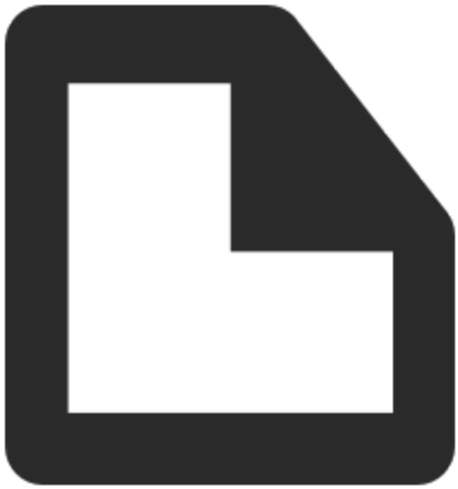


Lake Network Whitepaper

v1.0

https://lakenetwork.io
contact@lakenetwork.io
https://t.me/lakenetwork

Current Market

The need for storage and bandwidth is growing rapidly and will only increase as internet usage grows and more 4K, 8K, and VR content is consumed.

The data storage market is projected to reach $144.7 billion USD by 2022[1] and the cloud storage market is expected to grow to $88.91 billion USD by 2022[2]. This provides a huge opportunity for decentralized blockchain applications to participate in a marketplace and infrastructure solution, where consumers can participate and trade in a fair manner, either for their consumption or to provide services to their users.

Problems currently faced by end consumers:

1. **Security:** Service providers have the ability to read, modify, or even share a user's data, perhaps without even having to disclose doing so. Additionally, consumers' private information is frequently hacked by malicious actors.
2. **Privacy:** In many jurisdictions, consumer data protection laws are negligible or non-existent, resulting in organizations neglecting their user's privacy, or not reporting it in cases of hacks or exploitation. Even if they enforce the most stringent of security policies, most of the cloud storage providers do not encrypt users' data, hence it can be read or modified by a 3rd party once they gain access to it.
3. **Censorship:** Platforms and governments censor what type of content can be served, shared, and consumed, hindering free speech, open exchange of ideas and information, and ultimately ideologies that the internet was built on.

Problems currently faced by enterprises, video creators, and VOD websites:

1. **Security:** It is a challenge to secure data and effectively store and serve end users. There is a need for native encryption on platforms used by enterprises.
2. **Bandwidth:** CDN and bandwidth costs become disproportionately expensive when more users access the content served, increasing infrastructure costs and complexity.
3. **Monopoly:** Top video and storage platforms have a monopoly on the user base and infrastructure they own. Publishing content on their platform leads to the need to abide by their policies and pricing, which may not always be fair, reliable, or in the best interests of the creator/user.
4. **Complexity in scaling:** Self-hosted services are hard to scale and manage. Consequently, most businesses cannot gain access to affordable, secure, reliable, and a fair infrastructure which can compete with existing players (who already have favourable terms). This has led to many digital media companies taking longer to achieve growth, spending more than their competitors, and retreating from their "pivot to video" strategies that were started in 2017. The financial returns do not justify the large investments required in infrastructure.

A Business case for an opportunity in the existing Centralized Storage Market:

1. Akamai, a leading provider of CDN servers and distribution infrastructure: Generated $2.5 Billion revenue in 2017[5], with $2.1 Billion in operating expenses. The majority of the expenses have gone towards infrastructure. There is a potential for data centres and end users to receive a portion of this on a decentralized market.
2. Amazon Web Services (AWS) is moving towards a model where compute and storage are becoming cheaper, a lot faster than providing access to stored data and networking. From our estimates, 2/3rd of AWS's 2017[4] revenue is from networking, bandwidth, and storage, totalling $11.6 Billion.
3. Personal Cloud storage providers, Box and Dropbox posted revenue of $500 Million and $1.3 Billion in 2018

A Business case for an opportunity in the existing Bandwidth and Distribution Market:

1. In 2018, out of the 97,547 PB/month of internet traffic, video stood at 75,109 PB/month with an estimated growth of 31% CAGR to 159,161 PM/month by 2021. CDN and file sharing traffic stood at 75,000 PB/month and 6,800 PB/Month respectively.[3]
2. In 2018, media companies created and processed nearly three times as much video content than in Q1 2017.
3. 1.87B users will view videos on mobile in 2018, which is 78% of all video viewers
4. Long-form video (20+ minutes) exceeded 50% of time watched on every device and screen.

These statistics clearly indicate we are moving towards an explosion of larger files and content being created, stored, distributed, and consumed.

Why Lake

Lake is built on top of Tahoe LAFS (Least Authority File System) and provides a secure, decentralized, distributed, reliable, and private storage and retrieval infrastructure.

Lake offers the end users the ability to share their unused storage and bandwidth and in return receive tokens. On the other side, end users, developers, and enterprises can configure the network to their needs in order to store and retrieve their files.

Incentive Computations, Payments, Proof of Uptime, and Proof of retrievability mechanisms are built to make the platform fair and provide a reliable infrastructure.

Lake is:

1. **Secure**: It offers "provider independent security" by encrypting and encoding files before uploading them. The service provider never has the ability to read or modify users' data.
2. **Decentralized and Distributed**: Various nodes handle the storage, retrieval, encryption and decision making by design. Files are split into shares and stored across a distributed storage layer.
3. **Reliable**: A complete file can be recovered even if a part of it is lost, based on erasure encoding.
4. **Privacy First**: Clients and Storage Servers are only identified by Node IDs on the network. Files uploaded are referred to by their "Capabilities". These do not require user information and using it in conjunction with a VPN service offers network level privacy.
5. **Fast**:
    a. Files can be downloaded simultaneously from multiple storage servers allowing for swarming downloads, similar to downloading from seeds in P2P.
    b. Computation of Incentives, Payments, Proof of Uptime, and Proof of retrievability through State Channel implementations, moving related microtransactions off-chain for high throughput and low gas consumptions needing only on-chain persistence of finalized state and verifiability.

Lake Network

1) Tahoe LAFS

Tahoe LAFS or Tahoe Least Authority File System is a secure, decentralized, fault-tolerant and distributed cloud storage system. It forms the backbone of the Lake Network.
The primary reasons for the Lake Network to use Tahoe LAFS are:

a) Secure

Tahoe LAFS features "provider independent security" whereby "The service provider never has the ability to read or modify your data in the first place: never.". What this means is that files are AES encrypted on the client and encoded before being sent to the storage servers. Storage servers can never know what they're storing.

b) Decentralized

Decision making on Tahoe LAFS is decentralized by design. A request to GET or PUT files can be made to any working node on the network and responses are homogenous.

c) Distributed

On Tahoe LAFS, files, after encryption on the client, are optionally encoded on the client itself or on a Helper Node. The resulting "shares" are then distributed to multiple storage servers depending on the scheme configured for the specific network.

d) Reliable

File(s) undergo erasure encoding based on the scheme configured for the network such as a "3 of 10" scheme which ensures that the whole file can be retrieved even if only 3 of the total number of storage servers housing those shares(10) are up.

e) Private

Clients and Storage Servers are only identified by Node IDs on the network. Files uploaded are referred to by their "Capabilities". Neither of these requires any divulgence of user information and when used in conjunction with a VPN service, even network level privacy can be achieved. It is purely upto the service that runs on top of a Tahoe network instance to require user information or perform any sort of logging(which could be valid use cases).

2) EOSIO

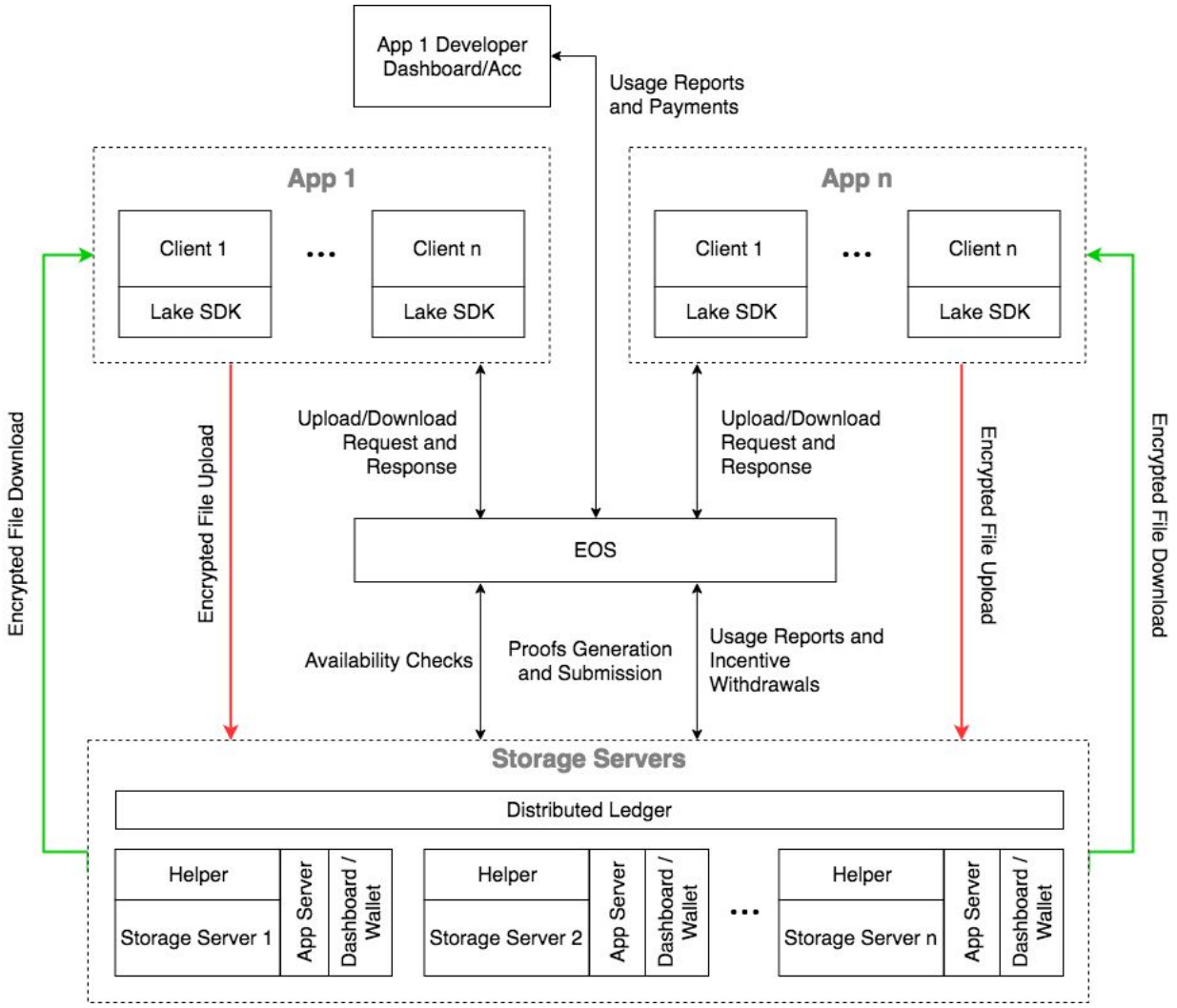
The EOSIO blockchain infrastructure serves as almost the middleware within the Lake Network managing P2P interactions between the client and the storage servers. The primary reasons for the Lake Network to use the EOSIO blockchain are:

a) **Decentralization**
EOSIO runs a dPOS consensus with 21 validators from a larger candidate set who are democratically voted for by token holders. A large and disparate validator tends to ensure much lower opportunities for attacks on the system while affording for the processing of transactions to be well decentralized.

b) **Low Latency**
EOSIO is on mainnet and it's architecture, system design and consensus mechanism affords for a very high number of transactions being executed per second(millions in theory and thousands as of today). This allows for massive scalability.

c) **Abstraction**
End users not needing to pay for transactions allows for the Lake Network to neatly abstract away the monetization layer from app users, expose it only to the developer of the app who in-turn is free to impose any monetization strategy he/she would like(tokenized DApp, ad-based app…)

3) Architecture



a) **Apps**

i) **Client** - Applications on various platforms built by developers/enterprises interested in leveraging the Lake Network for its file storage and retrieval services.

ii) **Lake SDK** - Light-weight SDKs, bundled into client applications, for Web, Desktop, Android and iOS that will be responsible for initiating Uploads/Downloads and performing file encryption.

b) **EOS Blockchain**

The EOS Blockchain holds all the smart contracts that perform the following functions:

i) **Request Management -** Manage incoming requests for upload/download of files from clients.

ii) **Authentication -** Authenticate if the requests are coming in from a valid client and resolve the storage configuration based on the requester's App Id.
iii) **Running Proofs -** Verify storage server's uptime and if it is indeed storing a share for a specific file.
iv) **Data Storage -** Store and allow for querying of all data related to running proofs, usage, payments and incentives.
v) **Resource Availability -** Determine which Helper is currently available for processing client's upload/download request.
vi) **Processing Payments and Withdrawals -** Computing usage of resources and determining how much is needed to be paid by the developer and how much incentive a storage server can withdraw.

c) **Storage Servers**
i) **Helper -** Used to perform erasure coding on the encrypted files it receives and move shares to appropriate storage servers as determined by the uploading app's config. Also processes requests to download files.
ii) **Servers** - Perform actual storage of shares received from the Helper.
iii) **App Server** - Application used to send heartbeat as proof of uptime, generate and supply Merkle proofs of storage and also initiate repairs in case of lost shares.
iv) **Dashboard/Wallet -** A web-based application that a storage server admin can employ to monitor/verify resource usage and initiate incentive withdrawal directly into their private wallet.

4) Features

a) **Security & Privacy**
Owing to Tahoe LAFSs architecture, the Lake Network is secure by design. All files are AES encrypted, before being encoded and distributed to various Storage Servers which cannot snoop onto any of the files stored even if they wanted to. Files are referenced via specialised identifiers and only those with access to them can really read/write them.
Lake does not require any applications built on top of it to disclose any of their user's personal information or public identifiers. Developers are free to set any level of privacy they deem fit or as demanded by their use case.

b) **Erasure Coding**
The Lake Network's erasure coding follows the k of N scheme where, of all the Storage Servers on the network, depending on the Application's requirements, files uploaded are encoded into N shares and distributed across N Storage Servers in a manner such that the entire file can be retrieved from any k active Storage Servers

in the event that (N-k) servers are unavailable. This helps ensure predictable availability and flexible redundancy of data.

c) Swarming Download

Since files encoded on the network are distributed across multiple Storage Servers, they can be downloaded from each of the nodes simultaneously allowing for swarming download of files much akin to downloading a file from multiple seeds on BitTorrent.

d) Distributed and Decentralized

All files uploaded are encoded and distributed amongst multiple Storage Servers depending on the Application's use-case and Developer's storage configuration. Erasure encoding on the files before distribution ensures that, depending on the configuration, a file can be retrieved despite some of the encoded shares being unavailable or some Storage Servers being offline. Also, computation on the Lake Network in decentralized with multiple Service Nodes running some or all of the services required for user authentication, payment management, incentive computation, reporting etc…

e) Integrity

The Lake Network maintains system integrity while being fully decentralized to provide some necessary guarantees to developers. It does so by running proofs and utilizing game mechanics to ensure that resource providers do not go offline, that files(and shares) are indeed stored on the servers they are meant to be, files are retrievable on-demand and that the usage of resources is accurately tracked and that payments and incentives are computed with complete transparency.

f) Flexible Pricing

The Lake Network will feature flexible payment options for developers to decide what option works for them depending on the use case of their application.

Free Tier

- Free Tier with daily and total storage/bandwidth limits.
- Limited Redundancy, Throughput and Availability.
- For test applications and prototypes.

Managed Services Tier

- Paid Plans with varying limits on storage/bandwidth limits.
- Network managed services with high Redundancy, Throughput and Availability.
- For Production and Enterprise applications.

Token Economics

1) Structure

The Lake Network will employ a dual token structure namely Lake Network Credits(LNC) and Lake Network Tokens(LNT).

a) Lake Network Credits

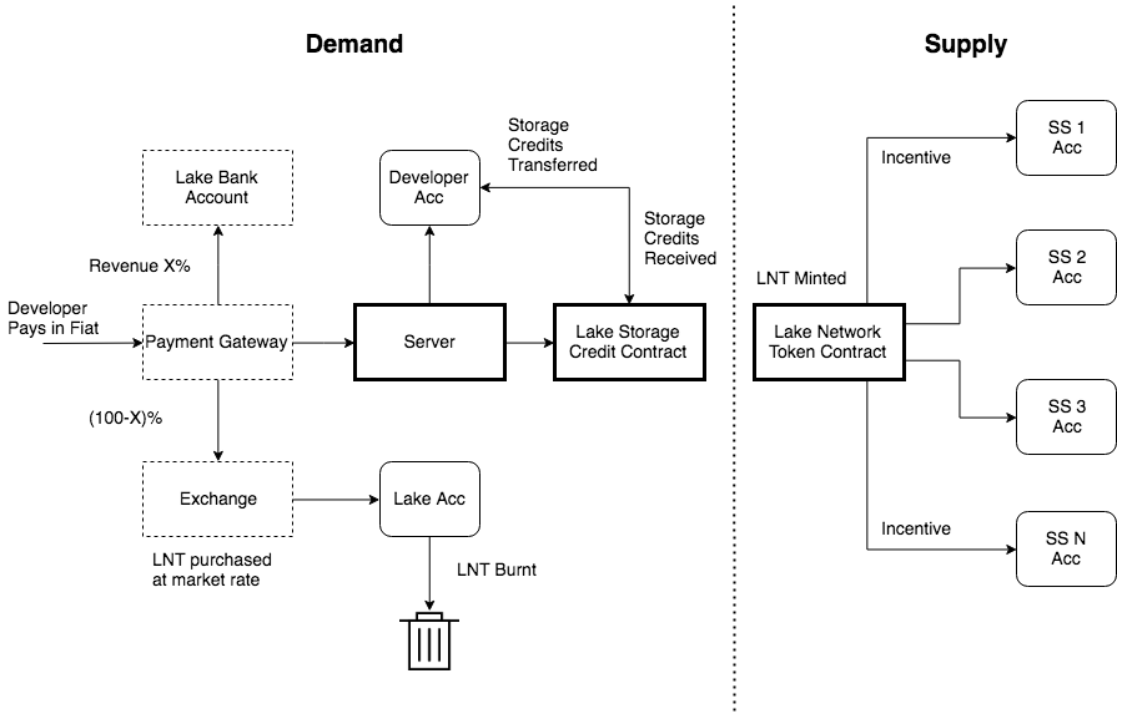
LNCs are an EOS based token that have no monetary value outside of the Network. They are tokens that developers purchase in exchange for their apps consuming resources(storage and bandwidth) on the Lake Network. They are purchased using Fiat currency (USD) and have a fixed price. This frees up developers from the hassle of owning/storing crypto assets, allowing them to conform to their native regulations and legalities. This also enables them to easily predict expense in the absence of the inherent volatility of a cryptocurrency.
Additionally, LNCs are non-transferable, limiting any possibility of the token being traded and accruing monetary value.

b) Lake Network Tokens

LNTs are also a EOS based token and will have monetary value, be traded on cryptocurrency exchanges and be subject to market speculation. They are tokens that are used to incentivise Storage Servers for their participation on the Network. Additionally, they are also what Storage Servers will stake to participate in the network and vote on the addition of new Storage Servers onto the network as described in the section on Storage Server governance below.

2) Model

The Lake Network will employ a Burn-Mint Equilibrium token model where LNTs are Burnt on the demand side and Minted on the supply side. This model is fundamental to the Lake Network having more stable token economics and the Network's value enhancing primarily via usage of the platform itself.



On the Demand side:

- A Developer would use their credit card to purchase LNC.
- A portion of that payment would be transferred into the Lake Network's bank account, to be accounted as the Platform's revenue.
- The remainder of the payment is used to purchase LNT, at market value, on one of the public cryptocurrency exchanges that it would be listed on.
- The LNT purchased is transferred into one of the platform's accounts and subsequently burnt with publicly available transactions as proof.
- LNCs purchased by the developer are moved into an account the platform creates on behalf of the developer and consumed in lieu of resource usage by the developer's app users.
- The LNT burnt would be representative of the Network's total resource usage.

On the Supply side:

- At regular epochs, LNTs are minted and transferred over to participating Storage Servers as the incentive.
- The amount of LNT minted at each epoch would be a function of the total network capacity i.e. the total amount of resources(storage + bandwidth) provided by all currently participating Storage Servers.
- The LNT minted would be representative of the Network's total resource capacity.

Without any direct correlation between the amount of LNT burnt and the amount minted, an equilibrium between the two is actually achieved owing to the price pressures that each activity puts on the other:

- If a greater amount of LNT is minted than is burnt, which would be indicative of the Network having capacity > usage, LNT supply would be inflationary which would cause a downward price pressure on the token(in the absence of market speculation). This would then mean, on the Demand side, more tokens could be bought for the same amount of money and consequently, a greater number of tokens would be burnt. This would, in turn, cause the supply to deflate and put upward price pressure.
- If greater amount of LNT is burnt than is minted, which would be indicative of the Network having usage > capacity, LNT supply would be deflationary which would cause an upward price pressure on the token(in the absence of market speculation) which would of course mean fewer tokens could be purchased/burnt for the same amount, resisting and reversing the upward price push.

3) TCR and Staking

The Lake Network will implement a custom Token Curated Registry to list and rank Storage Servers.

Listing

- The existing set of Storage Servers will decide when to open up applications for new Storage Servers to join the network.
- A Storage Server will require to stake LNT for the opportunity to participate in the network, as part of the application process.
- Existing Storage Servers will then vote on individual applications by staking LNT against each vote.
- If voted in, an applicant Storage Server gets to keep their stake. The ones that voted against their inclusion, lose their voted stake, which is distributed among all the Storage Servers that voted for.
- If rejected, the applicant Storage Server loses a portion of their stake which, along with the vote stakes of the Storage Servers that voted for them, is distributed among the Storage Servers that voted against their inclusion.

Ranking

- Storage Servers that get listed on the Lake Network will now have a stake in the system. The stake amount, in addition to past performance, current resource availability and tenure of the Storage Server are what determine their score on the network.
- This score is used to rank Storage Servers and upload/download requests are allocated to them, in this order.
- The Lake Network, additionally, runs some proofs to aid in arriving at the score but also help to determine if any of the Storage Servers are non-performant or acting maliciously which results in a slashing of a portion or all of the stake of the Storage Server in question.

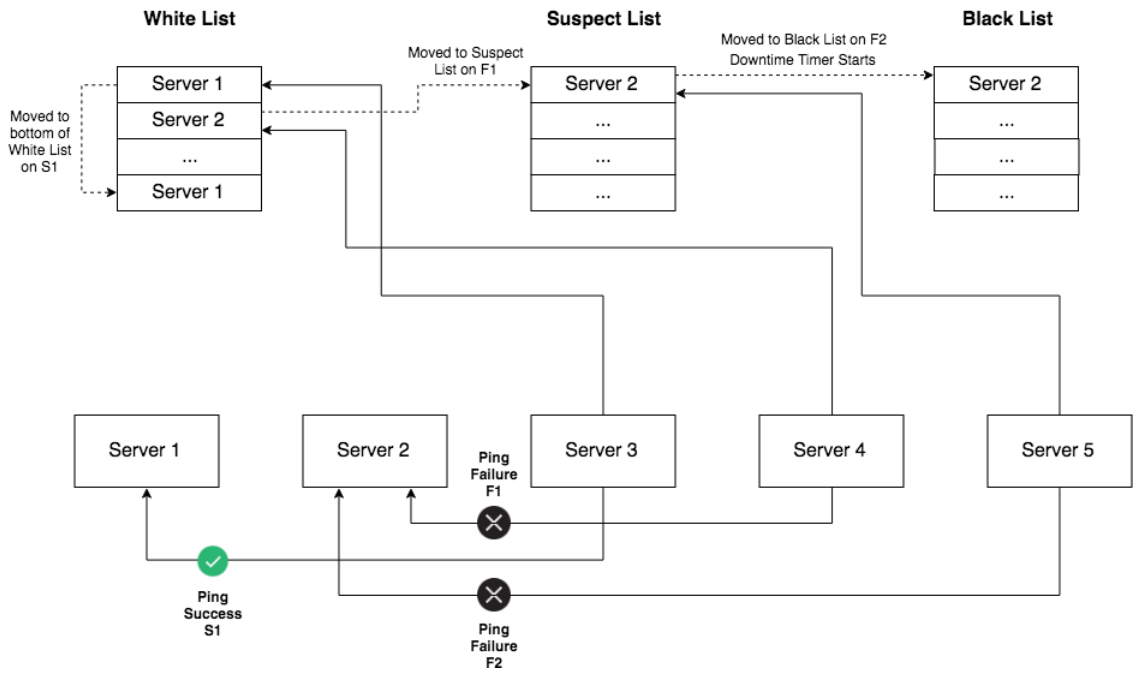
Proofs

1) Proof of Uptime

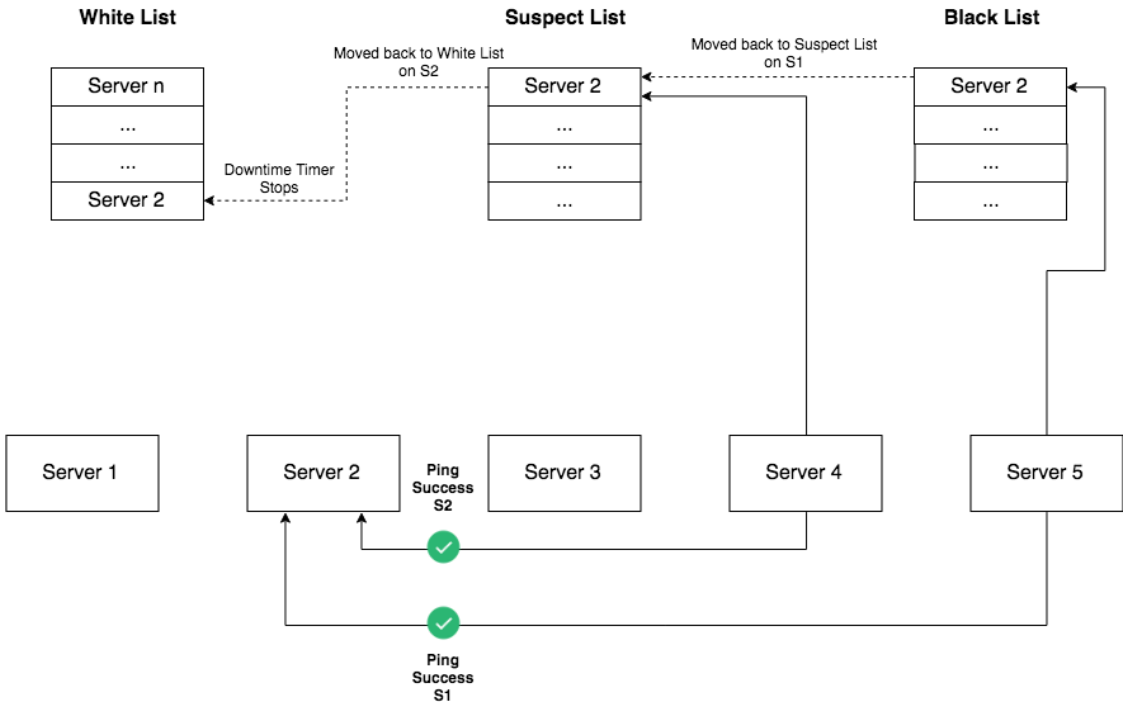
A check to ascertain a Storage Server's uptime on a daily basis.
All Storage Server's have access to 3 distributed lists of Storage Servers:

a) **White List -** List of all Servers currently thought to be alive.
b) **Suspect List -** List of all Servers thought to be either going down or coming back up.
c) **Black List -** List of all Servers confirmed to be down.

In a pseudorandom fashion, a Storage Server accesses one of the lists and selects the Storage Server on the top of the list to check it's liveness. In case it accesses the White List, it can report back if the Server it is checking is either up or down. If up, the said Server's score is updated and it's entry in the White List is moved from the front to the back. If the Server is found to be down, it's entry is moved into the Suspect List.
The focus now shifts to the Suspect List, with a Storage Server selected pseudorandomly to check the liveness of the server which has just been moved onto the Suspect List. If this check also reveals that the Server is indeed down, the check is repeated by at least one more server at which point, a final liveness check is performed and if the Server is found to be down again, it's entry is moved into the Black List and a timer tracking the downtime is started. Entries in the Black List are checked by pseudorandomly selected Servers with an exponential back-off for upto 3 days after which it is thought to be dead.
The process is similar for a Storage Server which is in the Black List or Suspect List and is coming back on but only that it is moved back into the Whitelist after 2 confirmations of liveness at which point the timer calculating the downtime is turned off and the results are logged.
This operation is performed off-chain with a Storage Server selected to submit the distributed list with the day's "score" as a transaction to the EOS Blockchain.
The overall uptime score is computed and taken into account when computing a Storage Server's incentive at the time of withdrawal of funds.



Scenario 1 - Storage Server going offline

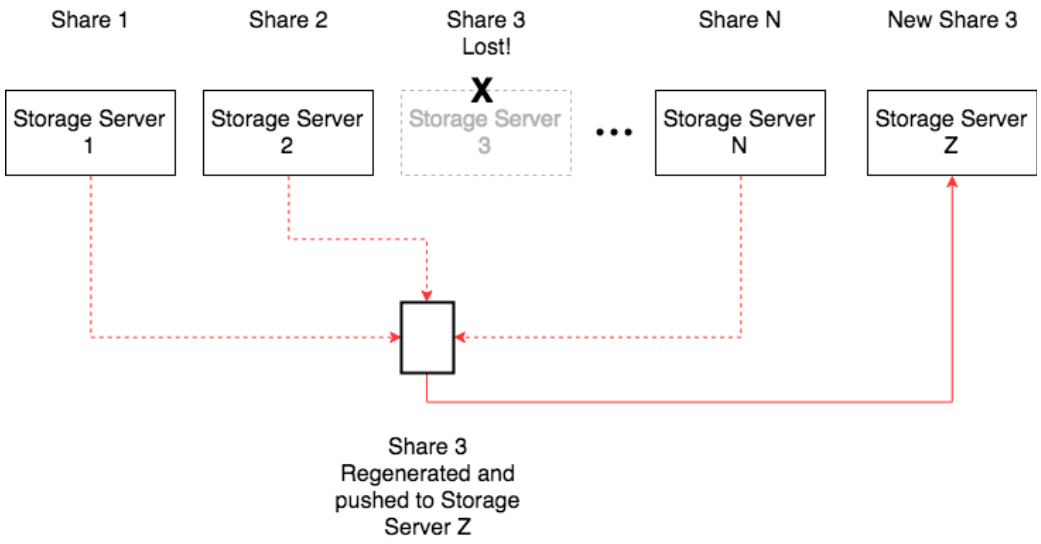


Scenario 2 - Storage Server coming back online

2) Proof of Retrievability

Tahoe LAFS provides a verification mechanism by which a formal Proof of Retrievability can be performed.
A pseudorandomly selected Storage Server is tasked with retrieving a random byte from a specific file. The byte is fetched from a Segment for an erasure coded file which is downloaded as part of the GET request for this byte of data. The Network is also made aware of the servers that were responsible for delivering this Segment. If the request is successful, it means that necessary system and cryptographic verifications indeed passed and the Storage Servers responsible for the delivery of the Segment are housing the erasure coded blocks of the file in question. A large enough sampling of byte ranges and periodic checks would help weed out any malicious/faulty Storage Servers that no longer hold necessary shares of a specific file.
In case of failure, the malicious/faulty Storage Server's reputation score is affected and the necessary reconciliation steps are carried out as detailed under the Replication Verification described below. This method is extremely succinct with very low complexity/overhead in computation and communication.

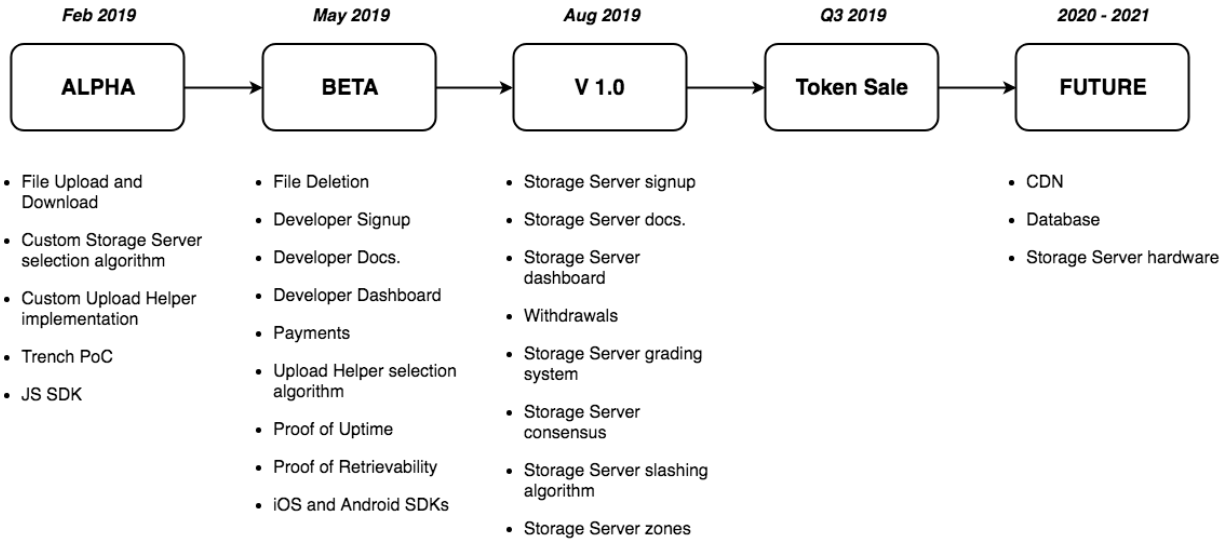
Replication Verification
Ensures that a developer's selected storage scheme of "k of N" is maintained in light of Proof of Uptime or Proof of Retrievability not holding up.



If a file's share is found to be missing as a result of a Storage Serving not being online(Proof of Uptime failure) or a Storage Server no longer hosting that specific share(Proof of Retrievability failure), the number of Storage Servers now hosting shares is no longer "N" but "N-1". The Service Nodes are tasked with requiring to "repair" the file's now broken

encoding and regenerate the missing share and move it to another storage server whereby maintaining the “k of N” scheme and ensuring a files availability and redundancy.

Roadmap



References

1. https://www.marketsandmarkets.com/PressReleases/cloud-storage.asp
2. https://www.marketsandmarkets.com/PressReleases/cloud-storage.asp
3. https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html
4. https://www.statista.com/statistics/233725/development-of-amazon-web-services-revenue/
5. https://akamai.gcs-web.com/static-files/1056afd9-af0d-426b-b13c-0171306fbe79