Storage problems and challenges faced by public chain and DApp developers

In a decentralized computing network, how to solve the access reliability of network nodes and how to deal with the sudden increase of large-scale network traffic effectively while taking into account efficiency and cost has always been a difficult problem to solve. And these uncertain and unstable problems have caused headaches for many blockchain projects that require storage services, because no team will rely on services for which reliability and stability are uncertain. So why don't so many blockchain projects want to develop their own or even involve storage services?

Keeping a complete ledger at each node is an important feature of the current blockchain, but it is also a pain point of blockchain data storage. The size of the blockchain ledger can only be determined by the local storage of each node, and has nothing to do with the total storage size of the overall blockchain

network. Since each node is an independent server (whether it is a physical machine or a virtual machine), the local storage capacity is destined to have a physical upper limit. In this case, the entire blockchain can only store a few terabytes of data, which is why most blockchain projects don't want to touch storage, because the blockchain is not designed to support storage expansion.

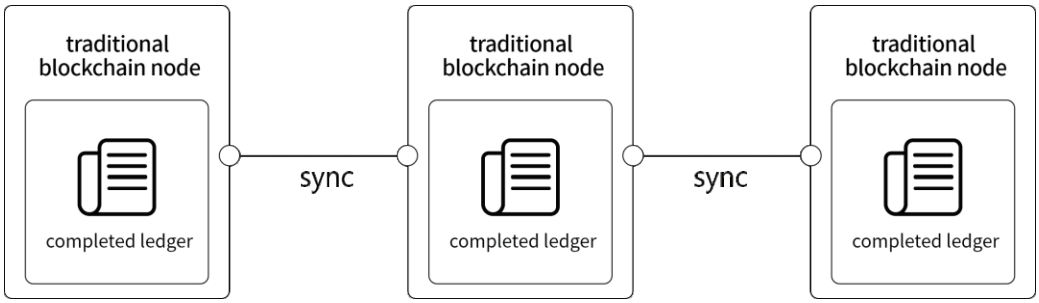

FIGURE 2. Storage of each node of Traditional blockchain

In order to solve the problem of reliability and efficiency of decentralized storage network, Stratos data mesh provide a meta service layer to execute the varies management services include indexing, routing, auditing tasks through Proof-of-Authority [3], and the meta services will ensure that the number of copies of all files in the P2P network is maintained basic quantity (default is 3) at any time. Multiple replications can greatly improve the reliability of file access, and different networks where different copies are located can provide sufficient redundant network resources, which can solve the overall reliability and efficiency problems effectively. At the same time, such a storage network is no longer an ordinary storage network, but a decentralized data mesh.

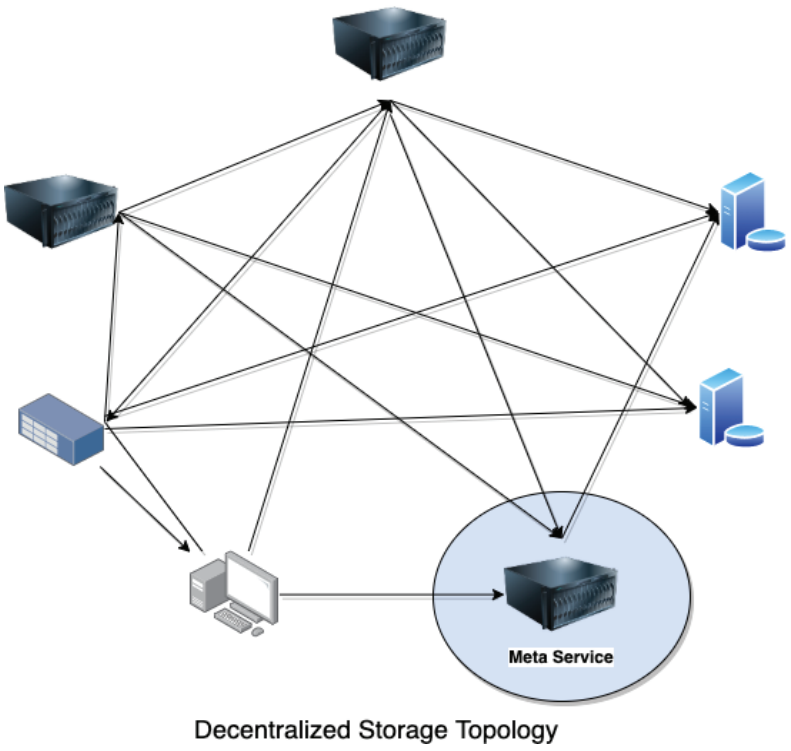

FIGURE 3. Decentralized Storage Topology diagram

CHAPTER 2

THE GOAL AND DESIGN PHILOSOPHY OF STRATOS

2

2. The goal and design philosophy of Stratos

2.1. Stratos data mesh

Stratos data mesh consist of three main components: decentralized storage, decentralized database and decentralized computation.

The decentralized storage and decentralized database provides fast storing/querying services for various data types. Whether the data is structured data or unstructured data, the Stratos data mesh will process them in different ways according to the different data type, such as speeding up data query and video playback.

The Stratos data mesh service is composed of the Stratos blockchain layer and the Stratos resource layer, which cooperate with each other to provide services for users.
The blockchain layer provides service fee settlement, miner incentive mechanism settlement, payment services, and verification services for the resource consumption in the data mesh. The resource layer provides computing, storage and network services. The storage layer of the resource node determines the storage method according to different data types, such as storing in a structured way or storing it in unstructured pictures, music, encrypted files or videos. The effcient query method will adopt different execution strategies according to different data types.

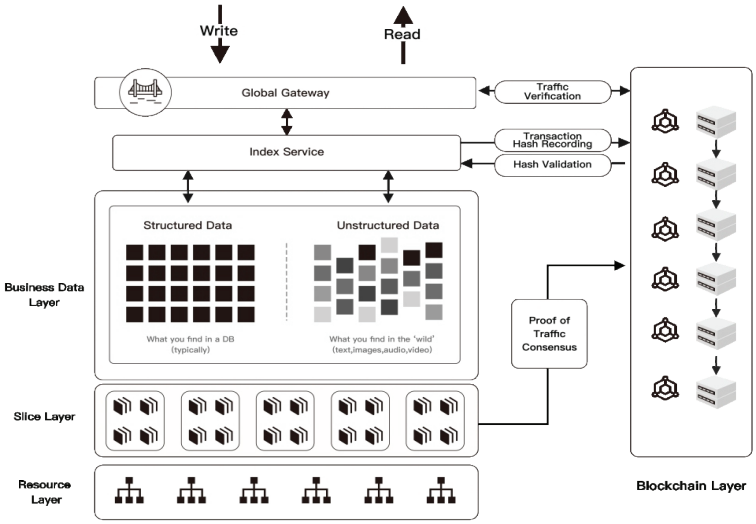

FIGURE 4. Data mesh structure

Combine with Stratos blockchain data mesh service provides the full scaled infrastructure for all kinds of enterprise level applications and customer-oriented applications.

Data mesh service is another new innovation in the information age following the Internet and computers. Stratos data mesh provide a decentralized resource management platform to help the customer to leverage the resources provided by the whole data mesh network. Decentralized computing is a great leap in the information age. The future era may be the era of decentralized computing. With the help of data mesh service, users can obtain unlimited resources through the network, and the obtained resources are not limited by time and space.

Stratos data mesh will provide the computing resources include storage, network traffic and computation capability which generally provided by centralized giant companies. The decentralized services provided by data mesh user no longer locked to any company and they can acquire the resources safe and secure. Small data centers have the advantages of low construction cost, low site requirements and low local energy requirements. Many companies own data centers. Many entities include school and institution have their own data center but never fully use their available resources. Since the deployment density of small data centers is much greater than that of large data centers, in terms of physical distance, small data centers will be closer to customers than large data centers, which can help customers reduce network delays and provide customers with richer services options based on geographic location. Stratos data mesh will be able to attract these small data centers to work together to provide decentralized computing services.

The Stratos blockchain collects and verifies each task performed in the resource node of each data center, records the usage of specific computing modules, memory modules and storage modules, and then writes to the chain after consensus process. The service provider receives the Stratos token directly which is calculated by the smart contracts. Users of data mesh will directly pay Stratos Token based on usage of resources.

2.2. Stratos design philosophy

STRATOS Data mesh service contains three layers:

- STRATOS blockchain layer
    - Provide workload calculation and settlement of the resources layer
    - Provide digital currency payment
    - Provide storage content verification

- STRATOS meta service layer

- Provide the indexing service of the computing resources on the resource nodes
- Route the resource access requests to proper resource nodes
- Audit the traffic (Include the data traffic and computation traffic) report
- Perform the network health checking

- STRATOS resources layer
  - Provide storage services
  - Provide computing services
  - Provide customized PaaS and SaaS services

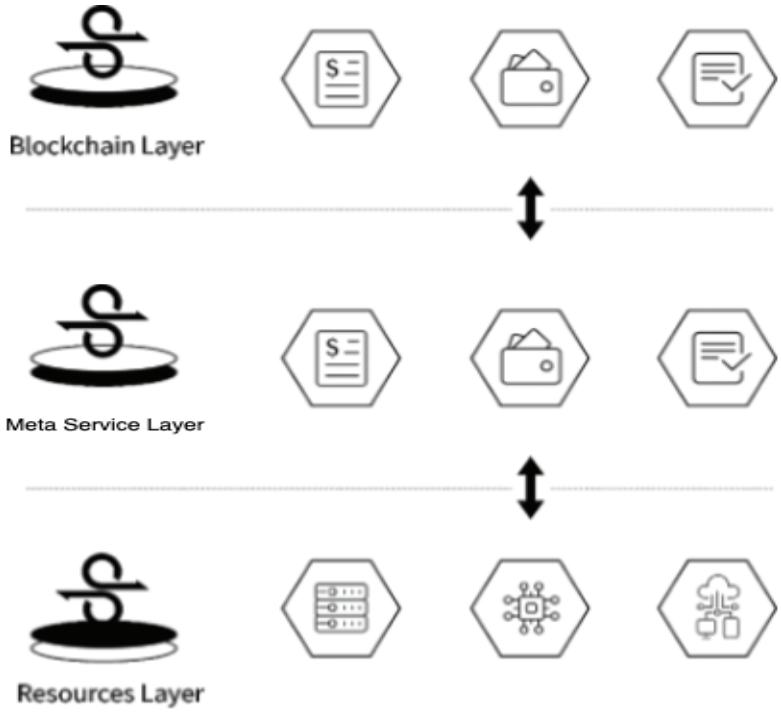

FIGURE 5. Three layers of Stratos

Stratos consensus algorithm iron triangle

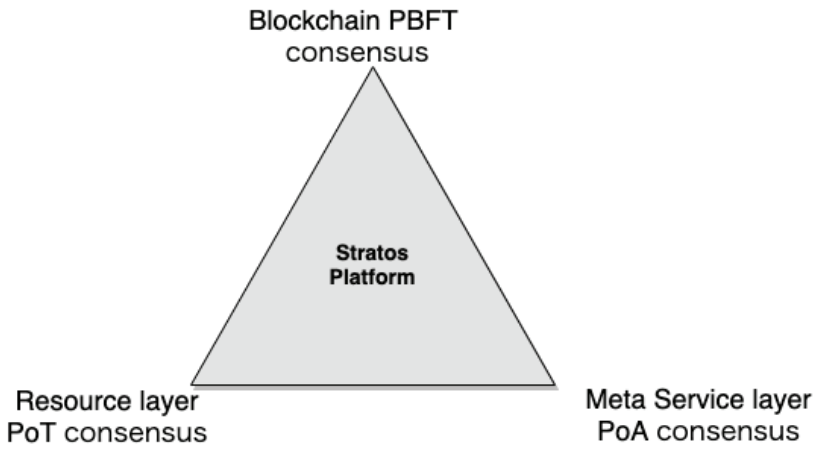

FIGURE 6. Stratos network consensus algorithm iron triangle

- Stratos blockchain uses the Practical Byzantine fault-tolerant consensus algorithm (PBFT) [2] to generate a block every 10 seconds
- The resource layer uses Proof-of-Traffic consensus algorithm to calculate the incentive return for each storage service provider
- The meta service layer uses Proof-of-Authority [3] consensus algorithm to elect resource nodes dynamically to perform indexing, auditing and routing tasks for the resource layer.

Three different consensus algorithms run on the three levels of Stratos Network, complement and depend on each other, and jointly ensure the stable and efficient operation of the entire Stratos ecosystem.

2.3. Stratos brief topology diagram

At the beginning of the white paper we already talked about “The completely decentralized design ideas cannot coexist with security and high efficiency at the same time” (“Impossible Triangle Paradox”). Stratos’s consensus mechanism and network topology design must be compromised and balanced between those three. Therefore, Stratos has fully considered how to achieve a safe and efficient balance as much as possible from the design point.

The following topology diagram is divided into three layers, each layer combines different types of consensus algorithms and fully considers the efficiency and scalability required by each layer under the premise of ensuring security. This is the “**three rules of three layers**” of Stratos.

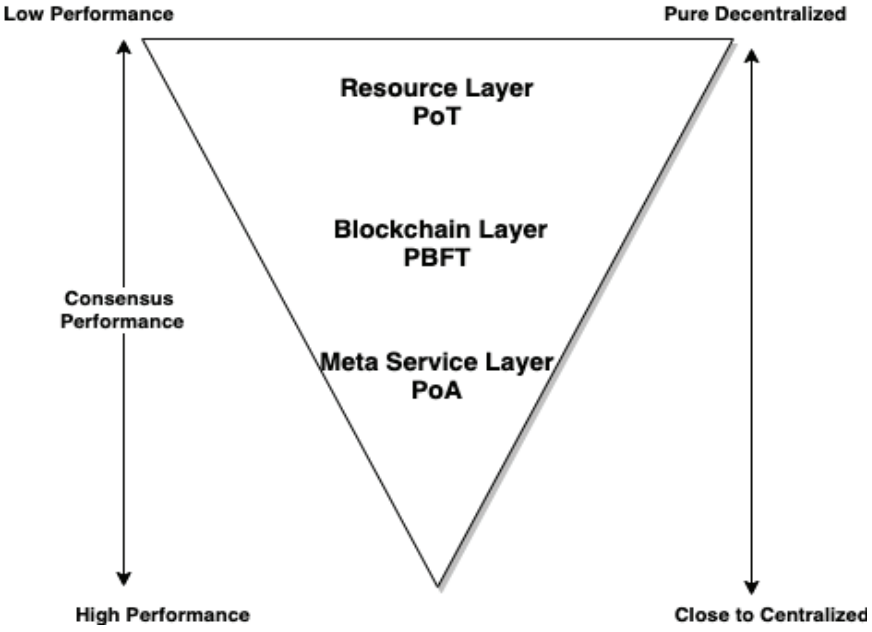

FIGURE 7. Three rules of three layers

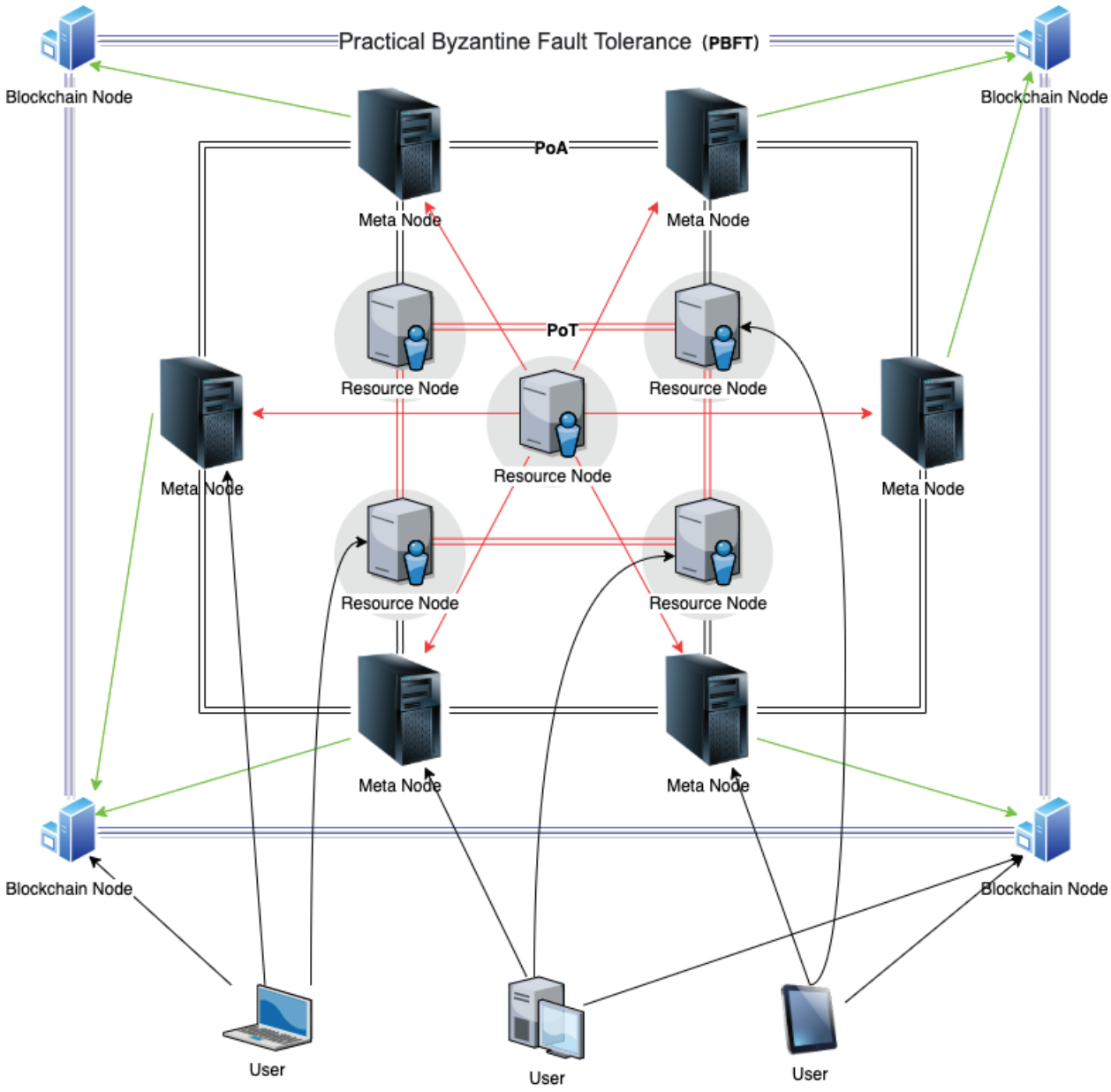

FIGURE 8. Stratos topology diagram

From the topology diagram, Stratos is a distributed computing storage network and a blockchain superimposed hybrid mode network. The meta service network applies a PoA consensus algorithm to elect meta nodes randomly execute indexing, auditing and routing services. The resource network which is composed of resource nodes sends traffic usages through a Proof-of-Traffic (PoT) consensus mechanism to the blockchain network which is composed of blockchain nodes. The blockchain nodes in the blockchain network use the practical Byzantine Fault Tolerance (PBFT) consensus mechanism to generate the data block that contains the traffic data, then send the incentives token through the smart contract and complete the transaction on the chain.

The goal of Stratos is to provide a QoS, efficient and secure services for the blockchain community. Both blockchain chain developers and DApp developers can choose Stratos as a low-cost, safe, reliable general computing infrastructure. High-performance storage networks distributed around the world and random coordination nodes can execute the deployment process automatically, transfer the huge amount data to anywhere around the world, response user requests efficiently.

2.4. Roles and functions of Stratos

2.4.1. Blockchain Node

Blockchain nodes are dedicated servers with sufficient computing power. The purpose of the blockchain node is to receive the traffic data reported by the meta node, then pack the traffic data into a blockchain according to the block generation logic and block generation cycle, and commit them to the consensus algorithm for verification. After the verification is completed, the block data is formed and stored in the node.

During the working period, the blockchain nodes will obtain Tokens for reward accounting through the smart contracts. The computing resources of the blockchain nodes are only used for blockchain ledge service, do not provide computing power, network traffic, and storage resources to undertake other computing tasks.

After the Stratos main network goes online, we will announce blockchain node election program publicly. Main blockchain nodes will be elected from all candidate nodes, and several nodes will be selected as backup blockchain nodes.

2.4.2. Resource Node

The resource node is the foundation facility of the entire decentralized computing and storage network. General speaking, a resource node is a general-purpose computing server that is deployed in a professional computer room and has sufficient computing power, network traffic, and storage resources. In addition to basic support capabilities such as communication, encryption, file system and database, resource nodes will also serve as a general computing platform, providing various application-oriented functional calling interfaces. The network traffic of the resource node will be regularly reported to the meta node, and the corresponding token will be obtained as an incentive.

The blockchain nodes are mainly constructed and provided by third parties. After the Stratos main network is online, any third-party computing resources which meet the Stratos requirements can join the

Stratos resource network.

2.4.3. Meta Node

The meta nodes are deployed independent from the resource nodes. The meta node is mainly responsible for

- Statistics of computing power, network traffic and storage resources
- Indexing the resources located on the resource nodes.
- Routing service for accessing the content on the resource nodes.
- Analyze computing task demands, allocate the computing power, network traffic and storage of corresponding resource nodes to meet corresponding requirements intelligently.

  Take the user storing files in the Stratos network as an example. After any meta node receives a request, either uploading of downloading, it creates a task for the request. Any meta node can pick up the task and then find the right resource node according to certain algorithm to fulfill the request. The factors that affect resource node location algorithm are:

  - *total storage space of resource nodes,*
  - *available storage space on each resource node*
  - *online time of resource nodes*
  - *number of resource node connections*
  - *location of nodes*
  - *number of random factors*

- Balance the storage and traffic in the resource network
- Make sure the slice replication number for each file and the accessibility of files in the resource network
- Dynamically expand/shrink the resource allocation.

CHAPTER 3

STRATOS CONSENSUS

3. Stratos consensus

3.1. Tendermint

Tendermint [4] (Consensus without Mining) was designed and published by the Cosmos team in 2014. Tendermint consensus mechanism provides proof of work that does not rely on high power consumption such as mining, but provides fast transaction processing capabilities, security and scalability for blockchain projects which adopt this consensus mechanism.

3.1.1. Tendermint Concept

- Tendermint is software that can safely and consistently replicate applications on different machines. Security and consistency are also key concepts of distributed ledger.
- Safety means that if even up to 1/3 of the machines are failed, Tendermint can still work normally.
- Consistency means that each machine that works normally always has the same transaction log and calculate the same state.
- Tendermint is a Byzantine fault-tolerant consensus algorithm.

3.1.2. Tendermint consensus algorithm and block structure

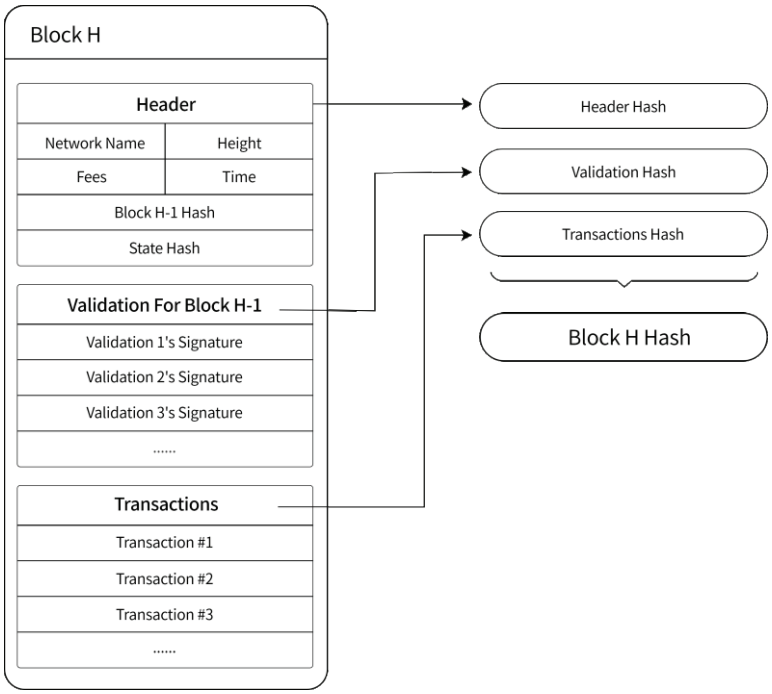

FIGURE 9. Block structure

3.1.3. Tendermint state machine workflow

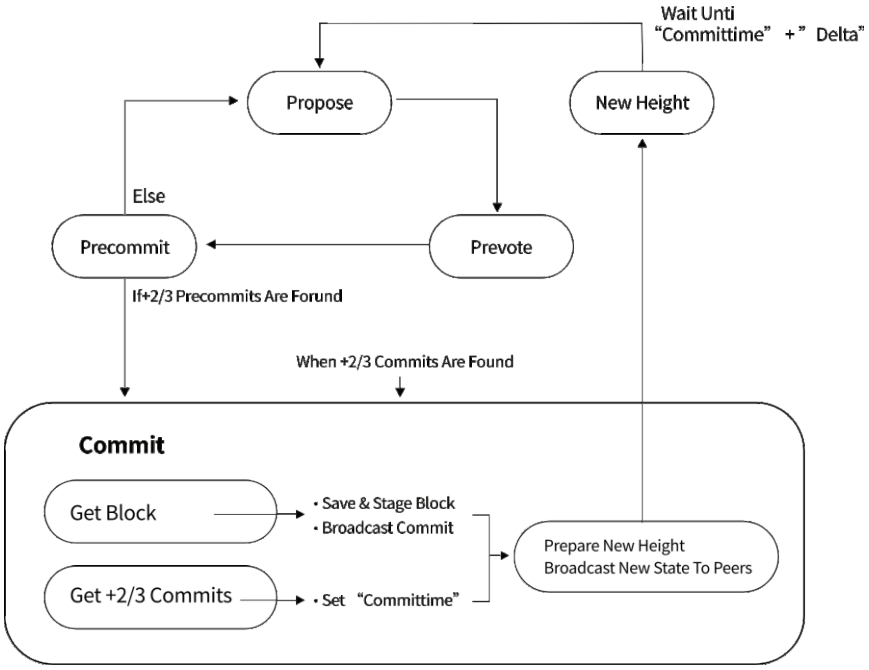

FIGURE 10. Tendermint state machine workflow

3.1.4. Tendermint Consensus algorithm main steps

1. Proposing
   Select the block producer node among the nodes which participating the consensus process.

2. PreVote
   The block producer node broadcasts the new block, then other nodes verify the new block. If more than 2/3 of the nodes confirm the validity of the block, then the process moves to the next stage, otherwise the process fails.

3. PreCommit
   After 2/3 nodes verify the validity of the new block, all nodes start to write the block to the local cache. If more than 2/3 of the nodes get it done successfully, then the process moves to the next step, otherwise the process fails.

4. Commit
   After the process receiving the signal from more than 2/3 nodes which wrote to the local cache successfully, all the nodes move the new block data from local cache to local database.

5. New Height
   Transition time window to prepare for the next block production.

3.1.5. Synchronous and Asynchronous

In the computer field, synchronization is the sequential execution in the entire processing process, and the result will be returned only after each process has been executed by order. Synchronization can be understood as serial, and the flow of the whole process cannot cross each other.
Asynchronous means just sending the called instruction, the caller does not have to wait for the called method to be completely executed, but continues to execute the following process. Asynchronous is a parallel processing method. You can perform other tasks without waiting for the execution of a program.

FIGURE 11. Sync .VS. Async

The difference between synchronous and asynchronous is that one needs to wait, and the other does not need to wait.

As for consensus, the proof-of-work of Bitcoin and Ethereum is actually an asynchronous consensus process, the entire consensus process has no any dependency between inside steps. Any node that gets the validated hash value can broadcast the new blocks to the network at the same time. Each node in the chain works independently during the consensus process and does not need to communicate and cooperate with each other.

For the Tendermint consensus method (BFT+POS) that we adopted, in addition to solving the energy consumption problem, it also solves the problems of POW transaction confirmation, transaction bifurcation, and low performance. The leap brought by it cannot be ignored. Moreover, many famous projects such as EOS, Tezos, Cosmos, etc. all use the consensus method of BFT+POS.

Then the key point of our thinking is that although BFT+POS has many advantages, but the consensus process is a synchronization process of the entire chain. The two stages from Pre-Vote to Pre-Commit requires the entire chain to exceed only 2/3 of the votes can continue to produce blocks, otherwise the consensus process will fail, then all nodes in the entire chain will conduct consensus processing again until they get 2/3 of the votes of the entire chain.

If more than 1/3 of the nodes in the chain have poor network quality or poor machine performance, then other faster nodes and network resources will have to wait for slower nodes (provided that more than 1/3 of the nodes are worse. If the number of bad nodes is less than 1/3, the chain will ignore these slower nodes and produce blocks directly)

Is it possible to adjust some steps in the Tendermint (BFT+POS) consensus, such as:

1. Remove the “New Height” round, because each “New Height” round is the entry point of a synchronous process.

2. Set the two steps of “Pre Vote” and “Pre Commit” are processed asynchronously, if the response cannot be made in the same time window, we will establish a buffer pool, save the temporary data and wait for the response of other nodes before processing, instead of pausing Consensus processing and just waiting.

Based on the above thought, we redesigned the Stratos consensus workflow.

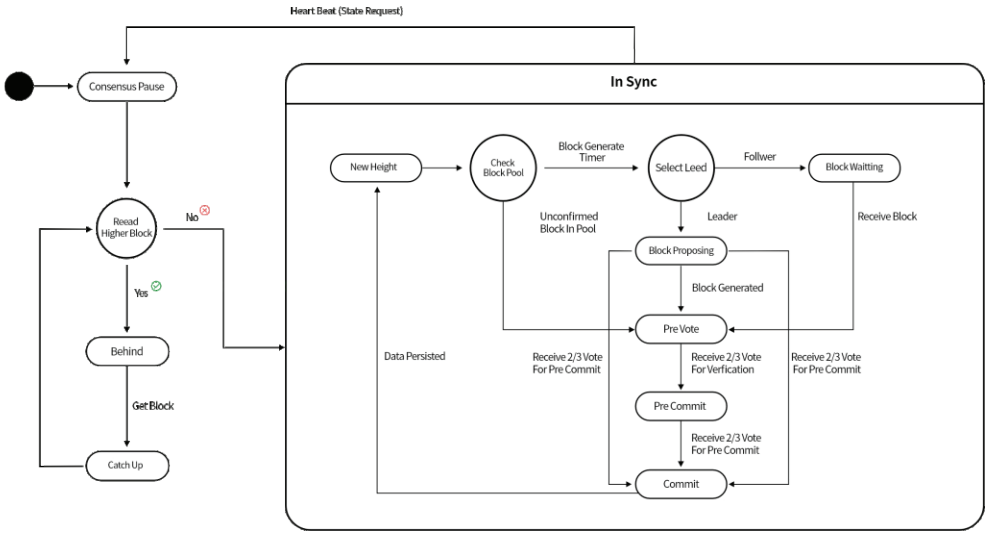

FIGURE 12. Stratos consensus workflow

The above are some thought of the team and design on the possibility to asynchronous Tendermint process. Next, the team will conduct more research and discussion deeply to determine the feasibility of the asynchronous Tendermint consensus. In the end, the development team will use Tendermint as the basis to develop an enhanced version of Tendermint or unique and efficient PBFT consensus algorithm for Stratos.

3.2. Resource Proof-of-Traffic Consensus

In a decentralized network, it is very difficult to prove that a certain resource provider truly provides a certain amount of storage, calculation, and network traffic resources through Zero-Knowledge Proof [5]. And zero-knowledge proof itself also requires a lot of calculation and verification. If Stratos uses zero-knowledge proof to verify the resources provided by storage network resource nodes, a large amount of

computing resources and network traffic will be used to verify storage, calculation, and network traffic resources. Then the available resources of the entire network can really provide to users will be greatly reduced. In this way, it completely violates the original intention of Stratos design. Therefore, Stratos cleverly separates the interest entanglements of each participant through the method of Proof-of-Traffic, so the storage, computing, and network traffic resources can be proven by multi-party verification of network traffic. Use Token to incentivize decentralized computing resource providers to ensure and promote the effective operation of computing resource supply and demand models.

3.2.1. Proof-of-Traffic Process

In the initial resource network, assuming that each ledging period is 10 minutes, the steps are roughly as follows:

1. Meta nodes generate task for all kind of network resource access.

2. Both clients and Resource nodes will receive and report the status of the task

3. Meta nodes gather the task status reported by both clients and resources nodes and record the network traffic data of the current period and reports the summary to the blockchain node.

4. Meta nodes sort the resource nodes based on traffic of each node.

5. Meta nodes select the top nodes in the entire network and generate the traffic report then send to the blockchain node. The remaining node traffic data will be rolled into the next ledger cycle for accumulation.

6. Based on the traffic sending to the blockchain, the top nodes in the entire network in terms of throughput in this ledger period will receive Token rewards in the blockchain node. In a given ledger period, the number of Tokens that a node should obtain is equal to the total amount issued in the current period in the Token issuance smart contract multiplied by the throughput of the node in this period and the sum of the throughput of all Token nodes obtained. After obtaining the Token incentive, the node that obtained the allocated Token is cleared, and the next ledger cycle is re-entered.

   For example, suppose that a total of 100,000 Tokens are awarded to the top 100 nodes in the entire network in terms of throughput in this ledger period, the throughput of node A in this ledger period is 100GB, the total throughput of the top 100 nodes in the entire network is 1000GB, then the reward of node A in this ledger cycle is 100GB/1000GB*100,000 = 10,000 tokens.

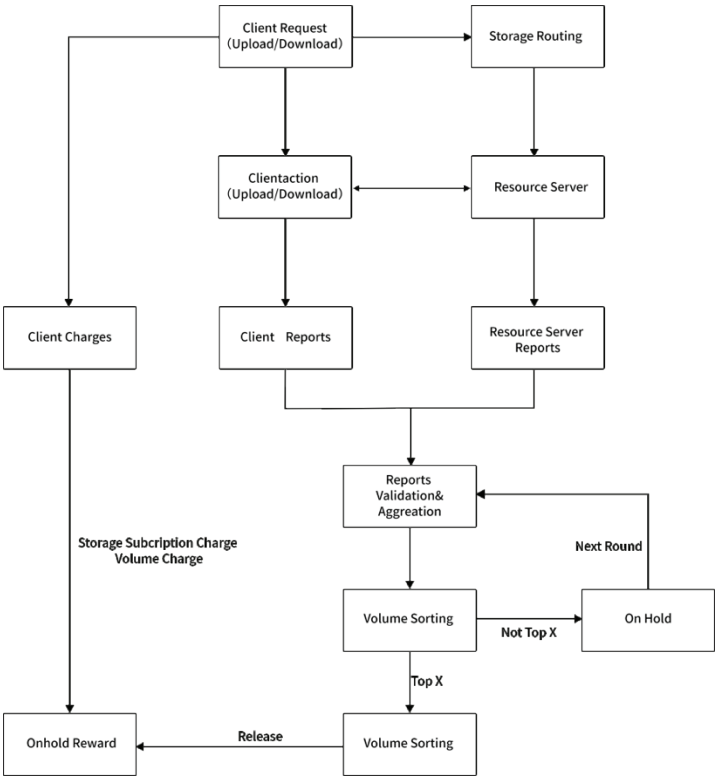

FIGURE 13. Proof-of-Traffic consensus workflow

3.2.2. Proof-of-Traffic Effective analysis

The core idea of Proof-of-Traffic is to calculate approximate the resources (storage, computing, traffic) usage via using the traffic actually used by the user. Here we will use the data traffic as an example.

In order to ensure that the data traffic is valid, the Stratos resource network meets the following three conditions:

1. File storage distribution is sufficiently random.
2. The file must have at least 2 or more backups.
3. The client will be assigned to download from a resource node randomly where the backup is located.

If condition 1 is not met, part of the using of resource nodes will be largely idle and some will be congested, and the amount of traffic cannot be calculated by the amount of usage.

If condition 2 is not met, it does not meet the fault tolerance conditions of the distributed system, and it couldn't avoid the cheating behavior of resource nodes imitating customers to brush traffic on their own files.

If condition 3 is not met, the client's report cannot correctly reflect the disaster recovery backup situation

of the storage network, resulting in a situation where some nodes actually fail but cannot be fed back.

Since the Meta node contains all the index data, the client only needs to report the status of file accessing tasks. The meta node can compare client report with the task report from the resource node and charge the client's traffic usage. The traffic fee and the user's monthly storage resource fee will be settled to all nodes that contain the replication nodes.

When the above three conditions are met at the same time,

1. The client traffic can accurately reflect the total amount of storage network resources.
2. Effectively prevent traffic cheating, because the revenue from traffic cheating will be shared by random other nodes, and the revenue from this behavior is negative
3. Effectively prevent the fake traffic report, because of the fake report will be clearly reflected in the report which is submitted by client

3.3. Proof-of-Authority Consensus for Meta Service layer

Proof of Authority (PoA) is a reputation-based consensus algorithm that uses the credit of identity and reputation as a verifier. The proof-of-authority model relies on a limited number of network validators, making it as a highly scalable system. The pre-approved nodes select the master according to the certain rules and the elected master manages the network.

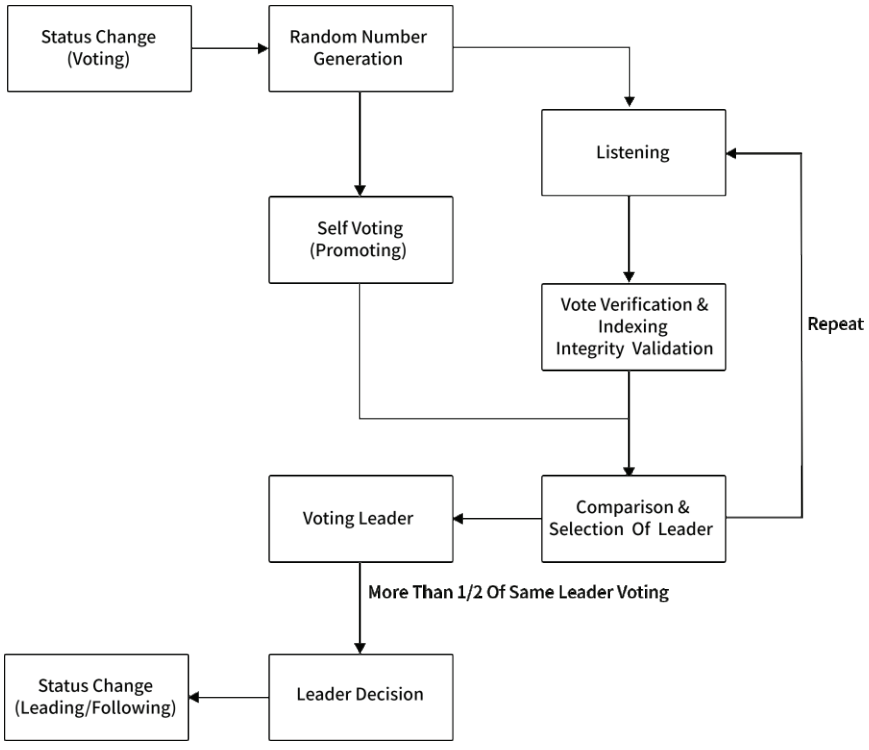

FIGURE 14. Proof-of-Authority consensus workflow

Stratos will use PoA consensus to select master node from the meta service network. Using PoA can ensure that no matter the numbers of meta nodes participating in the process, highly efficient consensus process always can be guaranteed.

Election Implementation

Server status:

- Voting: election status, looking for leader
- Following: following status, synchronize the leader status, participate the voting process
- Observing: Observing status, synchronize the leader status, doesn't participate the voting process
- Leading: leader status

Voting data structure:

Each vote contains two basic pieces of information: the address ID of server of voting to and a randomly generated positive integer.

Election process:

The new round election begins after the last elected master has completed the tasks.

Suppose there are 5 nodes are running POA consensus, from server1 to server5

1. Generate random number
   Each server generates a random integer. For example, Server1 generates 1052, Server2 generates 10879

2. Change status
   Each server updates its status to Voting, and then enters the leader election process.

3. Each server votes
   During the process, the ID of each server is different. In the first round of voting, each server will vote itself, generate a vote, and then send the vote to all other servers.

4. Accept votes from other server
   After each server receives a vote, it first judges the validity of the vote, such as checking whether it is legal vote for the current round of voting or whether it is from a server in the voting state.

5. Vote process
   For each vote, the server must PK other votes with their own votes. The PK rules are as follows:
   a. Check the random number of each server first, the server who has a larger random number will be the leader.
   b. If the random numbers are the same, then compares the machine AddressID, and the larger one is selected as Leader.

6. Vote Statistics
   After each vote, the server will count the voting information to determine whether more than half of the server have received the same voting information. If so, the top one server will be elected as the leader.

7. Change the server status
   Once the Leader is selected, each server will update its status. If it is a follower, the status will be changed to Following, if it is a Leader, it will be changed to Leading.

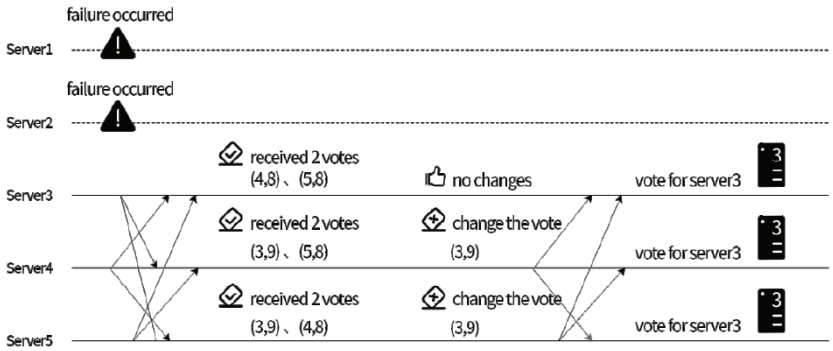

FIGURE 15. Proof-of-Authority election process

CHAPTER 4

STRATOS BLOCKCHAIN AND RESOURCE NETWORK

4

4. Stratos blockchain and resource network

4.1. Stratos Blockchain

Stratos Blockchain is implemented by the core R&D team from scratch and the main language is Go.

4.1.1. Merkle Patricia Tree

Merkle Patricia Tree [6] (also known as Merkle Patricia Trie) is an improved data structure that combines the advantages of Merkle tree and prefix tree.
The MPT tree has the following functions:

- Stores any length of key-value pair data
- Provides a mechanism to quickly calculate the hash identifier of the maintained data set
- Provides a mechanism for fast state rollback
- Provides Merkel proof to expand light nodes and realize simple payment verification

In the Account status development module, our development is similar to Ethereum's account management. Therefore, we choose to use the Merkle Patricia Tree which is used by Ethereum to organize and manage account data, transaction collections, smart contract data, and important data structures for commercial business data.

4.1.2. Database Development

Our team chose the LevelDB as the embedded database for blockchain node. LevelDB is a fast key-value storage library. The data in LevelDB is stored sorted by key and it also support multiple change in one atomic batch. The LevelDB is the database applied by Bitcoin for storing Chain states. Chain state, accounts, smart contracts, and transaction data of Stratos will all be written to LevelDB.

4.2. Stratos Resource Network

General speaking, the Stratos resource network is composed of resource nodes. Resource nodes provide computing power, storage, and content accessing services. They are also responsible for managing the network, allocating various resources, and ensuring a balanced ratio of all aspects of the network.

4.2.1. SDS Overview

SDS is the abbreviation name of Stratos Data Service, SDS is designed to be a distributed file system suitable for running on general-purpose hardware. SDS is a highly fault-tolerant system, suitable for deployment on various servers or personal computers. It can provide high-throughput data access and is very suitable for large-scale data applications.

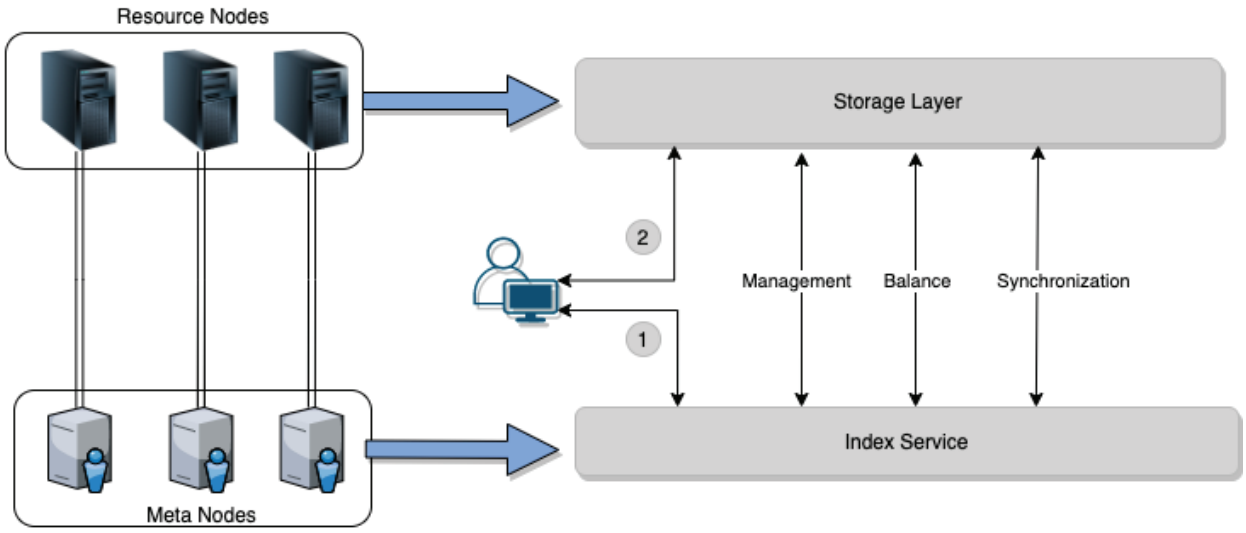

FIGURE 16. SDS Overview

4.2.2. SDS Working Principle

SDS is a decentralized file system used to store and manage various files, through a decentralized indexing service (namespace) to manage all files and relevant fragments globally.

1. The files in SDS are physically stored in blocks, and the block size can be defined by configuration parameters.
2. The SDS file system provides a unified abstract directory tree to the client, and the client accesses files through paths.
3. The indexing service manages the directory structure and file block location information, and the indexing service is responsible for the directory tree of the entire file system and the data block information corresponding to each path.
4. The storage management of each block of the file is undertaken by the resource node, the replications of each block of file will store on different resource nodes.
5. The resource node reports the file block information to the indexing service regularly, the indexing service will be responsible for keeping the number of copies of the file. The internal working mechanism of SDS is transparent to the client, the client requests to access SDS through the indexing service.

SDS uses Merkle Tree to store a single file in fragments. IPFS first proposed to use Merkle DAG [9] (Merkel Directed Acyclic Graph) for slicing files and storing them on the entire network, but this will inevitably cause the proliferation of cited information on the entire network and cannot improve the availability of files. Because the loss of any fragment will cause the entire file to be unusable. SDS file indexing only reaches the level of resource nodes. File indexing and file backup are performed on the entire network. At the same time, files are fragmented at the resource node level and indexed for storage. This can ensure the same file availability while minimizing the amount of information cited on the entire network. SDS further simplifies the Merkle DAG. The dendrogram itself is a directed acyclic graph but removes the situation that different branches of the directed acyclic graph merge into one branch, that is, a child node cannot have different parent nodes. This greatly simplifies the complexity of integrating fragmented files, while making parallel computing possible.

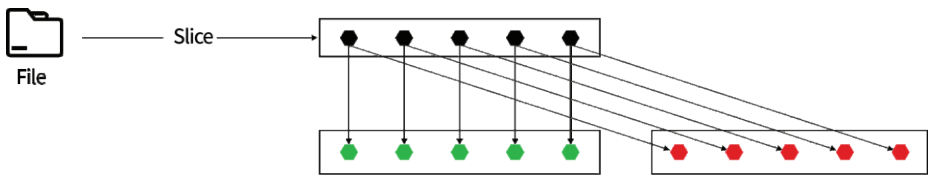

FIGURE 17. A file is divided into 5 slices(black), and each of 5 slices(black)are saved two copies (green and red respectively)

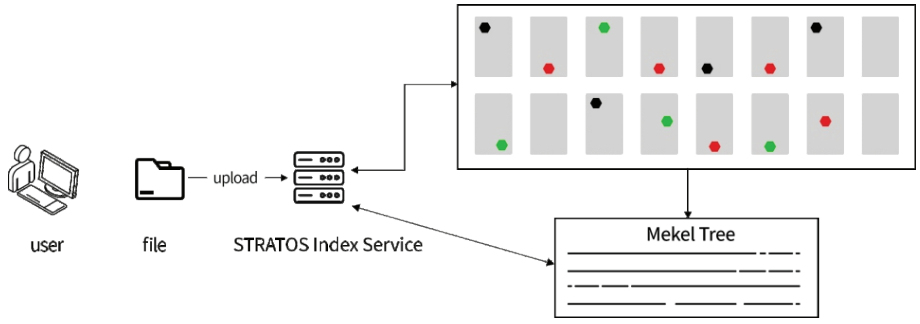

FIGURE 17. On the Stratos system, the slice and corresponding copies of each uploaded file are stored in di erent resource nodes managed by SDS.

4.2.3. SDS Index Service

SDS index are distributed and replicated on multiple nodes of the Stratos meta nodes. And the index is consistent on all the participated nodes. The following diagram shows how the index service working across the network.

Features of index service:

- Stratos meta nodes are independent from resource nodes
- Index service running on all meta nodes.
- In case any of the meta node out of service, backup meta node will activate its indexing service and storage to join the meta service network.

- Async data service to update the inactive instance. This process ensure that a new index node can join the cluster more efficient and won't affect the performance of the leader during the replication procedure.
- Discovery service are used for dynamic cluster configuration.
- All SDS meta nodes are responsible to do health checking and monitoring.
- Health checking service is responsible to add an inactive Storage module from a SDS node to the cluster in case of some nodes out of service.

4.2.4. SDS upload data

To write data to SDS from the client side, the client must have to communicate with the index service to confirm that it can obtain the resource node which store the data, then the client uploads the file slices to the connected resource node in order. the resource nodes who received the data slices is responsible for replicating copies of the slices to other resource nodes.

1. The client sends a file upload request to the index service, the index service checks the directories and files to be uploaded, then returns the result to the client.

2. The client loads the local configuration after obtaining the permission to upload the file, then requests the service from index service for uploading a data slice.

3. The index service will inquire the available resource node information according to the client local configuration. When possible, the index service will choose the better resource nodes back to user client based on the "high-quality storage" and "physical distance" etc... key conditions. This method is called "offsite Backup awareness" strategy.

The client will cache the data locally before starting to transmit the data block. When the cache size exceeds the size of one data block, the client will obtain the list of resource nodes to be uploaded from the index service. After that, a connection will be established between the client and the first resource node to start streaming data. This resource node will receive a small part of the data and write it to the local storage, then transmit the data to the second resource node. The second resource node also receives a small part of the data and writes it to the local storage, then transmits it to the third resource node, and so on. After the call and return step by step, and after the data transmission is completed, the client report to the index service that the data transmission is completed and then the index service will update the relevant metadata information record.

4. After the first data slice is transferred, the following slices will be transferred in the same way until the entire file data is uploaded.

5. All resource nodes will run the index service. The PoA consensus will elect a resource node as the master of index service. The master index service will synchronize the full index to all other resource nodes, so each node in the resource network is identical. In the event that the master index service is offline or re-elected accidentally, the elected node can seamlessly switch roles to assume the master index service.

6. The master node is also responsible for heartbeat service and QoS service, and performs health checks on other nodes. In case that any node can't maintain the quality of service or is offline, it is responsible for transferring resources on the node to other available nodes to keep the minimum replication number of file slices and update the index at the same time.

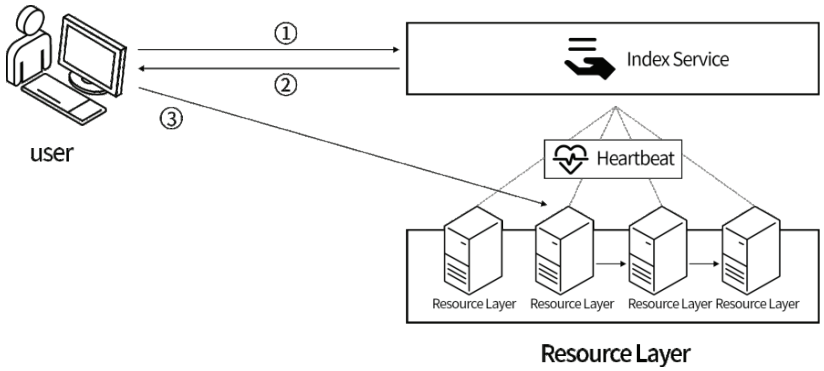

FIGURE18. SDS upload process

4.2.5. SDS download data

The client sends the requested file path to the index service. The index service obtains the meta-information of the file (mainly the file slice location information) and returns it to the client. The client connects to the corresponding resource node according to the meta and obtains the file slice one by one, then append and merge data on the client side to have the entire file.

1. The client initializes an RPC call to the index service for requesting to get file data.
2. The index service verifies whether the file exists. If the file does, it obtains the file metadata which include the slice ID and the corresponding resource node list.
3. After the client received the meta-data, it selects the closest resource node and requests to retrieve each slice in turn. The client verifies the slice is damaged or not, if the slice is damaged, the client will select a different resource node to request again.
4. The client talks to the resource node and transfers data via socket connection. The client caches the received data locally then writes it to the file. Transferring process will keep running until the all the slices are received in the client side and the entire file is merged successfully.

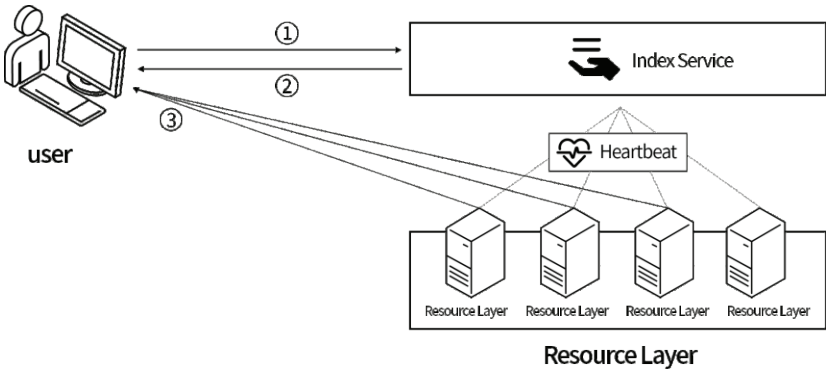

FIGURE 19. SDS download process

4.2.6. Bloom Filter quick search engine

A Bloom filter [10] is a space-efficient probabilistic data structure, conceived by Burton Howard Bloom in 1970, that is used to test whether an element is a member of a set. False positive matches are possible, but false negatives are not – in other words, a query returns either “possibly in set” or “definitely not in set”. Elements can be added to the set, but not removed (though this can be addressed with the counting Bloom filter variant); the more items added, the larger the probability of false positives.
Bloom proposed the technique for applications where the amount of source data would require an impractically large amount of memory if “conventional” error-free hashing techniques were applied. He gave the example of a hyphenation algorithm for a dictionary of 500,000 words, out of which 90% follow simple hyphenation rules, but the remaining 10% require expensive disk accesses to retrieve specific hyphenation patterns. With sufficient core memory, an error-free hash could be used to eliminate all unnecessary disk accesses; on the other hand, with limited core memory, Bloom’s technique uses a smaller hash area but still eliminates most unnecessary accesses. For example, a hash area only 15% of the size needed by an ideal error-free hash still eliminates 85% of the disk accesses.[2]
More generally, fewer than 10 bits per element are required for a 1% false positive probability, independent of the size or number of elements in the set.[2]

According to the specific feature of Bloom Filter, the SDS index service will use Bloom Filter to determine whether the abstract file directory tree and the block information corresponding to each file exist quickly, thereby improving the overall service performance of SDS.

CHAPTER 5

STRATOS APPLIED SCENARIOS AND BUSINESS APPLICATIONS

5

5. Stratos applied scenarios and business applications

5.1. Stratos data mesh

Stratos data mesh is deeply bound by blockchain and a resource layer. Stratos blockchain not only provides the resource layer's workload calculation and settlement service, but it also provides payment services and document verification for a series of data mesh services.
Any commercial customer or individual user who uses Stratos data mesh must have a valid Stratos wallet with a certain amount of Token for applying any decentralized computing services.

5.1.1. Code and Database hosting services

Stratos provides new structured data hosting and unstructured data storage services, user can query their data via sample APIs anytime and anywhere.
Developers can store any type of data in the Stratos decentralized data mesh completely according to their own needs. So, the developers and the Dapps no longer worry about the risks to come from the centralized system.

5.1.2. Web Service deployment for blockchain community

To optimize the use environment of the entire blockchain network, developers can choose to deploy web services on decentralized nodes. Stratos platform schedule the corresponding services provided to users with better resources and bandwidth through the Coordination module of Stratos.

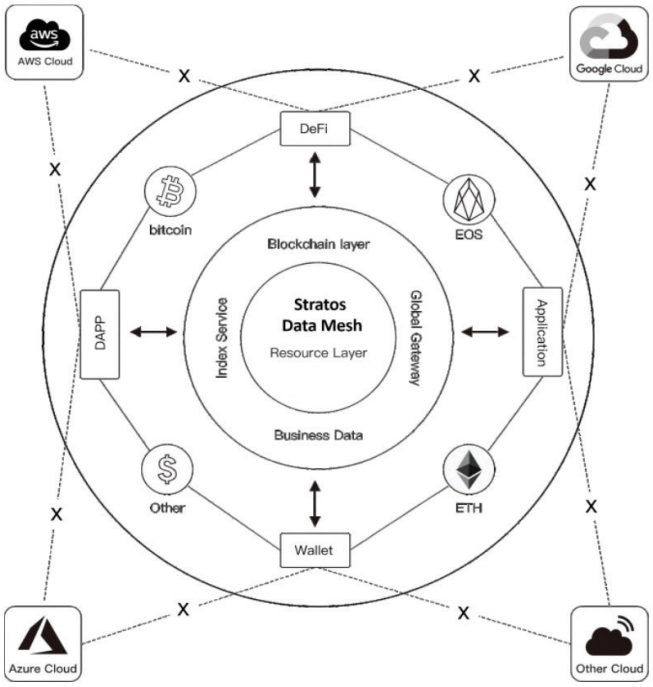

FIGURE 20. Data Mesh position in the blockchain ecosystem

The following diagram shows the applications in the ecosystem of the Stratos platform.

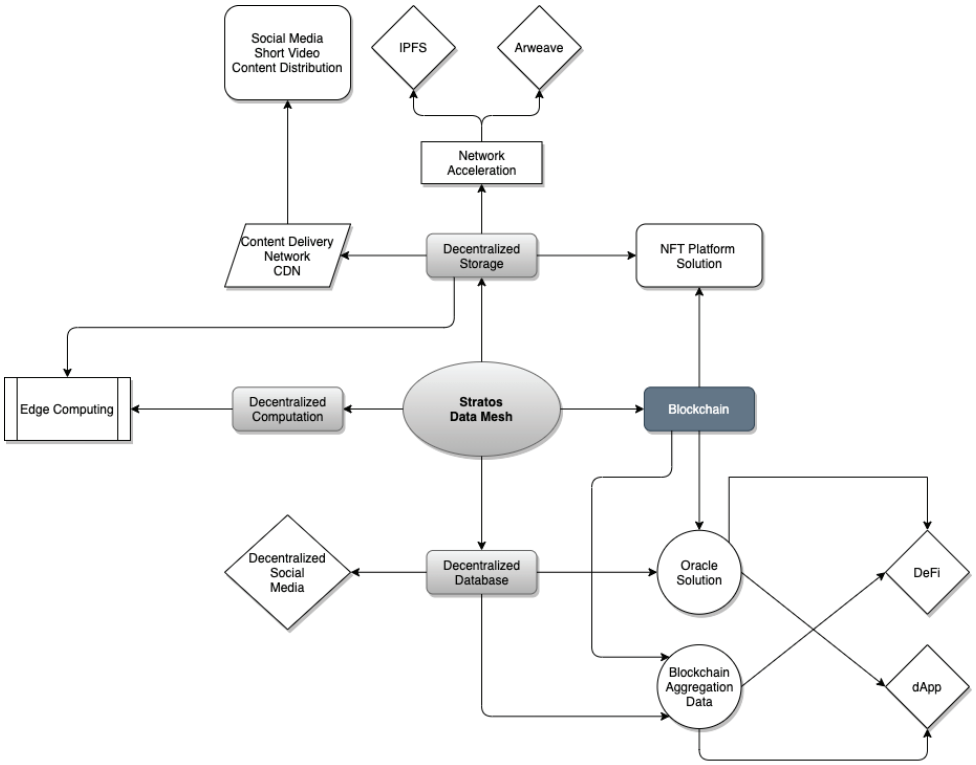

FIGURE 21. Applications ecosystem in the Stratos platform

The three core parts of Stratos:

- Stratos Blockchain
- SDS Storage Service
- PoT platform connect to Blockchain and Storage Service

The application currently being developed by the Stratos team:

- Decentralized storage
- Network Acceleration
- NFT Market
- Decentralized Database

The other parts are the various types of applications that third-party developers can develop based on our core platform in the future.

With the establishment of the ecosystem, the Stratos team will cooperate with third parties to develop a variety of rich applications for different users and expand the support capabilities of the platform.

CHAPTER 6

THE FUTURE OF STRATOS

6. The future of Stratos

At present, Stratos is still in its infancy. As the project evolves, it will surely attract numerous community developers and users, and then the entire Crypto community of developers will gradually perform unimaginable power and unleash great potential.

Stratos will develop into an information exchange and transmission infrastructure as important as the Internet in the future, which will greatly reduce the credit cost of the entire society and promote the transformation of the Internet of Information to the Internet of value. And will compete with the major monopoly Cloud vendors, and ultimately obtain half of the market.

Stratos data mesh will bring a huge revolution to the data industry and accelerate the process of data sovereignty transitioning from centralized system to new democratic decentralized system. Eventually, the ownership of all the data will be back to the one who create them, everyone will release a huge creation ability from data freedom.

CHAPTER 7

REFERENCES

7

7. References

[1] Bitcoin whitepaper: https://bitcoin.org/bitcoin.pdf

[2] Practical Byzantine Fault Tolerance Paper http://pmg.csail.mit.edu/papers/osdi99.pdf

[3] Gavin, Wood (November 2015) https://github.com/ethereum/guide/blob/master/poa.md

[4] Tendermint Paper https://tendermint.com/static/docs/tendermint.pdf

[5] Blum, Manuel; Feldman, Paul; Micali, Silvio (1988) Non-Interactive Zero-Knowledge and Its Applications

[6] Merkle Patricia Tree https://eth.wiki/en/fundamentals/patricia-tree

[7] Daniel J. Bernstein (2017-01-22) “Ed25519: high-speed high-security signatures”

[8] Groovy multi-faceted language https://groovy-lang.org/

[9] Merkle Distributed Acyclic Graphs https://docs.ipfs.io/concepts/merkle-dag/#further-resources

[10] Blustein, James; El-Maazawi, Amal (2002) Bloom Filters — A Tutorial, Analysis, and Survey

Power you data with Stratos