Holoweb: The Holonic Web
A Decentralised Network-Independent Protocol and Application Framework for Global Perceptual-Semantic Communication and Enhanced Sense-Making

Chris Larcombe, December 2020
https://larcombe.io

DRAFT PAPER *please note: a revised and more complete version of this paper will be released with the holoweb prototype implementation at a later date*

Abstract—**A nature-inspired protocol and framework is introduced for a globally scalable decentralised perceptual-semantic (holonic) web, prioritising sovereignty, enhanced sense-making, interoperability, privacy and efficiency.**

The approach is novel and alternative to the currently established web3 methodologies and semantic web standards, moving beyond conventional set theory and document-based data modelling conventions.

The design provides an alternative to HTML and the DOM, URI/URLs, DNS, HTTP, and a number of other web technologies and conventions.

Contents

I. Introduction

Since its introduction in 1990s, the Internet has evolved through multiple versions of the World-Wide Web [3] (“the web”), from a web of simple static documents to a web of increasingly complex and dynamic social applications.

However, in 30 years the basic structure and foundation of the web has remained largely unchanged in terms of its fundamental ontology. This ontology is one of documents, pages, files and folders, inspired by the material-physical office environment.

In this model knowledge and information is primarily represented as *(hyper)text* in documents, stored in *files* on *web servers*, which are then made accessible, visible and interactive via a *web browser*.

Information is *(hyper)linked*, by linking document objects (‘page elements’ within a *Document-Object-Model* or DOM) to other documents, containing more linked elements, etc. enabling a navigable cyberspace of explorable documents.

This model has proven tremendously successful and perhaps its simplicity has been key to its global and multi-cultural mass adoption. However, certain limitations introduced at the level of its foundational ontology, and at the level of its implementation (how it is being used and monopolised), are becoming increasingly relevant and now need to be urgently addressed.

A. *Limitations of the Current Web*

a) No ownership of identity and data: In general, individuals do not have physical ownership of their online identity (or identities) and the data they generate and use. In such cases the accessibility of personal identity and data depends on the actions and authority of a company or service provider operating a remote server (where identity and personal data is hosted).

Ultimately this limits the accessibility of information and creates (unnecessary) dependencies, which can result in temporary or permanent loss of access to personal identity and data, in a variety of different circumstances.

b) No sovereignty of perception: The centralised storage of personal, social and public data on a remote web-server means that data accessibility can be limited and controlled. A side-effect of this is a decreased ability, on the part of the end-user(s), to determine how data is displayed, experienced, and ultimately comprehended (because it is not directly or fully accessible). How much data is made available and precisely what data is made available is determined by the service provider.

Individuals also have a limited choice in how they perceive data online because the data is typically *exclusively* displayed in a fixed user interface provided by the same company or service provider hosting the data.

The interface is provided to the end-user simultaneously with the data, when it is served from a web-server in the form of a website page. On the current web it is not possible to switch or determine the interface being used to display the raw data. This inherently limits *sense-making*, by essentially restricting the end-user to one fixed perspective on the data being accessed.

Furthermore, because data is not directly accessible, and because there is an additional 'economic' incentive to sequester it, this also makes aggregating data from multiple sources, websites and applications challenging, which in turn further limits sense-making. There is also no standard for knowledge representation, and no formal data layer independent of interface (page layout).

c) Limited accessbility of public and social data: On the current web, the *vast majority* of our 'public' and social data is provided under the aforementioned limitations, where access to data is restricted and sense-making is limited.

While this situation can be *visualised* as being denied access to large database, the propagation of data between people (the spread of memes and information) through the *human* 'social graph' (network), is another useful perspective to take when considering these limitations.

Under present circumstances, our individual 'reach' (connectivity) is restricted, monetised, and therefore significantly limited and economically biased. Without an alternative for comparison, it is difficult to imagine the effect this has had and is having on our world-view and relationships (personal, societal, professional, etc.).

It is challenging to imagine what we might know, see, and understand, if the totality of our global public data and socially accessible data was fully accessible and downloadable, and how our collective world-view and relationships may differ as a result.

If our public data was actually public, it would be possible for anyone to create their own models and maps, ultimately enabling a more accurate, authentic and complete perception of society and the world as a whole.

d) The erosion and elimination of privacy: Another side-effect of the current design of the web and its dominant usage pattern is the reduction of privacy online, and in life in general, since our lives are *increasingly* linked to and broadcasted over the internet. This has particularly escalated in the year 2020.

In most cases, almost all of the data we generate is stored unencrypted on web-servers, making it possible for companies to process, use, distribute and sell our private and social data (data shared 'privately' in a social network), which we legally agree to by accepting their privacy policies and terms of use. In addition to this, our behaviour online (the way we browse and use websites) is continuously tracked[1], generating additional data, which is then combined with other meta-data such as device information, further eroding privacy.

Ultimately, statistical abstractions and models are created from our combined personal and meta-data, which are then monetised and/or privately traded on the global market to undisclosed parties.

e) Influencing human perception and behaviour: After obtaining statistical abstractions and models of our private, social and 'public' data, known and unknown parties are then able to use this knowledge to their advantage by privately and strategically influencing human perception and behaviour. This often occurs through the same systems that produced the data-sets in the first place, though it is not limited to this.

The potential for companies to instigate this process arises from the aforementioned lack of perceptual sovereignty (I-A0b) and limited data accessability (I-A0c), due to the inherent design and technical usage pattern of the web.

In practice, the 'influencing' of human perception and behaviour takes the form of *personally targeted* 'advertising', notifications, filtering ("filter bubbles"), ordering of search results, deletion of data/accounts, manipulation of view-counts and subscriptions, etc. all *covertly*, without any disclosure of 1) what algorithms are being used, 2) how our individual and collective perception is being directly influenced, 3) how our attention is being traded 4) who our individual/collective attention is being sold to and managed by.

Understandably, the vast majority of people are still unaware that this is occurring, particularly *the scale* on which this is occurring, and of the incredibly powerful effect this has had, and continues to have, on their personal beliefs, perceptions, decisions, relationships, and ultimately their lives.

Democracy, let alone self-governance, are arguably impossible at scale in such circumstances where the global information-space is monetised, controlled and limited to this extent. Never before has this been such an issue or threat to our health, freedom, economy and environment, nor to our individual and national sovereignty, than it has been in the year 2020.

In summary, these problems and limitations are a combined result of both the design of the current web and the way it is being used, the latter being linked to the global socio-economic, cultural and political landscape. In the following sub-sections we address these issues in the context of the design of the Holoweb, introducing the *Holoweb Protocol* and its sub-protocols, before more formally defining the Holoweb in Section II.

[1]Being continuously bombarded and interrupted with tracking requests, where the most/only convenient option is "Accept All", is frustrating and time-consuming for anyone who wants to opt-out.

B. Decentralisation

Many problems associated with the current web are commonly recognised to be arising out of excessive centralisation, particularly the centralisation of data storage.

In recent years there has been a reaction to this with an effort to build decentralised alternatives: decentralised webs [24, 14], messaging and social networks [7, 25, 10, 11], storage systems [9, 26], media platforms [30, 22, 31], communication and collaboration platforms [18], and decentralised application platforms [13, 20, 16, 28, 23, 27, 21, 17].

The vast majority of such efforts, following in the footsteps of Bitcoin [6], are based on or involve (as a required component) adopting a particular blockchain *methodology* that enforces a single consensus (i.e., "*the* blockchain"), requiring a consensus algorithm (typically variations of 'Proof-of-Work' and 'Proof-of-Stake'), and an associated crypto-currency. In such cases *most* or all of the following apply:

- Authority is centralised to a single blockchain hosted on a single network, creating a single point-of-failure, even though the physical network itself is decentralised
- In the most common case of Proof-of-Work, *mining* (competitive hashing) is extremely economically, environmentally and computationally expensive
- Scalability is limited because all devices (full nodes) in the network must host the entire blockchain, which can become extremely large due to there only only one blockchain (representing the single consensus)
- For there to be a single blockchain on a single network, there is typically a single shared codebase of software being deployed to all nodes in the network; this usually depends on a single, centralised core team of developers, who may also depend on centralised servers for their work and/or collaboration
- In some cases, the blockchain can only be used via a centralised web API and/or centralised exchange (to obtain the crytocurrency required to use the blockchain)
- The blockchain is public, which is not suitable for exchanging private data, even in an encrypted form
- The sustainability, stability and scalability of the network often depend on a cryptocurrency, which may be affected by unpredictable global market dynamics and potentially targeted market manipulation
- Peer-to-peer communication isn't possible without mediation through the entire network, which is limited, since there is no possibility of a peer autonomously communicating directly with another peer
- The design pattern of a "trustless network" is predicated on a lack of trust between human beings
- The security of the network and integrity of its data is based on the assumption *and trust* that no party on the planet with relatively[2] substantial resources (which are easily proven to exist in every case) will at no point choose to use their resources to take control of the network or shut it down

While this design pattern, the web3 status quo, may be appropriate for certain applications and use-cases, it is does not appear to be well suited to the task of building a global decentralised *web*, which requires extreme scalability, flexibility and resilience.

In our opinion, a decentralised web comparable to the current web in scale cannot exclusively depend on the success or scalability of any single network, with its particular technical, social and economic dynamics. We consider this to be the case even if that network is decentralised or distributed in its topology.

1) Network-Indepedence: Holochain [20], a distributed application platform, and Scuttlebutt [25], a decentralised social network, are two pioneering projects where blockchain technology is used without depending on a singular blockchain storing a shared global consensus. Rather, blockchains are multiple and agent-centric (identity-centric), storing an append only log of messages or entries authored and signed by the associated identity.

The Holoweb operates in a similar fashion to these two projects, where blockchains are multiple rather than singular, but differs in that there is explicitly no bias towards or dependence on any particular network implementation or topology (Holochain: a fully-connected topology / DHT – Scuttlebutt: social-graph topology / distributed relay network), and without being limited to one blockchain per identity (Scuttlebutt) or one blockchain per identity per network (Holochain).

The Holoweb is explicitly *network-independent* (network-agnostic) in its design, making no assumptions about *how* communication occurs between peers. This ensures that data integrity within the Holoweb is not only decentralised, but completely independent of any particular network or network implementation *and its associated dynamics* - it also means that the Holoweb is able to use any topology, and mix topologies, in order to meet the maximum possible number of use-cases. Our methodology is similar to that of PLAN Systems [18] other others [11, 19], using the concept of pluggable storage/networks.

Ultimately we consider this to be a necessary and stronger form of decentralisation, required for the Holoweb to function on a global scale. It also means that by design, the Holoweb is able to be integrated into (or use) multiple existing decentralised networks, such as IPFS [9], Holochain(s) [20] and Scuttlebutt [25].

This is summarised as follows in the *Holonic Web Network Protocol* (HWNP)[3]:

- **HWNP:1** the Holoweb is *network-independent*
- **HWNP:2** a holoweb can be stored and/or shared via one or more networks

[2]Relative to the size (total nodes) of the blockchain network.
[3]Definitions of terms are provided in Section II.

- **HWNP:3** a holoweb stream can be stored and/or shared via one or more networks
- **HWNP:4** holoweb nodes, percepts and blobs can be stored and/or shared via one or more networks
- **HWNP:5** the topology and design of a network used, serves to greatest extent possible the use-case and desires of the individual(s) using it to store and/or distribute holoweb data

C. Identity and Data Ownership

In the Holoweb identities are self-sovereign, i.e., self-generated, self-certifying, and self-hosted. Personal (self-generated) and downloaded data is likewise self-hosted by default. Self-hosted means that data is stored on a personal device, such as a laptop, mobile phone or personally managed server.

Like many other web3 projects, self-sovereign identity is enabled through public-private key cryptography, where a public key represents an identity, able to cryptographically sign data in order to prove authorship, ownership, agreement, etc.

This is formalised in the Holoweb through the *Holonic Web Identity Protocol* (HWIP) and *Holonic Web Storage Protocol* (HWSP) as follows:

- **HWIP:1** an identity of a peer is represented as a cryptographic public-private key pair
- **HWIP:2** an identity of a peer is socially and/or publicly represented as a public key (the public key of the public-private key pair, defined in HWIP:1), referred to as an *identity-key*
- **HWIP:3** identities are self-generated
- **HWIP:4** a peer can self-generate multiple identities

and,

- **HWSP:1** identities are self-hosted
- **HWSP:2** personal data (self-generated data) is self-hosted
- **HWSP:3** downloaded data is self-hosted (cached)
- **HWSP:4** to the greatest extent possible, data is physically stored where it is physically accessed
- **HWSP:5** to the greatest extent possible, data is backed up securely (ideally in multiple physical locations)

And finally, appending to the network protocol:

- **HWNP:6** to the greatest extent possible, all private data generated by an individual shared in a social context, can later be remotely deleted (unshared) by that individual – any exception requires the full informed consent of the individual before data is shared (over the network)

Ownership in this section is approached from a physical and practical perspective, rather than a legal perspective.

D. Private and Public Data

On the Holoweb privacy is built-in by default through the adoption of both *end-to-end encryption* and *on-device encryption*. This is implemented in the Holoweb network and storage protocols as follows:

- **HWNP:7** all data and meta-data sent over a network is always end-to-end encrypted
- **HWNP:8** encrypted data is not stored on or distributed via a public network (where data could be publicly downloaded in an encrypted form) - any exception requires the full informed consent of the individual encrypting the data
- **HWNP:9** to the greatest extent possible, the identity of an individual using a network remains anonymous

and,

- **HWSP:6** all self-hosted data and meta-data is stored in an encrypted form, and only ever stored in a decrypted form temporarily in RAM (volatile memory)
- **HWSP:7** high entropy encryption keys are used for encryption in every case
- **HWSP:8** where possible, two-factor encryption is used
- **HWSP:9** where possible, private data is only ever decrypted on a device where the hardware, firmware, software, and operating-system are fully auditable (open-source)

In order to ensure that public data remains public, and able to be used and aggregated freely, the following is appended to the network protocol:

- **HWNP:10** to the greatest extent possible, all publicly shared data (public data) is fully and freely accessible/downloadable at all times

E. Beyond Documents and Objects

As previously mentioned, the ontology of the World Wide Web has remained relatively unchanged for 30 years. This is the ontology of the material-physical office environment: files, folders, documents, pages, objects, etc. which is also adopted and used in our current computer operating systems and offline software applications.

One of the intentions of the Holoweb is to enhance sense-making and interoperability in general, which we believe requires a new data structure and an alternative approach to knowledge representation (see I-E1 below), with a new complimentary concept of 'application' (see I-F and II-D).

1) Holonic Mapping: In previous work [8, 12, 15, 29], a real-time collaborative mapping technology called Holomap[4] was developed capable of mapping objects of *perception* called 'holons', in a Non-Euclidian (fractal) space.

A *holon* is a whole-and-part, i.e., something that is both a whole and a part (of a larger whole/holon) simultaneously. However, in the context of Holomap the term was *extended* to refer to fractal *perceptual-semantic* objects within consciousness ("subjective experience", 'realities'), with the additional property of the holons being able to be nested inside themselves and located in

[4]Previously called Waysphere, Noomap, and now Holomap.

more than one place simultaneously (mirroring the idea of quantum super-position).

When implemented in software, this creates a fractal navigable space, *a web of holons*, where a user can zoom in and out to navigate between linked holons. The previous name for the software was Noomap, since *a holonic map* is essentially a mind-map ('noo' coming from 'nous', meaning mind - also relating to 'noosphere', the planetary sphere of mind and mental phenomena).

Anything within consciousness, any perception or component of perception, and thus any "thing", can be represented (mapped) using holons. From the perspective of Holomap everything is a holon, and every holon is a particular type of holon. Holomap also defines *holon-type* holons, which are holons defining types of holons. This relates to the concept of ontology, i.e., holon-types define ontologies. Different use-cases require different perspectives, different ontologies, and therefore different types of holon.

Holomap was successful as a mapping technology and prototype Holoweb[5], but demonstrated several limitations:

1) *The data structure was still partially object-based*
2) It was centralised in its design
3) It featured a single fixed interface (though others were developed and connected in a research and development context)
4) The meta-ontology (of holon-types) was too simplistic and limited

While holons as used in Holomap demonstrated a potential to be considered as a reasonable replacement for hypertext and HTML, the recomposability of the data was still inherently limited, since holons were encoded as atomic *objects* with *attributes*, encoded as linkable JSON objects (with key-value pairs representing different arbitrary attributes with assigned values) - the holons were linked extrinsically (only) by link objects.

This resulted in holons being created with potentially large sets of attributes, which themselves could not be recomposed, reused, or referenced, etc. In other words, while the holons themselves were recomposable, there was a risk of them becoming large 'blobs', containing what could be considered parts (sub-holons), represented not as holons but as attributes.

2) Holoform: For interoperability between software applications, a common standard data structure is required. However, greater interoperability is desired not only between software applications, but also between human nervous systems, independent of culture and language. This could be called perceptual-semantic interoperability. We have found that this requires going beyond the present established convention of using a document or object model *as the base-level* for representing information, as in the current web, the RDF-based semantic web [5, 16], and other web3 projects (which remain founded upon a physical-material ontology of documents, objects, files, folders, etc.).

This is the case in part because the idea of documents or objects with attributes is in fact subjective and observer-dependent (a convention of human perception), and the distinction of whether a component of our perception is considered to be an object itself or an attribute of an object is also subjective. Furthermore, 'documents', 'objects' and 'attributes' are themselves part of a specific ontology.

The Holoweb transcends this ontological barrier through the use of a neurologically and biologically inspired meta-ontology, informally called *Holoform*, which facilitates the definition and use of observer-defined ontologies, such as 'document' and 'object', in such a way that *every* definable element within perception/consciousness can be universally expressed as an addressable *semantic node*, that can be re-used within specific compatible perceptual-semantic contexts. In the Holoweb, everything is a *node*, or a *perceived* set of nodes called a *holon*[6]:

node
1) A circularly defined finite entity of structured data, which many contain references to or directly structurally contain zero or more nodes.

(perceptual-)semantic node
1) A node of a specific and well-defined (perceptual-)semantic type.

holon
1) An apparent or perceived set of perceptual-semantic nodes.
2) A recomposable fractal unit of perception, which can exist in multiple places and times simultaneously.

This ensures that no matter what perspective is taken, it is always possible to interoperate and translate between any ontology, without having to fundamentally restructure data. In this case, an object, from the perspective of one person, can behave as an attribute of an object from the perspective of another, simultaneously, without any modification of data.

This also means that data is completely recomposable, to the extent explicitly defined by the author of a holoweb ontology (called a holoweb *DNA*; see Section II-A5), specifically, when defining types of *semantic atom*, which are a type of semantic node.

a) Semantic Atoms: Semantic atoms express the smallest meaningful elements of perception and meaning, within a specific perceptual-semantic context/domain (holoweb). Semantic atoms can then be combined into semantic *sets*, *lists* and *links* (3 different types of semantic-node), forming more complex semantic 'objects' called *holons* (as defined above).

These four fundemental types ('ASLI'): atom (A), set (S), list (L) and link (I), define the meta-types of nodes

[5]Holomap (Noomap) may be considered as a centralised single-interface simulation of the Holoweb. The envisioned evolution into a Holoweb was mentioned in [8] (2012) and [12] (2014).

[6]The word 'holon' is redefined in the Holoweb, following previous work where a different definition was used.

within the Holonic Web, analogous to the 4 nucleotides of DNA (‘GACT’). Different *types of* atom, set, list and link, can be defined in a holoweb DNA and then instantiated later, to abstract and express any perception (anything within consciousness).

The types of node in the Holoweb are defined more formally in Section II.

When two separate holowebs with two separate ontologies are created, in the ideal case there exists a single, shared universal or global library of DNA (node types) to draw upon, in order to maximise interoperability – such that node types are reused rather than being recreated. This also saves a holoweb developer time having to create something that already exists, and maximises perceptual-semantic (and linguistic) interoperability.

Where this is not possible, adapting between two DNAs should be relatively simple, either through an adapter system or through minor code modifications. In other cases, it would also be possible to programmatically translate atoms into sets, or sets into atoms, etc. if absolutely necessary. Using a single shared data structure already makes this significantly easier than it would be on the current web (where APIs are required and each API returns data structures that do not conform to any particular web standard).

The following table provides a comparison between the Holoweb and the current World Wide Web:

World Wide Web	**Holonic Web**
Document-Object-Model (DOM)	Perceptual-Object-Model (POM)
Document, Object	Holon (II-A3)
Page	Space (II-C)
Files	Nodes (II-A1)
Physical-material	Quantum-Holonomic
Euclidian (2D)	Euclidian (2D/3D) within Non-Euclidian (fractal dimension)
Set Theory	Hyperset Theory [2]

In summary:

- the current web is based on the concept of documents, objects, pages, page elements, etc., which are physical objects in objective 3D Euclidian space, corresponding to a physical-material world-view – digital documents, pages, page elements, objects, etc. are presented in a 2D Euclidian cyber-space. Time is not explicitly included in the ontology and is typically perceived as an independent objective spatial or space-like dimension.
- the Holoweb is based on the concept of holons and nodes, which are non-physical perceptual-semantic abstractions in Non-Euclidian/multi-dimensional space[7], corresponding to a quantum-holonomic world-view – digital nodes and holons are presented in a 2D/3D cyber-space. Time is considered as a simultaneous intrinsically-existing set of moments (times) mapped with nodes/holons, and ‘space’ as an experience/structure of/binding time(s).

[7]Forming ‘objects’ in apparent 3D Euclidian space in non-local consciousness.

F. Perceptual Sovereignty

On the current web there is a tight coupling between interface (websites, web-applications) and data, both are provided simultaneously by a web-server when loading a website, and there is no built-in way to switch the interface (how data is being presented). Each website has its own separate interface and database, where data is dynamically served from a back-end (server application, database, server) to a front-end (HTML page, web browser). The interface is held within a web-page, and is represented programmatically in a Document-Object-Model (DOM).

The Holoweb departs from this model by breaking the tight coupling between interface and data, instead considering them as belonging to two separate and independent layers, to the extent that interfaces become *switchable* lenses (that can be used anywhere on the Holoweb, where there is compatibility). Ultimately, as a result the end-user is always in control of how they perceive all data on the entire Holoweb.

A further difference to the current web is that pages are replaced with *spaces* (see Section II-C), which contain *suggested* couplings between interfaces and data. The couplings are suggested and not fully determined, providing the end-user with the ability to ultimately override the interfaces being used in a space, with their own interfaces. This ensures sovereignty of perception on behalf of the end-user, while still enabling a provider with the means to present data in a specific way. Additionally, enabled by the recomposable data structure of the Holoweb, any data found in any space(s) can be re-linked from an entirely new space of the end-users’ creation, and then perceived through interfaces of their choosing.

On the Holoweb an interface is called a *Holonic Application*, abbreviated to *holapp*, oApp, ddApp, or just “app”. In general, holapps have the following properties:

- They are switchable lenses, relative to the space within which they are used
- They offer a particular way of perceiving data
- They can be used anywhere on the Holoweb where there is data compatibility (based on holoweb DNA)
- They make holons (and nodes) visible and interactive
- They provide a way to interact with data in one or more holowebs simultaneously

When used in a Holoweb Browser, a ‘meta-interface’ is required to provide a means to interoperate and switch between interfaces (holapps). We call this *meta-interfacing*, since there is an interface of switchable interfaces, within a single data context.

To include perceptual sovereignty in the Holoweb Protocol, the *Holonic Web Perception Protocol* (HWPP) is defined as follows:

- **HWPP:1** all data on the Holoweb is visualised and made interactive through optional switchable interfaces
- **HWPP:2** an individual is always able to choose or switch the interface at any time
- **HWPP:3** interface-data couplings are presented and stored in spaces

II. The Holoweb

The *Holonic Web Data Protocol* (HWDP) is the adoption of the ontology (see I-E) and set of definitions and structures provided in this section (II), that define the Holoweb and its constituent parts. The word 'holoweb' is defined and used as follows:

a holoweb

1) an explicitly meaningful and purposeful communication
2) a network-independent distributed data structure of identity-centric blockchains (*streams*), sharing a singular 'genesis-block' (*web-core*) and a corresponding set (*field*) of nodes (*percepts* and *blobs*)

the Holoweb

1) the set of all holowebs (data structure)
2) abbv. *Holonic Web*: a web based on *holons* (analogous to the *World Wide Web*).

Structurally, the Holoweb is thus composed of *holowebs* (II-A), which are composed of *web-cores* (II-A4), *streams* (II-A6) and *fields* (II-A, II-A1), which are in turn both composed of *nodes* (II-A1).

Experientally, the Holoweb is composed of *(holo)spaces* (II-C), containing *holons* (II-A3) and *(hol)apps* (II-D).

Topologically, the Holoweb (data structure) is equivalent to 2-layered graph:

- **layer-1**: an unlabelled directed multigraph, with loops and 20-dimensional edge[8] identity
- **layer-2**: a labelled directed multigraph (composed of 3-tuples[9] of vertices defined in layer-1), with loops and open (n-dimensional) edge[10] identity

In terms of its representation as a holon (of holons), i.e., the *holonic web*, the Holoweb can be mathematically modelled and represented as a hyperset [1, 2] (see also Section II-A1a).

Most simply, without the temporal dimension (linkage) provided by the linear blockchains (streams), the Holoweb $\overset{\circ}{H}$ as a hyperset can be represented as follows:

$$\overset{\circ}{H} = \{W_1, ..., W_n\}$$

[8]In layer-1, each edge represents 1 of 19 types of intrinsic-link, or an implicit link between two sequential nodes within a blockchain/stream.

[9]Where the 3-tuple represents a semantic-triplet, i.e., subject-relation-object.

[10]In layer-2, each edge is a node/holon.

where n is the total number of holowebs, and each web W_i (from $i = 1$ to $i = n$) is a hyperset defined as follows:

$$W_i = \{C_i, S_i\}$$

where C_i is the immutable *web-core* (see II-A4) of web i and S_i is the (hyper)set of *streams* (see II-A6) of web i. The set of streams S_i is further defined as:

$$S_i = \{S_{i,1}, ..., S_{i,m}\}$$

where $S_{i,j}$ is stream j (from 1 to m) of web i, and m is the total number of streams (corresponding to the total apparent observers) in web i.

Finally, C_i and $S_{i,j}$ are hypersets containing arbitrary semantic-nodes of different types, which are themselves hyperset or primitive elements (only 'semantic-atoms', see I-E2a, are modelled as primitives):

$$C_i = \{C_{i,1}, ..., C_{i,x}\}$$

$$S_{i,j} = \{S_{i,j,1}, ..., S_{i,j,z}\}$$

where x equals the total number of elements in C_i, and z equals the total number of elements in steam $S_{i,j}$.

As with regular sets, elements may be shared between many sub-sets (appear in more than one place). This corresponds to the idea of holons appearing in multiple places and times simultaneously.

In this model, at time t, the hyperset $S^t_{i,j}$, $W^t_i = \{C_i, S^t_i\}$ and thus $\overset{\circ}{H}^t$, contain all versions of every semantic-node at previous times $t - t'$ where $t' \geq 0$ (as separate elements). The web-core C_i never changes (it is immutable).

While *lists* (see II-A1) are not technically sets, they have an equivalent representation in the form of (hyper)sets, where the index of each element in a list as an integer is paired with its corresponding element in a set (of sets).

The Holoweb therefore, in general, at any arbitrary point in time (at all times) can be represented mathematically as a hyperset[11].

Similarly, with the temporal dimension (linkage) provided by the linear blockchains (streams) included, $S^t_{i,j}$ of $\overset{\circ}{H}^t$ can be redefined as follows:

$$S^t_{i,j} = \{\{0, C_i\}, \{1, S^t_{i,j,1}\}, ..., \{z, S^t_{i,j,z}\}\}$$

where $S^t_{i,j,k}$ is the kth element/semantic-node (from 1 to z) in stream $S^t_{i,j}$, where z is the total number of nodes in that stream, at time t. In this case the integers represent the position of each node in the stream.

[11]This includes a holonomic representation of layer-2, defined above.

A. Holowebs

In the Holoweb, each individual holoweb ('web') can be thought of as a particular kind of decentralised network-independent conversation or communication, which peers can participate in over one or more arbitrary networks (e.g., a local WiFi network, an HTTP server, a Holochain, etc.).

Information exchanged in communication is represented in the form of holons (II-A3) composed of nodes (II-A1), which contain perceptual-semantic data, and links (II-A2) to other such holons and nodes. Within a holoweb, an identity (II-B) expresses/shares these nodes in a *stream* (II-A6), which represents a "stream of consciousness" (on a particular 'frequency'); an apparent linear succession of moments of perception.

Each peer (identity) can participate in many different conversations (holowebs) simultaneously. A single conversation (communication) can also take place between two or more peers via multiple holowebs simultaneously. The nature and structure of the communication is defined by the DNA (II-A5) of the holowebs.

Each holoweb has an immutable core, called a *web-core* (II-A4), which includes a description of itself and the necessary elements for it to be used. The immutable core is also used as a shared genesis block for multiple identity-centric blockchains, called streams (II-A6), which contain nodes of a specific type defined by the DNA in the web-core (genesis block).

Finally, each holoweb has a *field*, where immutable blob and percept nodes can be appended outside of a stream/blockchain (referenced by nodes in a stream). Percepts and blobs have no inherent association with any particular identity.

Holowebs are designed to be easy to create, requiring no programming skills. A web is first designed in a template form, and it is then compiled before it becomes operational within the Holoweb.

1) Nodes: There are 4 fundamental types of node in the Holoweb (only 3 of which are instantiated after genesis of the Holoweb):

(Node)

1) A circularly defined finite entity of structured data, which many contain references to or directly structurally contain zero or more nodes.

(Perceptual-Semantic) Node[12]

1) A node of a specific and well-defined (perceptual-)semantic type (of a type defined in a holoweb DNA), with meta-data associating the node with an identity, a location in a holoweb *stream* and a percept.
2) An abstraction of a perception in linear time
3) A node in a holoweb stream (see II-A6)

Percept (Node)

1) A node of a specific and well-defined (perceptual-)semantic type (of a type defined in a holoweb DNA), not associated with an identity or holoweb stream.
2) The contents of a semantic-node (type and links to other nodes)
3) An abstraction of a percept(ion) outside of linear time

Blob (Node)

1) A node of arbitrary data (non-semantic)

While a semantic-node essentially maps a perception (or component of a perception) from the perspective of a specific observer in linear time[13], a percept maps the experience itself as it exists in the intrinsic field ('existence') outside of linear-time, on a specific multi-dimensional 'frequency' (defined by node-types/DNA). This is why it belongs to the field of a holoweb, rather than a stream.

Each semantic-node links to a single percept. Both perception (semantic-node) and percept can be referenced directly and independently.

This means that two (or more) observers can 'collide' in sharing an experience, a percept – also implying that two *identical* experiences *are the same experience* (they are one), even though from a linear perspective they occur (or appear to occur) at two separate times.

There are 4 fundamental (meta-)types of semantic-node (and therefore percept), which are used in holoweb DNA to define different *types of* semantic-node (of these 4 meta-types). These are defined as follows:

Atom (≈atom-type)

1) an atomic semantic-node containing one *intrinsic-link* (see II-A2) to a blob
2) a node containing a blob

Set (≈set-type)

1) a semantic-node implementing a fuzzy hyperset, containing a set of zero or more intrinsic-links (to nodes)
2) a set of nodes

List (≈list-type)

1) a semantic-node implementing a vector of elements/hypersets, containing a list (vector) of zero or more intrinsic-links (to nodes)
2) an array (vector) of nodes

Link (≈link-type)

1) a semantic-node implementing a 3-tuple of elements/hypersets, containing 3 intrinsic-links (to nodes), representing a subject, relation and object respectively.
2) an "extrinsic-link"

The form instances of these nodes take, within streams, is defined in II-A6.

a) Hypersets: the design of the Holoweb departs from conventional set theory through the adoption of hypersets [1, 2], also called non-well-founded sets, in modelling observer-dependent *holons* (and nodes). In

[12]Informally a perceptual-semantic node may be referred to simply as "a node", together with "a percept" and "a blob" (while technically all 3 are types nodes).

[13]This is not depend on or require clock-time or timestamps: the linearity is already inherent in the blockchain/stream structure through recursive cryptographic hashing.

this case a set can contain recursion (loops), which are required in a number of use-cases. We found this to be more efficient and well suited in practice when defining ontologies and perceptual-semantic objects, rather than exclusively using graphs and/or conventional sets.

b) Fuzzy hypersets: fuzzy logic [4] is used to represent non-binary set membership within hypersets, enabling nodes to model 'fuzzy' phenomena, which require non-Aristotelian logic. The use of fuzzy set membership is likely to used in special circumstances, though it has a large number of use-cases (that can equally be modelled through other means using classic set membership).

c) Extrinsic-links: the link meta-type (link-type) above represents an *extrinsic-link* (defined in contrast to an intrinsic-link), since the link defined exists as an independent node itself, outside of the nodes it links together in a subject-relation-object relationship. In contrast, the intrinsic-links occur within the context of sets and lists, which link (associate) nodes through set membership (inclusion).

Ultimately, nodes are linked to each other intrinsically or extrinsically in a variety of different ways. This results in a degree of hyper-connectivity relative to the current web, where (on the current web) it is only possible to link *from* document/page elements *to* documents and files (rather than their contents).

On the Holoweb any 'thing' can be linked to any other 'thing'; that is, anything that can be represented as a node, representing any arbitrary object of perception within consciousness.

2) Links: On the current web, 'URIs' (also called 'URLs') are used to link to documents and files ('resources'), including website pages and websites (index page). This system of URLs is used in two contexts: internally, by the developer in HTML and in code, and externally, by the end-user in the web browser (entering the address in the location bar).

For these two contexts the Holoweb uses two different systems: 1) The *Holonic Link System*, for internal use, and 2) the *Holonic Name System* (see Section II-E), for external use. Both are used to address specific holons or nodes only, rather than files.

Within the Holonic Link System there are two different types of link: *intrinsic* and *extrinsic*. Intrinsic links are more fundamental and lower-level, since extrinsic links (see II-A1c) are defined through nodes involving intrinsic links.

a) Intrinsic-Links: An intrinsic-link corresponds to the idea of a mathematical set or hyperset containing an element (a primitive, set or hyperset); in other words, it represents the relationship "part of", "contains" or "inside of". For example, a car C containing one driver d and two passengers p_1 and p_2, may be represented by a set $C = \{d, p_1, p_2\}$. The definition of C as a node would then include 3 intrinsic-links, which reference d, p_1 and p_2.

There are a total of 19 types of intrinsic-link in the Holoweb, which are used to address nodes and holons in different contexts in different locations within the logical structure of the Holoweb. The intrinsic-link types are either relative (context-sensitive) or universal (absolute), in terms of their resolution to a specific node/holon.

In more advanced use-cases, when designing ontologies in the Holoweb using DNA (II-A5), different types of holon (node) can be created, which require the use of specific types of intrinsic-link. For example, it may be important in a particular use-case that a specific holon type can only contain sub-holons authored by another peer, or the author of the holoweb (web-core). In general, these special cases are based on specifying where nodes are logically (structurally) stored, at the level of the ontology.

The intrinsic-link types are summarised and defined as follows:

ID	**Name**	**Form**
1	Web	[web]
2	Local Core Blob	[blob]
3	Local Core Percept	[precept]
4	Local Core Node	[node]
5	Remote Core Blob	[web][blob]
6	Remote Core Percept	[web][percept]
7	Remote Core Node	[web][node]
8	Local Stream	[peer]
9	Remote Stream	[web][peer]
10	Local Node	[peer][node]
11	Local Object	[peer][object]
12	Local Peer Node	[peer][node]
13	Local Peer Object	[peer][object]
14	Remote Peer Node	[web][peer][node]
15	Remote Peer Object	[web][peer][object]
16	Local Blob	[blob]
17	Local Percept	[percept]
18	Remote Blob	[web][blob]
19	Remote Percept	[web][percept]

Each item in square brackets above represents a 32-byte identifier. In the case of [web], [node], [blob] and [percept] this is a SHA-256 content-hash. In the case of [object][14] this is a random 32-byte Stream-'Object'-Identifier (SOI). In the case of [peer] this is a public key (*identity-key* of a peer). All links are immutable, except for links containing [object][15].

Intrinsic-link type (ILT) 1 links to a holoweb. ITL 2 to 7 link to nodes in web-cores. ILT 8 and 9 link to holoweb streams (the links resolve to the *stream-holon* of a stream, belonging to the specified peer). ILT 10 to 15 link to semantic-nodes/holons within streams. ILT 16 to 19 link to blobs and percepts belonging to a holoweb field.

All links above are universal (absolute), except links above beginning with the name 'Local', which are relative to a stream in a holoweb.

[14]The word 'object' is used in this context specifically to refer to *mutability* and the idea of "something which changes over time" – which may be (perceived as) a node or holon.

[15]The exact node a mutable link resolves to will change as the referenced node is updated, while the SOI always remains unchanged.

For all link types, the ID (integer) of the type is pre-appended to the byte-array (link content) to identity the type of link, facilitating link resolution. Therefore, the links are a total of either 33, 65 or 97 bytes in length.

3) Holons: Holons are *perceived* sets of nodes (II-A1), which means that they have no explicit representation in data structure. Every node can also be perceived as a holon. A holon is a whole-and-part, which may contain zero or more holons, including itself, and parts of itself within parts of itself, etc. This is mathematically best represented as a hyperset [1, 2].

In the Holoweb data structure, all nodes have two universally-unique identifiers: a content-hash and a randomly generated byte-array (universal-object-ID). Holons are immutable in such cases where the nodes that compose them are linked using immutable intrinsic-links, and the opposite is true when mutable intrinsic-links are used (the holons are mutable, i.e., they are 'objects'). For most use-cases, holons are likely to be fully mutable.

Holons can be recomposed to the extent allowed by the type definitions of the nodes composing them (holoweb DNA, II-A5). This can include (re)composition across multiple holowebs, in such cases where two holowebs have compatible DNA. In this case, a holon created in one holoweb may contain holons created in another holoweb.

The Holoweb itself as a whole can also be considered a holon, i.e., the set of all holowebs, which includes all web-cores fields and streams, and all holons (nodes) within each of those.

4) Web-cores: A web-core defines a holoweb and how it can be used. Web-cores are immutable (composed of immutable nodes), and are addressed by their SHA-256 content-hash. The structure of a web-core (≈web) is summarised and defined as follows:

- Title
- Author (public-key)
- Intention ("purpose")
- Holoname (see II-E)
- Description
- DNA (see II-A5)
- Membrane (optional)
  - Read-membrane (optional)
  - Write-membrane (optional)
  - Type-membrane (optional)
  - Network-membrane (optional)
- Space ("root-space")
- Unix Timestamp (author-defined)
- License (optional)

And additional meta-data:

- Author (public-key)
- Unix Timestamp (compile-time)
- Signature (of author)
- SHA-256 hash (of web-core content)

While web-cores are nodes, they are treated as a special type, which cannot be instantiated within a holoweb stream. Instead, web-cores are compiled from a web-template, a type of node (and holon), which can be instantiated in a stream.

When a web-core is compiled all the nodes within the web-template are combined into a single serialisable and distributable package. A timestamp, signature and finally a SHA-256 content-hash are appended to the web-core upon completing compilation. The hash is used as a universally-unique identifier for the holoweb.

An optional multi-layered membrane can be added to a web-core, which further defines (constrains) how the holoweb can be used.

There are 4 types of membrane, all of which are individually optional:

Read (≈read-membrane)

1) defines which peers can read the web-core and its associated accessible streams

Write (≈write-membrane)

1) defines which peers can create a stream

Type (≈type-membrane)

1) defines the maximum quantities of specific node types which can be appended to a stream
2) defines the maximum quantities of specific node types which can be appended to a stream for specific sets of peers

Network (≈network-membrane)

1) defines which networks are allowed to distribute the holoweb (web-core) and its streams

5) DNA: In the Holoweb, a holoweb DNA is used to define the ontology of a holoweb, which can be thought of as the language spoken in that holoweb (in its streams), associated with a specific spectrum of perception and meaning – a perceptual-semantic domain'. DNA is included as part of an immutable web-core, defining a holoweb.

Holoweb DNA is composed of 4 meta-types of nodes (≈atom-type, ≈set-type, ≈list-type, ≈link-type – see II-A1), 'ASLI', in analogy with the 4 nucleotides of biological DNA, 'GACT'. As biological DNA codes for proteins, holoweb DNA codes for holons, defining how nodes can be combined into holons. Sets of meta-type nodes encoding holons are also comparable to genes.

Because the meta-types are also nodes, DNA is both holonic (fractal) and recomposable. This has a variety of advantages:

- DNA can very easily be flexibly reused in multiple holowebs, creating interoperability between holowebs, without requiring authorship of any additional code – this also accelerates DNA design, as existing DNA can be reused rather than being recreated from scratch
- DNA can be created quickly and easily, since it is a matter of instantiating nodes/holons, rather than writing code – this also makes holoweb design fast and accessible
- Parts of a DNA can be addressed, referred to, and used in arbitrary contexts on the Holoweb (as with any other node)

The DNA of a holoweb can also be considered as defining the 'frequency' (not channel) on which communication via a holoweb operates. In this context, DNA can be thought of as a 'fractal antenna' associated with a spectrum of consciousness (perceptual-semantic domain).

The aforementioned 4 meta-types, when instantiated, make use of 4 separate sub-types: ≈types, ≈valid-quantity, ≈range, ≈valid-quantity-of-type(s). Valid-quantity-of-type(s) associates a type or set of types, with a range or valid-quantity, and optionally a set of ≈intrinsic-link types.

These types are used and instantiated within a meta-type, such as a ≈set-type, to define which other types can/must be created within them, how many, and how they must be linked (through which type of intrinsic-link). Each type definition (instantiation of ≈atom-type, ≈set-type, etc.) also includes a *definition* and a *holoname*, so the ontology is well-defined and each type can be easily referenced. The holoname is required for the Holonic Name System (see II-E).

The meta-type nodes in a DNA are then used to *validate* nodes and holons, before they are appended to a stream (in a holoweb where the DNA is used). If a node is not valid it cannot be appended to a stream.

Inspired by Holochain [20], holoweb DNA can equally be used in a peer-to-peer context to validate data integrity over a network. In this case, peer data received over a network is validated, and only stored and distributed if it is valid.

The validation of semantic-nodes against holoweb DNA is summarised below, in extending the storage and network protocols as follows:

- **HWSP:10** a node must be validated before it is stored, and only stored if it is valid

and,

- **HWNP:11** a node must be validated before it is transmitted over a network, and only transmitted if it is valid
- **HWNP:11** a node received over a network must be validated before it is further processed or used, and should only be further processed and used if it is valid

a) Holochain DNA comparison: The main difference in this implementation (of peer-validated data integrity) is that a Holochain DNA is 1) analogous to a Holoweb web-core, rather than holoweb DNA (which is only part of an immutable web-core), 2) associated with a (Holochain) network, whereas a holoweb/web-core/DNA is fully network-independent, 3) compiled from code (e.g., Rust), rather than (recomposable) data structures (a holonic web-template including holonic DNA), 4) used to validate arbitrary entries (of any structure), rather than nodes of a determined perceptual-semantic (meta-)structure. 5) able to define complex protocols, requiring code, whereas holoweb DNA is (currently[16]) limited to simple quantity-based protocols. 6) associated or equated with (the idea of) (an) application, whereas a holoweb DNA is entirely decoupled from (holoweb) applications.

[16]The design could be extended to include validation code per meta-type if necessary.

6) Streams: A stream represents a communicated/expressed stream of consciousness, on a specific 'frequency' defined by a holoweb DNA (see II-A5), and optionally constrained by a holoweb type-membrane (see II-A4).

Streams 'emanate' from the core of a holoweb (a web-core; see II-A4), and are append-only. There is one stream per identity per holoweb, and in most cases a stream will only be appended to from a single location (device), belonging to that identity.

A stream is made of (semantic-)nodes, connected to each other sequentially in a linear blockchain. Each node appended to a stream must be *valid* (validated), instantiating a type of node defined in the corresponding holoweb DNA (the DNA in the web-core, associated with the stream). In computer science terms, the web-core defines the abstract classes (or schema), which are then instantiated in the streams. The web-core represents the genesis block of a stream (blockchain), which is shared with other streams (blockchains) within that holoweb (where the web-core/genesis block is defined).

Since every type of node that can be added to a stream is defined in DNA as 1 of 4 different meta-types (atom, set, list, link), this means that a stream is essentially a linear sequence of 4 different (meta-)types of node. For a node such as a set or list to be valid it must link (intrinsically) to a number of other existing valid nodes, which means that holons (sets of nodes) generally, must be created in a certain order within a stream.

A stream resembles biological RNA and the coupled functioning of DNA and RNA, where sub-sets of DNA are copied/translated into linear sequences (RNA/amino-acids, i.e., sequential semantic-nodes in a stream), which are then 'folded' into a higher-dimensional non-linear form, i.e., proteins/holons.

Semantic-nodes in a stream, via a percept, can link to nodes in the same stream or in another stream, in the same holoweb or in a different holoweb. Nodes in a stream can also link to (semantic-)nodes, blobs and percepts in web-cores and web-fields. Where nodes link to a node in a stream, the link can be mutable or immutable.

Mutable links allow a link to be maintained (rather than broken) when the linked node is changed (updated). In this case, where a mutable link is used, the link addresses the *Steam-'Object'-Identifier* (SOI) of a node. The SOI is a steam-unique identifier (unique to a stream), which enables a node to represent the idea and perception of a unique 'object' (some-'thing') changing over time.

Every node in a stream of a regular holoweb has the following meta-data attributes[17]:

- Hash of previous node/block in stream
- Hash of the web-core
- Author (public identity-key)

[17]The * symbol indicates the attribute is optional/context-dependent.

- Unix Timestamp (author-local time appended)
- Node type (hash of a (atom/set/list/link-)type node)
- * Holoname
- **Percept (ID/content-hash of a percept)**
- Version number
- Stream-Object-Identifier (SOI)
- * Hash of previous node/block (of this 'object')
- * Fork history (array of node IDs)
- Signature (of author)
- SHA-256 hash (of this node)

The hash of the previous node/block creates the basic blockchain structure. The first node appended to a stream will always reference the web-core of the holoweb it belongs to. The hash of the web-core is also included in each node, for the purpose of data portability and efficiency in certain use-cases, though in most cases the holoweb (hash) can be inferred.

The node type must equal the hash of a node in the web-core defining a type, which will be either an ≈atom-type, ≈set-type, ≈list-type or ≈link-type. The referenced percept must also exist in the holoweb web-core or field, for the node to be valid, and this percept must be of a type equal to the node type specified.

When a node is updated a new node is appended to the stream with an SOI equal to a previous node (the node being updated). The SOI remains the same, which identifies it as the same node rather than a fork; its version number is incremented and the hash of the previous version of the node is also recorded.

If a node is forked from another node (the percept remains identical but the SOI changes), then the hash of the node being forked is recorded in the fork history of the new node, which is an array of absolute (non-relative) intrinsic-links. When a *forked node* is forked (a fork of a fork), a second link is appended to the array, and so on. When a node is forked, the fork has a new SOI, the version number is set to 1, and the author may change.

While a stream cannot be forked, it is possible to migrate from a stream to another stream in a different web (which may be a fork or upgraded version). This is called *stream migration* or *web migration*. For this scenario, 4 special types of node are introduced:

Stream (≈stream)

1) represents a stream
2) contains an optional ≈space node ('root-space' of the stream)
3) contains an optional ≈migration-from node
4) contains an optional ≈license node

Stream End (≈stream-end)

1) represents the end of a stream, that a stream has been terminated

Stream Migrate (≈stream-migrate)

1) represents migrating to another stream in a different holoweb
2) contains a ≈web node (the holoweb migrating to)

Stream Return (≈stream-return)

1) represents reversing a migration, returning to the current holoweb (from a holoweb that was previously migrated to)
2) contains a ≈stream-migrate node (in the holoweb returning from)

When a stream is created, the first node in the stream is always the ≈stream node. Optional nodes such as a ≈space, ≈license or ≈migration-from can be added afterwards, and then linked to the ≈stream node (by updating it). A license or a space can be added to the stream node at any time.

When a stream is terminated using a ≈stream-end, it is no longer possible to append any nodes to the stream. This action is permanent, which is required as a security measure in case encryption keys are ever compromised.

When a user wishes to migrate to another stream (if for example, the web is upgraded), a ≈stream-migrate node is created in the stream, which links to the new web. Once a ≈stream-migrate node has been added to a stream, it is no longer possible to append any nodes, apart from a ≈stream-return node, which reverses the action of migration. A ≈stream-return node is however only valid, if it links to a matching ≈stream-migrate node in the web previously migrated to.

When a new stream is created, if the stream is a migration from a previous stream in a different web, the second node in the stream will be a ≈migration-from node, linking to the ≈stream-migrate node in the previous stream/web. This is however only valid if the linked ≈stream-migrate node in the previous web, is the final node in the linked stream, and the web linked from this node matches the new/current web.

Migration from a stream to another stream (in another web), is only completed when a ≈migrate-from node is created as the second node in the new stream (in the new web), and that this node correctly links back to the ≈stream-migrate node (at the end of the migrating stream).

When a web migration occurs, one scenario is that the author of the web is the first to migrate to a new web, which may be a higher version of the current one, and that this catalyses a 'mass migration' of peers to the same (new) web. Another scenario may be that a peer creates a fork of the web, and migrates to this, and other peers see the migration and follow.

Web migration can be restricted for a holoweb if necessary, using the type-membrane of the holoweb. It is not necessary to add the 4 stream types (above) to a holoweb DNA, since these are considered system types (special node types).

B. Identities

As previously defined in Section I-C, an identity is essentially a self-generated public-private key pair stored securely on a personal device.

Everything in the Holoweb, such as a holoweb, a node, a holon, is authored and cryptographically signed by an identity-key (the public component of the aforementioned key-pair). Two additional public-private key-pairs are created per identity (with an identity-key), called a *connection-key* a *mail-key*.

The connection-key may be used as an anonymous self-certifying identity on multiple decentralised networks, associated with the identity-key only privately between peers.

The mail-key can be used to receive encrypted messages on multiple decentralised networks, and may be associated with a connection-key, though this is unnecessary since the mail-key itself can be used as a unique identifier (identity).

This extends the identity protocol as follows:

- **HWIP:5** an identity is self-generated with a supplementary public-private key pair for signing (called a *connection-key*) for optional use on multiple networks; this key is dissociated with the identity-key for anonymity where necessary
- **HWIP:6** an identity is self-generated with a supplementary public-private key pair for encryption/decryption (called a *mail-key*) for optional use on multiple networks; this key is dissociated with the identity-key for anonymity where necessary

Identities (public-keys) may be exchanged privately and publicly with appended network information, to initiate a connection between two peers on the Holoweb. This can occur through *any* communication method, online or offline, in order to *bootstrap* a connection to the Holoweb, after which point the Holoweb itself can be used to establish additional peer connections via holowebs.

C. Spaces

A space on the Holoweb, also called a *holospace*, is approximately equivalent to a page (website page) on the current World-Wide Web. It is an addressable place where information is presented and accessed. In the case of the Holoweb, information in a space is composed of holons, composed of nodes.

In the Holoweb a space is a type of node (holon), which can contain zero or more arbitrary nodes (“data”), and a list of application nodes (“suggested interfaces”). The interfaces (applications) are ‘suggested’, since it is always possible for the end-user to override this (enabling perceptual sovereignty). The applications are included in a list form so that they can be presented in order of priority, from the perspective of the author of the space. In this case, the first application in the list is the default application to load, when viewing the space.

Spaces are loaded and viewed in a Holoweb Browser, where application code is executed with the data in a space being made accessible to running applications. What is visible in a space and how it looks depends entirely on the application(s) in the space, which are programmed to make visible and interactive specific types of node in specific ways.

When a space (node) appears in a space, that space may be rendered as an object and may optionally be made ‘navigable’, in which case the browser will load the space. This is one way navigation between spaces can occur. Another way it can occur, is when an arbitrary node is linked to another space using an extrinsic-link, which is itself a node. In both cases, ultimately the application determines how links to spaces are interpreted, but in most cases when a space is linked from any node an option will be made available to navigate to that space.

A space may contain nodes from many holowebs, in which case the end-user must be able to access these nodes over at least one network. A special node type called ‘web-link’ may be added to the space to facilitate this process (this type is also used in the Holonic Name System; see II-E), which identifies networks distributing specific holowebs.

Navigation of the Holoweb can also occur within a space, without leaving the space, when nodes are dynamically loaded and rendered by an application, essentially performing graph-traversal. In this case, the space links to one or more nodes, which are linked to many other nodes, which are linked to many other nodes, (recursively) etc.

The experience of navigating the Holoweb will thus be varied, and depend on the nature of the data and the applications in spaces.

D. Applications

Holonic applications (also called *holapps*, *apps*, or *ddApps*) are interfaces, which act like switchable lenses within spaces, for viewing and interacting with holonic data in a Holoweb Browser. Applications make visible and interactive specific types of holonic data, and can be used in conjunction with any compatible holoweb. Compatibility is based on holoweb DNA; an application is designed for specific node-types (DNA).

When a Holoweb Browser “opens a space” (II-C), represented by a ≈space node (holon), an application is selected by default based on the contents of the ≈space holon. The end-user may then select a different application if desired. When an application is selected for the first time it is loaded and initialised. After initialisation, an application function *show* is called, to display the current space. In most cases this function will execute code which results in a 2D or 3D rendering of some or all of the linked data in the ≈space holon – generally, the application will render a specific set of types of node (holon) in the space.

When another application is selected, the browser switches to that application by simply calling the *hide* function on the current application, and the show function on the newly selected application. This enables the applications to perform animations and functions which relate to switching views. At any point in time, exactly one node (holon) in the space is defined as the ‘target-node’, which represents a center-point for attention, around which other nodes are arranged. The target node/holon is set when a node/holon is *targeted*. The target-node is maintained when switching between interfaces, so that a point of data (a holon) and its associated data (sub-holons) can be visualised from multiple perspectives in a space.

Similarly, one or more nodes/holons can also be *selected*, which may be considered as a lesser form of targeting – node/holon selection is also maintained when switching interfaces (applications). How a node/holons looks when it is targeted and/or selected will depend entirely on the application and data within the space.

Targeting and selecting nodes/holons and subsequently switching rapidly between multiple interfaces, may be used to facilitate *sense-making*, where each interface focuses on displaying a specific dimension or aspect of the data.

In advanced use-cases an end-user may choose to 'share their attention' between multiple local browser sessions, or with another peer in remote browser sessions. In this case, the targeting and selection of nodes, per space, is broadcast over one or more networks. How this appears in a space will depend once again on the application being used (how a remotely targeted or selected node will be displayed, etc.).

Structurally, an application is also represented as a type of node, ≈app, which can contain one (main) ≈module node, and zero or more ≈sub-module nodes containing code. It may also contain zero or more ≈file-resource nodes. This ontology is particularly implementation-specific, and is likely to evolve and change in future versions of the Holoweb. Generally, an application will contain both code and static (immutable) resources.

The current definition of an application holon (≈app) is as follows:

- Title
- Author (public-key)
- Holoname (see II-E)
- Version
- Timestamp
- Description
- DNA Membrane (containing zero or more ≈atom-type, ≈set-type, ≈list-type, ≈link-type)
- Module
- Sub-module (zero or more)
- File-resource (zero or more)
- License (optional)

The DNA membrane of the application contains a set of holon type definitions (node-types) and is used to facilitate: 1) the coupling of apps and data in spaces, 2) interoperability between applications, 3) explication of holoweb compatibility, and 4) high-level application development.

Executable application code is contained in a singular ≈module node, and an optional a set of ≈sub-module nodes. A ≈file-resource node is a semantic atom, containing a blob of binary data in a specific encoding; the encoding used will be identified at the beginning of the blob. In future versions of the Holoweb, we anticipate that application code will eventually be represented entirely as nodes, rather than blobs containing conventional linear code written in a programming language such as Javascript.

Because applications are stored in holowebs as nodes in a holoweb stream, they have a network-independent decentralised foundation. Both the applications themselves and the data they use are (meta-)decentralised, since they are able to operate on multiple decentralised networks simultaneously.

Holonic network-independent applications can be conventionally referred to as *ddApps*, rather than 'dApps', since the decentralisation of the app is itself decentralised (i.e., decentralised decentralisation; adding an extra 'd' to 'dApp').

E. Names

The Holonic Name System (HNS) of the Holoweb is used to reference (address) holowebs and nodes (holons) with human-readable *holonames* (referred to as 'names' in this section), in an *observer-dependent* holarchy of names. Holowebs and nodes (holons) can be accessed via their *holoname* (name) in a Holoweb Browser.

Unlike DNS, where there is a single global consensus on 'top-level names' ('domain names') depending on a specific server network (the global DNS root servers), the Holoweb takes a different approach, without enforcing a singular global consensus and network dependency. HNS is network-independent and observer-dependent ('agent-centric').

In the HNS, each individual can decide for themselves what they consider to be a *top-level* name-space (of names), which collectively results in an emergent consensus, or multi-consensus, for different peer-groups – in the Holoweb there is therefore no *singular* global top-level name-space, and therefore the resolution of a holonic address ultimately depends on peer connectivity and emergent social agreements.

Names are stored in and distributed via holowebs, using a specific node meta-data attribute (called *name*, or 'holoname'). Each holoweb has a name and an associated observer-dependent name-space. Each stream (of a holoweb) has an associated set of (sub-)name-spaces relative to a holoweb name-space; one for each type of node, within which nodes of that a specific type can be accessed (by a unique name, within that name-space). This enables nodes to be addressed in the format:

∼[holoweb-name]/≈[type-name].[node-name]

The address structure enables nodes (holons) in a specific holoweb, of a specific type, to be addressed by 'frequency' – as if "tuning into" the frequency, first the frequency of the holoweb (defined by its DNA), and then a more specific frequency within that.

In this case, the node name must only be unique in the namespace created by the type of the node, within the stream of the peer, for that specific web. This means that this address may resolve to more than one node, but only if two different peers have created a node of the same type with the same name, within their streams (in the same holoweb) – this scenario is also observer-dependent and depends on peer connectivity.

When an address resolves to more than one node, the end-user would be presented with a drop-down menu of options to select from.

If no node is specified after a holoweb name, the address resolves to the *root-space* of the holoweb, which is defined in its web-core.

Since the name of a root-space is always 'root', the following name:

$\sim$[holoweb-name]

will resolve to:

$\sim$[holoweb-name]/$\approx$space.root

If the author of the holoweb has updated the root-space in their stream, this will resolve to the latest version, rather than the version defined in the web-core.

1) Inter-Web Links: To create a holarchy of names, the holonic link structure is used via a system link-type of node called Inter-Web Link, with the name *iweb-link*. These nodes essentially link one holoweb to another and are used to form a name-space holarchy, where a holoweb name can be linked under/within[18] another holoweb name, recursively.

For example:

$\sim$[holoweb-name-A]/[holoweb-name-B]

The above address will resolve first to a holoweb A with the name 'holoweb-name-A', and then, based on the iweb-link nodes within this holoweb and its peer-accessible streams, to a secondary holoweb B with the name 'holoweb-name-B', and subsequently to the root-space of holoweb B. In this case there would be an iweb-link node in holoweb A, linking from A to B with the name 'holoweb-name-B'. In most cases the name of B will be equal to the name used in the link, but this is not enforced (the name used in the link is arbitrary).

As before in the first usage of the HNS described above, after any number of holoweb names separated by a backslash character, a type-name and node-name can be appended in the format [type-name].[node-name], to resolve to a specific node.

a) Name-space Settings: It is possible to moderate and constrain if/how a holoweb can link to another holoweb as above, using the membrane of a web-core, by setting the number of iweb-link nodes that can be created (and by which peers if desired) in the type-membrane. This is required to establish, for example, whether the name-space of a holoweb is determined by the author of the holoweb or its end-users.

b) Top-level Names: When an end-user wants to use a name-space as their top-level name-space, the name-space is 'coupled' to a Holoweb Browser. Multiple holowebs and/or holoweb streams may be coupled to a browser, resulting in an (inter-)subjective and custom name-space.

c) Link Relation: The relation of an inter-web link stores additional data about the link, such as which networks are distributing the linked web. This network data may include IP addresses and other network and device identifiers, enabling access to the linked holoweb and its streams.

With the inclusion of inter-web links containing network and device identifiers, HNS becomes a viable alternative to DNS, when appropriately implemented on sufficient networks.

2) Additional Uses: For referencing a specific node of a specific type, where the holoweb is unknown, the following address format may be used:

$\approx$[type-name].[node-name]

This will result in a basic search query, within the scope of locally accessible holowebs, and may resolve to one or more nodes.

For referencing a specific stream in a holoweb, the following address format may be used:

$\sim$[holoweb-name]/@[stream-name]

The stream-name provided must correspond to a stream containing a $\approx$stream node (see II-A6) with a matching name. If such a $\approx$stream node exists, the address will resolve to the $\approx$stream node itself, or the root-space of the stream, if a $\approx$space has been intrinsically linked from the $\approx$stream node. If desired, conventionally the stream-name may be set as a username of the author of the stream.

In some cases it may be desired to directly address a node without any probability of the address resolving to multiple nodes. In this case the following address format may be used, which always resolves to exactly one node in a stream (if the node exists and is accessible):

$\sim$[holoweb-name]/@[stream-name]/[type-name].[node-name]

In conclusion, the name system allows for any (named) node or holon on the Holoweb to be addressed with a human-readable string. Therefore, essentially any 'object' of perception, any 'thing' within consciousness/experience, can be addressed.

F. Conventions

A number of conventions are introduced to efficiently reference different components of the holoweb ontology and ecosystem.

a) Hol-'o'-web: In general, the use of the letter/symbol 'o', may be used as an abbreviation of 'holoweb' and/or 'holonic'. For example, a holoweb may be referred to as an *oWeb*: i.e., hol-"oh"-web, "oh"-web, oWeb. The symbol is also a circle, which connotes 'whole' (i.e., holon). Further examples include: oWeb, oApp, oDNA, oSpace, oStream, oName, oID, oCore, oNode, oHolon, etc.

[18]This *may* correspond to the idea of a web within a web, i.e., a sub-jurisdiction of a jurisdiction in certain use-cases.

b) Holonames: a holoweb name (see II-E) is case-insensitive (displayed in lower-case), alphanumeric with hyphens ('-' symbol), must be at least 1 character and must begin with an alphanumeric character.

c) Frequency: The tilde symbol $\sim$ and double-tilde (approximation) symbol $\approx$ are used to represent the idea of multi-dimensional *frequency* (waves), in relation to node/holon types and holowebs (which are defined in part by sets of node-holon-types, in holoweb DNA). A single tilde $\sim$ is used to prefix a holoweb name. A double-tilde $\approx$ is used to prefix a node/holon-type name.

d) General: a holoweb may be referred to as a web or oWeb, a holoweb application may be referred to as an app or oApp, a holoweb DNA may be referred to as a DNA or oDNA, etc. depending on context.

III. Implementation

At the time of writing, the Holoweb design and protocols have already been partially implemented in prototype software. The details of this are sufficient to require an additional paper, which will be published at a later date.

This work in progress can however be summarised now as follows:

a) Software Applications:

- **Holocore** a Holoweb 'core host', securely hosting identities and personal data, managing networks and browser sessions
- **Holoweb Browser** a mobile and desktop browser application for browsing the Holoweb, featuring a 2D and 3D graphical canvas

b) Network:

- **Stargate** simple native holoweb server network and protocol using a token-system
- **LAN** peer-to-peer connectivity directly over a Local Area Network

c) Protocols:

- **HWTP** a Holonic Web Transfer Protocol, for transferring serialised holons over a network (includes holon serialisation and deserialisation)

d) Holoweb System Webs:

- **Genesis Web** for creating holowebs, containing the meta-DNA of the holoweb
- **ID Web** for publishing profiles
- **Earth Web** for indexing holowebs
- **App Web** for publishing holonic applications
- **DNA Web** for publishing holoweb DNA

A. Genesis Web

The *genesis web* contains a 'living' specification of the Holoweb, that is the 'meta-DNA' or (meta-)ontology of the Holoweb, which functionally enables the creation of holowebs.

At the time of writing, the genesis web has been manually compiled using an ad-hoc compiler, developed for the purpose of creating the genesis web. The web currently has a total of 371 nodes, 317 percepts, and 170 blobs – including 72 types of node, and 55 instantiable node-types. This is however likely to change during the initial implementation phase in Q1 2021, and therefore will be published at a later date.

B. Network Integrations

In addition to the holoweb-native Stargate network, multiple existing networks will be integrated and/or connected to the Holoweb. These include (but are not limited to) Holochain [20], Scuttlebutt [25], IPFS [9], and Threefold [28].

We see these different networks as complimentary, each with their own unique advantages and disadvantages, which are aligned with different use-cases.

For example, Holochain appears to be particularly well suited for hosting large distributed/decentralised public holowebs, such as national or global webs, whereas Scuttlebutt is more aligned in localised, private social contexts.

IV. Conclusion

A universal/global perceptual-semantic graph with network-independent data integrity was introduced, providing an observer-dependent holonic map of *reality* in a single holon, resembling a mathmatical hyperset.

This distributed self-describing and self-defining data structure provides a foundation and framework for network-independent 'meta-decentralised'[19] applications (ddApps), where a higher level of decentralisation is achieved, than in currently established web3 methodologies.

Ultimately the Holoweb may act as an integrative component of the current web3 ecosystem, bringing together multiple technologies and networks, which are not presently interoperable or connected to each other.

The Holoweb protocol was defined in terms of 4 sub-protocols focused on identity, data, perception, and network, which at the time of writing are partially implemented in prototype software.

The Holoweb, as a web, is characterised by its unique ontology and perceptual-semantic data structure, network-independence, complete decoupling of data and interface (enabling perceptual sovereignty and enhanced sense-making), 3D/multi-dimensional holonic applications, inter-subjective Holonic Name System, hyper-connectivity[20] of data, and hyper-addressibility[21] of data.

The overall desired affect of the Holoweb is the *resiliant* facilitation of neuro-semantic coherence (enhanced communication), enhanced sense-making, and scalable decentralised collaboration[22], in arbitrary communities independent of language and culture, on all social holonomic levels, universally.

[19]Where decentralisation itself is decentralised.

[20]Hyper-, relative to the current web, which links document-objects to documents/files only.

[21]Hyper-, relative to the current web, where only websites and documents/files are directly addressable.

[22]Different forms of collaboration and co-creation are enabled through holonic applications, hosted on one or more network-independent holowebs.

References

[1] Peter Aczel. "Non-well-founded sets". In: (1988).

[2] Jon Barwise and Larry Moss. "Hypersets". In: *The Mathematical Intelligencer* 13.4 (1991), pp. 31–41.

[3] Jean-françois Groff Tim Berners-lee Robert Cailliau and Bernd Pollermann. "World-Wide Web: The Information Universe". In: *Electronic Networking: Research. Applications and Policy, 8(2), Westport k.* 1992. URL: https://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.324.3588.

[4] George Klir and Bo Yuan. *Fuzzy sets and fuzzy logic*. Vol. 4. Prentice hall New Jersey, 1995.

[5] Tim Berners-Lee, James Hendler, and Ora Lassila. "The semantic web". In: *Scientific american* 284.5 (2001), pp. 34–43.

[6] Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System.* https://bitcoin.org/bitcoin.pdf. 2008.

[7] Raphael Sofaer Dan Grippi Maxwell Salzberg and Ilya Zhitomirskiy. *Diaspora (social network).* https://en.wikipedia.org/wiki/Diaspora_(social_network). 2010.

[8] Chris Larcombe. *Waysphere - Holographic Mapping for Co-Creating New Worlds*. https://www.youtube.com/watch?v=A-xQKtFLmCU. 2012.

[9] Juan Benet. *IPFS - Content Addressed, Versioned, P2P File System.* https://github.com/ipfs/papers/raw/master/ipfs-cap2pfs/ipfs-p2p-file-system.pdf. 2014.

[10] The Matrix.org Foundation CIC. *Matrix - An open network for secure, decentralized communication.* https://matrix.org. 2014.

[11] Adam Apollo Harlan T Wood. *Core Network - The Last Social Network.* https://core.network. 2014.

[12] Chris Larcombe. *Noomap.* https://larcombe.io/noomap_paper.pdf. 2014.

[13] Vitalik Buterin. "A Next-Generation Smart Contract and Decentralized Application Platform". In: 2015. URL: http://people.cs.georgetown.edu/~clay/classes/fall2017/835/papers/Etherium.pdf.

[14] Tamas Kocsis. *Zeronet - Decentralized websites using Bitcoin crypto and the BitTorrent network.* https://github.com/HelloZeroNet/ZeroNet. 2015.

[15] Chris Larcombe. *Noomap: A Tool for Change.* https://www.youtube.com/watch?v=E3tHbf-QTyI. 2016.

[16] Andrei Vlad Sambra et al. *Solid: a platform for decentralized social applications based on linked data*. Tech. rep. Technical Report, MIT CSAIL & Qatar Computing Research Institute, 2016.

[17] block.one. *EOS.IO Technical White Paper v2.* https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md. 2018.

[18] B.D Wallace Drew O'Meara. *PLAN Systems.* https://plan-systems.org. 2018.

[19] David Ellams. *Our World - OASIS API.* https://ourworldthegame.com. 2018.

[20] Arthur Brock Eric Harris-Braun Nicolas Luck. *Holochain, scalable agent-centric distributed computing*. https://github.com/holochain/holochain-proto/blob/whitepaper/holochain.pdf. 2018.

[21] TRON Foundation. *Tron - Advanced Decentralized Blockchain Platform.* https://tron.network/static/doc/white_paper_v_2_0.pdf. 2018.

[22] *Steem - An incentivized, blockchain-based, public content platform.* https://steem.com/wp-content/uploads/2018/10/steem-whitepaper.pdf. 2018.

[23] Aaron Blankstein Muneeb Ali Jude Nelson and Ryan Shea Michael J. Freedman. *The Blockstack Decentralized Computing Network.* https://blockstack.org/whitepaper.pdf. 2019.

[24] Jared Rice Sr. *dWeb: The Protocol For Web 4.0.* https://distributedweb/whitepaper. 2019.

[25] Dominic Tarr et al. "Secure Scuttlebutt: An Identity-Centric Protocol for Subjective and Decentralized Applications". In: *Proceedings of the 6th ACM Conference on Information-Centric Networking*. ICN '19. Macao, China: Association for Computing Machinery, 2019, pp. 1–11. ISBN: 9781450369701. DOI: 10.1145/3357150.3357396. URL: https://doi.org/10.1145/3357150.3357396.

[26] Michael Sena et al. *Introduction to the Ceramic Protocol.* https://github.com/ceramicnetwork/ceramic/blob/master/overview.md. 2020.

[27] Decentralized Hive Ecosystem. *Hive: Fast. Scalable. Powerful. The Blockchain for Web 3.0.* https://hive.io/whitepaper.pdf. 2020.

[28] Threefold Foundation. *ThreeFold.* https://wiki2.threefold.io/threefold_grid_whitepaper.pdf. 2020.

[29] Chris Larcombe. *Holomap - Open-source Holonic Mapping*. https://larcombe.io/projects/holomap. 2020.

[30] Jun Li et al. "LBRY: A Blockchain-Based Decentralized Digital Content Marketplace". In: *2020 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS).* IEEE. 2020, pp. 42–51.

[31] Milo Trujillo et al. "What is BitChute? Characterizing the" Free Speech"Alternative to YouTube". In: *arXiv preprint arXiv:2004.01984* (2020).