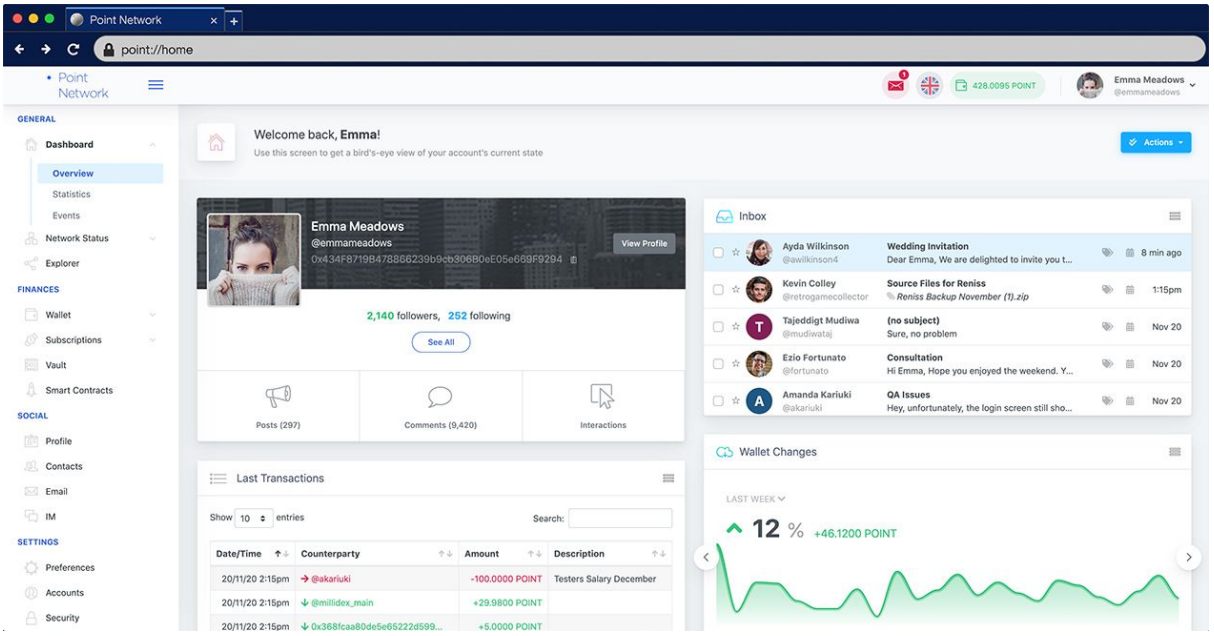


Point Network

Blockchain-Powered Decentralized Internet

pointnetwork.io

Table of Contents

Introduction

Point Network aspires to do for cryptocurrency space what Apple did for personal computing devices.

Tor, Bittorrent, Bitcoin, Ethereum - all these projects were stepping stones on the way to the Holy Grail—***decentralized internet***, also known as ***web 3.0***. Open for everyone, censorship-resistant, permissionless, privacy-oriented internet with programmable money—cryptocurrency—as its out-of-the-box payment system.

Point Network is a proposed implementation of this vision.

Our goals are: to build protocols and propose standards to serve as the foundation of the future internet, provide consistent and straight-forward User Interface to ease adoption by a large number of people, and finally construct a harmonious ecosystem for currently quite diverse decentralized applications (dApps).

What app do you have most often on your screen? You might have answered ‘Gmail’, or ‘Facebook’, or ‘Netflix’, but chances are, the application that stays most of the time on your screen is your browser, the window to the internet. Most cryptocurrencies do not stand a chance competing with it for attention, since their UI is usually a wallet with extra features, and they are used once and then forgotten until the next time you need them. Point Network comes with its own ***web3.0 browser***, and all interactions with the network happen inside it.

We invite you to read on to explore our solutions to the major pain points of both cryptocurrencies and traditional internet. After we discuss the storage layer, you will see how this enables web 3.0 re-invention of the most popular internet apps, how it allows for the creation of the apps impossible within the current paradigm, and what different benefits can this change bring. Hopefully, you will join us with your own ideas and resources in this exciting technological adventure.

The Internet Is Broken

Your Gmail, Google Drive, Google Photos, Hotmail/Outlook, Yahoo Mail, iCloud, Dropbox, your browser history, Skype calls, searches, 24/7 location history etc. - everything can be accessed by multiple parties:

- a number of their engineers, some of which could decide to target private sexual photos of thousands of young women including their friends and colleagues, or spy on Google accounts of teenagers,
- the data center employees and manufacturers - another attack vector,
- administrators of transit points for email, since there's no encryption in the protocol,
- your boss/IT person from your company in the case of G Suite,
- the hackers (when they manage to successfully compromise the servers),
- numerous government employees (when their law enforcement representatives obtain a subpoena),
- and intrusive government agencies such as the NSA, which just archive all private data of all users all of the time without even bothering going through the proper legal channels.

Won't defending from this help the criminals? The fact is, criminals are already on the front line of the technology: they know how to use Tor, encryption, WiFi wardriving, stolen KYC identities, etc. It is mostly public that is being harmed by the mass surveillance and security vulnerabilities left unpatched.

If you think you have nothing to fear because all your communications are in safe hands - think again: a 29 year-old NSA employee was able to physically take out tens of thousands of highly classified top secret documents without raising suspicion, CIA lost hundreds of documents and software tools in the 2017 Vault 7 leak, as did the NSA a year earlier.

Are you backing up your email? How many years of personal history and important information would you lose if Google decided to delete your whole Google account due to a copyright issue with a linked Youtube profile?

What about secure end-to-end messengers? ***Whatsapp*** hardly had recovered from the attack allowing to plant malware onto devices with one call, as US and UK are ***already*** pressuring them to weaken security and provide full access to the chats. Telegram so far holds its reputation as one of the most secure and resilient to overreaching requests of the governments, yet with a SIM swap attack it's trivial to get access to your normal chats if you do not have your password set as your second authentication factor (most users don't). Secret chats are the exception, although there are limitations, such as there are no group secret chats, and so on.

Do you use something like ***Protonmail***, a secure email service? It's innovative since it does not send passwords to the backend, the JavaScript program decrypts your inbox right in your browser. Only, where does this JavaScript file come from? Your

browser downloads it each time from the ProtonMail servers. If a hacker, a ProtonMail employee, or an intrusive government agency decides to modify this JavaScript file to make it in addition send your password in plain-text to another location so that they can decrypt and read your messages later, not only can you do nothing against it, you won't even be able to notice.

You might think you're safe with ***your own private email server*** at your home. Well, it's still not good enough: it's not end-to-end encrypted, there's no encryption in transit, and no encryption on the other side.

Cryptocurrency does have a potential to change all that, but it hasn't yet. We're not winning this battle so far. Cancerous practice of demanding KYC documents on every single exchange and crypto platform means Equifax-level disasters waiting to happen. dApps, Metamask, web wallets relying on centralized Infura nodes. Decentralized exchanges and web wallets boasting non-custodial level of security, yet their users are being hacked after the websites are replaced due to BGP hijacking. The same kind of vulnerability in the basic fabric of the internet that Russian attackers seemed to have used to re-route outside Visa and Mastercard traffic through their servers. And attacks on centralized SSL (HTTPS) infrastructure can allow to generate valid and trusted SSL certificates for any website.

Deplatforming and account suspensions, email addresses being easily spoofable, as easy as faking the caller ID on your phone screen, Bit.ly unilaterally deciding to block hundreds of links in Andreas Antonopoulos' books, Adobe deactivating all Venezuelan accounts overnight, Uber "god mode" spying... need we go on?

The internet is broken.

How do we fix it?

Let's tackle our first challenge.

Point Storage

Problem: The new internet needs open, permissionless, censorship-resistant, scalable storage layer. Blockchains meet all the requirements except the last one. Storing arbitrary data on a blockchain is:

- **Expensive:** at the moment, to store just 1 MB of data on Ethereum blockchain would cost you about $600[1].
- **Non-scalable:** All full nodes have to store a copy of all data, whether they need it (right now or at all) or not. Even with mostly financial data, the blockchains already grew to enormous sizes: full Bitcoin blockchain is 200+ GB, and Ethereum blockchain is 1+ TB.[2]

All of this makes decentralized applications such as social networks on blockchain completely unfeasible and impractical, especially considering the perspective of further global adoption.

Some Proposed Solutions

IPFS is a proposal to only store an identifier of a file (its hash) on a blockchain, while the file itself could be uploaded to the nodes on the IPFS network, much like BitTorrent files can be downloaded just by the torrent's hash inside a magnet link. However, nodes on IPFS have no incentives for storing and delivering files other than altruistic, or if node owners themselves are interested in the content. Therefore, if nobody "pins" your content, as soon as you go offline, all your posts and uploads and data would become unavailable to the network.

Point Network Solution:

Much like IPFS, Point Storage addresses the content by its hash. This small 20-byte identifier can be posted to the blockchain instead of the full content.

But unlike IPFS, Point Storage creates a decentralized market for storage providers. This means, you can upload your files to storage providers in different parts of the world, in exchange for a micropayment via a *lightning network* and make this file available for download. Providers commit to storing your file for the promised time period, facing penalties to their collateral if they manage to lose it.

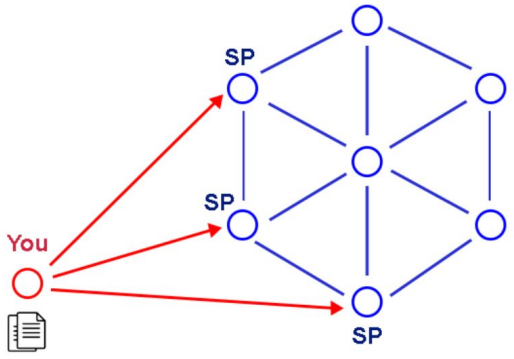
Now anyone on the network can query the network for your file using its identifier. Storage providers will send the file to them, earning a micropayment for used bandwidth, which will be pushed through the Lightning Network. And you can post this identifier on the blockchain for global discovery.

---

[1] See https://medium.com/coinmonks/storing-on-ethereum-analyzing-the-costs-922d41d6b316

[2] As a consequence, there are very few full archival Ethereum nodes left in the world. This is also one of the reasons many users resort to relying on third-party data providers instead of running their own full node, and even dApp developers choose to rely on centralized third-party providers, such as Infura.

Example (simplified):

You want to share a tweet with a picture on a decentralized Twitter. You enter the data in a browser and drag the picture into the upload area, then click “Tweet”. Under the hood, Point Network calculates the hashes of both the post and the picture, uniquely identifying them by their content.	 → a5425df3d3c1967... PER MY LAST EMAIL... → 46556b0ea9d8b39...
Then it sends a transaction to the Smart Contract responsible for decentralized Twitter, storing only these hashes on the blockchain, but not the full content behind them.	`contract DecentralizedTwitter { function tweet(address post, address picture) { // store... } }` Status: Success Block: 8151139 1 Block Confirmation Timestamp: 13 secs ago (Jul-14-2019 07:50:01
While the transaction is being confirmed, storage layer starts the upload. It contacts several storage providers (SPs) and sends them the image file and the text file.	

Storage providers, after downloading the files, give you a cryptographically signed guarantee that they promise to store this file, in exchange for a micropayment.
 	
If you realize the file is lost, you can challenge the SP on the blockchain, providing this signed guarantee, and if it's truly lost, the SP is punished by having its collateral automatically slashed.
 	
For redundancy, the files are stored with multiple providers, so that if one malfunctions or goes offline, others are still available.
	`I guarantee to store files a5425df3d3c1967... and 46556b0ea9d8b39...`
	
	`Guaranteed by my collateral on the blockchain`
	
	Storage Provider 0x952fa74b2...
	
	SP → You
	
	`Micropayment: 0.00000015 POINT`
	You → SP

Another user ("Reader") is reading the feed for decentralized Twitter from the blockchain. All the browser can see on the blockchain are hashes.	`tweets[]` • 0xa5425df3d3c1967..., 0x46556b0ea9d8b39… • 0xf72acd8bds9329a…, 0x000000000000000... • ...
But this is enough for the browser to immediately locate these files on the network. It queries the network for them to connect to the SPs storing them, selects one (the one with the cheapest bandwidth price, or the closest one, or the one with the highest speed etc.) and downloads the files by their hashes, pushing a micropayment to the SP.	SP -> Reader `Micropayment: 0.00000002 POINT` Reader → SP
And now the browser can display the actual tweets that were behind the hashes on the blockchain. All of this magic is quick and abstracted from the user. The user experiences it almost in the same way as on the traditional internet: as simple as posting a tweet, and clicking the link to view it.	

For more detailed explanations of the inner workings, please refer to the accompanying technical whitepapers.

Possible Applications:

- **Decentralized Dropbox/Google Drive.** You now have your own decentralized cloud. Only more secure, private and robust. Right now Google and Dropbox can see all your data and files (and modify them without you noticing), and so can the hackers if they manage to successfully attack the cloud. In Point Network, you choose what files and folders to be available to the network, and which of them would be encrypted with a derivative of your private key, and decrypted on download, making them accessible only to you.

- **Storing data for decentralized applications.** In the next chapter we discuss different kinds of decentralized applications that can benefit from this storage layer, but the main principle is the one you've seen in the example of decentralized Twitter above: large pieces of data are stored off-chain, yet their small identifiers are etched onto the blockchain, in the smart contracts of specific dApps.

- **Public Archives and Storage Auctions.** Some files might be deemed worthy of preserving not only by one person, but by a large group of people. In that case, they can "chip in" for the archival of certain files or groups of files to a special archival auction smart contract. SPs are participating in the auction, bidding on taking the privilege to store the files and committing to it on-chain. What could be the result? You might see something like "Users of Point Network have collectively chipped in to store the whole Wikipedia archive on 70 storage providers for 200 years."

- **Finally, solving blockchain storage scalability problems.** Not only can people suggest specific hashes and collections of hashes for the archival auctions, but they can do it for all kinds of content falling into a certain category and limited by certain rules. For example, people can similarly chip in to archive and update in real time the Bitcoin blockchain, the Ethereum blockchain, and so on. Now everyone can run a full node cheaply. You can treat the storage network as your virtual hard drive, and instead of everyone in the world having to store all data, they fall back to the knowledge that they always can, at any moment, pull any block and any transaction from one of the hundreds of providers that had committed to store them for money. Blockchain size doesn't matter as much anymore, however many terabytes it might grow into now, it's always available. Thus, it solves the blockchain scalability issue.

Now, if we stopped right here, we would be another decentralized storage project, competing with Filecoin, Swarm, Storj, Sia and other strong contenders. However, the storage part is only the first step that gets us to the bigger picture.

User Identities and Contacts

Identity is an important part of Point Network. In Bitcoin and Ethereum, identities are represented by gibberish-looking addresses derived from the private keys. For Bitcoin it might look like `1NaQLbizyqg4ySMXNsodwUdsvrgJS7xDXg`, for Ethereum - `0xb20c2cf83e98fe0c407ca9ffb9a19c03217ead7f`.

Although Point Network makes use of the public keys, private keys and addresses, we set our goal to make the interface as user-friendly as possible. The identities are represented by handles like **@emmameadows** - a string of characters that is supposed to easily refer to a certain user or an organization.

Identity handle is everything in Point Network. It is, all at the same time:

- what you can give to your friends to find you on Point Network and add to their Contacts;
- your wallet address to which people can send you payments;
- your email address;
- your instant messaging handle;
- your "phone" number for audio and video calls;
- your website (if your handle is **@emmameadows**, you have **https://emmameadows.z** as your website space, plus all the subdomains on it).

Please note: tying user activities to one unique user identifier in any way, which is what traditional internet does, would completely defeat the purpose of our whole project. Therefore, Point Network must support and make it easy for the end user to create and switch between different identities, uncorrelated with each other. On the one hand, we do not wish to recreate the traditional internet model allowing multiple parties to map and track users' activities, while on the other hand, we still want to provide users with the ability to develop their public personas and have meaningful ways to make connections with each other. Therefore, instead of making it completely anonymous for each action, we introduce the notion of identities. They do ***not*** mean any kind of KYC entries in the 'proof of identity' systems, but rather digital identities that can be generated in a permissionless manner.

Identity functionality is provided by a special smart contract curating a registry for all identities. Users can register handles and match them with the corresponding public keys they own. This part is similar to other comparable systems, such as Namecoin, Ethereum Name System (ENS), and others.

Identities can be controlled not only by one EOA (Externally Owned Account) key pair, but also by a threshold multi-signature setup, in which actions have to be

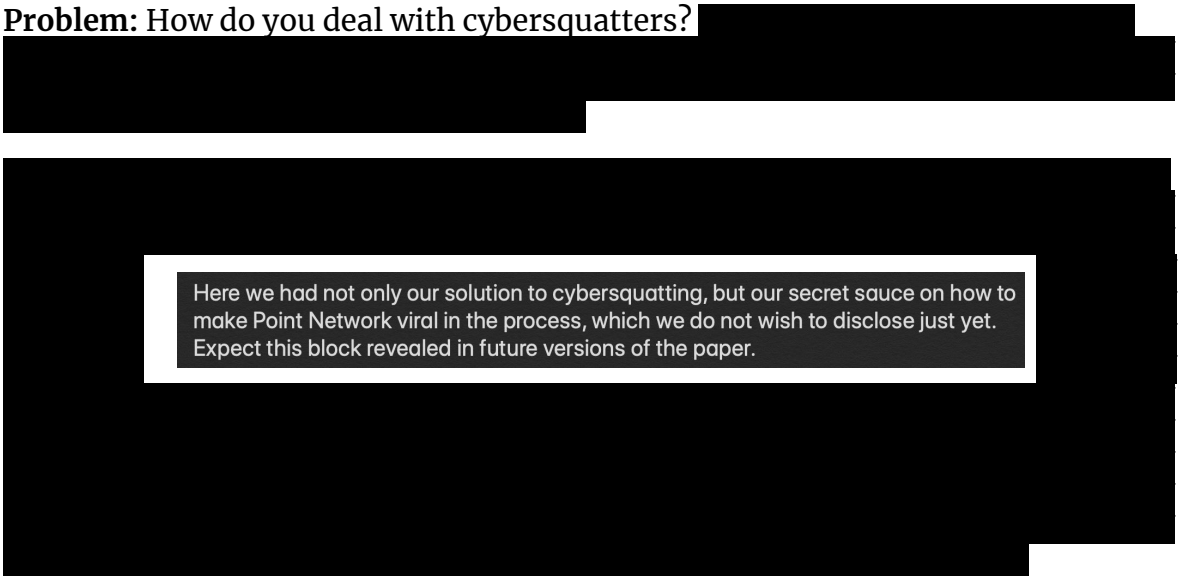
approved by a quorum of EOAs/identities. The contract allows for extensive control over the quorum variables, different quorums for different kinds of interactions etc. This kind of setup would be ideal for corporate, non-profit and similar identities.

The concept of **Contacts** (also known as the contact list, friend list, connections etc.) is familiar to all users of the modern communication devices. It has the same significance in Point Network, only instead of a phone number/email address/etc., the contact's identity handle is what's being stored.

Identities are attached to the public-private key pair authorizing their use, but are not their digital equivalents. An identity ownership can be transferred to another user (and another key pair) without revealing the private keys. Starting from that point, all users interacting with the identity will be able to notice that the keys for it have been changed.

◆ ◆ ◆

Problem: How do you deal with cybersquatters?



Decentralized Email

Problem: Despite email being one of the most popular communication tools, there is currently no secure and private email provider. Our emails often contain sensitive, personal and confidential information, with various photos and documents attached. In "The Internet Is Broken" chapter we have mentioned how the emails can easily be accessed by a large set of actors, ranging from hackers to government agencies. We also mentioned why email providers positioning themselves as secure, such as Protonmail, and self-hosted email servers, do not solve the issue.

Solution: The true end-to-end encryption is only viable if it's baked right into the application. Then its functionality cannot be arbitrarily changed by anyone else. Here is how it works.

- **@alice** wants to send **@bob** an email on Point Network. For her, the experience is just the same as using the normal email interface, but here's what happens under the hood:
- the software looks up **@bob**'s public key in the "Identities" smart contract;
- using well-known encryption algorithms, it encrypts the message with **@bob**'s public key and persists it on the decentralized storage layer as a file;
- it pushes a small transaction on the blockchain, calling a special "Mail" smart contract, leaving the hash of the encrypted and persisted message;
- the next time **@bob**'s software scans the blockchain, looking out for only the relevant information, it finds a new email notification addressed to **@bob**, from **@alice** with a hash of the message attached;
- it queries the storage layer for this hash, downloads the encrypted message from a storage provider, decrypts the contents with **@bob**'s private key and shows the email to **@bob.**

As a result, we can finally have the email system with the highest possible level of security and privacy.

◆ ◆ ◆

Problem: What if we need an email message to be sent to several people in the conversation at once?

Solution: We could encrypt copies of the message with each of their public keys, or, better, use a ***hybrid*** encryption scheme, prescribed for such situations, where the message itself is encrypted only once, with a randomly generated symmetric key,

and then this shared key, which is much smaller than the content, is encrypted with each of the recipients' public keys.

◆ ◆ ◆

Problem: In traditional internet, it is hard to establish the authenticity and integrity of the message. The email address can easily be spoofed in the email headers, and the contents can be modified in-transit. Modern tools employ different, ranging from heuristics to signing messages with protocols such as DKIM, however they fail to provide the highest possible level of security and force to rely on centralized digital authorities.

Solution: In Point Network, when a sender pushes an email notification to the Smart Contract "Mail", the sender of the transaction is recorded in the notification as part of it. Thus, we have a guarantee that the message originated from the sender and not someone else. Furthermore, addressing the message by its hash ensures data integrity, providing a guarantee that it has not been tampered with at any point in transmission.

◆ ◆ ◆

Problem: Email spam.

Solution: In addition to the traditional anti-spam mechanisms, users can adjust the settings in the "Mail" Smart Contract to require a certain small amount of money to be attached when an email is sent to them. Collectively raising this amount up to a reasonable level will make spamming mostly unprofitable. (This is a modified version of an idea that predates Bitcoin, called HashCash[3], and which gave rise to the mining algorithm of Bitcoin.)

◆ ◆ ◆

Problem: Some people are more popular than others and celebrities can receive more messages than they are able to go through.

Solution: Such users may decide to increase the amount required to be able to send messages to them to more-than-average. In this case, only people that are serious about communication and deem it worthy will do it. Of course, it can be set up in such a way that people who are added to that person's Contacts (such as friends and colleagues) are spared the need to pay for the messages.

[3] http://www.hashcash.org/hashcash.pdf

Decentralized Text, Audio and Video Chats

Problem: This problem is similar to the one we've touched upon in regards to email. For example, Whatsapp boasts end-to-end encryption, but it was caught having critical backdoors allowing attackers to take over your device and have unlimited administrative access to the data and the phone's functions. Whatsapp and some other applications also nag users to turn on the setting that would allow them to store backups of all messages to the cloud, from which point on, it's all there for the taking, not end-to-end encrypted anymore. Telegram doesn't do this, but is still far from the ideal. The end-to-end encryption is only enabled for secret chats, and secret chats are only available for one-on-one conversations. All your normal chats and group chats are still being stored in plain text on Telegram servers. Reputation helps users trust Telegram not to abuse this power, but why trust third parties in the first place? This is not the crypto way, we can do better.

Solution. By applying the same principles as for the email, with small modifications, we can now have the instant messenger with the highest level of privacy and security.

Encryption is done the same way as in the email example, only instead of uploading the message to the storage layer and then notifying the recipient, we propose sharing the encrypted messages through ***pubsub*** functionality of the network to reduce the lags. Nodes can either charge the same price for spending their bandwidth for message routing (through micropayments), or, if the participants can connect directly to each other, they may use the direct connection.

After the messages have been quickly delivered, each of the participants' software can now buffer the messages and persist them in bulk on the storage layer to have the message history always available to them. On the same principles, users can have group chats and secure audio and video calls.

◆ ◆ ◆

Problem: Not all people want their messages attached to their identity in case they're publicly leaked by the second party or due to the second party being compromised. Are the messages on Point Network always attributable to the identity used to sign the messages?

Solution: Not necessarily, we can use a shared symmetric session key instead. This way, each party can trust the message authentication, yet any third party would have no proof that the leaked message was not forged by the opposite side, since they both have had the session key sufficient to sign it, or faked by anyone else, since the session key has to be leaked as well. And in the case where digital signatures are actually needed, that person can ask the other side to specifically do so, and they, by using the "Sign" feature, can attach a cryptographic signature to the message, signing it with the identity key. Now it can be publicly attributable to that identity, verifiable by anyone.

Building the Internet of the Future

ZDNS

Decentralized name servers idea is one of the oldest and most straightforward uses of cryptocurrency after digital money. One of the oldest cryptocurrencies in existence is Namecoin, providing users with an ability to register domain names on the blockchain and resolve them accordingly. One of the earliest presentations of Ethereum included a demonstration of how one can implement Namecoin in a few lines of Solidity code. Today we have dApps such as Ethereum Name Service (ENS), EOS Name Service etc., that have extended this basic idea with additional functionality.

Decentralized DNS is an important part in fixing the internet security. It takes care of several attack vectors on basic internet infrastructure: name servers providing domain name resolution, compromised internet access points (internet providers, WiFi APs) trying to connect you to another server instead, compromised centralized SSL certificate infrastructure, plus it also provides opportunities for end-to-end encryption.

That's not all that ZDNS is concerned about, however. In the cybersquatting section of Identities chapter we've described the protocol to register an identity, which provides a user with a human-readable handle. Part of owning the handle includes the command over the domain space in the virtual '.z' zone, including subdomains. For example, for *@mike* the domain space would be *mike.z*. In a special ZDNS contract, functioning like a decentralized DNS registry, the owner can then set up different DNS records, following the standard protocol for name registries. For example, users can set up A records pointing to the IP addresses, which is what other DDNS implementations do as well. Point Network provides such functionality for other services, but ignores these records by itself. For Point Network, the most important record type is *Z record*, which points to a hash, which points to a storage location of a *routes.json* file. After downloading it, the file itself may look something like this:

```
{
        "/":                    "0x183ddc6138290dc10c9eef7ffd39581630d1f10d",
        "/contacts":            "0x709342b1b0add23356400f4153cf3448b279adcd",
        "/about-us":            "0x313369d7ed1d287d961cef4a7552a01be6723811",
        "/articles/{id}":       "0x412b3e52a20545115066f097c206ee9136aa00ee",
}
```

Each of these hashes represent a ZHTML template. It will become clear how this file is used when we cover ZHTML section.

Point Browser and ZProxy

Point Network comes with its own ***web3.0 browser***, which functions similarly to the browsers you're used to, plus additional extended functionality designed specifically to work with Point Network protocol.

Point Network nodes contain a module called **ZProxy**, which is locally listening on a designated port. Point Browser contains instructions for connecting to the decentralized internet through this proxy. When you open *example.z* in Point Browser, it first naturally tries to resolve *example.z* to an IP address, which it also does through ZProxy. ZProxy returns `127.0.0.1` for all existing domains ending in *.z*, gives no results for non-registered domains, and initiates a command to open for a default OS browser for all other domains, such as ending in *.com*.

After it's been resolved, the browser then asks the server for the URI entered into the address bar. Again, this query is intercepted and executed through ZProxy. ZProxy looks up the *routes.json* file hash, fetches it from the storage layer, matches the URI to one of the routes (or gives 404 response if not found), downloads the ZHTML template by its hash, executes it and responds to the browser with the rendered result.

Isolation Layer

Each cryptocurrency project is unique due to decisions the creators had to make. Invisible forces of 'survival of the fittest' kind then reveal which combinations of decisions turned out to be the best in the end. Point Network developers have to make lots of choices if the project is to have a chance at standing out and succeeding, not being a slightly modified copy of something else.

One of such choices is the level of convergence of the new internet and the old internet. Much like VMs and blockchains, we decided (for various reasons, including security) it's best to isolate the new space completely and grow it from zero.

The 'old web stuff' will not be directly accessible from the browser initially. As mentioned, the links pointing to the standard http and https content will prompt Point Browser to open a confirmation dialog, after which it will ask your OS to open the default browser and pass the URL there.

This doesn't mean you won't be able to enjoy the same content, because much like with the blockchains, we expect users to archive and mirror the content they need on the decentralized file system of Point Network. If you need a '*bootstrap-4.3.1.css*' file available for your website, you can upload it with a simple command or drag-and-drop in one of the internal browser tabs, and now it's accessible. And

chances are, if you're looking for such popular files as *'bootstrap-4.3.1.css'* [4], they've already been uploaded by someone else to a large number of providers, since as mentioned, the more popular the file gets, the more accessible it becomes.[5]

Another choice we had to make is to disable JavaScript in the first versions until we come up with a robust security model and test it. So, if we have no backend and no frontend languages available, how can we ever create dynamic applications that would compete with feature-rich 'old-web' apps?

ZHTML

This part is what differentiates Point Network web space from that of IPFS/Filecoin/Swarm etc. While they can only serve static content (and simulate backend capabilities using insecure JavaScript code, allowing for the same freedom to collect analytics data, send encryption keys to the private communication in dApp masking them as something else in Metamask or directly to the 'old-web' server etc.), we propose extending the standard HTML protocol into what we call ZHTML to make the dApps backend-capable. Extending HTML this way is not new, perhaps you've seen it done in SSI, ASP, PHP and similar languages. The language itself resembles Django/Nunjacks/Twig templating engines.

Where do these backend scripts execute if there are no servers? Much like traditional blockchain scripts (Script in Bitcoin and EVM in Ethereum), executing on everyone's computers that run node software, ZHTML executes on everyone's computer that asks for this website and page, in an isolated virtual environment.

When ZProxy intercepts the HTTP request from the browser, it matches the URI with the route in the current *routes.json* file for this domain, it downloads the ZHTML template, executes it, and outputs the rendered HTML back to the browser.

Here's what it might look like:

```
{% extends '794c3e6bf726f86550b26f531e902f19217476ed.zhtml' %}

{% block contents %}
   <h2>Welcome!</h2>
{% endblock %}
```

This already starts to showcase the dynamic abilities of ZHTML. Since some website code is repeated throughout all pages, it made sense for the programmer to take it out into a *layout.zhtml* file, which Deployer (see below) turned into a storage hash pointer. Here's the *layouts.zhtml* file:

[4] Note: what matters is not the file name, but the file contents, because the file is addressable by its hash. If the file is renamed but has the same contents, it is considered equivalent by the network.

[5] This is due to the fact that the nodes in between decide to cache the popular file and serve it themselves, taking the whole micropayment, rather than proxy it to the original hoster of the file. This mechanism, in turn, forms a self-organizing decentralized CDN, where popular files automatically become more available.

```
<!doctype html>
<html lang="en">
<head>
   <title>Example Blog</title>
   <link type="text/css" rel="stylesheet"
         href="275e924dba9a04abbfd6718ae6790209a1b34c27.css" />
</head>
<body>
   <h1><a href="/">Example Blog</a></h1>

   {% block contents %}
   {% endblock %}
</body>
</html>
```

Block *contents* will contain `<h2>Welcome!</h2>` after ZHTML compilation.

Rendering Dynamic Content

Let's take a look at our *routes.json* file again:

```
{
        "/":                        "0x183ddc6138290dc10c9eef7ffd39581630d1f10d",
        "/contacts":                "0x709342b1b0add23356400f4153cf3448b279adcd",
        "/about-us":                "0x313369d7ed1d287d961cef4a7552a01be6723811",
        "/articles/{id}":           "0x412b3e52a20545115066f097c206ee9136aa00ee",
}
```

As you can see, there are not only fixed URLs, but also wildcard constructs, such as `/articles/{id}`. This means that all URLs adhering to this pattern, e.g. `/articles/5`, will be caught by this route and the associated template. On top of that, `5` will become a value of a local `id` variable when rendering this template.

Here's how this template may look like:

```
{% extends '794c3e6bf726f86550b26f531e902f19217476ed.zhtml' %}

{% block contents %}
   {% set article = Contract.at(ThisWebsite.contract).getArticleById(id) %}

   {% if article is null %}
       404 Article Not Found
   {% else %}

       <h2>{{ article.title }}</h2>

       {% set article_text = Storage.get( article.contents ) %}

       <div class='article'>
           {{ article_text | nl2br }}
       </div>

   {% endif %}

{% endblock %}
```

You can see the perfect blend-in of the static HTML structure, ZHTML backend engine, interactions with the blockchain smart contracts, and interactions with the storage layer. Here's what's happening:

- `ThisWebsite.contract` contains an address for the smart contract governing this website. `Contract.at()` returns the ABI interface for this contract.
- `getArticleById()` is a function in this smart contract that will take the id of the article as its argument, and return the `Article` structure for this article.
- `article.title` is a title of the article, which is a string that is stored right inside the blockchain.
- `article.contents`, however, is a hash to the article body on the storage layer. `Storage.get()` queries the storage layer, and, if successful, loads the article contents into `article_text` variable, which then gets displayed.

As you can see, this goes way beyond static websites. The ZHTML is very versatile and flexible enough for all kinds of decentralized applications. Website owners can change the `article.contents` hash in the smart contract, and the website at this URL will start to display an updated version of the article. However, what about impactful user interactions right from the browser?

Interactions

Let's say we want to enable comments on the websites. It's trivial to implement the Solidity code for storing the comment data, so we will not include it for brevity. However, here's how we will need to extend the article template on the ZHTML side:

```
...

<h2>Comments:</h2>

{% set comments = Contract.at(ThisWebsite.contract).listComments() %}
{# returns an array of Comment structures #}

{% for comment in comments %}
    <div class="comment">
        <p>From: {{ comment.from }}</p>
        {% set comment_text = Storage.get(comment.contents) %}
        <div class="comment-body">
            {{ comment_text | nl2br }}
        </div>
    </div>
{% endfor %}

<form method="post">
    <input type="hidden" name="__method" value="addComment" />
    <textarea name="__storage[contents]" required></textarea>
    <input type="submit" value="Add Comment" />
</form>
```

The way the comments are being fetched and displayed are the same as with the articles. However, here's how adding the comment works:

- The form with the POST method is being sent to the server. ZProxy, as with all requests, intercepts this one.
- It fetches the default contract governing this website.
- It takes all the data in the form and sends it to this contract, to the method designated in the special `__method` field (in this case, `addComment()`).[6]
- If the textarea name was just '`contents`', ZProxy would try to send the whole textarea value to the function as `contents` argument (and would fail, due to type mismatch and length validation). However, `__storage[contents]` tells ZProxy to first upload the contents on the storage layer, and then put this hash instead as the `storage` argument to `addComment()` method.

Deployer, and Other Tools for Development

Of course, we want all these processes to be as transparent as possible not only for the users, but also for developers. Additional tools for interactions with Point Network have to be developed, such as Website Deployer, Solidity IDEs and tools such as Remix browser/Truffle suite, block/transaction/network/website explorers and so on.

Deployer in its first primitive version is already released and available for experiments (see the Demo section). Its job is to take the website and smart contracts, upload them onto blockchain and storage layer (or update if needed), replace file names with storage hashes etc.

[6] Note: all user interactions with the smart contracts are confirmed through the MetaMask-like dialogs in the Point Browser, giving the user a detailed description of what's being sent, access to the source code, and a test run showing what would change inside the smart contract, ceteris paribus, after this transaction is applied. UI/UX workgroup will have to work on how to allow users to whitelist certain actions/websites to make the dialogs less annoying.

Decentralized Applications

What we've covered already allows for a wide variety of decentralized applications, although many details remain to be figured out. Here's some apps we think should be possible and we would like to see implemented on top of Point Network:

- ***Decentralized social media***: Decentralized Facebook, Twitter, Reddit, Medium, forums etc. We've already described how it can be possible.
- ***Decentralized Youtube/Periscope/Twitch.*** For decentralized Youtube, for example, it's a matter of uploading a video to the Point Network, sharing its hash with the smart contract, and displaying it in the browser using a standard HTML5 video tag.
- ***Pay-per-minute video streams.*** Using *payment channel* technology through the *lightning network*, this would allow people to have paid video sessions with different specialists (consultants, medical experts, etc.) + group sessions (virtual classrooms, office hours, webinars etc.)
- ***Decentralized Patreon.*** Support your favorite content creators, with a verified blockchain tag (if you want) on your social media profile.
- ***Decentralized Github.*** Since the storage layer can be abstracted via FUSE and appear like just another folder, it should be trivial to make git work with Point Network storage. We'd also need a smart contract that would allow people to manage repositories, approve commits, add comments etc.
- ***Decentralized packet managers.*** See How one developer just broke Node, Babel and thousands of projects in 11 lines of JavaScript for why we might need this. (Notice how they've fixed the data without the user's consent, which revealed the inherently centralized nature of *npm* and left open the question of how it could be used for nefarious purposes one day).
- and so on - collectibles like *Cryptokitties*, ways to support Open Source software like *gitcoin*, games, shops, DeFi ("decentralized finance"), prediction markets - the list goes on and on of what might be possible.

Obviously, we don't expect to have this all implemented by ourselves, so the developers in the community will have ample chance to help.

The ultimate goal is to eventually do away with providers and traditional internet completely, forming a mesh network of interconnected devices. If your cell phone antenna has enough power to reach the cell phone tower dozens of miles away, it's enough to reach nearby devices, even if they're far away. Practically, this would mean, for example, when you travel to a new country, instead of buying a SIM or asking for someone's wifi, your device just connects to nearby devices, accesses the network through them, awarding them with micropayments, even as you keep traveling in a car to a new place.

Tools for Digital Organizations

One of the inspiring ideas that came with Ethereum was DAO - a Decentralized Autonomous Organization. Unfortunately, this term is now negatively associated with ***the DAO*** - a specific DAO implementation which turned out to be vulnerable, almost cost investors millions of dollars, produced much controversy and had to be closed down.

Nonetheless, the idea behind it was very innovative. Of course, even traditional businesses and companies can benefit greatly from Point Network - having access to decentralized versions of Google Docs, Calendar, Slack, Zoom etc. The identities could also be multisig-enabled, allowing the managers in the organization to approve certain actions, website uploads, moving the funds, appointing and hiring people on positions, and so on. But what about the idea of having a completely decentralized digital organizations, governed by smart contracts?

Lately, cryptocurrency projects spend a lot of effort to distance themselves from being classified as *security tokens*, due to existing laws and regulations in some countries. We also argue that POINT itself is a *utility token*, representing digital rights to receive services from storage providers.

However, we feel that the idea in itself, of digital organizations being groups of people organized to pursue a business, investment or even a decentralized software charging fees for transactions or selling in-game items, with dividends paid to digital shareholders and the code development being supported by their financing, is an exciting one, and worth discussing and experimenting upon, both from legal and technological standpoints.

Wallets, Currencies, Exchanges and Payments

This is one of the most boring parts, since every cryptocurrency has it: Send, Receive, Transaction history...

Nevertheless, we can still make it more interesting.

The most important tangent is, of course, working on privacy solutions. Bitcoin and Ethereum privacy levels are, unfortunately, far from desired.

Additionally, the crypto community already has a number of interesting ideas:

- **Vaults**, where a large amount of coins might require a second factor to unlock it, and/or the time window in which you can reverse attacker's transaction if you notice the wallet was compromised.
- **Credit card-like subscriptions** (pull payments), where your landlord, gym or a subscription service can charge your address monthly or weekly, without your interaction, until the subscription is cancelled by you or the address is out of funds: see EIP 1337;
- **Escrow services**, including trustless MAD escrow without a third-party interaction;
- **Cryptoasset inheritance planning**;
- **Decentralized exchanges** (including cross-chain);
- and so on.

We expect even donations to Open-Source projects to become more popular, when it's virtually one-click, users can see in real time how many other people are donating, and to obtain the blockchain-verifiable flair 'I support project X' on their social media profile (if they wish to publish it).

Current Progress

If you're reading this far, the idea, no doubt, seems exciting to you. However, be fully prepared that full execution of this can take years. As you might have noticed, this is a very ambitious project with lots and lots of moving parts that have to interact with each other. Right now we are in the research and prototyping phase.

At the moment this paper is published, we have our first prototype which you can check out here: https://pointnetwork.io/demo-1.

We want to avoid reinventing the wheel, wherever possible. That's why we have chosen to build on top of Ethereum code. We also use Node.JS as our primary implementation language/environment. We use Electron for cross-platform browser capabilities. We plan to use Raiden Network as the lightning network implementation. We want to experiment first on top of existing Ethereum mainnet and testnet, which is why POINT would be ERC20 token at first. Eventually however, we want Point Network to have its own blockchain history, and develop independently from Ethereum, which would help us to innovate fast without going through the motions with Ethereum Improvement Proposals.

That doesn't mean the projects would become competitors however. Our technology is and would always be Open Source, and will still be available for Ethereum users. We will also try to make the systems work together and not be agnostic of each other. And vice versa, Point Network would greatly benefit from Ethereum development moving forward. For example, we plan Point Network consensus algorithm to be Proof-of-Stake, and we plan to become one of the first early adopters of Ethereum 2.0 code, building on top of it.

We did not choose Proof-of-Work because this would have opened the project to security risks, since we cannot attract as much hashpower at the beginning as the cryptocurrencies with the largest pool of miners (Bitcoin and Ethereum), and as such a 51% attack on our consensus would cost very little, just diverting a few mining resources. We did not choose Delegated-Proof-of-Stake because we could observe all the shortcomings of this approach in other chains. Proof-of-Stake seems to be the most secure and decentralized approach.

Proof-of-Stake coins cannot be airdropped for free, since the security of the network hinges on stakers valuing their staked coins enough not to attempt attacks that would destroy the collateral. The currency has to be worth something, so the more likely scenario is that POINT will be sold as an ERC20, and later migrated to become its own currency with its own blockchain (similar to EOS and others).

How You Can Help

We encourage the community to hold off measuring contests for transactions per second, hashpower and so on, and remember why we've started all this in the first place. This is not just about Cryptokitties and having fun, although it's a pleasant side-effect. This is not about personal enrichment of anyone either, even if the rapid appreciation of cryptoassets has been nice to the "hodlers" and early adopters. This is primarily about enrichment of everyone, taking back our privacy, our digital and financial freedom and our rights. This is about innovation pushing forward through century-old laws and rigid government policies.

There are two ways to fix the current situation. The first one is to fight in courts and use the law to make programs stripping away our rights illegal; second is to try to use technology to counter it. Edward Snowden, the man who revealed the existence of such programs to us, had to admit that unfortunately, the law begins to fail us. Thus, if we fail on the technology route, and status quo prevails, and nothing changes, then Snowden revelations were for nothing. No pressure.

So if you share the ideals of the free internet, free speech, encryption, right to individual privacy, free markets, innovation, and you want your offspring to live in a freer society and not one of totalitarian surveillance from birth, you can help.

First of all, we invite you to subscribe to our updates on pointnetwork.io. This is how we can stay in touch with you and you can receive the latest news from us.

How fun would it be to have been working on there when early versions of HTTP, FTP, SMTP or SSL protocols were figured out? If you're a developer, come work with us. Email us your CV to info@pointnetwork.io. Even if you don't think you can be of much help, email us anyway, there's always something.

With the project of our size, we also need funding, and not shy to ask for it. If you're ready to support us this way or introduce us to someone who can, please email us at info@pointnetwork.io.

If you don't fall into any of these categories, you can still help a lot by spreading the word and giving feedback. Please share this paper in the social media. Comment on it on bitcointalk or share your feedback with us privately, if you want.

And if we ever go back on our principles or there's any suspicion of coercion, please fork our code to make it adhere to the values described here. This is not a matter of allegiance or loyalty or convenience. Of course, the community would become divided, and there could even appear out-of-nowhere some people shaming you for doing this and implore you to compromise. Do it anyway. This is how we win.

Similarly, if we stop developing the project, it's Open Source. Take over and continue from where we've left. Once the idea is out there, it is hard to kill.

Web 3.0 is here to stay. Welcome to the new beginning.