NEBULIS

-FIRST MEMORANDUM-

Philip Saunders

27th September, 2016

-Abstract-

This memo presents an overview of Nebulis, a decentralized, uncensorable internet which uses the Interplanetary File System (IPFS) [1] as a storage and transport layer, and the Ethereum blockchain for DNS utility.

At its core, Nebulis is a federated system of smart contracts. The entrance point or scheme is “ipfs://” (which we refer to colloquially as the “ox”). The root contract contains rules governing the creation of new clusters. Clusters are federated, autonomous contracts which govern top level domains. Cluster contracts each have their own indexed databases for domain records, which map the owner’s Ethereum address (a cryptographic public key secured by a private key under the user’s control) to a UTF-8 encoded string URL. This pointer is then further mapped to an IPFS hash to the resource, Dapp or website in question.

The system awards a metering token called Dust (DST) to the user following the creation of a record. We examine the dynamics of how Dust disincentivizes cybersquatting and releases dormant pointers back into the wild. Finally we describe the caching and fast resolution system, Gravity, which provides the missing link between Nebulis and the browser.

-Principles-

A working domain name system should satisfy three constraints:

1. **Ownability:** resource and title should be separate, and it should be possible to pass ownership of pointers.
2. **Swappability:** it should be possible to swap out the content to which the pointer points, i.e. it should not a fixed link.
3. **Finality:** it should be secure, auditable, uncensorable and tamper-proof, under the exclusive control of users.

-Alternatives-

The first alternative to Nebulis, DNS, is currently monopolized by the Internet Corporation of Assigned Names and Numbers (ICANN). The DNS satisfies the first two of our constraints but fails on the third.

On October 1st, 2016, President Obama relinquishes ICANN from the jurisdiction of the United States, leading to a possible takeover by the UN. This move could have fundamental consequences for the freedom and security of users.

Under the US constitution, users are protected by the First Amendment, which guarantees the right to free speech. Handing de facto control of the web to the United Nations would end this legal protection for all users [2]. It would run the risk of member-states attempting to place special conditions and restrictions over user’s speech, violating finality. The founding principle of Nebulis is the belief that the right to free speech belongs to all mankind; that this sacred principle should be encoded deep into our methods of communication.

The second alternative is IPFS’s homegrown namespace, the Interplanetary Name-Space (IPNS), which provides human-readable names for IPFS links. IPNS is purported to be a native namespace under the IPFS umbrella. But a cursory analysis finds that it meets the second and third of our constraints but not the first. It is swappable and final, but not ownable. This is problematic given the value and scarcity of good names. It ought to be possible for users to pass ownership of pointers from one to another.

The third alternative is Ethereum’s own homegrown storage and transport layer, Swarm. This is the approach of Nick Johnson’s proposed Ethereum Naming System (ENS) [3] which is proposed to use Swarm as a storage and transport layer. However, while useful for the purpose of providing native storage and transport for Ethereum Dapps, Swarm is almost indistinguishable from IPFS in terms of basic functionality while being significantly behind in development maturity. It is rescued from redundancy only through considerable support from the Ethereum Foundation.

A more institutional argument against Swarm is that like the first alternative, it presents the possibility of our third constraint of finality being violated. The Ethereum Foundation would be in a position rather like ICANN now, and given some of the controversies surrounding the Foundation’s intervention after the collapse of The DAO, this balance of powers may present a vulnerability to manipulation via opinionated forks. The optimum situation is to keep the storage and record layer in completely different baskets so that no such event would be possible now or in the distant future (even if this would seem unlikely at present).

There has always been a semantic difference between The Internet and "internets" in the indefinite article. Perhaps The Internet we have is as presumptuously named as "The DAO"- there's no reason why multiple internets cannot coexist in peace for different purposes.

-History-

The earliest version of the Internet emerged in 1969. ARPANET was a packet-switching network and the first ever TCP/IP protocol implementation, created by the United States Department of Defence. Some of the applications invented for ARPANET are still with us, including email and File Transfer Protocol, (FTP), which was used as to quickly transfer data from a client to a server. It is still possible to access FTP through most browsers via ftp:// and Filezilla.

The internet as we know it came into being with the contributions of Tim Berners-Lee in 1989. The major difference between the Internet before and after HTTP is that, instead of just being a way to send files, it was literally a "web" of hypertext that anyone could view in a browser, and with a much richer set of instructions that didn't depend on detailed knowledge of the respective file systems.

IP addresses are what link devices to the network, and are administered by IANA on a geographical basis, rather like the computer-equivalent to telephone numbers. The IP address namespace has run through several versions (we're now on version 4 and 6). From the beginning, the domain name system emerged in a relatively decentralized fashion, with multiple zones and nameservers administering sections independently. However it is still controlled by governments and large corporations, and its server-dependent architecture creates many single points of failure. This has led to problems with censorship and unjustified restrictions, most notably in countries like China, Russia and North Korea, not to mention concentration of user data in the hands of companies like Google and Facebook.

IPFS came about in 2014 as the brainchild of Juan Benet. The major transformation that IPFS offers is that content is no longer linked by the IP address of the server which hosts it, but by a hash of the content itself. Instead of being geographically addressed, it is content addressed. Files added to IPFS are then kept in a distributed hash table, similar to BitTorrent.

Like torrents, more popular content is kept in multiple places, and delivered by the closest neighbour to the requesting party. This feature makes IPFS uncensorable; you can't shut it down because there is no server to shut down. Other benefits compared to HTTP is vastly reduced bandwidth and greater speed the more popular a piece of content gets. IPFS also incorporates many features of Git version control and the Unix file system, making it much easier and more intuitive for web developers to build on.

-Ethereum-

Bitcoin in 2009 was the first implementation of blockchain technology [4]. A blockchain is an append only database which keeps an immutable record of every transaction in its history. Blockchains are supported by a network of interconnected nodes, all of which hold the same copy of the record, making the network secure against attempts to defraud it. For currencies, this feature prevents double spending; while also being very useful in applications like registries, land titles, timestamps, and many use cases yet to be realized.

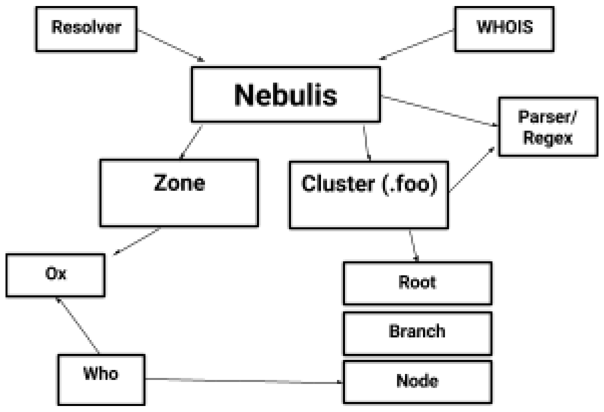
This invention solved one of the major problems throughout human history: how to resolve conflicts over scarce resources without the need for a trusted, centralized legal and financial institutions to oversee them.

Vitalik Buterin proposed Ethereum [5] as an alternate blockchain which had its own general purpose Turing-complete programming language and virtual machine. Instead of having to build a new blockchain for different applications (such as Namecoin) you could have the security and network effects of a single blockchain and the ability to create any kind of decentralized application on it using smart contracts. If Bitcoin is a decentralized monetary system, Ethereum could be described as a decentralized crypto-legal system.

This ability to encode conditionals and logic makes Ethereum much more advantageous and easier to work with than attempting to hammer a DNS into a lower performance blockchain like Bitcoin, which has been the approach of projects like Blockstack.

-Nebulis-

The system is divided into a few elements :



These are the core contracts of the system:

- **Nebulis:** controls the creation of new clusters and zones, redirecting queries as well as basic governance functions.
- **Whois:** Query the existence and availability of domains and clusters.
- **Resolver:** Takes URL queries and returns the IPFS resource if it exists.
- **Who:** "Accounts"- whenever a new user creates a domain they are assigned a Who contract which holds all domains associated with that address as well as keeping a direct address link to the database nodes where the domain records are stored.
- **Zones:** Groups of users run managed as one group. Useful for domain name registrars running on Nebulis. Zones are Who contract-factories. Naming convention for zones is NB1, NB2, NB3 etc.
- **Clusters:** Top-level-domains. Anyone can create a new cluster as long as they amass a certain amount of Dust. Once a cluster is created, anyone can register a domain on it.
- **Root/Branch/Node:** Every cluster has its own database. Entries are indexed alphabetically (or in their UTF-8 assigned order) to three levels of depth.
- **Parser/Regex:** By default only certain valid character sets are activated, although when a cluster is created it is possible to specify a greater custom character set.
- **Ox:** When dormant domains are ejected, the Dust tokens associated with them are send to the Ox contract and paid to miners via

.

These are the core elements of the prototype, although it is expandable and upgradeable via routers and interfaces.

-Dust-

In order to run or interact with contracts on the Ethereum network it is necessary to pay gas, which acts as a kind of crypto-fuel which is used up by computation. Gas is a workaround to the Halting Problem, which is the problem of determining from the given input whether a program will finish running or not. On Ethereum, a certain amount of gas is supplied with each contract creation or interaction. Once this is used up, the contract execution stops or reverts, protecting the network against malicious or accidental infinite loops and denial of service attacks.

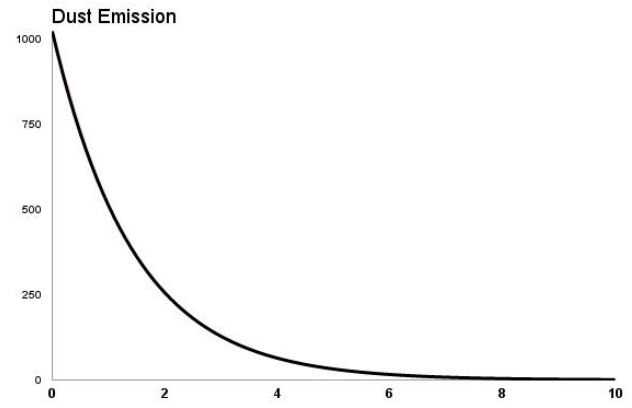
Nebulis has a similar utility token: Dust (DST). Whereas Gas is comparable to fuel, Dust is more like "rent".

Dust is coined into existence when domains are created, and is priced at the gasprice of creating a domain. For each domain, 1 unit of Dust is created and awarded to the user. Note that it is not necessary to use Dust to buy the domain in the first place, all that is required is to pay the gasprice at time of creation. This means that acquiring Dust from more dense clusters is more expensive, because as the database expands gasprice of interaction will be more. But dust everywhere has the same amount of governance leverage no matter what cluster it was generated in. This provides an incentive for people to register domains in less dense clusters.

-Deterioration-

1 unit of Dust is equal to 1024 microns, which is the smallest denomination.

The reason for the irregular denomination is that for every Nebulis epoch- which is 42 days in Unix time- if a particular domain is dormant, then the **check()** function is triggered on a schedule which causes the Dust associated with the domain to halve from 1024 microns to 512 and so on over 10 epochs until it gets down to 1. If there are still no resolver queries by the next epoch and remaining microns is less than 1 the **eject()** function is triggered; the record is deleted in the node and in the user's contract, and the Dust that associated with it is sent to the Ox contract where it is paid to miners. In this way Dust is recycled back into the system.



The rate at which Dust decays follows the pattern log2(n), which results in 10 halvings of 1024 before getting down to 1 micron: 1024, 512, 256, 128, 64, 32, 16, 8, 4, 2, 1.

The purpose of this is so that no domain can be left permanently unattended, while allowing enough leeway (420 days) so that it doesn't need constant attention.

When Dust is emitted from the user's variable balance it doesn't disappear, instead it is locked in the state variable. Only after the **eject()** function is triggered is the full token sent to the Ox contract and paid to miners.

It is possible to buy more dust to increase the gravity of domains for longer. The rules are slightly different however: for above 1 Dust, the rate of emission for dormant domains is 1024 microns per epoch. If a given domain is powered with 5 Dust and left unattended, for the first 4 epochs 1024 microns will be taken each time until the balance reaches 1 Dust, at which point it decays at the normal logarithmic rate of deterioration. Which in this case is a total of 14 epochs before the **eject()** function is triggered.

At any time the user can trigger the **restore()** function which restores the domain to its initial gravity. Prior, any Dust that was emitted was locked in the **emission** state variable and can't be spent or sent away. If the user triggers **restore()**, the Dust will be taken out of lockdown and restored back in full.

-Clusters-

Clusters are top level domains within Nebulis and are associated with their own storage. To start a cluster it is necessary to amass 1024 Dust, a task which is defined and executed within the scope of the root Nebulis contract.

It is necessary for the user to have a certain amount of skin in the game before being able to create a new cluster. This acts as a basic guardrail and a proof of demand, while still being significantly easier than current procedures for creating top level domains.

The name"Nebulis" itself evokes the idea of a vast nebula, an interstellar cloud of dust and gases, hydrogen and helium where stars are created. Over a period of time this interstellar medium amasses, until the material collapses under its own gravitational weight and forms new stars.

The **amass()** function creates a new ID and object which any other Nebulis user can send Dust to as long as they have the ID.

There are a few parameters that it is possible to set when creating clusters, but most notably is character modules, which is the range of characters which are valid in a particular domain. This makes a difference because each index has its own storage contract, and the larger the character set activated the denser the cluster will be and therefore the more gas it will cost to interact with it. By default two character sets are activated on the creation of a new cluster: basic latin (a-z), a dash (-) and digits (0-9). There is a possible choice of six character sets available to activate if the founders of a cluster choose:

1. Basic Latin
2. Digits
3. Punctuation
4. Arithmetic Operations
5. Latin Extended (Uppercase)
6. Currency

The character sets are UTF-8 encodings, and it is also possible to install a custom character set, which is useful for clusters that wish to activate other alphabets such as Cyrillic or Mandarin characters. This, for example, is the character set for digits: **"\x30\x31\x32\x33\x34\x35\x36\x38\x38\x39"**

You can look up the full standard for reference at www.unicode.org. Installing a custom character set involves finding the UTF-8 codes of the set, and feeding them as a byte parameter (formatted above) into the **customize()** function in that particular cluster's Parser contract. We will discuss how such governance decisions are made and permissions work later in this memo.

Clusters are also the level at which registrations are made. The flow for registering a domain is this:

1. Check availability of domain via Whois.
2. Create your own Who or Zone contract (Zones are more complicated to administer).
3. Get the address of the cluster.
4. Invoke the **genesis()** function with required parameters.

If no "Who" address is assigned in the parameters, the function will automatically create one for you.

-Pointers-

We will colloquially refer to URL's as "pointers", although it is important to note this is not to be confused with pointers in some object oriented languages like C++ which refer to the memory address of a variable.

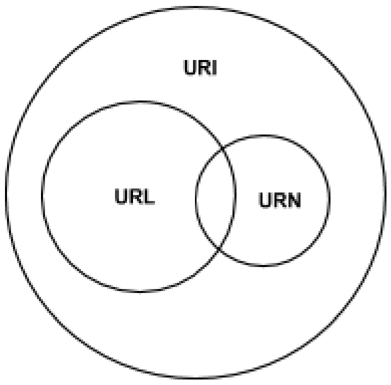
A pointer in the Nebulis context is another word for URL, a UTF-8 encoded string which can be used to fetch an IPFS link to the resource.

URLs were defined in RFC 1738 [7]. The following is a description of the difference between URIs, URLs and URNs.

- **Uniform Resource Identifier:** A generic way to identify any given resource by name or location or both, using a specific protocol.
- **Uniform Reource Locator:** A type of URI used for web addresses, which begins by defining the access mechanism (e.g. http://, ftp://. ipfs:// etc).
- **Uniform Resource Name:** A complement to URLs which can define and resolve queries to particular namespaces, such as ISBNs and telephone numbers.

URLs and URNs are both URI's, but a URI is not necessarily a URL or a URN.

The following diagram is a description of the relationship between the three:



For now Nebulis will focus on URLs, but will extend it to include the registration of all kinds of identities, namespaces or handle registries.

There are four elements to the pointer:

1. The ox: **ipfs://**
2. The prefix: **www** by default, although custom prefixes can be stored.
3. The midfix: **hello**
4. The postfix (cluster/tld): **world**

As for the scheme, it should be noted that the canonical syntax for IPFS ox is /ipfs/ instead of ipfs://, and was discussed in a now archived thread [5] on Github. This is also the syntax used in the official HTTP gateway- https://ipfs.io/ipfs/. The purpose of this choice is to keep web links harmonious with the Unix file system style.

The reason for the more traditional syntax within Nebulis is because this can be more easily applied within the context web browsers and user expectations. The objective should be to make the shift to the new web as seamless as possible- ideally unnoticeable. Browsers should be able to switch between http and ipfs without laborious attempts to rehabituate users to the aesthetic preferences of programmers.

-Storage-

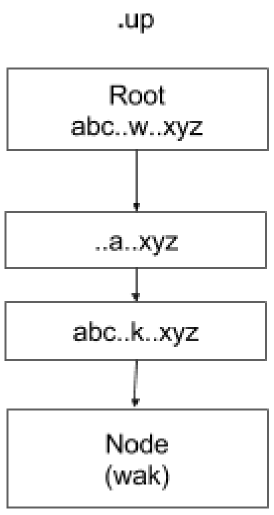
Each cluster has its own storage, which in the full release will be indexed to three levels of depth into the pointer. Let’s say we have a particular pointer:

ipfs://www.wake.up

This will be stored in the “up” cluster. The storage system has three elements, the root, branches and node. The root of the storage system contains search functions for traversing down the tree to find nodes, as well as setter and getter functions for updating state and creating new branches and nodes where none have existed before.

The reason for having several layers of depth is to avoid hitting the block gas limit as the database expands, as would probably happen if the database was kept in a single storage contract. Early on however it may be excessive to index to three levels into the string; and may impose slightly higher gas costs on early users who would have to pay the gas cost of spearheading new storage nodes where none have previously been created.

Considering that in the web there are over 215 million domains, it is important to build any system with the expectation of large sizes. If we have a cluster with a legal character set of 34 characters (a-z, 0-9, -), that means each cluster could have a maximum tree of over 39.000 storage contracts, each of which is more than capable of handling hundreds of thousands of individual entries without blowing the gas limit. For clusters which have authorized an expanded character set, the range is even higher.



-Governance-

In any smart contract system, one of the most important issues to discuss is governance and decision-making. Some systems like The DAO have attempted to implement some variant of democracy, where token holders vote on how funds are spent on their behalf. Others have implemented a representative model where curators or groups of curators take a decision on behalf of users, which the users agree to, and others some variation of both.

The main areas of conflict and decision-making revolve around a two major issues:

1. **Phishing:** a form of fraud, where an attacker would mimic a site on the http web (on ipfs) in order to trick a user into submitting sensitive data. For example, if Bank of Antarctica has a log in on the web without running a synchronized domain on Nebulis, there is a risk of the same domain being snapped up and mirrored in a fraudulent manner.
2. **Squatting:** buying “ipfs://www.google.com” (etc.) only for the sake of reselling it back at an exorbitant rate.

There is no hard and fast way of preventing these behaviours from occurring, nor are the costs of absolute security and predictability worth the benefits. But there are a number of approaches to mitigating it, without violating our third constraint of finality of ownership. In many theories of property rights it is considered to be permissible to act in any respect as long as it does not interfere with anyone else's equal freedom to act. This is the essence of John Stuart Mill's harm principle. One's right to swing one's fist ends where the other's face begins, and vice versa. Furthermore, one can cooperate or deal with others on the basis of informed consent.

Some would argue that a fake Bank of Antarctica website attempting to trick users into giving up their passwords and usernames is permissible because on some level it is "voluntary". However the key component of consent is that it is informed and warrantied; that it is based on full knowledge of the agreed terms. Phishing, while on a very superficial level is "voluntary", fails the test of informed consent.

The first obvious step of mitigating these behaviours is that there should be an active community in Nebulis which sets certain habits of behaviour. This is critical; many negative patterns can be curtailed informally by a positive and engaged community before it becomes a problem.

The second step, building on the first, is there should be a repository for "flagging" such examples, and perhaps browser extensions which can provide a warning to prevent these problems from occurring.

The third step is that clusters should have their own curators which are elected by stakeholders, who are permissioned under multisignature constraints to trigger the **eject()** function on particular domains which are being used in a provably fraudulent manner. This should be a last resort and subject to scrutiny by the community of particular clusters, and it should be possible for stakeholders to replace curators if this privilege is ever misused to curtail a user's legitimate property rights over a domain.

Cybersquatting is a little more complicated, since domains have been fairly acquired. Squatting is not technically wrong, although it is a nuisance. Some business models such as domain registrars depend on some form of "squatting" or administering Nebulis on behalf of customers. Therefore the third step of forced ejection should not be employed, as this would be a violation of property. Only the first two steps of informal community enforcement and browser flagging should be used in aggravated cases.

-Resolution-

Resolution means matching a pointer to the hash of a resource if it exists. There are two kind of resolution in Nebulis: formal and informal. Formal resolution is achieved through querying the Resolver contract which traverses the whole system in order to find the IPFS link at the end. This can be also considered a form of crypto-legal proof.

Recall that the reason for using a blockchain and not a regular database is that it has features which guarantee the mathematical veracity of the information stored on it, without the need for a trusted authority overseeing it; while Ethereum allows that security along with a much richer set of conditionals and instructions. In this context, ownership data is validated by thousands of nodes around the world.

The downside of formal resolution is that it is expensive. Queries that traverse the full tree cost gas. If every time a user queried a web address they had to pay $0.10 to get back the resource, no one would use the internet.

That's why we need informal resolution; which is to say, off-chain simulacra of the full state of Nebulis which can instantly retrieve the resource. Ideally multiple simulacra should be available and run separately, including regular databases or fast private chains, so that any regular browser query can double or triple check. Then if there is ever a dispute or any doubt about the veracity of simulacra it is possible to formally resolve a resource.

-Caching Verbs-

In Solidity, events are a convenience utility for getting back return values in the front-end when functions are executed. You can declare an event in a contract then invoke it later on, feeding in the required parameters, like the followed "update" example:

```
event Updated(string type,
              bytes32 id,
              uint time,
              bool private);

Function Change(...) returns(bool) {
       …
       Updated("Customer",_id, now, false);
}
```

It is possible invoke the "Update" event at any time with different parameters. As long as the listening node has the address of the contract, the front-end can listen to the logs and display the information from the blockchain to the user.

The purpose of IPFS caching verbs is to provide a simple syntax of Ethereum events so that the state of Nebulis can be recreated by various simulacra. All of this can be conceived as similar to how Google "crawls" and indexes the web. When you do a Google search you are mostly searching through this indexed cache. They are also similar to HTTP verbs like Get/Post/Put/Delete- the difference being that caching verbs are descriptive instead of proscriptive.

Using these event verbs it will be possible for any party to examine and audit the Ethereum history and recreate the full Nebulis database off-chain (provided they are working with a full node, since not all clients store the full logs).

1. **GEN:** Triggered when a new domain, cluster, zone or user is created (which are specified in the type string parameter), including Nebulis itself. Depending on the Gen type, not all parameters will need to be filled. Parameters:
    a. `string type`
    b. `string content`
    c. `bytes32 id`
    d. `bytes32 resource`
    e. `uint timestamp`
    f. `address owner`
    g. `address location`
    h. `address parent`
2. **ALT:** Triggered at any change in state, which includes resource swaps, domain transfers, dust transactions. Parameters:
    a. `string type`
    b. `string content`
    c. `uint amount`
    d. `uint timestamp`
    e. `address from`
    f. `address to`
    g. `address location`
    h. `bytes32 old`
    i. `bytes32 new`
3. **VOID:** The Void event is invoked when anything is deleted or ejected. Parameters:
    a. `string type`
    b. `string content`
    c. `uint timestamp`
    d. `address owner`
    e. `address location`

In later memos, we will describe in more detail a protocol for parameter combinations depending on the type- for example, for an “alt swap” the “amount” field can be set to zero, whereas in an “alt transaction” the and fields can be left empty.

-Gravity-

Gravity is a project which will be run as a part of the Nebulis roadmap, and will be an off-chain simulacra as indicated above. The Gravity database will provide the link between Nebulis and browsers, including tools and a command line interface for interacting with with the system. The tools will be aimed at facilitating existing browsers to support IPFS and it will also be possible for other parties to clone the database and run independent simulacra for security purposes.

-Roadmap-

To begin with, Nebulis will be incorporated as a nonprofit foundation in order to support continual development not only of the core system on Ethereum but also the surrounding tools. The goal would also be to provide support for the IPFS project over the long term, while also promoting the ideals of decentralization and privacy online.

We will aim to release the core Ethereum system in Q4 of 2016 and to launch Gravity soon after.

In Q1 of 2017 we will aim to launch a browser called Nebula, which will be compatible with IPFS, Gravity and Nebulis.

In addition, we will facilitate developer outreach and education for web developers in transitioning to developing on IPFS and Ethereum.

There are two primary namespaces on the Internet- DNS and the IP-address namespace. A long term goal of Nebulis would be to replacing the IP-addressing protocol for hardware with a hash based system.

Future updates to the Nebulis umbrella will be outlined and presented in a similar format as this; as public memos for feedback.

-References-

[1] Juan Benet- IPFS: Content Addressed, Versioned, P2P File System

[2] BBC- US Ready to hand over internet’s naming system

[3] Nick Johnson- Ethereum Naming System

[4] Satoshi Nakamoto- A Peer-to-Peer Electronic Cash System

[5] Vitalik Buterin - Ethereum White Paper

[6] Standard URI for IPFS and IPFS- #Issue 1678

[7] Tim Berners-Lee et al- RFC 1738