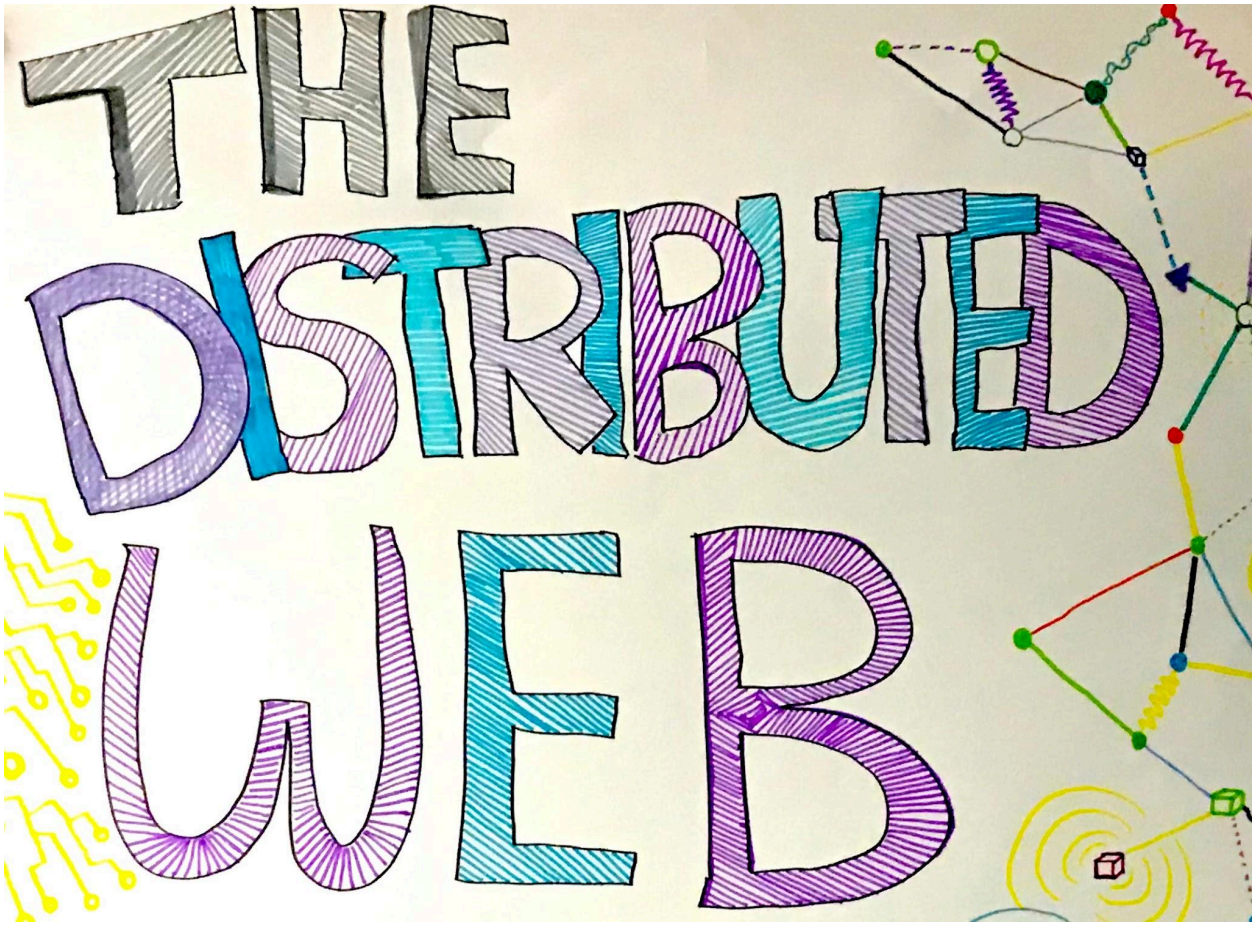


- Guidebook
- Github

Agenda

- Intro, Structure and Motivation for the Session (5 mins)
- The current structure of the Web has problems, let's identify them! Which problems can be traced back to centralization (censorship), and which cannot (cyber-bullying)? (15 mins)
- The Distributed Web (25 mins)
    - Identifying reasons the web is centralized, and listing corresponding solutions.
    - What are the other issues with the Centralized Web?
- What are we missing? (10 mins)
- The conversation doesn't stop here! (5 mins)
- (extra) Features IPFS uses to create the Distributed Web

The current model has problems, let's identify them!

Group exercise! Organize in groups of N people, introduce yourselves and list what problems the Web has today. Write down what you identify on sticky notes, and sort them into two piles, one set of problems which can be traced back to centralization, and one which cannot. It can be:

- *Situations*: "When I'm on a plane, I can no longer use X service"
- *Applications*: "Etherpad doesn't work when I'm offline"
- *Technology*: "Most of my bookmarks return 404 nowadays"
- *Issues*: censorship, surveillance, net neutrality

(TODO: in the end of the exercise, take pictures of the wall and post them here)
https://docs.google.com/document/d/1fAf_bghe0EMk8Cre892v22hc0MNCHAp3K697hUIdaL4/edit?ts=59f58f6e

http://bit.ly/2ydR048



🤗

Why do we have a Centralized Web?

Identifying cause, effect, and proposed solutions.

Our Hypothesis

- Over-centralization leads to unhealthy concentrations of power that restrict the adaptive and creative potential of humanity. A global communications network has the potential to let everyone participate in shaping the future -- but right now, many are shut out, in walled gardens, or stuck in spectator seats. Fixing this can begin with changing the structure of the communications network itself.
- The Web evolved centralized structures because existing technical architectures and protocols, funding mechanisms, and social structures made where we are today the default situation to end up in.
- De-centralizing the future Web will require changing the things that have fueled the growth of its centralized systems.
- Decentralization itself as a motivating principle (driven by politics or aesthetics), is not enough for widespread adoption. Protocols and services of the distributed web must have some new use case, or be fundamentally better in some way than what exists now.

This is not intended to be a comprehensive list, but we had fun brainstorming issues with the centralized web and how they could potentially be addressed. We hope you'll find this useful or interesting, and have more things to add!

Here's how the Digital Life Collective frames decentralization.

Color Scheme:

- Technical
- Political/Social
- Economic

Reason	Solution
DNS - Service Discovery Where is stuff? The web needs an "address book" to refer to locations and that address book today is Centralized.	***Content Addressing + Self-Certified Naming*** Address content by its fingerprint rather than its location so that anyone in the web can help you find and retrieve it.
IPv4, Hierarchical Addressing, Tree Routing Who are you talking to? Every machine	***Process Addressing by Cryptographic Identity*** Need a universal way to refer to nodes in the

requires a single address, and due to not having enough addresses for all the machines in the world, NAT were created and with them barriers of access.	network and authenticate their identity, without going through an intermediary
TCP/IP Stack is not available everywhere What language are you using? Due to the constraints of different platforms, there's no common transport that can be used by every single node in the network	***Modular and adaptable Network Stack (libp2p)*** Support for multiple transports and for translation points for when nodes don't support the same communication the same
Gatekeeping and Access Control Some information requires restrictions on who can access it, and accountability for changes that are made.	***Capability Systems*** Cryptography that gives access only to selected participants. (Verifiable claims.)
History Tracking Version control high sensitive information so that no malicious party can alter information without the respective governments of the data knowing that is happening.	***Cryptographic File Versioning + Graph Data Structures*** Having something like version control on Github. Underlying data structure is a tree of hashes, can use this many more ways
Monolithic products with good UX Centralized services (i.e. Slack, Dropbox, Paypal) are easier to use than decentralized alternatives, and have better UX (Note: we're using Google docs right now!). The funding and organizational structure of centralized orgs make this possible	***Interoperable open-source components*** Decentralized services can build and release modular, open source components that are easy to build on, and strive for better UX. Distributed teams tend to build modular systems, so we should use this to our advantage. (See the hi:project.)
Funding, Accounting & Billing Services require a way to track, bill, provide support and serve the users of their platform. The infrastructure to support this processes has been historically centralized out of convenience and ability to actually maintain it.	***Blockchain-based infrastructure*** Public distributed ledgers (i.e. blockchains) make new forms of funding and accounting possible
Some (many?) governments prefer to have only a few doors to knock on 15 years ago, there were hundreds of services, now the 'frightful five' account for the majority of our time online.	
The network effect Describes the situation in which the utility and value of a product or service increases with	**Open reflexive value** Develop highly valued concepts within an open network that cannot be undertaken

the number of others using it. This can work in non-commercial (eg. the Web) and commercial contexts (eg. Facebook, Google), locking in the participants irreversibly.	viably by a centralised commercial entity, pulling people away from centralized commercial services.

What are other issues with the Centralized Web?

Issue	Solution
Data Silos Information is centralizing in a few data monopolies, since organizations with better data are more able to profit, predict, and grow, leading to a winner-takes-all cycle. User data is monetized for advertising, and used to train increasingly advanced AI which could eventually pose an existential threat. Meanwhile, scientific data and research that could advance collective knowledge is paywalled or difficult to access	***Store Data Together*** Organizations can share infrastructure and provide that infrastructure as a service to users (e.g Public Library) privately and anonymously.
Bandwidth is not increasing rapidly The Internet is getting slower as the Internet pipes are not getting fast rapidly enough for the amount of data we have passing through them.	***Leverage Native Network Caching*** Use content addressing to fetch copies of data from the closest location.
Throttling flows: Censorship Companies and governments can block content or applications. This can be done for many reasons, some legitimate, but structurally this is a symptom and exacerbation of concentration of power	***Censorship-resistant networks*** Encryption or p2p propagation of content makes censorship impossible, “locking the web open”. Additionally, mechanisms must be built for communities to self-moderate (banlists, web of trust)
Data asymmetry: Surveillance Our data goes into corporate and government black boxes that know a lot about us, and we know very little about them. The security model of the Web doesn’t default to data encryption at rest.	***Regulation…? Encryption*** Regulation to protect privacy might help somewhat but long-term the incentives for central platforms is to collect and store as much information as possible. Encrypt everything, find ways to be profitable other than monetizing user data
It doesn’t work offline or disconnected Most of the applications we use to connect with our loved ones, run our business and to acquire new knowledge, break if our connection to the backbone is not present.	**Content Addressing, Process Addressing and Self-Certified Naming** Infuse the Web with p2p primitives that make applications be resilient against netsplits.
Links break all the time (Permanence)	**Archive and content address data**

The content that is linked to on the web gradually disappears over time. This means that the collective knowledge that is put online is gradually lost, and history disappears.[1]	Organizations such as the Internet Archive have been working for decades on archiving the Web and ensuring that humanities information is not lost. The next step is making the archived information easily accessible by the interface that users are accustomed, the URL/Search bar in the Browser.
The Web is being kicked out of IoT New proprietary protocols are being designed and deployed to IoT keeping the Web and its openness out of the equation	Enable the Web to support more runtimes through more versatile network stacks (libp2p)
The Web is being kicked out of mobile Zero-rating plans mean that many people today think Facebook == Internet.	Teach people how to make abstractions
Net neutrality - companies want to be monopolies so they can set prices Barriers to entry create monopolies, and then monopolies create more barriers to entry, or acquire startups and find ways to kill competition.	***Regulation, or build alternatives that don't get acquired*** Attempt regulation in the public interest, or make alternatives that are better, cheaper, and can't be acquired

Looking towards the future:

Do cycles of centralization/decentralization mean that we will get new concentrations of power if we manage to pull apart the lock-in of existing protocols and monopolies?

[1]

How do we design protocols and networks that can be seamlessly upgraded after their adoption and widespread deployment?

How will the distributed web self-govern?

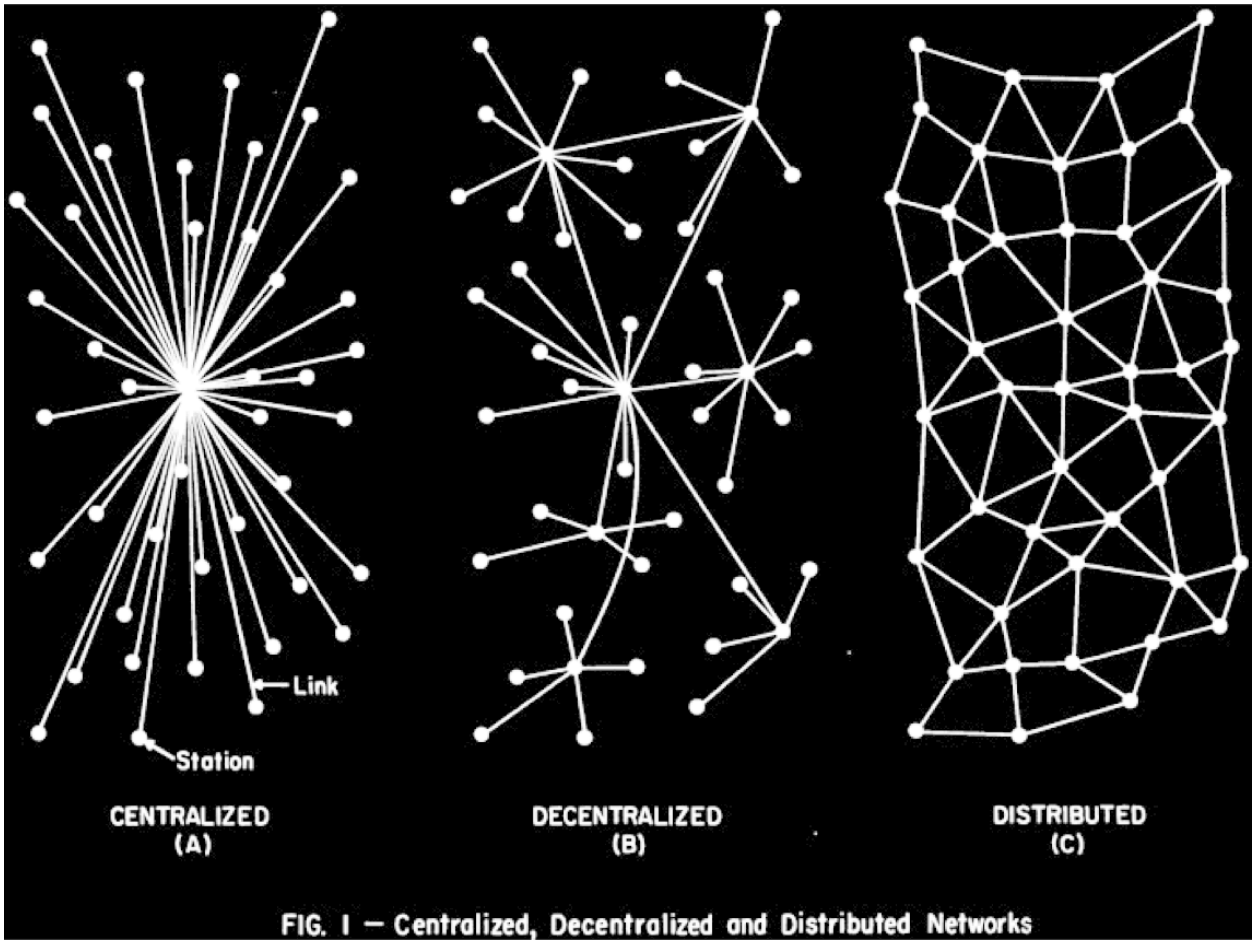
[Please consider adding your decentralisation project to this ecosystem map.]

Questions? Comments? Anything to add?
Please add to the discussion here, or on the Mozfest issue ticket for this session
https://github.com/MozillaFoundation/mozfest-program-2017/issues/131

(extra) Features of the Distributed Web

There is a plethora of materials available on the Web that compare the Centralized vs the Distributed Web and its features. However, it is often not clear exactly what are the core features that the Distributed Web offers from a technical perspective and how they work. This section presents to the reader a first attempt to categorize these with visual instruments that can be reused, remixed and shared with others.

The best way to start visualizing the key features of the Distributed Web is by looking at the network categorization by Paul Baran, one of the original Arpanet Engineers. Paul Baran puts Centralized and Distributed at the two ends of a spectrum, leaving with a hybrid known as Decentralized or Federated network where peers form clusters in a certain hierarchical shape.

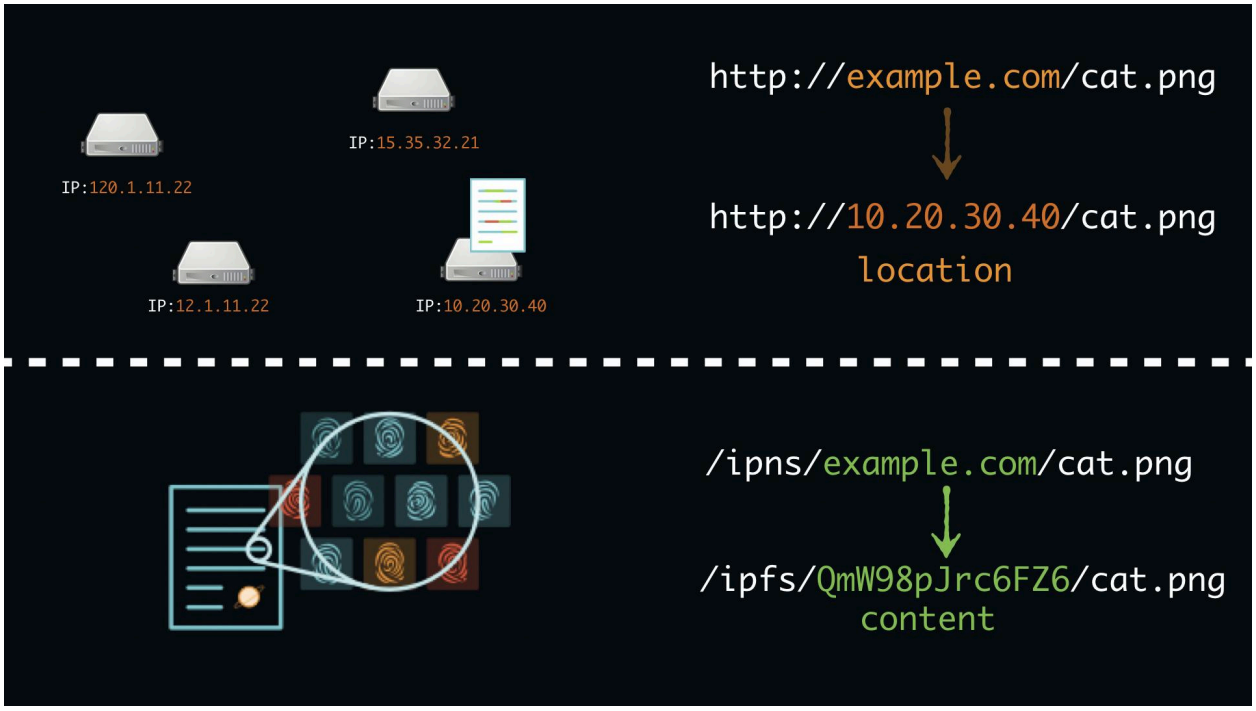


What is implicit in this image is that in order for a Distributed Network to work, every peer must be capable of speaking the same protocol so that the mesh is resilient to network splits.

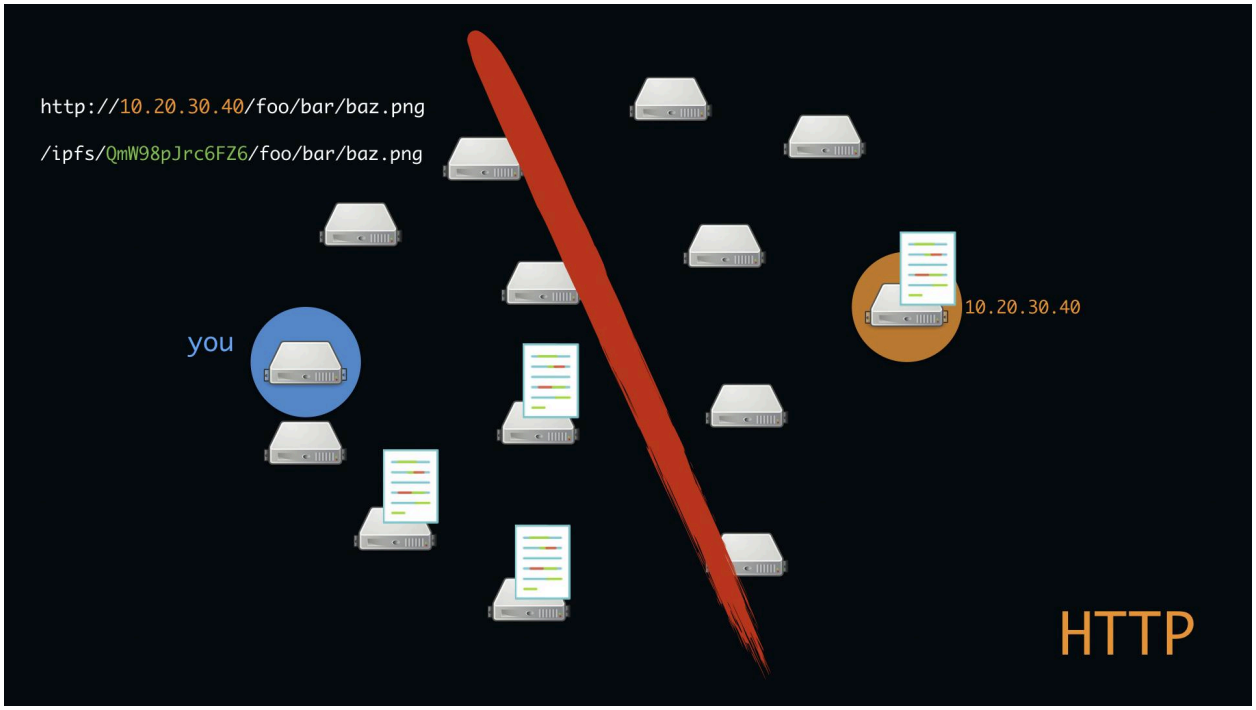
The following list of features are additions to this distributed model that come after this network categorization, giving the Distributed Web more resilience to network splits, make it secure and faster.

Content Addressing

Changing the Web to work over Content Addresses rather than Location Addresses means that instead of using the old DNS translation of name -> machine, we actually focus on identifying the content we are looking for by its Cryptographic Fingerprint, an unique ID that cannot be forged and that helps you and everyone in the network find the real content that you are looking for.



This translates to a more resilient and Open Web. With Location Addressing, a netsplit (a separation of the network either by a failure or by voluntary action by an institution), would break your ability to resolve and fetch content that you knew before if you happen to end up on the side of the netsplit where the name resolver (DNS) does not exist.

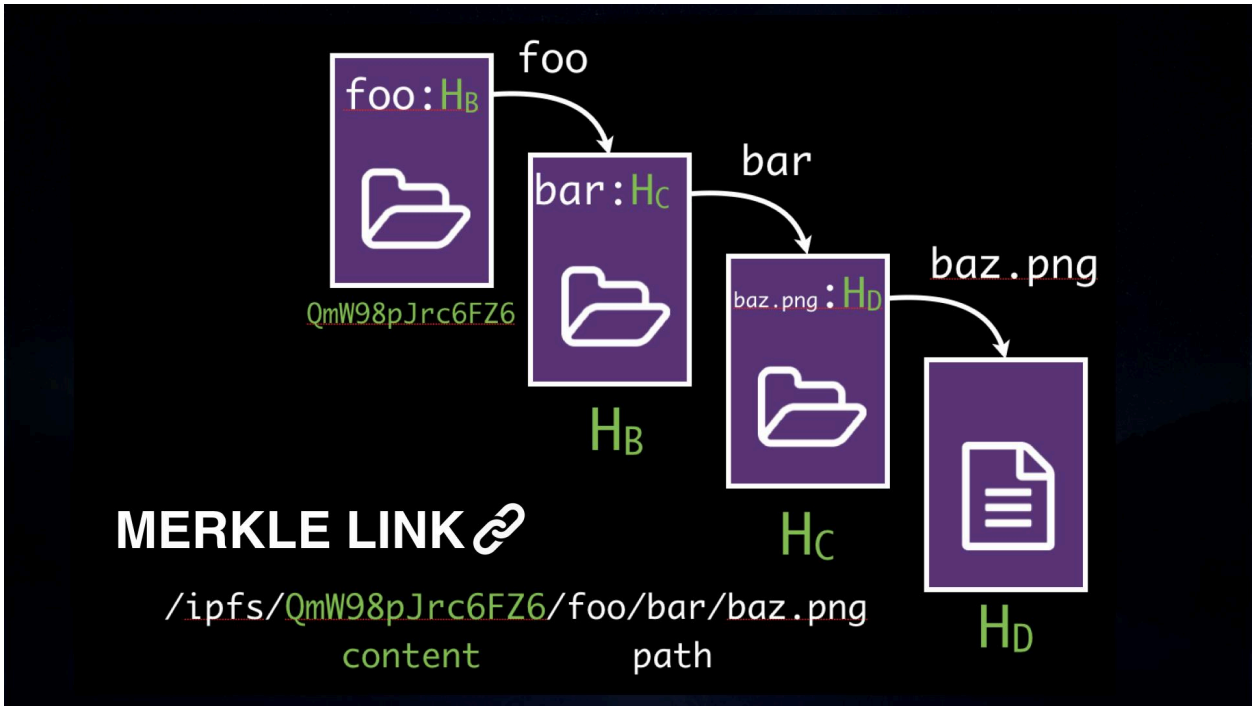


With the Distributed Web and its Content Addressed links, any node can be a provider of the data you are looking for and more, you have the ability to validate that the data you received is indeed what you were looking for, something that is impossible in the Centralized Web.

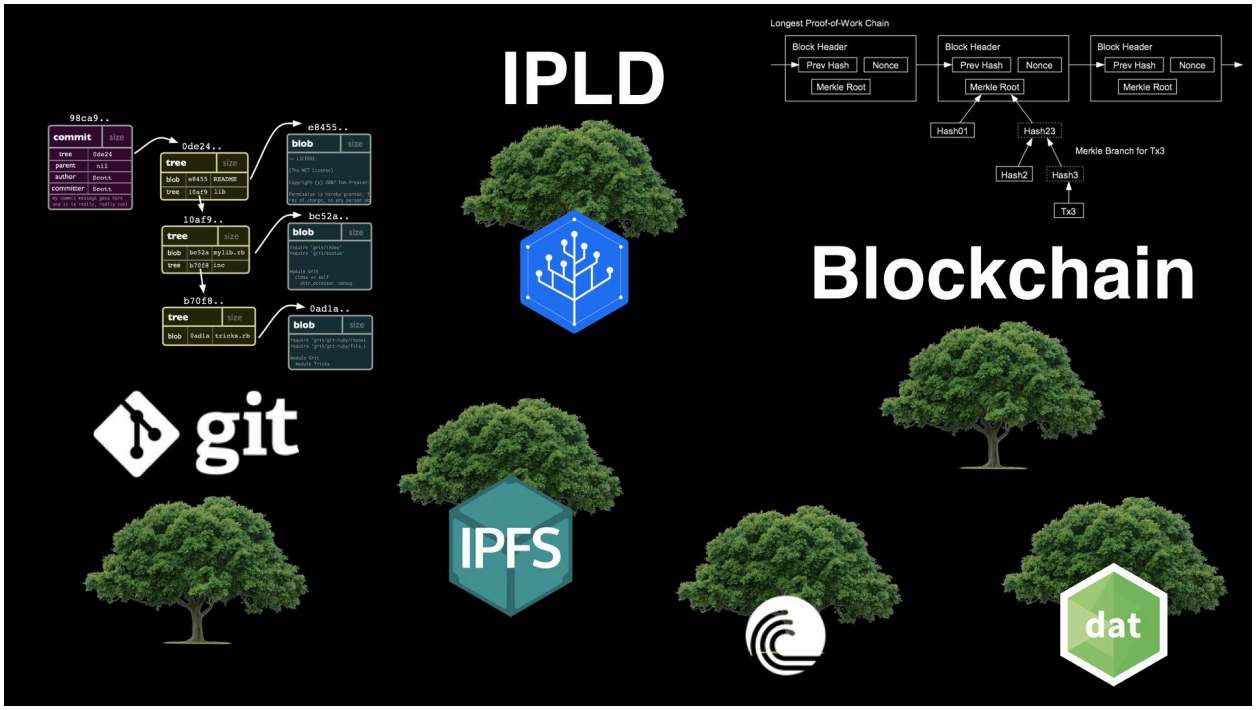


Cryptographic File Versioning

Cryptographic File Version is achieved for free by using Content Addresses. It gives the users a way to identify multiple versions of a file and rollback to a specific version.



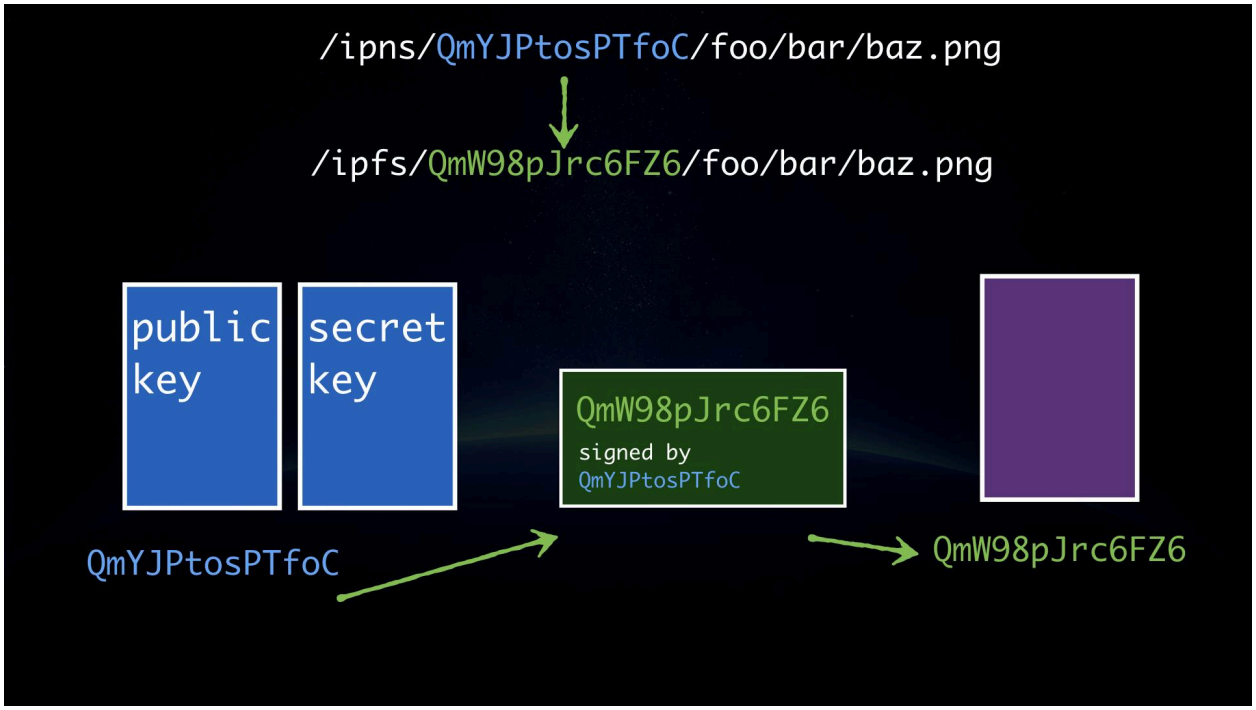
The Distributed Web is not the first system to benefit from this type of linking, others systems such as Git, Bittorrent and even Bitcoin have been using Content Addresses (aka Merkle Links) since its creation.



Self-Certified Naming

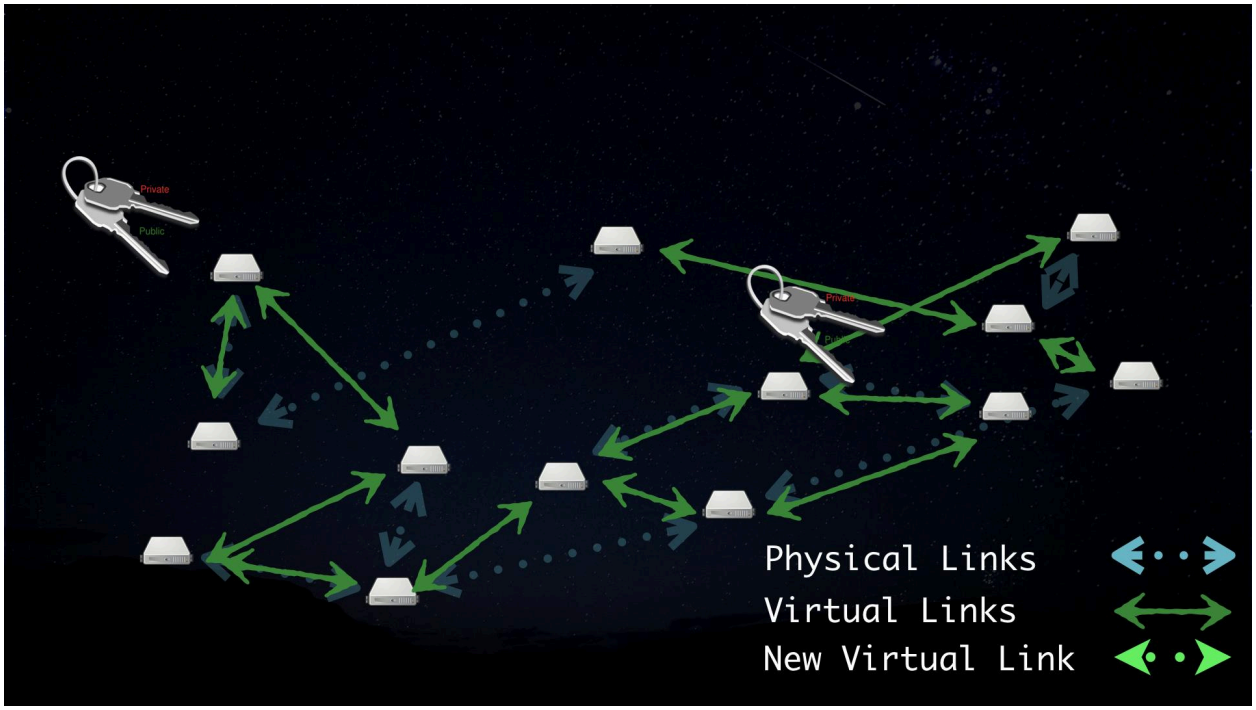
Self-Certified Naming shares a lot of the properties with the names we are familiar with (i.e DNS). The key difference is that they can be verified by the receiver by checking the cryptographic signature of the record.

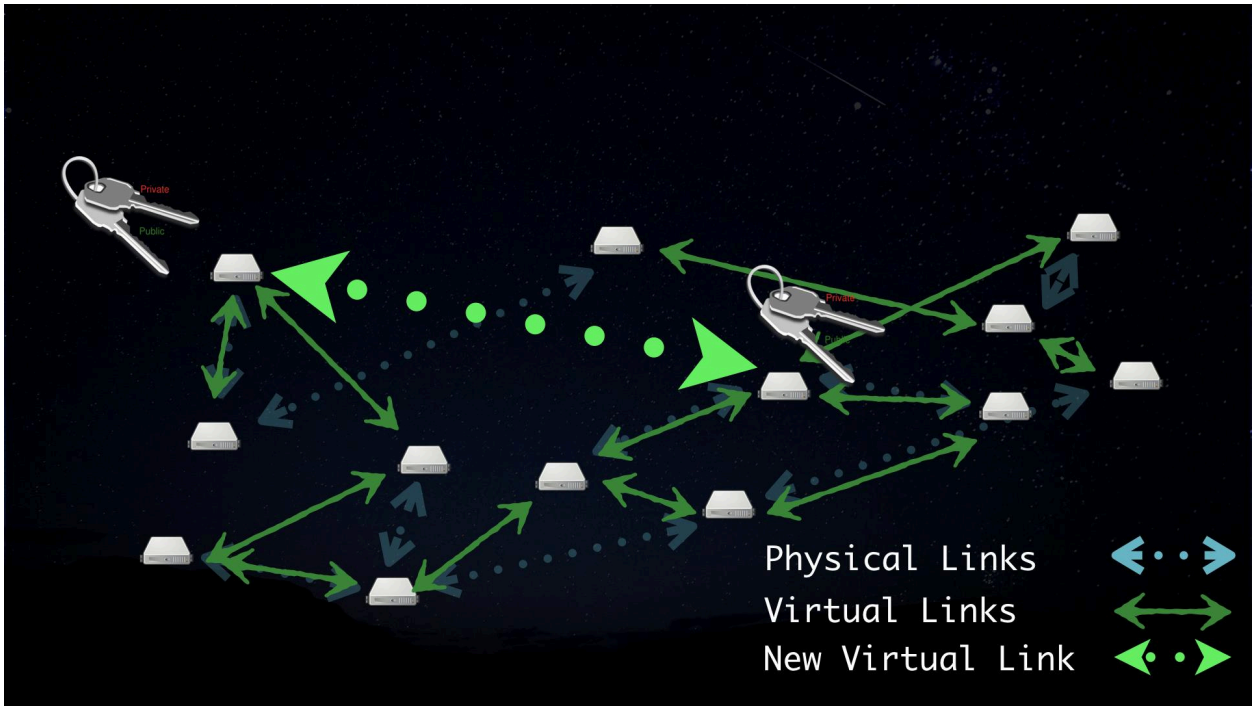
This feature means that any node in the network can generate their own names and broadcast updates, knowing that no one will be able to forge their signature and give wrong information to the users.



Process Addressing by Cryptographic Identity

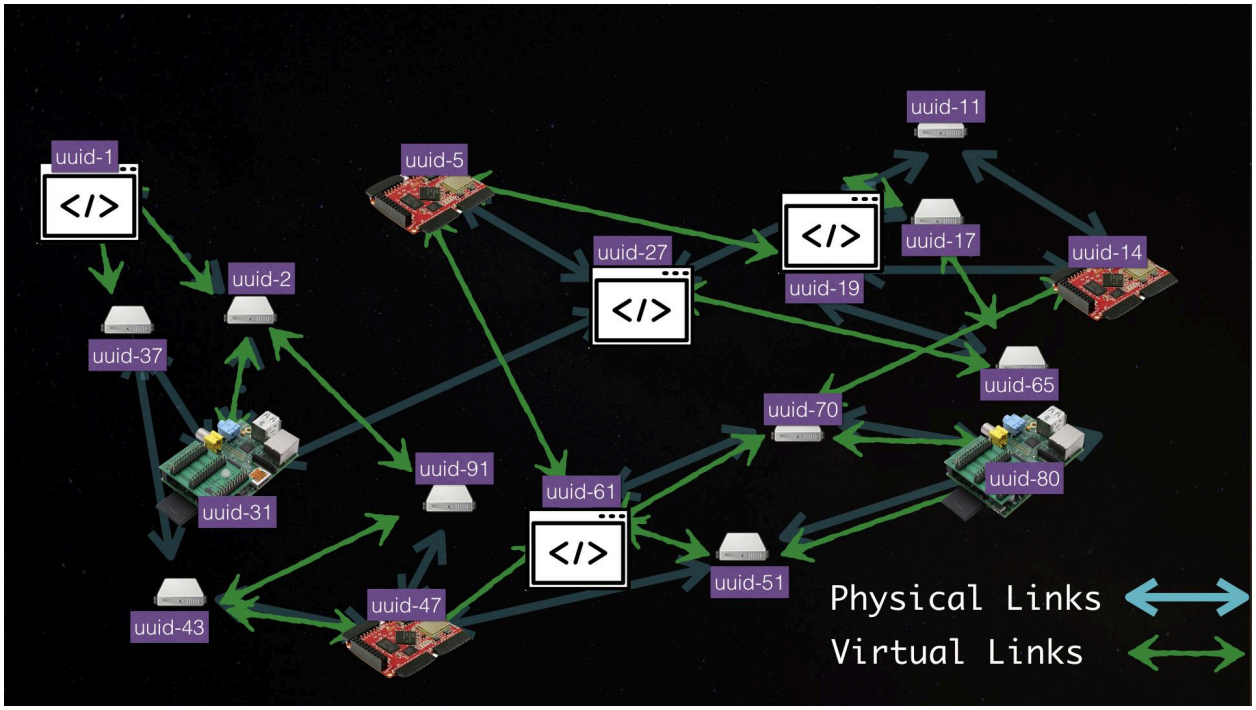
In networks, processes have to be addressed by a routable ID so other processes to contact them. In the Distributed Web, an upgrade was made and processes became addressed by a Cryptographic Identity, their Public Key. Given the structure of Asymmetric Cryptographic, also known as Modern Crypto, used to generate the Private and Public Key pair, we can discover the process using a wide variety of means and also authenticate it once we manage to establish a connection. This is a primitive that was simply not present on the Centralized Web.



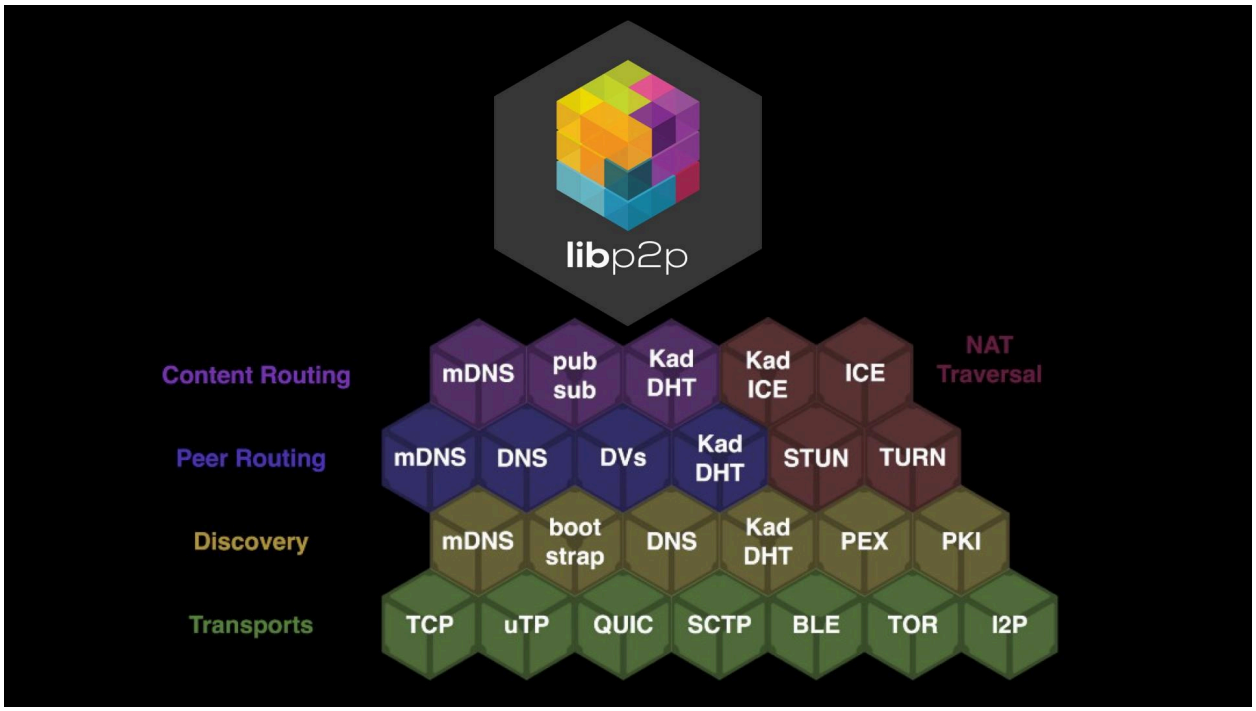


Modular and Adaptable Network Stack (libp2p)

An ever-evolving, innovative and open web is built by an heterogeneous network of with each device having very different capabilities, both hardware and software. It is already happening today and we see a variety of physical and virtual links connecting our gateways and service providers.



In order to support this ever-evolving mesh, we need a versatile networking stack that is future-proof, capable of adapting to change. That stack exists today and it was made for the distributed Web, libp2p.



Native to the Web (it works on the browser!)

For a very long time, P2P were excluded from the browser runtime due to its limitations (transports, lack of crypto primitives and so on). Today, that is not the reality anymore as there is one implementation of the IPFS protocol fully implemented in JavaScript. This is is possible thanks to the modular design of IPFS and libp2p that enables it to work on many different runtimes.

We identified that having a native P2P protocol was something critical for the success of the Distributed Web.

