Dmitry Kochin[1], Alexander Neymark[2]

April, 2017 - March, 2019

Table of Contents:

[1] linkedin.com/in/kochin

[2] linkedin.com/in/neymark

1 Data storage issue and modern world

Data sharing and storage are the Internet's keystones. Millions of people read news, sign up in social medias, use messengers, upload documents, videos and images on a daily basis.

1.1 Key data features

1. Requires swift access
2. Continuous and free access (or reasonably priced) is prefered.
3. Data should be uncorrupted (should be possible to filter spam and fake news)
4. Access should be of **low cost**
5. There should be no censorship and limitations

Unfortunately, today we suffer from fake news and information control - sometimes the amount of information exceeds our ability to tell authentic reports from unreliable ones, while sometimes information is so scarce that people start relying on rumors just because some forces may try to cover the truth. Some websites are blocked, private correspondence is read and monitored, some news never reach those interested in it.

Those are important issues we can't solve without applying new technologies.

1.2 Major data sharing and storage challenges

1. **Connection quality**. Data transfer speed has crucial significance. All data sources have various internet access and different way of sharing data. All gadgets may have different status: one PC is on, while another is off, someone uses ones gadget while someone is taking a day off, some users may had not paid for wifi while some may even broke their smartphone occasionally,
2. **Data reliability**. Quite often we can't tell who spreads rumors and fake news and who may be lying to us. News agencies eagerly pass fake news around in pursuit of clicks and ratings. Youtube videos actively misinform people and may even lead them to panicking. It is hard to battle malicious info since it is hard to track the source.
3. **Cost of data**. You have created unique content and you want to monetize it. You actively sale your content through various channels, but you can never tell for sure how many copies were sold, so you lose money. Perhaps, mediators simply underpay you. And you can't do anything about it.
4. **Censorship and limitations**. You know those situations pretty well. Websites get under restrictions private correspondence is monitored. "Liking" wrong content in a wrong country may lead you in jail. Censorship is spreading the world and classic techniques can't do much about it.

How one can exist in such a mad world? Hopefully, we got OASIS - network, where access to information is never infringed.

1.3 What is Oasis and how it works?

Oasis is a decentralized database. It works in a way similar to blockchain.Anyone may become a user and store data on one's computer.

Oasis is decentralized - it can't be banned, shut down, blocked or limited in access. As long as even a single node stays, the access to its files will remain.

Let's take a look at an example to understand Oasis better. Everybody knows that torrents is peer to peer file sharing technology. For instance, you have a movie on your PC and you can share it with anyone who has access to this torrent. Why torrents can't be blocked? Because you can't block all devices in the world. As long as somebody sharing the content - you can download it.

What can be blocked? Commonly regulator blocks websites that catalogues what devices contains what file. This is the very difference between torrents and Oasis. Torrents are files, they contain no file

description. Simply put, you don’t know what computer contains the file you want.You need to look for website that describes it, and it is often locked.

Oasis solves this problem. But Oasis is more than just file storage. This is exactly the database where files, documents and documents are stored and can be found using keywords and descriptions..

Thus, Oasis presents great business opportunities:

- Information can be stored without introducing new technologies; data storage principles for businesses will remain unchanged. All you need is to access Oasis network (instructions will be provided).
- No censorship, no restrictions, your services will be very difficult to block.
- There is no risk that a computer with information will break or be turned off; other computers will instantly replace it - access to information is 24/7.
- Fast and reliable. Access speed is millions of requests per second.
- Information can be stored both openly and encrypted.
- Storage accounting is carried out by the blockchain and payment for storage takes place in Oasis project tokens. But a business can pay both in rubles and dollars under an agreement with the company Ethereus (in order for a business to avoid converting fiat to cryptocurrencies).
- Ethereus company will buy tokens by itself and will pay for them to node owners.
- Security is the same grade as bitcoin wallets. Information is stored securely and very conveniently.

Oasis is a new generation data storing system.

3. The concept of a public noSql database

For the last few years the blockchain community has been rapidly growing. Today blockchain not only gives people the opportunity to make safe financial transactions, but also provides a broad range of other services. Smart contracts have become a big breakthrough in this field. Smart contract is a program that is executed in the blockchain core allowing flexible custom processing of each transaction. Smart contracts in Ethereum [2] are Turing-complete and allow programming algorithms of any complexity.

That is why Ethereum has created a new market of decentralized applications – applications that are running directly on the blockchain inheriting its distribution, decentralization and security as well. However, this **market growth is restrained by the absence of an appropriate public data storage**. Serious applications require large and fast data storage and should be able to perform complex search within the files.

There are some implementations of decentralized data storages such as IPFS (4.9.3), cloud file storages (4.9.4) or special blockchains (4.9.6). But all of them have significant disadvantage - they do not allow complex search within the stored data. There are also distributed databases (4.9.5), and they have all the required features except the main one - Byzantine Proof Tolerance (4.9.1). So they can not be used in public untrusted environment.

The Byzantine problem is an experiment meant to illustrate the pitfalls and design challenges of

attempting to coordinate an action by communicating over an unreliable link, where failures of communication are possible. With that in mind, **Oasis programmers developed a resilient public decentralized noSql database, Oasis.DDB, that can be built in any blockchain with smart contracts; it supports replication, sharding, secondary indexes, full-text search and allows users to modify and delete data.** The database is **public** in the sense that anyone can establish a Oasis.DDB node and participate in the network processing user transactions and earn money from this. Equally, anyone can use it for storing data.

At the same time, Oasis.DDB maintains the same powerful processing system and speed that you find in private (intra-corporate) noSql databases. Since Oasis.DDB can be made compatible with any blockchain that supports Turing-complete smart contracts, members can use it to perform transactions on any platform. Learn more here (chapter 4.4).

3.1 Revenue sources for nodes

Oasis.DDB gains its revenue from the following sources:

- Storing content on the node server
- Retrieving content from the node server

When the user places the content in the database on the node server, he or she pays the node's service fee for storing and processing content. The system does not only motivate node owners to utilize the process, but also prevents attackers from littering the database.

4. Oasis.DDB architecture

4.1 Architecture features

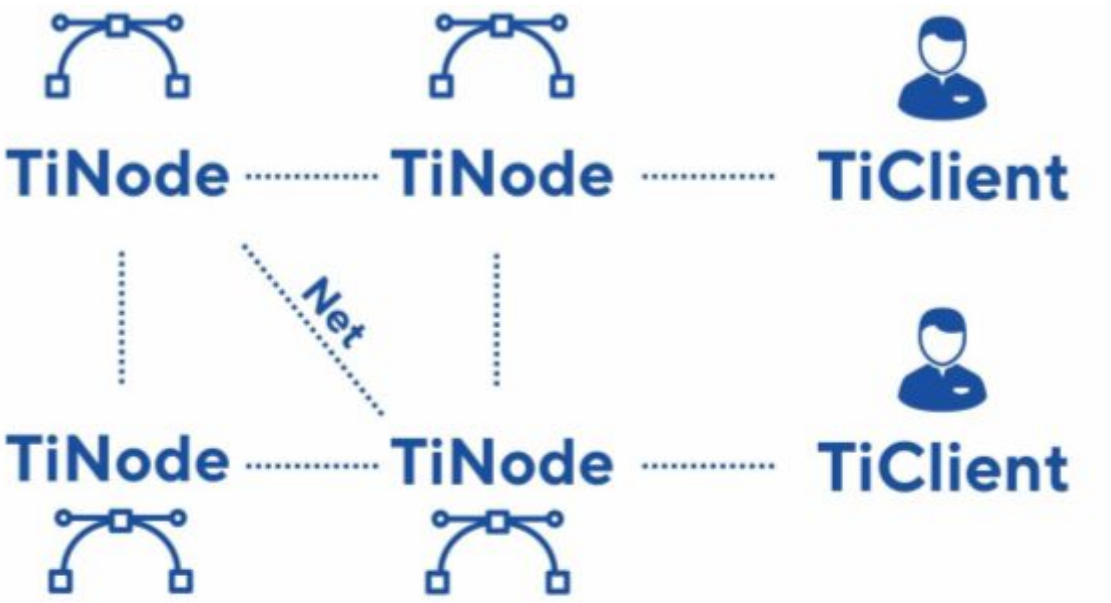

Fig. High level overview of the platform architecture

Oasis.DDB, as a decentralised high speed low latency database, conforms to the following requirements:

1. **Decentralization**. The Oasis.DDB platform represents a decentralized network of servers (nodes). Client applications connect to nodes within the network. At the heart of each TiNode is the blockchain that fosters decentralization. At the moment, various blockchain versions can be used for Oasis.DDB.

2. **Stability.** The platform is resistant to malicious activity of the participants (stability to Sybil attacks, the Byzantine Generals Problem, etc.). Blockchain intrinsically resists the malicious behavior of individual network members. However, if there is something else besides the blockchain in the system, additional efforts are required to ensure sustainability. We will return to this issue when we consider data storage (chapter 4.4).

3. **Anonymity** of individual servers and users, as well as **privacy** of communications between servers and users. Traditionally, IP masking and traffic encryption methods such as TOR [4] or I2P [5] have been used to provide anonymity and privacy for both platform clients and individual nodes by hiding their IP address. However, these methods are too slow and server-synchronized data require far more powerful speed processing. To resolve the issue, we use a Hyperboria mesh network [6] that uses the cjdns protocol [7].

4. **Storage of data**. The ability to store data and conduct a search through a large amount of structured data.

5. **Scalability.** The capability of the platform to handle a growing amount of work in order to accommodate member growth.

6. **Open-source**. All platform components have open-source code and are published with an open license.

7. **Publicity.** Anyone can join the network support system by installing the open software of the system.

8. **Profitability.** Users can profit from the platform.

9. **Speed**. The platform includes fast processing to achieve real-time computation and overcome the lag caused by current cloud-based models.

10. **Expansion possibilities**. The platform supports third-party applications (dapps) and provides a new model for building successful and massively scalable applications.

4.2 Architecture layers

At high-level we can distinguish the following layers of the platform.

1. TiClient - client application
2. Oasis.DDB - public decentralized database
3. Smart contracts
4. Blockchain
5. Hyperboria Network

Users interact with Oasis.DDB with client application TiClient. TiClient connects to Oasis.DDB and blockchain. Oasis.DDB is used to store and retrieve user data. TiClient and Oasis.DDB use smart contracts on blockchain for financial transactions and to ensure stability of critical operations. All the nodes are connected through Hyperboria network to provide speed, anonymity and end-to-end encryption of all communications.

4.3 Data storage

Oasis.DDB deals with a large amount of data, so we must choose the right place to store it. The distributed data repository should be made available for applications running on top of the blockchain with the following qualities:

1. Distribution
2. Publicity
3. Resistance to the Byzantine Generals' Problem and other forms of attacks in a public network
4. Sharding support (the ability to replicate only a part of the data on each node in order to increase data storage capacity)
5. Speed
6. Ability to store structured data
7. Ability to delete data

The problem is that there is no implementation currently available that meets all these

requirements. There are some decentralized file storages but they have drawbacks. The main one is that no solution provides tools for searching the files by their content, which is critical for most applications. We will overview current solutions in chapter 4.9 and underline their limitations. And in the next chapter we are presenting our own universal solution for decentralized structured data storage - Oasis.DDB.

4.4 Oasis.DDB

4.4.1 Oasis.DDB: Overview

There are many implementations of distributed databases that meet all the above requirements except one - Byzantine Fault Tolerance (see 4.9.5). Therefore, they can not be public. So we propose Oasis.DDB, which brings BFT to noSQL distributed databases and preserves their other qualities.

Oasis.DDB is a new-generation decentralized database with the following innovative interfaces:

1. **Distribution**
   Oasis.DDB supports an unlimited number of replicas, each of which can be a coordinator (see 4.4.4). By requesting any one of them, the user gets access to all data.
2. **Publicity**
   Oasis.DDB is created for operation in the public sphere. New nodes can be added to the network and will take on part of the load at any time.
3. **Resistance to the Byzantine Generals' Problem** and other types of attacks in a public network
   All data placed in Oasis.DDB is signed by owner (see 4.4.2), so nodes cannot arbitrarily change the data, nor can they corrupt data when replicating other nodes. Attempts to substitute are immediately detected through changes in the electronic signature. Any participant who does so, or attempts to do so, will be instantly removed from the network. External blockchain (for Oasis.DDB) is used for ODDB deposits, setting access rights and mutual settlements between the nodes.
4. **Sharding support**

   Each node is responsible for storing a certain range of primary data keys. Data replication has scalability, so it can grow with the network.

5. **Speed**
   Due to the data storage principles (see 4.4.2), the read/write speed in Oasis.DDB will be almost identical to similar private databases, such as Apache Cassandra.
6. **Ability to store structured data**
   Data stored on Oasis.DDB complements its platform. It can be a JSON document with a structure, which is useful for a particular applications.
7. **Ability to delete data**
   Data deletion is supported in Oasis.DDB. Although instant data deletion cannot be guaranteed, data will be deleted if the nodes act non-maliciously. A malicious node can never delete the data however it can not store everything, since only certain primary key intervals can be forwarded to it.
8. **Query language with an ability to conduct search using more than the primary key**
   We use secondary indexes similar to integration methods of ElasticSearch with Cassandra in the Elassandra project, that allow secondary key search as well as a full-text search.

In addition to the Etherus blockchain, Oasis.DDB can also be used for other blockchains. It relies on a blockchain, which supports Turing-complete smart contracts. Therefore, it can be used for other distributed

blockchains like Ethereum, RChain, NEO, and others.

4.4.2 Oasis.DDB: Data organization principles

Since the database needs to satisfy a wide range of blockchain applications, be flexible for rapid processing power, be resistant to the malicious behavior of other DB nodes, provide a sufficient level of replication, and have mechanisms to motivate participants to support the network, the DB is designed with the following properties:

1. The database is public, the user (client) of the database is identified by its public key. The public key is the user ID.
2. Each user can send transactions to the database. And each transaction must be signed by this user.
3. The new owner-signed record is created by the user.
4. Only the owner (or the user for whom the trust is installed through the permissions mechanism implemented as a smart contract on the blockchain) can change the record after creation.
5. Everybody can read all the records.
6. Every unique user identification code creates separate records.
7. More complex permissions can be installed using a smart contract in the blockchain (for example, trust between specific users, rights to create or delete tables, etc.).
8. All permissions must be checked for transactions and replications.

The mandatory cryptographic signature of each record ensures that no record can be changed or removed by a malicious party without knowing the private key. Data storage remains resilient to the Byzantine Generals' Problem attack, even without a consensus mechanism while speed remains the same as that of noSql databases.

On the other hand, an attacker can generate a Sybil attack, where a single adversary controls multiple nodes on a network, as it is unknown to the network that the nodes are controlled by the same adversarial entity. We can solve this issue with "motivation" or our incentive system.

4.4.3 Oasis.DDB Incentive System

A public network is a type of network where anyone has access and can connect to other networks or the Internet. Incentives are usually given to motivate participants and to encourage ethical participation.

Oasis.DDB similar to Ethereum Swarm [26] offers the following incentives:

- Reward for data extraction
- Reward for data storage

Rewards are allocated from the funds of the user who makes inquiries. Since payments through the blockchain are slow, two methods can be used for fast payments: off-chain transactions and "chequebooks." In off-chain transactions, the user needs to create an off-chain channel with each node of the database, or use intermediate channels between nodes. Since such channel requires its own funding repository, such an approach can be very expensive, so the "chequebook" approach is preferred. Before accessing the database, the user deposits part of their funds in a smart contract - "chequebook", and funds can be used as payment or reward.

The chequebook contract assumes the following:

● The contract monitors the total amount issued to each recipient at the time of the connection.

● When sending a cheque, the owner must memorize the total amount sent to each recipient.

A cheque is cashed if:

● The address of the contract corresponds to the address on the cheque.

● The cheque is signed by the owner (user ID - public key).

● The total amount on the cheque is larger than the amount in the previous cheque given to the same recipient.

Participants use “cheques” to reward nodes. The recipient node can only save the last received cheque from each user and he cashes it by depositing it in the "chequebook.”

4.4.4 Reward for data extraction

The data on the DB nodes has a certain level of replication. Specifically, the data with a specific key is stored only on a part of the nodes, for example, on *N* of them. However, the user can refer to any node for the data, which then acts as a "coordinator.”

On a user request the coordinator determines these *N* nodes by the data keys and routes the request to them. The data returned by the nodes is checked by the coordinator for compliance with electronic signatures and compared to the timestamp, after which the most recent record is returned to the user.

For this to work, the incentive has to fulfill the following conditions:

1. Faster nodes receive more payment.
2. Nodes that return old data would receive less payment.
3. Delinquent nodes (that fail to return the data at all) receive no payment.
4. The coordinator receives a fixed fee.

The coordinator issues an invoice with the data, which includes information on the nodes used. Later, the user writes a cheque for each. Then, the coordinator sends the cheques to the nodes. It also sends the update of the data to nodes that failed to return valid data.

To protect against malicious coordinators and delinquent users, each node maintains a list of users from which it expects payment. If the debt level exceeds a certain threshold, the node may stop accepting requests from these delinquent users and coordinators. And the lists are updated as cheques have been received.

4.4.5 Reward for data storage

The reward for extraction indirectly incentivizes storage of data but does so only for popular and often requested data. To encourage long-term data storage, especially if data are rarely requested, some sort of data storage incentive is needed.

This piece on Ethereum Swarm [26] describes the system of rewards for storage. Nodes enter into a data storage contract with the information owner for a period of time. The storage can be paid at the time of data storage (update) or after a certain period of time provided that the data is actually stored. In the event

of a loss of data is detected during the duration of the contract, the node may be penalized, as each node requires an initial registration with a security deposit.

When you store data, the node returns a receipt that proves that it has accepted the file for storage. This receipt then allows you to check the storage situation of the associated data and, if necessary, to initiate a legal smart contract to penalize the offending node.

Since data is not static, a record with the same key can be rewritten several times. This means that not only can the original record correspond to the presented receipt, but a record with the same key that is newer to the timestamp can also correspond.

When the user initiates a data deletion operation, instead of physically deleting data, the data is replaced with a special "zero" record. The record can be physically deleted after expiration of its storage contract.

4.4.6 Oasis.DDB: Full-text search

In simple noSql databases, a quick search with a small number of nodes is possible only with the primary key. A thorough keyword search is difficult to achieve without secondary indexes and full-text search capabilities. In this respect, Oasis.DDB differs from noSql databases. We suggest a solution similar to Elassandra [27], which uses the local full-text indexes of ElasticSearch [28] on each node of the distributed noSql Cassandra database. Full-text queries are sent by the coordinator (see 4.4.4) to all nodes, to be mixed and returned to the client. Since additional indexes are created locally and independently on each node, Byzantine Generals' Problem is no longer a concern here.

4.4.7 Conclusion

Built according to the above principles Oasis.DDB solves the problem of fast public data storage for decentralized applications, which need to perform advanced searches on the stored data. This is a unique solution by the moment. Oasis.DDB is public and can be used by any decentralized application on Ethereum. In the future Oasis.DDB can be ported to any blockchain with Turing-complete smart contracts.

4.5 Cjdns and Hyperboria network

To solve anonymity and privacy requirement, we use Hyperboria network. Cjdns (Caleb James Delisle Network Suite) is a networking protocol and reference implementation, founded on the idea that networks should be easy to set up, protocols should scale smoothly, and security should be ubiquitous. Cjdns' project page boasts that it implements "an encrypted IPv6 network using public-key cryptography for address allocation and a distributed hash table for routing." Essentially, the application creates a tunnel interface on a host computer that acts as any other network interface and is powerful in that it allows any existing services you might want to face a network to run as long as that service is already compatible with IPv6.

All traffic over Hyperboria is encrypted end-to-end, stopping eavesdroppers operating rogue nodes. Every node on the network receives a unique IPv6 address, which is derived from that node's public key after the public/private keypair is generated. This eliminates the need for additional encryption configuration and creates an environment with enough IP addresses for substantial network expansion. As the network grows in size, the quality of routing also improves. With more active nodes, the number of potential routes increases to both mitigate failure (think of “malicious generals”) and optimize the quickest path from sender to receiver.

Overall, Cjdns is not anonymous, nor is it intended to be. Rather, users use pseudonyms to hide their identities. To better conceal your identity, you can periodically change pseudonyms to make it unclear whether the requests come from one or several sources.

Advantages of the Hyperboria mesh network:

1. It is agnostic towards how the host connects to peers. Meaning, it doesn't matter much if the peer we need to connect to is over the Internet or at a physical access point.
2. It is encrypted end-to-end, stopping eavesdroppers operating rogue nodes.
3. Every node on the network receives a unique IPv6 address. This eliminates the need for additional encryption configuration and creates an environment with enough IP addresses for substantial network expansion.
4. The IPv6 addresses assigned to the nodes are not related to their location, which makes it impossible to know the physical location of the IP node.
5. High processing power.

Disadvantages of the Hyperboria mesh network:

1. The communicating nodes only know each other's IP, not those of other nodes with which they are not directly connected to.
2. Traffic is conducted via the shortest path, so all intermediate nodes know which nodes communicate but do not know what, and where they are.

The first drawback can be circumvented by the following:

1. A node connected to a network communicates with tunnels only with trusted nodes, if any exist.
2. Hyperboria's own intermediate node is created, and the node is linked by the tunnel to the system. Instead of your own site, you can use a trusted one, if one exists. Oasis.DDB users can use this method to connect to the network.

The second drawback can be reduced by altering the IPv6 identifier and the key pair of your own node. For intermediate nodes, passing traffic will look like it is coming from different nodes.

Despite these drawbacks, **Hyperboria is compatible for linking the Oasis.DDB nodes into a network, because it allows users to show their real IP address only to trusted nodes or to a trusted existing third-party Hyperboria node.** In other words, the fact that any traffic on the network is automatically encrypted makes it unnecessary to connect nodes through direct tunnels. At the same time, the connection speed remains high, and the platform gives you the security to publish IPv6 node addresses for connecting clients and for load balancing.

Users who want anonymity can benefit from additional connectivity to the nodes where some of the nodes are published as TOR Hidden Service and access to them is only achieved through TOR. For greater anonymity, you can use TOR with the VPN. Thus, the anonymity of the platform is realized by placing it in Hyperboria, and privacy is provided by mandatory encryption of all traffic, regardless of overlay services. Discerning clients can use TOR + VPN for connection with nodes.

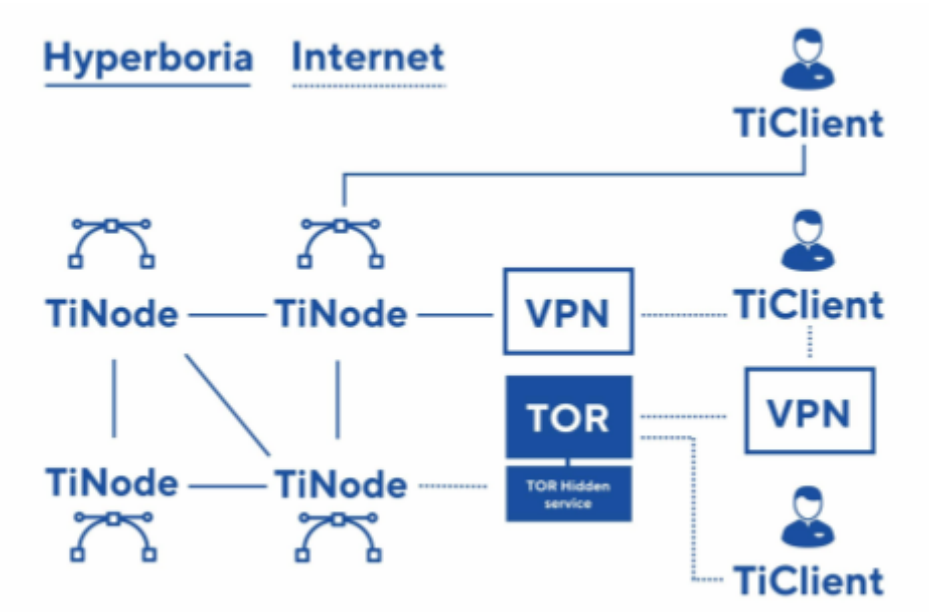

Fig. Anonymity and privacy

4.6 Client

Users interact with the system using the TiClient program. The client supplies the user interface, stores keys, interacts with the blockchain and TiNode nodes, as well as with other clients via chat protocols.

4.7 Chats

To sum things up, a chat software implementation is necessary for business communication and improved collaboration. In Oasis.DDB, a chat using end-to-end encryption, such as BitMessage, will be implemented, to connect network users.

4.8 Node and client interaction schematic

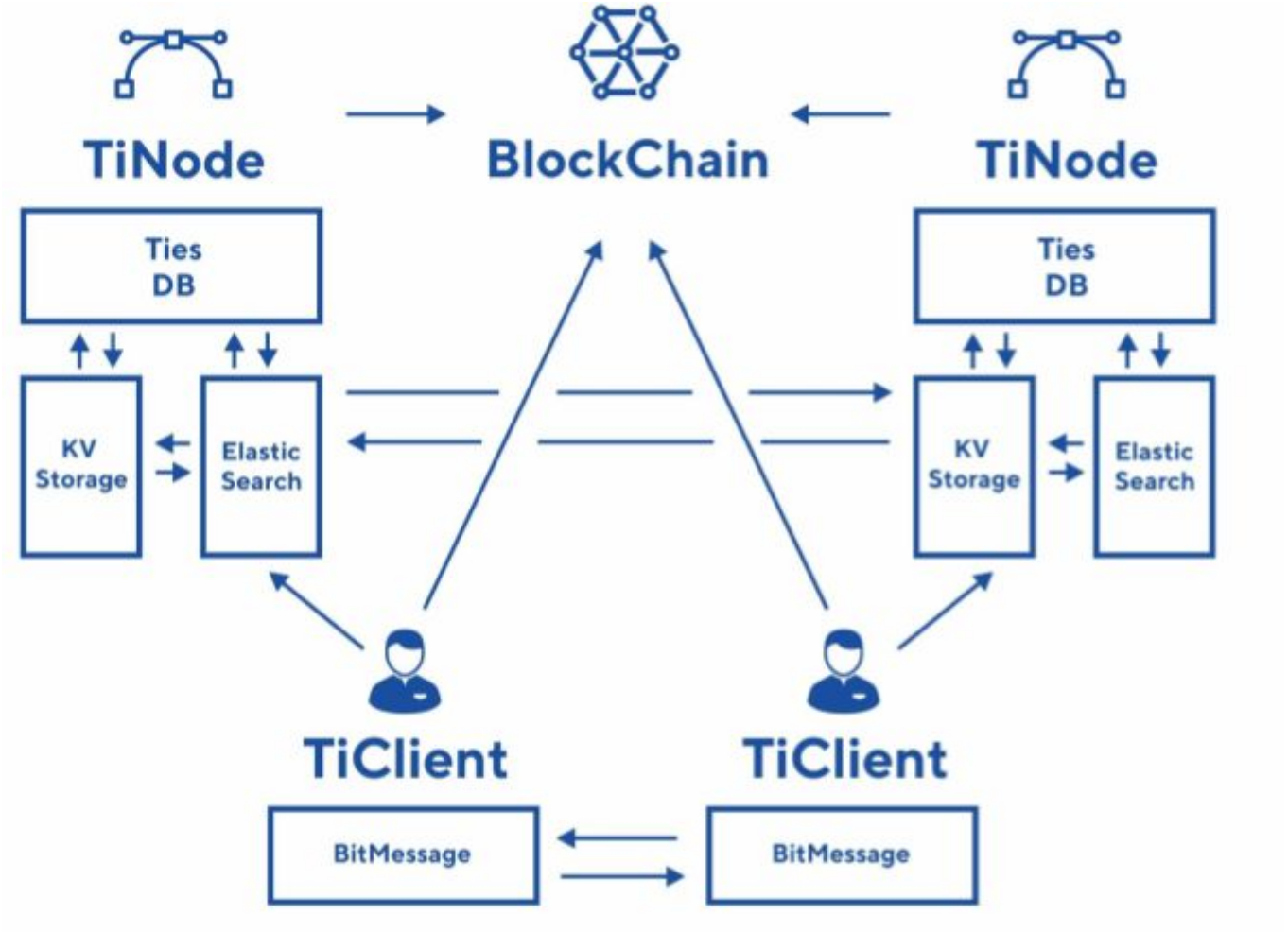

Fig. Node and client interaction schematic

4.9 Determination factors for the solutions used

In this chapter we consider the available technologies and justify our efforts to make a better approach to public decentralized data storage. While we do not use the solutions reviewed in this chapter we considered them thoroughly and found them inappropriate for being used in Oasis.DDB. Here we explain why.

4.9.1 Byzantine Generals' Problem

Open-source-platforms engender certain challenges, first and foremost of which is the so-called Byzantine problem (or "The Byzantine Generals' Problem") [1]. Reliable computer systems must handle malfunctioning components that give conflicting information to different parts of the system. This situation can be expressed abstractly in terms of a group of generals of the Byzantine army camped with their troops around an enemy city. Communicating only by messenger, the generals must agree upon a common battle plan. However, one or more of them may be traitors who will try to confuse the others. The problem is to

find an algorithm to ensure that the loyal generals will reach agreement.

The “Byzantine Generals’ Problem”, otherwise known as “Byzantine failures”, are considered the most general and most difficult class of failures among the failure modes. The so-called fail-stop failure mode occupies the simplest end of the spectrum. Whereas fail-stop failure model simply means that the only way to fail is a node crash, detected by other nodes, Byzantine failures imply no restrictions, which means that the failed node can generate arbitrary data, pretending to be a correct one, which makes fault tolerance difficult.

In this problem, several army factions surround a castle they hope to sack. A general leads each faction and one general is the lead general. Only a simultaneous attack ensures victory. Also, since the factions surround the castle, they are dispersed, making centralized command difficult. The generals must send messages between the factions to relay the attack time. However, some generals are traitors and will not obey the command or will relay the wrong attack time to the other generals. The generals do not know who is loyal and who is a traitor and there is no way to find out.

Problem: How can we ensure a coordinated attack to sack the castle?

Implications

In a distributed system, any inputs (messages) to the system (the agreed upon time of the attack) must be trusted. Digital networks usually have millions of members (the generals) who are dispersed globally and since there is no centralized command (no central governance), it is impossible for you to know each of the members. So how can you trust the other members of the network and ensure that the inputs to the distributed ledger are accurate and that the ledger, itself, has the correct information?

The Oasis.DDB architecture solves this problem and regulates a successful “coordinated attack” by providing a decentralized, self-governing network that vets all incoming transactions and uses secure cryptography to sustain trust despite a lack of central governance (see 4.4.2 and onwards).

4.9.2 Blockchain as a database

The blockchain is already a distributed data repository. So why not to store data directly in blockchain?

New implementations of blockchain, such as Ethereum and Etherus allow you to also store smart contracts, among other data. Smart contracts allow distributed applications (for example, dApps in Ethereum), which also store user information. Currently most of simple applications store all of their data in blockchain.

However, blockchain as data storage has significant drawbacks:

1. **Blockchain is immutable.** Everything stored in blockchain remains there forever and cannot be removed. This is a serious drawback given that most of the information of users` interaction is temporary and can be deleted later on. Eternal storage of information also works against anonymity.
2. **Data capacity is limited.** Each node is a complete replica of other nodes. As a result a popular application may cause the rapid inflation of the blockchain in size on all the nodes simultaneously. At some point, the blockchain may become too large in size and exceed the capacity of mass-produced hard disks. If this occurs, it would need an expensive equipment that could lead to unwanted centralization.
3. **It is slow.** The throughput of the Ethereum blockchain, the most production-ready blockchain in the market, is just about 15 TPS. It is absolutely not enough for a popular decentralized application.

4. **Primitive key-value storage without ability to perform complex search within users' data.**

Thus current blockchain implementations have several disadvantages making their use as a data storage ineffective. They are slow (a dozen transactions per seconds for a total network), have limited capacity due to excessive replication and immutability and are primitive in functionality (they are simple key-value databases without ability to perform complex search). Therefore, blockchain does not meet the requirements for the decentralized data storage we are looking for.

4.9.3 IPFS

IPFS [8] (InterPlanetary File System) is a distributed file system based on DHT [9] (Distributed Hash Table) and the BitTorrent protocol [10]. It uses content addressing to merge and integrate different file systems.

Advantages:

1. Devices only store needed files.
2. There's no need to trust peers, since addressing is done via the hash of contents.
3. IPFS provides resistance to "flooding" (i.e., loading useless files into the network) by peers downloading only the necessary files.
4. IPFS has a high transfer rate (thanks to BitTorrent).

Disadvantages:

1. IPFS only stores files (unstructured data, no content search).
2. User can only leave the network once file distribution is complete.
3. Other participants must be online to guarantee data storage.
4. The files are static (i.e. unchangeable).
5. IPFS does not delete files.

Both IPFS-based social networks, AKASHA (Ethereum + IPFS) [11] and the trading platform OpenBazaar [12], have all the disadvantages of the IPFS system, including storage limitations, and users are only able to leave the network once file distribution is complete. We don't find IPFS suitable for our data storage and have to find better solution.

4.9.4 Decentralized cloud file storages

Such repositories allow you to merge individual devices into a common cloud storage, similar to Dropbox [13], but with lower costs. Owners of such services (also called "farmers"), provide a place to store other people's files for a certain cost. They use cryptographic proofs to measure certain data such as proof of storage or proof of retrievability. Both trader and "farmer" use cryptocurrency as a medium for trade. Such projects are mainly created with DHT technology and content addressing. Some entrepreneurs also use blockchain and smart contracts.

The most prominent distributed storage protocols are Sia [14], Storj [15], Ethereum Swarm [16], MaidSAFE [17]. All of them are built using similar principles, while Ethereum Swarm is a decentralized platform for applications that also houses dApps.

Advantages:

1. Files are stored in the cloud and are available whether the owner is online or not.
2. High transfer rate.

3. Guaranteed storage security and file extraction.
4. You can delete unwanted files.

Disadvantages:

1. Storage of files only (unstructured data, no complex search).
2. Files are static.
3. Storage is a paid option.

Distributed file repositories are not suited well for storing structured dynamic information (such as user data of a social network) because their data search capacities are severely limited. For instance, in such repositories, we are not able to search by keyword, by location, or by a specific publication of the user.

4.9.5 Distributed databases overview

The CAP theorem [18] makes it impossible to obtain the fully distributed database that would ensure consistency, availability, and partition tolerance.

In our case, we need a distributed database resistant to partitioning but constantly available (Availability + Partition Tolerance + Eventual Consistency), because we need to quickly receive a response from the system on request. This limits our selection of near-noSQL databases, because ACID [19] SQL DBMSs primarily provide consistency.

There are many implementations of distributed noSQL databases like MongoDB [20], Cassandra [21], RethinkDB [22] that are easy to use and configure in a cluster with replication and sharding. Replicas are incomplete, that is, they imply sharding, which is a type of database partitioning that separates very large databases the into smaller, faster, more easily managed data parts.

The client works with one of the replicas, and the data is automatically synchronized with the others. For load balancing, sharding can be used when part of the data is stored only on part of the replicas. Adding a new replica to the cluster scales the cluster, and some implementations (for example, Cassandra) allow the replica to automatically control part of the cluster work.

NoSQL databases provide "eventual consistency," that is, the system is eventually consistent. If no updates are made to a given data item for a “long enough” period of time, then, eventually, all reads to that item will return the same consistent value.

NoSQL databases can store both a simple key-value, and maintain the internal structure of the value, as well as additional indexes. The most advanced ones also have basic transaction support and an SQL-like query language (for example, Cassandra).

In all of the above, this database class may seem ideal for use in a blockchain. But if a malicious replica is added to such a cluster, which starts to tell other replicas in the cluster that all data needs to be deleted, the result will be docile deletion of the data with the remaining replicas and corruption of the database. Thus, the coordinated work of replicas is now possible only in a trusted environment (a cluster of such databases is not stable to the problem of Byzantine generals). If a maliciously working replica is placed in a cluster, it can cause the destruction of the entire cluster data.

Advantages:

1. High speed
2. Linear speed and storage size scaling

3. Resistance to unavailability of certain replicas

Disadvantages:

1. Trust factor – vulnerability to the Byzantine Generals' Problem.

So, distributed databases have the only disadvantage but still the essential one. So they can not be used in decentralized system without modification.

Note, that Oasis.DDB brings Byzantine Proof Tolerance to noSQL database (see 4.4.2 and onwards). Due to special data organization and incentives mechanism Oasis.DDB does not need a consensus procedure, where nodes on the network share information about candidate transactions, because the user's encrypted signature protects the information. That is, harmful nodes can not delete and change data on other nodes without showing noticeable impact.

4.9.6 BigChainDB

The blockchain implementation, called BigChainDB [23], or IPDB (InterPlanetary DataBase) claims a powerful transaction speed (1 million per second) and enormous storage capacity (due to distributed storage with partial replication). BigChainDB gets these benefits through starting with a big data distributed database and then adding blockchain characteristics - decentralized control, immutability and the transfer of digital assets.

Unfortunately, BigChainDB's architecture is fundamentally flawed in that each node has full rights to write to the common data store, which means that its system is vulnerable to the problem of Byzantine Generals'. In other words, all BigchainDB nodes connect to a single RethinkDB cluster. If something bad happens to that RethinkDB cluster, all other nodes on that blockchain fall because they lack an independent storage. The authors of this project know about it, promising to think about this later [24]. However, fixing the fundamental flaws in the underlying architecture after the release of the product is very laborious and often impossible, as this can lead to a significantly different product with a different architecture. Such an easy approach to the fundamental problem causes criticism from the project community [25], since the high speed and mass characteristics of BigChainDB, demonstrated in the absence of BFT (Byzantine fault tolerance), are in fact those demonstrated by the RethinkDB and MongoDB databases used for data storage. But since you still need complete trust between the nodes, why not use the specified databases directly?

Our summary of the BigChainDB is the following:

Advantages:

1. Speed and storage are comparable to distributed noSql databases

Disadvantages:

1. BigChainDB is an ordinary noSql database that additionally has all the blockchain drawbacks.
2. Permanency (the data can't be deleted legally, but can be deleted maliciously).
3. Susceptible to the Byzantine Generals' Problem, thus it can't be used in a public network.

For these and other reasons, we have concluded that BigChainDB is unsuited for Oasis.DDB data storage.

4.9.7 Conclusion

Having carried out thorough analysis of existing storage solutions we conclude that no current solution

meets the high demands of emerging decentralized applications. What they need is decentralized public database.

5. ODDB tokens

ODDB tokens are the digital currency used by members on the Oasis.DDB platform.

5.1 Operations with ODDB tokens

1. Purchase and sale of tokens is conducted via the internal platform exchange or via an outside exchange which lists ODDB tokens.
2. All deals transacted over the platform are paid for with ODDB tokens (which can be converted into other cryptocurrencies (e.g., BTC, ETH, or Ripple) and into fiat currency).

6. Conclusion

This paper reviewed the proposal of Oasis.DDB: that it is a public distributed noSql database with powerful processing speed, that it supports a secondary index and full-text search capacity, and that it can be used in conjunction with any blockchain that supports smart contracts.

7. References

1. Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine Generals Problem. ACM Transactions on Programming Languages and Systems (TOPLAS), 4(3):382–401, July 1982. http://research.microsoft.com/en-us/um/people/lamport/pubs/byz.pdf.
2. https://ethereum.org/
3. https://www.rchain.coop
4. https://www.torproject.org/
5. https://geti2p.net/
6. https://hyperboria.net/
7. https://github.com/cjdelisle/cjdns
8. https://ipfs.io/
9. https://en.wikipedia.org/wiki/Distributed_hash_table
10. https://en.wikipedia.org/wiki/BitTorrent
11. https://akasha.world/
12. https://openbazaar.org/
13. https://www.dropbox.com/
14. http://sia.tech/
15. https://storj.io/
16. https://github.com/ethersphere/swarm
17. https://maidsafe.net/
18. https://en.wikipedia.org/wiki/CAP_theorem
19. https://en.wikipedia.org/wiki/ACID
20. https://www.mongodb.com/
21. http://cassandra.apache.org/
22. https://www.rethinkdb.com/
23. https://www.bigchaindb.com/
24. https://docs.bigchaindb.com/en/latest/bft.html и https://github.com/bigchaindb/bigchaindb/issues/293
25. https://reddit.com/r/Bitcoin/comments/4j7wjf/bigchaindb_a_prime_example_of_blockchain_bullshit/
26. viktor trón et al. “Swap, swear and swindle incentive system for swarm”, http://swarm-gateways.net/bzz:/theswarm.eth/ethersphere/orange-papers/1/sw%5E3.pdf
27. http://www.elassandra.io/
28. https://www.elastic.co/products/elasticsearch