WHITE PAPER

Dynemix cryptocurrency platform

Dynemix is a blockchain platform of the next generation designed to become the first worldwide-adopted cryptocurrency capable of competing with conventional consumer-level payment infrastructure on equal footing and substituting it entirely eventually.

Dynemix features a novel BFT consensus protocol, which was designed from scratch to meet the specific requirements of the system. A number of unique features introduced with the protocol allow the Dynemix blockchain to solve all major issues that current generation blockchains are facing and endow the system with the potential to become a new generation of money.

Dynemix features a breakthrough economic model of a decentralized algorithmic central bank, which is described in a separate paper.

Dynemix operates in conjunction with the **Liberdyne messenger**, which is described in a separate paper.

The following white paper is dedicated to the general description of the project and the technical description of the blockchain.

The following white paper contains an informal description of the project. The provided specifications are not final and are subject to amendments, if necessary, up to the mainnet launch. Any updates to this white paper can be made without prior notice.

The following white paper does not contain any coin purchase offerings or any other information on the initial coin distribution.

www.liberdyne.com

Contents

I. Introduction

In 2009, the first modern decentralized cryptocurrency, Bitcoin, was created. It was based on a distributed ledger called a blockchain.

In 2014, a fundamentally new blockchain platform – Ethereum – was born. Unlike Bitcoin, it was not strictly a cryptocurrency and was not primarily meant to become just a way of payment or a storage of value, but rather to be a decentralized virtual machine (also often called a distributed Turing machine, since its state machine used the Turing-complete instruction set) for running various kinds of applications, including so-called “smart contracts”, which allowed value transfers and the automatic enforcement of other various types of obligations under multiple conditions. Native ETH coins of the platform were considered a “fuel” for these applications, and blockchain was used as a database for storing applications as well as all concomitant data (accounts, messages, transactions, states etc.).

Most projects after Ethereum followed its footsteps and introduced variations of smart contract platforms with emphasis on different specific features mostly claiming to reach a better speed, scalability or lower fees, whereas the concept of an actual cryptocurrency, i.e. a decentralized payment system, was left for good.

Some more recent projects tried to address the issue through the introduction of the concept of an algorithmic stablecoin. These projects, however, were poorly executed and didn’t cover the entire array of problems that needed to be solved to make the concept work. We can claim that to date, there is still to platform that is even remotely close to the embodiment of the initial ideas behind Bitcoin to become an actual decentralized cryptocurrency.

We are here to fill this gap and finally introduce a decentralized payment system of the next generation capable of becoming the first mass-market cryptocurrency. To embody our vision, we endowed the platform with all necessary properties and developed unique solutions to known major issues.

We can conditionally split the requirements for the ultimate cryptocurrency into four categories and address each of them separately:

a) **User experience.** General audience is used to the level of convenience provided by the conventional fiat payment infrastructure. Current blockchains are far too inferior to trigger a substantial interest and raise a desire to switch to crypto. Our system must be designed so that users won’t feel a significant drop in convenience in comparison to credit cards and common banking apps.

b) **Decentralization.** The system should be resistant to oligopoly control, censorship attempts and cartel formation. As well, the functions of system support and rewards for performing these functions should be distributed among as much participants as possible. Decentralization should not be sacrificed to a large degree for speed and scalability.

c) **Economic potential.** The platform should be capable of transitioning from the initial speculative stage to the development of a real economy. To accomplish this, we should build not just an algorithmic stablecoin that is pegged to an allegedly stable exogenous currency but a fully-fledged decentralized central bank.

d) **Entry threshold.** Most current blockchains feature a very sophisticated infrastructure, which requires a set of specific knowledge even for the most common usage. We should introduce our platform via a simple and user-friendly software and assure that all important functions are accessible through a coherently designed uniform interface.

The following white paper contains the description of Dynemix blockchain, which incorporates a number of novel solutions for each of the issues addressed above. The economic model of Dynemix is described in a separate paper. The description of the Lyberdyne messenger, which will be used as a platform for the distribution of Dynemix, is also available as a separate paper.

II. Problem Statement

1. Required technical features

To accomplish our goals, at first, we need to address two of the stated features that we want to see in a perfect decentralized payment system: it should be sufficiently decentralized, and it should provide a user experience similar to existing centralized payment systems (or even better, if possible) to be able to offer serious competition.

The properties required to accomplish these tasks are as follows:

1) Guaranteed instant transaction processing

In conventional centralized payment systems, any transaction sent to the network will be instantly processed by the operator, except when it does not fit the requirements (e.g. if the transaction amount exceeds the user balance) or a fault occurs. We should achieve the same properties in our blockchain system and design it in such a way that it can assure that all currently pending transactions are added to the next block.

Current state of the problem:

In Bitcoin, transactions to be included in a block are selected by miners at their own discretion. There is no guarantee that any particular transaction will be ever included in the blockchain. As conceived by the creator, miners are incentivized to add transactions to a block by obtaining the commissions included in each transaction. The protocol, however, contains no strict rules about transaction selection; hence, miners are free to produce even empty blocks if they wish (which led to the emergence of a practical problem called “spy mining”).

Mostly the same situation can be seen in all widely known blockchain projects to date. Only the developers of DAG-based systems claim to provide processing of every transaction, but their design relies on altruistic behavior by participants, so it cannot be considered a proper solution to the problem in a Byzantine environment. Some blockchain developers also addressed the problem (e.g. Honey Badger), but no one fully provided the stated properties.

2) Quick finality

There should be no scenarios where users have to wait more than 30 seconds for a transaction to be completed. In centralized systems like Visa and MasterCard, it usually takes just a few seconds (although we should note that, even being confirmed, the transaction takes a lot more time to be finalized, but this does not significantly affect the consumer experience), so not many users will agree to wait much longer.

Due to the distributed system’s specificity, we cannot match the speed of centralized systems (because we need to use a consensus protocol, which takes time to process and synchronize the data between peer nodes). We must, however, get as close as possible.

Current state of the problem:

In Bitcoin, a transaction that has been put into the blockchain is considered to be reliably confirmed after six blocks have been mined on top of it. This takes more than an hour, which feels like an eternity in comparison to credit cards.

Processing speed, however, is one of the parameters on which most developers concentrated later, so the majority of current-generation projects claim to reach finality latency close to that of centralized systems or even surpassing them.

Although the Byzantine fault tolerant (BFT) consensus model, which is used in most recent protocols, allows reliable finality to be reached quicker, most of these claims are based on heavily centralized system architecture. Nevertheless, matching our targeted latency boundary is feasible for many recent BFT-style designs.

3) High transaction processing rate

According to Visa, VisaNet processes an average 1,700 transactions per second (TPS). If we want to create a system that can be as widely used as conventional means of payment all around the world, it should be capable of handling about 10,000 TPS. Moreover, this should be achieved within the main transaction database without second-layer technologies like payment channels.

Current state of the problem:

Bitcoin can process about 7 TPS the with the current block size, which is extremely insufficient. Though this is nothing but an arbitrary choice, in any case, Bitcoin's architecture cannot provide thousands of TPS on-chain with an acceptable decentralization level. Developers try to implement second-layer solutions (such as Lightning Network), but in our opinion it feels more like concealing the problem rather than solving it.

Since scalability has recently become the unofficial primary benchmark of a blockchain system's advancement level, developers started a contest for the highest TPS claim. Most of these claims have nothing to do with real-life scenarios, as they are based either on the capabilities of second-layer solutions (which are theoretically limited only by hardware and the bandwidth of all nodes combined) or they do not take into account the actual possible hardware and bandwidth capabilities of peer nodes in a practical environment.

A realistic throughput boundary that can be practically achieved by non-sharded designs with the current state of hardware and communication infrastructure lies roughly between 1,000–5,000 TPS and can be pushed further only with the help of sharding.

4) Free transactions

In centralized systems like Visa and MasterCard, commissions are charged to payment-system participants, but ordinary consumers can perform most operations free (consumers usually pay banks only for cards issued and account maintenance). Therefore, for most users, transactions seem to be free of charge. Only those few who actually know how the system works understand that the vendor includes a transaction fee in the price of a service or good.

A free-of-charge payment system can become very attractive both to users and to businesses, especially for microtransactions. The absence of commissions can give it a very strong advantage over other payment systems of either centralized or decentralized design.

Current state of the problem:

Bitcoin has rather high transaction fees, which grow even higher during periods of increased system load. This may be partly caused by the Bitcoin Core developers' refusal to increase block size. Other altcoins of similar design will most likely face the same situation if they become similarly popular.

Some more recent projects claim to provide free transactions, but in reality, they either feature different ledger designs (e.g. Nano, IOTA), which have their own significant downsides in comparison to blockchains, or the burden is simply shifted to third parties (developers, validators etc.).

5) Financial privacy

Since blockchain is a public ledger, any person can view all of the transactions in the network. Hence, if the user discloses his ID to a third party, this party will instantly know the user's current balance and entire financial history. This is a significant downside from the point of view of an average consumer who intends to use such a system for everyday payments, which means that the system should be capable of hiding transaction attributes and balances from the public.

Current state of the problem:

Surprisingly, first-generation projects are the only ones among popular platforms that offer banking secrecy compatibility, but this fact has a reasonable logic beyond it – the technical solutions offered are applicable only to UTXO based platforms and put an intense load on the system, which is why they are less compatible with the decentralized virtual machine concept. The most popular platforms that offer a means of hiding transaction attributes are Monero, Dash and Zcash.

Not many developers of recent projects care about this factor, and some even state that anonymity should be excluded from cryptocurrency operations. We can state that no better solutions than the ones used in the first-generation blockchains mentioned above have been introduced to date.

6) True decentralization

We already have a working centralized payment infrastructure, and there is no need for another similar system that differs only in the presence of the word "blockchain" in the title. Since making the system more centralized is apparently a way of resolving the scalability issue, many developers have chosen this path.

We do not support this approach and believe that decentralization is a crucial feature, which is why the system should be designed in a way that prevents any possibility of oligopolistic control and censorship attempts or any other possible kinds of coordinated selfish behavior.

Current state of the problem:

Bitcoin was the first working concept of a decentralized payment system, and its creator paid a lot of attention to the notion of decentralization. The very reason of Nakamoto consensus invention was the need to find a fault-tolerant and Sybil-proof solution for setting transaction timestamps without involving a trusted intermediary.

Although the proposed solution seemed appropriate in the early stages, the appearance of mining pools dramatically changed the situation, and currently the major fraction of Bitcoin's hashrate is controlled by just a few actors. The oligopolistic trend shows that the PoW concept is far from perfect in terms of providing decentralization, not even mentioning other known issues, such as AsicBoost.

Since that, most developers have concentrated on scalability and speed issues, trying to provide as many TPS as possible. The easiest way to speed up a decentralized system is to centralize it (thereby reducing its security as well), which is the path most projects followed. We can see a clear trend of moving back to something that looks more like client-server architecture, while pretending to be a peer system.

That has led to the creation of pseudo-decentralized systems, where instead of one trusted processing party, blocks are formed by a limited group of pre-appointed validators, thus making these systems not much less centralized than Visa or MasterCard.

To date, we can state that the overall situation with decentralization among the majority of recent platforms is very sad, despite the fact that the decentralization concept is actually the very reason for cryptocurrency technology's existence.

2. Overcoming entry threshold

After we create a system that is perfectly capable of being a consumer-level payment platform, we will face another problem: how to spread such a system among a wider audience.

The crypto community today is a rather closed system of professionals and enthusiasts that grows at a moderate rate. One of the main obstacles to the rapid growth of the number of people involved in the crypto community, in our opinion, is a high entry threshold caused by the following factors:

a) **The complexity of use**

Very specific knowledge is required to use the existing tools. Not all IT professionals, to say nothing of inexperienced users, understand the mechanisms of cryptocurrencies (cryptowallets, public and private keys, light and full clients, smart contracts etc). Ordinary users lack access to a simple and understandable product that would allow for the use of cryptocurrency in two clicks.

b) **The need to invest fiat funds**

To obtain crypto coins, an ordinary person first needs to understand how the exchange service works and second needs to spend a certain amount of fiat money. One may try to obtain cryptocurrency by mining, but nowadays mining has turned into a professional activity that is absolutely impractical without special knowledge, experience and significant investment.

Considering that cryptocurrencies face strong prejudice created by media among the general audience, the risk of losing money mostly prevails over the audience’s curiosity.

c) **The lack of real use scenarios**

Cryptocurrencies are mostly used as an investment tool, and the real economy within the cryptocurrency environment is just beginning to develop. The volatility of the exchange rate does not help the development of the real economy either, but this does not seem to bother many developers, who are well enriched from bubbles in the stock market. As for the users who are not interested in investing their savings, most cryptocurrencies cannot offer them any relevant methods of use.

As a result of these obstacles, many people unrelated to the IT industry refrain from using cryptocurrencies, despite the fact that they may have an interest in them. In fact, the money decentralization concept introduced by Bitcoin is beneficial for the general audience and not only for geeks, but there is too much confusion caused by presenting Bitcoin as a bubble and a financial scam by media. Combined with the other mentioned threshold factors, this heavily reduces the potential for cryptocurrencies to proliferate.

Having assessed all of these factors, we came to the conclusion that if we really intend to introduce a cryptocurrency as a payment instrument for the consumer market, we must present not a mere blockchain protocol but a complete product on top of the protocol with the possibility of integration into a convenient ecosystem. If we simply create a technical platform, we will face the same entry threshold that the users of other cryptocurrencies are forced to overcome. In this regard, for the cryptocurrency to accomplish this task, a basic platform app should be created that will be capable of becoming a popular product itself, gaining interest among users regardless of the presence of an integrated cryptocurrency.

We decided to incorporate Dynemix into the Liberdyne messenger to solve the issue of the entry threshold. In addition to being coherent with the peering principles upon which blockchains are designed, the messenger also incorporates a technical solution for helicopter coins that allows us to introduce a breakthrough model of a decentralized central bank and incorporate powerful marketing features that can greatly increase the rate of the platform’s growth.

III. Dynemix Cryptocurrency Platform

1. Brief overview

Dynemix is a decentralized, permissionless, account-based blockchain system powered by a unique Proof-of-Stake BFT consensus protocol.

Dynemix is designed to achieve one goal – to become the first widely adopted decentralized means of payment (commonly called a cryptocurrency), compete with conventional centralized payment infrastructure on equal footing, and even substitute it eventually.

Dynemix operates in a symbiotic relationship with the Liberdyne messenger, which allows it to introduce features not previously available and to progress toward the stated aim.

Dynemix features a breakthrough economic model of a decentralized algorithmic central bank.

To achieve the declared goal, Dynemix is endowed with the following features:

a) **Transactions are free.**

Unlike other blockchain systems, which feature market-determined fees for all transactions, common transactions in Dynemix are free. Fees are charged only for business-related transactions that require multi-outputs.

b) **All transactions are finalized within 6 to 16 seconds after being sent to the system.**

Unlike other blockchain systems, which allow transactions to be added to the block at the discretion of the block producer, every transaction broadcast into the Dynemix network is added to the next block, which is instantly finalized. There is no need to wait for additional confirmations.

c) **Dynemix scales to 10,000 TPS and higher.**

Our sharding solution allows Dynemix to scale up as much as needed. The system can scale to 10,000 TPS and beyond with moderate hardware and bandwidth requirements.

d) **Dynemix is highly decentralized.**

Unlike many recent blockchain projects, which solve the scalability issue by adopting highly centralized architecture, Dynemix is designed specifically to provide the highest level of decentralization, which surpasses even the level of classic PoW blockchains, such as Bitcoin.

e) **Dynemix is highly secure.**

With the help of a two-layer security model, Dynemix manages to increase scalability without the need to sacrifice security.

f) **Dynemix can be minted on a common PC at home.**

Due to its excellent optimization, the use of sharding technology and the new consensus protocol, it is possible to run a full node on common home-class hardware. No professional hardware is required.

g) **Dynemix introduces a new economic model.**

The system features a unique coin issue and reward distribution system that provides a solution to the problem of economic development.

h) **Dynemix is compatible with financial privacy.**

Dynemix is designed to use a new cryptographic solution that encrypts transaction data and can allow financial privacy to be provided to all users.

i) **Dynemix solves the entry-threshold issue by being integrated into Liberdyne.**

Dynemix is integrated into the Liberdyne messenger to provide users with a more familiar experience. Newcomers do not have to study the peculiarities of cryptocurrency technology to start using Dynemix. Instead, the user only needs to install Liberdyne and start using a familiar type of app that happens to have additional functions, which are available through a simple and intuitive interface. Minting functions are set up automatically, so the user does not need to do anything at all.

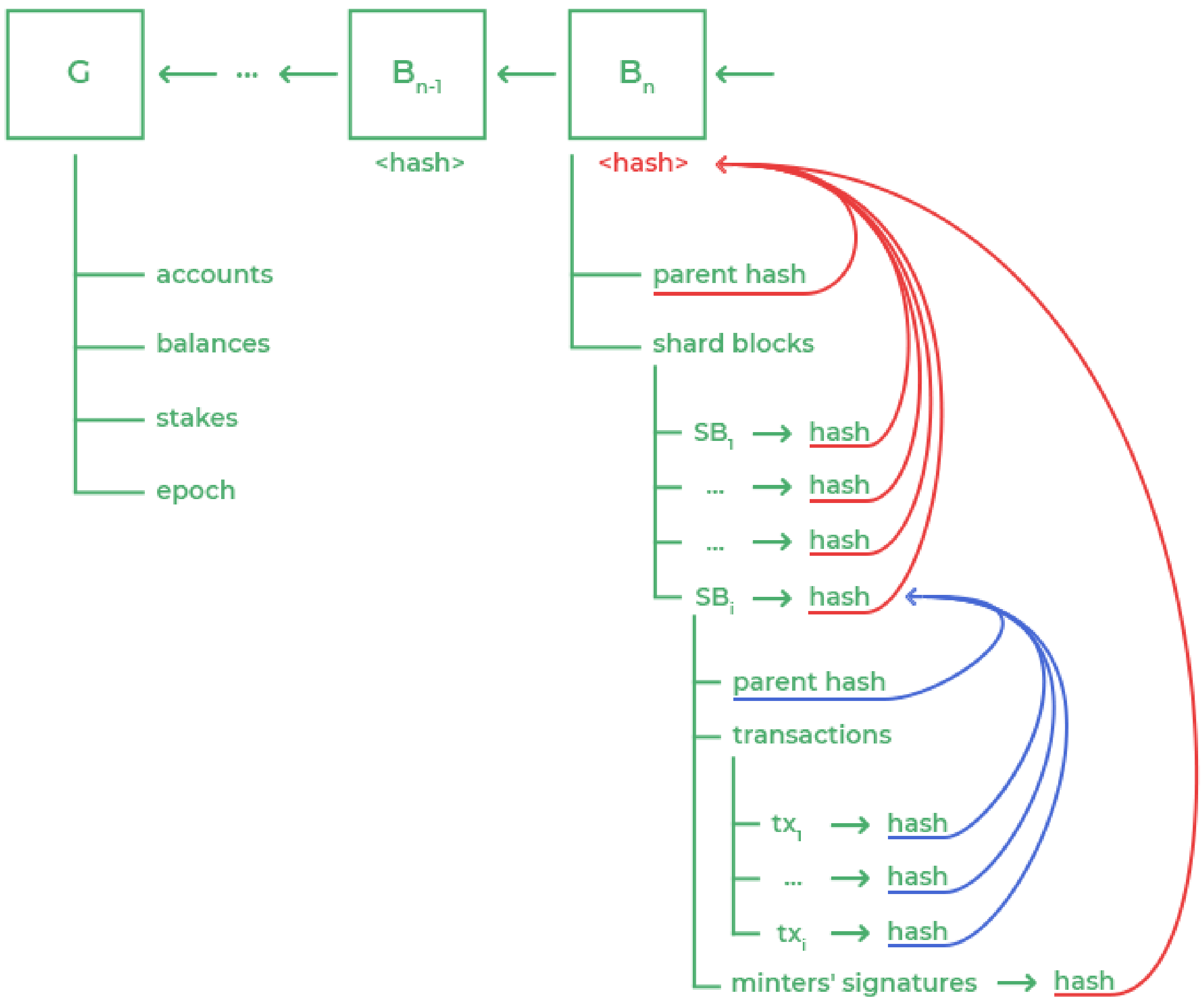
The combination of these features makes Dynemix the ultimate blockchain payment platform and a true breakthrough in cryptocurrency technology.

2. Data structure of Dynemix

Dynemix is based on a perpetually growing transaction ledger called a blockchain. Dynemix blockchain consists of a sequence of master-blocks that are added in the linear order.

Unlike most other blockchain systems, participants in Dynemix can operate not only directly with complete data blocks but can also employ a simplified data structure, which is derived from blocks and called *a quilt*.

1) Blockchain – master structure



Essentially, each master-block represents a collection of shard-blocks and contains the following data:

- the hash of the previous master-block;
- a set of shard-blocks;
- a set of minters' signatures for shard-blocks.

Each shard-block contains:

- the hash of the previous master-block;
- a set of transactions.

All data is organized into a hash tree.

2) Quilt – derived structure

Quilt is designed to provide better optimization and scalability. Operating in the quilt mode is most efficient in the case of balances' homomorphic encryption implementation, which is the particular setting for which quilt was designed. Even without encryption, however, quilt can provide an increase in performance, although the result is not as impressive as in the specified case.

In the early stages, quilt will not be engaged, and Dynemix will fully operate in the blockchain mode, which assumes the propagation of complete data blocks among all nodes.

At a certain point, however, balance encryption implementation should be considered in order to comply with financial privacy standards. The system was initially designed to provide a smooth subsequent transition to an encryption scheme.

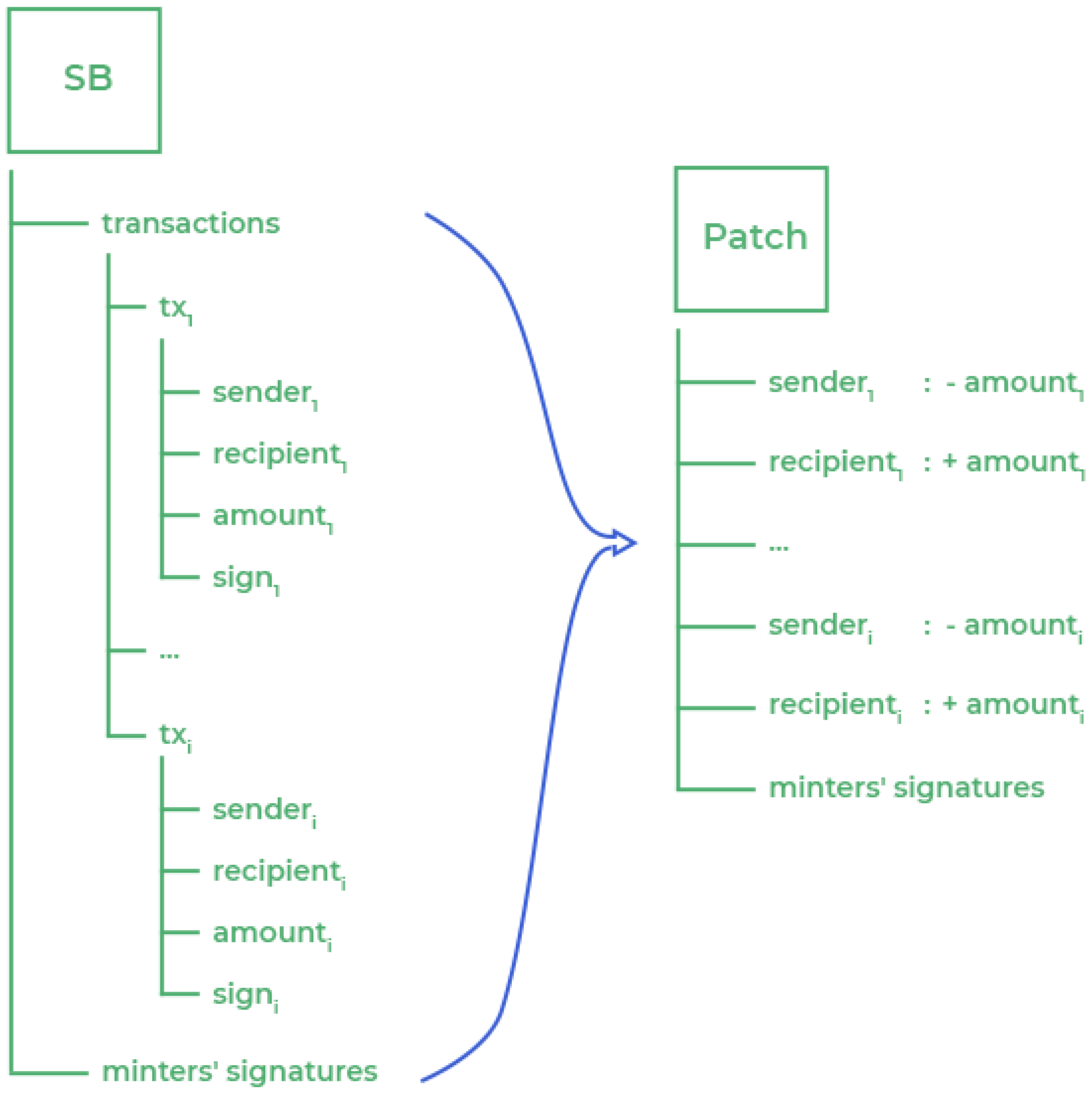
Balance encryption will inevitably affect performance. The weight of transactions and computational overhead will dramatically increase, thereby possibly dropping throughput on the blockchain layer by an order of magnitude. To partially compensate for the performance decrease while maintaining the desired level of scalability and decentralization, full nodes will be allowed to operate with a pruned data structure that is derived from the underlying blockchain.

Once minters of a shard construct a valid shard-block, they derive a data structure called a *patch* from the newly constructed block. While the shard-block contains transactions with all attributes included, the patch contains only changes of the state that are caused by transactions. The hash of the patch is included in the hash tree of the shard-block.

At the master consensus stage, instead of exchanging complete shard-blocks, minters exchange patches and partial hash trees of shard-blocks, along with signatures. Thus, after reaching a master consensus, each minter will now possess a complete set of patches, which form a quilt. Essentially, a quilt represents the set of changes applied to the system state by a master-block of a given height.

We assume that by the start of each round all minters will possess the current state of the system. Upon applying a quilt, minters will obtain a new state. Operating with quilts allows a significant decrease in computation and communication overhead that occurs after the shard consensus subslot.

It can be easily observed that once full nodes begin operating in the quilt mode, the system will become less secure, as the validity of the state change is not verified beyond a shard consensus. The inevitable loss in security is circumvented by the presence of authorized master-nodes that perform fishermen functions.



For a more detailed description of the overall security model, please refer to the next chapter.

Although Dynemix will initially be launched without quilt, the following description of the system accounts quilt's implementation as the final design that the system should obtain. Please note that some of the described functions of the system will not initially work completely in accordance with the description below.

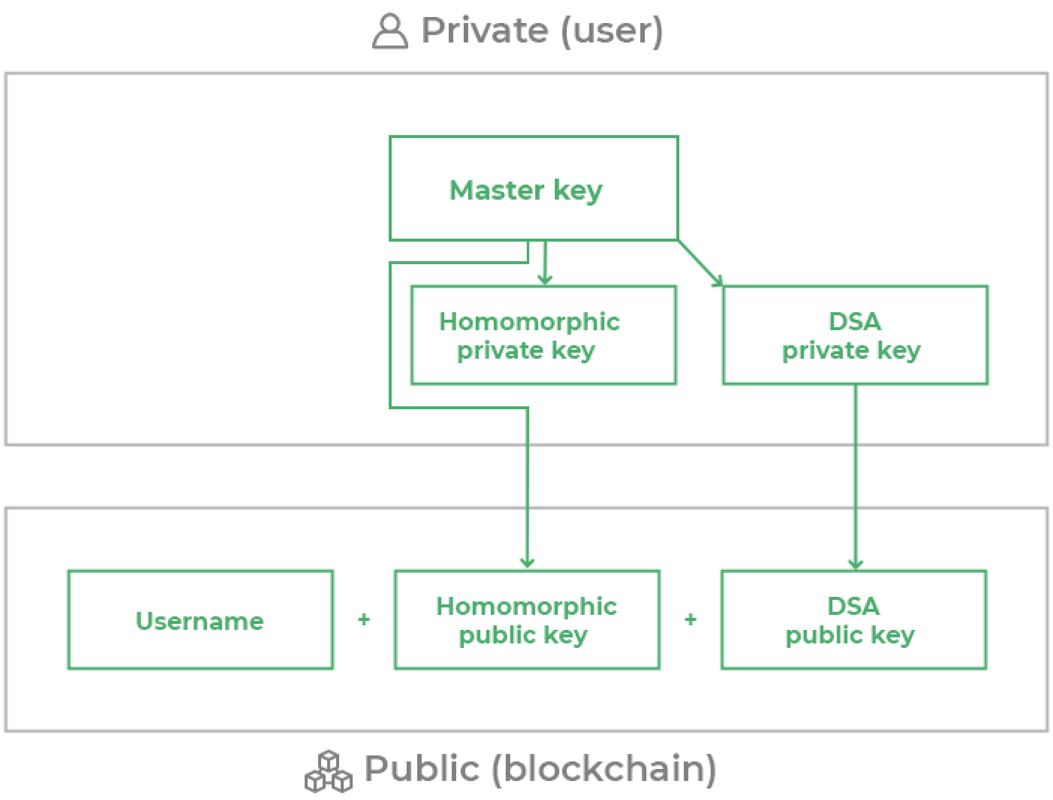
3. Accounts in Dynemix

1) Dynemix Namespace

A user can interact with the Dynemix system on two levels:

- **With the help of public/private key pairs.** This level resembles the approach used in most blockchain systems. Capabilities on this level are limited to issuing transactions.
- **With the help of a registered account.** Most actions within the Dynemix/Liberdyne system require registration.

During the registration process, not only are key pairs generated, but a username is also attached to the public keys. The entirety of all unique usernames chosen by the users during a registration forms *the Dynemix namespace*.



Registration is a special type of a transaction. Once a user registers, it is impossible to alter or delete the username from the blockchain. That prevents any censorship attempts in the namespace service.

Registration is implemented for the following purposes:

a) **Improving user experience**

The username database is stored within the blockchain, which guarantees the possibility of resolving a username to the public key at any time. Hence, any interactions within the system (for example, sending coins) can be performed via a username. This makes the system simpler and easier for users to understand.

The namespace can be also used by other projects within the Dynemix ecosystem or even third-party centralized services. For example, by being combined with integrated decentralized cloud storage, the namespace can allow users to create different types of content that can be accessed by various services, apps or other users via the username.

b) **Statistics generation**

Since the system is completely decentralized, it is difficult to determine even the approximate number of users. The number of public keys with non-zero balances may help, but users can easily generate an infinite number of one-time keys and split their coins among them.

Registering multiple accounts does not make much sense, since users could still employ different public keys as separate wallets but have only one registered account at a time.

c) **Spam protection**

This feature is mostly useful for the Liberdyne messenger, as well as other potential projects based on the Dynemix platform, but it can be also used to counter the transaction spam issue, if such occurs.

Initially, we plan to support free registration without any restrictions (except for CAPTCHA protection from bot registrations). In the case that we face significant trouble with spam and undesired content distribution, it is possible to complicate the registration process to make the Sybil strategy for malicious activities more resource-intensive.

2) Verified namespace

Since Dynemix is a decentralized system and the process of registration is decentralized as well, it is impossible to use standard methods of cyber-squatting resistance.

To solve the problem of cyber-squatting it is reasonable to add a centralized custodial service – the verified namespace.

The verified namespace works similarly to verified accounts in centralized social networks (e.g. Instagram and Facebook). Once a publicly known user wants to get a confirmation mark indicating that a particular account belongs to him, he will provide the requested information and get a verified username. These usernames will form a separate namespace and will have priority over identical usernames from the general namespace in some scenarios.

Verification can be also used to add a legal provable binding between a Dynemix account and a certain entity (a sort of certification), thus allowing entities that wish to use the platform for business purposes to gain more trust from users.

Another critical feature of the service is its ability to restore an account in case of key loss. Account names may be valuable in themselves as a marketing tool, especially for large companies that make significant investments in brand marketing. The risk of irretrievable loss of the account may discourage such companies from using all of the system's capabilities. Custodial account delegation may solve this problem and provide the necessary enforcement of the ownership of the account.

4. Dynemix units – dynes

Dynemix units are called dynes. Dyne is a derived unit of force in the CGS system of units. Currently, the term is barely used, so we will likely be able to give the word a second life and at the same time avoid confusion.

We like the word for its association with power and, besides, it just sounds cool.

Dynemix is not designed to support Turing-complete smart contracts, and hence tokenization will not be supported either. Dynes will be the only units circulating within the Dynemix ecosystem.

Adding certain on-chain derivatives may be considered, however, if the platform's economy requires it during the evolution process. It is difficult to foresee the path of development at the initial stage.

Dynes are utility units, which bear no other function except being a universal medium of exchange and are not backed by any real assets or obligations. Within the platform, dynes can be also used as stakes in the minting process.

The initial issued volume will reach 1,000,000,000,000, or 10^{12}, dynes.

The minimum fractional unit is equal to 0.000001, or 10^{-6}, dynes.

Starting from the second master-block of the blockchain, new dynes will be issued as rewards for system support with each new master-block.

5. Dynemix transactions

1) Transaction types

Account transactions	
Register	Registers a new user account
Register verified	Registers a new verified account
Recover	Allows public keys to be reset with the help of the master key (changing the password or master key)
Coin transfer transaction	
Transfer	Transfers coins between public keys

Service transactions	
Stake/unstake	Stakes or unstakes a specified number of coins for minting
IMIN	Indicates readiness to participate in the minting of the next block
Proof of fraud	Incriminates a minter in an attempt at fraud, slashes the minter's stake and restricts him from further participation
Administrative transactions	
Administrative	Allows special administrative measures to be applied in the case of their adoption

2) Spam protection

Since we follow the concept of free transactions, it raises the problem of spam – instead of sending one transaction with the desired amount (say, 100 dynes), a malicious user could deliberately send 100 transactions of one dyne each, hence creating an unreasonable load on the system. To solve this issue, we adopt several restrictive measures:

a) **Limiting the number of transactions**

The system will accept only one transaction from each address in each block. This limitation will not significantly diminish the user experience, since Dynemix has a 10-second block time, and in very few cases will a user need to make transactions faster than that. If the user needs to make several consecutive transactions, he may send them to the network and wait until they are processed.

Business scenarios often require multiple transactions to be processed from one account at a time, however. To meet the needs of business accounts, Dynemix supports multi-output transactions. To protect the system from spam, fees are charged for each output of such transactions. Unlike most blockchain systems, which feature market-defined commissions, fees in Dynemix are predefined to improve the user experience.

b) **Limiting the transaction amount**

Another antispam measure is limiting the minimum amount of a user-generated transaction to one dyne. In case the total market cap grows high enough, this limit can be lowered to meet user needs.

In case these measures prove insufficient, additional restrictions can be applied, including but not limited to:

a) **Limiting the daily amount of free transactions**

According to the gathered statistics, it is possible to estimate the daily average number of transactions and limit free transactions to a certain number that would satisfy the needs of most users.

b) **Charging monthly fees**

We can set monthly fees for using the system. When a user issues his first transaction in a month, it should contain a predefined service fee, which is obtained by minters. After paying the fee once, the user can freely issue any number of transactions until the prepaid period expires. This approach resembles the conventional banking experience, as users usually pay annual or monthly card issuance and service fees to banks, which is why this approach should be familiar to most users.

6. Financial privacy

Most cryptocurrencies disclose transaction data, i.e. the number of coins sent, the sender and the recipient. Since we are creating a platform for everyday payments, it would be advisable to solve this issue and ensure compliance with banking secrecy. A system that allows anyone to see the current balance and the entire payment history of any user is obviously less attractive to consumers than a system that can assure their financial privacy.

When we started the project, several platforms (Monero, Zcash, Dash etc.) offered solutions that allowed the amount and/or the sender and/or the recipient of a transaction to be hidden.

The problem was that all those platforms were UTXO-based blockchains, while Dynemix was meant to become an account-based system (a UTXO model requires higher traffic and storage overhead, which is intolerable for a system that is meant to scale up to thousands of TPS). Besides, the existing solutions drastically increased the transaction weight, RAM use or computation time.

No balance-hiding solution for account-based blockchains existed at the time, which is why a completely new approach was required.

Financial privacy in relation to blockchain consists of two conditions, each of which requires a separate technical solution.

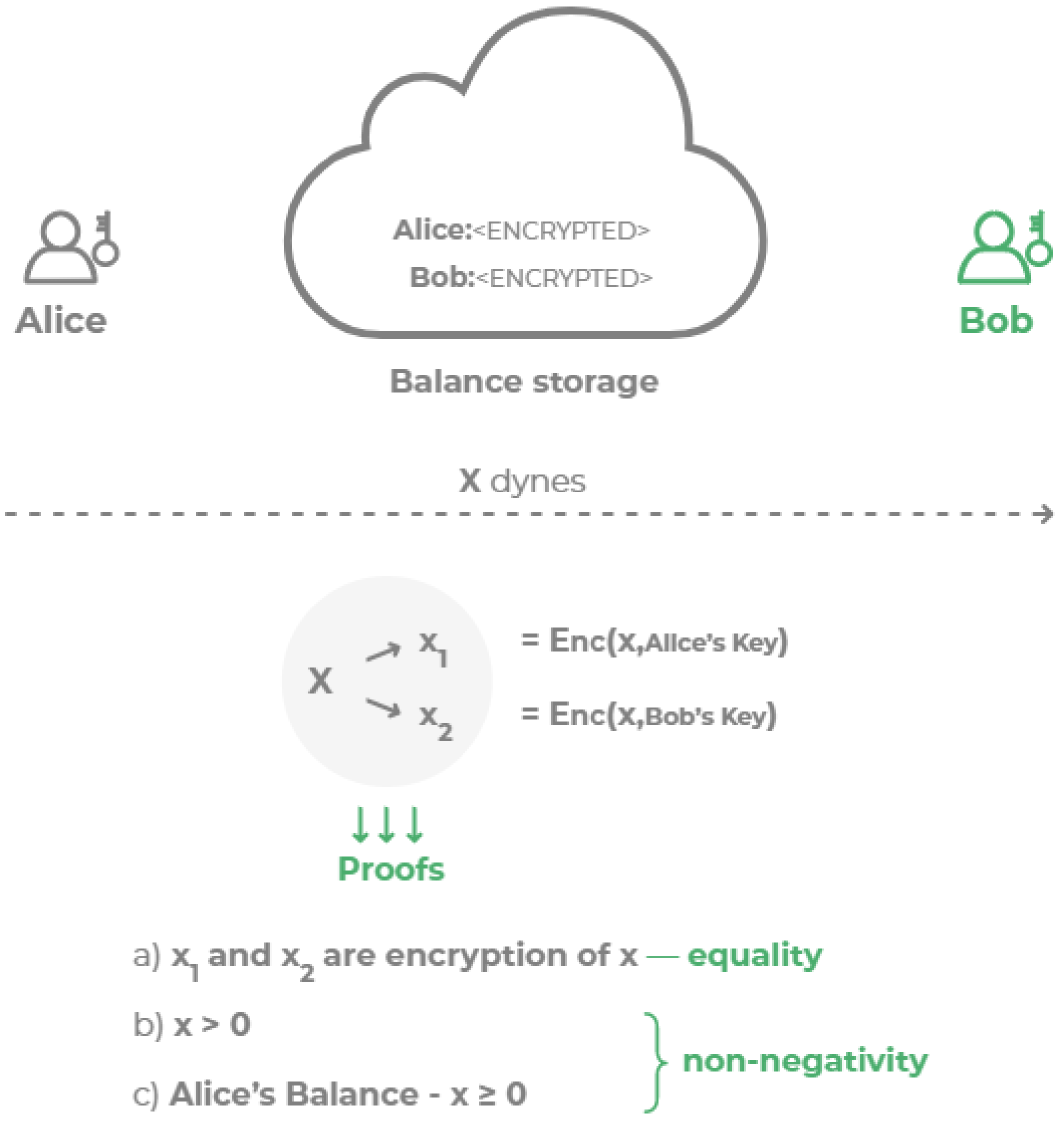
1) Hiding transactions' balances

We believe that this condition is of the utmost importance – hiding transactions' amounts allows for the hiding of account balances, which are sensitive information that the majority of users would not want to be disclosed.

To solve this issue in an account-based system, we can use homomorphic encryption, as it allows computations on the ciphertext, generating an encrypted result which, when decrypted, matches the result of the operations as if they had been performed on the plaintext.

Instead of storing balances and transaction amounts as plaintexts, we can store encryption keys and encrypted balances.

Assume Alice wants to send x dynes to Bob. She also wants her balance and the transaction amount to be hidden from Eve. As our blockchain is public, the only way to achieve this is to introduce encryption.



Instead of subtracting x from A and adding x to B, roughly speaking, we subtract encrypted x from encrypted A and add encrypted x to encrypted B.

Of course, Alice and Bob have different encryption/decryption keys, which is why x should be encrypted twice (with Alice's key and with Bob's key) to be able to be subtracted from Alice's balance and added to Bob's balance.

Then the following problem arises – Alice should prove that:

- both ciphertexts encode the same value;
- this value is positive;
- her new balance is at least non-negative.

To achieve this, we need (non-interactive) zero-knowledge equality and range proofs.

After a minter receives the transaction, he checks the validity of the attached proofs and performs the subtraction of x_A from Alice's balance and the addition of x_B to Bob's balance.

If Alice wishes to perform a transaction with the disclosed amount, she can easily do so. This may be necessary if Alice wishes to leave evidence of a transfer of a certain amount to Bob in the blockchain.

2) Hiding recipients' IDs

This condition is also important, because disclosing the recipient of the transaction allows for the identification of connections between users, but we suppose that this is not as crucial for most consumers as is balance hiding.

Since Dynemix supports multi-output transactions and given homomorphic encryption implementation, the solution to this issue is trivial – instead of sending a transaction with one output (Bob's account), Alice sends a transaction with multiple outputs, one of which is Bob's account and the others are randomly chosen registered accounts. The amount transferred to random accounts is equal to zero, while the amount transferred to Bob is the actual desired sum.

From Eve's point of view, Alice sends an unknown number of coins to multiple recipients, and Eve cannot conclude that either of them are real recipients rather than just random extras, nor how many coins each of them received.

Alice can arbitrarily choose the number of extra outputs, assuming that the more outputs she includes, the more she confuses Eve, but at the same the more she must pay (as fees are charged by minters for each additional output's processing).

Alice can also issue multiple transactions with multiple outputs, thus hiding the mere moment of the actual transaction's issuance. The degree of obfuscation depends only on the number of dynes Alice is ready to spend as fees for extra outputs.

A significant advantage of this approach is its high customizability on the client level. Since no patterns are predefined by the protocol, it complicates analysis attempts for a potential adversary.

Interaction obfuscation turns out to be a paid feature, but it is justified by the excessive load put on the system by such transactions. We suppose that balance encryption will be sufficient for most users and that multi-outputs will be used infrequently.

3) Problems of proposed solution.

Before homomorphic encryption can be implemented, we must solve several major problems that arise with the proposed solution.

a) **Constructing suitable ZK-proofs**

Although homomorphic encryption theoretically can allow our standards to be met, as for a single account only an encrypted balance and public keys must be stored on-chain, thus saving a lot of space, the problem lies in the construction of the required ZK-proofs.

The only currently available working solution is an additive modification of the El-Gamal encryption scheme, which requires calculating a discrete log in order to decrypt a balance. This task is NP-intermediate, and thus the solution is completely impractical. At the same time, constructing ZK-proofs for much more convenient cryptosystems (e.g. Paillier) could require a lot of time and resources for proving and verifying, and the proofs could be much larger than expected.

We do not find it reasonable to use any solution that cannot allow a transaction to be quickly constructed even on a mobile device, which is why currently available concepts seem inappropriate. To develop a practical implementation that can meet our needs, further research is required. We are developing a system with a foundation for the quick and painless implementation of homomorphic encryption (the architecture is initially optimized for operation with encrypted balances) when a suitable solution is fully developed. We cannot reliably claim when this may happen, however.

We can note that homomorphic encryption and ZK-proofs as technologies are still in their infancy and improved solutions have yet to be developed. We should also note that the technology is relevant not only for blockchain systems but to various spheres of the computer industry, which raises our expectations for the potential rate of progress.

For example, hardware giants announced RAM-encryption technologies that can provide more security to cloud computing, but the currently presented solutions still feature computations with raw data on the CPU layer. Building a fully secure VM would require fully homomorphic encryption and computations directly on the encrypted data. For this reason, we can expect low-level optimizations that will help us build a more scalable system.

b) **The high risk of hidden attacks**

The encryption of all account balances obstructs the detection of attacks.

If the adversary exploits a bug that allows the creation of new coins in an arbitrary amount, it can have a devastating effect on the system. When all balances are publicly observed, it will likely be detected quickly, and countermeasures will be applied (a rollback, in the worst case). When balances are encrypted, however, it may stay unnoticed longer, which is an unacceptable outcome.

Although the state transition remains verifiable and even in the presence of master-nodes, which are expected to help keep the system secure and ensure the sufficient level of security, we suppose that it would be dangerous to implement the described technology on the platform scale until it was redundantly researched and tested.

Any possible additional measures to mitigate the stated risks should be also considered, even if they come at a cost.

7. Dynemix state transition

As stated, a cryptocurrency platform that can meet all of the requirements to compete with centralized payment systems cannot be based on any existing protocol for reasons of technical limitations. We had to design our system from scratch and apply many unique features to achieve our goal.

More details will be available in the technical documentation. Herein, the protocol will be described in broad strokes, and its most important features will be explained in the next chapter.

1) Stakes

a) **Staking**

Everything starts with stakes. Any registered user can send a stake transaction with an arbitrary amount, thus indicating his intention to participate in block production.

As we intend to democratize block production and attain a high level of decentralization, we cannot set a high minimal stake threshold. Requirements that are too low, however, can lead to abuse and negatively affect security, which is why a point of balance should be determined.

A stake transaction's amount is publicly visible; hence, minter candidates have to partially disclose their balance to the extent of the stake amount.

b) **Restaking**

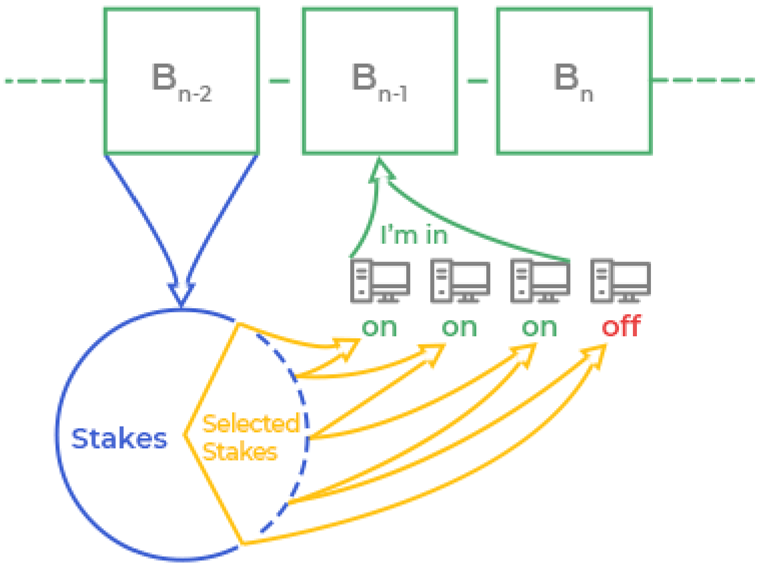
Since we initially assume that minting will be performed not only by users who adhere to a professional approach but also to a substantial extent by common users with consumer-level hardware, we cannot predict the percentage of stakeholders with 100% uptime. Although we implemented an adaptive algorithm for minter window selection that accounts for an average proportion of the available stakeholders, if the dispersion grows too high, it can have a negative effect on several properties of the system.

If we encounter such a problem, it may be solved by setting a TTL for stake transactions or suspending the stake if the stakeholder skips a block. In this case, stakeholders will need to restake intermittently to indicate their availability.

c) **Unstaking**

If the user wishes to unblock the staked dynes to be able to spend them, he sends an unstake transaction. The staked amount becomes spendable after two blocks are minted above the block that contains the unstake transaction, which makes Dynemix more minter-friendly than other PoS systems, as the stake in the latter is typically blocked for a long period of time for double-spend protection.

2) I'm in



Block B_{n-2}

In a pseudorandom manner with the master-block's hash as a random oracle, a set of eligible minter candidates for block B_n is selected from the array of accounts with active stakes. The stake size correlates with the probability of being selected for minting, as the weighted sampling algorithm is employed.

Block B_{n-1}

Stakeholders picked by the algorithm send special IMIN transactions to the minters of block B_{n-1} to indicate that they are ready for minting. This is required to exclude inactive stakeholders and ensure that during the minting of block B_n most of the assigned minters within each shard will be available.

IMIN transactions provide a solution to the problem of unavailable stakeholder nodes, which are considered faulty in terms of the consensus algorithm.

Since we are assuming that a substantial proportion of stakeholders will consist of ordinary users whose nodes may be unavailable much of the time, there is a serious risk that liveness may be threatened by a large number of inherently faulty nodes. This is why an algorithm that verifies availability immediately before the minters are designated is required. The solution is detailed in the next chapter.

3) Sharding

Block B_{n-2}

According to the average number of transactions per block in a recent period, the optimal number of shards that the system will split into for minting block B_n is determined.

Block B_{n-1}

According to IMIN transactions, a minters' committee for block B_n is formed.

Block B_n

The minters' committee is algorithmically sorted by shards and split into a set of shard committees.

Since all information required for sharding is contained in blocks B_{n-2} and B_{n-1}, the shard committee's designation for block B_n becomes consistent throughout the network after block B_{n-1} is propagated.

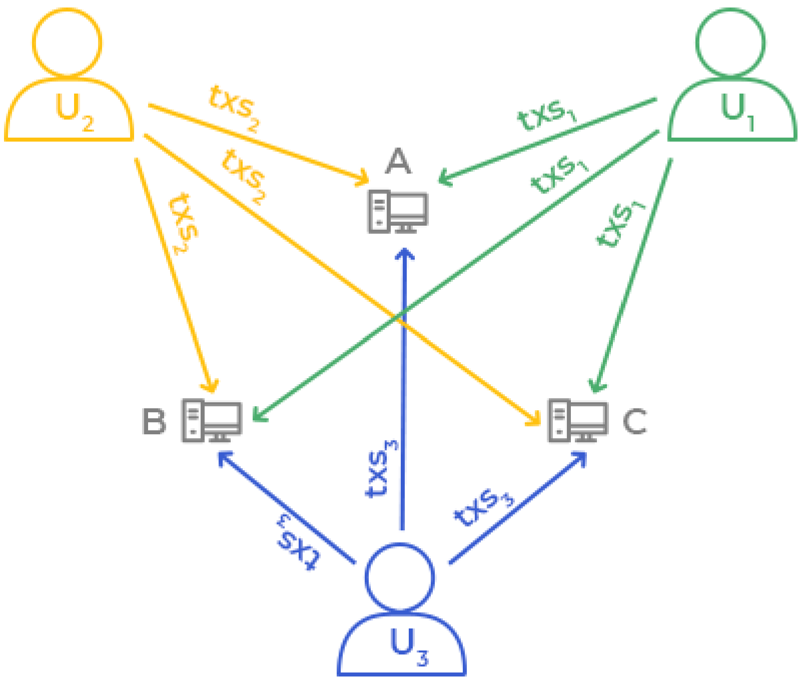
If the system splits into s shards and i IMIN transactions were recorded in block B_{n-1}, then each shard committee consists of $\frac{i}{s} = m$ minters. The size of the window of stakeholders selected in block B_{n-2} is algorithmically adjusted to maintain the required number of shards with constant m. In Dynemix, $m = 10$ (although it may be subject to adjustments if needed).

Determining m is a security/overhead trade-off. The more m we set, the more traffic overhead each minter node will suffer, which makes sharding less efficient, but at the same time, the more security will be provided, as the chance of controlling the supermajority in the shard for the adversary statistically decreases.

4) Collecting transactions

After the shard committee is appointed, minters collect incoming transactions.

Transactions are assigned to particular shards according to the public keys of the issuers (in the case of registration transactions, according to the username). Upon receiving block B_{n-1}, each node in the network knows to what shard committee any given transaction in the pool is assigned.



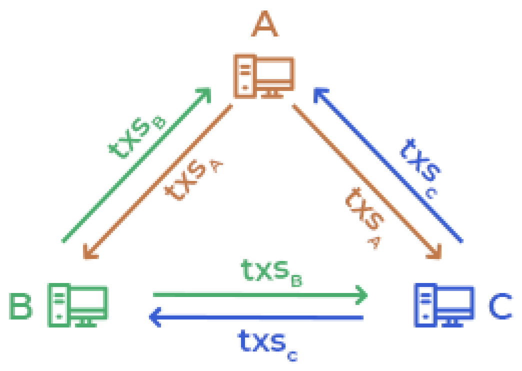
To ensure fast verification, transaction issuers or other nodes that store transactions should send them directly to all minters of the corresponding shard. If the nodes behave accordingly, under perfect conditions all minters of the shard committee should have the same set of transactions by the end of the subslot.

Still, even if the transaction senders try to multicast transactions to all assigned minters, due to the network delays and imperfect time synchronization, the sets may partially vary. To ensure the consistency of the transaction set, minters synchronize the collected data.

5) Synchronizing the transaction set

During this phase, each minter multicasts their collected data to other members of the shard committee.

Each minter signs the hash of the collected transaction set and submits the set to other members of the shard. By the end of the time slot, the transaction set should become consistent within the shard.



Minters may behave Byzantine and send different versions of the transaction sets to other members of the committee or submit no data at all. This situation is managed in the next phase.

6) Verifying each other's transaction commitments

To reach a Byzantine agreement, honest nodes should possess identical transaction sets. Since Byzantine actors in the previous phase can act inconsistently, an additional round of data exchange is required.

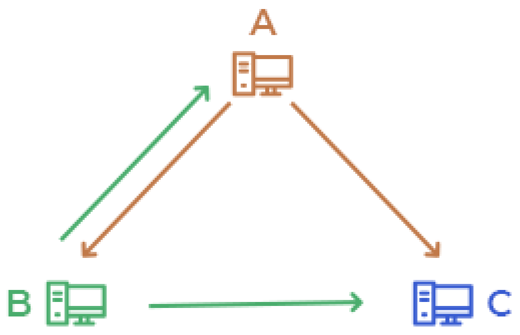
During this subslot, minters mutually exchange hashes of all the transaction sets received from other minters to verify that each member of the committee has the same total set available and that no one is trying to commit fraud.

If all minters are honest and non-faulty, each will get the same number of identical hashes, which means that the transaction set is consistent within the shard and that minters can proceed to the next phase. If there are Byzantine actors among the minters, however, additional actions are required.

Assume a simplified model wherein we have a shard committee of three minters. Alice and Bob are honest, Chuck is Byzantine, and consensus is reached by $\frac{2}{3}m$.

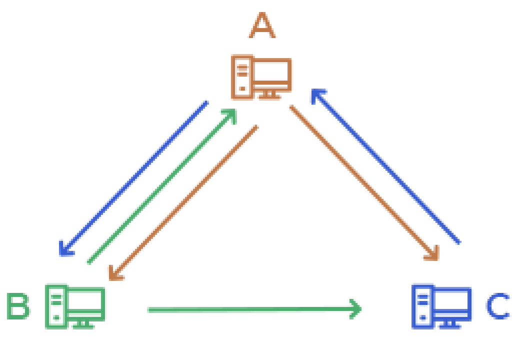
a) **Chuck commits nothing**

Alice and Bob exchange messages stating that Chuck has not committed anything. Since they exchanged their transaction sets in the previous phase, they can proceed to the next phase.



b) **Chuck commits a set to Alice and nothing to Bob**

Bob receives a hash of Chuck's set hC from Alice, and since he did not receive the set from Chuck directly during the previous phase, he requests Chuck's set from Alice. After Bob receives it, the transaction sets become consistent, at least between Alice and Bob.

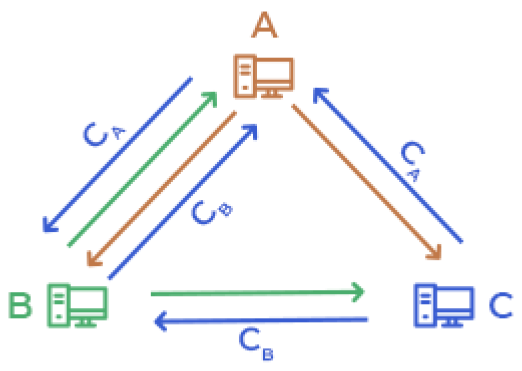


Bob may suspect that Chuck is an adversary, but since Chuck's malicious intentions are not provable, as he may simply have experienced connection problems in the previous phase, Bob does not apply any punitive measures.

Alice may suspect that Chuck is an adversary, but from the other side, Bob could also be an adversary trying to frame Chuck. Alice cannot reliably determine whether each of the assumptions is true, and she does not apply any punitive measures either.

c) **Chuck commits sets to Alice and Bob, but the sets are not identical**

Bob receives a hash of Chuck's set hC_A from Alice and finds that it differs from the hash of Chuck's set hC_B that he has received from Chuck. Bob requests Chuck's set C_A from Alice, and Alice, having come to the same conclusions, requests C_B from Bob. After they exchange Chuck's sets, the transactions sets become consistent between them.



Now, as Alice and Bob both know that Chuck is the adversary, each of them creates a special proof-of-fraud transaction, accusing Chuck of fraud, with both signed hashes of Chuck's sets (hC_A and hC_B) attached as a proof, and adds it to the transaction set. If the shard-block proposed by Alice and Bob wins the consensus round, Chuck's stake is slashed.

The simplified model described does not take into account the possibility of multiple minters' being controlled by the adversary, who can abuse the situation described in paragraph **b** by sending messages to some minters too slowly, thus keeping the transaction set inconsistent by the end of the subslot.

To counter this opportunity and prevent the adversary from delaying the consensus, another rule is adopted – minters request the set only if it is known by at least $f + 1$ other participants by the beginning of the subslot. Otherwise, the set is ignored by all committee members.

7) Shard-blocks and the shard consensus round

After all minters within the shard synchronize the known transactions, each of them processes transactions and builds a shard-block that contains the whole synchronized set.

Minters exchange signed hashes of the shard-blocks to reach a consensus. Dynemix uses a synchronous authenticated multivalued Byzantine agreement model. If the required threshold of replicas proposes identical shard-blocks within the time subslot, this block is considered approved by the shard committee.

Minters can propose only one shard-block during this phase, which is why they should build the same block as most other minters probably will; otherwise, the consensus will not be reached, and nobody will receive a reward. The optimal strategy to reach consensus is including all known transactions in the shard-block.

If a minter proposes two different shard-blocks, he is accused of fraud by the honest minters and his stake is slashed.

In Dynemix, the shard consensus threshold is set to $> \frac{2}{3}m$.

As a Byzantine agreement protocol used in Dynemix can tolerate f faulty players of $n > 2f$ participants in a (weakly) synchronous communication setting, the required majority value c in a committee of m minters can be set within $\frac{m}{2} < c \leq m$. Setting c is a safety/liveness tradeoff. The more c we set, the more resources the adversary must possess to approve a deviating shard-block or fork the blockchain if synchrony is not held, but, at the same time, the fewer faulty minters can be tolerated. We have chosen $c > \frac{2}{3}m$ for shard consensus as an estimated point of balance. This value can be changed within the stated boundaries if needed.

8) Shard-block exchange

After consensus is reached in the shard, minters establish connections with other shard committees and mutually exchange patches of shard-blocks (before the quilt layer is implemented, minters will exchange complete shard-blocks) with them. After the exchange is complete, all external minters request hashes of the shard-block from all shard committee members to ensure that nobody behaved in a Byzantine way.

If Byzantine behavior was detected (e.g. somebody signed different versions of a shard-block), honest minters issue a proof-of-fraud transaction in the next block to penalize the malicious actors.

9) Master-block and master consensus round

After each minter has collected all patches (or complete shard-blocks), each builds a quilt and constructs a master-block. Minters exchange hashes of the master-block to reach the second BA agreement.

The master consensus threshold in a minter committee of M members is set to $\frac{3}{5}M$.

IV. Dynemix Explained: Features And Tradeoffs

Now that we have briefly described the architecture of Dynemix, we should explain why we implemented the mentioned solutions and how they help us reach the stated goal of building a next-generation blockchain payment system, as well as what tradeoffs we must accept on the way to this goal.

1. Setting

Dynemix was developed to operate in a specific setting, which should be outlined to clarify certain features of the protocol.

In the beginning of the white paper, we stated a set of properties that we strive to obtain. One of the essential features is a high level of decentralization, which requires the following condition:

- **The system should be designed to let the maximum number of users be involved in the system's support processes**

We intend to go back to basics and design a system that can be operated by common users with the help of home-class hardware, which corresponds to the original ideas of Satoshi that were embedded, but not embodied, in Bitcoin.

1) Honest nodes

The first difficulty we face is the need to revise the notion of an honest node.

In a conventional setting, honest nodes not only refrain from doing anything that goes against the rules but also refrain from not doing something that is expected of them. In relation to many BFT-style blockchain protocols, this refers to being constantly available at least for the nodes known as current stakeholders (or any other types of nodes that are expected to participate in a state transition). In the case that an average number of stakeholders who are offline exceeds the threshold, such protocols will eventually lose their liveness.

The developers of Algorand relaxed these requirements and introduced the notion of "lazy honesty," which refers to users who honestly follow all their prescribed instructions when they participate in the protocol but are asked to participate to the protocol only very rarely (e.g. once a month) and with advance notice.

This notion does not fit our setting either. If we assume that users can participate in minting with home PCs, we cannot expect them to hold any particular uptime – users can arbitrarily go online and offline at any moment, regardless of being selected for minting in any particular round, which is why we need to relax the boundaries of availability even further and operate with the notion of *irresponsible honesty*.

Honest but irresponsible nodes are actors who do not commit any actions that deviate from the protocol but who can stop participating at any arbitrary moment and hold any arbitrarily long (up to infinite) period of unavailability.

The described notion seems to have a lot in common with *the sleepy model of consensus* researched by Pass and Shi. Their model is abstract, however, and does not allow to easily derive particular practical conditions for which we are designing the protocol. To avoid confusion, we use a different term, which also places more emphasis on the partially subjective nature of the problem.

Although the notion of irresponsible honesty seems to fit our setting, to provide a more precise description of the targeted practical environment, we should further refine the model and introduce the notion of *weakly responsible honest nodes*.

In terms of Dynemix, weakly responsible nodes are actors who can arbitrarily refrain from participating even if asked to participate, but who, once accepted, hold some expected uptime, during which they perform all of their instructions. We assume that the more time passes since the node committed its consent to participate, the more likely it is to lose availability.

What does this all mean in practice? We assume that the vast majority of users will use the Liberdyne messenger (or any other third-party client that operates in approximately the same manner) as a client for interacting with the Dynemix network. Once a user obtains enough dynes to place a minimal stake, and his hardware satisfies the requirements, the client engages in minting.

After the stake is placed, we do not expect an average user to hold constant availability. It is obvious that, given that the client is run at home, the user can simply close the app or shut down the PC at any time. Under such assumptions, it is easy to conclude that practically any arbitrary uptime/downtime proportion is possible, which makes it look like the model of *irresponsible honesty* (as described above) or *the sleepy model* (as defined by Pass and Shi).

We also rely on another assumption, however, which brings us to the model of *weak responsibility*. Despite the user's ability to go offline at any time normally, in case he is selected to participate while being online, we expect the user to prevent losing availability and refrain from quitting by choice. In practice, this means that once the client figures out who has the right to participate in the creation of the next block, it alerts the user and prevents any kinds of soft shutdown until all the duties are carried out. The user is incentivized to wait, which makes us rely on his not closing the client until the time slot passes. At the same time, the more time passes, the less the user may have the opportunity to wait, if the need to shut down is urgent. Of course, this does not refer to situations when connection is lost due to a network or hardware malfunction, but this is expected to occur very rarely.

According to the described model, we expect the following distribution of participants:

- **Weakly responsible nodes** – the majority of participants. This category is represented by low-stakes users, who will likely engage in minting via their home/office hardware. The presence of such actors will greatly benefit decentralization.
- **Responsible nodes** – a minority of participants. Stakeholders who have placed stakes large enough to secure frequent participation will likely hold constant uptime. At a certain point, it will become economically feasible to run a node in a data center to reduce omissions.
- **Irresponsible nodes** – the smallest minority of participants: those who lose connection during participation. We assume that a small share of such nodes will be also present, as faults are inevitably expected to occur.

This model is inherently easily managed by Nakamoto-style protocols; it is not so trivial, however, to cope with it for a BFT-style linearly consistent protocol. To conform to the described model, the following measures were applied:

a) **IMIN transactions**

After a stakeholder is chosen to participate in block creation, he is expected to confirm his consent by issuing a special IMIN transaction directly prior to the process start. This measure circumvents each participant's initial lack of awareness of each other online status. After observing a set of IMINs in block B_{n-1}, all participants come to a consistent view on the set of the selected stakeholders who are actually available at the time, which prevents offline nodes from affecting liveness.

We assume that there is always a known average proportion of available and ready stakeholders with only some insignificant deviation. If the statistically determined proportion does not hold within tolerable boundaries of variation, it may negatively affect system throughput; i.e. if the number of the collected IMIN transactions is much lower than expected, the system will split into a smaller number of shards. In most cases this will have no consequences; if the system is running at the capacity limit, however, some transactions may not fit in and will be left in the pool.

IMINs help maintain liveness in the presence of weakly responsible nodes; the question of safety, however, is ambiguous.

On one hand, we assume that the adversary keeps his nodes always available and ready to fulfil his sinister plots, which is why non-responsible nodes who refuse to participate despite being chosen contribute to a higher relative number of adversarial nodes in the minter committee.

On the other hand, providing an opportunity to behave in a weakly responsible manner without any penalties contributes to a larger global stake pool size, which in turn decreases the relative share of the adversary and reduces his chance of being selected for minting. Furthermore, lowering the participation threshold boosts the efficiency of the economic model of Dynemix.

It is hard to predict which of these features will outbalance the other one in practice.

Although IMINs provide a solution to the problem of weak responsibility and assure stronger liveness guarantees, at the same time they can negatively affect liveness from the other side. In case no IMIN transactions are added to block B_{n-1}, a minters' committee for block B_n cannot be formed, which terminates the protocol's execution.

Minters are incentivized to process IMIN transactions (minters of block B_n distribute rewards to minters of block B_{n-1}, which is why minters of the current block are always interested in the creation of the next block), hence, considering the redundant number of minters for each block, the mentioned situation is extremely unlikely to happen. If this issue is considered a feasible threat, it is possible to implement an algorithm to resolve it.

It is important to emphasize that IMINs as a solution for weak responsibility can only be implemented into a protocol that can guarantee a fair block proposal, which is, in relation to the issue at hand, adding all known IMIN transactions into the next block regardless of the proposer's preferences. If the protocol uses a conventional block-proposal algorithm, a malicious actor can simply censor the IMIN transactions of other stakeholders, thus permanently seizing control over the system. In Dynemix, this issue is resolved by our novel consensus solution called Guess My Block game, which is described below.

b) **Constant reshuffling of minters**

We assume that after a minter issues an IMIN transaction, he is expected to reliably hold availability only for a short period. For this reason, he is obliged only to participate in the creation of a single block and the subsequent reward distribution phase.

c) **No leaders or other single points of failure**

The Dynemix protocol is completely leaderless. If we assume the presence of irresponsible nodes, it is obvious that leaders can also behave irresponsibly. If we apply a view-change procedure in case a single shard leader fails, it will either lower the performance of the entire system or simply prevent consensus in the shard, which opens up an opportunity for an attack on liveness or fairness.

d) **More important actions first**

We assume that the more time that has passed since a minter issued an IMIN transaction, the less we can rely on his availability. As the protocol execution advances within each round, this fact is taken into consideration.

2) Time

Dynemix operates in a setting with *synchronized clocks* adjusted to *the global time*. The time is divided into discrete slots of 10 seconds (this is subject to adjustment if required), which each correspond to a block created during that time slot. A time slot is split into a number of subslots corresponding to the phase of the protocol.

We assume that all nodes, once connected to the system, synchronize clocks and have an approximately consistent view on the current protocol execution phase. Nodes are allowed to experience minor discrepancy in their local views on the current time, given that this discrepancy is insignificant on the scale of the determined subslots.

Known time synchronization algorithms for peer networks can be classified into two categories:

a) **Symmetric (peer-to-peer)**

In symmetric protocols, each participant has equal influence on the system time. This approach can be used for blockchains mostly in a limited-view manner, which corresponds to communication overlay of various peer networks, including blockchains.

Intuitively, the symmetric approach seems to be the most relevant to a blockchain system, as it satisfies the principles of decentralization; it also, however, has several shortcomings.

The first one is its highly questionable Sybil resilience. The attacker can spread his infected view over the system, thus massively breaking consistency, or can perform an eclipse attack targeting particular nodes. Although there are several proposed techniques for the Byzantine setting, they may not be robust enough when the liveness and consistency of a blockchain is at stake.

Another problem is scalability. We assume that peer nodes cannot have a complete view of the system and are limited to a local view, which consists of other nodes with which the node can interact. As the system scales up, the relative size of the local view of each node shrinks, which makes the protocol more vulnerable, especially in the presence of a powerful adversary.

b) **Asymmetric (client-server)**

The mentioned shortcomings of symmetric design can be mitigated by moving to an asymmetric scheme. In asymmetric design, specially appointed nodes serve as master-servers for the rest of the network (or a more complex multi-level hierarchy can be used), which allows the number of time-setting servers to be limited as desired.

Asymmetry solves the problem of scalability, but providing more resilience still requires additional measures. The obvious solution would be binding a resource to the time-setting powers of a time server, which can significantly hamper a Sybil attack without centralizing the process. In relation to Dynemix, this means creating Proof-of-Stake time servers. This will likely require a sophisticated solution, however, which would require additional research and obviously would not contribute to better overall performance.

Having assessed all pros and cons, we conclude that currently the most optimal solution is simply to use public NTP infrastructure for time synchronization. This solution can reasonably be criticized for relying on an external source, which does not seem appropriate for a truly decentralized system; it provides certain benefits, however, that make this decision much less controversial:

- NTP is the most popular time synchronization solution to date. It has existed for decades and tested against various attack vectors. Given that scalable time-sync protocols for blockchains (and peer networks with Byzantine actors in general) are still in their infancy, it may be reasonable to turn to a more reliable solution, at least until a robust decentralized protocol is designed and tested.

- Engaging NTP will bind the system to the external infrastructure to a certain extent, which can provide more robustness. Given that time sync will be left out of bounds of the blockchain protocol, it will not be possible to distinguish the particular fraction of NTP infrastructure that is used by the system. For this reason, the adversary will not be able to aim specifically at Dynemix, but instead will have to attack the entire NTP infrastructure, which is used by a huge array of various systems.

3) Communication

We use the Δ-bounded model of communication. Whenever each node sends a message, this message is received by a recipient node within Δ delay. All messages are cryptographically signed, which ensures authenticated communication based on the global PKI setup.

The protocol does not require constant strong synchrony and makes progress as long a threshold of designated replicas hold a bounded delay Δ during each round of execution. At the same time, nodes that temporarily lose connectivity are allowed to rejoin the protocol and restore consistency.

Essentially, we adopt the model of *weak synchrony* as described by Guo, Pass and Shi. We consider this model the most realistic description of the practical setting that we assume.

4) Lock-step and responsiveness

Dynemix is designed to operate in a *lock-step* execution manner, which means that the protocol makes progress in strictly predetermined time intervals, which are set according to the estimated communication latency, bandwidth and computation delay assumptions of the peer nodes. It may seem much better to achieve *responsiveness*, however. The protocol is considered responsive if the progress is made according to the actual communication and computation delays of the participants, independent of the expected upper latency bounds.

Responsiveness is thought to be a feature of asynchronous protocols. One can, however, envision a sophisticated hybrid solution that could bring responsiveness into the synchronous model as well. For example, in Thunderella (and several other solutions), the mentioned property is achieved by packing an asynchronous algorithm into an underlying synchronous algorithm, thus achieving responsiveness on the fast optimistic path and keeping the minority corruption tolerance on the slow fall-back path, which is engaged once the optimistic conditions are not met.

As we stated in the beginning, fast finality is one of the essential features of a decentralized payment system, which raises a question: can we upgrade Dynemix to achieve responsiveness? Since we are designing our protocol from scratch, we are not bound by any particular models or restrictions, and it looks tempting to apply some tricky solution that can help speed up the system.

As an answer to that question, we would say rather more "no" than "yes." There are several fundamental factors that deprive us of such an opportunity.

a) **All replicas need to wait for the end of the time slot to make sure that everyone had a chance to commit.**

If we allow the protocol to proceed after obtaining a threshold of signatures at the shard consensus stage, slower minters will be deprived of the opportunity to participate and receive a reward. Such a situation will give an unfair advantage to those who run nodes in fast data centers on powerful hardware and will disrupt the system of essential economic incentives of the protocol.

If we tighten the optimistic conditions from obtaining a threshold to obtaining all signatures, given the total number of participants, it is highly unlikely that not a single one of them would experience a communication fault or simply behave unresponsively, which means that such optimistic conditions will happen so rarely that it barely makes sense.

A similar situation occurs at the master consensus stage. We need to assemble the most complete set of shard-blocks, which means that we can either proceed only after all expected shard-blocks are propagated or wait until the respective subslot expires.

Moreover, even after obtaining all signatures or collecting all shard-blocks, it is still preferable to wait until the end of the time slot to assure the opportunity to present a proof of fraud in case adversarial behavior is detected. From that perspective, responsiveness makes the system less secure.

b) **Time slots should be consistent among all nodes, including those who currently do not participate in the consensus.**

Most BFT protocols separate the participants of the state transition procedure from the external nodes. For example, in PBFT, an external node (denoted as a client) sends an input value to one of the participants (denoted as the leader), who then reaches a consensus with the other participants (denoted as backup replicas) and provides the agreed output value to the client.

In variations adapted for blockchains, the leader assembles a block from transactions that he somehow obtained beforehand, drives a consensus with other replicas and propagates the resulting output-block through the network. Commonly, input values are propagated through the network via a gossip protocol, which ensures that each time a node is selected as a leader, it possesses a large chunk of currently pending transactions. A node that issues a transaction does not need to send it to particular nodes at a particular time – gossip assures that the transaction will get to a leader with an expected modest delay with high probability.

> Dynemix, on the other hand, features a completely different input value-submission algorithm. A critical condition for it to work properly is that the client submits transactions to a predefined set of replicas during the prescribed time slot. All nodes in the system (including both clients and replicas) should have a consistent view on these subslots for each round of protocol execution, which contravenes the notion of responsiveness.
>
> This algorithm is implemented to achieve the crucial property of conventional centralized payment systems: impartial instant processing of all transactions that were sent to the system, which is undoubtedly a higher priority than achieving responsiveness.

2. Censorship and transaction fees

In the beginning of the white paper, we mentioned two important features of a decentralized payment system that we intend to achieve:

- **The system should instantly process all transactions that users send to the network.**
- **Transactions should be processed without fees.**

These features will help raise the level of user experience to the level of existing conventional payment systems, which is an essential condition that can allow a blockchain platform to compete with those systems on equal footing.

Most current generation blockchains (whether of a PoS or PoW design) rely on a leader who proposes block candidates. Other nodes vote for the proposed block and the network either accepts or rejects it.

According to this approach, the leader assembles the block at his own discretion and is free to reject any transactions. He may consider the proposed fee insufficient or may simply have personal feelings about the transaction issuer. For this reason, users must compete for the leader's favor (in most cases by offering a higher processing fee), which severely deteriorates the user experience compared to conventional payment systems, as they process all transactions without any preference. Furthermore, this approach creates the preconditions for censorship.

Such an approach is especially dangerous for Dynemix, since we have implemented a solution for maintaining liveness in the presence of weakly responsible nodes (the described above IMINs) that strongly relies on impartial block proposal. The ability to censor transactions can lead to the capture of the system by the adversary.

Although there is a number of projects that addressed the problem of censorship (for example, Honey Badger introduced multi valued transaction set proposal), to date no one managed to get close to the stated parameters.

As we intend to process transactions for free, it complicates the task even further – why would the leader put any particular transaction into the proposed block, if he has no incentives for doing so? This issue may seem insoluble within the currently available concepts, and to date, we have seen no solution that can help achieve both stated features assuming the non-altruistic behavior of participants. That is why we had to develop a new design of a block proposal algorithm and introduce a completely different approach.

3. Guess My Block game

1) Brief description

Our solution to the problem is offering minters to play a game with incomplete information, in which the optimal winning strategy implies adding all known transactions to the proposed block and the resulting consensus between players is reached via a synchronous authenticated multivalued Byzantine agreement.

In Dynemix, no leaders are appointed in shard committees. All minters propose their own shard-blocks on equal footing and have equal votes during the multivalued consensus round. The voting is carried out not by accepting or rejecting a shard-block proposed by the leader, but by proposing identical or non-identical shard-blocks by players.

✓ ***If the majority (supermajority) required to reach a Byzantine agreement propose identical shard-blocks, their blocks win and all minters who proposed those blocks receive a reward. The minters who proposed different versions of the shard-block or proposed nothing are not rewarded.***

✗ ***If identical shard-blocks are proposed by an insufficient number of minters, a consensus is not reached, the shard-block is skipped, and no one receives a reward.***

If someone proposes multiple different versions of a shard-block (behaves Byzantine), his vote is not counted by honest minters and his stake is slashed.

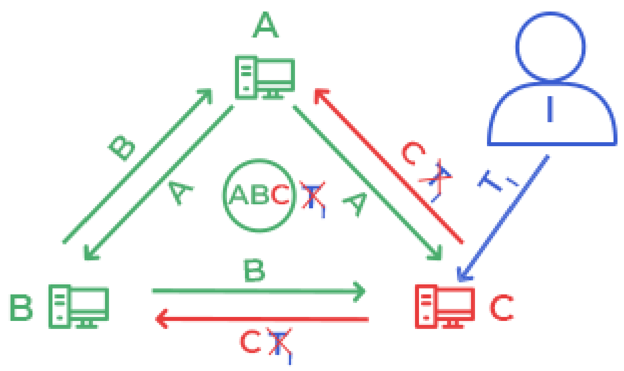
This game implies the need to guess the block that most of the minters will propose without being aware of the preferences of all the others. The optimal winning strategy is to commit a block assembled according to the default rules, which means including all known transactions (or any other set of rules that is reliably expected to be supported by the majority of players).

2) Game explained

Assume that Ivan is a transaction issuer, and that the shard committee consists of $m = 3$ minters, where Alice and Bob are impartial rational players and Chuck does not like Ivan and intends to deny his transaction. At the same time, none of the minters can coordinate his actions with others, Ivan is not aware of Chuck's intentions and a consensus in the shard is reached by $\frac{2}{3}m$.

Also assume that all messages between the participants are delivered within the expected Δ-boundary and that nobody behaves Byzantine (the Dynemix protocol can tolerate communication faults as well as Byzantine behavior, but to simplify the model these conditions are neglected in the following description).

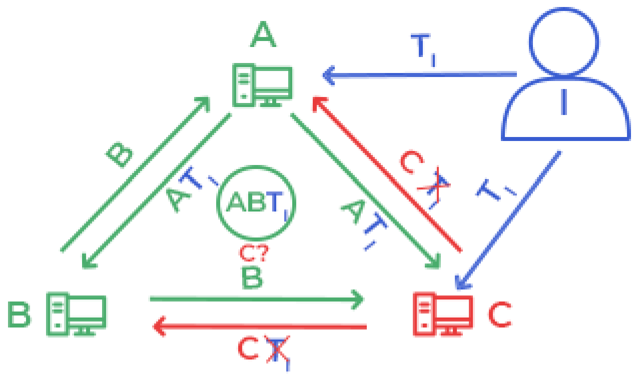
a) **Ivan sends the transaction only to Chuck.**



Chuck receives Ivan's transaction T_I but has no intention of adding it to the block. For this reason, during the transaction sets exchange phase, he submits his set to Alice and Bob, but does not include T_I. He receives Alice's and Bob's sets and sees that they are not aware of T_I. Since none of the minters included T_I into their sets, they all propose identical shard-blocks without T_I and each of them gets a reward. Chuck wins.

For this reason, Ivan should send his transaction to all minters of the shard.

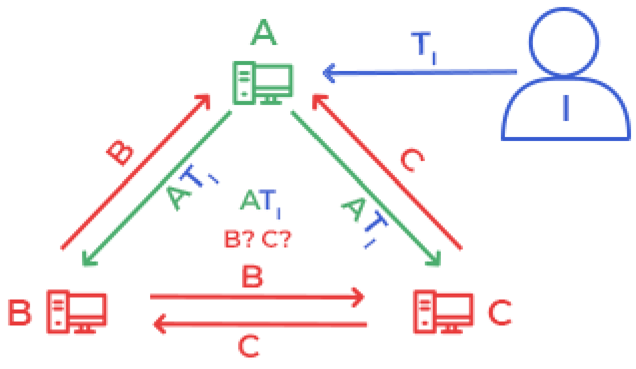
b) **Ivan sends the transaction to Alice and Chuck.**



Chuck receives Ivan's transaction T_I but has no intention of adding it to the block. During the transaction sets exchange phase, he submits his set to Alice and Bob but does not include T_I. He receives Alice's and Bob's sets and learns that Alice was aware of T_I, while Bob was not. Since Alice sent Chuck a set with T_I, Chuck assumes that she sent the same set to Bob.

During the hash exchange phase, he receives the hash of the Alice's set hA from Bob, and since it is identical to the hash of the set that he received directly from Alice, he is now sure that Alice and Bob are both aware of T_I. Chuck knows that if Alice and Bob propose identical shard-blocks, they will win the consensus round regardless of his decision. He has no choice but to commit a block with T_I included if he expects to receive a reward.

c) **Ivan sends the transaction only to Alice, but both Bob and Chuck dislike Ivan.**



Now the game gets complicated. Not only does Chuck want to reject Ivan's transaction, but Bob also intends to do the same. At the same time, Bob and Chuck are independent minters, their actions are not coordinated and they are not aware of each other's censorship preferences.

Even if Bob and Chuck received T_I, neither would include T_I in their sets, but to their misfortune, Ivan sent the transaction to Alice, who is impartial. She includes it in her set, and after the exchange phases, Bob knows that Alice and Chuck are assuredly aware of T_I, and Chuck knows that Alice and Bob are assuredly aware of T_I.

Both Bob and Chuck would prefer not to include T_I in the block, and if they propose identical shard-blocks without T_I, they will win the consensus round, T_I will be rejected, Bob and Chuck will receive a reward and honest Alice will be left without a reward.

Since they are uncoordinated, however, they cannot be sure of each other's intentions (meaning their mutual agreement not only to reject T_I but also whether to reject or accept any other transactions). Chuck may suspect that Bob also dislikes Ivan and might be willing to reject T_I, for Bob did not submit T_I to Chuck in the set exchange phase, but this is a too weak assumption to rely on, unless Ivan is the most hated person in the world.

Chuck and Bob may turn to the rushing strategy: that is, holding their shard-blocks until they see all the other proposed blocks and committing their blocks afterward. If they do so, however, both will only get the hash of Alice's shard-block with T_I included and eventually each of them will be forced to make one of the following decisions:

- To wait more and risk missing the moment when the time slot expires, and thus not getting a reward.
- To commit a shard-block with T_I, thus winning the consensus round together with Alice regardless of the decision of the third player and getting a reward.
- To commit a shard-block without T_I and hope that the third player will side with him and commit an identical block.

The third option is obviously non-optimal, since it relies on a probabilistic assumption that the same censorship preferences will be supported by the consensus supermajority. To a certain extent, this assumption may be eligible with a single T_I and only one remaining waiting player, whose decision will determine consensus, which means that the voting simply turns binary.

If there are ten minters in the shard, however, and Chuck intends to reject the transactions of different issuers (say T_I and T_D) simultaneously, he cannot rely on others having exactly the same preferences. Some may agree with rejecting T_I, but at the same time, they may not agree to reject T_D, which is why they may prefer to side with the honest minters.

Since the voting is not binary and implies a huge number of potential proposals, no reliable assumptions on the censorship preferences of other players can be made, and no unitary adversarial strategy under the uncoordinated adversary assumption can be predicted.

When each rational minter makes a choice, he presumes that there can possibly be up to $m - 1$ honest minters (ones who follow the default rules, in our case) and up to $m - 1$ adversaries, which makes the odds even, but at the same time, honest minters will propose only one variation of a shard-block, while the adversaries can possibly propose up to $m - 1$ different variations chosen from 2^t combinations, where t is the amount of all known transactions, making including all transactions into the block the optimal strategy for both rational and altruistic players. The default known censorship rules may be considered a focal point of the game.

3) Game's guarantees

Guess My Block game ensures with high probability that each T_n will be added to the next block, given that both of the following assumptions are true:

- T_n was received by at least one player, who does not intend to reject T_n.
- Players who intend to reject T_n do not form a coordinated consensus threshold.

Guess My Block may not be efficient if the overwhelming majority of minters start arbitrarily rejecting transactions from issuers, ultimately reducing their sets to zero. This threat is less relevant to blockchains that feature commissions for transaction processing, but this may hypothetically cause a problem with the free-transaction design of Dynemix.

The fundamental limitation on the way to creating a solution for a fully rational environment is the inability to non-interactively prove that certain data were transferred from one node to another. As the transaction sender cannot prove that he actually tried to commit a transaction to the minter, we cannot apply any punitive measures on minters who refuse to add certain transactions to their sets.

On the other hand, such a kind of behavior is equal to the liveness attack as it practically stalls the protocol. If we assume that our BFT consensus inherently features $\frac{1}{3}m$ liveness threshold, the stated attack vector doesn't create any additional threat and hence doesn't require specific counter measures.

4) Guess My Block in the master consensus round

Master-blocks are assembled according to the principles of Guess My Block as well. Instead of transaction sets, minters exchange patches and shard-block headers (or complete shard-blocks) to collect the most complete set.

Minters are rewarded only in the case that a shard-block signed by them is included in the master-block, which incentivizes all members of each shard committee to spread their block as widely as possible.

Considering that each rational minter will obviously vote for a version of the master-block that contains his own shard-block, the most complete set becomes the focal point.

During voting, the shard committees are disbanded, and each minter behaves independently. The consensus threshold in a minter committee of M members is set to $\frac{3}{5}M$.

With the growth of the system, a master consensus round will start creating a larger communication overhead. Unlike shard-committees, which feature a fixed number of participants, master-committees scale according to the size of the account base and the average TPS. At some point, the number of replicas will reach the thousands, which raises the question of further optimization.

In this case, it is reasonable to appoint a number of delegates form each shard, who will be authorized to participate in a master consensus. The largest stakeholders among the minters who signed the shard-block seem to be the optimal choice.

It can be clearly observed that at this stage we apply a lower threshold than during a shard consensus. By doing so, we respect the weakly responsible model described above. We assume that some minters may not make it to the master consensus, which is why the liveness threshold at this stage should be slightly relaxed. In addition, validity cannot be breached at this stage (in the meaning of state output), as all shard-blocks are already settled by the shard committees, which is why we consider it an appropriate measure to provide more liveness at the cost of consistency guarantees.

One may ask a reasonable question – why can we not simply set the threshold to $n > 2f$, which is a known lower bound for a weakly synchronous communication model? Indeed, we can. As we are developing a protocol for practical implementation, however, we are not restrained by the assumptions of any particular model, and we can take different approaches if we consider them to meet our needs.

Unfortunately, protocols that tolerate minority corruptions can tolerate zero corrupted replicas under asynchrony, which is a fact that concerns us. Although we rely on weak synchrony, we still respect the probability of some extreme situations in which the system can be partitioned due to a massive network failure (caused by a war, a global cataclysm etc.). In this situation, we would prefer to retain at least certain minor consistency guarantees that can assure an acceptable level of safety.

With the chosen threshold, the system can hold consistency in the presence of up to $n > 5f$ corrupted nodes even under asynchrony, which seems sufficient given the very low probability of the occurrence of asynchrony, especially with the presence of a powerful adversary who can control communication.

With the help of the described algorithm, we achieve the following:

- assurance that all assembled shard-blocks will be included in the master-block;

- prevention of the grinding of different combinations of shard-blocks to find a random oracle that favors the adversary.

It is worth noting that during the master consensus round, there can be a short opportunity for the adversary to try a grinding attack. To be able to grind a hash of the master-block, the adversary should meet the following conditions:

- control the BA threshold of minters in at least one shard;
- receive all other shard-blocks before grinding.

If the adversary fulfills both requirements, he will have a very short time slot to grind different versions of a shard-block in order to find a master-block with a hash that will provide him a better representation in the next minter committee.

We do not consider this attack vector to impose a serious threat for the following reasons:

- Obtaining a consensus threshold in even a single shard is not easy to achieve, unless the adversary controls a comparable proportion of stake in the global pool. In other cases, he will gain an opportunity to take over a shard only extremely rarely at best.
- A time window for grinding is so short and the number of values is so huge that the probability of finding a hash that can provide significant advantage is neglectable even for a very powerful adversary.

5) Coercive attack

Although the adversary is required to possess $> \frac{2}{3}$ (the consensus threshold) of the total voting power to arbitrarily apply any desired censorship rules, the adversary who controls $\geq \frac{1}{3}$ (the liveness threshold) of the voting power can try to force rational minters to follow his censorship rules under the threat of halting the consensus and preventing minters from receiving rewards.

An attack can be executed in different ways, but the basic conditions are the following:

- The adversary controls $\geq \frac{1}{3}$ of the voting power and convinces the majority (or other required threshold) of the other stakeholders of the seriousness of his intentions;
- The adversary publicly declares his block assembly rules and actually follows them.

If all participants except the adversary are purely rational, they will conclude that if the adversary always proposes shard-blocks assembled according to his declared rules, he will either win or break the consensus. The only option to get a reward for others is to follow the declared rules.

We do not consider this attack scenario realistic in the practical setting of Dynemix for two reasons.

- The adversary will unlikely be able to convince the required threshold of honest stakeholders to submit to his authority and let him control the system in this way. It is more likely that honest minters will reach an agreement and simply ban the adversary via a hard fork. Considering the size of the stake required for the attack described, the adversary will suffer tremendous damage.
- Such behavior is not completely rational from the point of view of the adversary himself. Considering the setting of Dynemix, the adversary cannot expect all minters to behave purely rationally. Instead of forcing others to follow his rules, he may end up simply stalling the protocol, which is against his own interests.

If we assume that the adversary is irrational and actually intends to harm the system, we can do nothing against a malicious actor who controls $\geq \frac{1}{3}$ of the voting power, as it is an inherent liveness threshold of any BFT consensus model based on an honest supermajority assumption. The only solution is to go beyond the protocol rules and slash the stake of the adversary via a hard fork.

Guess My Block is designed to operate in Δ-bounded (weakly) synchronous setting, which means that it can actually tolerate up to f corruptions out of $n > 2f$ replicas as long as synchrony holds for the required threshold of replicas. We can take advantage of this property and simply set the shard consensus threshold to $> \frac{1}{2}m$, which renders the described attack pointless, as it would require almost the same amount of resources as a consensus takeover attack.

This will reduce safety guarantees, however, which we consider a higher priority (especially in relation to validity). For this reason, we decided to keep the threshold of shard consensus at the level of asynchronous BFT protocols, assuming that the described attack does not pose a serious practical threat.

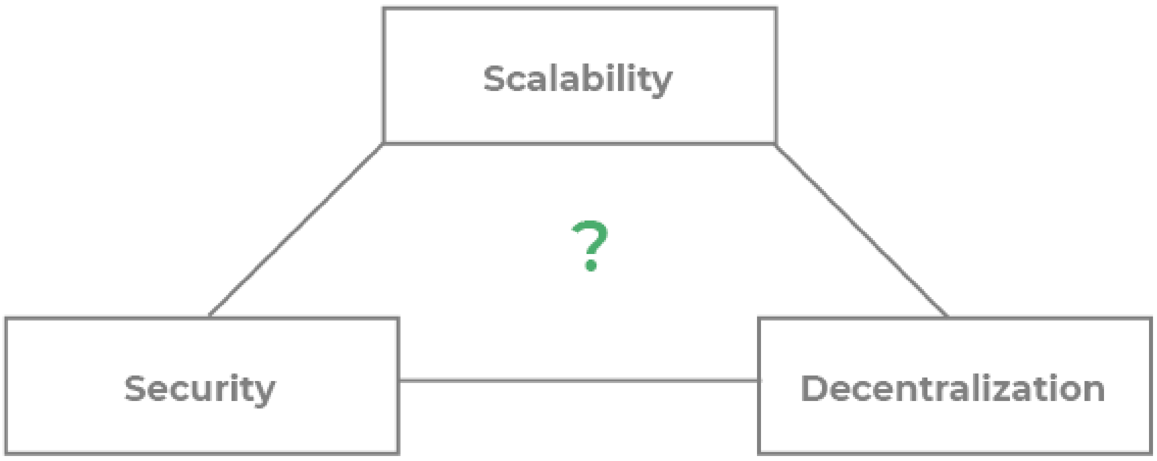
4. Scalability

In the beginning of this white paper, we stated an important feature of a decentralized payment system that we intend to achieve:

- **The system should be capable of processing at least 10,000 TPS.**

Most first-generation blockchains suffer from the scalability issue. Many developers of recent projects have addressed the problem and proposed a number of solutions, which include different tradeoffs.

The main problem at hand is a trilemma that states that higher scalability, security and decentralization cannot be reached simultaneously and that only two properties can be targeted.



Known solutions to the scalability issue can be conditionally classified into two categories, depending on which vertex of the triangle is sacrificed:

a) **Reducing the number of validators to a small group (sacrificing decentralization)**

This approach solves the problem by limiting a possible validator set to a small professional group that possesses powerful hardware and broad bandwidth, which allow them to process and synchronize large data flows. This helps significantly raise the network delay, bandwidth and computing power assumptions.

The set of validators can be limited directly by the protocol rules (DPoS) or by applying a high participation threshold.

Though such a system may be formally considered permissionless, in practical terms this approach makes the system semi-permissioned, as the participation opportunity is drastically limited, which inevitably leads to the emergence of a coordinated oligopoly, whose interests include preventing new players from joining.

Unfortunately, following this approach leads to the severe centralization of the system, thus making it essentially pointless. This is a tradeoff that we cannot accept, which is why we had to search for another solution.

b) **Sharding the system (sacrificing security)**

Another approach is based on splitting the network into independent or semi-independent partitions, thus distributing the load between them respectively.

Though sharding may be considered an effective solution to the scalability issue, it negatively affects safety. Sharding makes the system vulnerable to a dangerous attack vector of a fraudulent block approval.

In classic blockchains, blocks, which are propagated across the entire network, contain all the data necessary to verify that the state change occurred according to the rules of the protocol.

If an adversary tries to commit an unsubstantiated state change (for example, adding to his balance an arbitrary number of coins that emerge out of thin air), the adversary's deviating block will be rejected by all nodes. Since the adversary is disincentivized to make such attempts (in PoW systems, he suffers from the waste of resources, and in PoS systems with slashing rules he loses the stake), non-sharded blockchains are highly sustainable against this attack vector and feature strong validity guarantees.

The sharding concept stipulates partial data propagation to ensure the increase in scalability, which is why the security of the entire system is reduced to the security of each shard. Considering that cross-shard interaction is based on trust between validator committees, succeeding in one shard allows the adversary to affect the entire network.

As the stated approaches engender radical tradeoffs, we had to develop a non-trivial solution to maintain the desired level of decentralization without sacrificing security.

5. Brief introduction to sharding

Our sharding solution corresponds to the specific settings and limitations caused by the targeted properties we intend to achieve. To help better understand our approach, we will briefly describe known sharding concepts and issues from which each of them suffers.

Typically, we can use two different sharding concepts:

a) **Without the global state**

This approach isolates the shard state and hence requires additional cross-shard communication about the majority of transactions (particularly transactions that change the states of accounts assigned to different shards).

Though sharding with isolated states provides more scalability, it also begets more problems.

The first and most important issue is security. As the state transition is performed and validated only within the shard, a successful attack on a shard breaches the security of the entire system. The most dangerous vector in this case is adaptive corruption or coordination. Since validator committees in isolated shards cannot be quickly reshuffled, the only option to mitigate the risks is to appoint very large committees and assume that the attack delay will be assuredly greater than the reshuffle rate.

In addition to the security issues, complete sharding poses another serious problem – delayed finality.

The global state contains information only about shard assignments, and account states are stored only within shards to which accounts are assigned. Each shard works independently, and in the case of a cross-shard transaction, the problem of atomic commits arises.

Proposed solutions to the problem feature sequential updates of the states of both the sender's and the recipient's shards in four steps, which significantly increases finality latency.

Finally, complete sharding reduces data availability, which causes concerns, especially in relation to crucial data, such as the states of the accounts.

b) **With the global state**

Partial sharding involves the sharding of a state-transition procedure, but at the same time does not isolate shard states, assuming that all shard states are assembled into the global state, which is then propagated through the entire network.

This approach helps solve certain security issues, such as adaptive corruption and coordination, by allowing the reshuffling validator sets as quickly as needed. It still features lower overall security than non-sharded designs, however, due to the partial data propagation.

In addition, partial sharding allows us to solve the finality latency issue, and to increase availability of crucial data.

On the other hand, it provides weaker scalability.

6. Problems of sharding in Dynemix

1) Our sharding scheme

Having assessed all the pros and cons of the described designs, we concluded that currently the optimal solution for the scalability issue in Dynemix is partial sharding. We also find it feasible, however, to implement complete sharding architecture in the future. A possible solution would require further research, and it is more reasonable to start with partial sharding to assess the system behavior in practice under real-life conditions before moving toward the implementation of complete sharding.

The main reason for such a decision is that we do not treat scalability as a property that should be enhanced as much as possible. We need to reach only a presumably sufficient level, exceeding which does not bring any more tangible benefits. There is absolutely no sense in providing a million TPS if we assume that the demand can simply never grow that high.

According to our estimates, which are based on the available statistics on conventional centralized payment infrastructure, 10,000 TPS is the approximate upper level of demand under current conditions. Given that we are creating a potentially more advanced infrastructure than has been available to date, we may assume that the variety of use cases may expand in the future, pushing the demand beyond the stated boundary, but in any case it will unlikely be exceeded by an order of magnitude (especially considering that Dynemix is not designed to support Turing-complete smart contracts).

Partial sharding allows us to reach the required scalability while at the same time providing solutions to several important issues, which is why we consider this option preferable. Complete sharding should be considered only if we face a substantial lack of scalability of the current design.

In Dynemix, each transaction is assigned to a particular shard according to the transaction's issuer. Minters of the shard process the transaction and update the states of both the sender and the recipient (or multiple recipients if the transaction has multiple outputs). The resulting state changes are then propagated as patches throughout the network so that the minters of the next block know the entire state of the system and there is no need for cross-shard communication during the transactions' processing.

This approach allows transactions to be finalized within one consensus round and excludes the opportunity to abuse the train-and-hotel problem for double spending. With the help of a global state, we apply constant resharding, which helps instantly adapt to the current level of demand, provide a solution to the weakly responsible model of participation and resist adaptive corruption.

It still brings a number of security issues, however.

2) Adaptive attacks

In the section describing the Guess My Block game, we noted that the game puts a number of restrictions on the system design that negatively affect different properties of the platform.

One of those restrictions is the inability to involve a large number of participants. With the growth of the shard committee, communication overhead may become intolerable for most players, and scalability and decentralization will be negatively affected. This means that to provide sufficient scalability and keep the sharding solution efficient, we have to keep the size of the committee small, but this inevitably reduces security in the adaptive corruption model.

Assume that the adversary intends to bribe minters to approve his fraudulent transaction or censor certain third-party transactions. As we apply sharding, this means that, instead of bribing the majority of stakeholders from the global pool, the adversary needs only to choose one shard committee after its designation and bribe the consensus supermajority within a single shard.

As far as full nodes operate in the blockchain mode, the adversary can only commit a censorship attack. Validity is not jeopardized, for shard-blocks are additionally verified by all nodes in the system (in the same way as in non-sharded blockchains). Switching, however, to quilt opens an opportunity to approve an invalid state change, since validity stops being verified beyond a shard consensus. This threat is countered by the authorized master-nodes concept, which will be described further. Nevertheless, we still find it crucial to resist the corruption of shard committees.

Theoretically, this problem may be solved by appointing a very large committee (which will increase the attack delay assumption), but we cannot afford this due to the Guess My Block restrictions. We need to solve the problem in a different way.

Assume that we have a committee of ten minters, and given the BA threshold of $> \frac{2}{3}m$, the adversary needs to bribe only seven of them. This task looks feasible if the adversary has enough time to coordinate the minters' actions.

With the help of the global state, however, we can apply the constant reshuffling of shard committees to deprive the adversary of the opportunity to coordinate minters. As shard committees are formed only for the creation of a single block, the adversary only has time between the propagation of block $B_{n-1}$and the shard-block commitment of block B_n, which takes about 6–7 seconds. Bribing minters in such a short period of time does not seem possible, making Dynemix secure in the presence of a mildly adaptive adversary.

The same can be applied to adaptive coordination. If minters assigned to one shard committee try to coordinate their actions to approve a fraudulent shard-block that grants benefits to all attack participants, they will not have enough time to reach an agreement within the time slot at their disposal.

We can conclude that Dynemix is not designed to be secure in the instant adaptive corruption model. We do not, however, consider the assumptions on which this model is based practically attainable in the Dynemix setting.

3) Consensus takeover attack

One of the typical attacks on blockchains features an adversary who obtains the amount of resources required to affect a consensus. By controlling a liveness or consensus threshold of the participating replicas, the adversary can influence several properties of the system. In relation to most decentralized SMR systems, we can talk about *validity*, *consistency* and *liveness* as the set of properties that can be compromised by the adversary.

The design of Dynemix allows to include an additional property in this list – *fairness* that is in our case, agreeing on an output derived from an uncensored set of inputs. In most blockchains, the set of transactions to be included in a block is chosen by the block proposer (or the leader), who acts at his own discretion, which is why fairness cannot be guaranteed during a state transition. With the help of Guess My Block, however, Dynemix provides certain fairness guarantees, which is why we can include fairness in a set of properties that can be held under normal conditions.

Another important feature of the platform is a two-layer consensus, which means that the stated properties can be actually violated twice during a single round of protocol execution. This makes the attack scenarios more variative and complicated.

Finally, Dynemix can function in the blockchain mode (meaning that all nodes operate with complete shard-blocks and master-blocks) or in the quilt mode (when only patches are exchanged, and a quilt is built during the master consensus phase). These two types of operation also feature different optimal attack strategies and adversarial thresholds.

In this section, we will describe the most dangerous attack scenarios and try to evaluate the level of resilience that can be provided with the current design.

First of all, we need to apply measures that impede the Sybil strategy.

a) **Fair weighted sampling**

The first step is applying fair weighted sampling on a minters' committee designation stage to assure that there is no economic benefit for the stakes of lower sizes, and there are no incentives for stakeholders who split the stakes between Sybil accounts. This is also essential for the economic model of Dynemix to function properly.

Since shard committees feature a low number of participants, however, there is statistical dispersion that provides a non-zero chance that after a certain amount of rounds the adversary eventually may get lucky enough to have more than an average number of his Sybil accounts simultaneously assigned to a particular shard, which can lead to a single-shard consensus takeover.

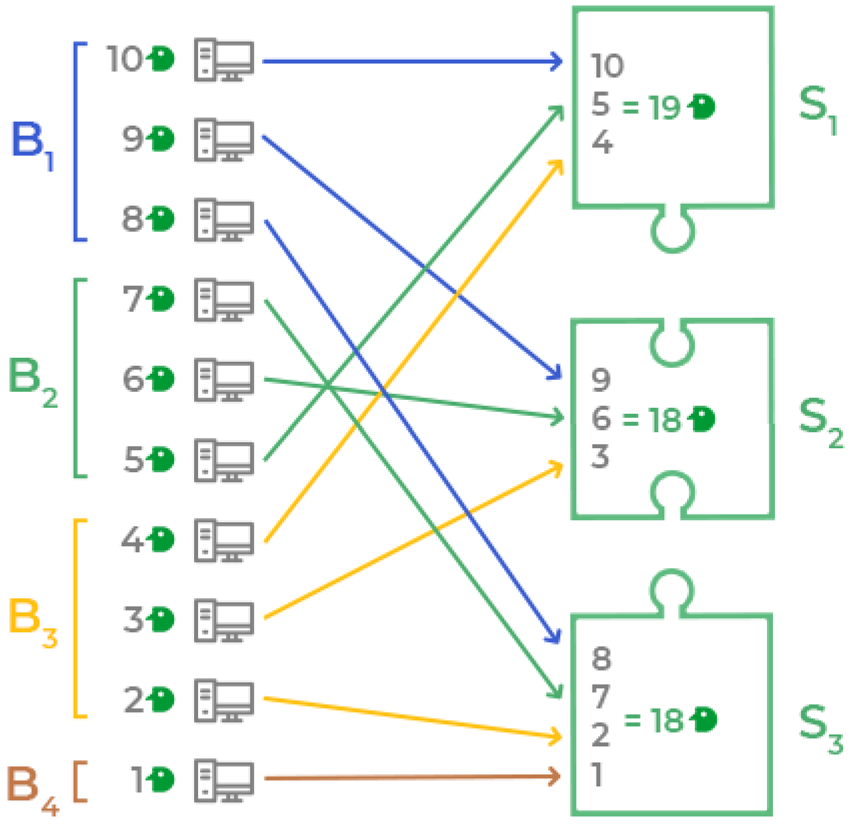
Under such conditions, if the adversary tries to control shards, the optimal strategy is to split the stake among the highest possible number of Sybil accounts (the total stake of the adversary divided by the minimal stake amount) and wait until the dice rolls in the favor of the adversary. Considering the ten seconds' block time, more than 3×10^6 reshuffles will occur each year, with the number of shards possibly reaching the hundreds, which provides an optimistic forecast for a patient adversary.

b) **Bin sorting**

To further impede the Sybil attack, the second step is added – sorting minters among shards in a way that assures a proportional stake distribution. Since the stated task is a variation of the partition and bin-sorting problems and hence is NP-complete, the optimal solution can create excessive computational overhead, which is why we use a simple approximate algorithm, as we do not need high precision.

All selected minters are sorted by the sizes of their stakes and split into bins according to the number of shards. Minters from each bin are then sorted by shards as the picture below shows, excluding the final round, when a greedy algorithm is applied to assure better optimization.

The algorithm forces the adversary to develop an adaptive strategy instead of simply splitting the stake among the highest possible number of Sybil accounts. It also solves the problem of uneven resources at stake, as simple random sampling would occasionally create shards in which minor stakeholders would form the Byzantine agreement supermajority.



Now let us investigate the most dangerous attack scenarios that could possibly be executed by a rational adversary.

a) **The system operates in the blockchain mode.**

When Dynemix operates in the blockchain mode, a single shard takeover does not pose a serious threat, since validity and consistency are assured during the master consensus; hence, the attack opens up only a censorship opportunity. Censorship starts posing a real threat when the adversary is capable of getting substantial representation in each block. Under such conditions, the adversary can censor third-party IMIN transactions, which can eventually allow the adversary to take over the entire system.

If the adversary intends to seize control over the system, the most obvious solution is to acquire the amount of resources close to the master consensus threshold ($\frac{3}{5}M$). Once the adversary controls about $\frac{3}{5} - \varepsilon$ of the global stake pool size, he will be able to control the master consensus and apply arbitrary censorship.

By violating fairness and liveness at the shard level, however, the adversary can develop a strategy that will allow success with many fewer resources.

Say the adversary possesses the amount of resources that allows him to continuously obtain a consensus threshold in a number of shards and a liveness threshold in a greater number of other shards, respectively. He censors the IMIN transactions of other stakeholders in those shards in which he obtained a consensus threshold, and blocks the consensus in those shards he obtained a liveness threshold. By doing so, he improves his relative representation in the next minters' committee. If this strategy is consistently executed, eventually he may have a chance to obtain a master consensus threshold, thereby seizing control over the system.

According to our approximate tests, in such a scenario, controlling about 30–35% of the global stake pool can provide an opportunity to succeed within a reasonable time frame.

b) **The system operates in the quilt mode.**

When Dynemix switches to the quilt mode, an eventual shard takeover can allow the adversary to breach validity, since the complete transaction data is not propagated globally. For this reason, a rational attacker does not need to execute complicated strategies and is more likely to concentrate on a simple shard takeover and an approval of a fraudulent transaction. This attack would require the possession of fewer resources and poses fewer risks for the adversary than the master consensus takeover attack described above.

According to our approximate tests, controlling only about 2% of the total stake would eventually allow the adversary to gain control of at least one shard for one round within a reasonable time interval.

We should emphasize that the stated thresholds (2% and 30%, respectively) were obtained as results of approximate tests. The precise actual amount required for the attacks described will depend on the degree of the stakes' dispersion and the adversary's strategy, which is why further research on the subject is required for a more precise assessment.

In the quilt mode attack, resilience seems to be insufficient, considering the possible consequences of an attack, and additional measures are required, which include the following:

a) **Targeting a sufficient global stake pool size**

The percentage of resources in the adversary's possession can have a totally different interpretation in absolute numbers, depending on the size of the global stake pool. For example, if we evaluate Bitcoin's practical attack resilience, we can figure out that a 51% attack could be described as a 0.3% attack if we express it through the amount of required resources relative to the total market value of the platform at the time of the attack.

If we operate under an assumption of a 2% adversary's stake (which is very approximate), we can conclude that creating a global stake pool of the size equal to 15% of the circulating volume raises the quilt mode security guarantees to the level of Bitcoin. To secure a certain level of redundancy, we target 20% stake pool size.

We applied a number of measures to assure a sufficient global stake pool size (which at the same time helps raise the level of decentralization):

- Setting a small minimal stake amount

  Although lowering the minimal stake threshold allows the adversary to create more Sybil minters, at the same time it lowers the participation threshold for the general audience, which is expected to provide more security and decentralization, thus balancing out the negative side effect. We need to find an optimal point of balance in this trade-off.

- Securing low hardware and bandwidth requirements

  This helps common users participate in minting without excessive overhead, which, together with the previous measure, should lead to the involvement of more ordinary users.

- Allowing funds to be unbonded quickly

  Since the stake amount can be unblocked and spent almost instantly, participants are incentivized to use all available coins as stakes, which greatly increases the size of the potential stake pool.

- Targeting mild inflation

  The economic model of Dynemix can target a desired rate of inflation, which is why rational owners of large numbers of coins will likely try to participate in the minting process instead of keeping savings in the free form to compensate for inflation.

- Adopting the model of weak responsibility

  Allowing sporadic participation attracts additional stakeholders who cannot maintain constant availability.

For a more detailed description of the coin issuance model, please refer to the Economics paper.

Given the measures described, we expect the global pool to reach the targeted 20% size, which means that Dynemix's quilt mode may potentially operate on the same level of security as Bitcoin or even surpass it (although we should note that further research is required to reliably confirm this claim).

b) **Building the second line of defense**

Although the measures described help strengthen security and presumably even raise it to the level of classic PoW blockchains, we still find the security guarantees that we can provide in the quilt mode insufficient.

As we intend to create a global digital currency payment system, more security is never superfluous.

During the development of Dynemix, we considered the following measures:

- Adding another round of consensus

  Since the problem of additional security can be solved by increasing the size of the shard committees, which we cannot afford due to the limitations of Guess My Block, we can simply add another BFT consensus round that involves the required number of participants.

  Say ten minters play Guess My Block and decide on a shard-block candidate. Upon receiving the BA result from the committee, another 500 randomly sampled notaries take a binary vote on the question of whether to approve or reject the shard-block candidate, thus confirming its validity. The second consensus round finalizes the shard-block.

  Although this approach allows the shard takeover threshold to increase greatly (pushing it to $\frac{2}{3} - \varepsilon$ of the global stake pool) by increasing the size of the validator sample to a scale that nearly nullifies the statistical dispersion, it negatively affects finality latency and throughput; a consensus round of hundreds of participants will take significant time to process due to the excessive communication overhead.

  Although there are some proposed solutions to reduce communication complexity for BFT algorithms, we do not find the inherent tradeoffs of these solutions acceptable. For example, using BLS threshold signatures can greatly reduce the amount of data exchanged between participants, but the concomitant increase in computation overhead can balance out the positive effect.

- Involving authorized notary (witness) nodes

  Instead of adding another BFT consensus, we can offer authorized nodes to testify that the block was seen and verified. This approach does not necessarily require an agreement to be reached, and there can be different models of this concept's implementation.

For example, we can use the reputation-based assignment of notaries, DPoS voting or any other method of choosing a set of trusted nodes that can vote for shard-blocks and/or master-blocks they consider valid. The more votes each block gets, the more reliably finalized it becomes (different designs can weaken the consistency guarantees of the system to eventual or probabilistic consistency).

There can be of various sets of rules on the notarization procedure, but this model has a fundamental flaw: since a small group of predefined players can decide whether the entire system accepts a block, this creates incentives for cartel formation and censorship strategies.

- Involving fishermen

  If the previous solutions presume that the block is not accepted until it is signed by a certain number of verifiers, the fishermen concept presumes the opposite – the block is considered valid until a fisherman triggers a fraud alert.

  This approach ensures better decentralization, as fishermen cannot directly influence the finalization of valid blocks, but on the other hand, it inherits a dilemma that is hard to resolve. If we can manage to provide a reliable solution to the fisherman's dilemma, this concept seems to be the most optimal for our system.

7. Master-nodes for additional safety

Our solution for the second line of defense is based on the concept of authorized fishermen. It allows security assumptions to be greatly increased, while at the same time retaining the decentralization achieved in the quilt mode and not delaying finality.

1) Fishermen concept

According to the initial notion, fishermen are nodes that perform various kinds of spot checks of data chunks in search of faulty data. If a fisherman finds a fault, he propagates an alert through the network and gets a reward.

a) Unintentionally faulty data

Fishermen may be an effective solution to provide redundant fault tolerance, but blockchains do not need it, because blockchain technology features high common fault tolerance by default.

If we assume that a state change is replicated on a number of nodes that should get the same output in order to reach consensus, we can claim that sufficient redundancy is provided at the consensus round. There is a near-zero chance that a number of independent minters will simultaneously suffer the same common fault, which makes fishermen useless in this case.

b) Maliciously faulty data

Though there is clearly no point in the fishermen searching the blockchain for unintentionally faulty data, they might be interested in searching for maliciously corrupted data. If there is a possibility of a shard consensus takeover, there is also a chance of finding an invalid state change.

The problem is that the adversary will simply not give out the data. Suppose the adversary creates a fraudulent shard-block and the fisherman requests the transaction data. The adversary knows that if he discloses the data to the fisherman, the latter will certainly propagate it through the network and every node will be aware of the fraud. That is why the optimal adversarial strategy is to keep the data unavailable.

c) **Unavailable data**

If the adversary ignores the fisherman's request for the data, which is the optimal behavior under the assumption that the data is fraudulent, the fisherman will have no proof of adversarial behavior and will not be able to trigger an alert and receive a reward.

The only option is to allow fishermen to trigger an alert if the node ignores their requests, but this solution begets a dilemma.

2) Fisherman's dilemma

Since concealing data is the optimal strategy for the adversary, we can state that fishermen have no economic interest unless we provide an opportunity to act in the case of data non-availability. On the other hand, if we grant fishermen such authority, it creates preconditions for its abuse.

a) **Either fishermen or minters can abuse their opportunities.**

If we try to set rules for a rational environment, any model of economic incentives will feature a contradiction, as neither is the fisherman able to prove that the minter received the request and ignored it, nor can a minter prove that he submitted the requested data.

Suppose we granted fishermen permission to trigger an alert if minters do not satisfy the request. In this case, any fisherman can initiate a DDoS attack on their chosen minter at a near-zero cost. A malicious fisherman triggers an alert, and the entire network spams the minter with data requests. The minter cannot ignore it; otherwise, he will be accused of fraud. Nor can he prove that the fisherman's claims were deliberately false.

Suppose we added a kind of a stake requirement for fishermen, so that in the case that a fisherman triggers a number of false alerts, we can apply punitive measures on him. Then malicious minters will intentionally conceal valid data from fishermen in order to trigger alerts and present the data afterward, thus draining their stakes until no fishermen are left.

In addition to the problem of unbalanced incentives, there is a problem of near-zero success expectations.

b) **The fisherman has low motivation, knowing that he will not succeed because the adversary is aware of his presence.**

If we suppose that the fisherman and the adversary are players who try to think a step ahead of each other, their attempts to develop a winning strategy in their rivalry will face a contradiction.

The adversary knows that if he tries an attack, a fisherman will likely discover it and his resources will be wasted. This is why the adversary will likely refrain from attacking while fishermen are present in the system.

The fisherman understands this and concludes that there is no economic sense in suffering the waste of resources with a near-zero chance of succeeding against the adversary.

Under these assumptions, both the fisherman and the adversary find themselves playing a draw duel like cowboys in westerns. The only difference is that, in our case, none of the players is likely try to draw a gun due to expectations that his move will be effectively countered by the opponent, which means that the optimal solution is simply not to play.

Taking into account the above, it can be stated that fishermen can strengthen the security of the system by doing absolutely nothing – the mere threat of their presence forces the adversary to adjust his strategy.

On the other hand, there is always a possibility that some daredevil may commit an attack supposing that no fishermen are active due to the problem of low success expectations, which is why we do not consider this model sufficient for building security guarantees upon.

3) Major stakeholders as fishermen

Our solution is built upon an assumption that there is always a category comprising nodes that are indirectly incentivized to keep the system secure, which is why they can perform the functions of fishermen even without any direct economic incentives.

As we have explained, nodes can operate in either the quilt mode, which has lower hardware and bandwidth requirements, or the blockchain mode, which, not being fully affected by sharding, significantly raises the overhead. Nodes that keep the record of the entire blockchain are called master-nodes.

Suppose the adversary overruns the consensus in a shard and commits a fraudulent shard-block. Other minters, who operate in the quilt mode, obtained only the patch of the shard-block and, unable to verify whether all the transactions initiated the state change occurred fully according to the rules of the protocol, accept the patch as valid.

To verify the shard-block and make sure that it is valid, a node must obtain the entire shard-block from the adversary, but the adversary can ignore such requests, being aware of the consequences.

Suppose there are a number of fishermen who constantly search blocks for faults and are authorized to trigger an alert in the case of data non-availability. The adversary understands that he has little time to profit from his fraudulent behavior. The only option is to try to exchange his coins for an off-chain asset as quickly as possible, before the fraud is detected.

The adversary will likely turn to major players who can instantly accept a large transaction, which will include stock exchanges, market aggregators, providers of financial services etc. Given that such players are most likely to become victims of attacks and that they operate with significant coin volumes, they have strong incentives to run master-nodes to assure better security for themselves.

At the same time, as such services aggregate large numbers of resources, they will likely have significant constant coin reserves. Considering that minting provides additional income and due to the fact that, in Dynemix, minters are allowed to unstake almost instantly, which means that staking most of the reserves will not deteriorate their liquidity, such services are incentivized to participate in minting as the largest stakeholders.

Combining these circumstances, we can conclude that the largest stakeholders will presumably run master-nodes, which will verify the entire blockchain and will be the most likely victims of fraud attempts. Under such assumptions, it seems logical to grant such players the authority to alert the network in the case of a fraud suspicion. Since there is still a small chance that even a major stakeholder may abuse his powers to harm arbitrary minters, an aggregate claim of three master-nodes is required for a full-scale alert.

Suppose that Alice, Bob, Carol and Dave run stock exchanges. Malicious Malory captures a consensus supermajority in a shard and adds a fraudulent transaction to the shard-block. This transaction transfers funds to Malory's account on Alice's stock exchange.

a) **Alice does not run a master-node.**

Alice is careless enough not to run a master-node while dealing with large money flows. She accepts Malory's fraudulent transaction and commits a counter off-chain asset to Malory.

Bob runs a master-node and requests data from Malory, but Malory does not respond. Bob suspects Malory of fraud and sends alerts to other authorized fishermen nodes.

Bob presumes that if Malory is an adversary, she may likely have tried to transfer funds to a stock exchange (possibly Alice's). Even if Bob does not care about the safety of the system and the suspicious shard-block does not directly affect Bob's funds, he is not interested in letting Alice profit off of this situation, since Alice is Bob's competitor. Furthermore, causing economic damage to Alice is indirectly beneficial to Bob.

Carol and Dave, having come to the same conclusions, agree with Bob, and together they trigger an alert in the system.

After the network ensures that Malory's shard-block is fraudulent or unavailable, the shard-block gets banned, as do Alice's funds that were received from Malory. Malory gets away with the off-chain assets she received from Alice, and Alice is left with nothing.

This is why Alice should either run a master-node herself or at least wait until an alert time window ends, assuming that, if no alert was triggered by the master-nodes, Malory's shard-block must be valid. Considering that Alice runs financial services, she is interested in providing a quick response to her clients and keeping herself safe, which is why she will likely prefer to run a master-node of her own.

b) **Alice runs a master-node but does not trigger an alert.**

Alice runs a master-node and finds out that Malory's shard-block is unavailable, but she is too lazy to communicate with other master-nodes to trigger a global alert. She has already made her decision and does not accept the transaction from Malory, as she is suspicious that the shard-block is fraudulent. Since she has no incentive to trigger the alert and does not care about the safety of other users, she leaves the situation to the discretion of other participants.

In the case that all master-nodes suddenly turn lazy, they may find themselves in a fork, because, not being alerted, the network accepts the fraudulent shard-block as valid. This is why it is not reasonable to avoid their fishermen's responsibility if they count on keeping their data consistent with the entire network.

This example shows that major stakeholders are naturally interested in verifying blocks for their own protection, and we do not need to apply additional incentives for fishermen. At the same time, they are not expected to abuse their powers to DDoS minters, as it makes no economic sense for them and they are most interested in keeping the system operating stably. Moreover, the identities of major actors will likely be publicly known, and direct malicious behavior may cause reputational losses as well as various off-chain punitive measures applied to such actors.

Thereby, we circumvent the fisherman's dilemma and create a solution that can operate in a rational environment.

4) Master-nodes and decentralization

The presence of a limited set of nodes with special authorities raises a question of centralization. At first glance, it may seem that our solution reduces the decentralization level of the system, but in practice it does not.

Master-nodes cannot directly influence the state transition process but can only point out a possible fault. After a fisherman triggers an alert, the entire network investigates the situation and decides the fate of the specified shard-block.

The only threat that a malicious master-node can pose is the possibility of increasing the traffic load for a minting node by triggering an arbitrary false data non-availability alert. To accomplish it, the adversary needs enough coins to simultaneously have three of his nodes among the largest stakeholders. Considering that Dynemix's design does not stimulate stakes aggregation (minting pools or similar aggregators), the adversary will likely need to obtain the required amount so that it is in his direct possession. Given the number of coins needed to do this, it is hard to imagine a reason that a player of this magnitude would engage in such an activity.

In the case that numerous abuses are detected in the system, it is possible to adjust the rules and tighten the alert requirements, but we believe that this measure would be excessive, and the problem is unlikely to emerge.

In addition, the system can easily operate in the quilt mode even without a single master-node. The absence of master-nodes will lead to the following consequences:

a) **Security will be reduced to the level of quilt-layer guarantees.**

If none of the authorized fishermen actually run a master-node and perform fishermen functions, the adversary may succeed with a shard takeover attack.

The adversary, however, will not be able to distinguish whether any of the fishermen are actually performing data checks at any given time and adjust his strategy in advance. Given that any unsuccessful attempt will result in the loss of his Sybil accounts' stakes, we may state that the mere implementation of the fishermen concept already provides additional security, although we cannot formally define the degree of the security boost.

b) **History can become unavailable.**

Although no one restricts nodes from keeping the full blockchain history, there is no point in full nodes keeping data longer than proposed by the protocol specification.

As the system scales, the growth of storage overhead may become an obstacle to storing the entire blockchain, unless it is required for a particular purpose (for example, if the node is hosting a block explorer or a similar service).

c) **Data-availability guarantees will be reduced.**

If we assume that no one in the network accumulates shard-blocks, the transaction data will be obtainable only directly from the minters who built the block. Given the small number of participants in shard committees, there can be situations when the transaction data is temporarily unavailable.

This will not affect the opportunity to access the balances of any accounts or to send and receive payments, which means that the system will still work fine, and users' funds will not be threatened. Without complete data, however, upon receiving funds users will not be able to identify the sender or amount of each transaction.

Taking into account that master-nodes cannot influence the state transition process and that their presence is not strictly necessary for the system's operation, we can state that the decentralization level reached in the quilt mode is not seriously affected by the presence of authorized master-nodes.

8. Overall security

1) Opposite attack strategies for different layers

The concept of authorized master-nodes brings with it a massive security boost due to the fact that breaking the master-node layer defense employs a totally different attack strategy than the one the adversary would need to execute to take over a shard consensus.

a) **A shard takeover strategy**

To approve a fraudulent shard-block, the adversary needs to control $> \frac{2}{3}m$ members of a shard committee. To accomplish this the stake should be split among a large number of Sybil accounts. Given the bin-sorting of the global appointed minters' committee among the shards, concentrating large stakes in a small number of Sybil accounts will not commonly allow the adversary to control more than 1–2 members of each shard committee, and the adversary will have to target different bins by placing stakes of different sizes.

b) **A master-nodes takeover strategy**

To breach the master-node-layer security, the adversary needs to control all except two authorized master-nodes. Given the 200 nodes limit, this would require controlling a stake of the size more than 198 times larger than the stake of the third largest stakeholder. This means

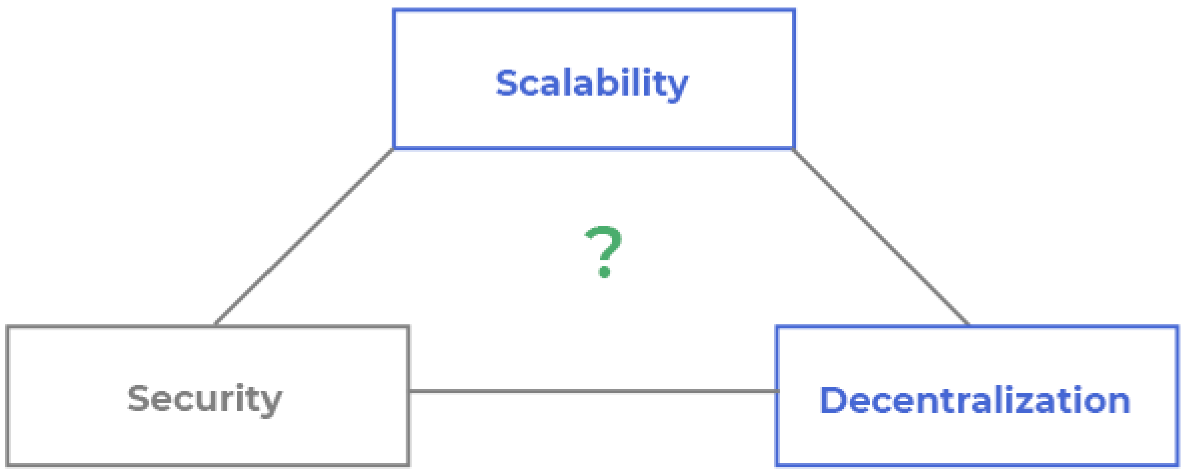
> that the adversary needs to accumulate large stakes on a small number of accounts, which contradicts the shard-takeover strategy.

Combining the number of coins needed to control the required number of master-nodes and at the same time having the chance to take over a shard consensus, we may presume that this model provides sufficient security. We cannot formally define the amount of resources needed for a successful attack, as it depends highly on the actual stakes dispersion, but intuitively we can presume that security guarantees will likely be more than sufficient.

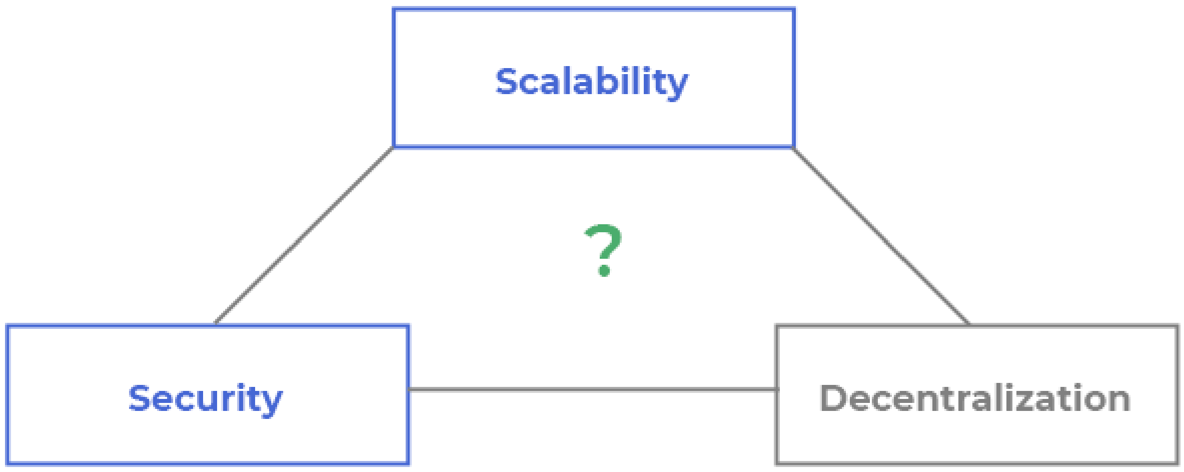
The master-node security layer can be also breached by a successful DDoS attack. A simultaneous DDoS of 198 professional nodes, however, which can also engage several instances run in different locations, is a highly unlikely scenario.

2) Merging two triangles

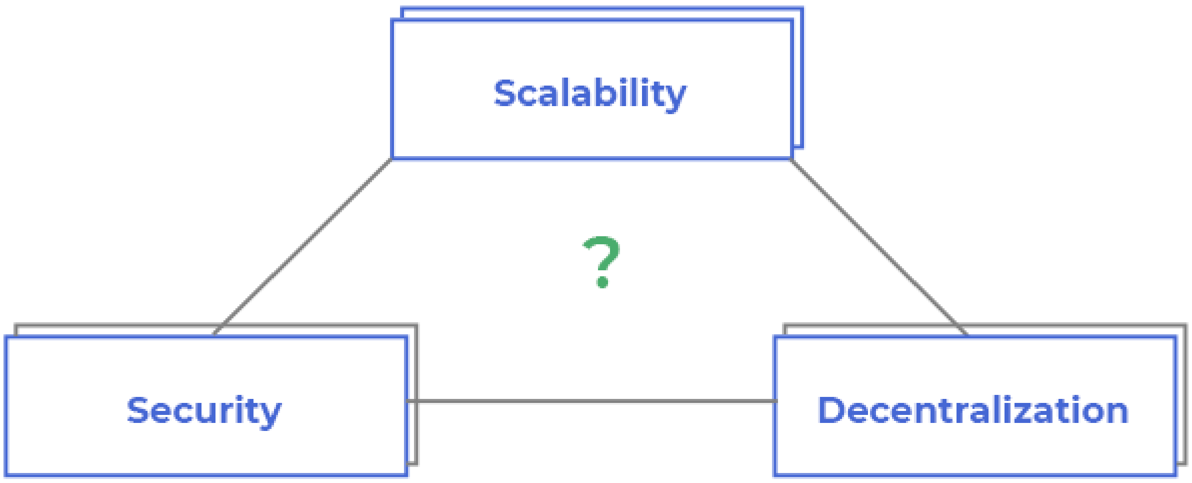
By merging two layers with different architecture, we have managed to provide a multi-layer solution to the scalability trilemma without the need to sacrifice either security or decentralization.



The quilt layer provides more decentralization, as it features much lower hardware and bandwidth requirements for nodes that operate on this layer. At the same time, security on this layer, though comparable to the security level of classic PoW blockchains, does not seem to be strong enough to satisfy our high standards. We may state that the quilt layer features high scalability and decentralization but lacks security.



The blockchain layer, being not fully affected by sharding, lacks decentralization, for as the system scales up, the computation, storage and traffic overhead grow linearly. At the same time, it provides strong security, as the adversary would need a large number of resources to gain control of the fishermen nodes. That is why, on the blockchain layer, decentralization is sacrificed for the sake of security.



Being combined, two layers compensate for each other's shortcomings, at the same time keeping the advantages of each layer intact. The master-node layer provides additional security for the quilt layer but does not reduce the overall decentralization achieved.

9. Achieved scalability and its further possible increase

1) Efficiency of the current design

The main purpose of sharding is to distribute the computation and communication overhead of peer nodes, allowing the system to process more transactions with the same hardware and bandwidth assumptions.

If our sharding solution features the global state, which is updated on all nodes, it raises the question: how does it reduce the overhead if it still requires propagation of the same amount of data as classic non-sharded blockchains?

To begin with, our solution allows communication overhead to be optimized for participating replicas and at the same time to provide more DoS resilience by significantly increasing the total number of participants.

Another important matter is that our sharding scheme allows to solve several issues that emerge from other features of the protocol design. Namely, it helps with the proper implementation of Guess My Block game and the economic model of Dynemix.

Finally, the protocol was initially designed to operate in the quilt mode, which provides a significant performance boost, especially after the implementation of homomorphic encryption. At the same time, the system can be initiated with all nodes operating in the blockchain mode (hence being master-nodes), and smoothly transition to quilt and the subsequent encryption implementation without the need for significant architecture reconsideration.

We find it dangerous to start directly in quilt mode, due to the safety issues described above. During its infancy, the system should operate in the blockchain mode, which provides stronger validity guarantees and features a higher adversarial threshold without engaging authorized fishermen master-nodes. After the potential for scalability in this mode is capped, the system can be switched to the quilt mode, which will further increase its scalability. Finally, when the appropriate balance encryption scheme is developed, it can be painlessly implemented into the system, which will already be operating in the quilt mode.

As the protocol allows subsequently switching between modes of operation, we can model the stages of development and estimate how the system will scale over time.

In our tests and calculations, we modeled the computation and communication capabilities that most users can meet using affordable hardware. Our particular setup is as follows:

- 10-megabit bandwidth;
- 300 ms average latency;
- Intel Core i5 7360U (a mid-range outdated laptop CPU);
- Low-range SSD.

We assume that users should not experience a 100% hardware load, as it would be inconvenient, which is why we limit the use of the CPU to a single core.

a) **The blockchain mode (or the master-nodes mode)**

Dynemix blockchain is a chain of master-blocks that are assembled from a set of shard-blocks. Shard-blocks contain transactions themselves as long as a lot of metadata, which is why the complete master-blocks take significant time to be transmitted and validated.

Given the described setup, the system will be able to process approximately 600 TPS using a single CPU core for computation.

b) **The quilt mode (or full nodes mode) without encryption**

This mode operates with the state of the entire system, which is updated by quilts – data structures that contain only a part of the information of the blockchain blocks. Full nodes obtain the current global state and then only apply quilts to execute state transitions. A quilt can be derived from the master-block, but the block cannot be restored from the quilt.

Full nodes experience greatly reduced computation, traffic and storage overhead, as they do not need to process transactions (except while directly assembling shard-blocks).

Given the setup described, the system will be able to process approximately 2,400 TPS using a single CPU core for computation.

c) **The quilt mode (or full nodes mode) with encryption**

Homomorphic encryption with ZK-proofs for each transaction will severely increase both computation and communication overhead, and the only option for providing a scalable solution is switching to the quilt mode. The problem is that, currently, we cannot predict the actual parameters, but we can assume that the more the overhead increases, the more efficient the quilt solution will become.

We can clearly see that the system is capable of providing decent scalability even with very modest setup assumptions. 2,400 TPS is an enormous number, which can be capped only when the system gains worldwide popularity. We assume that by that time we will be able to significantly raise hardware and bandwidth assumptions, which means that Dynemix will be unlikely to suffer from insufficient scalability.

Now, upgrade our setup to the following specs:

- 100-megabit bandwidth;
- 300 ms average latency;
- Intel Core i7 9700 (a mid-range outdated desktop CPU);
- Mid-range SSD.

In this case, we assume that the user can share the entire bandwidth and CPU power, considering that the same level of performance is achievable with only a partial load on average consumer-class modern hardware. Under such assumptions, performance will rise approximately to the following numbers:

- 4,000 TPS in the blockchain mode;
- 16,000 TPS in the quilt mode.

Given the specified results, we believe that the current design will provide sufficient scalability at any stage of the system's development, while at the same time keeping hardware requirements tolerable for most consumers around the world, thus maintaining the desired level of decentralization.

For this reason, we did not go for a complete sharding scheme – we simply do not see the necessity to push scalability further, and we find it more reasonable to employ a more robust and efficient partial sharding design.

We still cannot build any reliable precise assumptions about performance, however, in the case of the implementation of homomorphic encryption. We concede that an actual encryption scheme could require a much higher overhead than expected, and it may happen so that we will face the necessity of pushing scalability further by adjusting the design and accepting other tradeoffs.

2) Splitting state into active and archive

With the growth of the userbase, the storage overhead may become the first bottleneck of the system. A billion users' states would weight about 120 GB raw and 540 GB in an encrypted form, which likely exceeds the amount of storage that an average non-professional minter is ready to share. Synchronizing a state of that size would also be a serious challenge for most users.

One possible solution to reducing storage overhead is to shard and archive the state of inactive accounts. Considering the available statistics, we may expect that many addresses may be inactive for long periods and that only a minority of users will issue transactions at least monthly.

Under such assumptions, it is reasonable to split the system state into different layers, which will then be stored and accessed according to different rules:

a) **Active state**

This layer includes everything needed for block production (stakes, IMINs) and states of accounts that were active at least once within a specific period of time (e.g. last month). If an account doesn't change its state in the determined time interval, its state converts to "archive."

b) **Archive state**

After the account stays idle for a month, its state is archived and then deleted by most nodes. The archive is sharded and stored on a redundant number of nodes, from which it can be easily recovered whenever necessary. As user clients also store states along with cryptographic proofs of their validity, when an archived account issues a transaction, the client includes the state, and the transaction is processed without any additional delay.

These measures can significantly lower the storage overhead at the cost of a slight increase in traffic overhead, which is overall an acceptable tradeoff.

3) Implementing complete sharding (without global state)

This is a more challenging goal. Though we could keep many elements of the current design, it would still require a serious architecture reconsideration, as new solutions for several emerging problems need to be developed.

Fortunately, certain features of the current design can significantly help build an appropriate sharding solution with isolated states. In particular, the Guess My Block game can reliably ensure that both parts of a cross-shard transaction will be processed instantly and simultaneously, which means that we can provide a solution to the train-and-hotel problem that will fit our finality latency standards: namely, we can use a two-phase consensus to finalize a cross-shard transaction.

We will still have to accept certain tradeoffs, however, that will inevitably degrade some properties of the system. We find the current design almost perfectly balanced from the point of view of targeted properties and setting, which is why we consider complete sharding the last resort for the scalability issue.

Furthermore, it is not currently clear whether we will need to push scalability this far and whether a complete sharding solution will actually provide a significant scalability boost that would be worth the concomitant trade-offs.

10. Partitioning and forking

1) Partitioning and CAP theorem

From the point of view of CAP theorem, Dynemix is consistency-focused. We can conditionally call it a CP system, but according to the initial notions of CAP theorem, it is more appropriate to call it "C and moderately P."

The problem is that Dynemix cannot fully tolerate *arbitrary partitioning*. As the system operates according to *the weakly synchronous* model, liveness holds only as long as a required threshold of participating replicas is allocated in the same partition.

In the case that a node is not able to get enough responses from the other replicas by the end of the subslot, it assumes that it has been isolated in a partition and locks in *the recovery mode*, whereby it waits until a valid master-block is seen. Upon restoring connectivity and receiving the current master-block, the node restores consistency with the rest of the network and rejoins the protocol.

It is easy to conclude that if the system splits into a number of partitions, none of which contain enough replicas to assemble at least a single shard-block and then reach a master consensus ($\frac{3}{5}M$), all replicas will enter the recovery mode and the protocol will stall.

Given the assumed network topology of Dynemix, we consider this highly unlikely. A practical partitioning threat refers mostly to situations in which a small fraction of nodes gets isolated from the major network segment, which should be handled fine by the Dynemix protocol.

Nevertheless, what if this actually happens and the system stalls due to the violation of synchrony? We assume that this is an extraordinary situation, which indicates a global cataclysm. In this case, it is reasonable to let the system be recovered in manual mode after the consequences are overcome.

We can theoretically propose an automatic fallback recovery algorithm, but the problem is that by doing so we will inevitably extend the communication model to asynchrony or partial synchrony, with all their respective consequences. We would likely no longer be able to guarantee either adaptive corruption tolerance even against a mildly adaptive adversary or sequential consistency, which seems like a heavy price to pay.

We have chosen consistency over availability for the following reasons:

a) **We believe that consistency is preferable for a consumer-level payment system.**

If we assume that the system will be used by a general audience, we cannot go for availability and allow weak (eventual or probabilistic) consistency. This results in the possibility of forks and complicates the notion of finality (as we noted, fast finality is one of the crucial features we need to achieve).

Common users perceive a payment system as something solid and reliable. This means that if the transaction is processed, no rollback is expected to occur. If the user is isolated in a partition, it is obviously better not to provide him with an incorrect state and not to accept his transactions than to allow them to be rolled back after the system recovers.

Availability may be an option for specific blockchains focused on professional use, when we assume that users understand the consequences of inconsistent system operation, but this is clearly not our case.

b) **Homomorphic encryption does not allow forks to be merged.**

If balances are not encrypted and the system suffers from partitioning, it is possible to adopt merging rules that are applied when the network recovers. Transactions from one fork can be relocated to the other fork (unless double spends occurred), and users will not suffer a rollback when the lesser chain is orphaned. If balances are encrypted but the system uses a UTXO model, relocation of transactions is also possible.

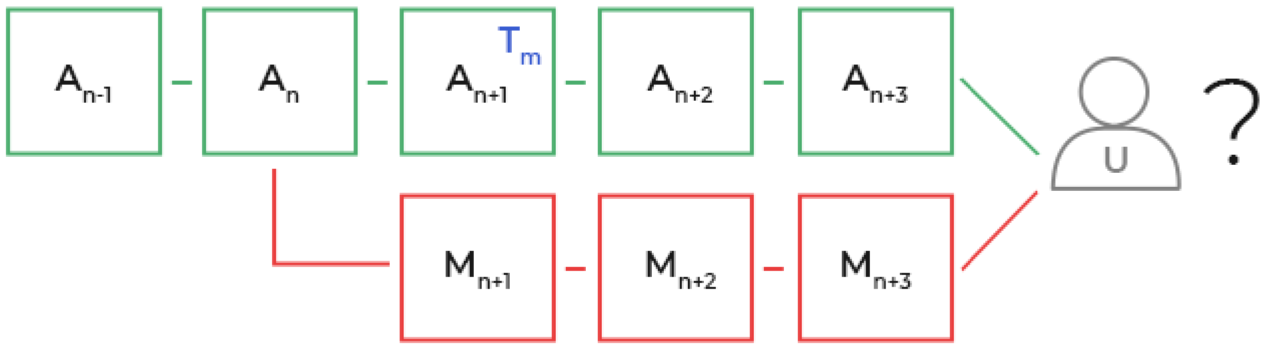
A homomorphically encrypted account-based state, however, does not allow fork merging. The reason is that ZK-proofs become invalid if the state changes. This is why if a user receives funds in partition A, his transactions in partition B will become invalid for partition A.

2) Forks

By being consistency-focused and featuring a BFT-style consensus model, Dynemix is designed to strongly maintain a linear blockchain structure, which means that no forks can occur in the system unless an attack is committed.

All shard committees for block B_n are formed according to the information included in block B_{n-1}. A master-block is considered valid if at least $\frac{3}{5}M$ committee members voted for it. This means that two valid blocks of one height can appear only if a number of minters sign both of them. This cannot happen unintentionally, and it clearly indicates the malicious behavior of such minters.

Since we can be sure that each fork appearing in the system signifies an attack, we can apply punitive measures to the participants in the alternative fork.



Assume that chain A is the main fork of the network that is minted by honest minters and that Malory is an adversary who wants to commit a double spend. To accomplish this, Malory issues transaction T_M, which transfers her funds to a stock exchange. Minters include the transaction in block A_{n+1} , after which Malory exchanges her funds into an off-chain asset. Now Malory wants the network to switch to an alternative chain in which T_M never happened and her funds stayed in her account (or were transferred somewhere else).

She chooses an arbitrary branching point, which obviously should occur before block A_{n+1}. Suppose she chooses block A_n as the parent and starts alternative chain M from height $n + 1$. After she completes all the preparations, she reveals her fork to the network. Suppose it happens on height $n + 3$. Now all nodes in the network must decide which chain they are going to join. The decision is made according to the following rules:

a) **Nodes that witnessed transition $A_n \rightarrow A_{n+1}$ will not buy the bait.**

All nodes that are constantly online and keep the record of the entire blockchain (at least the hashes of blocks) know that block A_n was followed by block A_{n+1}. At the time block A_{n+1} was propagated, block M_{n+1} was not observed in the network and hence must have been revealed later, which is why the nodes consider chain A valid and chain M adversarial. They inspect the latter and slash the stakes of minters who participated in that chain (if any of them have not unstaked).

b) **Nodes that witnessed any block in chain A since A_{n+1} will not buy the bait either.**

If the node did not witness the transition but is aware of any block on chain A between the branching point (height $n + 1$) and the current block (height $n + 3$), it comes to the same conclusions as in the previous case. For example, if the node is aware of block A_{n+2} and did not see any alternative blocks at that height, it means that A must be the valid chain.

c) **Newcomers or nodes that did not synchronize with the network since A_n will have to choose a fork.**

If the node was last online before the branching point or if it simply has just joined the network, it cannot know which chain was minted secretly by the adversary and which one was built by honest minters.

3) Chain selection for non-witnesses

At this point, we address the problems of *subjectivity* and *nothing at stake*. PoW design is considered objective, for any node can choose the correct fork without the need to trust third parties, while PoS is inherently vulnerable to posterior corruption and a newcomer may be unable to distinguish the adversarial chain from the honest chain relying on the protocol rules exclusively, which is why PoS protocols are subjective.

We do not support this division, however, as we do not believe that overall objectivity is practically achievable with the help of any blockchain design. Despite the fact that Bitcoin may be objective on the protocol layer, in practice user–protocol interaction necessarily includes intermediate layers (e.g. client, operating system, hardware etc.), which cannot satisfy the same objectivity conditions. This is why we consider any blockchain implementation in practice subjective to the same extent.

If we assume that a user can verify a Bitcoin blockchain only with the help of software and hence needs to trust the provided result and presume that the software was not corrupted, providing the current state of the blockchain along with the software doesn't make it any more subjective.

For this reason, the problem of chain selection for newcomers can be solved trivially – the hash of a recent master-block should be provided along with the client, or a set of trusted bootstrapping nodes should be predefined for initial synchronization. We do not believe that any more sophisticated solution is required, as subjectivity cannot be circumvented no matter how complicated an algorithm we propose, and we see absolutely nothing wrong with that.

The only category left unaddressed so far is that of nodes that were last online before the branching point and did not witness any blocks above n height.

The main problem with proposing a secure algorithm for this case is *nothing at stake*. While the work applied to a block constructed according to PoW design cannot be transitioned to an alternative block of the same height, in PoS protocols the same stakes can be used to construct an arbitrary number of parallel forks at a near-zero cost, which makes them vulnerable to posterior corruption. A possible solution to the problem was described by Vitalik Buterin and denoted as a weakly subjective model. In short, the solution looks as follows:

- Adopt a slashing algorithm, which penalizes stakeholders who behave incorrectly (for example, vote for more than a single block at one height);

- Lock stakes as security deposits for N blocks to prevent stakeholders from reverting history starting from a branching point less than N blocks deep without the threat of losing their stakes;

- Set checkpoints at the height at which stakes are unlocked, beyond which no alternative forks are taken into consideration.

This means that if the node was last online fewer than N blocks ago, the blockchain keeps roughly the same level of security as PoW blockchains. If the node was unavailable for more than N blocks, the protocol cannot guarantee secure chain selection for it anymore.

According to the initial idea, N should be set to cover a large time interval, possibly reaching months to be effective, which means that stakeholders have to freeze their deposits for a long time.

Although this solution partly solves the nothing at stake problem (at least for N range), we do not believe that such a design is optimal for Dynemix for the following reasons:

a) **Locking stakes for a long period severely deteriorates several properties of the system.**

The main argument against long-term deposits is a concomitant dramatic change in the quality and volume of the global stake pool.

One very important achievement of the Dynemix design is the involvement of ordinary users in the support of the system. Not only did we manage to lower the hardware requirements for minting nodes to the level of consumer hardware, but we also built an economic model that does not create any obstacles to users' becoming stakeholders.

Unlike most blockchains, in which block constructing tends to be a more professional activity with a high participation threshold, Dynemix is designed to be a system in which an average user can participate on equal footing with professional block producers. This brings Dynemix much closer to being a true peer system and the embodiment of the initial ideas behind cryptocurrency technology.

The ability to quickly unbond and spend a user's funds is an essential feature that helps achieve this goal. If we deprive users of this opportunity, this will instantly rearrange the stake pool in the favor of professional participants. The decision to freeze funds for half a year requires serious consideration and is obviously not something an average user can easily afford.

Moreover, this will also affect major stakeholders who provide financial services and need liquidity. If we allow a quick unbond, stock exchanges and similar actors can stake most of their reserve without any threat to their activities. Long-term deposits, however, can deprive them of the necessary liquidity. This feature is also crucial for the economic model of Dynemix, which relies on the assumption of near-equal liquidity of staked and free coins.

This shows that implementing long-term deposits will almost certainly decrease decentralization and security both for full nodes and master-nodes. This is an unacceptable tradeoff, especially considering a highly disputable positive effect of this solution.

b) **The proposed solution helps only a small fraction of users.**

The weak subjectivity model may be relevant to blockchains that allow forks (featuring a common prefix design and the longest chain rule), but Dynemix is strongly consistent, and no forks are supposed to occur (unless a very powerful adversary commits a forking attack).

For this reason, a potential attack can affect only a small share of users who satisfy both of the following conditions:

- They were last online before the branching point;

- They have at least one of the adversary's nodes in the bootstrapping list.

In the case of an attack, the model described can only provide more protection for this category and should not generate any notable boost to the entire system's security, as most nodes will stay completely immune to attack attempts.

Furthermore, the affected nodes will only be subjected to a delay caused by the necessity of choosing the correct chain. Considering that these nodes will presumably need some time to synchronize with the network in any case, this does not seem to be an outstanding achievement for the adversary.

In addition, the weak subjectivity model provides no solution if the branching point occurred more than N blocks into the history, which means that the adversary can simply prepare for the attack and commit it after N passes. In this scenario, the attack will likely affect many fewer users than in the case of a more rushing attack, but at the scale of the entire system, the difference may be barely notable.

Given the stated arguments, we see no reason to implement the proposed model in its initial form. Instead, we adapted the principles of the model to the design of Dynemix.

The deposit lock period is set to two blocks. This allows users to quickly unstake and provide the necessary liquidity for all categories of minters. Such a short deposit period protects them only from double signing any data before each block is propagated and settled, since it provides just enough time to penalize Byzantine actors but does not solve the nothing at stake problem in terms of posterior corruption.

If the node encounters two conflicting blocks at one height that descend from an arbitrary branching point in history, given that the node has not witnessed any blocks after the branching point, the following resolution algorithm is proposed:

a) **Figure out whether both forks are valid.**

The node finds a branching point and requests a set of IMIN transactions with cryptographic proofs from the common parent master-block (A_n) and the signatures of the minters of the first divergent master-block (A_{n+1} and M_{n+1}) from each chain. According to IMIN transactions and the hash of A_n, the node establishes the shard committee's designation for $n + 1$ height. The node verifies that both A_{n+1} and M_{n+1} were built according to the rules: i.e. both were signed by the required threshold of minters.

On this stage, the node needs to download only a small portion of data from three blocks, which is affordable for common users even after the system scales up (approximately less than 5 MB of data for a 100 million userbase).

If block M_{n+1} does not meet the requirements, the decision is instantly made in favor of chain A. If Malory has managed to obtain enough signatures to construct a valid M_{n+1}, however, the node proceeds to the next step.

b) **Figure out the preferences of trusted nodes.**

If Malory presented a valid fork, this already means that she has put a lot of effort into the attack, considering the number of minters' signatures she has managed to obtain (or control initially). On that point it is reasonable to bring in subjectivity and to offer the user decide him- or herself.

To help the user with the choice, the client investigates which fork is supported by nodes that are considered to be run by powerful actors who are unlikely to engage in fraud. This may include stock exchanges, developers, block explorers etc. This may also include nodes that gained a certain reputation from the point of view of the personal experience of the user's node. There can be different approaches to the construction of such a list, and no particular rules are needed on the protocol level.

The information collected should be sufficient for the user to figure out which fork is legitimate. At the same time, this kind of research will not consume any noticeable amount of traffic and can be performed in a couple of seconds regardless of the current system load.

Some users, however, may prefer investigating both chains more precisely.

c) **Download data from both chains and estimate which of them contains more voting power.**

If the user for some reason cannot make a decision in the previous step and wants to discover how much stakes each fork accumulates, he may download data from both chains and conduct research in the following order:

- The node establishes the total stakes in the system and builds the list of stakeholders at height n (the accuracy depends on the depth of the research).
- The node downloads data from each chain block by block and discovers the votes of the stakeholders. Service transactions (i.e. stake, unstake and IMIN) refer to the parent block and thus are TaPoS-compatible. They cannot be replicated by the adversary without re-signing, which is why whenever the node finds such a transaction issued by a stakeholder, it is considered a vote for a particular chain. If a stakeholder is found placing transactions in both forks, the node concludes that this account is compromised and does not count its votes.

By the end of the investigation, the node will have the following information (the accuracy depends on the depth of the research of the stakes at height n and the number of verified blocks from both concurrent chains):

- the number of accounts that voted for each chain;
- the number of stakes accumulated in each chain;
- the current amount at stake (funds that were not unstaked and that Malory actually risks losing).

This information will provide a comprehensive picture of the forks and allow the user to make a choice. The ratio of certain parameters may be also set for non-interactive selection. For example, if the client concludes that more than 99% of the stakes voted for chain A, this obviously must be the valid chain, as Malory is extremely unlikely to be able to collect that many stakeholders' signatures, even if the branching point occurred far back in history, which means that the client is able to make a reliable choice without user involvement.

The last step, however, is relevant only to the period when the system load is relatively low and the investigation will require a tolerable traffic overhead. If the system scales up to a 100 million userbase, downloading the complete data from both chains may seem unreasonable for the average user. If Malory sets a branching point a month deep and builds a chain up to the current height, the data required for the complete investigation will likely exceed a terabyte.

For this reason, we believe that, given the properties of the Dynemix protocol, we should not necessarily concentrate on a formal approach to the chain selection rules. Subjective selection may be a preferable option for most users, who will likely prefer to make a choice according to the information on trusted players' preferences rather than download and process terabytes of data.

Bringing a bit more subjectivity to the table also favors security. If we apply a formal approach, Malory will know particular conditions that will allow her to succeed. Transitioning chain selection rules from the protocol layer to the client layer, however, makes Malory's perspective uncertain. She cannot reliably predict whether a single user will fall for her trap, and, if someone does, whether she will be able to take advantage of it. Although there can be nothing at stake, at the same time there can be nothing in the guaranteed expected return.

11. Finality

Earlier in this chapter, we addressed the issue of finality, stating that it should be achieved as quickly as possible (at worst, within 30 seconds after the transaction is issued). At the same time, we did not explain what finality exactly means in relation to Dynemix.

The notion of finality differs for blockchain systems of different consensus design.

In Bitcoin and other systems that feature probabilistic consistency, finality increases exponentially with each new block mined on top of the block that contains the transaction. The degree of relative finality can be considered individually depending mostly on the importance of the transaction to the recipient. If the transaction features low value, one may consider a couple of blocks sufficient. In the case of a valuable transaction, it may be reasonable to wait for 6–10 confirmations.

In the case described, reaching finality is a prolonged procedure. It is a fundamental boundary of any system based on the common prefix design, which is why we initially rejected this architecture for Dynemix.

In consistency-oriented blockchains with a BFT model of consensus, finality is reached after nodes achieve a consensus or other special events occur, and in many systems, it depends on the number of approved blocks on top to a much lesser or even zero extent.

In Dynemix, we can outline the three main checkpoints of finality:

a) **Shard-block commitment (approximately 4–5 seconds after the transaction is received by a minter)**

After the members of a shard committee reach a Byzantine agreement, the shard-block may be considered finalized under the assumptions that both shard- and master BA supermajorities are honest and the system's liveness is not breached.

This means that the transaction may be reverted only in tree cases:

- The system is under an attack that requires controlling the BA supermajority within the shard (a shard takeover attack);
- The system is suffering an attack on liveness or partitioning due to a network malfunction.
- The master consensus supermajority is corrupt and intends to ignore the shard-block.

Since neither event is expected to occur under normal conditions, this checkpoint provides finality guarantees strong enough for most transactions, except ones of very high value.

Since shard-blocks (or patches) are exchanged only among the minters of the current block, however, most users will not have access to them and hence will have to wait for the next checkpoint to get the confirmation.

b) **Master-block approval (approximately 8–9 seconds after the transaction is received by a minter)**

After a master consensus is reached, the master-block cannot be reverted, which means that, upon receiving the master-block, the user can consider the transaction finalized with only a negligible probability of reversal.

If there is a fraudulent shard-block approval by the corrupted consensus supermajority, however, the shard-block that contains the transaction can be banned, and the transaction can be reverted, which means that in some cases users may prefer to wait until the final checkpoint is reached.

c) **Block validation by master-nodes (approximately 18–19 seconds after the transaction is received by a minter)**

If the user deals with high value transfers, it is reasonable for him to run a master-node and verify shard-blocks himself. In this case, the transaction may be considered fully finalized when the user verifies the shard-block that contains the transaction and concludes that it has been assembled according to the rules of the protocol.

If the user does not run a master-node, he should wait until the next block is committed. By that time, the previous block should be verified by the master-nodes, and in the case that no alarm was triggered, the block must be valid, and the node may consider it fully finalized.

After the last checkpoint is reached, the transaction cannot be reverted unless a massive attack is committed that can fully breach all levels of the system's security. Additional blocks minted on top of the block that contains the transaction do not grant any stronger finality guarantees.

V. Economics of Dynemix

The economic model of Dynemix is described in detail in a separate paper. In the following chapter, we will briefly outline its main features and some issues that need to be resolved before we can implement it on the full scale.

1. Economic Model of Dynemix

1) Brief Description

The economic model of Dynemix features a breakthrough design of a *decentralized algorithmic central bank* capable of simultaneously balancing the value of native coins against fiat currencies and directly maintaining a targeted *interest rate*, which allows to indirectly target an optimal level of *inflation* and *output growth*.

Unlike previous attempts to build a cryptocurrency that is capable of performing the functions of money, which were concentrated around the concept of a stablecoin, we took a step further and designed a model of a stable economy. The key difference is that our model engages all channels of monetary transmission:

a) When Dynemix is used as an ancillary currency in addition to an exogenous fiat currency, it balances the demand discrepancies that originate from the exchange rate and nominal interest channels, thus keeping the exchange rates stable. A significant upgrade over typical stablecoins is the ability to balance the value against all exogenous assets simultaneously instead of being pegged to a particular asset or a set of such, which provides independence.

b) When Dynemix performs as a fully-fledged economy and an exclusive legal tender within a specific economic zone (up to the entire global economy), it can target a desired natural rate of interest directly and desired inflation and GDP growth indirectly. This feature is exclusive to our model and is not supported by any available cryptocurrency.

In addition to that, the economic model of Dynemix doesn't require any special assets as seigniorage-style models do. Dynemix operates exclusively with native utility coins, which circumvents the engagement of securities and the necessity to comply with the respective legal procedures and requirements of regulators.

2) Model's Prerequisites

The model requires a specific setting before it can be activated. Namely, the substantial part of the economy should be comprised of the real market.

We assume that the simplified structure of demand is as follows:

- The transactional fraction: coins that are perceived as a medium of exchange
- The speculative fraction: coins that are perceived as a source of yield

When the speculative motive dominates the demand, monetary effects spread very quickly, and expectations are mostly chaotic, which leads to a *random walk* price fluctuations on the stock markets in the short term. As long as coins serve to a large extent as income-generating assets, the efficiency of the model is highly limited. For example, being implemented into Bitcoin in its current economic state, the model would smoothen the price fluctuations to a highly limited extent at best.

On the other hand, when the real economy develops to a substantial degree, it will introduce consistent trends, which will form unidirectional expectations. Combined with concomitant lags in the transmission scheme, this will highly raise the efficiency of the model. Simply put, if we want to implement a decentralized central bank, we need to build an environment similar to that of conventional central banks.

2. From Launch to Implementation

With the help of helicopter coins and the Liberdyne messenger, we managed to create prerequisites for the development of the real economy and the eventual subsequent implementation of the economic model of a decentralized algorithmic central bank. A possible path from the mainnet launch to the activation of the model can be described as follows.

IMPORTANT NOTICE!

In this section, we provide a possible scenario for the development of the platform that is not targeted as preferable.

We would like to especially emphasize that herein we do not provide any guarantees of capital gain for coin acquirers. Moreover, we have already repeatedly expressed our negative judgement toward the use of dynes for speculation.

At the same time, we are aware that, despite our efforts, Dynemix can face the consequences of speculative manipulation, which, however, are not expected to completely eliminate the possibility of further economic development. Moreover, speculations can benefit the economy of the platform in the way described below. For this reason, we address the model of economic development that involves a large initial influx of speculators, assuming that in other cases the process should run smoother.

1) User types

Depending on the behavior model, users can be divided into three categories:

a) **Investors, who will purchase dynes only due to their capital gain expectations**

Actions of this category are mostly harmful for the platform, as speculations hinder the development of the real economy, but with the help of the Liberdyne/Dynemix architecture, we have managed to create a model that presumably allows the platform to benefit even from the speculative demand.

b) **Users who are interested mostly in earning dynes by providing system support (reward hunters)**

At the early stage, many will join the system with the expectation of receiving significant rewards. With the growth of the userbase and the depletion of proliferation potential, support of the system will cease to bring tangible income, but most users in this category will continue to use the system because by that time they will be able to assess all the benefits the platform provides. Therefore, over time, users from this category will move to the next category.

c) **Users who are interested in the system directly for its functionality (common users)**

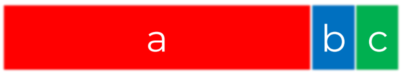
Over time, this category should become dominant and ultimately be the basis for sustainable economic development. Though users will continue receiving rewards for their system support, with a large userbase each participant's reward will be insignificant, so the system functionality should become the main reason for using Dynemix.

As the platform develops, each of the categories will be present in different proportions and have a certain effect on the state of the system. Eventually, we should come to a state in which the absolute majority of users will represent the third category.

The modeled period for the development of the Dynemix platform can be divided into stages.

2) Investment stage

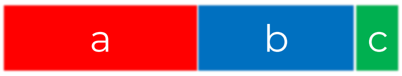
a) **Initial investments**



Assume that we face the worst scenario and nearly the entire initial token supply accumulates in the hands of investors who believe in the potential growth of the system and in the corresponding increase in the value of tokens. We can assume that many of them do not even intend to use the system and perceive dynes only as a tool for speculation, which renders the perspective of the development of the real economy highly uncertain.

If Dynemix faces such a scenario, we can assume that speculations in any case will set some substantial market value of dynes. Although it may be nothing but a consequence of irrational exuberance and will not correspond to the actual potential of the platform, it will still contribute to the development of Dynemix as bait for reward hunters.

b) **The influx of reward hunters**

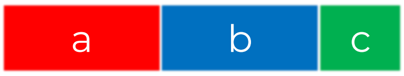


If the described scenario occurred in a typical PoS blockchain system, speculators would stake their tokens and accumulate all newly issued coins, thus keeping their share constant. Under such conditions, there would be no efficient means of escaping the speculative cycle.

In Dynemix, however, on the early stages stakeholders will receive only 50% of the issued coins, while 50% will go to DOGs. Since our platform is designed so that almost any user can easily perform DOG functions without any investment and since the same amount of reward is distributed among all the DOGs online, it incentivizes them to start using the Liberdyne messenger as soon as possible in the chase of a larger reward.

The more speculative pressure the platform suffers and the higher the market cap rises, the more attractive Dynemix becomes for reward hunters, as rewards grow higher in a fiat equivalent. Under such conditions, the influx of speculators inevitably causes a corresponding influx of reward hunters, which attracts more speculators and creates a feedback loop.

c) **Cycle of growth**

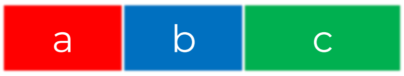


In the subsequent growth cycle, more reward hunters will be lured and the userbase will continue to grow. We should keep in mind that many reward hunters may also perceive dynes as a speculative asset, which makes their behavior closer to that of investors, but a fraction of them will assuredly transform into common users, thus forming a basis for future economic development.

The main reason for such a transformation is the decrease in the amount of the rewards for each participant with the growth of the userbase and the inevitable eventual depletion of the investment potential. At some point, rewards for DOGs will become so low that they will stop being perceived as tangible yield-generating assets. Users will start preferring to simply spend dynes for various low-value goods and services rather than hodl them in the expectation of capital gain.

3) Stable stage

a) **The point of saturation**

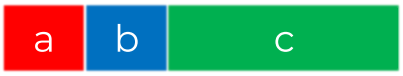


The feedback cycle between investors and reward hunters cannot function eternally, and eventually the system will reach the point of saturation, at which the potential of growth will be depleted. Unlike the inherently scarce Bitcoin supply, Dynemix features constant exponential emission, and at some point, the growth potential will inevitably drop below the inflation rate. After that point, dynes will start to lose their attractiveness as a value-storage asset and will eventually transform into a pure medium of exchange.

At the same time, the system will not always be attractive to reward hunters, as the size of the reward for each of them will constantly be reduced with the growth of the userbase. This stimulates the initial growth of the system, but in the mature state the platform becomes less attractive to reward hunters.

We rely on the fact that at that point a substantial fraction of the userbase will consist of common users, and a sufficiently saturated financial market will form within the economic zone of Dynemix, which will help the system overcome the subsequent events.

b) **Market volatility**

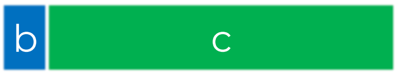


Eventually, the speculative bubble will burst, and investors will start exiting Dynemix en masse. The process of investors' exiting in the worst case will be accompanied by sharp jumps in the exchange rate. This can be quite a difficult period, but we expect that at this point the system will be in demand by a large number of consumers, and a lot of the real market infrastructure will be developed within the ecosystem endowing dynes with significant utility value.

During this process, some reward hunters will also exit, but most will likely transform into common users and remain in this role subsequently.

The economic model of Dynemix has potential to circumvent this stage: since the implementation of a decentralized central bank turns dynes into a fairly stable monetary unit, it can influence expectations of investors, who instead of exiting the ecosystem of Dynemix can prefer to convert dynes into financial assets nominated in dynes, hence keeping their capital within the Dynemix economic zone. The extent to which such a behavior will prevail depends on a wide array of exogenous factors, which is why no reliable assumptions can be made at the current stage.

c) **Stabilization**



Eventually, dynes will completely lose their investment attractiveness and cease to be of substantial interest to reward hunters. The exchange rate will stabilize, and dynes will be used

> as intended – a medium of exchange. At this point, we can implement the economic model of a decentralized central bank turning Dynemix into the first fully-fledged cryptocurrency.
>
> Thus, the architecture of Dynemix and Liberdyne is expected to ensure a smooth transition from the investment stage to the development of the real economy. At the same time, the platform should be capable of benefitting from the speculative period, using the investment capital for the initial attraction of the user audience.

VI. Payment channels and smart contracts

1. Payment channel technology summary

One of the major flaws of the first blockchain implementations was the scalability issue. Since no solution was found that could allow this issue to be resolved on the blockchain layer without sacrificing security or decentralization, the developers turned to second-layer solutions.

Payment channels are an off-chain solution that can take the load off the main layer and allow fast transactions to be made without the direct use of a blockchain.

The most commonly known implementation of the payment-channel technology is Lightning Network. It is believed to be the solution to Bitcoin's block-capacity issue.

We strongly believe, however, that it is not. The reason is that Lightning Network is not a part of Bitcoin. It is a separate payment system that can use bitcoins as transferable units, while at the same time not actually being a part of the Bitcoin protocol, apart from its opening and closing points. Technically, each payment channel is a separate two-replica SMR system with specific liveness and safety properties.

For this reason, it simply cannot solve any Bitcoin architecture issues directly. It is the same as claiming that fiat cash can solve the scalability issue because anyone can exchange Bitcoins for fiat cash, perform any number of transactions he wishes and purchase Bitcoins again in the end.

We are quite skeptical toward payment channels, and we do not consider this technology as beneficial as it is often presented.

Payment channels feature some technical restrictions that make their implementation very specific.

- Channels can be opened only between two accounts.
- Channels' capacity is limited by the amount of the opening transactions.
- Channels exist within a predefined amount of time.

How does this affect the user experience? In most cases, money is transferred one way (from a client to a merchant). Situations when back-and-forth transfers are needed are extremely rare and often can be resolved by debt-clearance solutions without the direct involvement of the payment system. Considering that a payment channel requires at least two transactions to be sent to the blockchain (opening and closing) from each party, it only makes sense to open a channel between Alice and Bob, when Alice certainly knows that she will need to make more than two coin transfers to Bob in a short period of time.

2. Payment channels network

To increase the capabilities of the technology, a relaying solution was introduced.

Suppose Alice wants to send coins to Bob but does not have a direct open channel to Bob. At the same time, Alice has an open channel to Carol, and Carol has an open channel to Bob. Alice can use Carol as an intermediary to transfer coins through the existing-channels route. Such a route can also be complex and involve multiple relays.

The routing system can unite all users in a single network and provide fast and secure off-chain coin transfers, but, unfortunately, its design features certain limitations, which are the reason we are not optimistic about this technology.

Since, after opening a payment channel, the specified number of coins is blocked in the blockchain until the moment the channel is closed, it is impractical to open payment channels just in case. This means that if Alice opens a channel with Carol, she pursues one of the following goals:

- She wants to transfer some coins directly to Carol. In this case, using this payment channel as a relay for the coin transfer $A \rightarrow C \rightarrow B$ is pointless, since it will reduce the capacity of channel AC, thus forcing Alice to open another channel to be able to settle her deal with Carol, when it would be easier for Alice just to open a new channel directly with Bob.
- She predicts that Carol will be willing to become an intermediary for coin transfers to various recipients, possibly including Bob or someone who has a channel to Bob.

From Carol's point of view, the situation looks quite similar. If she opens a channel with Bob, then she pursues one of the following goals:

- She wants to transfer some coins directly to Bob. In this case, using this payment channel as a relay for the coin transfer $A \rightarrow C \rightarrow B$ is pointless, since it will reduce the capacity of channel CB, thus forcing Carol to open another channel to be able to settle her deal with Bob.
- She predicts that someone will be interested in transferring coins to Bob and wishes to become a relay for that transfer.

This shows us that being a relay is not something that any user will be willing to do. Becoming a relay only makes sense if the user possesses a significant number of coins in the blockchain, which she is ready to block, and she can ensure that enough users will be willing to use her services.

These obstacles make relaying a purely professional activity and transform a payment channel network into something resembling a banking system endowed with the following properties:

a) **Relays take fees for their services.**

For Carol, being a relay requires possessing and blocking the number of coins equal to the overall capacity of the channels opened simultaneously to all other relays and/or clients. The only reason Carol may wish to bear this burden is to get an economic return.

A very important condition is whether the channel is opened in a PoW or a PoS blockchain. In PoW systems, users cannot earn coins for staking, so hodlers may be initially interested in becoming relays.

PoS is a totally different matter – since being a relay requires the blocking of coins, Carol will be willing to become a relay only if she can charge fees comparable to a possible income from block production. Otherwise, she will prefer to use the same number of coins as stakes for minting.

At the same time, the route between Alice and Bob may include multiple relays, each of which will charge its own fees, thereby possibly making the total fee higher than a fee for a direct on-chain transaction.

b) **Relays are points of failure.**

A payment-channels network is not a peer network, as most blockchains (including Dynemix) are.

Its overlay topology may be roughly described as hybrid star-mesh. Multiple clients are connected to a certain central relay, which forms a star topology. Then relays are connected to each other through a partially connected mesh network.

Such topology means that in the case of Carol's failure all her clients will be fully disconnected from the network and their coins will be blocked in the blockchain until the channels' TTLs expire or Carol goes back online.

Clients cannot simply solve this problem by connecting to multiple relays, since each payment channel requires the blocking of coins in the blockchain; therefore, clients will have to block a separate share of coins for each connection.

c) **Relays have censorship abilities.**

Though Carol cannot steal coins from Alice or Bob, she has strong censorship abilities, since she can refuse to fulfill her part of the deal by her own will, and Alice's and/or Bob's coins in the channel will be blocked until the channel expires. Due to the limitations of the payment-channels technology, such actions cannot be penalized.

In most cases, these opportunities do not pose a serious threat, since such behavior does not provide any benefit to Carol. Moreover, she is economically incentivized to behave correctly in order to attract more customers to her mediation services.

Carol may be vulnerable to pressure, however. The more clients Carol has and the more she risks losing, the more cooperative she will become. Considering that Carol's node is a point of failure for all her clients, we may conclude that a payment channel network is highly vulnerable to censorship.

These features make the user experience with payment channels drastically different from the first-layer blockchain. A payment-channel network lacks decentralization, fault tolerance and censorship resilience – very important features of blockchain systems. At the same time, most common users do not recognize the difference between the first and the second layer and consider Lightning Network a part of Bitcoin.

This is why we do not consider the payment channel technology a good solution to the scalability issue. The technology is not bad in itself, but the misrepresentation of its nature confuses cryptocurrency users, who for the most part do not have sufficient competence to understand all the flaws of the mentioned solution.

In defense of payment channels, we can say that, for now, they are supposed to be used mostly for microtransactions, and it seems to be the best way to use them, since payment channels can provide the necessary speed for slow blockchains like Bitcoin. At the same time, the flaws mentioned above become less crucial when the stakes are low.

3. Atomic swaps and cross-chain transactions

HTLC (the underlying technology that powers payment-channel networks) can be also used for atomic cross-chain swaps. An exchange of different cryptocurrencies can be performed directly by two parties by opening payment channels in both blockchains and closing them after the parties fulfill their obligations or the TTL expires.

The proposed solutions for atomic swaps via HTLC seem generally reasonable, as they solve the problem without intermediaries and thus keep decentralization on the level of the first layer.

Unlike simple swaps, the problem of cross-chain transactions between arbitrary blockchains requires a more complex solution.

To date, there are several concepts for resolving the matter by involving second-layer intermediaries or building a top layer above the blockchains that is controlled by a consensus of validators (layer 0), thus creating a heterogeneous multi-chain (e.g. Polkadot). This should become a decentralized solution for cross-chain transactions.

The problem is that the currently proposed protocols are not as decentralized as we would wish them to be. For example, Polkadot uses Delegated Proof of Stake as a consensus model for the top layer (Relay Chain). DPoS indeed can be faster than most PoS alternatives, but we are quite skeptical about the level of decentralization that can be achieved with such a design.

Considering that joining such systems requires modification to the blockchain protocol and granting of authority to the top-layer validators, we do not find solutions of this kind appropriate for Dynemix, and the system will likely not go beyond cross-chain swaps. The final decision will be up to the community.

4. Payment channels in Dynemix

Taking into account the above, atomic swaps via HTLC seem to be a viable option. Therefore, adding the support of HTLC to Dynemix to enable atomic swaps seems an option worth considering. The final decision will depend on the attitude of the community.

As for the payment channels themselves, there is not much use for this technology for Dynemix, since our system does not suffer from the scalability issue. Considering that payment channels form a separate network with its own rules (followed by the drawbacks mentioned above) and that recognizing the payment-channel network from the blockchain layer may be troublesome for most users, we would advise avoiding the use of this technology unless it becomes clear that it provides some really significant benefits (like the mentioned atomic swaps in the case of their mass adoption or the possibility of implementing off-chain smart contracts).

It is also worth mentioning that a full-scale payment-channel network cannot be formed on top of Dynemix for economic reasons. Since Dynemix features free transactions, there are no economic prerequisites to become a relay – if a relay charges commission for its mediation services, users will prefer sending transactions directly to the blockchain free of charge. At the same time, it makes no sense to perform the relay functions in a payment-channel network without receiving a reward, since users could engage in minting instead. The situation may change, though, if Dynemix begins to support off-chain smart contracts, which are not supported on-chain.

5. Smart contracts in Dynemix

As mentioned, implementing virtual machine technology can negatively affect the platform's stability and scalability, as well as degrade certain features that allow Dynemix to be a breakthrough payment platform.

For these reasons, we will not include any smart-contract support in the first version of the system.

If the system, however, shows a performance reserve after being tested under substantial loads (with at least 10 million active users), adding some limited smart-contract support may be considered. This should most likely not include Turing-complete code support, but rather specific templates with a limited set of variables that can actually benefit the cryptocurrency as a means of payment and at the same time not completely turn it into a decentralized virtual machine.

As mentioned, adding certain types of smart contracts can allow not only the implementing of scripted business scenarios, but also the possibility of creating payment channels on top of Dynemix and using a variety of off-chain smart contracts with the help of DLC or any other possible techniques that can be developed in the future. Of course, such contracts will have limited functionality and will not bear the same properties as smart contracts operated on the blockchain layer, due to the flaws of second-layer solutions, but nevertheless it is a viable option for implementing smart contracts without increasing the load on the blockchain.

We should note that such a decision would require a hard fork, so the possibility of implementing smart contracts will depend on the community's attitude.

VII. Decentralization

During the development of Dynemix, we paid a lot of attention to the matter of decentralization.

First-generation blockchain systems based on PoW protocols turned out to be not as decentralized as expected. At the same time, more recent projects' developers tend to trade decentralization for speed and scalability.

We do not support such an approach, and we have put a lot of effort not just into ensuring a sufficient level of decentralization but also into making Dynemix one of the most decentralized blockchain systems to date.

Let us explore what has been achieved with the help of our approach.

1) State transition

a) Safety and liveness thresholds

Since Dynemix features sharding and a two-layer consensus, the question of safety and liveness thresholds is complicated.

The system makes progress as long as at least one shard-block at each height gets approved and $\frac{3}{5}$ of minters are honest and online. We may intuitively assume that the share of the global stake pool that is required to stall the protocol lies between $\frac{1}{3}$ and $\frac{2}{5}$. At the same time, a consistent violation of safety would require roughly $\frac{2}{3} - \varepsilon$. A more precise assessment can be provided after the test period.

All in all, we may state that, from the point of view of raw numbers, the safety and liveness thresholds are close to most asynchronous BFT protocols.

We have applied a set of measures, however, that ensure the growth of the global stake pool size, including

- setting a moderate minimal stake boundary;
- securing low hardware and bandwidth requirements;
- allowing funds to be unbonded quickly;
- targeting inflation;
- adopting a model of weak responsibility.

With the adoption of these measures, the share of resources needed to affect the state transition in relation to the market cap of the platform is expected to be significantly higher than in many other blockchain systems.

b) Participants

Resisting professionalization of the state transition processing was one of the key goals of our project. Although we needed to provide sufficient speed and scalability, at the same time we managed to set a low participation threshold.

The involvement of ordinary users is ensured by providing the opportunity to participate at will without the need to hold uninterrupted availability (the model of weak responsibility) and having a low minimal stake threshold. Furthermore, the entire economic model of Dynemix is designed to favor regular users instead of minor elite groups.

c) **Initial coin distribution**

Given our devotion to decentralization, we do not plan to distribute the initial coin volume in a manner that can allow the concentration of a large share among a small group. The bulk of coins will be distributed via a prolonged public sale at a market price, which seems the optimal way to assure the widest distribution.

d) **Issuance model**

Even if certain entities manage to acquire a large share of coins at the initial stage, it will not help them keep control of the system later on. Unlike most PoS designs, which make the rich get richer (because the largest stakeholders obtain most of the issued coins), Dynemix features a unique issuance model that disperses many of the issued dynes among a large number of participants (helicopter money), thus constantly reducing the relative size of the initially acquired shares and preventing the concentration of power.

2) Hardware and bandwidth requirements (entry threshold)

During the development process, we devoted a lot of attention to optimization. Since we intended to create a truly decentralized system that was at the same time capable of processing a high transaction flow, we had to implement a number of unique solutions.

Given the implementation of quilt technology, Dynemix features two types of minting nodes: a full node and a master-node. Although master-nodes provide a number of benefits to the system, they cannot directly affect state transition, which is why decentralization is considered at the full-node layer.

Full nodes store only the current state and information about recent state changes. This does not require much storage space even under an intense load. A very important achievement is maintaining the storage and traffic overhead dependent only on the current system load, regardless of the time that has passed since launch (i.e. the growth of the blockchain does not affect the overhead). Full nodes are not obliged to store the entire blockchain and can securely synchronize with the system any moment without the need to download a complete set of missed blocks.

Sharding helps optimize the validation process on both the blockchain and quilt layers, and, according to our estimates, a mid-class home PC will be capable of participating even after the system significantly scales up.

Master-nodes store the complete blockchain with all transactions included. Running a master-node requires much more resources, but this mode is expected to be employed for professional purposes by services such as block explorers, exchanges etc. Though they store a lot of useful data, master-nodes are not crucial, and, even in case of their complete shutdown, the robustness of the system will not be threatened (although the additional line of defense will become unavailable).

Considering that full nodes are the main type of nodes that ensure system availability and resilience, we can state that Dynemix provides an exceptional level of decentralization.

3) Development

Our system features no obstacles to launching forks. Major stakeholders will not be able to compromise undesired forks; therefore, from this point of view, Dynemix's decentralization level is very high and equal to that of other open PoS systems.

4) Overall level

We can conclude that we managed to greatly increase the decentralization of every component, thus creating – without any exaggeration – one of the most decentralized platforms to date. Moreover, we have put a lot of effort into ensuring that the achieved parameters will not degrade as the system scales up with popularity.