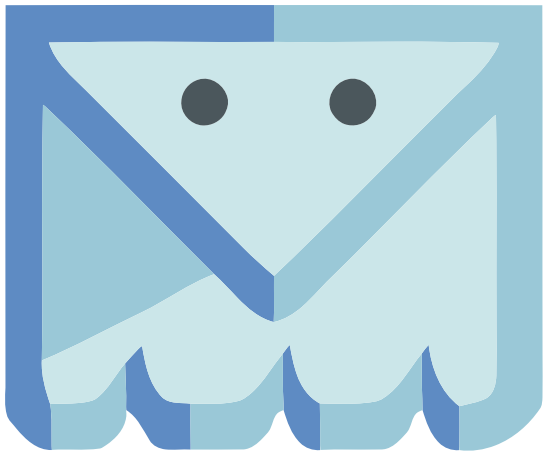
A PROTOCOL FOR DECENTRALIZED CONTENT EXCHANGE

White paper

Phantasma Protocol



Authors:
Sérgio Flores
Miguel Ferreira
Alexandre Paixão
Bernardo Pinho

version 1.0.6
2018-05-12 5:00 UTC

ABSTRACT

The recent advent of smart contract blockchain networks like NEO and Ethereum opens a huge opportunity for a vast new array of innovative decentralized services and applications (dApps). The dominant paradigm of the 'Cloud' (cloud computing, storage, services etc.) can now be seriously challenged - decentralized systems are becoming increasingly competitive by giving two main advantages: ability to aggregate an immense set of disparate small economic resources (thus gaining economies of scale), and the increased transparency and freedom from not being dependent on a trusted third party (i.e. Amazon's AWS or Microsoft's Azure).

This new computational and economic environment is however still in its infancy, and lacks a great many of the basic building blocks and infrastructure required for the full development and realization of decentralized applications, systems and services. One of the most needed components, is a service for customizable communication and data sharing between dApps.

To tackle these issues, we present Phantasma - a new generation of decentralized content sharing platform.

Contents

1 Introduction

1.1 The problem

Traditional content-sharing systems were isolated machines running in small local area networks (LAN). Since then, in the early 2000s, a new paradigm of remote centralized services emerged. These were the so-called cloud platforms, and anything-as-a-service.

This trends towards centralization and greater specialization in service-providing brought along with it substantial cost reductions and improved economies-of-scale. However, we are currently facing the limitations of such solutions. These have inherent disadvantages, such as increased risks from having a single point-of-failure, and being locked in proprietary closed systems.

A brief history of content distribution :

- In 2000, Scour Exchange was shut down. It was one of the first multimedia search engines.
- In 2001, Napster, an extremely popular P2P network was also shut down.
- In 2002, Soribada was shut down. It was the first P2P network to gain traction in Asia. It later reopened and was again shut down in 2015.
- In 2004, Suprnova was one of the first websites with a focus on torrent distribution to disappear.
- In 2009, PirateBay, which was then the leader of content distribution via torrent, was also shut down.
- In 2010, Limewire, a free program for P2P content distribution was shut down.
- In 2012, Megaupload's domain was seized, and Kim Dotcom, the owner was arrested. All users of the service lost their files.
- In 2013, Lavabit, the email provider used by Edward Snowden was shut down due to a gag order. All users of the service lost access to their emails.
- In 2014, iCloud was hacked, leading to massive leaks of Celebrities' private photos.

As long as user content is owned by a third party, this will continue to happen. The extent of this problem persists even in Cloud storage services such as Apple's iCloud which has faced multiple hacks over the years despite maintaining a centralized system with high-end security.

In the last 5 years, however, a new computing paradigm started to emerge, arising from the ubiquity and proliferation of small affordable devices (smartphones, smart-appliances), and increasing PC penetration. The widespread availability of such devices, and their under-utilization, presents a unique opportunity to leverage an enormous amount of computing resources (cpu cycles, storage, bandwidth). Recent blockchain projects have been launched to explore this market niche, such as Golem (cpu), Storj/Sia/Filecoin (storage), Rightmesh/OpenGarden (bandwidth), etc.

At the same time, a burgeoning ecosystem and market is developing for decentralized applications (dApps) running on smart-contract platforms (Ethereum, NEO, etc). These applications have many of the same computing requirements of traditional applications, compounded by the more costly decentralized infrastructure and increasing architectural complexity. In this climate, the demand for incresingly reliable and available resources is ever present.

An additional critical requirement for the decentralized applications space is usability and developer-friendliness. The remarkably complex logic workflows pose complex design hurdles to developers and systems architects, and present one of the main obstacles to the expansion of blockchain technology among end-users and companies.

One of the most challenging development spheres for decentralized applications designers is how to manage data - how to store, access and manage it ((B. Krämer, 1997)). Data is a major bottleneck in dApps, accounting for a major fraction of transaction execution costs in some popular Ethereum dApps (references). Storing real world data on the blockchain is prohibitively expensive. Managing it becomes impossible, considering how little infrastructural support there is. Developers are faced with mounting implementation details and pitfalls when trying to scale up and develop more complex and inter-connected dApps in the decentralized world.

1.2 The solution

In this paper, we present Phantasma: an innovative platform to revolutionize application development in the key area of data management and content-sharing.

1.2.1 Unique Value Proposition

Phantasma introduces a decentralized content distribution system running on the blockchain, with strong emphasis on privacy and security. Phantasma is a platform where users control their own content instead of relying on third party entities to host, secure and manage it for them.

Relying on third party services for something as critical and personal like email is dangerous. Companies can be hacked, bankruptcies can happen and government entities can force access into emails. Also, for those who are content creators, such as the ones creating videos, games, music and other monetized content. Giving up a large percentage of their revenue can be very discouraging.
Not only that, content creators are fully tied to the platform holders, who often enforce drastic rule changes which lowers their revenue, events over which creators have no choice but to accept.

Interoperability with existing systems is extremely important in order to ease adoption. Therefore, it is imperative that blockchain technology for end-users are created. Taking email as an example, Phantasma based-mailboxes will be compatible with standard email addresses, and thus can communicate with email systems outside the blockchain.

Privacy is also extremely important. So while it is distributed, all data within Phantasma is also encrypted. This ensures total control of content, with no one but the content owner deciding who can access it and how it will be distributed. In other words, ownership of the private keys of wallet means ownership of all content stored in that wallet.
Since everything is encrypted, without the private key nobody can read it or steal it from the owners, not even the Phantasma developers can access it.

1.3 The role of decentralization

Phantasma is the new standard for seamless and secure data sharing, management and integration, across a multitude of communication partners, connected users and dApps in the NEO ecosystem. The goal of Phantasma is to provide a new and adaptable framework to fill the needs for data management from dApps. By providing a uniform, standard interface (API) with all the fundamental data access semantics that applications require, Phantasma will boost the productivity of dApp developers, broaden its potential and energize the entire NEO ecosystem. The targeted focus on access control with any desired granularity supports a vast array of use cases, customizable by every user according to their unique needs.

Phantasma is an autonomous and decentralized network running as a smart-contract on the NEO blockchain. The code executing the platform is public and immutable, without any elevated permissions for the creator/admin (Phantasma team). The platform de-intermediation allows for trust-less confidence in the system, subject to timely review and analysis by community experts, isolating the health of the ecosystem from any particular localized shocks.

Once the full set of proposed software that powers the Phantasma protocol is production ready, the network would be self-sustaining and completely detached from the original developers and deployed as its own blockchain.

1.4 Why use NEO?

When considering the modern scalable blockchain infrastructure on which to develop Phantasma, we mainly deliberated between Ethereum and NEO, as these were two production-ready available smart contract platforms. Between the two, our final choice was the NEO blockchain.

We believe that Phantasma is a perfect match for NEO, and a great example to showcase the best that this blockchain has to offer. When deciding which platform to use, it came down to the following factors:

Transaction Throughput – Ethereum is still bound by Proof-of-Work, which limits it to 10-30 transactions per second. While some solutions are currently in development to increase the throughput (eg: Casper, Plasma and others), most do require some trade-offs, and more importantly, none is yet available. On the other hand, NEO has fast transaction speeds, very cheap (currently 0) gas costs, and high throughput, mainly derived from the use of Proof-of-Stake (PoS) consensus implemented with the Distributed Byzantine Fault Tolerance (DBFT) algorithm;

Second Layer Scalability – Transaction throughput scaling is a current problem in the blockchain space. Both options do propose future solutions, with Ethereum having Raiden and others, and NEO having its own proposals for future advances in scalability with the advent of sharding and state-channels (Trinity);

Ecosystem – When talking about the maturity and time-tested reliability of the system, both of these platforms are strong. NEO has an extensive development environment, a large and growing community of developers, partner support staff, and a healthy

number of applications being integrated with it. The excited and supportive user community of NEO believe in it and are looking to support nascent projects which raises awareness of NEO's competitive advantages with regards to the pioneering Ethereum;

NEOX – NEOX as cross-chain interoperability agreement. This is a very important feature of NEO and part of the reason for our final decision since it gives us a guarantee that in the future, we can move the project into a separate blockchain while still maintaining interoperability with NEO.

1.5 Phantasma Overview

The Phantasma Protocol is composed of five components that also represent different stages of the project. The component structure is designed in such a way that it will be possible to bootstrap the network from the initial components, and then gradually expand the system by developing the remaining components.

1.5.1 Blockchain

Initially, to bootstrap the network, Phantasma will exist as a service running on top of the NEO blockchain. Later on, it will be adopted to use the NEOX cross-chain protocol and from there proceed to forking Phantasma into its own blockchain. This will give us complete freedom to take the data collected during the first phase, redesign the transactions in order to obtain maximum performance, avoid GAS costs and reduce strain on the NEO blockchain.

1.5.2 Smart Contract

The backbone of Phantasma is a smart contract running on the NEO blockchain.

Messages – The main function of the contract is to handle the messaging protocol, which runs on top of a small VM allowing mini-contracts to run inside the Phantasma smart contract.

Storage – The content storage mechanism is also controlled via smart contract, which allows proper regulation and distribution of storage claims and data blocks integrity.

Tokens – Phantasma uses a token as fuel for the protocol, and this token follows the NEP5 formal specification.

Distribution – Part of the contract will be dedicated to token sale, including distribution interfaces and asset swaps.

1.5.3 Software Development Kit

A software development kit that will allow any developer to use Phantasma to create their own dApps. In technical terms, it will allow the abstraction of dApps' communication with the Phantasma smart contract, exposing all available features of the protocol and updated with support for new features and components as they become available.

It will initially be available in the C# language, with upcoming support for other languages being added eventually (eg: Javascript). Besides the actual SDK, a proper documentation full of examples will also be included.

1.5.4 Relay Nodes

Most current blockchains are bound by transaction and block confirmation speeds. While NEO is currently the fastest production-ready smart contract platform, the block times of 15 seconds will not be fast enough for certain classes of applications. The Phantasma relay will be a secondary off-chain layer sitting between the NEO blockchain and Phantasma's apps, allowing not only faster delivery of cryptographic signed content, but also act as a queue system, packing and merging content as necessary to reduce strain in the blockchain.

1.5.5 Distributed Content Storage

Most real-world data is voluminous enough to make it prohibitively expensive on save content on-chain. There is currently no perfect solution for this, but rather, there are a group of competing standards. Hence Phantasma will be backend-agnostic and it will be possible to support various backend storage systems to store the actual content of messages.

We do endorse and allow support for third-party data backends (eg: IPFS, Bluzelle), however we will also develop our own data storage mechanism, the Phantasma Data File System (PDFS), which will come in the third phase of the project and from then on will be the recommended storage backend. The PDFS will be developed as an extension of the Relay layer, and in this way gives Phantasma the capacity to store generic byte data on its own. Interaction with the Phantasma smart contract will allow storage of content hashes and a system for distributing rewards among storage nodes.

Some of the possible backends that would be usable in Phantasma:

- Core decentralized protocols like IPFS and Swarm;
- File storage blockchains such as Storj and Filecoin;
- Decentralized databases for general data (Bluzelle, Genaro);
- Phantasma Storage: our own built-in phantasma-incentivized decentralized storage;

All content resources saved on Phantasma will have a unique identifier in the form of a hash value of its contents. The hash function used varies according to each backend, i.e. the SHA-256 algorithm for IPFS, or Kademlia Hashing for Phantasma Storage.

To access a resource, a developer must first read the message containing the resource identifying hash value, and then in a second pass fetch the actual resource from the intended backend. The Phantasma SDK will make those operations as transparent as possible.

1.5.6 Products

In our roadmap we also include development of our own products along with the SDK, since designing new technology requires the ability to put ourselves in the position of the forthcoming developers.. It is only through this way that we can make sure our concepts align with real world demands.

- Decentralized Email
- Decentralized Chat
- Box Marketplace
- Oracle Nodes
- Content Streaming

Please consult our road-map to check the intended dates for the development and final releases of each product. We will expand Phantasma into two separate teams, one responsible for the blockchain tech, and other which will develop the end user apps.

2 Platform Details

2.1 Core Concepts

The Phantasma platform and framework is designed on the basis of a publish-subscribe data dissemination model, which can be described with the concepts presented in this section.

Boxes – They act as content hubs for generalized data, responsible for providing a general interface for content storage, data piping and access sharing. Each box is associated with a specific NEO address, however, for better accessibility, they will be named.
The allowed characters will be lowercase latin letters from A to Z, digits from 0 to 9 and underscore. All other characters will not be allowed by the smart contract, in order to protect against homograph attacks.

User – Any entity that uses the Phantasma infrastructure to share or consume data; This could be either a smart contract pulling data from a box, or an application (either mobile, desktop or even web based).

Messages – Data items that can be stored and shared via Phantasma, optionally linking to off-chain stored content, and stored inside Boxes;

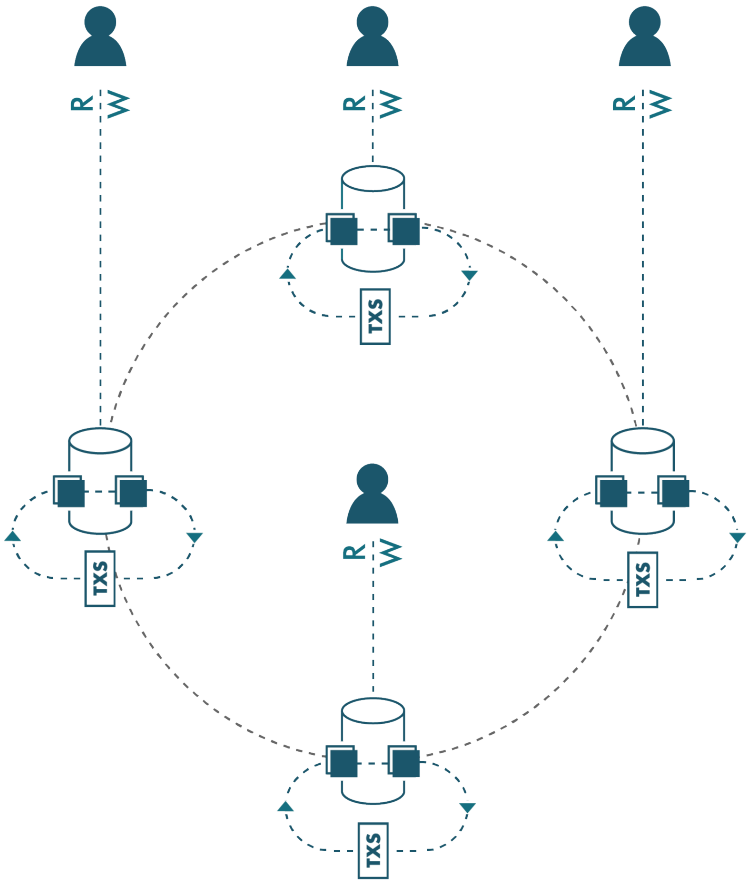
Gate – Permissioned gateway to a Phantasma Box. Gates are defined via the programmable Phantasma VM, supporting fine or coarse-grained customized permissions, thus allowing for multiple access modes for each Box. Gates can also specify different payments models, such as free-to-access, pay-once access or subscription based access;

Since boxes are linked to NEO addresses, any user can create any number of boxes, each box with its own customized gates. After creation, the user can immediately start pushing content to the box.

The access modes to a box can be fully controlled via the Phantasma VM opcodes, allowing emulation of Read-only and Write-only modes, along with time-based or condition-based access, and basically anything else that can be programmed given the opcodes and data available inside the blockchain.

Tokens – The access to a gate can be verified via token permissions - the SOUL tokens. These tokens are the fuel of the Phantasma network, meaning that operations inside the Phantasma network will require them, including creation of boxes, allocation and renewal of space for storage;

For example, to access any Box, a user must send a Read or Write request to it, which then goes through the Gate script. The Gate script can validate if the user has the required permissions to access it, and charge the predefined cost for that type of access.



These are the fundamental Phantasma entities and primitives on which the platform is built. With these building blocks, keeping this well-defined and versatile interface, it is possible to implement a vast array of practical applications - some of which we are planning to build as a showcase of the protocol.

2.2 Phantasma Virtual Machine

One of the main points of smart contracts is immutability. However sometimes having a little bit of reprogrammability is desired.

Enter the Phantasma VM - a blockchain-emulated general-purpose execution environment that puts forward the capacity to deploy sub-contracts to NEO without paying the full costs of a NEO contract. The Phantasma VM is an immediate user-friendly way to add smart-contract programmability for a marginal cost. With Phantasma, it will no longer be necessary to pay a large gas fee to deploy a small set of rules in the blockchain, instead, those programmable rules can be deployed inside of boxes residing in our smart contract. This component is what makes it possible to push arbitrary code to Phantasma boxes, creating the bridge that generalizes on-chain and off-chain storage, and enabling automatic message responses necessary for advanced use cases of Phantasma.

The quintessential example of using the Phantasma VM is programming the write and read access to Phantasma boxes, which is something can be programmed with a couple operations. Other examples include programming on-chain reactions to when new messages arrive in a specific box. Examples of features that could be programmed using the VM include a on-chain payment with automatic delivery of content or a decentralized box auction system.

2.2.1 Virtual Machine implementation

The VM runs inside the NEO smart contract, containing a large set of opcodes which include opcodes for data manipulation to Oracle reading support opcodes.

A large part of the opcode set is a direct mapping to the opcodes supported by the NEO VM ((Zhang, 2016)). This decision eases the switch towards moving Phantasma to a new blockchain with a NEOX bridge, since NEOX requires a NEO-compatible VM. Thus, the Phantasma VM is able to run directly on top of the future blockchain, completely removing the need to be emulated on the NEO VM, allowing better performance and reduced gas costs.

The Phantasma VM is not Turing-Complete by design, that is, for any set of inputs it is always possible to compute in advance when the VM is going to stop. This is necessary in order to estimate the gas cost of a specified Phantasma script, otherwise it would be possible to write a malicious script that consumes all gas of a user. Non-Turing-Completeness is achieved by limiting the opcodes available in the VM, removing particularly opcodes that could lead to loops and recursion. Note that this design choice is currently specific to the NEO implementation of the Phantasma VM and the limits could be lifted in the Phantasma blockchain if and when deemed necessary.

Appendix B contains a proposal of an opcode sheet of the Phantasma VM.

2.3 Phantasma Relay

Blockchain as a ledger system has significant advantages when compared to traditional databases. However, one major disadvantage is the throughput speed, which is several orders of magnitude slower than traditional database backends. Even with NEO's transaction throughput, most types of applications would still be prohibitively slow and costly to run on a blockchain. The natural approach is to move most resource-intensive tasks off-chain: most of the heavy computation, communication and storage requirements; while their locations and identities (endpoints) are preserved and accessed via the blockchain.

The speed of a blockchain can be roughly estimated as a function of many nodes that are required to participate in the consensus. Thus as a rule of thumb, to achieve greater scalability, we need to reduce the consensus set size. This is what NEO does by using a small set of validators. The same idea can be generalized to the use of application-specific decentralized systems with a limited number of nodes, or a limited number of connections (consensus set) per node.

2.3.1 Design Philosophy

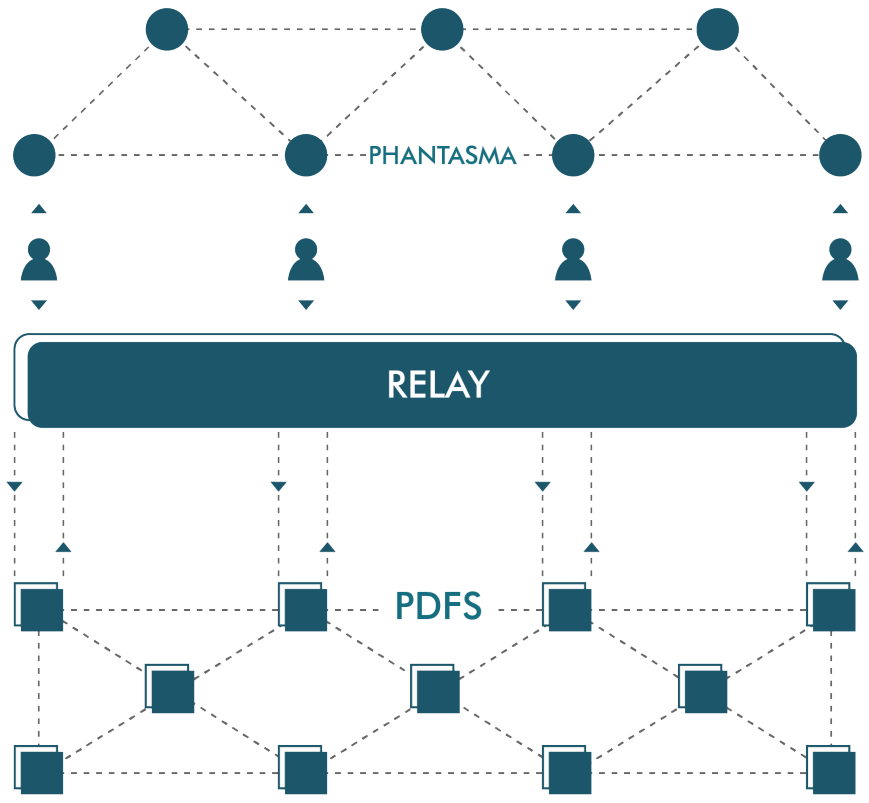
The Phantasma relay is part of our solution for maximizing the protocol's efficiency, and can be be described as a second layer scaling solution, taking ideas from existing large-scale communication protocols (eg: Skype protocol (S. Baset, 2017)).

The following points describe the design philosophy of the Relay:

- Speed as a prerequisite - Delivery of data from a single source to a ring of receivers should be done with maximum swiftness;
- Ephemerality as a choice - Not all data should be written on the blockchain, as the blockchain is a single shared entity that grows continually in size. Bloating the blockchain with data meant for single-time consumption would be a hindrance to its performance, and therefore we will allow for it to be avoided;
- Integrity and trust as siblings - For obvious reasons data should arrive intact, without partial loss of content. Furthermore, when leaving the expected trust of a blockchain one should also expect maximal trust from a subservient layer. This can of course be guaranteed by use of a cryptographically secure scheme, backed by asymmetric key algorithms;

The "Relay" can be summarized as a second layer sitting between NEO blockchain and Phantasma based applications, and capable of the following operations of top of Phantasma messages:

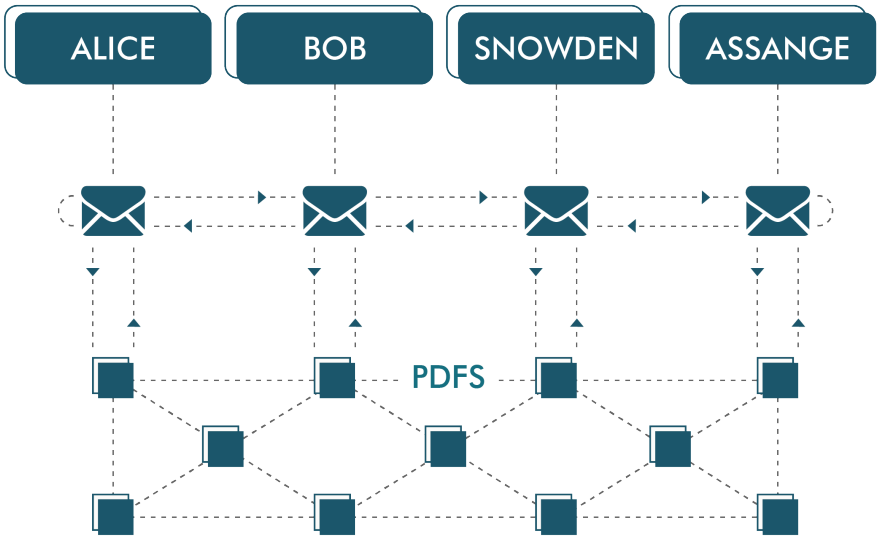
- Cache - Data can be cached at the Relay level, allowing extremely fast reads if desired.
- Merge - Data can be merged as necessary in order to minimize writes and reads to the blockchain, as long as consistency and semantics are preserved.
- Sign - Data must be signed in order to confirm the identity of message sources.



The Relay will operate running on top of UDP protocol, since it is a simple and effective low-cost protocol. It does not require the establishment of a constant connection, nor a guarantee of message delivery. These are features fit for fast data delivery to multiple recipients, and specifically useful for peer to peer applications - as is the case of the Phantasma Relay. The full specifications for the Relay's protocol will be detailed in a separate technical document.

2.4 Phantasma Storage

The Phantasma Data Filesystem (PDFS) is a decentralized data storage file system, developed to be the default storage backend for Phantasma dApps, and implemented on top of the Kademlia distributed protocol ((P. Maymounkov, 2002)) and as an extension to Phantasma Relay.
Users (running Phantasma Nodes) may further contribute to the network, and be rewarded for it, by running a full PDFS node as well. Thus all users are incentivized to participate in the token economy.



2.4.1 Overview

The content is stored as an opaque byte array, subdivided into blocks of fixed size, and for each block a content hash is calculated and added to the contract, along with the metadata to help retrieve the whole original file.

Uploading something to PDFS can be done either by the ALLOC opcode in the VM or the CreateContent() method. Both require a payment in tokens based on the number of blocks uploaded. This payment is a fixed fee in tokens.

After a successful alloc call, the user gets a valid PDFS content ID, which can be then used to upload the content to the PDFS network (check Appendix B for protocol specifications). Content upload (writes to PDFS) are validated based on the content hash: Since the hashes are stored the blockchain, whenever someone tries to upload a block to the network, it will be rejected if the hashes do not match.

2.4.2 Minimum Replica Set

It has been shown that taken the a specific set of consumer based devices, there is a wide distribution of the availability of the devices, defined in terms of fraction of available computation time per day ((J. Douceur, 2011)). Therefore it is extremely important to make sure that replicas of Storage blocks are distributed evenly over a set of endpoints, taking into account that highest priority allocations remain available in the network always. This does require that Storage nodes do a certain ordering and balancing of contents, but since the Phantasma Protocol specifies the inclusion of priority flag per allocation, it is possible for nodes to take an aggressive stance pushing low priority blocks to low reputation nodes if they are running out of storage space to keep high priority blocks.

The general assumption must be made that one or more Phantasma nodes must contain a certain content block for the whole content to be retrievable at any time. If there are N blocks in a specific content, the network may require K replicas of that block as a minimum to accept the block, where K must be greater than or equal to N. When K equals N, a single failure would signify a temporary erasure of the block. When $N = 2$ and $K = 4$, the content block is still retrievable if any two of the four replicas go offline, and as K grows toward ∞, the reliability of the block existence grows toward 100

2.4.3 Proof-of-Retrievability

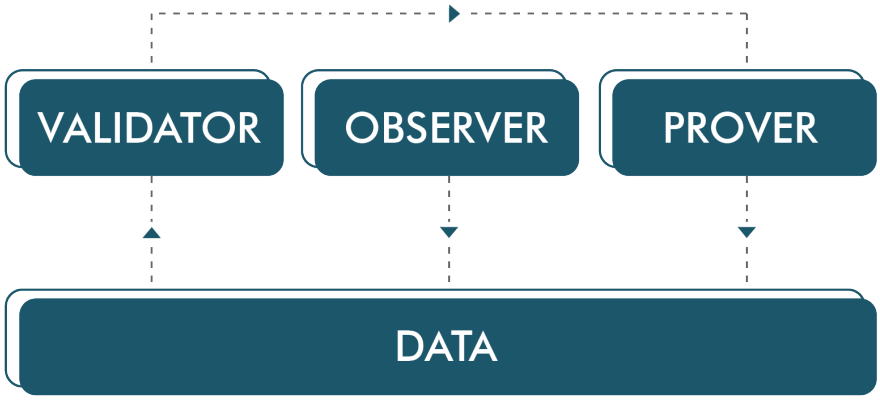
The PDFS network uses a Proof-of-Retrievability algorithm (Bowers, 2009), in which someone who stores content must provide proof that they are still the have full content available for retrieval. Proof of Retrievability is an area with extensive research, including the works of Kevin D. Bowers, Juels-Kaliski and Shacham-Waters.

In the case of Phantasma we propose a model based on a split consensus group derived from a reputation counter, which is calculated via an on-chain moving average. In this model entities in the network can start a process of verification of other entities, which involve a large chain of steps consisting of interactions between both two nodes and each of those nodes and the smart contract running in the blockchain.

Role groups

Each node in the system can fall beneath one of three roles:

- Observer - A node who can only fetch content and does not store content itself;
- Prover - A node who stores content blocks and does require proving the validity of those blocks, with incentive of receiving tokens for correct proofs. This node reputation is lower or equal to that of the current system average;
- Validator - A node who stores content blocks and must validate other nodes with incentive of receiving tokens as compensation. This node reputation is greater that of the current system average;



These roles are not static, instead they can change at any point in time, depending on the moving direction of the ecosystem as a whole and the actions of the individual nodes. Every time new proofs are submitted and the rewards are mined, the reputation of the participants is updated. The system global average threshold point is calculated as follows:

$$\lambda = \lambda 0 + \frac{R - \lambda 0}{N}$$

Where $\lambda 0$ is the previous global average, R is the current reputation value of the last participant and N is the total number of participants involved. The λ value is then used by the smart contract to decide who can request proofs as a validator (those with reputation above the λ value) and who can submit the answer to the proofs, which are those with a reputation under or equal to λ.

Phase I - Verify

1. A node ("initiator") picks another node from list of its connected peers. The selection scheme is up to the node implementation. The initiator must be part of the "validator" set.
2. Asks the selected node to list its current available contents.
3. The selected node answers with a list of content IDs.
4. The initiator node selects one of the content IDs and fetches list of blocks that form the specific content.
5. The initiator calls StartProof(this, target) via smart contract, and at the same time it tells the target node which (contentID + blockID) was chosen from the received list.
6. The target node must now send the full ordered bytes for the requested block.
7. The initiator hashes the received data and if the hash matches the expected value obtained in the previous step, it sends a random 256 bit nonce to the target node.
8. The target node should now send the received nonce to the blockchain with ShowProof(initiator, this, nonce).
9. The initiator calls AcceptProof(this, target), and the smart contract adds both the initiator and the target to the rewards queue.
10. Optionally, if the slave does not provide the block or provides invalid data, the master should call FailProof(), which is available only after 5 minutes passed since StartProof().

Phase 2 - Reward

When proofs are accepted in the verification phase, the validator participants are submitted to a queue, which has a maximum limit of 128 participants, while the proofer participants are sent to a second queue with same size.

Forcing the flushing of the proof queue is done by calling ObtainRewards(), which can done by anyone invoking the smart contract.

The are a few rules to collect the rewards, otherwise nothing will happen until ObtainRewards() is called again.

(i) Minimum of 5 participants in the queue.

(ii) All signatures/addresses must be ordered from lower to greater

(iii) All signatures must be different. This allows a guarantee that every participant in the queue is unique and is helped by the previous rule.

(iv) All addresses in the validator queue must have above average reputation at the time of their inclusion.

(v) All addresses in the proofer queue must have reputation less or equal to the reputation average at the time of their inclusion.

(vi) Enough time has passed since the last rewards

(vii) Every participant in the Validator queue has paid 1 token as validation fee

The rewards are also time-based, with variable rewards times helping preventing spamming attacks. The reward wait time increases every time ObtainRewards completes with success.

If all conditions are full, one participant is selected randomly from the Proofer pool and another from the Validator pool. The randomization is done via Mersenne twister having the NEO consensus pseudo-random value act as the seed, concatenated with a signature from address who invoked ObtainRewards.

Once both participants are chosen as winners, half of the tokens are distributed to each participant and a new round of rewards can now start.

2.4.4 Content Privacy

In Phantasma, all the data is secure by design, achieved via state-of-the-art encryption algorithms. All content is securely stored in an encrypted storage backend, whose records are referenced in the Phantasma smart contract. To access the actual content, users must have the appropriate permissions to be able to receive the actual decryption keys for the stored content.

Data encryption on Phantasma is backend-agnostic, running on a top layer on top of raw storage and using symmetric ciphers such as AES. Phantasma Storage uses AES-256-CTR with SHA-256 for key generation. Distribution of keys for access groups is based on the concept of broadcast encryption ((Naor, 1994)). The algorithm behind broadcast encryption requires additional coordination between the group participants but also has the added advantage of possibly sending a message to millions of receivers, without requiring a sub-linear growth of the ciphertext.

Each write and read goes through a storage node (i.e. a Storage-bridge or full PDFS node), which is responsible for encrypting and decrypting the data, and also generating and storing the data encryption keys (AES keys). To ensure the privacy of the data, the nodes cannot have direct access to the encryption keys. Therefore, the keys are kept privately only in the Phantasma Node of the creator and owner of the data, which will then encrypt and send the keys to every consumer/reader on demand. A ReadData response carries with it the particular AES key, encrypted under the reader's public key (NEO key, ECC algorithm). The key-management layer is a crucial and required component of Phantasma Nodes, responsible for interfacing with all supported data backends.

2.4.5 Storage Cost

Allocating content in the blockchain has a fixed storage by data block, and this storage is valid for a fixed period of time (eg: one month). When space for content is allocated, it is necessary to specify both a priority and a storage mode. The mode is simple, blocks can either be read-only or writable, and in the latter, the associated SOUL value of storage is greater. For the former, it represents an integer number specifying the inherent importance of the content. By specifying different priority levels, it is possible to adjust the storage cost, raising it for content that should never be deleted, and decreasing the cost for more volatile and violative content.

The prescribed value of SOUL for storage can be upfront, meaning an up-loader can submit a per-determined SOUL for the content to reside in the chain for 1 month or 1 year or more, with the number of required SOUL adjusting accordingly. It is also possible to prescribe the minimum SOUL deposit first and later amend the storage allocation period by additional SOUL installments.

2.4.6 Failure Handling

In a system like Phantasma, there are two main parts that could be attacked or suffer from technical outages:

- NEO network. While NEO was built to be an extremely fault tolerant blockchain, in the end, NEO is still under development and sometimes the network gets a bit stressed when certain heavy conditions are triggered. In this situation Phantasma could still continuing function partially, taking support from both the Phantasma relay cache system and caching mechanisms of NEO local nodes. If the failure is of a short duration, the Phantasma ecosystem can possibly withstand it with minimal disruption.
- Phantasma Relays. Since any device running a Phantasma dapp can potentially act as a Relay node, a situation where most of the nodes are offline is difficult to imagine. However it is still possible that a number large enough of nodes are offline and in turn fetching specific blocks becomes impossible. Due to the nature of a decentralized storage system, this is a situation that needs to be accounted for, and it is recommended that those who need to access certain data blocks make sure that they are running their own Relay nodes, with the specific blocks flagged for local storage.

2.4.7 Specific network attacks

Denial of Service – A distributed denial of service attack has serious disadvantages when targeting a distributed system, mainly because there is not a single point of failure that can be targeted. Targeting specific nodes is a possibility, but Phantasma Relay nodes should be able to detect abnormal network activity and immediately start distributing their blocks to other Relay nodes, taking block priority into account. While in theory it would be possible to target every single Phantasma node, the resources to carry such out would be of an order-of-magnitude higher than power available to most organizations.

Identity Hijacking – Due to the way that Kademlia exchanges messages between nodes, it is possible for a malicious node to hijack messages intended for other nodes. Similar to what the Storj protocol does (S. Wilkinson, 2017), we extend Kademlia to encrypt the messages, using the NEO ECC key pair, which allows for verification and helps to avoid identity hijacking.

Protocol Poisoning – A malicious entity could try to have a large number of nodes join the Phantasma network under their control, and then try manipulate the Storage Proof-of-Retrievability by brute-forcing the first phase of the proof between the controlled nodes. However since tokens are only distributed in the second phase, the block confirmation conditions give preference for peers with oldest minting timestamps, which pushes any puppet nodes to the back of the queue. Creation of new malicious nodes is also prevented by specifying minimum period from registration until first block inclusion.

2.5 Phantasma Blockchain

2.5.1 Overview

Some of the current available blockchains go beyond mere currency transaction systems, and instead work as platforms for distributed applications via programmable smart contracts. This is the case of Ethereum and NEO, which also supports a storage mechanism used to save application state. This state can be described as quasi-immutable storage, given that while smart contracts are free to rewrite the state as many time as necessary, all previous states can be recovered from the transactions stored in the immutable blockchain. However, to foster a healthy growth of the space required to store the blockchain data, current platforms do encourage a frugal use of the storage mechanism. In the existing model all operations done inside smart contracts have associated fees in the platform gas, and the cost of writing to the storage is proportional to the amount of bytes stored. Storing a single kilobyte in a blockchain can be extremely expensive given current gas prices, therefore currently the kind of applications that be done in a distributed way is limited to a subset of apps that have minimal storage of data as a requirement.

Our main motive behind creation of a new blockchain platform is to introduce a new model for storage in smart contracts. Phantasma proposes a model where storage cost and location can be offloaded in a distributed way to the participants of the network, instead of forcing full redundancy of the whole state in every network node. Not only that, but we also introduce the concept of ephemeral storage, in the sense that a smart contract can choose how the data should be handled inside the network, forcing either a higher level of redundancy to keep static content alive, or dropping data that no longer is necessary by the network.

2.5.2 Smart contracts

Current smart contract languages are usually turing complete, however they are still limited in the way that they can operate with data. Manipulating large amounts of data is currently out of the scope of existing smart contract languages. The Phantasma blockchain has a focus on data and thus the following capabilities are supported:

Data Manipulation – In order to manipulate large amounts of data, it is necessary to be able to operate on data in an abstracted way. High-level concepts like Map-Filter-Reduce will be available in Phantasma smart contracts. **Storage** – Phantasma smart contracts will be able to receive large amounts of data as input, and either allocate storage for them in the blockchain or drop it as necessary. **Encryption** – Encryption and privacy are basic requirements of distributed applications, and Phantasma smart contracts will have access to the underlying protocol that allows distribution of encrypted content to specific parties. **Token** – Any smart contract platform has ability to issue tokens, and with native token support in the Phantasma blockchain, every token is a first class citizen of the platform.

The programming of Phantasma smart contracts will be done using the same tools used for the NEO network, due to adherence to the NEOX protocol, which guarantees compatibility between contract virtual machines.

2.5.3 Migration

Since the Phantasma project will initially start using NEO as the operating blockchain, later on it will be necessary to migrate to the new blockchain when the mainnet is launched. NEOX allows transferring of tokens between blockchains, therefore this feature will be used to transfer user tokens. Applications would have to upgrade in order to operate in the new blockchain. However, any Phantasma application still running on the NEO blockchain would continue to operate, albeit lacking access to the new features introduced by the new blockchain.

3 Third-Party Development

3.1 Application and Development Framework

Developers of Phantasma-based dApps will use our supporting software development kit (SDK) in order to access the Phantasma features. The SDK handles the communication between Phantasma nodes and the dApps, allowing the sending of custom transactions to the Relay nodes, interfacing with the underlying Phantasma smart-contract.

Features to be supported by the SDK:

- Create new boxes
- Send and read messages from boxes
- Opcode script builder to generate logic gates deployable to Phantasma VM
- Allocate storage for content, as well as upload and retrieve it
- Communication with Relay nodes
- Programmatically participation in the Proof-of-Retrievability

3.1.1 Available Programming Languages

The SDK will be initially written in the C# language, allowing it to be consumed by any language in the .NET platform. This is the language where our developers have more experience. However, since Phantasma defines a standard for a dApp protocol, later on, other developers can contribute with their own implementations of a Phantasma library written in any other programmable languages.

3.1.2 Service API

An API for interoperability with other languages and services is essential for a platform, and Phantasma will provide that. The SDK will include some thin-client functionality to be able to provide a clear and complete service API, so that external applications can also interact with Phantasma dApps. Plus the Phantasma website will provide an extensive API reference complete with code examples to allow any developer to integrate their own products with Phantasma.

3.1.3 Debugging

The SDK will also have debugging utilities to help developers easily get Phantasma dApps up and running, with the integrated debugger based in the work our team achieved before with the NEO smart contract debugger.

Using a debugger that supports blockchain emulation allows very fast development times without requiring deploying to a private or test network.

The interfacing with the NEO blockchain will be accomplished using the neo-lux client library, which we have also developed.

3.1.4 Smart contracts

Initially only sub-contracts will be available, however once the Phantasma blockchain test-net is released later, full smart contracts will be available. Interoperability with these smart contracts will be available via SDK, as well as executing transactions and transfer of tokens.
Development of smart contracts for Phantasma will take advantage of the tools and enviroments developed for NEO, thanks to the usage of a compatible virtual machine.

4 Use Cases

4.1 Decentralized Email

With Phantasma, we can easily create a full-fledged email-like centralized system. The Phantasma-based email application has the advantage of being more secure, configurable by the Phantasma VM script, with the data being completely in control of the inbox owners .

Phantasma email can be implemented with a single box for input, with an optional additional box as output, where sent messages would be stored. The emails are then sent on the NEO blockchain to the destination's input box.

Additional behaviour for the basic Phantasma email client can be implemented: automatic newsletters or action triggers on some input messages (i.e. autoresponders), etc.

A Phantasma-based autoresponder offers an advanced level of easily customizable and cheap efficiency. With a familiar Mailchimp-like UI (user interface), the powerful Phantasma VM would allow for specific behaviour to be click-configurable as triggers on the Input box, thus allowing for automatic email responses or complex state transitions as a fraction of the cost of the current available solutions.

Another interesting application is for distributing newsletters. A Phantasma-based Newsletter could be easily and efficiently implemented as a simple box with a cost-free, permissioned gate. The recipients of the newsletters would be added to the gate's access list, and get cost-free (in both SOUL tokens and gas costs) access to the newsletter.

Finally, another important point would be the communication with traditional email systems. While existing blockchain-chain email based systems do exist, they suffer from being a closed-system with no proper communication with the exterior systems. Phantasma proposes the creation of "Mail providers" which would run Phantasma nodes as bridges to the SMTP/IMAP protocol, allowing messages to move from and to the blockchain. Of course, spam is the largest deterrent for this, therefore the bridges would work based on a mix of provider-based and user-based whitelisting of domains, with support for auto-inclusion of PGP signed messages ((D. Shaw, 2007)).

4.2 Oracles

Phantasma makes it very easy to provide oracle-like services on NEO. Therea are 3 methods available to do an oracle, depending on the actual data volume and requirements:

- Fully Onchain: for low data and query volumes, a fully onchain approach is well suited. The data is all written and read on the NEO blockchain. A possible application of this method would be a provider of certain daily quotes (i.e. closing day quote of the NYSE).
- Onchain with Backend Storage: similar to method 1, but when dealing with more real-world levels of data volumes, an application could use one of the available Phantasma backends to store it's provided data.
- Endpoint only with Phantasma Relays: this method is the solution for high-throughput data flows. Applications that need to be constantly reading or writing data should use this solution. Here, the Phantasma box will hold only an endpoint ID address with the actual Phantasma relay providing the data. A reader/client then needs to have its only relay running, and connect to the endpoint to start receiving data.

There are multiple kinds of applications for this type of data: trading engines, meteorological data providers, etc. In the future, we are planning to make a Phantasma oracle feed source aggregator, which would work as a functional marketplace for data sources, allowing both new developers to find and use existing contents and other developers to sell access to their own box contents. And of course, for systems based on oracle pattern, it is absolutely necessary that the data being pushed to the blockchain comes from a trustful source, so a reputation system on the marketplace could improve the viability of the concept.

4.3 Digital Commerce

Digital Commerce is also a good fit for the Phantasma platform, allowing sales of digital products, be it videos, videogames or other files.
By taking advantage of the programmable sub-contracts that can run in the Phantasma VM, it would be possible for example to create a box with the permissions set in a way that anyone could read from it, but only owner could write, and use this box as a product listing. They could then create a second box, where anyone could write messages but only the owner could consume, allowing purchase orders to be sent out to this box, and the content to be automatically unlocked by the Phantasma Storage.

4.4 Video Streaming

If you are a content creator that usually releases videos for free, you can host your videos in the network, allowing you to decide who will see your videos. There will be no more days of bothering your followers with unwanted ads, with Phantasma you can host the videos for minimal costs and have full control of how to monetize your content, regardless of the types of videos you are creating, with no demonetisation for certain types of

videos or control by a third party.

For providers who supply paid content to their followers, whether it is in the form of courses or specific paid videos, it would also be interesting for them to use the Phantasma network by taking advantage of the programmable sub-contracts to automatically distribute access to video content and even allowing video previews by restricting access to a partial amount of storage blocks.

While platforms like Udemy and Teachable have a business model that provides you with that capability for a significant percentage of your sales, with Phantasma these costs are reduced tremendously due to the power of the blockchain, which is based on the amount of content that you have to host and can be subsidized by helping hosting the content yourself or having the viewers participate in the hosting. This provides a much better scaling solution for creators with large audiences and the cost of hosting the content becomes significantly lower.

In technical terms, video streaming is supported via seekable random access in the PDFS layer, possible due to storage being segmented in fixed size blocks. Fast transfer speeds are achieved via the use of the Phantasma Relays, which can use the caching capacity to deliver pre-fetched blocks at high speeds.

5 Ecosystem

5.1 Data Economy

Phantasma aims to create an incentivized environment to securely distribute and monetize data for dApps. To achieve this goal, utility tokens are the critical fuel that transmits value from producers to consumers to keepers of data, and allows any developer to create an application with a built-in economic system.

Producers – Anyone who has valuable data to provide to the ecosystem, to a specific dApp(s). These can be individuals, part of a coordinated team/organization, or even a general provider of data feeds.

Consumers – Users of dApps which required data to function. In a P2P application like email, each consumer will also be a producer.

Operators – Users running Phantasma full nodes to ensure the network's reliability, availability and speed. More details below.

5.2 SOUL: Phantasma Network token

The SOUL token is the fuel of the Phantasma network, serving as the basis for all the economic incentives binding consumers, producers and operators together. The name "SOUL" was chosen as an allegory of traveling between gates. As new applications and use cases are developed within the network, the SOUL token will provide a backbone for the economy of the network, so that developers and users of the network can use SOULs for future applications and use cases.

Phantasma encourages the natural flow of then tokens between users: consumers need tokens to pay for data, these tokens go to producers and developers, who need them to pay for storage and network services. From there, the Node rewards are redistributed to all node operators, which should expectedly include many simple users / consumers. Thus the ecosystem is designed to foster the increasing use of SOULs among all participants in the network, with an appreciating outlook for the token value.

For ease-of-use and compatibility with current NEO-based software this token will be NEP-5 based. For those unfamiliar with the concept, NEP-5 defines the standard interface that these tokens conform to, and need to implement, and is the NEO equivalent of ERC20 Ethereum tokens.

Besides following the NEP-5 specification, the token will also support a distribution pool that is essential to the reward claiming functionality of PDFS layer. The distribution is described in detail in the Proof-of-Retrievability section.

5.3 Infrastructure Incentives

Phantasma users can also contribute to the healthy running of the system, by allowing their own device storage to be used partly by the decentralized network and being rewarded for it by receiving tokens in return.

In order to support the redistribution of tokens to Storage participants, the Phantasma smart-contract includes a "Storage pool", which when combined with the Proof-of-Retrievability, allows a fair stochastic distribution of tokens among node administrators.

The ecosystem follows a token renewal system where every payment follows this split system (subject to change in order to maintain market equilibrium):

- 95% of the paid tokens is sent to the payment target.
- 5% of the paid tokens is sent to a reward pool.

The payments and rewards are managed by the Phantasma smart contract, which records the token balances of all users of the system, as any standard NEP-5 implementation does. The rewards are sent instantly to the addresses of the participants, again following the NEP-5 standard token transfer mechanism.

Specifically, the payment in tokens is required for CreateBox, AllocStorage and optional for ReadContent, WriteContent calls.
In the later two calls, the amount of tokens required can vary from zero to any amount, and that amount is determined by the pre-defined code path in the Phantasma VM and can be previewed using the InspectContent call.

5.4 Ecosystem Initiatives

In preparation for the official launch of Phantasma, we will run a marketing campaign with rewards to stir initial interest from the crypto community, and allow us to raise the initial awareness about the platform.

After the platform is launched, there will be an incentivisation program for 3rd party Apps developed on Phantasma, and community programs for organic social marketing to raise awareness of the platform. These initiatives will allow the team to work together with the community to develop Phantasma.

Our planned activities for community growth will include competitions for PoC's of Phantasma based-dApps (and later development of those). Any developer is welcomed to participate and a pool of rewards will be distributed to the winners of each competition. We intend to aggressively pursue partnerships with leading blockchain developers and companies, by showcasing the advantages of using Phantasma for the basic infrastructure.

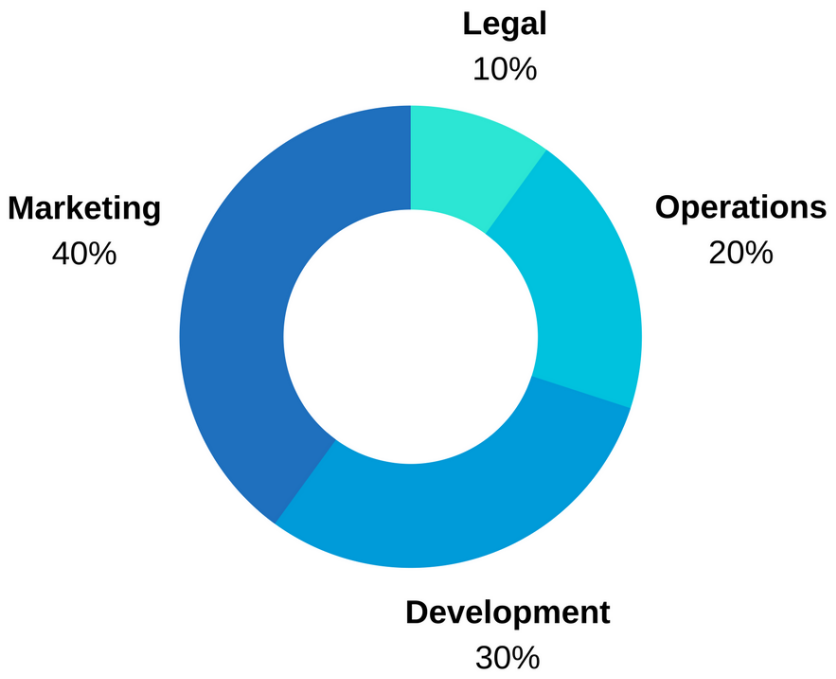
6 Project Funding

6.1 Distribution

To fund the full development of the Phantasma network and enable us to fulfil its potential, the team will do a single funding round, with a private and public token sale.

While the majority of the tokens will be available to the public during the public sale, we will be required to reserve SOUL available as rewards for third-parties who contribute to the development and improvement of the Phantasma ecosystem.

6.1.1 Projected Use of Contributions



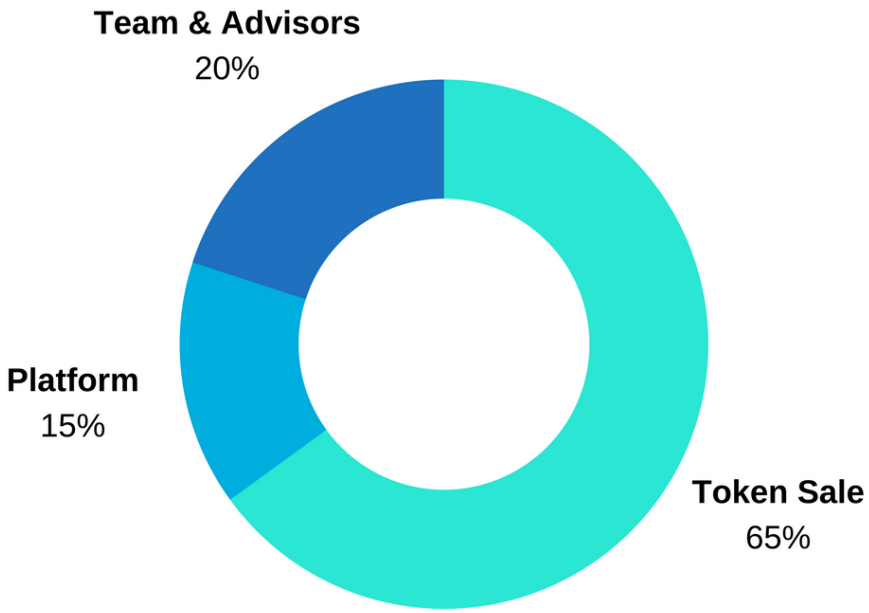
- 40% reserved for Marketing and Growth – Marketing funds to start and expand the platform, with advertising, sponsorships, partnerships, conferences, reward programs, contests.
- 30% reserved for Development – Used to continuously develop the tech that powers the Phantasma platform and its dApps.

- 20% reserved for Operations – Administrative and overhead costs for running the business.
- 10% reserved for Legal and Contingency – In order to maintain the desired high standards and fluidity of operations in the company.

6.1.2 Token Metrics

- ICO Token price: 0.23 USD
- Token Total Supply: 100 000 000 SOUL
- Tokens available for sale: 65 000 000 SOUL (of which 1/3 are Bonus tokens, that will be burned if unclaimed)

6.1.3 Token Distribution



- 65% available for public and private sale;
- 20% reserved for team and advisors;
- 15% reserved for platform growth and developer incentives;

6.1.4 Sale Details

- The tokens will be distributed in a sale done with a NEO smart contract. The smart contract will have support for pausing and postponing a sale in case of adverse network conditions.

- In order to have access to the sale, interested parties will have to register in the Phantasma whitelist, and participate in a KYC process in order to get their NEO address approved for sale.

- The smart contract will support assigning different tiers to each participating NEO address, in order to apply different rules for team members, strategic partnerships and public sale buyers.

- If the hard cap is not reached during the crowdsale, the total token supply will also be lower, as the remaining tokens will not be minted (equivalent to an unsold token burn).

6.1.5 Lock-up Period

The smart contract will have a lock-up period (also known as vesting period) for each type of address tier. Any party with vested tokens can unlock their tokens after the vesting period was reached, by calling UnlockTokens via smart contract.

Team tokens will be locked with a tiered vesting period as following:

- Cliff of 6 months
- Vesting of 3 years, with 10% unlocked every 3 months, for a total of 30 months

Advisor tokens will be locked with a tiered vesting period as following:

- Cliff of 2 months
- Vesting of 12 months, with 10% unlocked every month, for a total of 10 months

For the addresses belonging to presale contributors and strategic partnerships the following rules apply:

- 100% of base (non-bonus) tokens will be released and transferable immediately after the token sale
- Bonus tokens will have a vesting period of 90 days, vesting linearly for roughly 3 months, with a third of bonus tokens released every month

For the addresses belonging to public sale contributors, their token share will be split into a single part, and lock-up period will be the following:

- 100% of tokens will be released and transferable immediately after the token sale

6.1.6 Transparency Policy

The contribution wallets will be of public knowledge and can be tracked through blockchain tracking platforms for NEO. Ongoing reports on the progress and development of the network will be provided regularly to the public, along with public access to the Phantasma Github, which will allow developers and investors alike to be up-to-date with the network's development. A required portion of the the contributions raised will be converted into fiat currency in order to have runway for at least two years of platform development.

7 Roadmap

7.1 Planned Roadmap

The Phantasma project will have two separate teams, one working on the core tech and the other building working products. These products will be open-source and will showcase the power of the Phantasma tech.

Q3 2017
Proposal and design of Phantasma. Development of initial Proof of Concept.

Q4 2017
Submission of Proof of Concept to City Of Zion dApp contest, where it was one of the top 5 winners. Website launch and team formation.

Q1 2018
Initial draft of the white paper. Presentation of Phantasma at NEO devcon in SF. Continued development of the core protocol.

Q2 2018
Phantasma public token sale. Prototype of the first Phantasma dApp : secure email.

Q3 2018
Release of Phantasma SDK, which will let any third-party developers start building their own Phantasma based products, along with release of Oracle Nodes, as an example of SDK usage.

Q4 2018
Phantasma relay nodes will finished and released, allowing Phantasma apps to break free from the transaction speed limits of blockchain. The SDK will be upgraded to support the new feature and Phantasma Chat will be developed to showcase the Relay nodes.
A development competition will be held at this point.
Start of split of Phantasma from NEO network into its own blockchain, using the NEOX protocol for cross-chain interoperability.
Testnet will be released.

Q1 2019
Release of Phantasma Storage, as expansion of the Relay system. The SDK will be updated to support Storage.
Release of main net, native tokens to be issued according to NEP-5 ownership. Porting of the SDK to support native blockchain. Storage nodes will start being rolled out to public, so that anyone can take part in the Phantasma network.

Q2 2019
Phantasma Digital Commerce will be released to showcase how to create more complex

logic with the programmable blockchain gates.

Q3 2019

A service to discover Phantasma dApps will be created from point, forward, once the third-parties developer support has reached critical mass.

Q4 2019

Phantasma video streaming will be released to showcase how to do a dApp dealing with heavy data loads.

8 Team

8.1 The Phantasma Team

The team is mainly composed of Portuguese developers with experience in blockchain, smart contracts, mobile and web development. The team is based in Lisbon, Portugal.

Sérgio Flores
Co-Founder of Phantasma, City of Zion developer and blockchain consultant.
Sérgio came up with the idea for Phantasma after extensive development work on NEO, when he saw how cloud-hosted content were so vulnerable on traditional centralized servers, and how NEO could support a system to solve this. Sérgio is a senior experienced developer with 20 years of business and customer facing software development, and a track record of tackling complex structural systems (filesystems, drivers, compilers, debuggers - which was the case for his recent development, a NEO debugger).

Miguel Ferreira
Co-founder, senior 'bigdata' and compiler developer and blockchain expert.
Miguel is a senior experienced developer, with years of entreprises systems engineering and academic research on distributed systems. A crypto enthusiast deeply involved with Blockchain and the dApp ecosystem, he sees Phantasma as a new frontier in bringing scalable application development to the people. NEO and Ethereum (Solidity) developer.

Alexandre Paixão
Marketing expert and tech enthusiast.
With a background in management and business development, Alexandre complements the team on the non-technical side of business growth and brings important skills like Marketing and Public Relations to the table, which are key to the increasing awareness and development of the Phantasma platform.

Sérgio Pereira da Silva
Business developer, Fiscal and Legal lead
Economist by trade, crypto lover, Sérgio Pereira is responsible for building valuable partnerships with the larger NEO ecosystem, and evangelizing for Phantasma real world use cases among companies worldwide.

Bruno Freitas
Software developer
Graduated in Computer Engineering, he entered the crypto space as a hobby in early 2017. Found "AntShares" (NEO now) project and immediately got interested, which resulted on being accepted as a City of Zion developer for his work. Experienced with C# and .Net stack technologies. Specialized in cross platforms applications, Bruno is in charge of the mobile development of Phantasma.

Bernardo Pinho
Software developer, QA/Tester
The most recent and youngest member of the team, Bernardo is a crypto enthusiast and passionate developer. A recent graduate from a top engineering university, he is completely devoted to blockchain development, with previous experience on the Ethereum ecosystem.

Rafael Barbosa|| Graphic Designer
Rafael from an early age began to work as a graphic designer attending to ESAD Matosinhos, one of the best schools in Europe, and developed projects for international companies such as NOS Primavera Sound Porto and others. One of his major interests in the design world is helping business promote themselves effectively.

A Phantasma VM instruction set

A.1 Instruction Set

The VM uses a stack-based architecture, in the same vein as NEO and Ethereum virtual machines.
Every instruction occupies one byte, with the exceptions being annotated. The list included here is not an exhaustive list, but an implementation proposal that can later change.

States

VERBS
Write – True if writing
read – True if reading
inspect – True if inspecting
own – True if the caller is owner of box

Boxes

this – Address of current box being executed
entry – Address of box who started the execution

Data

timestamp – Timestamp obtained from blockheader
consensus – Pseudo-random number obtained from block header

Note - The state changes whenever the context of the VM changes, eg: if calling PULL, the context switches so ”read” is now true and things like ”own” will change. The state reverts back whenever control returns to a previous context.

A.1.1 Opcodes

Core

PUSHDATA – size, data
Pushes a string of bytes into the stack

PUSHX – number
Pushes a number into the stack

Arithmetic

All arguments are interpreted as BitInteger.

ADD
Adds the two top values in the stack. The result is pushed back to the stack.

SUB
Subtracts the two top values in the stack. The result is pushed back to the stack.

MUL
Multiplies the two top values in the stack. The result is pushed back to the stack.

DIV
Divides the two top values in the stack. The result is pushed back to the stack.

MOD
Divides the two top values in the stack. The remainder of the dividison is pushed back to the stack.

MIN
Takes the minimum value of the two top values in the stack. The result is pushed back to the stack.

MAX
Takes the maximum value of the two top values in the stack. The result is pushed back to the stack.

String

CAT
Concatenates the two top values in the stack. The result is pushed back to the stack.

LEFT
Keeps only the bytes left of the top value in the stack. The result is pushed back to the stack.

RIGHT
Keeps only the bytes right of the top value in the stack. The result is pushed back to the stack.

SIZE
Puts the size of the top value in the stack. The result is pushed back to the stack.

Logical

All arguments are interpreted as BigInteger.
EQUAL
Value comparison of the two top values in the stack. The result is pushed back to the stack.

NOT
Bitwise negation of the top value in the stack. The result is pushed back to the stack.

AND
Bitwise AND of the two top values in the stack. The result is pushed back to the stack.

OR
Bitwise OR of the two top values in the stack. The result is pushed back to the stack.

LT
Boolean Less-than comparison of the two top values in the stack. The result is pushed back to the stack.

GT
Boolean Greater-than comparison of the two top values in the stack. The result is pushed back to the stack.

LET
Boolean Less-than-or-Equal of the two top values in the stack. The result is pushed back to the stack.

GET
Boolean Greater-than-or-Equal of the two top values in the stack. The result is pushed back to the stack.

Control

SKIP
Jumps forward N bytes if the top value in the stack evaluates to false.

RET
Stops execution.

OK – Stops execution and pushes the value ”true” to the top of the stack
FAIL – Stops execution and pushes the value ”false” to the top of the stack
THROW – Throws exception in the NEO VM

Data

PUSH
Pushes new content into a box (taken from the stack), puts true/false into the stack

depending if the operation was successful

PULL
Pulls latest content from a box (taken from the stack), executes it and puts the result back into the stack.

INSPECT
Inspects latest content from a box (taken from the stack), and puts the token cost back into the stack.
DELETE
Deletes the content on top of a box (taken from the stack), puts true/false into the stack depending if the operation was successful

REPLACE
Replaces the content on top of a box (taken from the stack), puts true/false into the stack depending if the operation was successful

CALL – Executes content (taken from the stack) from a box (taken from stack) and pushes the result into the stack.Inputs to the called box must be passed in the stack.The box specified as input must be older than the ”this” box. This prevents infinite recursion.
COUNT – Gets amount of content in box. Pushes result into the stack.
NAME
Gets the name of specified box. Pushes result into stack.

FIND
Finds the address of the box with name taken from the top of the stack, pushes the result into the stack.

Payments

BALANCE
Pushes the number of balance of tokens of a box into the top of the stack.

PAY – size, amount
Pays a certain amount taken from the top of the stack in tokens to the box.

TIME –
Pushes the amount of time since last payment into the stack. If a payment never happened, pushes Unix Epoch as value.

Storage

ALLOC – size(4),zeros(4), data
Allocates N blocks in Phantasma Storage with the indicated hashes.
Number of blocks = total size / blocksize. Number of hashes must match.
Each hash is 4 bytes of data. After the opcode executes, a new PDFS link is created, the opcode is replaced in memory and the execution of the VM stops returning the link.

This opcode can only be executed once, and this happens always in ”write” mode, since content always is executed through a push with ”write” mode first, before reads / pulls can be done.
After being executed it is immediately replaced by the LINK opcode. This allows the PDFS contentID to be retrieved later at any time, and makes it impossible to alloc more than once the same content.

LINK – ID(8)
Puts a link to Phantasma Storage into the stack.

ELNK – service, size, data
Marks a link hash to third party external storage in a specific service (eg: Bluzelle/Storj)

SIZE – contentID
Returns size in bytes of specified content, pushed to stack.

WRITE – Writes content from top of the stack to storage using key taken from stack

READ – Reads content from source taken from the top of the stack from storage using key taken from stack

Miscellaneous

SHA256
Hashes the byte contents of the top value in the stack with Sha256 and pushes the result into the stack
STATE – [state]
Puts the value of the specified VM state into the top of the stack
INCLUDE – box
Adds the box to the list of boxes that caller

EXCLUDE – box
Removes the box to the list of boxes that caller

B Phantasma Data File System Reference

B.1 Node Protocol Messages

Here is a proposal of the message protocol for the initial Phantasma relay implementation.

QUERY
Allows anyone to request data blocks.
Request => (contentID, blockList)
Response => (contentID, blockbytes, signatures)

INFO
Allows anyone to request info and availability of a contentID.
Request => (contentID)

JOIN
Allows a node to join the network as a storage node.
The node must validate its own identity by signing a message with its NEO keys.
Once it joins, it receives a response containing a map of the network.
Request => (identity)
Response => (network)

ATTACH
Allows a node to attach itself to a master.
Request => (target)
Response => (ok/error)

DETACH
Allows a node to detach itself from a master.
Request => (target)
Response => (ok/error)

UPLOAD
Allows anybody to upload block from a contentID (the hash data must match what was stored in the blockchain)
Request => (contentID, blockID, data)
Response => (ok/error)

Bibliography

B. Krämer, N. Völker (1997). "Safety-Critical Real-Time Systems". In: *Book*. URL: https://www.amazon.com/Safety-Critical-Real-Time-Systems-Bernd-Kr%C3%A4mer/dp/0792380223.

Bowers, KD. (2009). "Proofs of Retrievability: Theory and Implementation". In: *Ari Juels*. URL: https://link.springer.com/chapter/10.1007/3-540-44676-1_30.

D. Shaw H. Finney, R. Thayer (2007). "OpenPGP Message Format". In: *RFC 4880*. URL: https://tools.ietf.org/html/rfc4880.

J. Douceur, R. Wattenhofer (2011). "Modeling Replica Placement in a Distributed File System: Narrowing the Gap Between Analysis and Simulation". In: *European Symposium on Algorithms*. URL: https://link.springer.com/chapter/10.1007/3-540-44676-1_30.

Naor, A. Fiat; M. (1994). "Broadcast encryption". In: *Semantic Scholar*. URL: https://pdfs.semanticscholar.org/d4bd/74e4b3007724ee0b78532d232bbbd4c3c2ef.pdf.

P. Maymounkov, D. Mazières (2002). "Kademlia: A Peer-to-peer information system based on the XOR Metric". In: *PDOS-MIT*. URL: https://pdos.csail.mit.edu/~petar/papers/maymounkov-kademlia-lncs.pdf.

S. Baset, H. Schulzrinne (2017). "An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol". In: *IEEE*. URL: http://ieeexplore.ieee.org/document/4146965/.

S. Wilkinson, V. Buterin (2017). "Storj, A Peer-to-Peer Cloud Storage Network". In: *Storj*. URL: https://storj.io/storj.pdf.

Zhang, E. (2016). "Neo Opcode reference list". In: *Neo Smart Economy*. URL: https://github.com/neo-project/neo-vm/blob/master/src/neo-vm/OpCode.cs.