TAN

New decentralized content distribution and computing network based on blockchain technology

Rev.1

ABSTRACT

This white paper introduces the TAN network, which is a new blockchain and token, as an incentive mechanism for decentralized content distribution and distributed computing networks.

The TAN network and protocol solve various challenges faced by today's content distribution and distributed computing industries. First, the tokens on the TAN blockchain are used as incentives to encourage individual users to share their redundant computing and bandwidth resources, as data caches or relay nodes, and edge distributed computing nodes. This improves the quality of data calculation and transmission and solves the "last mile" problem, which is the main bottleneck of traditional content calculation and transmission pipelines, especially high-resolution live streaming, VR virtual reality, large file distribution and edge computing . Second, if the network density is large enough, most viewers will extract data from peer-to-peer cache nodes, allowing cloud computing platforms to significantly reduce the cost of content delivery network (CDN) and edge computing (Edge). More importantly, by introducing tokens as an incentive mechanism for end users, the TAN network allows the cloud computing platform to deepen audience participation, drive revenue growth, and differentiate its content and viewing experience from competitors.

The TAN blockchain introduces three main new concepts:

1．Multi-level BFT

An improved BFT consensus mechanism that allows thousands of nodes to participate in the consensus process while still supporting very high transaction throughput (1,000+ TPS). The core idea is to have a small group of nodes, which form a committee of validators, and use a PBFT-like process to generate a blockchain as quickly as possible. Then, thousands of consensus participants, called guardians, determine the chain generated by the verifier committee at the regular checkpoint block. The name multi-level BFT consensus mechanism reflects the fact that the verifier/guardian division provides multi-level security assurance. The validator committee provides the first level of consensus-10 to 20 validators, and the committee can quickly reach a consensus. The Guardian Pool constitutes the second line of defense. With thousands of nodes, it is more difficult for an attacker to destroy the integrity of the network, thus providing a higher level of security. We believe that this mechanism has achieved a good balance between transaction throughput, consistency and decentralization, which is the three pillars of the so-called "impossible triangle".

2．Aggregate signature broadcast plan

The basic all-to-all broadcast of the checkpoint block hash can work between guard nodes, but it will incur secondary communication overhead, so it cannot be expanded to more than 1,000 nodes. On the contrary, we propose an aggregation signature gossip scheme, which significantly reduces the complexity of message delivery. Each guardian node continuously combines some of the aggregated signatures from all its neighbors, and then broadcasts the aggregated signatures. In this way, each node's signature share can use the broadcast protocol to reach other nodes at an exponential rate. In addition, signature aggregation keeps the size of node-to-node messages small, thereby further reducing communication overhead.

3．Resource-oriented micropayment pool

An off-chain "resource-oriented micropayment pool" built specifically for content distribution and distributed computing. It allows users to create an off-chain micropayment pool, any other user can use off-chain transactions to exit the pool, and has double payment capabilities. Compared with offline payment channels, it is more flexible.

This white paper will describe these concepts and the TAN blockchain in detail. TAN network launched ERC20-compliant tokens. The TAN blockchain mainnet code has been released, and it is planned to launch the first mainnet launch on March 15, 2021. At that time, each ERC20 TAN token will be exchanged for local TAN tokens at a ratio of 1:1.

4．Goal introduction

Content distribution and distributed computing market

CDN, or content distribution network, refers to adding a new network architecture to the existing Internet to publish the content of the website to the network "edge" closest to the user, so that users can obtain the content they need nearby, and increase users. The service form of the response speed of visiting the website.

Since Akamai proposed the concept of CDN in 1998, the development of the CDN industry has experienced twists and turns. Around 2000, affected by the Internet bubble, the development of CDN nearly stagnated, and CDN service providers dropped from 50 to 10. Since then, the Internet wave from 2004 to 2006 has injected vitality into the CDN market, and the CDN industry has begun to recover. After 2007, CDN entered a rapid development track with the explosive growth of video applications.

American CDN service provider Akamai is firmly in the leading position of global CDN service provider. In 2017, it achieved CDN business revenue of US$2.5 billion, a year-on-year increase of 7%, and accounted for 33.8% of the global CDN market in 2017.

In the second echelon, Amazon CloudFront, EdgeCast, CloudFlare, MaxCDN, etc. Among them, Amazon CloudFront accounted for approximately 16.2% of the global market size; the rest accounted for approximately 19.5% of the global market size.

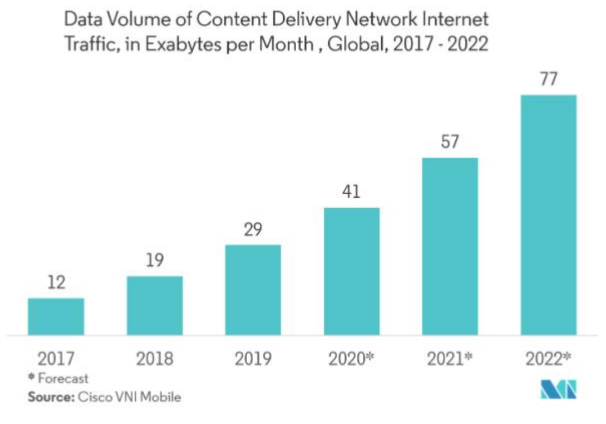

Figure 1. Global CDN market growth

At the same time, according to a report released by Fior Markets, the global edge computing market is expected to grow from US$2.8 billion in 2019 to US$18.36 billion in 2027. Moreover, it will grow at a compound annual growth rate of 26.5% during the

2020-2027 forecast period.

The major players in the global edge computing market include Cisco, Amazon (AWS), Huawei, General Electric, Nokia, IBM, Intel, Machineshop, Microsoft and Litmus Automation.

The vertical market segments of the edge computing industry include smart cities, wearable devices, data centers, retail, transportation, logistics, agriculture, healthcare, energy, utilities, and industry. As major players increased their investments in rapidly changing technologies to drive the growth of the edge computing market, the transportation sector accounted for the largest market share of 32.57 in 2019.

It is expected that North America will occupy the largest market share in the global edge computing market at 42.32%, which is attributed to the increase in technology acceptance and the influx of suppliers in North America. In addition, the growing demand for fully automated and connected cars is also driving the continued growth of the edge computing market in North America.

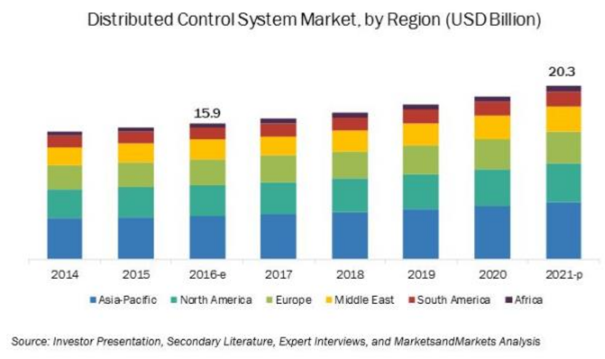

Figure2. Global distributed market growth

4.1 Problems encountered in the current

Content Delivery Network (CDN) and Distributed Edge Computing (Edge) play an important role in the Internet ecosystem. It provides the backbone infrastructure to deliver the video stream to the final audience. A major limitation of today's CDN networks is the so-called "last mile" delivery problem. Generally, CDN providers have established data centers called nodes (POPs) in many places around the world and expect these POPs to be geographically close to the audience. However, the number of these nodes is limited, so they are not close enough for most viewers, especially in underdeveloped areas. This "last mile" link is often the bottleneck of today's distributed pipelines, and often leads to a less than ideal user experience, including intermittent data transmission, poor network delay quality, and frequent rebuffering.

For platforms that provide these services, another major issue is the cost of CDN bandwidth. For popular sites, the bandwidth costs of cloud computing servers and CDNs can easily reach tens of millions of dollars per year. Even if the platform has a proprietary self-built computer room, maintenance costs are often high.

With the advent of the 4K, 8k, and 360° VR streaming media era, as well as new technologies that are about to become popular, such as cloud gaming and big data artificial intelligence computing, these problems have become more prominent.

5．Market opportunity

The mission of this project is to use blockchain technology to create the first high-quality decentralized content delivery and edge computing network, and encourage video viewers to share redundant computing and bandwidth resources to meet today's content distribution and distributed computing challenges . The TAN network can be regarded as a huge "shared computer in the world" formed by the computing power, memory and bandwidth resources contributed by the audience.

Specifically, audiences around the world can contribute their devices as "caching and computing nodes" to form a transmission and computing network, responsible for any specific data cache transmission and edge computing to local optimizations around the world. user. The TAN network can effectively solve the technical challenges discussed in the previous section. First, compared with CDN nodes, viewers' devices are geographically closer to each other. This reduces the round trip time of the data packet and improves the quality of the data stream transmission, thus solving the "last mile" transmission problem. Second, with a sufficient number of cache nodes, most viewers will receive data streams from the cache nodes, which will help cloud sites reduce their computing and CDN bandwidth costs. Third, the cache node also reduces the round-trip time, making edge computing and next-generation video technology practical.

In order to encourage viewers to contribute their computing and bandwidth resources, we have introduced TAN tokens as an incentive mechanism. Cache nodes can earn tokens when forwarding data streams to other viewers. TAN tokens not only motivate viewers to join the network as cache nodes, but also greatly improve the market efficiency of cloud platform operations by simplifying the video transmission process.

The full launch of the TAN protocol introduced a new blockchain and native token structure, among which:

● Compute nodes earn tokens, which are used for data flow calculation and forwarded to other viewers

● The audience selectively receives tokens from the operator as a reward for participation；

● Cloud computing platforms can increase new revenue by selling cheaper and more efficient services, and deepen user engagement through TAN；

● Sites using cloud computing can share up to 80% of CDN costsThe TAN protocol is based on the following new concepts；

5.1 Multi-level BFT

An improved BFT consensus mechanism that allows thousands of nodes to participate in the consensus process while still supporting very high transaction throughput (1,000+ TPS). The core idea is to have a small group of nodes, form a committee of validators, and use a process similar to PBFT to generate a blockchain as quickly as possible. Then, thousands of consensus participants, called guardians, can determine the chain generated by the validator committee at regular checkpoint blocks. The name multi-level BFT consensus mechanism reflects the fact that the verifier/guardian division provides multi-level security assurance. The validator committee provides the first level of protection-with 10 to 20 validators, the committee can quickly reach a consensus. The Guardian Pool constitutes the second line of defense. With thousands of nodes, it is more difficult for an attacker to compromise, thus providing a higher level of security. We believe that this mechanism has achieved a good balance between transaction throughput, consistency and decentralization, which is the three pillars of the so-called "impossible triangle".

5.2 Aggregate signature broadcast scheme

The naive all-to-all broadcast of checkpoint block hashing can work between guarding nodes, but it will incur secondary communication overhead, so it cannot be expanded to more than 1,000 nodes. On the contrary, we propose an aggregated signature broadcast scheme, which can significantly reduce the complexity of message delivery. Each guardian node continuously combines partial aggregate signatures from all its neighbors, and then broadcasts the aggregate signature. In this way, due to the broadcast protocol, the signature share of each node can reach other nodes at an exponential rate. In addition, signature aggregation keeps the size of node-to-node messages small, thereby further reducing communication overhead.

5.3 Resource-oriented micropayment pool

An off-chain "resource-oriented micropayment pool" specially constructed for video streaming. It allows users to create an off-chain micropayment pool, any other user can use off-chain transactions to exit the pool, and has double payment capabilities. Compared with offline payment channels, it is more flexible. In particular, for video streaming use cases, it allows viewers to pay for video content retrieved from multiple cache nodes without having to conduct on-chain transactions. By replacing on-chain transactions with off-chain payments, the built-in "resource-oriented micro-payment pool" significantly improvesthescalability of the blockchain.

6. TAN distribution network

CDN distribution and edge computing networks focus on the timely distribution of computing large-bandwidth content on the basis of low latency and high performance. A high number of concurrent users means that more peer-to-peer resources are available, so peer nodes can pull data from each other more efficiently. As more peer nodes become available, the overall system capacity increases. In addition, the robustness of the system is improved in a peer-to-peer network, because nodes do not need to rely on a central server to retrieve content. This is especially important in the event of a server failure. In contrast, for CDN-based centralized delivery, high concurrent users put pressure on CDN servers for scalability.

However, the disadvantage of a pure peer-to-peer network is availability. Peers come and go at any time, which makes it difficult to predict the availability of any given peer. There are also uncontrollable differences between nodes, such as upload and download capabilities. On the other hand, the CDN service is more reliable and robust, so it can be used as a reliable "backup" when the stream cannot be obtained from the peer node.

Our goal is to achieve maximum CDN bandwidth reduction without sacrificing quality of service (QoS), which is essential for established CDN distribution platforms such as Netflix, YouTube, Twitch, and Facebook. This means that whenever possible, we want peer nodes to pull streams from each other rather than from the CDN. In order to achieve this goal, it is essential that peer nodes can effectively identify neighboring nodes. If a node can identify multiple peers that are close together, it is possible to find peers that can provide data stream segments more consistently. On the contrary, if the identified peers are "farther away" in terms of network hops, the nodes may not be able to consistently pull streams from the peers and cause a decrease in user experience, such as freezes, frequent rebuffering, and so on.

To solve this problem, TAN has designed and is currently implementing a strategy that combines an ultra-optimized tracker server and user smart client. Essentially, the tracker server provides high-level guidance for the client (such as a list of candidate peers), and the client implements peer filtering algorithms at a finer granularity based on multiple variables to find the best service Adjacent nodes.

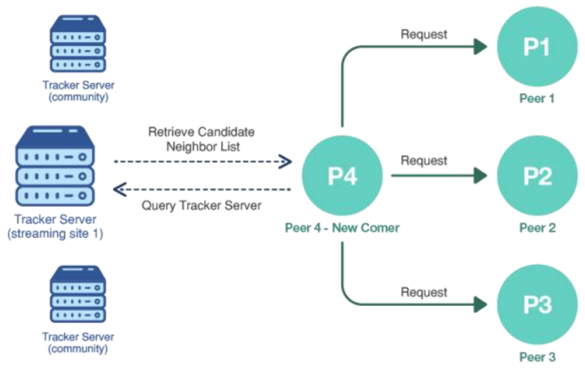
Tracker Server (community)
Tracker Server (streaming site 1)
Tracker Server (community)
Retrieve Candidate Neighbor List
Query Tracker Server
P4
Peer 4 - New Comer
Request
Request
Request
P1
Peer 1
P2
Peer 2
P3
Peer 3

Figure 4. The interaction between the tracker server and the player client

7. Tracking service optimized based on geographic location

In order to provide each client with a list of candidate peer nodes, each time a peer joins the network, the tracker server records geographic location information, including its IP address, latitude/longitude, and many other performance parameters. With this information, the server can organize nodes in the spatial database. TAN's ultra-optimized spatial database is optimized for storing and querying data representing objects defined in geometric space. When a peer node joins the network, the server can perform a spatial query to retrieve a list of nearby candidate peer nodes very quickly and efficiently, as shown in Figure 4. Tracker servers and spatial databases can be maintained through data streams. Use TAN networks and/or sites for content delivery by community peers.

As we mentioned before, peer nodes may leave the network at any time. Therefore, the tracking server also needs to know which nodes are active. For this, the active peer node needs to maintain a socket connection with the server and send a heartbeat signal consistently. If the server does not receive a heartbeat within a certain period of time, it is considered that the peer node has left the network, and the spatial database is updated accordingly.

An important difference is that the "distance" between two peer nodes is measured by the number of router hops between them rather than the geographic distance. Generally, network distance and geographic distance are highly correlated, but not necessarily absolute. For example, two computers may be physically placed side by side, but connected to different ISPs, so there may be many hops between them. Therefore, in addition to geographic information, the tracking server also uses the connectivity between IP addresses collected in the past to analyze and select neighbor candidates. For example, the candidates returned by the spatial query can pass another filter to exclude those candidates that are not connected to the same ISP as the viewer's ISP.

8. Blockchain ledger

TAN ledger is a decentralized ledger designed for industries that require distributed computing and content distribution services. It powers the TAN token ecosystem, incentivizes end users to share their redundant computing power, bandwidth and storage resources, and enables them to more actively interact with cloud computing platforms. In order to achieve these goals, many challenges need to be resolved.

One of the challenges is to support ultra-high transaction throughput. Although many blockchain projects face transaction throughput issues, the expansion of real-time data streams is different and may even be more complicated. Usually, it may only take a few seconds for data flow and calculation time. In order to achieve the finest granularity of token rewards-one micropayment per piece of calculation and data-even 10,000 concurrent user requests may generate thousands of microtransactions per second, which is far more than today Maximum throughput of public chains, such as Bitcoin and Ethereum. For example, a popular live broadcast such as a large-scale e-sports game can attract more than 1 million viewers to watch a live broadcast at the same time. The data forwarding required by the CDN may push the required transaction throughput to millions per second. range.

One side effect of high throughput is rapidly increasing storage consumption. Storing micropayment transactions places high demands on storage. With tens of thousands of transactions added to the ledger every second, the storage space of an ordinary computer may soon be exhausted.

At the same time, a quick consensus is usually required for applications. For bandwidth sharing rewards, users who contribute redundant computing power and bandwidth usually want to confirm the payment before sending the next one.

Finally, as with any blockchain, the security of the ledger is of paramount importance. Security is highly related to the degree of decentralization. In a consensus mechanism based on Proof of Stake (PoS), decentralization means that the rights and interests of consensus participants are evenly distributed. Ideally, the consensus mechanism should allow thousands of independent nodes to participate in the block finalization process, each node has a similar amount of equity and each node has a local copy of the blockchain. In order to destroy such a system, the attacker needs to control a large number of independent nodes, which is difficult to achieve.

In order to achieve these goals, we designed a PoS consensus algorithm based on the Byzantine Fault Tolerance (BFT) protocol. When more than 2/3 of the nodes running the ledger software are honest, the algorithm provides good guarantees, such as consistency (and Name security). However, the traditional BFT algorithm does not allow a high degree of decentralization. Even in normal (non-wrong proposer) situations, they usually produce O (n 2) message delivery complexity, where n is the number of nodes participating in the consensus protocol. When we have thousands of nodes, it takes a long time to reach an agreement. In this article, we propose a novel multi-level BFT consensus mechanism that allows large-scale participation and achieves a throughput of 1000+ TPS even when the transaction confirmation time is as short as a few seconds.

Although this level of transaction throughput is much higher than that of Bitcoin and Ethereum, it is still not enough to handle small payments with the granularity of "pay by byte". In order to further increase throughput, TAN ledger provides local support for off-chain expansion, with a "resource-oriented micropayment pool", which further enlarges the supportable throughput by several orders of magnitude.

We noticed that off-chain payment support not only improves throughput, it also reduces the number of transactions that need to be stored in the blockchain. Most importantly, we have introduced state and block history pruning techniques to further reduce storage space requirements. In addition, our storage system adopts a microservice architecture, which can adapt to different types of machines and storage backends, whether it is a powerful server cluster running in a data center or a commercial desktop.

9． Consensus mechanism

9.1 Multi-level BFT

The TAN ledger is built on a new multi-level BFT consensus mechanism that allows thousands of nodes to participate in the consensus process while still supporting very high transaction throughput (1000+ TPS).

The core idea is to have a small group of nodes, form a committee of validators, and use a PBFT-like process to generate a blockchain as quickly as possible. With a sufficient number of validators (for example, 10 to 20 people), the validator committee can quickly produce blocks and still retain a high degree of difficulty to prevent opponents from destroying the integrity of the blockchain. Therefore, it can be reasonably expected that the verifier will have a very high probability of generating a blockchain without a fork. Then, thousands of consensus participants, called guardians, can finalize the chain generated by the verifier committee. The "end" here refers to persuading every honest guardian that more than 2/3 of all other guardians see the same blockchain.

Since there are many more guardians than validators, it may take longer for the guardians to reach a consensus than the validator committee. In order for the guardian to complete the blockchain at the same speed as the validator committee generates a new block, the guardian node can process the block in a more coarse-grained manner. More specifically, they only need to agree on the hash of the checkpoint block, that is, a block whose height is a multiple of some integer T (for example, T = 100). This "leapfrogging" termination strategy takes advantage of the immutability characteristics of the blockchain data structure-as long as the two guardian nodes agree on the hash of a block, they will have the completeness of the entire blockchain up to that block. The same copy. Only finalizing the checkpoint block provides enough time for thousands of guardians to reach a consensus. Therefore, with this strategy, two independent processes, namely block production and finalization, can be advanced at the same speed.

Under normal circumstances, completing a checkpoint block is similar to the "submit" step of the famous PBFT algorithm, because each guardian has stored the checkpoint block locally. In addition, the checkpoint block has been signed by the validator committee, so it is very likely that all honest guardians have the same checkpoint. Therefore, we only need the agreement of honest guardians to confirm that indeed more than 2/3 of all guardians have the same checkpoint hash.

To implement this protocol, a simple full broadcast of the checkpoint block hash can work, but it generates secondary communication overhead, so it cannot be extended to a large number of guardians. On the contrary, we propose an aggregated signature broadcast scheme, which can significantly reduce the complexity of message delivery. The core idea is quite simple. Each guardian node continuously combines partial aggregated signatures from all its neighbors, and then disseminates aggregated signatures and a compact bitmap that encodes the list of signers. In this way, each node's signature share can reach other nodes at an exponential rate using the broadcast protocol. In O (log n) iterations, if there is no network partition, all honest guardian nodes should have a string that aggregates signatures from all other honest nodes. In addition, signature aggregation keeps the size of node-to-node messages small, thereby further reducing communication overhead.

As mentioned above, the verifier committee consists of a limited set of verifier nodes, usually in the range of ten to twenty. They can be selected through an election process or a random process, and may be rotated to improve security. To be eligible to join the validator committee, a node needs to lock a certain number of shares for a period of time, and if malicious behavior is detected, it can be reduced. The block that the committee reaches a consensus is called the settlement block, and the process of settling the block is called the block settlement process.

The guardian pool is a superset of the validator committee, that is, the validator is also a guardian. The pool contains a large number of nodes, possibly in the range of thousands. After a certain amount of tokens are locked for a period of time, any node in the network can immediately become a guardian. The guardian downloads and checks the blockchain generated by the validator committee, and tries to reach a consensus on the checkpoint through the above-mentioned "leapfrogging" method. By allowing the public to participate, we can greatly improve transaction security. The block that the guardian pool has reached a consensus is called the final block, and the process of finalizing the block is called the block finalizing process.

This multi-level BFT consensus mechanism reflects the fact that the verifier/guardian division provides multi-level security assurance. The validator committee provides the first level of protection-with 10 to 20 validators, the committee can quickly reach a consensus. However, it is sufficiently resistant to attacks-in fact, if each validator node is run by an independent entity, it already provides a security level similar to the DPoS mechanism. Therefore, when a transaction is included in a settled block, it is already considered safe, especially for low equity transactions. The Guardian Pool constitutes the second line of defense. With thousands of nodes, it is more difficult for an attacker to destroy the integrity of the blockchain, thus providing a higher level of security. In case the validator committee is completely controlled by the attacker, the guardian can re-elect validators, and the blockchain can be restarted, starting from the latest block determined by the guardian. When the transaction is included in the final block, it is considered irreversible. We believe that this mechanism strikes a good balance among the three corners of the so-called "impossible triangle" in terms of transaction throughput, consistency, and decentralization.

The multi-level security solution is very suitable for scenarios such as big data distributed computing and big video streaming content distribution. For example, for streaming media platforms, most transactions are small payments (such as payment for peer-to-peer bandwidth, virtual gifts to hosts, etc.), which are usually of low value but require quick confirmation. For this kind of low-equity payment, users only need to wait for the block to settle, which is very fast and only takes a few seconds. For high equity transfers, users can wait longer until the block containing the transaction is completed, which may take a little longer, but it is still within a few minutes.

9.2 System model

Before delving into the details of the block settlement and finalization process, we first list our system design. For the sake of discussion, without loss of generality, below we assume that each node (whether it is a verifier or a guardian) has the same amount of equity. The general case of extending the algorithm to different nodes with different amounts of equity is simple.

Validation committee failure model: There are a total of m validator nodes. Most of the time, at most one-third are Byzantine nodes. They may be completely controlled by the attacker, but this rarely happens. We also assume that there is a direct message channel (such as a direct TCP connection) between any pair of validator nodes.

Guardian pool failure model: There are a total of n guardian nodes. At any time, at most one-third are Byzantine nodes. We do not assume that there is a direct message channel between any two guardians. The messages between them may need to be routed through other nodes, some of which may be Byzantine nodes.

Timing model: We assume a "weak synchronization" model. More specifically, the network can be asynchronous and can even be partitioned within a limited period of time. There is a sufficiently long period of time between asynchronous cycles, in which all message transmissions between two honest nodes arrive within the known time range Δ. As we will discuss later in this article, during the asynchronous period, the ledger just stops generating new blocks. Even if there is a network partition, it will never generate conflicting blocks. In the synchronization phase, block production will naturally resume, and activity can eventually be achieved.

Attacker model: We assume a powerful attacker. They can destroy a large number of target nodes at the same time, but not more than one-third of all guardians. They can manipulate the network on a large scale, and even partition the network within a limited time. However, they are computationally bounded. They cannot forge fake signatures, nor can they reverse cryptographic hashes.

9.3 Block settlement process

Block settlement is the process by which the verifier committee reaches an agreement and generates a blockchain for the guardian pool to finalize. Inspired by recent research work on proof of stake including Tendermint13, Casper FFG14 and Hot-Stuff15, we designed and implemented the block settlement algorithm described below. It adopts a strategy of rotating block proposers, in which validators take turns to propose new blocks. Then, the committee uses a protocol similar to Casper FFG and Hot-Stuff to vote on the blocks to determine their order.

The verifiers rotate in a round-robin manner, playing the role of block proposer, and are responsible for proposing the next block for the verifier committee to vote. To enable cyclic rotation, each proposer maintains a local logical clock called epoch. Suppose there are m validators. In t period, the validator with index (t mod m) is selected as the proposer in that period. We note that the important things are:

1) epoch t should not be stagnant, so as to ensure the activeness of the proposer's rotation;

2) The t of different verifiers should be mostly synchronized, that is, all verifiers have the same t value most of the time, so they can agree on which node should produce the next block.

Below is the agreement between our proposer voting and block proposal.

```
Algorithm 1: cyclic block protocol

t ← 0,  proposer ← 0
voted ← false, received ← false, timeout ← false

loop begin
  proposer ← t mod m
  if ( proposer == self.index ) and (not proposed yet) begin // node
elected as the proposer        propose one block
  end

  voted ← the node has proposed or voted for a block for epoch t
  received ← the node has received m /3 + 1 EpochChange (t
+ 1) messages     timeout ← timeout reached
  if voted or received or timeout begin
    broadcast message
EpochChange (t + 1)     end

  if the node has received 2 m /3 EpochChange (t + 1) messages begin
    t ← t+1 // enters epoch t+1
    voted ← false, received ←
false, timeout ← false     end

  sleep for some time
end
```

Algorithm 1. Cyclic Block Protocol

The protocol defines a message EpochChange which can be seen as a synchronization message passed between verifiers to help them advance to the next period t+1 together. Essentially, if any of the following conditions are met, the validator broadcasts the message EpochChange (t + 1) to all other validators:

1) The node has proposed or voted for the block in period t

2) The node has received m /3 + 1 EpochChange (t + 1) messages from other validators

3) The node has timed out in period t (timeout is set to 4Δ).

On the other hand, the verifier enters t + 1 EpochChange (t +1) message from other nodes when it receives 2m /3.

Here, we show that the agreement meets the above two requirements.

Final step: All honest nodes will eventually enter epoch t + 1. In the worst case, all honest nodes (at least 2m /3 + 1 nodes) have reached the timeout and broadcast the EpochChange (t +1) message. Under the assumption of the timing model, all these messages will be delivered within the time Δ after they are sent. Therefore, each honest node will receive at least 2m /3 EpochChange (t + 1) message and then enter epoch t + 1.

Epoch synchronization: This means that the era of all honest nodes "moves together". To be precise, the time at which any two honest nodes enter the epoch t + 1 differs by at most 2Δ. To prove this, we note that since there are at most f failed nodes, for the first honest node to enter epoch t + 1, at least m / 3 other honest nodes must broadcast the EpochChange (t + 1) message. Then this honest node also broadcasts an EpochChange (t + 1) message according to the protocol. After at most Δ, any honest node should receive at least m /3 + 1 EpochChange (t + 1) message, which triggers them to also broadcast EpochChange (t + 1) message. After Δ, all honest nodes receive the 2m /3 EpochChange (t + 1) message and enter epoch t + 1. Therefore, the first honest node will enter epoch t + 1 after up to 2Δ, and the last honest node will enter the same epoch.

In practice, when the network delay is small enough, all honest nodes should enter epoch t + 1 almost simultaneously. As a result, they can agree on who is the next proposer. We also noticed that for actual implementation, the EpochChange (t + 1) message can be combined with other types of messages (such as block voting) to improve efficiency. In this way, under normal circumstances (no proposer fails), no additional synchronization overhead will be added to the system to change the epoch.

9.4 Validator block consensus

We designed a protocol to try to resolve the proposed block, involving a PBFT-based voting procedure between all validators, similar to Casper FFG and Hot-Stuff. In the TAN ledger, the header of each block contains a hash pointer to its parent block (that is, the previous block in the chain), similar to Bitcoin and Ethereum. If neither block is an ancestor of the other block, the two blocks are in conflict. If there are multiple conflicting block proposals in the same epoch, the honest validator will keep all these proposals until one of them is resolved, and then discard all conflicting blocks.

The block settlement protocol runs one by one. The current proposer sends a block proposal to all validators. The validator reacts by broadcasting a vote on the block. All messages are signed by their sender.

The header of the proposed block may carry a "submission certificate", which consists of at least (2m /3 + 1) signed votes for its parent block. We note that under the assumption that no more than m/3 validators fail, at most one block per height can obtain a certificate of submission. The submission certificate of a block indicates that the block and all its predecessors have been submitted. If its parent block does not get ⩾ 2m /3 + 1 signature votes, the proposed block may not carry a submission certificate.

For validators who are not the current proposer, their job is to vote on the proposed block. Once a validator receives a new block, it will broadcast a signature vote to all validators so that the proposer of the next epoch can collect it to form a submission certificate. If two consecutive blocks A and B both receive the submission certificate, the parent block A and all its predecessors are considered resolved. To ensure security, we require honest nodes to never vote for blocks that conflict with resolved blocks. When there is a fork (due to proposer error or asynchrony), honest nodes should vote for the block on the longest fork.

The figure below illustrates the block settlement process. Suppose the proposer of height 101 has a problem. It proposes two conflicting blocks X 101 and Y 101, which leads to two branches. Assuming that neither block X 101 nor Y 101 has ⩾ 2m /3 + 1 votes, then the headers of X 102 and Y 102 do not contain the submission certificate (indicated by nil in the figure). However, at some point branch X grows faster, and two consecutive blocks X 102 and X 103 both get ⩾ 2m /3 + 1 votes. After that, the upper branch X up to block X 102 is considered resolved. And the lower branch Y can be discarded.

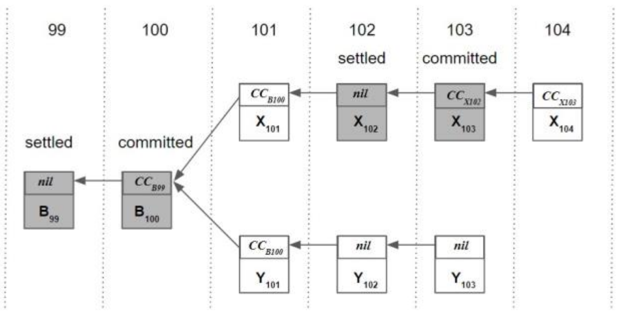

Figure 6. Block confirmation process

The above example also illustrates an advantage of our implementation compared with other PBFT-based protocols (such as Tendermint)-a block that has not received the submission certificate can also be included in the resolved chain, as long as it is a successor block solved. For example, the block X 101 in the example does not get the commit-certificate, but after the block X 102 is settled, it is considered settled. This reduces the waste of computing power and helps increase transaction throughput.

9.5 Performance analysis

Security:

Security means that all honest verifiers agree to the same blockchain. More precisely, if an honest verifier accepts a block A, then any future blocks accepted by other honest verifiers will appear in the blockchain that already contains A. The security argument is similar to Casper FFG and Hot-Stuff, which are omitted here. We just want to point out that security comes from the requirement that honest nodes never vote for blocks that conflict with resolved blocks.

Liveness:

Liveness means that the validator committee is always improving, that is, it is always able to generate new blocks and reach consensus. Here, we show that under our timing model, during the synchronization period, the committee can always achieve the activity goal. First of all, in the "Block Proposal" section, we have proved that the development of the block can always move forward, and all honest verifiers move forward together. In the era when the proposer is an honest verifier, it will propose a new block. For the block settlement process, the liveness depends on an unlimited number of epochs during the synchronization period, two consecutive proposers are honest, and wait long enough to form a submission certificate. We noticed that with the round robin, this situation will certainly happen infinitely frequently, because at least 2/3 of the verifiers are honest.

Transaction throughput:

With 10 to 20 validators, the committee can generate and resolve blockchains fairly quickly. The average block production and settlement time is on the order of seconds, which results in a high throughput of up to 1,000 transactions per second.

9.6 Block confirmation process

In this section, we will discuss the block finalization process in detail. As mentioned above, the guardian only needs to reach a consensus on the hash value of the checkpoint blocks, and the height of these blocks is a multiple of a certain integer T (for example, T=100).

To explain why it is sufficient to complete only checkpoint blocks, we noticed that the transaction execution engine of blockchain software can be regarded as a "deterministic state machine", and transactions can be regarded as a deterministic state transfer function. If two nodes run the same state machine, they will reach the same end state after executing the same sequence of transactions from the same initial state. Please note that even if certain transactions are invalid, as long as the state machine can detect and skip these transactions, this is the case. For example, suppose there is a transaction that attempts to spend more tokens than the source account balance. The state machine can simply skip this transaction after performing a sanity check. In this way, "bad" transactions have no effect on the state.

In the context of blockchain, if all honest nodes have the same copy of the blockchain, it can be ensured that they reach the same end state after processing all blocks in turn. But there is a caveat-the blockchain may contain a lot of data. How can two honest nodes effectively compare whether they have the same blockchain?

Here, the immutability characteristics of blockchain data structures become highly relevant. Since the header of each block contains the hash value of the previous block, as long as two nodes have the same checkpoint block hash value, it is very likely that they should have the same block from creation to checkpoint chain. Of course, each guardian node needs to verify the integrity of the blockchain. In particular, the block hash embedded in each block header is actually the hash of the previous block. We noticed that a node can perform an integrity check on its own and does not need to communicate with other nodes.

Interestingly, the immutability feature also enhances tolerance to network asynchrony and even partitioning. Due to the network partition, the guardian may not be able to reach a consensus on the hash of the checkpoint. However, after the network is restored, they can continue to vote on the next checkpoint. If they can reach an agreement, regardless of whether they reach a consensus on the current checkpoint, all blocks until the next checkpoint are finalized.

In order to provide Byzantine fault tolerance, an honest node needs to ensure that at least two-thirds of the guardians have the same checkpoint block hash. Therefore, before the node can mark the checkpoint as completed, it needs to receive the signature of the checkpoint hash from at least two-thirds of all guardians. This is to ensure safety and is similar to the "submit" step in the well-known PBFT protocol.

Since the guardians only need to vote on the checkpoint hash every T blocks, they have more time to reach a consensus. Therefore, a straightforward implementation of checkpoint termination is to follow the PBFT "submit" step, in which each guardian broadcasts its signature to all other guardians. This requires each node to send, receive, and process O(n) messages, each of which may be several kilobytes long. Even if there are T block times, this method still cannot scale to more than a few hundred guard nodes, unless we choose a large T value, which is undesirable because it increases the block determination delay.

10. Support Turing complete smart contract

The TAN ledger provides a smart contract operating environment that is fully compatible with the Ethereum virtual machine. It provides comprehensive support for Turing's complete smart contract. The Solidity-based Ethereum smart contract can be ported to the TAN ledger effortlessly. Solidity has developed a large developer community, and the prospect of allowing a proven talent pool to also contribute to TAN without having to reinvent the wheel is the main consideration for achieving compatibility with the Ethereum virtual machine.

Smart contracts provide rich user experience and new attribution models for various platforms and apps based on TAN books. Smart contracts can be written to support smoother payment and consumption models, such as pay-as-you-go or pay-per-use models. Unlike traditional annual or monthly subscriptions, user consumption can be priced at bite-size granularity, so that users only pay for what they use. This is a viable way to allow low-cost, short-term content to be processed in an economically wise manner, thereby bringing benefits to the cloud platform and users. The TAN ledger’s ability to track micropayments and video clips enables the execution of such smart contracts.

Smart contracts can also be designed to benefit content providers as a way to distribute royalties fairly and transparently. The traditional royalty settlement process has all the complexity and ambiguity, which can be adapted through clear smart contract terms agreed upon by the creator and the issuer, and provided to users who consume content. Using smart contracts on the TAN ledger to achieve fully digital ownership of items, innovative payment and consumption models, and transparent royalty distribution, provide an additional layer of social and economic interaction, and supplement the core functions of video/content delivery.

11. Support off-chain micropayments

As discussed in the introduction, supporting high transaction throughput is necessary for blockchains that focus on computing power and traffic. We build support for off-chain payments directly into the ledger to facilitate a large number of transactions.

Resource-oriented micropayment pool

We designed and implemented an off-chain "resource-oriented micropayment pool" built specifically for computing power and traffic. It allows users to create an off-chain micropayment pool, any other user can use off-chain transactions to exit the pool, and has double payment capabilities. Compared with offline payment channels, it is more flexible. In particular, for the CDN use case, it allows viewers to pay for data content retrieved from multiple cache nodes without the need for on-chain transactions. By replacing on-chain transactions with off-chain payments, the built-in "resource-oriented micro-payment pool" significantly improves the scalability of the blockchain.

The following diagram describes how the resource-oriented micropayment pool works in the application:

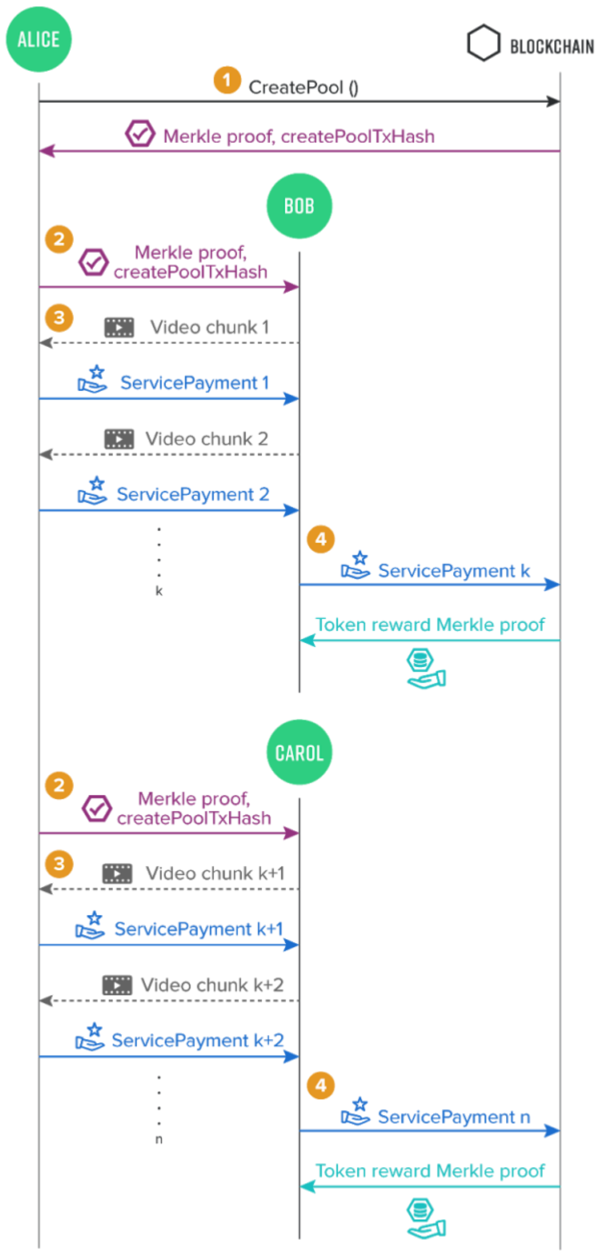

Figure 7. Resource-oriented micropayment pool showing viewer Alice making off-chain transactions to cachers Bob and Carol for data blocks

Step 1. Create a micropayment pool: As the first step, Alice publishes an on-chain transaction to create a micropayment pool with time lock and collateral reduction.Create Pool (resourceId, deposit, collateral, duration)

In order to create a pool, Alice needs to specify a "resource ID" resourceId that uniquely represents the digital content she wants to retrieve. It may refer to video pieces or live streaming.

The deposit amount must be at least the total value of the resources to be retrieved. For example, if the resource is a video file worth 10 tokens, the deposit must be at least 10 tokens.

Collateral is needed to prevent Alice from double spending. If the validator of the blockchain detects Alice's double-spending attempt, the collateral will be cut. Later in the article, we will explain that if mortgage> deposit, the net return of double spending is always negative, so any rational user has no motivation for double spending.

The duration is a time lock similar to standard payment channels. Any withdrawal from the payment pool must be made before the time lock expires.

The blockchain submits the Merkle proof of the Create Pool transaction to the blockchain and create Pool TxHash, that is, the transaction hash of the Create Pool transaction is returned to Alice.

● Step 2. Initialize the two-sided protocol handshake: Whenever Alice wants to retrieve the specified resource from the peer (Bob, Carol, or David, etc.). She sends the Merkle proof of the Create Pool transaction on the chain to the peer. The receiver verifies the Merkle proof to ensure that there are sufficient deposits and collateral in the pool for the requested resources, and both parties can proceed to the next step.

● Step 3. Off-chain micropayments: Alice signs Service Payment transactions and sends them to peers outside the chain in exchange for some designated resources (such as a video file, a live stream segment, etc.). The Service Payment transaction contains the following data:

Target Address, transfer Amount, create Pool-TxHash, target Settlement Sequence, Sign (SK A, targetAddress || transferAmount || createPoolTxHash || targetSettlementSequence)

Target Address is the address of the peer from which Alice retrieves the resource, and transfer-Amount is the token payment amount that Alice intends to send. The target Settlement Sequence is to prevent replay attacks. It is similar to the "nonce" parameter in Ethereum transactions. If the target publishes a Service Payment transaction to the blockchain (see next step), its target Settlement Sequence needs to be incremented by 1.

The receiver needs to verify the off-chain transaction and signature. After verification, the peer can send Alice the resources specified by the Create Pool transaction.

In addition, we noticed that off-chain Service-Payment transactions are sent directly between two peers. Therefore, there is no scalability bottleneck in this step.

● Step 4. On-chain settlement: Any peer (ie Bob, Carol, David, etc.) that receives a ServicePayment transaction from Alice can publish the signed transaction to the blockchain at any time before the time lock expires to withdraw tokens . We refer to the released Service Payment transaction as an "on-chain settlement" transaction.

Please note that the receiver needs to pay gas fees for on-chain settlement transactions. In order to pay less transaction fees, they will need to publish on-chain settlement only when necessary, which is conducive to the scalability of the network.

It is not difficult to see that when Alice switches from one node to another to retrieve resources, no on-chain transactions are required. In the context of data streaming, this means that viewers can switch to any cache node at any time without having to perform on-chain transactions that might prevent or delay data transmission. As shown in the figure, if Bob leaves, Alice can switch to Carol after receiving the k chunks sent by Bob,and continue to receive data fragments without the need for on-chain transactions.

In addition, the total amount of tokens required to create a micropayment pool is (mortgage + deposit), no matter how many peers Alice retrieves resources from, it can be as low as twice the value of the requested resources. Using computational complexity language, compared with the one-way payment channel method, the number of reserved tokens is reduced from O (n) to O (1), where n is the number of peers from which Alice retrieves resources.

12. Ledger storage system

Using public ledgers to facilitate micropayments for streaming media is very challenging, not only because of the high transaction throughput, but also the extremely high requirements for storage space management. In order to achieve the granularity of "pay by byte", each viewer can send a payment every few seconds. Therefore, even with only 10,000 concurrent users, it can generate thousands of transactions per second. Even if the off-chain payment pool has greatly reduced the number of transactions on the chain, the block and status data may still expand rapidly

To this end, we designed a storage system to solve this problem, and it can be adapted to different types of machines, whether it is a powerful server cluster running in a data center or a commercial desktop computer.

Storage microservice architecture

In order to take advantage of the processing and storage capabilities of server clusters, the key design decision is to adopt the popular microservice architecture common to modern Web service backends, where different modules of the ledger can be configured to run on different machines. In particular, the consensus module and the storage module can be separated. The underlying consensus module can run on multiple machines, using the MapReduce framework to process transactions in parallel.

The TAN ledger stores both transaction blocks and account state history, similar to Ethereum. The bottom layer of the storage module is key-value storage. TAN ledger implements multiple database interfaces, from stand-alone LevelDB to cloud-based NoSQL databases, such as MongoDB, which can store almost unlimited amounts of data. Therefore, the ledger can be run on a single computer, or it can be configured to run on a server cluster.

History pruning mechanism

Although the microservice architecture is very suitable for powerful server clusters, we still face storage space limitations when running ledger on low-end home PCs. We have designed a variety of technologies to reduce storage consumption.

Similar to Ethereum, the TAN ledger stores the entire state of each block, and the root of the state tree is stored at the head of the corresponding block. In order to reduce the space consumed by state history, TAN ledger supports a state history pruning mechanism, which uses a technique called reference counting, as shown in the following figure.

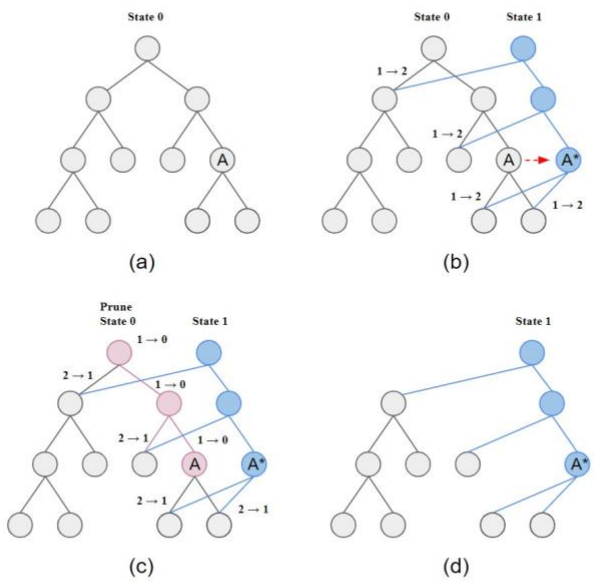

Figure 9. Using reference counting for state history pruning

The state of the ledger (ie, the token balance of each account, etc.) is stored using Merkle-Patricia trie. Figure 9(a) depicts the initial state tree, the root of which is represented by state 0. Each node in the tree has an attribute called "reference count", which is equal to the number of parent nodes of the node.

In the initial state tree, each node has only one parent node, so the reference count is set to 1.

In Figure 9(b), after applying the transaction in the new settlement block, account A is updated to A*. Therefore, a new Merkel state root state 1 is created, and a Merkel branch connecting the new root state 1 and A* (blue nodes and edges) is created. As new nodes were added, we updated the reference counts of the direct children of these new nodes from 1 to 2.

At some point, we decided to delete state 0 to save some storage space. This is done by recursively deleting nodes with a reference count of zero starting from the root state 0, until no nodes can be deleted. Whenever a node is deleted, the reference counts of all its child nodes are decremented by one. Figure 9(c) illustrates the process, and Figure 9(d) shows the result of the pruning. In order to achieve the maximum level of state storage compression, once a block is finalized by the guardian pool, we can delete all previous history records of the block. The ledger can also be configured to retain a limited state history, for example, a state tree of the last 1000 blocks, depending on the available storage space.

It can be seen that using reference counting technology, the time complexity of pruning a state tree is O (k log N), where k is the number of accounts updated by transactions in a block, and N is the total number of accounts. Usually, k is several hundred In the range of one thousand. Therefore, pruning the state tree should be very effective and not take much time.

Managing the space consumed by transaction blocks is even simpler. After a block is completed, we can simply delete all previous blocks, or keep a limited history similar to a state tree.

Using these technologies, ordinary PCs and laptops are sufficient to run the guard node.

State synchronization

One of the pain points of using the previous generation of blockchain is the state synchronization time. After starting a new node, it usually needs to download the complete block history all the way from the genesis block. This may take several days to complete and has become a barrier to user adoption.

The state and block history stored in the full node can help significantly reduce synchronization time. After the new node is launched, the first step is to download all validator and guardian join/leave transactions and block headers containing these special transactions until the latest final block. With these special transactions and headers containing verifier and guardian signatures, the new node can derive the current verifier committee and guardian pool. Since there are relatively few changes to the verifier and guardian set, the amount of data that needs to be downloaded and verified in this step should be minimal.

In the second step, the new node downloads the state tree corresponding to the latest determined block. And it needs to be confirmed that the root hash value of the tree is equal to the state hash value stored in the latest final block. Finally, the new node verifies the integrity of the state tree (for example, the validity of the Merkle branch). If all checks are passed, the new node can start monitoring new blocks and begin participating in the consensus process.

13. Dual currency system and token mechanism

In order to protect the network, install proper governance and manage the use of the network, the TAN blockchain will use a dual currency system. TAN tokens will be used to mortgage, protect and manage the TAN network, and personal operations (data fragment transactions, smart contract operations, etc.) will be paid using TANGam tokens.

There are two reasons for introducing two agents:

First, this allows separation of the utility and purpose of each agent. TAN agents are strictly used to stake out and protect the network, while TANGam tokens are used to power network operations based on basic service businesses. This is necessary because it will essentially reduce circulating supply, but video clip transactions and smart contracts will require a highly liquid token that can facilitate millions of daily transactions.

Secondly, two tokens are needed to solve the consensus problem that may arise when the same token is used for staking and operation. Since the tokens used for operation must be liquid, it is easier for malicious actors to accumulate a large number of frequently traded tokens on the open market. If the same tokens are also used for stocking, they may threaten the security of the TAN network. By separating the two functions (collateralization and operation) into different tokens, this risk is greatly reduced.

TAN token supply and mechanism

As an ERC20 token, the supply of TAN tokens is currently fixed at 1 billion. When the mainnet is launched, each holder of ERC20 TAN tokens will receive native TAN tokens on the new blockchain at a ratio of 1:1. The supply of native TAN on the new blockchain will also be permanently fixed at 1 billion, which means that new TAN tokens will never be created.

The main reason for fixing the supply of TAN tokens is that the cost of any malicious actors who want to obtain enough tokens to threaten the network will be very high, which will discourage them. Since new TAN tokens will never be created, the only way to get more tokens is to buy existing tokens. Over time, accumulating a controlled amount of TAN tokens will become more expensive.

14． TANGam token supply and mechanism

TANGam is the operating token of the TAN blockchain, used as the "gas" for microtransactions and smart contract operations for payment data fragments. TANGam tokens are also built on the TAN blockchain, and 5,000,000,000 TanGam will be generated when the mainnet is launched. The initial supply of TANGam will be distributed to all TAN token holders at the time of token exchange, providing enough TANGam for the network to operate effectively.

During the token exchange, each TAN token holder will also receive 5 TANGam tokens for each TAN token they hold. Initially, the number of TANGam tokens will not increase until the multi-level BFT consensus mechanism is launched and the guardian pool is formed. After that, both validator and guardian nodes will be required to stake TAN tokens to perform their respective functions. Both validators and guardians will earn TANGam in proportion to the number of TAN tokens they pledge, and the total reward is equal to the target increase in the supply of TANGam. The target growth of TANGam's supply was initially set at 5% per year. The rate can be dynamically adjusted according to the video platform's requirements for Gamma. In other words, the supply of TANGam will increase by 5% within a year. If you run a guardian node and pledge TAN tokens, your share of these new TANGam tokens will be equal to the total amount of TAN tokens you pledge Percentage of pledged TAN tokens.

To help maintain an appropriate amount of TANGam in circulation, all TANGam used as Gas for deployment or interaction with smart contracts will be permanently destroyed. By associating both TANGam generation and destruction with network usage/adoption, the number of TANGam tokens will maintain a healthy balance relative to demand.

14.1 Verifier and guardian node

The validator set will initially consist of nodes operated by TAN Labs, followed by other validator nodes operated by major strategic partners. Eventually, guard nodes that implement high standards (node availability, hardware and bandwidth requirements, etc.) and hold a sufficient number of TAN tokens may be eligible to participate as rotating verification nodes. Our ultimate goal is to establish a verifier set composed of TAN laboratories, video platform partners, and community members, in which no entity or group has sufficient control over the network to carry out malicious acts. If any verifier engages in malicious behavior, the guardian pool should be sufficiently diverse as a second line of defense to prevent malicious behavior and remove malicious verifiers. Malicious actors who take action to harm the network will also reduce (confiscate) their pledged TAN.

We expect that the guardian node function will be launched in a major upgrade after the mainnet is launched. An independent client will be released, allowing users to operate guardian nodes and pledge their TAN tokens. According to the current construction, the protocol can support up to 1,000 guardian nodes without sacrificing transaction throughput. In order to obtain the best set of guardian nodes, we hope to set a range of approximately 100,000-1,000,000 TAN tokens, allowing each guardian node to bet. These numbers may be adjusted based on further testing and community feedback between now and the launch of the mainnet.

15. Development Plan

In this white paper, we introduced the TAN protocol, which is a new blockchain and token, as an incentive mechanism for decentralized content distribution and distributed computing networks. The TAN network encourages users to share their computing and bandwidth resources, and solves many technical and business challenges.

In addition to the initial release of the native TAN network, we classify many other technical aspects of the protocol and network as future work:

- Anti-piracy: This network plans to expand anti-piracy copyright protection technology. Because the token can be used for data streaming and caching certain content, the token acts as a "copyright restriction" condition within the network, because the content can be marked for payment or copyright protection.

- Universal service platform: The TAN protocol will gradually be extended to handle more types of services (such as cloud games, cloud rendering, etc.) to allow end users to obtain cheaper, high-bandwidth and high-computing services.