The Network Subsystem consists of the following engines.

The Future plans subsections contains plans not in the specs yet.

For now this is just a rough overview with lots of context/details missing.

Registry

The Registry stores advertisements that arrive from the network,

such as NodeAdvert

, TopicAdvert

, DomainAdvert

,

which provide the necessary information to connect/subscribe/join these.

Router

The Router is responsible for routing messages between local engines and remote nodes

via a Transport protocol, which is selected by the Router for outgoing connections.

It maintains a session for each remote node that may use multiple transport connections.

Currently this is a simple per-node message sequence number that allows detecting a lost message

when one connection times out and switching to a new one.

Future plans

The Router should be able to support asynchronous encrypted sessions

via an asynchronous key agreement protocol such as X3DH or PQXDH by Signal,

via multiplexed underlying transport connections.

It requires the use of prekeys, which is an input to the DH key exchange.

Such prekeys would be advertised via the NodeAdvert

for the initial connection

and later updated by direct messages.

Once this async session setup is in place,

we can add store-and-forward relays that enable asynchronous communication in the system.

Transport

Transport opens and maintains connections to remote nodes via various transport protocols with different guarantees.

Currently this is done via QUIC and TLS.

Future plans

Add more transport options such as Tor and a mix network,

as well as censorship-resistant obfuscated transport protocols or proxies.

Add local peer discovery protocols for LANs to be able to discover nodes on the local network and connect to them directly.

PubSub Topic

A Pub/Sub topic

has a set of publishers

and subscribers

where the publishers can publish events

,

and the subscribers can read them.

The Topic ID is a public key and allows the topic owner to designate publshiers.

It is backed by a Blocklace data structure

that is a CRDT that stores a causal DAG of events as an append-only log,

and gives a partial order of events.

Future plans

The partial order of events given by the Blocklace can be extended to total order

by an additional protocol that can be enabled for the topic if needed

using the Cordial Miners (or similar/improved) protocol.

Based on the Blocklace CRDT, which is an append-only log,

add more complex CRDT types, such as Set (for the membership protocol).

Add epochs that allow garbage collection of old events that are not relevant anymore.

Add fine-grained permissions for publishers based on the type of operation in the event

(e.g. some publishers may send only ACKs)

Add end-to-end event encryption which allows relays that help forwarding messages but they cannot read them.

This would be a simple per-topic symmetric key to start with.

Storage

The distributed storage protocol stores immutable objects

split up into chunks

in the network.

An object is stored as a Merkle tree of chunks,

where the chunk size is a parameter of each object and usually a few MB in size.

The object ID is derived from the root chunk ID.

Objects may become mutable by sending an updated version to a pub/sub topic.

Objects may expire by specifying an expiry timestamp.

A Storage Commitment signed by a node allows nodes to commit to storing an object for a given time period.

These commitments are then sent to one or more pub/sub topics.

Each node stores a local database of storage commitments it knows about,

which allows building a local index of which object is stored by which nodes in the network.

Future plans

Connect object expiry to pub/sub topics epochs, as another option next to absolute timestamps.

Access control for object requests based on domain membership and/or access control lists.

Domain

Future plans

A Domain has a set of members and optionally a set of public nodes that are advertised in a DomainAdvert

that can be used to join the domain and to send external requests to it.

Join requests are approved or rejecetd based on an authentication protocol specified in the DomainAdvert

,

this can be none for public domains, a shared key, membership certificate, ZK proof, etc.

External requests are used to request information from the domain by non-members

and may require an authentication token.

The domain consists of a set of CRDTs each backed by a pub/sub topic.

Two CRDTs are present in all domains:

1. The set of members:

each member updates its own entry continuously,

and advertises their NodeAdvert

and topic subscriptions in the domain.

The maintenance of the members data structure also serves as a membership protocol,

and the topic subscriptions of each member is used for clustering members in the domain

based on their subscriptions.

1. The index of CRDTs:

The set of top-level CRDTs in the domain is maintained in an index that is also a CRDT.