Ecosystem Review

January 2021

Table of Contents

Overview

This overview of the decentralized social ecosystem is structured by protocols, applications, and topics. The protocols and applications sections contain summaries of existing projects.

Protocols

- ActivityPub
- GUN
- Hypercore Protocol (Dat)
- IPFS
- Matrix
- Peergos
- Solid
- XMPP

Applications

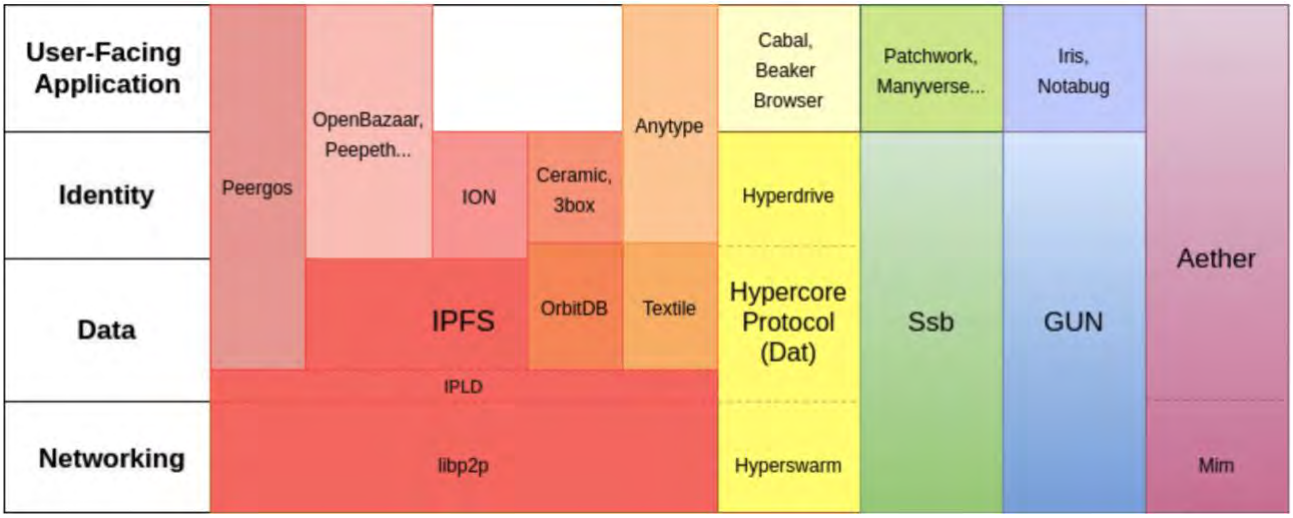
- Aether
- Blockchain social networks
- Diaspora
- Mastodon
- SSB social networks

The topics section compares how decentralized protocols handle key topics and includes relevant projects not covered elsewhere.

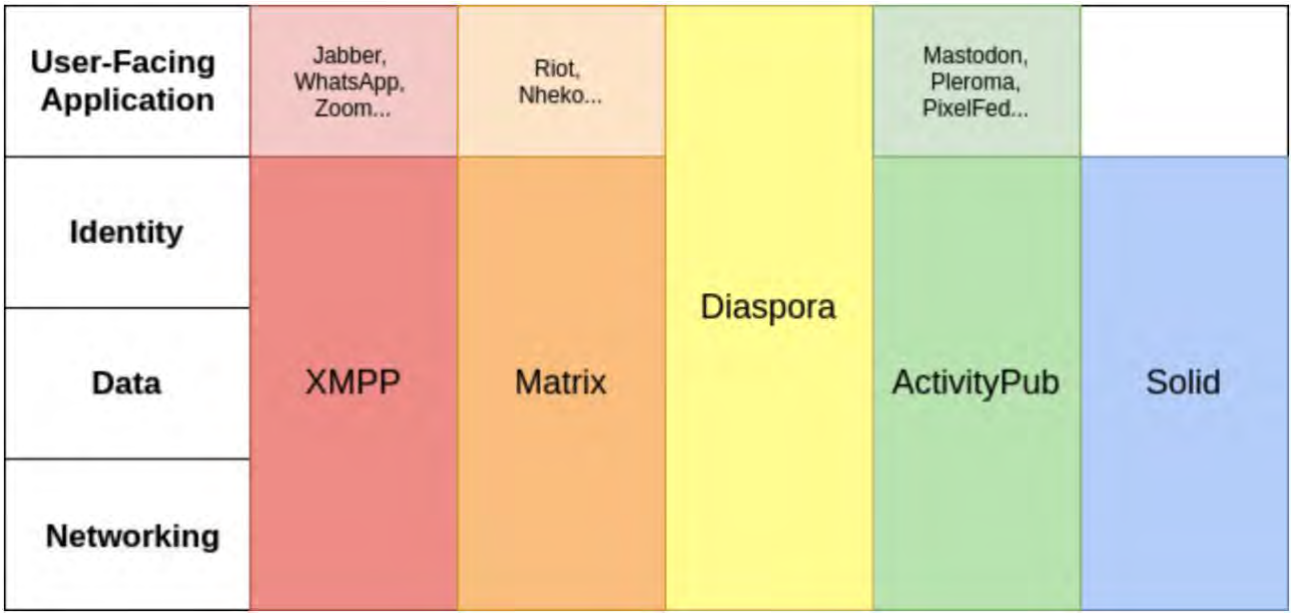
Topics

- **Data**
  - Data models
  - Mutability
- **Discovery**
  - Curation
  - Search
- **Governance**
  - Overview of existing models
- **Identity**
  - Decentralized identity
  - In federated applications
  - In P2P applications
  - Blockchain identity systems
  - DIDs
  - Key management
  - Reputation and trust
  - Failure modes
  - Sybils and spam
  - Impersonation
- **Moderation**
  - Community-based
  - User-driven
  - Experimental
- **Monetization & Business Models**
  - Application level
  - Provider level
  - Protocol level
- **Network structure**
  - Federated networks
  - Passing messages between systems
  - Replicating data between systems
  - P2P networks
  - Hybrid
- **Privacy**
  - User metadata
  - Private accounts
  - Direct messages

In the P2P ecosystem, there is sometimes not a clear distinction between protocols and applications, making it unclear what a project encompasses. This diagram attempts to clarify which layer of the stack these P2P projects operate at.



The distinction between protocols and applications is clearer in the federated social ecosystem, where applications follow a familiar client-server model. Diaspora is both the name of the social application as well as the protocol, and Solid does not yet have a social network application.



Contributing Authors

Written by Jay Graber (@arcalinea)

Contributions from:

- Andre Staltz (@andrestaltz)
- Burak Nehbit (@nehbit)
- Christopher Lemmer Webber (@dustyweb)
- Danny Zuckerman (@dazuck)
- Eugen Rochko (@Gargron@mastodon.social)
- Golda Velez (@gvelez17)
- Ian Preston (@Ianopolous)
- Jeromy Johnson (@whyrusleeping)
- Molly Mackinlay (@momack28)
- Matthew Hodgson (@ara4n)
- Mark Nadal (@marknadal)
- Paul Frazee (@pfrazee)
- Rahul Kothari (@BlockchainRahul)
- Sarven Capadisli (@csarven)

Further Reading

More extensive lists of decentralized web projects:

- Redecentralize
- Decentralized Web projects by gdamdam
- A list of P2P resources by kgryte

Protocols

ActivityPub

ActivityPub is a federated protocol that defines a set of interoperable social network interactions through specific APIs. Any server that implements this protocol can communicate with the rest of the network. It reached W3C recommendation status in 2018. It is one of several related specs produced by the Social Web Working Group.

ActivityPub consists of two layers: A server-to-server federation protocol and a client-to-server protocol. In order to federate with the ActivityPub ecosystem, a service only has to implement the server-to-server protocol.

Identity

Users in ActivityPub are conceptualized as actor objects. Actor to actor communication bears a resemblance to email. To be spec compliant, each actor *must* have an "inbox" and an "outbox" endpoint, as URLs which are accessible on the server. They also *should* have "following" and "followers". They *may* have "liked" collections and many other predefined possibilities.

Although not part of the ActivityPub spec, in practice WebFinger is used to discover actor profiles.

Server-to-server federation is authenticated using HTTP Signatures. The server creates a public and private keypair for each actor and a publicly accessible JSON-LD document retrievable over HTTP which contains its public key. Each message the server sends on behalf of an actor is signed by its key. When a remote server receives a POST to its inbox, it verifies the signature on the HTTP request. To verify object integrity, linked data signatures are used to sign the object with the public key of the actor who authored it.

A paper from the 2017 Rebooting the Web of Trust conference describes how distributed, cryptographic identities could be added to ActivityPub.

Networking

ActivityPub is a federated server-to-server protocol that passes messages between systems.

ActivityPub servers do not proactively network with each other, so they are unaware of each other's presence until a user finds and follows someone on another server. Servers maintain a list of remote accounts its users follow and subscribe to their posts.

ActivityPub messages are not limited to HTTP only. This allows it to potentially be extended in more P2P directions.

Data

ActivityPub messages are objects wrapped in an "activity", indicating what it is. There is an Activity Vocabulary that defines Activity, Object, and Actor types common to social web applications.

ActivityPub is not opinionated about how messages are persisted on the server as long as each server follows the protocol message requirements. Server implementations may cache frequent requests, such as follower actor objects, public keys of other servers, and images and attachments on posts.

Moderation & Reputation

Moderation is primarily handled by server implementations. Server admins can block individuals or entire instances. Banning instances can lead to the isolation of an ActivityPub server instance if it is banned by many others, limiting its users to communication with each other.

ActivityPub defines a "block" activity to help users control their experience.

ActivityPub adoption has reached a threshold where spam and harassment have become ongoing problems that protocol developers currently seek to address. Keeping Unwanted Messages off the Fediverse contains

a list of suggested solutions. OcapPub, a proposed object-capability based upgrade of ActivityPub, is a direction being pursued by one of the ActivityPub authors.

Social & Discovery

Messages are addressed to a user at their home server or published to a public inbox. Normal DNS and IP address routing are used to find the server addressed.

If posts are limited in visibility (followers only, direct message), they will be delivered to a user's inbox, such as https://example.com/users/alice. The "outbox" is a URL where an actor's recent activities can be retrieved from.

Servers also may accept delivery of messages addressed as “public” to a shared inbox available to all on the server, but are not required to. Social network implementations with public feeds may publish posts to the public inbox, such as https://example.com/inbox.

"Like"s and "Follow"s may be used by servers to determine which public messages to accept/retrieve.

There is no global search capability, as each server monitors a different set of messages.

Privacy & Access Control

Mastodon is currently adding E2E encryption to ActivityPub. Previously, messages were unencrypted on the server.

Interoperability

Any service that implements the ActivityPub server-to-server protocol can interoperate with the ecosystem. A service like Twitter would need to add WebFinger and JSON-LD representations of users and tweets.

The client-to-server protocol defines a standard way for user client software to connect to ActivityPub servers. In practice, it is rarely used. None of the major Fediverse services implement it. The vision of how it would work if it were widely used is that a user application could mix and match different servers like Mastodon, Pleroma, PixelFed, and any new service that implemented it.

Bridgy Fed is a project to connect IndieWeb sites with the ActivityPub and OStatus federated networks.

Diaspora, another federated social network, chose not to adopt ActivityPub.

Scalability

The ActivityPub ecosystem scales up by adding more server capacity to the network. This study on the Mastodon ecosystem analyzes the emergence of points of centralization as the network scales up.

Metrics

the-federation.info maintains statistics of the known OStatus/ActivityPub Fediverse.

Implementations & Applications

W3C Implementation Report

Watchlist for ActivityPub Apps

Notable applications:

- Mastodon (the largest federated network built on ActivityPub) has 2699 nodes and 2.6M users as of 5/2020.

- Pleroma is another federated social network. According to stats at the-federation.info, Pleroma has 620 nodes with 35K users as of 5/2020.
- Pixelfed is an ActivityPub based image-sharing platform.
- Friendica is a decentralized social network with support for ActivityPub, as well as the OStatus and Diaspora protocols.
- PeerTube is a free and decentralized video platform.
- Plume is a federated blogging application.

Related

ActivityPub inherits from a few other protocols that will not be covered in full, but are briefly summarized here.

ActivityPub was based on pump.io and ActivityStreams. Conceptually, these were preceded by OStatus. Pump.io is an activity streams social networking server. OStatus is an open standard for federated microblogging and describes how a suite of protocols can be used together.

The IndieWeb protocols and community are also related to ActivityPub through a shared vision of social federation developed around the same time period. The IndieWeb was inspired by the Federated Social Web Summit in 2010 and formed around the idea of interconnecting individual websites rather than federating social platforms. A co-founder described the vision as, "someone should be able to take their web site and be able to use their web site to participate in the same distributed social network — federated social network."

Links

- W3C ActivityPub Spec
- SocialHub, ActivityPub discussion forum
- Notes from an ActivityPub implementor
- Reading ActivityPub

GUN

GUN is a decentralized graph database with a conflict resolution algorithm (CRDT) and synchronization protocol. It includes a library of tools for merging conflicting data and handling routing, security, and storage.

In GUN's graph store, entries are JavaScript objects under UUID keys. Objects can be data of any type, including files, JSON, or other documents. Data is stored in the browser by default, with backup "superpeers" to ensure persistence. Peers connect to other peers and choose what data to synchronize and persist.

Identity

GUN's user system allows the creation of a human-readable username and password. Usernames are global but not unique. As with many other decentralized systems, there is no password recovery mechanism.

Multi-device login is handled by encrypting a user's cryptographic keypair, which is stored in the GUN graph. Keypairs are not derived from the password. Instead, a PBKDF2 proof is derived from the password and AES keys are derived from that to encrypt the keypair. GUN treats this method as "secure enough" for applications in which private keys do not control financial information.

Authentication is performed by doing a GUN query for the account username, subscribing to it, and then attempting to brute force decrypt the keys of all accounts that match the username until a match is found. Once an account has been loaded once, it's cached on device, loading from local storage or the local hard drive.

GUN's SEA (Security, Encryption, Authorization) module provides the capability to directly create a public/private keypair for a user, without a username and account.

Network

GUN uses a gossip-based protocol along with a topic-based PubSub protocol to sync data between peers. GUN peers fall back to the gossip protocol when the more optimized PubSub routing protocol fails. Messages can be routed across different transport layers (WebSockets, WebRTC, multicast UDP, etc.).

Peers subscribe to graphs relevant to their application's logic, although the global GUN graph is accessible to all peers.

Planned future network upgrades include the addition of a DHT. A tokenized incentivized mesh proposal is also on the roadmap.

Data

Peers subscribe to the data they need and the network retrieves it from any peer (including browsers, where GUN stores data in local storage). Running always-online peers, a "superpeer", is recommended for most applications to ensure availability of data when most browser-based peers may be offline. A superpeer is an IP addressable machine running node.js that persists data to disk.

There is a public space and a subset of that is the user space. In the public space are all graphs without a public key as their ID. In the user space, graphs are signed with the user's keys and their IDs must include the user's public key.

GUN uses a CRDT (Conflict-free Replicated Data Type) to merge data. Conflicts are handled by a conflict resolution algorithm that uses lexical sort. GUN is strongly eventually consistent, meaning that peers will eventually converge upon the last updated value when nodes that are offline eventually receive updates.

GUN focuses on mutability by not using an append-only log (which implements updates, insertions, and deletion as a layer on top of the immutable log). Deletion in GUN works by overwriting bytes with null, or by de-referencing portions of a graph. A content-addressed graph space is used to implement immutable, append-only data.

Privacy & Access Control

Access control is built into the user system and can be combined with SEA, GUN's encryption utilities, for more advanced use cases.

Cryptographic keypairs are assigned to roles, groups, or data points. This information is either used to derive a shared ECDH secret to decrypt (read), or to load collaborative multi-writer edits (signed).

Iris-lib is a decentralized social networking library built on GUN. It provides an API for end-to-end encrypted chat channels and private contact list management.

Interoperability

Plugins, such as backup storage on centralized databases or file systems, can be used to extend GUN.

Scalability

Test relays (superpeers) on GUN can handle about 10k simultaneous connections:
http://guntest.herokuapp.com/stats.html

Metrics

- 10M ~ 30M monthly downloads

Monetization

The GUN protocol is developed by a VC-funded company, which funds the development of Iris as well. The business model is based on consulting and integrations. Future business models include a proposed paid service through a blockchain-based tokenized bandwidth incentive network.

Implementations

GUN is used for P2P chat/social apps, encrypted video conferencing, real-time GPS tracking, and AR/VR multiplayer games, among other applications.

- Internet Archive uses GUN for their dWeb library metadata.
- Hacker Noon integrated GUN for annotations.
- Meething is a Mozilla backed secure and decentralized video conferencing service that uses GUN.
- Party and Maskbook, encrypted browser extensions, use GUN.
- Notabug, a decentralized Reddit clone, uses GUN.
- Unstoppable Domains and Dtube use GUN for messaging.
- Iris is a web of trust (WoT) based social network built on GUN.

Iris

Iris-lib is a library built on GUN that allows the integration of decentralized social networking features into applications. An experimental social network application, Iris, was built to demonstrate its features: public messaging, private chats, WoT, and contact management. Iris-lib uses GUN for networking and data storage. The team is funded by GUN and also accepts donations.

Iris uses WoT attestations to link human readable names to keypair and other identity attributes. Users only see messages in their WoT, from users who have been upvoted by someone in a chain from someone they upvoted. Downvotes are also possible. Reputation is not represented by a static score, but by how a user's personal WoT regards them. A percentage threshold of confidence in a person's identity is calculated by the number of attestations relative to the size of the network.

For interoperability, Iris allows importing content from other sources. "Message author and signer can be different entities, and only the signer needs to be on Iris. For example, a crawler can import and sign other people's messages from Twitter. Only the users who trust the crawler will see the messages."

Links

- Site
- 3Box comparison of P2P DBs: GUN, OrbitDB, Scuttlebutt

Hypercore Protocol

Hypercore Protocol (abbreviated Hyper), formerly known as Dat, is a distributed append-only log.

Dat was created in 2013 as a protocol designed to help data analysts collaborate on changes to data sets, but has since expanded to other use cases. Dat was renamed Hypercore Protocol in 2020 to reflect the extent of the changes. The older Dat protocol was "a P2P hypermedia protocol that provides public-key-addressed file archives which can be synced securely and browsed on-demand." Hyper places the core of the protocol at a lower level of abstraction. Hyperdrive, a public-key-addressed file drive, is implemented at a layer above. The focus of Hyper is the P2P distribution of an append-only log and its main purpose is to be a building block for other applications. The switch from Dat to Hyper was motivated by performance and scaling improvements, and the addition of mounts for nested hyperdrives. In addition, a DHT was added for peer lookup.

Identity

In Hyper, nodes are identified using a connection ID which helps avoid duplicate connections. This ID is largely abstracted from the application layer.

Users publish their information in a data structure identified by a public key. Data in the structure is signed using the matching private key. There is no means for revoking lost keys, as in PGP, and no backing central identity provider that can re-issue a private key, but the community is actively working on solutions which include DNS and identity providers.

In the applications space, Hyperdrives are often used to identify users, behaving as "profiles" (see Beaker Documentation).

In a recent roadmap, the core team has identified plans for authenticating messaging channels by proving ownership of hypercore data structures.

Networking

Hypercore data structures are identified by a public key. They may also be identified by URLs which use the hyper:// scheme and the base64-encoded public key as the domain. A hyper:// URL may reference any kind of Hyper-based data structure, including Hyperdrives.

Hyper uses the Hyperswarm networking module. Hyperswarm combines a Kademlia-based DHT for global discovery with MDNS to discover peers on local networks. Users join the swarm for a “topic” and query periodically for other peers who are in the topic. The topic is comprised of the hash of the public key which identifies the data being shared. When ready to connect, Hyperswarm helps create a socket between them using either UTP or TCP. A set of bootstrap servers is used to help connect a new node to the network.

The Hyperswarm DHT includes a hole-punching protocol to help nodes establish connections through NATs and firewalls.

Hyperswarm is also used to bootstrap messaging channels between nodes in order to host services and communicate ephemeral application state. An API called Peersockets piggybacks on existing Hypercore replication channels to send messages. The core team has outlined plans in the roadmap to add connection-topic swarming which should be implemented by mid-summer 2020.

Data

Hyper focuses on mutable data by organizing content under keys. Hyper's primary data structure is Hypercore, a signed append-only log. In contrast to SSB, which uses a linked-list append-only log, Hypercore uses a Merkle-tree-based append-only log where each entry generates a new leaf and new root. A cryptographic keypair is used to sign the root of the Merkle tree as data is appended to it. Once published, data in the log is immutable.

Hyperdrive is a filesystem data structure implemented on top of two hypercores. The first hypercore (the "metadata core") records file metadata in the form of POSIX-like "stat" objects. It uses a hash trie to provide efficient lookup of entries in a folder. The second hypercore (the "content core") records file content as a sequence of blobs. The metadata core references the entries in the content core. Unlike IPFS, entries in a Hyperdrive cannot be shared individually; they are referenced as paths beneath the Hyperdrive's public key.

Files stored in a Hyperdrive can easily change at any time. Only the original creator of a Hypercore data structure can append new data, making it a single-writer data structure. Multi-writer data structures can be built by using multiple hypercores that reference each other's elements.

An improvement added to Hyperdrive that was not present in Dat is the ability to mount hypercore-based structures like a folder of files, creating composed filesystems from multiple different authors.

Applications approach data schemas using a loose consensus and will often publish JSON, Markdown, and other forms of media. Paths within Hyperdrives will sometimes be considered significant; for instance, social media feeds leverage the /microblog/ folder to identify user posts.

A node must be actively sharing a topic for it to be available. To ensure persistence, nodes can host their own data on a server that stays online or pay a hosting service.

Moderation & Reputation

Hyper is a layer below moderation and reputation. There is no concept of "leeches" to evaluate whether nodes are being good seeders of content.

Social & Discovery

Nodes create "discovery keys" for topics by hashing the public key of the hypercore. Data they want to associate with that topic can be added to the signed log. The community and core team are actively working on DNS to give short names to keys, so that users can have a "foo.com" address for their topic.

Beaker Browser has implemented an informal social protocol on Hyper based on profile drives and private address books. Users publish "microblog posts" by writing files to the /microblog/ folder of their profile drive which applications then merge and sort by the ctime metadata of the files. (This system functions similarly to RSS.) Social relationships are not currently public.

Privacy & Access Control

By default, Hypercore Protocol connections are encrypted using the Noise protocol encryption framework. Hyperswarm does not hide user IPs or mask metadata.

Hyper data structures can only be accessed by users with knowledge of the public key. Because public keys are not guessable, this means they must be shared out of band prior to access (it is not possible to access the data by watching the network).

Fine-grained access within Hypercore data structures, such as access to individual files or folders in a Hyperdrive, is not yet implemented.

Interoperability

Hyper is not interoperable with any other protocol and has not prioritized implementing access over HTTP through gateways, though it would be possible.

The Hyperdrive data structure is designed to be POSIX-compatible which enables Hyperdrive access via a FUSE interface.

Scalability

Scalability statistics are not available for the Hyper network.

Metrics

There are about 300 to 400 nodes in the Hyperswarm DHT as of June 2020.

On Github, Hyper has 38 contributors on the Hypercore module and 781 dependent projects. The Hyperdrive module has 27 contributors and 531 dependent projects.

Governance & Business Models

The Hypercore protocol's development is led by two companies, Blue Link Labs (which develops Beaker Browser) and Hyperdivision consulting. Development is coordinated through community effort in the Hypercore Protocol GitHub Org. A biweekly "community consortium" call is conducted between Blue Link Labs, Hyperdivision consulting, and 6-10 other companies in the community.

Blue Link Labs's business model is to run cloud Hypercore hosting services. Hyperdivision consults with companies implementing Hyper in their software. Many of the community projects are grant-funded.

Implementations & Applications

The Beaker Browser is a P2P web browser for Hyper sites.

Kappa DB is an append-only log database that uses hypercores.

Applications using Hyper include:

- Blahbity Blog, a Twitter clone built in Beaker Browser.
- Sneaky Wiki, a wiki application built in Beaker Browser.
- Peer-to, an online art exhibition only available through Beaker Browser.
- Mapeo, an offline-friendly mapping software to support indigenous land rights.
- Cabal, a P2P chat platform using Hypercore.
- Ara, a blockchain content platform.
- CoBox, a distributed, encrypted, offline-enabled data hosting cloud platform.
- Wireline, a network stack for P2P and serverless cloud computing.

Links

- Hypercore Protocol
- Beaker Browser
- Comparing IPFS and Dat

IPFS

IPFS is a content-addressed protocol for peer-to-peer hypermedia storage and distribution. The IPFS network is mainly used as a storage layer for decentralized applications. Its main purpose is to add, share, find, and transfer files in a globally distributed file system. Components of IPFS, such as libp2p, the P2P networking library, and IPLD, the data model for content-addressed Merkle DAGs, are used separately by applications that do not participate in the IPFS network.

The IPFS alpha launched in February 2015. IPFS protocols are designed for upgradeability.

Identity

IPFS nodes have a peer ID, the hash of their public/private keypair. When peers connect, they exchange public keys and check to make sure they match the node IDs. Communications are encrypted using these keys. Node IDs are pseudonymous and can be reset as needed to maintain privacy. Node private keys are stored in the IPFS config by default.

IPFS can serve as a content addressed storage system for decentralized identity solutions (such as Microsoft's ION). IPID is an implementation of the DID (decentralized identifiers) specification over IPFS, using IPNS.

Network

IPFS is a distributed protocol. Every node gets to participate in the network in any configuration, enabling different kinds of network topologies to emerge. Nodes can connect to each other independently of the client (browser, mobile, desktop, command line).

When nodes join the network, they bootstrap off of long-lived peers or those in their local area network. IPFS comes with a default list of trusted peers, which can be modified. A node can opt in to be part of the main public network and/or an alternative network. Nodes joining the main public network will join the DHT as either clients (consumers) or servers (providers of the content routing service). Nodes can also directly connect to peers they're interested in either through peer ID or through subscribing to relevant PubSub channels.

All nodes in the IPFS network use libp2p, the modular networking library, to make peer-to-peer connections. Libp2p is transport agnostic, leaving the choice of transport protocol up to the developer and allowing an application to support many different transports at the same time. Peers dial to each other using a multiaddr, a self-describing network address that lets peers know a node's preferred way to be dialed. The use of multiaddrs is intended to future-proof addresses and allow multiple transport protocols and addresses to coexist. All connections in IPFS are end-to-end encrypted and authenticated using public/private key cryptography.

Gateways allow IPFS to be accessed over HTTP, which makes content stored in IPFS accessible through a standard browser.

Data

IPFS is commonly known as a distributed file system, but the data layer is closer to a graph database with elements of linked data. IPFS uses IPLD for representing any piece of data available in the network. The IPLD data model treats all hash-linked data structures as subsets of a unified information space. Higher level abstractions can be built on top of it, like the OrbitDB database or Textile threads.

Files are located in the IPFS network by their CID, a content identifier that is based on the hash of the file. The hash of a file does not change, so this form of addressing does not allow updates. However, mutable addresses can be built on top by using the hash of a public key as an address. IPFS's native version is IPNS (InterPlanetary Name System), although other versions (such as ENS) are compatible. The keypair associated with the address is used to sign content that is published under it. DNSLink is also used to map a domain name to an IPFS address. It is currently faster than IPNS and has the advantage of being human-readable and memorable.

IPFS does not have built-in incentives for nodes to persist content to the network. Users store their own data by 'pinning' it to their local IPFS nodes, communities collaborate in backing up data through tools like 'collaborative clusters', and enterprises pay 3rd party pinning services like Infura to ensure availability and reliability. Projects like Storj and Filecoin are building blockchain networks for incentivizing persistent IPFS storage.

If all users stop hosting a piece of data, it is removed from the network. However, if a node chooses to continue hosting it, it can still be located by its content ID.

Moderation & Reputation

IPFS is mostly a layer below identity and reputation, but libp2p has some low-level primitives around connection management which can be used to encode peer reputation. Peer reputation is based on how reliable the peer is at returning requested data. Both libp2p and IPFS support the explicit configuration to avoid or block known bad IPs.

Each IPFS node is in full control of the data it pins. Nodes can add a denylist to their configuration, optionally using the one used by public gateways to block DMCA takedown content, malware, and other illegal or pernicious content. There is a proposed design for an autonomy-preserving content moderation system by which nodes can subscribe to denylists from entities they trust to help avoid or filter unwanted content.

Social & Discovery

IPFS uses a DHT for finding content in the network. Each host advertises the data they're storing once per day, which can be looked up through the DHT. Nodes also discover peers through their local area network and by bootstrapping with other nodes. The DHT is also used for bootstrapping PubSub channels which groups can subscribe to for topic-based updates to content they care about.

Privacy & Access Control

Content published to IPFS is public by default. Encryption can be used to add privacy and access control layers on top of IPFS.

Interoperability

IPFS gateways allow the network to be accessed over HTTP in browsers without native IPFS support.

The IPLD data structure is designed to allow any kind of hash linked data to be ingested into IPFS, including blockchains like Bitcoin and Ethereum, and git repos.

Scalability

The IPFS public network currently has hundreds of thousands of nodes. Private networks also run IPFS without connecting to the main DHT and are not included in the node count.

IPFS nodes have historically had high resource consumption, although improvements and 'low power' settings for weaker devices have since been added.

Metrics

- Automated metrics about IPFS related projects
- 100ks of nodes as of May 2020
- ~4000 contributors

Governance & Business Models

IPFS is developed by Protocol Labs, a VC-funded company that raised over 200 million in a token sale for Filecoin. The core implementations working group, consisting of both employees of the company and

external contributors, has decision-making authority over contributions to the IPFS protocol. Libp2p, IPLD, and Filecoin are stewarded by separate working groups.

Contributors from the open source community either volunteer their time, or are funded through companies that have raised money to build on top of IPFS.

Implementations & Applications

Implementations of IPFS exist in Go and JavaScript, and a Rust implementation is under development. Projects like Textile, OrbitDB, and 3Box have built additional layers of tooling on top of IPFS to support a wider range of applications.

Examples of tools that have expanded the use cases of IPFS include:

- Textile buckets - dynamic folders for decentralized applications, distributed over IPFS.
- OrbitDB is a serverless, P2P database that uses IPFS to store data and IPFS PubSub to sync databases with peers. It uses CRDTs to resolve conflicts.
- 3Box has a web application framework that stores data in IPFS.

Libp2p is used, independently of IPFS, by other decentralized networks such as Polkadot, ETH2, and Matrix, which is experimenting with it as a transport layer for the P2P version.

A list of applications that use IPFS: https://awesome.ipfs.io/

Notable P2P applications include:

- OpenBazaar, a marketplace.
- Peepeth, a social network on IPFS and Ethereum.
- Peergos, a private distributed file sharing protocol and application.
- Dtube, a YouTube alternative.
- Everipedia, a wikipedia alternative built on IPFS .
- Audius, a music streaming service.
- Anytype, a locally hosted Notion-like writing platform.

Enterprise adoptions and integrations include:

- Microsoft uses IPFS for their DID implementation.
- Cloudflare runs a popular IPFS gateway hosted at https://cloudflare-ipfs.com/.
- Netflix switched to IPFS for docker container distribution, improving performance 2x.
- Opera IPFS is supported by default in the Opera browser for Android.

Links

- Docs
- Mapping the Interplanetary Filesystem
- Comparing IPFS and Dat

Matrix

Matrix is a protocol for replicating a signed history of JSON objects in real-time across a set of nodes with (optional) end-to-end encryption. It was started in 2014.

Identity

A Matrix identifier takes the form of *localpart:homeserver, where * is a “sigil” character which is used to identify the entity’s type. The sigil character "@" states that the entity is a Matrix user ID, and the "localpart" is an identity allocated by that homeserver. For example:

@bob:matrix.org

Other sigil IDs include "!" for Room ID, "$" for Event ID, "+" for Group ID, and "#" for room alias.

User accounts, once created on a homeserver, cannot be migrated. To change servers, a user must make a new account. Automated tooling exists to help with inviting the new account into rooms the previous account was in. There are eventual plans to switch to user keys which will enable portable identities. A Matrix account could also map to a DID.

Users have a Matrix user ID, but can also use 3rd party IDs (3PIDs). Matrix identity servers map 3rd party IDs such as email addresses and phone numbers to Matrix IDs. The use of this service is optional. A globally federated cluster of trusted identity servers verify and replicate the mappings, although this is considered a stopgap solution until a fully decentralized identity solution is adopted.

User IDs used in conversations will soon be decoupled from permanent IDs, allowing one to decorrelate users from their messages.

Network

Matrix has a federated and a P2P version.

Federated

In the federated version of Matrix, all messages are currently sent out full-mesh within a conversation. A node broadcasts in parallel to every other node present in the room. Experimental work on stochastic spanning tree "fan-out" approaches to improve efficiency are being researched.

P2P

Matrix has released a P2P version that runs client-side. P2P Matrix avoids the problem of homeservers accumulating metadata and simplifies signup by not requiring new users to pick a homeserver. The new P2P implementation runs the homeserver on the client. The P2P network is currently separate from the federated network, but the end goal is to connect the two in a hybrid federated/P2P model. Network transports being considered for P2P Matrix include libp2p, Yggdrasil, or Hyperswarm. In P2P Matrix, the hostname of the user is its libp2p or Yggdrasil public key.

https://matrix.org/blog/2020/06/02/introducing-p-2-p-matrix

Data

Matrix messages are stored in per-conversation Merkle DAG data structures and conversations are replicated across all participating servers. Matrix is architecturally most similar to Git.

A Matrix "room" refers to a persistent PubSub topic. Users who want to chat join the same room and their conversation history is then replicated on each user's server.

Moderation & Reputation

The Matrix team has been working intensively on tools for moderation, detailed here.

Most moderation takes place at the room level. Rooms have moderators who can remove undesirable content. A redaction, or “remove message”, can request all participating servers to remove a message. Users can also remove their own messages. However, due to the decentralized nature of Matrix, if the client does not remove the redaction, the message will not be removed. Moderators can kick or ban users from rooms. They can also ban servers from rooms, in cases where malicious traffic must be avoided.

Users in a room have “power levels”, a number between 0 and 100 that indicates how much power the user has in the room. A higher level has more permissions. Moderators have the ability to change power level permissions. Users can report abusive content to the server admin, who has the ability to remove it. Users can also block other users to prevent them from contacting them.

Social & Discovery

All conversations on Matrix take place through rooms, which people either join (if public), peek into (if viewable), or are invited to.

Features supporting more advanced social functionality are being developed, such as this proposal for tracking events related to existing events: Proposal for aggregations via relations

Privacy & Access Control

Matrix homeservers have access to metadata about conversations because the homeservers of all users in a given conversation have to store that conversation's metadata. P2P Matrix mitigates this privacy issue.

Matrix recently introduced end-to-end encryption by default for private messages. This was on the roadmap since the beginning because conversations are replicated over every server participating in a room, and there is no guarantee against servers looking into conversations. For Matrix E2E encryption, each device has its own identity key, and per-user keys are gossipped between devices when the user logs in.

Olm E2E encryption encryption for private messages

All Matrix APIs are protected behind GDPR consent flows. Users can specify history retention per-room, per-server, and per-message.

Monetization & Governance

Matrix is governed by The Matrix.org Foundation CIC (a UK non-profit foundation), with the largest contributor being Element (previously called New Vector), a startup formed by the original Matrix dev team, which has raised a total of $18M. Income primarily comes from contracts with governments and large clients. The Matrix standard evolves under the stewardship of a Spec Core Team that manages contributions.

Interoperability

Matrix can be bridged with IRC, Slack, Discord, Telegram and others.

Implementations & Applications

Matrix is designed to support multiple communication cases - chat, VoIP, IoT, VR/AR, social, etc., but so far effort has been concentrated on providing a federated chat experience with good UX at scale.

Matrix supports multiple clients (most notably Element (previously called Riot), the flagship app from the core team) and has bridges to many other chat systems (IRC, Slack, Discord, Telegram etc.).

The majority of traffic is currently instant messaging.

Scalability & Metrics

Matrix “rooms” replicate message content across all participating servers but lazily replicates file content. A server only retrieves a file if a user on that server requests it.

The public network has 17.9M known addressable users (as of June 2020), with more in private federations or on servers which don't report stats. 30% of publicly visible users are on the matrix.org homeserver. There are many smaller servers as well.

Scalability study: https://arxiv.org/pdf/1910.06295.pdf

There are over 1,500 active developers in the community.

Links

- matrix.org

Peergos

Peergos is a P2P end-to-end encrypted storage and application protocol on top of IPFS. It is also a social file-sharing application by the same name. The goal of Peergos is to provide a global multi-user filesystem that provides privacy, identity, login, and secure sharing for decentralized applications. It is designed to be independent of DNS and TLS certificate authorities, and to protect privacy through quantum resistant encryption.

Peergos was started in 2013 with the aim of a secure decentralized file storage and sharing network with a secure email replacement.

Identity

Peergos users are identified by unique usernames linked to public keys. The uniqueness of usernames is ensured through a global append-only log for public key to username mappings that is mirrored on every node in the Peergos system. Names are taken on a first come first served basis. Currently, a single server determines the canonical state of this log and other nodes sync to it. Long-term considerations include decentralizing the name server through a blockchain architecture.

Peergos allows multi-device login through a password-based interface. A user's private keys are derived every time they log in using their username, password and a published salt. Specifically, a signing keypair, boxing keypair, and symmetric key are derived. Users store their friend's keys in their encrypted storage space in a TOFU keystore. Users can verify their friend's keys out of band using QR codes or fingerprints.

Network

Each user must be registered to at least one Peergos server (a server can host any number of users and any server can choose to mirror data for any user). Peergos servers run an instance of IPFS, which handles networking and connection management. A server is any device storing user data, and could be a mobile phone, a cloud server, or hardware plugged in at home.

Data

Data in Peergos is content-addressed: stored in mappings from hash to data. During upload the client splits files into 5 MiB chunks which are each independently encrypted (along with encrypted metadata) and stored in a merkle-CHAMP (compressed hash array mapped prefix trie) in IPFS. Directories can't be distinguished from small files. To hide file sizes and split files up into 5 MiB chunks which aren't linkable, the size within a chunk is also rounded up (padded before encryption) to a multiple of 4 KiB. This means all chunks can only have 1 of 1280 possible sizes. Servers do not have visibility into the file sizes, the number of files, directory structure, or which users have access.

The user publishes the IPFS node ID (hash of its public key) of their server in the PKI. This allows writes and new friend requests to be securely tunneled to their server. It synchronizes their writes and publishes their latest root hashes.

Social & Discovery

Users can follow each other. Follow requests are sent through a user's storage server, which is contacted via its public key. Follows are one-way and allow sharing files and sending messages. Critically, the server never sees who is following who (even follow requests are blinded). Users store their own social graph encrypted in their Peergos space.

Privacy & Access Control

All encryption happens on the client, which could be a native Peergos client or a browser. Data is always encrypted on the servers. Servers do not have access to metadata or sensitive information.

Access is controlled through cryptographic capabilities. Access is hierarchical and stored in an encrypted structure called cryptree.

A read-only capability consists of the hash of the file owner's public key, the hash of the writer's public key, a random label, and a symmetric encryption key. Access to files gained through social follows can be revoked by rotating cryptographic keys, but the interface does not display keys to users. Users simply click "revoke access to".

To make a file or folder publicly visible, a user can publish its capability. A user can also share secret links to files, like a Google Doc "share" link, which lets anyone who views it view the file. These secret links don't expose the file to the server. The file is not transmitted unencrypted over the network, as the key to decrypt it is in the URL itself (in the hash fragment which isn't sent to the server), and is interpreted locally in the browser.

A write capability includes the private key corresponding to the writer key, which is used to sign updates.

Peergos has undergone an independent security audit by Cure53.

Scalability

Peergos can handle arbitrarily large files, including random access, upload and download, and on under-powered devices like mobile phones. This is largely due to the independent encryption of each 5 MiB section, as well as the "zero IO" seeking within a file.

Governance & Business Models

Peergos was developed by the core team on a self-funded volunteer basis for years. It has received grants from Protocol Labs, the company that stewards IPFS, and from Santander.

Implementations

Peergos is a private and access controlled layer on top of IPFS which can be used to build applications. Other than the Peergos reference implementation client which allows users to store and share private files, there are a few demo applications, including a read-only viewer for PDF files, and an editor for text or code.

The goal of allowing people to deploy apps on Peergos that are viewable from the browser is currently limited by sandboxing constraints within browsers. COOP/COEP increase the security of in-browser applications.

Links

- Peergos
- Peergos book
- Source code
- COOP and COEP explainer

Solid

Solid (derived from "social linked data") is a proposed set of conventions and tools for building decentralized social applications based on Linked Data principles. It relies as much as possible on existing W3C standards and protocols, tying them together into a common framework.

The main concept behind Solid is that users store their personal data in a "pod" (personal online data store), and applications request access to it, rather than storing user data and accounts on an application server.

Identity

Solid uses WebID URIs as universal usernames. The WebID URI's primary function is to point to the location of a public WebID Profile document.

Example WebIDs: https://alice.databox.com/profile/card#me or http://somepersonalsite.com/#webid

A WebID is a globally unique, decentralized identifier. It enables cross-service federated sign-in and does not tie a user's identity to a server. A WebID Profile Document is in Linked Data format and contains identity information such as a username, profile image, preferences, and public key certificates.

Solid requires cross-domain, decentralized authentication mechanisms not tied to any particular identity provider or certificate authority, so it uses the WebID-TLS protocol instead of passwords. Instead of creating an account with a username and password for each service, a user selects a security certificate for the site from a popup. The server matches the private key stored by the user's browser with the public key stored in that user's WebID Profile Document to authenticate them.

Username and password authentication mechanisms are an active area of research. Solid recommends that servers implement secondary account recovery mechanisms, such as email recovery, in case browser certificates are lost.

There is some discussion of using DIDs in addition to WebIDs.

Network

Solid allows inboxes for any resources, such as actors or articles. An example of an inbox for annotations related to a particular article.

Solid provides a HTTPS REST API and a WebSockets API for a PubSub mechanism. Smart clients currently retrieve messages with pull/get, rather than through server push.

Notifications use the Linked Data Notifications standard.

Data

A data storage space is called a "pod" (personal online data store). All a user's data is stored in their pod. Users may self-host their pods (on their own server) or use a "pod provider", a federated pod server. Applications read and write data into the pod depending on the authorizations granted by the users associated with that pod. A person may have multiple pods, for example for work and for home. Data may be replicated across pods. A key concept of the pod is that users may switch pods easily without losing their data (such as their contacts and chat history).

Solid strongly encourages the use of RDF-based Linked Data (RDF in the form of JSON-LD, Turtle, HTML+RDFa, etc.) for interoperability with the ecosystem. The Solid content representation spec encourages consistent naming conventions.

An example payload:

{

```
  "@context": {
    "@language": "en",
    "sioc": "http://rdfs.org/sioc/ns#",
    "foaf": "http://xmlns.com/foaf/0.1/"
  },
  "@id": "",
  "@type": "sioc:Comment",
  "sioc:reply_of": { "@id": "http://example.org/article" },
  "sioc:created_at": {
    "@type": "http://www.w3.org/2001/XMLSchema#dateTime",
    "@value": "2015-12-23T16:44:21Z"
  },
  "sioc:content": "This is a great article!",
  "sioc:has_creator": {
    "@id": "http://example.org/profile",
    "@type": "sioc:UserAccount",
    "sioc:account_of": { "@id": "http://example.org/profile#alice" },
    "sioc:avatar": { "@id": "http://example.org/profile/avatar.png" },
    "foaf:name": "Alice"
  }
}
```

Social & Discovery

Solid aims to create tools that enable the building of interoperable decentralized social applications. WebIDs allow user addressing across all Solid-conforming apps. Resources (including WebIDs) are self-describing, hosted on pods, and have HTTP endpoints with standard naming conventions to make discovery easier. Any app that can parse the format can read the inbox and send responses.

Privacy & Access Control

A user's data is stored on their pod, which applications request access to. A user grants access to their data by selecting a security certificate to use with an application. If the user revokes access of an app to their data, the app will no longer be able to read or update the user's data, but would still have access to the data that the user sent out prior to the revocation.

Data is always encrypted when in transition between an app and a pod. Pod providers can encrypt data at rest on a pod, but are not required to.

Solid uses the Web Access Control Spec for management of access control lists.

Interoperability

Solid provides specs and recommendations for interoperability between Solid ecosystem social applications. There are weekly meetings for the Solid Data Interoperability Panel to discuss data interoperability across applications.

Solid is potentially compatible with federated social applications. For example, a Solid pod could be used in the implementation of an ActivityPub server. The minimum requirement to be conformant with the Solid protocol is the use of JSON-LD, which ActivityPub uses. dokieli (a Solid app) is able to send notifications to ActivityPub. A discussion of the links between ActivityPub and Solid highlights similarities, although work is required to close the gap between the specs.

Governance & Business Models

Pod providers can choose whether to charge for hosting a pod. Possible business models include charging users for storage or advertising through applications.

Inrupt is a company committed to the development of Solid. It plans to monetize through consulting and pod hosting fees.

Implementations & Applications

A list of Solid apps. Apps built to use Solid data pods are considered part of the Solid ecosystem.

Solid does not have many widely used social applications yet, although there has been research such as A Demonstration of the Solid Platform for Social Web Applications.

An example application is:

- dokieli for publishing articles and posts

Links

- Solid
- Solid spec
- Solid repo
- Solid paper
- Solid Community Group Meetings

Scalable Secure Scuttlebutt (SSB)

Scalable Secure Scuttlebutt (SSB) is a distributed gossip protocol designed for social sharing. Identities are cryptographic keypairs, feeds are a signed append-only log sequence of messages, and nodes use a gossip protocol to disseminate content. Feeds can be thought of as essentially personal blockchains, as they consist of immutable, timestamped content.

SSB is based on the idea that your social network mirrors your actual communication network, and your network peers mirror your actual peers. SSB focuses more on moving lightweight social data rather than large data, unlike protocols like BitTorrent, Hypercore, and IPFS. SSB's design philosophy avoids centralization and singletons, and strives for a maximally distributed architecture. Users are distributed across a few different client apps that work on desktop and mobile.

Identity

A user's identity is their ed25519 keypair which is used to sign posts, verifying their authenticity. Messages are addressed to a user's public key, for example:
@3QHXrXl762sf7P/Q1RMtscA7IRipfUFnE5tpie5McvE=.ed25519

Users can pick a human-readable nickname that is associated with their key, but nicknames are not unique because there is no global registry.

SSB does not currently support multi-device login because keys are stored on devices.

Network

Nodes request all messages in the feed that are newer than the latest message they know about. The networking component of SSB maintains a table of known peers which it cycles through asking for updates for all followed feeds. Messages are passed through the SSB network via a gossip protocol. Messages may be passed through third parties, which improves availability.

Pubs, bot-user nodes with public IP addresses that stay online, ensure uptime and availability. Pubs are essentially the bootstrap nodes and mail-bots of SSB. Pubs offer invite codes to new users, follow users, and rebroadcast messages to other peers. SSB has no NAT-traversal utilities, so users connect to a pub to distribute their messages. Users can also sync over LAN. Identity is not tied to pubs, unlike homeservers in Matrix or ActivityPub, so a user can join one or multiple pubs. A scuttlebutt DHT invite plugin that shares connection invites over a DHT was created in 2018.

Data

Each post is appended to the last in a linked list append-only log establishing chronological ordering from the first post. Once appended, posts are immutable. Content cannot be easily edited or deleted, but the append-only log ensures that the network converges towards the same state despite partitions.

When a follow relationship is initiated, the posts of the user being followed begins to be synced to the follower's node. Those messages and files are stored locally on the user's computer, indefinitely, for applications running SSB to read. A SSB application is constantly sharing data with other nodes in the background.

Each message contains:

- A signature
- The signing public key
- A content hash of the previous message
- A sequence number
- A timestamp
- An identifier of the hashing algorithm in use (currently only "sha256" is supported)
- A content object

Because of the append-only nature of SSB feeds, there is no ability to permanently delete a piece of content. Applications can work around this by honoring edit or delete messages appended to the feed, but the original content stays in the append-only log that is shared among all nodes and other applications could choose not to honor such messages. An example of a workaround is ssb-revisions, a basic API that enables applications to use mutable messages by displaying the updated version.

Moderation & Reputation

There is no global moderation and no specialized moderators in SSB. A “flag” message is used to send a strong negative signal about bad actors. Applications built on top of SSB allow users to “block” and “ignore”. An ignore will simply not show that data to the user's node, although their node will continue to pass their data through the network. A block will cause the user's node to refuse to replicate data from that feed, segmenting it off from their portion of the network. If enough people block a user or group of users, their part of the network will become partitioned from the rest.

Social & Discovery

There is no global feed of content in SSB. All content is surfaced through social discovery. Out-of-band sharing, sending a SSB link through another channel, can also surface new content.

SSB clients decide the number of hops away from primary follow relationships to store or replicate data. For example, a client could store data from 2 hops away, but replicate data from 3 hops away to keep it available for others, but not show it in the user interface.

Privacy & Access Control

SSB applications can easily support encrypted DMs (with up to 7 participants) because user identities are cryptographic keypairs. Whoever controls the private key of an identity can publish to that feed. Messages cannot be faked, omitted, or re-ordered, due to the signed append-only log nature of the feed.

There is ongoing research and development on providing "Facebook Groups" style of access control to larger private groups (in the dozens/hundreds of users).

Governance & Business Models

The SSB ecosystem is supported through a variety of grants, donations, income from side projects and consulting, and a few companies that have raised money to build applications on SSB, including Ahau and Planetary.

Pubs, the most resource-intensive nodes, are currently volunteer supported.

Interoperability

SSB viewer, an HTTP server for read-only views of SSB content, brings read-only interop from SSB to the web.

SSB applications generally do not bridge to other applications. A proof-of-concept experiment involving cross-posting to Twitter and importing tweets into SSB demonstrates the possibility for a simple interop.

There has been community discussion of using IPFS for blob data storage in SSB, but it has not been implemented as a feature.

Scalability

New users add capacity to the network as they join as their nodes participate in hosting and sharing content.

A growing number of room servers help with scalability since rooms host no data, but allow tunnel connections between clients, so the more clients there are, the more connections there are available.

Pub servers would need to be expanded to keep up with a sudden influx of new users.

A potential scalability issue is the size of the append-only log feeds stored on a user's device growing over time.

Metrics

According to a network crawl run by a developer in Nov 2019, there were around 16,000 nodes on SSB visible to his node.

Implementations & Applications

A list of applications built on SSB.

Social application clients on the SSB network include:

- Patchwork, a desktop application
- Patchbay, a fully compatible alternative to Patchwork
- Oasis, a desktop application
- Feedless, an iOS application
- Manyverse, a mobile application
- Planetary, a mobile application

Other applications include:

- git-ssb, a git interface using SSB
- Ticktack, a blog publishing app with private messaging
- Infinite Game, a calendar, event tool, and time picker

Links

- Overview
- SSB concepts
- Dark Crystal SSB protocol docs
- 3Box comparison of P2P DBs: GUN, OrbitDB, Scuttlebutt

XMPP

XMPP, the Extensible Messaging and Presence Protocol, is a protocol designed for instant messaging and generalized routing of XML data. It was originally developed in the Jabber open source community in 1999 to provide an alternative to closed instant messaging services.

XMPP uses a federated client-server architecture that is similar to email, but introduced several modifications to facilitate real-time communication.

XMPP is a large protocol, with many Extension Protocols that have been added over time. Due to the maturity of the protocol, it is well documented and has many implementations and applications.

Identity

Every user has a unique XMPP address, called a JID (Jabber ID). A JID is a username followed by the homeserver, for example:

alice@example.com

Users can choose a global, memorable nickname, but these are not globally unique.

Network

XMPP is implemented in a distributed client-server architecture.

Data

The basic protocol data unit in XMPP is an XML "stanza", a fragment of XML that is sent over a stream.

An XMPP client may store data on the server. Whether it is accessible to others or not depends on the client implementation.

Default settings for a Jabber server store information including: saved contacts ("buddies"), offline messages, error logs, and perhaps chat messages and file uploads for a limited number of days.

Moderation & Reputation

Due to the federated nature of XMPP, moderation actions are only respected if they are supported by the client.

Some examples of moderation in XMPP: An experimental spec for groupchat moderation

Social & Discovery

XMPP includes the ability for users to advertise their network availability or "presence". This is not strictly necessary for the exchange of data, but facilitates real-time interaction by indicating the recipient is online and available. Users can subscribe to each other's presence statuses.

Users have contact lists called "rosters". Rosters are often stored on the server to protect the user's social graph.

Privacy & Access Control

XMPP specifies that channel encryption should be used for the XMPP and HTTP channel via SSL or TLS.

The original protocol did not include end-to-end encryption by default. Encryption methods were later added as extensions to the core protocol.

XMPP includes a method for authenticating a stream using the Simple Authentication and Security Layer (SASL).

Interoperability

XMPP was designed with the goal of connecting users through multiple instant messaging systems. This was done through transports or gateways to other IM protocols, but also to protocols such as SMS and email.

In February 2010, the social networking site Facebook opened up its chat feature to third party applications via XMPP. Some functionality was unavailable through XMPP and support was dropped in April 2014.

Similarly, in December 2011, Microsoft released an XMPP interface to its Microsoft Messenger service. Skype, its de facto successor, also provides limited XMPP support.

Scalability

A single ejabberd node with 2+ million concurrent users

Implementations & Applications

A list of XMPP clients. XMPP has use cases beyond chat such as IoT, gaming, social, and WebRTC.

Apple and Google use XMPP for push notifications.

WhatsApp, Zoom, Grindr, and Jitsi use XMPP for chat.

Governance & Business Models

XMPP was developed by the open source Jabber community without funding.

Applications that use XMPP are varied and have many different business models from advertising to SaaS.

Links

- XMPP About
- XMPP Wikipedia
- XMPP Core Memo

Applications

Aether

Aether is a Reddit-like P2P social network - a public discussion forum designed with ephemerality and mutability in mind. It is available as a desktop application. Aether's P2P protocol is called Mim. In practice, Aether is the only application using Mim, but it could support other applications.

Identity

Identities in Aether are based on keypairs, like in SSB. As with SSB, users can choose a custom nickname, although it is not unique. Multi-device usage is currently possible, but difficult (requires manually porting a user config file). There is a strategy for multiple device logins under development that would allow users to store and sync encrypted keys from a remote backend.

Network

Aether is a flood network where every node stores everything. All nodes download content as well as provide it to other nodes. Communities with objectionable content can optionally be blocked by the user, so that content is not stored on their machine.

Aether comes with a list of bootstrap nodes. Nodes find each other through a probabilistic search over a network table. This is a random process in which no patterns of data distribution can be discerned, so only the last hop of where data came from is visible, not the source. Once nodes are connected, they sync their data.

Syncing nodes request caches of data from others. Every node stores pre-packaged answers to common queries. Data from each endpoint (boards, threads, posts) will be cached every day and stored for 7 days, after which it is no longer cached. Bootstrapping nodes do not download the entire history of the network - only the last 7 days by default.

New posts have a 0 to 10 minute delay, as it takes time to traverse the P2P network.

Data

Aether's data structure is a DAG (directed acyclic graph). It allows insertion and updates of content. Deletion of content is achieved by inserting a blank update. The backend keeps a local SQLite database up-to-date with the state of the network. The frontend compiles and presents the graph data.

Posts are ephemeral. After a certain period of inactivity on a post (the default is 6 months from the last reference), it automatically gets deleted by nodes on the network. If a post continues to be referenced, it stays up, so actively discussed topics do not disappear.

Moderation & Reputation

Each sub-community has its own moderators, which communities elect or impeach themselves. Like subreddits, communities largely govern themselves. Mods (moderators) can approve or delete posts.

Users can report posts. Reports go to mods of the community. Users elect moderators in communities and can impeach them. If a user votes to impeach a mod, the mod's actions will be ineffective for that user, even if the vote wasn't won. All mod decisions are public history.

Moderator elections begin once a community has more than 100 active users within the last 2 weeks. Voting thresholds for moderators is a 51% positive vote of at least 5% of the community's members. To prevent hostile takeovers, the population of communities is determined on a 2-week trailing basis and mods can temporarily lock a community to new entrants.

Users who wish to be mods can enable “mod mode”, which allows them to proactively attempt to approve or delete posts. Their moderation history can be observed by others considering electing them as an official mod.

An allow-list is used to moderate the collection of communities. When users join, the default is to only display SFW communities on a vetted allow-list, to ensure the safety and comfort of a user's initial experience.

Spam

Aether combats spam by requiring users to complete a small "proof-of-work", a brief hashing function, before they post. This rate-limits the amount of posts that a user can send out to the network, making it more expensive to flood it with spam.

Social & Discovery

Users can search communities, content, and people. The user interface has tabs to discover newest content and popular content.

There is no follow functionality for other users at present.

Privacy & Access Control

All posts are public.

Monetization

The Aether P2P version is supported through Aether Pro, a SaaS version of Aether designed for private work deployments.

Interoperability

Aether P2P is working on implementing a chat that bridges to Matrix P2P.

User Experience & Scalability

Aether is currently only available on desktop. There is no mobile app as mobile devices have limited disk space and energy stores to run the P2P network.

However, a mobile app is under development that uses the following method: Running an Aether host in a docker image on a homeserver and allowing mobile devices to connect to it.

Creating a mobile app in a way that requires users to run homeservers, rather than simply providing full nodes that would allow a mobile app to download content, is a design decision that maximizes decentralization at the expense of user convenience. Every mobile user running a homeserver will also still be providing capacity to the network to help it scale in a purely P2P manner.

Links

- Mim protocol

Blockchain Social Applications

This section covers some aspects of social applications that store data on a blockchain and/or use an associated cryptocurrency for monetization.

Steemit

The Steem cryptocurrency was created for content monetization on social sites. Steemit, a Reddit/Medium-style social network, was the first site built to use Steem. User identities and post data are stored on the Steem blockchain. There are over a million accounts on Steemit.

Identity

User identities are stored on the Steem blockchain. There is a monetary incentive to create many accounts to upvote posts so as a spam and sybil prevention mechanism, new account creation requires an email and phone number and must go through a centralized review process.

A Steemit account functions as a cryptocurrency wallet and users are responsible for their own key management. There is no account recovery available and funds can be lost or stolen if the key is compromised. Accounts cannot be deactivated or deleted, since they are permanently stored on the Steem blockchain.

Data

Text data is stored on the Steem blockchain, but larger data like images are stored off-chain in a database.

Moderation

Steemit takes a bottom-up approach to moderation. Content is moderated through the up and down votes of users, instead of through the actions of a moderator. Low voted comments may be hidden. User reputation determines the weight of votes in the network and since reputation accumulates with age, older accounts have more voting power.

Steemit's approach to spam, plagiarism, and abuse relies on a single mechanism: user voting to downgrade undesirable posts. To add a negative signal to a post and downgrade its rewards, users could "flag" or "downvote" - two terms used for the same function of downgrading a post. A discussion of the downvoting and flagging problems on Steemit goes into the drawbacks of various approaches.

Users can flag content they find objectionable, but they cannot block other users due to the potential for abuse of the block feature in a system that monetizes upvotes. For example, a user might block users with high reputation who might downvote them so they could upvote their own posts within a circle of participating accounts.

Monetization

Steemit mined 80% of Steem in the first week. Steemit benefited from the appreciation of Steem during the 2017 cryptocurrency run-up, but had to lay off much of its staff when the price of cryptocurrency declined.

Other than depending on Steem price appreciation, Steemit monetizes through users promoting their posts. When users perform certain actions on Steemit, they earn Steem. Creating posts that get upvoted qualifies users to earn from a rewards pool. Upvoting posts that later become popular can earn voters a curation reward. Votes are weighted by reputation, which accumulates with age, so older accounts of early adopters have more power in the network. This, as well as the fact that Steem tokens could be mined easily early on, means that Steemit’s incentives are geared towards early adopters.

Links

Steemit frontend application

Peepeth

Peepeth is a “permanent Twitter” built on Ethereum and IPFS. Users log in to Peepeth through MetaMask and their identities are registered by posting account information to a smart contract that associates it with their address. Account information is stored on the Ethereum blockchain.

Post data is stored in IPFS and a link to the IPFS address is stored on Ethereum. Permanent posts are embraced as a design choice meant to encourage users to be more mindful of what they write.

The main Peepeth client has a few monetization strategies, including a 10% transaction fee on user tips and charging for verifications of social accounts.

Bitcoin Fork Social Networks

Memo is a Twitter-style social network that stores each action in a transaction on the Bitcoin Cash blockchain. Data stored on the blockchain is immutable and public. Memo uses a reputation system based on the ratio of how many people a user follows are mutual, compared with how many have muted the user.

Twetch is a Twitter alternative that stores post data on the Bitcoin SV blockchain. Almost all actions are monetized and have a transaction cost. Users create a moneybutton account for sending and receiving money and must fund their account to write, comment, like, or follow. There is a small charge to create a post as a spam prevention mechanism. A cross-post to Twitter feature exists in Twetch.

Links

- Blockchain Social Networks
- When Blockchain Meets Online Social Networks

Diaspora

Diaspora is a federated social network released in 2010. It uses a server-to-server federation protocol and is compatible with Friendica and Hubzilla.

Diaspora nodes, called "pods", are hosted by different individuals and institutions. User accounts, which are tied to pods, are called "seeds".

Identity

Users joining Diaspora pick a pod to register their identity with. User identities contain the username, the hostname, and the port if their server does not listen on the default ports. An example username:

alice:example.org

Diaspora uses WebFinger to discover users from other pods. User information is returned via hCard, an open microformat standard for identity.

It is not possible to move a user account to another pod once created.

Network

Messages sent between servers are serialized to XML, then signed using the Salmon Magic Signatures protocol.

Data

User data in Diaspora is tied to their pod server. A user can export all of their data at any time, to reduce the risk of their pod going down.

Moderation & Reputation

Like Mastodon, moderation is done at the server level. When ISIS joined the Diaspora network in 2014 after they were censored by Twitter, all the larger pods moved to block their server.

In this GitHub issue about admin reports, it is possible to see how communities struggle when tools for moderation are not built into federated networks from the start. This user requested a way to forward spam reports to the source pod, as that was the only way to remove content from the entire network. Without this feature, pod administrators resorted to manually searching for and contacting other administrators.

Social & Discovery

Like Twitter, Diaspora uses hashtags and mentions for content discovery.

Privacy & Access Control

Profiles in Diaspora can be public or private.

Private messages in Diaspora are encrypted.

Diaspora allows users to group their contacts into "aspects". Limited posts will only be shown to contacts in the selected aspect.

Communication between pods is encrypted, but data stored on pods is not. Administrators have access to all profile and post data.

Monetization

Diaspora was initially funded through a Kickstarter that raised $200,000. It has not developed a business model.

Interoperability

Friendica instances are a part of the Diaspora network and natively support the Diaspora protocol.

Diaspora posts can be propagated to accounts on WordPress, Twitter, and Tumblr.

Diaspora has not integrated with ActivityPub. A Diaspora developer reasoned that although ActivityPub tried to make an extensible protocol that could work for everything, it still did not cover some Diaspora use cases. A stricter specification that could expand definitions as use cases were offered would be better for ensuring interoperability, in his opinion.

Metrics

Number of Diaspora users

Links

- Diaspora Federation Protocol
- Diaspora Discourse Forum

Mastodon

Mastodon is a federated Twitter alternative, released in 2016. It is the most popular client using the ActivityPub federation protocol.

Each server is called an "instance". The entire constellation of instances that can interoperate is called the “Fediverse”.

Identity

Users create an account on an instance, but can communicate with users on other instances. A full username is a user’s handle plus the name of the instance the user belongs to, for example:

@alice@mastodon.social

Usernames are unique to each instance, not to Mastodon as a whole, so @alice@mastodon.social can coexist with @alice@modern.town.

If a user moves to a new instance, they can redirect or migrate their old account. Redirection sets up a redirect notice on the old profile which tells users to follow the new account. Migration forces all followers to unfollow the old account and follow the new, if the software on their instance supports this functionality. Previous posts will not be moved.

Since user identities are tied to instances, if an instance goes down, user accounts and data go with it if they are unable to migrate.

Account credentials are managed by the user’s instance, so if users forget their password, they can ask for a password reset. Whether users can delete their own accounts or not is a setting dependent on the instance admin.

For user verification, there is no central authority to check identity documents, but link-based verification can help cross-reference links associated with a user. For example, a user can link to their Mastodon profile from their personal homepage, and receive a verification checkmark on their Mastodon profile by their personal homepage link to confirm that they are the owner. An identity proof framework was added in 2019, which currently only supports Keybase. It allows users to link their Keybase cryptographic identity to their Mastodon account.

Mastodon uses WebFinger to translate user mentions to actor profile URIs. WebFinger specifies the path at which to find a user's profile information provided by the server. The WebFinger endpoint is always under /.well-known/webfinger, and it receives queries such as /.well-known/webfinger?resource=acct:alice@mastodon.social.

Network

Mastodon started out using OStatus for federation and later switched to ActivityPub. ActivityPub is both a server-to-server and client-to-server standard. Mastodon only implements the server-to-server protocol, which is all that is necessary for federation with other ActivityPub servers. The ActivityPub "activities" Mastodon implements are documented here.

Data

Mastodon is a Ruby on Rails application that uses PostgreSQL and Redis for data storage.

Moderation & Reputation

Moderation takes place at the server level in Mastodon. Each instance sets its own moderation policies, either through the unilateral decisions of an admin or through collective discussion and agreement. Admins

can ban entire instances, cutting off their visibility. If an instance gets banned by many others, its users can still talk with each other, but they will be isolated from the rest of the Fediverse.

Users can apply content warnings to their posts themselves, to indicate the nature of the content and hide it behind a click.

Users can report posts to moderators, submitting it for a moderation decision.

Some documented challenges with moderation in Mastodon include heavy burdens on admins to address harassment and spam. Open APIs to help with moderation were added to Mastodon in 2019, to help admins and developers build custom tooling for moderation.

Social & Discovery

Mastodon users are presented with three timelines: a home timeline with posts from accounts the user follows, a local timeline with posts from the local instance, and a federated timeline with all public posts that the user's server has received from remote instances. Users essentially have access to posts of people followed by people on their instance. Feeds are chronological, non-algorithmic, and contain no ads.

There is no universal search on Mastodon, as each server monitors a different set of messages. Searching for the same keyword on different instances yields different results. To aid content discovery, Mastodon has public relays which rebroadcast anything sent to it to anyone who subscribes to the pub.

A "Profile Directory" tab allows users to browse recently active profiles or new arrivals from both their own instance or the Fediverse to discover who to follow. Trunk is a community-built tool that helps users find and follow people by category.

Users can select featured hashtags to be displayed on their public profile so people can browse their posts by hashtag.

Privacy & Access Control

Posts can be public, unlisted, private, or direct. Public posts are shown on local, federated, and hashtag timelines. Unlisted posts are not, but can be accessed through the link. Private posts are only visible to followers. Direct posts can only be seen by the people mentioned in them. Users can also hide their network, so the lists of followers and people they're following are not visible. Servers only receive posts that are public or addressed to a person on that server.

Mastodon can be served through Tor as an onion service.

Mastodon is currently adding E2E encryption. Previously, direct messages were unencrypted.

Monetization

Federated social networks require both hosting and development costs to maintain. Each instance is funded by its own administrator and community. Mastodon’s development is funded through a Patreon run by the main developer, Eugen Rochko. It currently brings in about 70k a year, which supports him working on it full time, and covers hosting costs and a moderation team for the popular mastodon.social instance.

User Experience

Mastodon's main initial selling point was its familiar interface that behaved like TweetDeck. Prior to Mastodon, projects like GNU social and Diaspora tried and failed to get mainstream adoption. Mastodon succeeded in large part because it created a familiar user interface that looked and behaved like Twitter, allowing users who were dissatisfied to find a new place to land with minimum effort.

Notable design choices in Mastodon that differ from Twitter: Instead of a 280 character limit, there is a 500 character limit. "Likes" are not broadcast to third parties. "Retweet" and "Like" numbers are not shown until a post is clicked on. If a user with an unlocked account gets a follow request from an account that has been

silenced by the server's moderators (either manually or at their domain), the user will get a follow request instead of automatically allowing the new account to follow.

User-level content controls include: Users can set content warnings on their own posts. Filtering on posts can automatically hide keywords and phrases. "Boosts" (retweets) from someone can be hidden. Accounts can be muted or blocked. Entire servers, including all of its posts, can be blocked by a user. Following, muting, blocking, and domain-blocking lists can be imported.

Upload options include: Users can choose where the thumbnail of a picture is focused when opening. Users can enter custom alt-text to image uploads. There is an OCR button to help extract text from an image for the visually impaired. Audio files can be uploaded. Admins can upload custom emojis to their servers.

Interoperability

Mastodon is compatible with all federated applications that use ActivityPub. These include Pleroma, another social application, Pixelfed, a photo-sharing application, and PeerTube, a video-sharing application. Mastodon users can follow a Pleroma, Pixelfed, or PeerTube user from Mastodon.

Mastodon users used to be able to find their Twitter friends using bridge.joinmastodon.org, but the service was shut down after the developer lost access to API keys and was not granted another set.

All Mastodon user data is available for export.

Scalability

Mastodon.social, the instance started by Mastodon's main developer, initially became the server of choice for new users. The server architecture was scaled up several times, until new registrations were closed in 2017. He has now started mastodon.online to support more registrations. Anyone can run a Mastodon instance, but in practice, the majority of users have concentrated within a few.

When a surge of new users joins an instance, server admins can run into scaling issues, as any web host who becomes unexpectedly popular does.

Mastodon hosting providers have emerged as a service to help individuals interested in being admins, but without sysadmin experience, to spin up servers.

A 2019 analysis of the Mastodon ecosystem found that the majority of posts are concentrated on a few instances and outages in 10 instances would remove almost half of all posts from the network.

Metrics

Mastodon has around 2.9 million accounts and 2785 nodes as of June 2020, according to fediverse.network. From Mastodon, 3.8 million accounts can be reached within the Fediverse.

Links

- Official website
- Documentation

SSB Social Applications

This section is a more in-depth examination of the user experience of social applications in the SSB ecosystem.

Desktop applications include Patchwork and mobile applications include Manyverse and Planetary.

Manyverse is the first mobile application for SSB and the first SSB app that implemented DHT invites. Users can connect through upbs, over LAN, through a DHT, or over Bluetooth.

User Experience

Key management is one of the biggest challenges of SSB, as users often lose and forget their passwords. Users are in complete control of their identity. That means if they lose their cryptographic key, they can permanently lose access to their account. To address the problem of key management in a decentralized manner, a project in the SSB ecosystem, Dark Crystal, has implemented a social key recovery system. It splits keys into shards to store with family and friends who can be trusted to help reconstruct a lost key.

The P2P bootstrapping process introduces frictions for new users. First, new users typically join a pub to get connected to the network after they download an SSB application. Then, there is a period of waiting time during the initial sync when logs are being downloaded, like the syncing time of a blockchain. A user that has not opened an SSB application in a while will encounter this synchronization delay again while their node catches up to the state of the network.

The inability to edit or delete content also runs contrary to user expectations. Because of the append-only nature of SSB feeds, there is no ability to permanently delete a piece of content. Applications can work around this by honoring edit or delete messages appended to the feed, but the original content stays in the append-only log that is shared among all nodes, and other applications could choose not to honor such messages. An example of a workaround is ssb-revisions, a basic API that enables applications to use mutable messages by displaying the updated version.

Topics

Data

Data Models

Solid

Solid uses RDF (Resource Description Framework), which is a standard model for data exchange on the web. RDF was adopted as a W3C recommendation in 1999 as a standard for graph-based data, intended for decentralized information sharing. RDF extends the linking structure of the web to use URIs to name the relationship between things, allowing structured data to be shared across different applications. RDF is based on the idea of making statements about resources of the form *subject-predicate-object*, known as triples. It is an abstract model with several serialization formats, so the particular encoding for resources or triples varies.

RDF has all the advantages and generality of structuring information using graphs. It is the foundation for publishing and linking data in the Semantic Web. If it were fully adopted, advantages of using RDF would include being able to have the same API everywhere, as opposed to requiring code to integrate with each custom web API. Disadvantages include being non-human-readable and historically complex for developers to use.

XMPP

XMPP is an XML (Extensible Markup Language) protocol for near real-time messaging. The basic protocol data unit in XMPP is an XML "stanza", a fragment of XML that is sent over an XML stream.

Matrix

Matrix transports messages using JSON, instead of XML like its most similar predecessor, XMPP. In the years since XMPP was standardized, JSON has become a popular alternative to XML because it's less verbose, parses quickly, and is human-readable. Conversation history in Matrix is tracked through DAGs.

ActivityPub

ActivityPub uses streams of JSON-LD, based on the ActivityStreams data format. ActivityPub implementations mostly use a subset of JSON-LD, related to namespacing, versioning, and extensions. JSON-LD allows for extensibility so that ActivityStreams do not have to contain all the vocabulary needed for future applications.

IPFS

IPFS uses a custom data structure, IPLD, which is designed to treat hash-linked data structures as subsets of a unified information space. IPLD is a single namespace for all hash-based protocols, such as git and cryptocurrencies. IPLD objects can be expressed in various serialized formats such as JSON, CBOR, YAML, and XML.

SSB

SSB uses append-only logs of signed JSON. Each SSB user has their own signed hash-based linked list, called a *sigchain*. This design ensures that messages are not lost, the order is not confused, data can be moved across untrusted machines, and replication of data is simple. Drawbacks include: inability to subscribe to only parts of the data, no support for mutable data, and inability to update from multiple machines, resulting in no multi-device usage.

Hypercore

The Dat protocol renamed itself to Hypercore to reflect a transition that placed the core of the protocol at a lower level of abstraction. The primary data structure of Hypercore is a signed append-only log that is P2P distributed and intended as a building block for other applications. The former Dat protocol's main structure, a public-key-addressed file drive, is now implemented as Hyperdrive on top of Hypercore. In contrast to SSB,

which uses a linked-list append-only log, Hypercore uses a Merkle-tree-based append-only log where each entry generates a new leaf and a new root.

Mutability

Federated applications allow users to edit and delete content, as these operations are handled at the server level. To ensure content is deleted across the entire network, applications must honor delete messages.

P2P applications have more variance around mutability, as some data structures can lead to immutable content.

SSB & Hypercore - Messages added to the append-only logs used by SSB and Hypercore are immutable. Applications can choose not to display messages indicated as deleted, but the data cannot be overwritten.

IPFS - Data is content-addressed, so once added to a network, content is discoverable by its hash. If users stop hosting it and no copies are left online, it will be inaccessible. However, if a copy remains stored on the network, it is re-discoverable by its immutable reference hash.

Aether - Aether attempts to design a more ephemeral P2P content network. "Stale" threads that have not been referenced for 6 months get dropped by clients in the network.

Blockchain social applications - Many blockchain social applications do not allow deletion of content, as data posted on a public blockchain is immutable.

Discovery

Being able to discover great content is key to an engaging social network. This section covers methods of content discovery in decentralized social networks.

Decentralized social networks take two main approaches to content discovery: "local is better" or "share everything". "Local is better" applications, such as Mastodon and SSB apps, embrace a small-world approach as part of their design philosophy. They do not attempt to offer universal search or content recommendation, as they are not trying to replicate the experience of centralized social apps. "Share everything" approaches, such as Aether, or applications that treat a blockchain as the database, instead choose to share all of the data among all of the peers.

Curation

Most decentralized social networks use a chronological feed or rely upon upvotes and downvotes to surface content.

Mastodon

Mastodon's feed is chronological. Users are presented with three timelines: a home timeline with posts from accounts the user follows, a local timeline with posts from the local instance, and a federated timeline with all posts that have been retrieved from remote instances. Servers store content from users followed by members of the server.

Mastodon public relays rebroadcast anything sent to it to anyone who subscribes to the pub.

SSB

Content is propagated and discovered through follow relationships in the SSB network. When a follow relationship is initiated, the posts of the user being followed begins to be synced to the follower's node. Those messages and files are stored locally on the user's computer, indefinitely, for applications running SSB to read.

Search

In decentralized networks, whether federated or P2P, there is often no global search functionality as no node has a unified view of the network.

Mastodon

Mastodon has no global search functionality. A GitHub issue discussing global indexing: https://github.com/tootsuite/mastodon/issues/9529

To overcome the difficulties of new users finding people to follow to get connected to the network, a community-built tool, Trunk, exists to help users find and follow people by category. Users have requested a directory for importing friends from other networks. Mastodon users used to be able to find their Twitter friends using bridge.joinmastodon.org, but the service was shut down after Twitter API keys were lost and not reissued.

SSB

A search of the network will yield results from the dataset the user's node has access to.

Decentralized Search Engines

Yacy is a decentralized search engine created in 2003. Users can define their own web index and start a web crawl.

Blockchain-based search engines include Almonit, Presearch, and BitClave.

Governance

Decentralized protocols need governance systems that can make decisions about protocol evolution, moderation, design, and other topics. The purpose of this document is to review how existing projects govern themselves. As all the protocols and applications covered are open source, we will look into general open source governance as well.

ActivityPub

ActivityPub was developed by the Social Web Working Group (SWWG), which ran 2014-2018, and culminated in the recommendation of a set of social protocols including ActivityPub. It has reached W3C recommendation status and is currently stewarded by the W3C.

Matrix

Matrix was initially developed within Amdocs by developers building a chat tool. Amdocs funded development from 2014-2017. In 2017, the core developers started a company, New Vector, which now drives development. The Matrix protocol and specification is stewarded by the Matrix.org Foundation. There is an annual vote of confidence in the project lead.

XMPP

XMPP was developed by the open source Jabber community without funding. XMPP remains non-profit. The IETF created a XMPP working group in 2001 and eventually published RFC 3920 and RFC 3921. In 2007, the Jabber Software Foundation renamed to XSF (XMPP Standards Foundation) with the focus of developing open protocol extensions to the IETF's base XMPP specifications. It has several sponsors, a Board of Directors which oversees the business affairs of the organization, and a council that approves XMPP Extension Protocols. The council is elected by members of the XSF. These members define and implement XMPP extensions for new features. XSF jointly works with IETF to create more RFCs and extend the protocol.

The organizations sponsoring XSF are ProcessOne, Tigase and USSHC.

IPFS

IPFS was built by the company Protocol Labs and continues to be stewarded by it in conjunction with an open source community. The core implementations working group, consisting of both employees of the company and external contributors, has decision-making authority over contributions to the IPFS protocol. Libp2p, IPLD, and Filecoin are stewarded by separate working groups.

SSB

The SSB community has created an Open Collective Consortium that manages donations. It creates an annual budget and allocates funds for bounties, or to a project approved by a monthly rotating "adjudicator". Sunrise Choir is a non-profit company run on donations, which aims to make SSB developer-friendly. The team members are core contributors to SSB.

Blockchain Governance

Most blockchain protocols, like other open source projects, operate through rough consensus, guided by companies and foundations that direct resources towards development. Bitcoin has a lead maintainer, who has oversight over all aspects of the project and coordinates releases. The role has been voluntarily passed along through the years. Companies such as Chaincode Labs and Square Crypto contribute to funding protocol development. The Blockchain Governance Initiative Network recently formed for collaboration on issues.

A few projects have experimented with on-chain governance, in which funding and decision-making is executed in a decentralized manner linked to the blockchain itself. Dash allows masternodes to cast votes for

how to allocate a "treasury", consisting of ten percent of the block awards to pay for projects that benefit Dash. Decred has a similar treasury allocation process and uses a blockchain-anchored proposal system to submit and vote on proposals. DeepDAO provides an updated listing of tokenized DAOs.

Open Source Governance Links and Resources

- How the Node.js Foundation Utilizes Participatory Governance to Build Its Community
- Debian Constitution
- Roads and Bridges
- Governance without Foundations
- Apache Foundation Governance
- Wikipedia Original Statement of Principles
- Gitcoin - bounties for git issues and grants for ecosystem projects

Identity

Centralized identities are administered and controlled by a single authority. Centralized social networks offer users identities that are administered and controlled by the service. Decentralized social networks offer forms of identity that give users varying degrees of control. Decentralized identities may be federated, user-centric, or self-sovereign.

We will call entities with identities "actors", because non-human entities such as companies, organizations, and bots may have identities on a social network.

Identity allows an actor to:

- control an account and access private data
- communicate with another actor
- establish visible reputation and credibility

Desirable qualities for decentralized identities:

- Allow authentication and migration between services
- Allow communication across services
- Unique, global, and memorable

Decentralized Identity

OpenID was originally developed by the creator of LiveJournal in 2005 as a decentralized identity system. The goal was for users to own their identity and to make logins easier, but the protocol has since mostly been abandoned due to its complexity and poor user experience.

OAuth is currently the most successful identity standard. OAuth was created as an authorization protocol to securely transfer user credentials from one site to another. OpenID Connect is an authentication layer built on top of OAuth 2.0. OAuth allows different identity providers, but in practice identity providers became centralized because users could not run their own identity provider, or manage to choose among a long list of identity providers.

Identity in federated applications:

Email is the most successful federated social application. As a result, many user identifiers in federated applications look similar to email addresses. Federated identity systems rely on DNS.

- Diaspora - User identities in Diaspora are tied to their pod and cannot be migrated. Diaspora uses the WebFinger protocol to discover users from other pods. User information is returned via hCard, an open microformat standard for identity.
- Mastodon - User identities in Mastodon are the username followed by the instance the user belongs to: @alice@mastodon.social.
- Matrix - User identity in Matrix is a username followed by the homeserver: @bob:matrix.org.
- Solid - Solid uses WebID URIs as universal usernames. The WebID URI's function is to point to the location of a public WebID Profile document: https://alice.databox.com/profile/card#me.
- XMPP - User identity in XMPP is a username followed by the homeserver: alice@example.com.

Identity in P2P applications:

P2P systems that put identity entirely in the hands of users must deal with key management, key verification, and key backup. Account recovery is usually not possible because there is no third party to recover an identity if a user loses their password or key. The following P2P identity systems are independent of DNS.

- Aether - Identities in Aether are keypairs. Users can choose a custom nickname, but it is not unique. Multi-device usage is possible but difficult and requires manually porting a user config file across devices.

- GUN - GUN's user system creates a username and password, with usernames global but not unique. Multi-device login is handled by encrypting a user's cryptographic keypair, which is stored in the GUN graph. Keypairs are not derived from the password. PBKDF2 proof is derived from the password and AES keys are derived from that to encrypt the keypair. GUN treats this method as "secure enough" for applications in which private keys do not control financial information. "Auth" is doing a GUN query for that account, subscribing to it, and then attempting to brute force decrypt the keys of all accounts that match that username. Once an account has been loaded once, it's cached on that device, loading from local storage or the local hard drive.
- Peergos - Peergos users are identified by unique usernames linked to public keys. The uniqueness of usernames is ensured through a global append-only log for public key to username mappings that is mirrored on every node in the Peergos system. Names are taken on a first come first served basis. Currently, a single server determines the canonical state of this log and other nodes sync to it. Long-term considerations include decentralizing the name server through a blockchain architecture. Peergos allows multi-device login through a password-based interface. A user's private keys are derived every time they log in using their username, password and a published salt.
- SSB - SSB user identities are cryptographic keypairs, stored locally. Multi-device login is not possible because keys are stored on user devices. Users can pick a human-readable nickname that is associated with their key, but nicknames are not unique because there is no global registry.

Blockchain Identity Systems

In 2001, Zooko Wilcox-O'Hearn named three desirable properties of decentralized network identifiers: human-meaningful (memorable), decentralized (global), and secure (unique). This became known as Zooko's triangle. Prior to the invention of cryptocurrency blockchains, which enabled decentralized global consensus, it was thought that only two of these three properties could be achieved at one time. Now, many projects have created blockchain-based protocols for naming systems that fulfill all three properties.

- Namecoin - One of the first forks of Bitcoin to create a blockchain with an alternative use case, Namecoin functions as a blockchain-based key/value pair registration and transfer system. The Namecoin chain is used for NameId, which combines identities on Namecoin with OpenID, and Dot-Bit DNS, which distributes a DNS registry over the network.
- ENS - The Ethereum Name Service gives users a .eth domain associated with an Ethereum address, or allows them to manage DNS names they already own. It is managed by a smart contract on the Ethereum blockchain. Names are allocated through an auction process.
- Blockstack - Blockstack originally built a DNS system for decentralized app development on the Bitcoin blockchain and later migrated to a custom blockchain with smart contract programming functionality.
- Handshake - Handshake is a blockchain designed for allocating ownership rights to top level domains through an auction process. It uses a UTXO model, like Bitcoin, and implements smart contract programming functionality on top.

Decentralized Identifiers (DIDs)

DIDs are a new type of globally unique identifier that do not require a centralized registration authority and can serve as a decentralized public key infrastructure. The concept is being formalized into an emerging W3C standard. The term "DIDs" was originally coined by the W3C Verifiable Claims Task Force in the spring of 2016. The groups researching decentralized identity that converged on the concept of DIDs were inspired by the potential of using blockchains as decentralized, yet global identity registrars.

DIDs aspire to be a self-sovereign identity. They differ from other globally unique identifiers in that they are globally resolvable, decentralized, and cryptographically verifiable. DIDs require a global key-value database in which the database is a blockchain, distributed ledger, or decentralized network.

The format of a DID is: a scheme identifier, followed by the DID method, followed by a method-specific identifier. A simple example: did:example:123456789abcdefghi

DIDs point to DID documents. Entities holding DIDs may be authenticated via proofs or verifiable credentials.

As of 2020, a Peer DID Method Specification is under development, which does not require any central source of truth and is suitable for private relationships.

DID Implementations

DID implementations can store DID documents directly on the blockchain, construct them dynamically based on a blockchain record, or store a pointer on the blockchain to a document in a decentralized storage network like IPFS or STORJ.

There are DID implementations, but few applications, as it is still new and untested. The current biggest user of DIDs are applications using 3Box, as 3Box creates a DID for the user that is associated with their Ethereum address.

- 3ID/Ceramic - 3ID is an identity system that links a user's Ethereum address to a DID. They are in the process of migrating to a blockchain-agnostic DID network called Ceramic.
- Sovrin - Sovrin is a permissioned blockchain identity network that implements DIDs. Consensus in the Sovrin network is maintained by approved validator nodes.
- uPort - uPort is a DID implementation built on Ethereum.
- ION is a Microsoft-led DID system. It is an implementation of Sidetree, a blockchain-agnostic DPKI protocol, that runs on Bitcoin. It stores transaction data in IPFS.
- IBM - IBM is helping to create, operate and maintain permissioned decentralized identity networks that implement DIDs built using Hyperledger, IBM's permissioned ledger.

Key Management

Systems that place identities fully in the hands of users, such as P2P systems, blockchain identity systems, and DIDs, encounter the problem of key management. Providing a key management method that is secure yet convenient for users is a major design challenge. Users commonly lose and forget both passwords and cryptographic keys.

The increasing popularity of cryptocurrencies has created new solutions for secure private key management. The most secure solutions, such as hardware wallets and third party custody services, are appropriate for high stakes keypairs that may control large amounts of money but not suitable for social applications that are accessed more frequently and casually. Web wallets, such as the MetaMask browser extension for Ethereum, provide a more usable solution for decentralized applications. Many decentralized applications built on Ethereum perform authentication through MetaMask. In the long term, a better interface for decentralized applications would rely on key management being handled by the user's browser. New browsers with support for cryptocurrency, such as Brave, handle key management for multiple wallets natively in the browser.

To recover lost keys, users must turn to a third party. Applications striving for decentralization have attempted to split that trust across multiple parties. Social recovery systems give users a process to place key backups in the hands of trusted friends and family. Some examples of social key recovery implementations include Dark Crystal, a user interface for splitting keys into shards that are shared with trusted friends and family, and Argent, an Ethereum wallet that allows users to back up wallets among "Guardians", which can be trusted people, devices, or third party services.

The decentralized app ecosystem has also attempted to come up with usability improvements beyond password-based authentication. Torus is a key management system that allows users to use OAuth with existing user accounts to authenticate with decentralized applications. It uses a Distributed Key Generation protocol and distributes key shards across a network of nodes running a private BFT network, which sends the shards back to be reassembled when the user authenticates. Magic Link is a service that provides an SDK for applications to easily build email "magic link" logins compatible with private keys. On the backend, it uses DID tokens and delegates key storage to an AWS HSM.

Reputation & Trust

Reputation in decentralized networks is established using many of the same mechanisms as reputation in centralized networks: ratings, peer connections, and metrics such as follower counts. Reputation systems in decentralized networks also suffer from sybil attacks, spam, and impersonation, addressed below.

Failure modes

- Sybils and spam - Spam and the creation of many fake users to carry out attacks or misinformation campaigns are problems for existing centralized social networks. These problems are also present in decentralized networks and approaches to combat them are still evolving. Federated architectures allow server administrators to intervene and block or filter malicious accounts. However, ongoing harassment and abuse through sockpuppet accounts in Mastodon has motivated research into directions such as OcapPub, an object-capability based version of ActivityPub. Steemit, a blockchain social network, requires new user registrations to be approved by a centralized service in order to combat the problem of fake accounts created to rig the voting system that determines monetary rewards for posts. Aether requires a hash computation to be performed for every event posted, raising the computational power required to mass spam the network.
- Impersonation - Attempts to impersonate users for purposes of fraud or defamation are widespread on centralized social networks. This threat also exists in decentralized social networks. Networks that do not have a unique human-readable identifier, such as SSB, are likely more vulnerable to this form of attack, as comparison of public keys is a complex behavior that users are not familiar with. In federated naming systems that include a server address in the username, the full address including the server name must be displayed to users in cases where there may be possible collisions.

Links

- OAuth
- Decentralizing the Social Web
- What are Decentralized Identifiers
- DIDs
- DID Primer
- Rebooting the Web of Trust Papers
- DKMS

Moderation

One of the most acute problems with centralized platforms is the need to develop one-size-fits-all moderation policies for billions of users. Decentralizing moderation puts decisions about what content to block or allow in the hands of users and communities. Community-level moderation can happen through servers, chatrooms, or topic-based communities. User level moderation can allow users to opt into different content preferences, or control their interactions with other users.

Matrix

Moderation in Matrix happens at three levels: Server administrators, chatroom moderators, and users.

Servers in Matrix have terms of use that users agree to when they join. Chatroom moderators can remove people from individual rooms. Users themselves can filter out content they don't want to see.

Most moderation takes place at the room level. Rooms have moderators who can remove undesirable content. A redaction, or “remove message”, can request all participating servers to remove a message. Users can also remove their own messages. However, due to the decentralized nature of Matrix, if the client does not remove the redaction, the message will not be removed. Moderators can kick or ban users from rooms. They can also ban servers from rooms, in cases where malicious traffic must be avoided.

Users in a room have “power levels”, a number between 0 and 100 that indicates how much power the user has in the room. A higher level has more permissions. Moderators have ability to change power level permissions. Users can report abusive content to the server admin, who has the ability to remove it. Users can also block other users to prevent them from contacting them.

Mastodon

Moderation rules in Mastodon are local to each server. Each server admin can create their own moderation rules as well as a theme for their server. Their TOS may include rules about whether data can leave the server, etc. Users choose which server to join, opting into the moderation policy, theme, and TOS they prefer. Moderation actions can be applied to individuals or entire instances.

Community created guides for new Mastodon moderators provide a sample of problems faced and tools available to address them. Some documented challenges with moderation in Mastodon include heavy burdens on admins and moderators to address harassment and spam. Open APIs to help with moderation were added to Mastodon in 2019, to help admin and developers build custom tooling for moderation.

Reddit

Reddit is a centralized social platform, but takes a decentralized approach to moderation by allowing moderators of individual subreddits to make the majority of moderation decisions. The creator of a subreddit becomes the head moderator and can delegate to other moderators. Reddit admins are employees of the company with "supreme mod" authority over content on the site.

To help community moderators, Reddit has developed an automated tool, AutoModerator, to help proactively identify, filter, and remove objectionable content. Many moderators also create or use custom moderation bots.

Notabug, a decentralized Reddit alternative, has a different moderation structure. Posts are created in any "topic", without requiring a subreddit for a topic to be created beforehand. "Spaces" are more like subreddits and can accept or block content, but they do not affect the original posts in the topics. A given topic can have multiple spaces which curate and moderate differently. The automated rules for how spaces filter posts can be publicly inspected.

Aether

Each community has its own moderators, which communities elect or impeach themselves. Like subreddits, communities largely govern themselves. Mods (moderators) can approve or delete posts.

Users can report posts. Reports go to mods of the community. Users elect moderators in communities and can impeach them. If a user votes to impeach a mod, the mod's actions will be ineffective for that user, even if the vote wasn't won. All mod decisions are public history.

Moderator elections begin once a community has more than 100 active users within the last 2 weeks. Voting thresholds for moderators is a 51% positive vote of at least 5% of the community's members. To prevent hostile takeovers, the population of communities is determined on a 2-week trailing basis and mods can temporarily lock a community to new entrants.

Users who wish to be mods can enable “mod mode”, which allows them to proactively attempt to approve or delete posts. Their moderation history can be observed by others considering electing them as an official mod.

SSB

Moderation in SSB depends on a bottom-up approach driven by users. In the SSB ecosystem, users block, flag, or ignore other users that they find objectionable. An ignore will simply not show that data to the user's node, although their node will continue to pass their data through the network. A block will cause the user's node to refuse to replicate data from that feed, segmenting it off from their portion of the network. If enough people block a user or group of users, their part of the network will become partitioned from the rest.

Steemit

Steemit takes a bottom-up approach to moderation. Content is moderated (https://steemit.com/steem-standards/@arhag/moderation-standard) through the up and down votes of users, instead of through the actions of a moderator. Users can earn a curation reward for upvoting posts that later become popular. Low voted content will earn less rewards and may be hidden. Votes are weighted by reputation, which accumulates with age, so older accounts of early adopters have more power in the network.

Steemit's approach to spam, plagiarism, and abuse relies on a single mechanism: user voting to downgrade undesirable posts. To add a negative signal to a post and downgrade its rewards, users could "flag" or "downvote" - two terms used for the same function of downgrading a post. A discussion of the downvoting and flagging problems on Steemit goes into the drawbacks of various approaches.

Users can flag content they find objectionable, but they cannot block other users due to the potential for abuse of the block feature in a system that monetizes upvotes. For example, a user might block users with high reputation who might downvote them, so they could upvote their own posts within a circle of participating accounts.

Monetization & Business Models

Open protocols allow any application to build on them. Open access creates competition and makes it harder for any single application to establish a sustainable business model. Many decentralized projects run on volunteer work and donations. This funding method will not be covered. This document will explore examples of decentralized projects that have succeeded or failed to monetize, as well as emerging and experimental monetization possibilities.

Three layers of business models can be considered: application level, provider level, and protocol level. Application level business models for decentralized social applications are similar to current business models in centralized social applications. Providers are servers (in federated systems) or supernodes (in P2P systems) that provide capacity to the network and may charge for their services. Protocol level monetization was not a proven business model until cryptocurrencies came into existence and it is still at an experimental stage.

Application Level

Protocols must be open source in order to be used and adopted. Applications built on top of open protocols do not necessarily have to be open source. Gmail is an example of a highly successful application built on top of open, federated email protocols. Every popular centralized web application is built on top of the open protocols of the web. Because of this, many of the business models for decentralized applications are already familiar.

Advertising

- Charge advertisers for user impressions or clicks through the applications.
- Users can pay to not be shown ads.

In-app purchases

- Charge for promoted tweets.
- Charge for custom curation or moderation services, or take a cut of monetized services offered by third parties.

Transaction fees on

- User monetization of premium content.
- User tips and donations.

Provider Level

If applications access user data through a provider (as in federated systems where a user signs up to a server or in Solid, where applications access user data through pods), the provider can have a separate business model from the application.

- Charge a commission of the revenues of each application.
- Charge users a membership fee. (However, upfront membership fees for users tend to discourage adoption of social applications, where users have come to expect free applications.)
- Charge users for premium features like extra storage.

Protocol Level

Protocol level business models have been explored in recent years through cryptocurrencies.

One method of protocol-level monetization is the creation of a token used for transactions internal to the protocol. Facebook's Libra, which will allow users to send payments to each other, is an example of this

approach. Brave browser created BAT, Basic Attention Token, for transactions between publishers, advertisers and users. Advertisers pay in BAT to place ads. Publishers receive most of the BAT from ad revenue and Brave takes a percentage. Users of Brave browser earn BAT when they view ads. They can't withdraw it and instead can only donate it to publishers of their choice. According to Brave's research, in 2020, users could earn up to $200 by consuming ads. Publishers haven't been vocal about their earnings, but freecodecamp has said that they earned $2000 between early 2018 to mid 2019.

Existing cryptocurrencies can also be used for protocol-level business models. Brave originally used Bitcoin instead of BAT for in-browser micropayments. In a server-based federated system, servers that provide services to the network could also function as Lightning hubs that route Bitcoin payments through the social graph using payment channels and collect fees for doing so.

Namespaces are a limited resource across a common protocol. For this reason, business models could be developed around username registrations, like how domain names are sold on the web. Currently, Twitter prohibits the trading of usernames, but a black market has emerged anyways, illustrating the latent demand for good names. Legitimizing a username marketplace could be one method of monetization for a decentralized Twitter.

Links

- Decentralized Protocol Monetization and Forks
- The Golden Age of Open Protocols

Network Structure

Federated Networks

Federated approaches to decentralized network architectures can be categorized as federation where you pass messages between systems (SMTP, XMPP, ActivityPub), and federation where you replicate data between systems (Git, Matrix, NNTP).

Passing messages between systems

Federated networks that pass messages between systems include SMTP (email), XMPP (chat), and ActivityPub (Mastodon). This is a straightforward approach for systems mainly concerned with one-to-one communications, such as email and chat, because messages do not get replicated across servers that do not concern them.

ActivityPub, a federated social protocol that passes messages between participating servers, is used in one-to-many social applications like Mastodon, a decentralized Twitter alternative. Unlike Twitter, there is no global shared state across all servers so there is no way to browse all public posts. The federated timeline shows public posts that the user's server knows about. The majority of posts surfaced are from people other users of the server follow.

Delta Chat is a chat application that uses email servers to send messages, via Push-IMAP.

Replicating data between systems

Matrix is a federated protocol that replicates data between systems. Messages in Matrix are replicated over all the servers whose users are participating in a given conversation - similarly to how commits are replicated between Git repositories. The conversation history of the room being replicated across servers allows nodes to resync to the conversation if they go down. To ensure privacy of conversations in rooms, since those conversations are stored across many servers, Matrix has prioritized E2E encryption.

If bluesky adopts a federated architecture, further work should be done to assess how message passing and message replicating systems affect scalability. Certain types of messages, such as popular topics, could be widely replicated to ensure global availability, and others, such as messages between two users, could be shared only with participating servers.

P2P Networks

P2P networks can be analyzed on a spectrum of fully distributed and P2P, to partially distributed. Many P2P systems have supernode architectures and points of logical centralization.

SSB is a fully distributed P2P social protocol that avoids logical centralization and singletons as a design experiment and matter of principle. SSB was designed for social applications, where data can be passed along follow relationships among users. Data is propagated through a gossip protocol among people who follow each other. As part of its commitment to full decentralization, SSB has no DHT, no universal human-readable namespace, and comes with no bootstrap nodes. New users seeking to connect to the network must often use a pub server, which is a peer that is always online, has a public IP address, and offers invite codes to new users.

In IPFS and Hypercore, the focus is on content and not social relationships, so there are central directories to aid in discoverability of content and nodes that guarantee persistence of content if the original host is offline. “Gateways” are essentially supernodes and DHTs used for discovery of content are points of logical centralization.

Content in IPFS and Hypercore is accessible over HTTP through gateways. These are nodes that run the P2P protocol and allow HTTP requests to access content in the P2P network through it. Some of these gateways, like the one provided by the IPFS developers themselves, are quite large and become points of centralization. However, anyone can configure a gateway and run one themselves as the system is permissionless.

DHTs, distributed hash tables, keep track of who has what data in the network. The data itself may be spread out across many different nodes, but the DHT serves as the central index of where it is.

Hybrid networks

Hybrid federated/P2P approaches also exist, such as a client-server architecture where you can optionally run the server client-side.

Matrix's new P2P implementation takes a hybrid approach, moving away from a strictly federated model towards P2P functionality. The new P2P Matrix client essentially runs the server client-side, allowing it to participate in the existing federated Matrix ecosystem as a P2P node.

Light clients in blockchain architectures could also be considered an example of a hybrid approach to P2P network participation. Full nodes participate fully in the blockchain network, storing the entire history and validating all blocks and transactions as they come in. Light clients do not store the full history of the chain, which can be many GBs, but request the relevant history from a full node. This allows light clients to transact without having to store the entire history themselves.

Privacy

Designing for public communication requires less focus on privacy than social applications designed for close social circles. However, privacy is still important to consider on several counts: protecting user metadata, respecting private account settings, and supporting private direct messages.

User Metadata

At a large enough scale, user metadata collected by federated applications becomes a cause for privacy concerns. Examples of these kinds of concerns can be found in this privacy report on Matrix, conducted by a privacy-focused nonprofit, and this response.

Private Accounts

Mastodon and Matrix provide private accounts, where the account can be located, but the data posted by the account is only shown to approved followers.

Mastodon has account-level and post-level privacy controls. When an account is locked, follow requests must be approved. Since posts are copied to the instances of followers, locking an account gives a user more control over where their posts will be distributed. Individual posts, as well as the default post setting, can be set to "followers-only".

Matrix has private rooms, which can be joined upon invitation. Users can also "knock" to request to join a room.

Direct Messages

Many decentralized social applications use E2E encryption to preserve the privacy of direct messages.

- Matrix - End-to-end encryption guide for Matrix clients.
- ActivityPub - Mastodon is adding E2E encryption to ActivityPub. Previously, messages were unencrypted on the server.
- SSB - SSB, as a P2P protocol, included E2E encryption for direct messages from the start, so that unencrypted messages would not be passed through untrusted peers in the network.

Some more E2E messaging encryption options:

- Noise protocol, used by WhatsApp
- Messaging Layer Security (MLS)

Decentralized Social Applications Focused on Privacy

- Peergos - Peergos provides capability-based access control for files on top of IPFS. Files are kept private. All encryption happens on the client, which could be a native Peergos client or a browser. Data is always encrypted on the servers. Servers do not have access to metadata or sensitive information. Access is controlled through cryptographic capabilities.
- ZeroNet - ZeroNet is a P2P browser built on BitTorrent and Bitcoin, designed with a focus on privacy. Instead of having IP addresses, ZeroNet site addresses are Bitcoin public keys. ZeroMe is a proof-of-concept Twitter-like social network on ZeroNet. Other sites on ZeroNet include ZeroTalk (like Reddit), ZeroBlog (microblogging), and ZeroMail (encrypted mail).
- Freenet - ZeroNet was preceded by Freenet, a privacy-preserving P2P overlay network. In Freenet, all data is encrypted and communication is routed through peers, similar to Tor. It cannot be used to access the web; it only allows access to content that has been inserted into the Freenet network. It has an anonymous microblogging service, Sone. Freenet uses a web-of-trust plugin to help manage spam and moderation in an uncensorable medium.

- Zbay - Zbay is a Slack-like messaging application with monetary transactions, which uses the Zcash blockchain as a database and transaction settlement layer. User identities are Zcash addresses. Usernames are registered by sending a message to an address everyone has a viewing key for, and providing the new user's public key. Private messages can then be sent to the user's address using encrypted transactions.