LIBP2P: A MODULAR PEER-TO-PEER NETWORKING STACK

Crosslink Taipei 2019 🇹🇼

libp2p

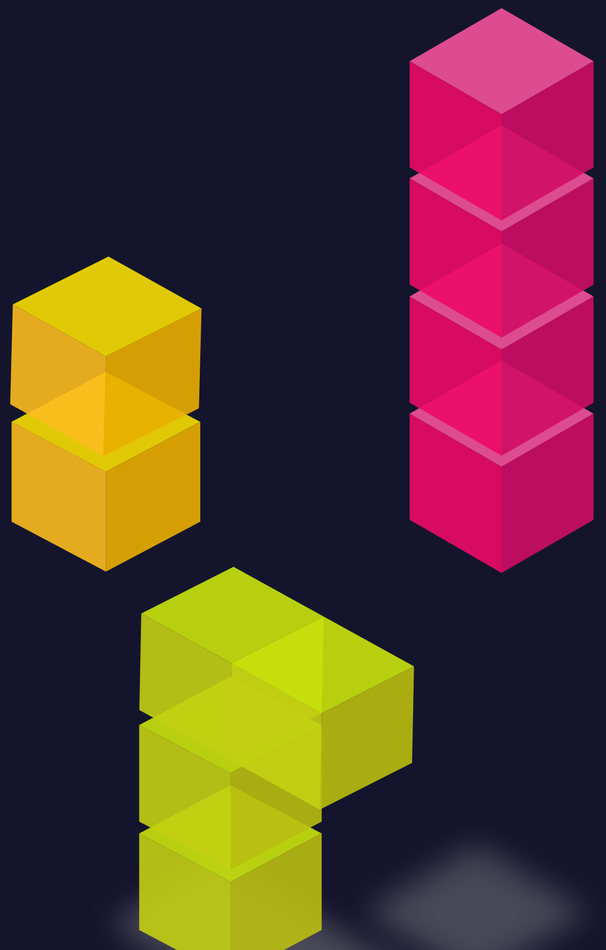
Raúl Kripalani

tech lead, libp2p @ Protocol Labs

🔮 = future

today

AGENDA



- introduction
- addressing and identity
- transports
- connection bootstrapping
- secure channels
- multiplexing
- protocol negotiation
- peer discovery and content routing
- pubsub
- connectivity (NAT, relays)
- what’s next?

A MODULAR PEER-TO-PEER NETWORKING STACK

Composable building blocks to assemble ***future-proof*** **p2p networking layers**.

Runs on a many runtimes: server, browser, mobile (experiments). soon: embedded.

Originated in IPFS. Implemented in 7 languages; bustling cross-project community.

Created, stewarded and sponsored by PL; owned by the community. It's a public good.

Licensed under MIT, soon Permissive License Stack (ASLv2 + MIT).

Transports

Multiplexers

Secure Channels

Peer Discovery

libp2p

Pubsub

NAT Traversal

Peer Routing

Content Routing

ADOPTERS

IPFS

Polkadot

Ethereum 2.0

Filecoin

MetaMask

OpenBazaar

0x

Golem

Harmony

Keep Network

Validity Labs

Enigma

Cosmos (considering)

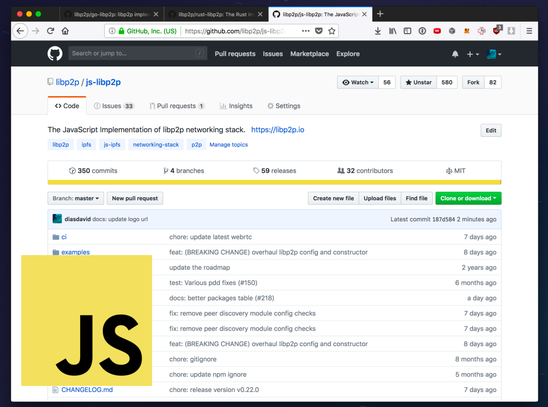
Coda Protocol (considering)

Matrix (experimenting)

... more

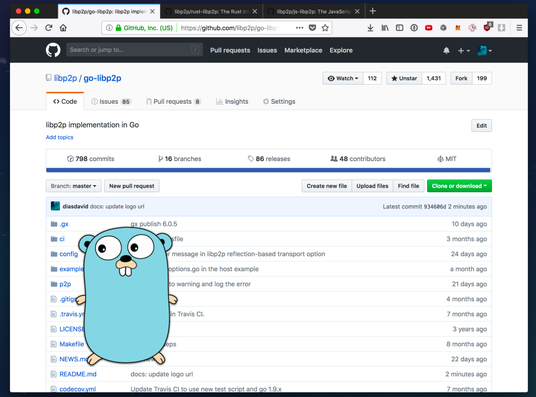
7 native implementations

js-libp2p



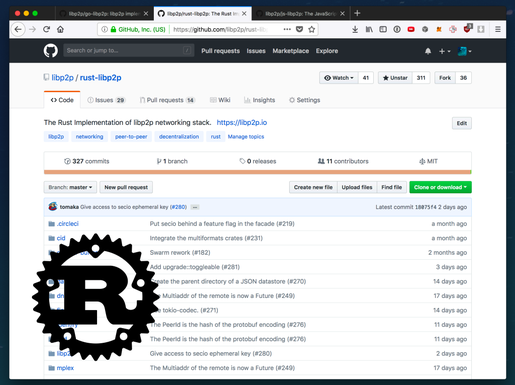
Protocol Labs

go-libp2p



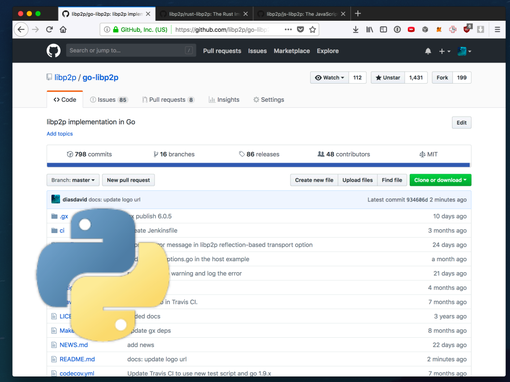
Protocol Labs

rust-libp2p



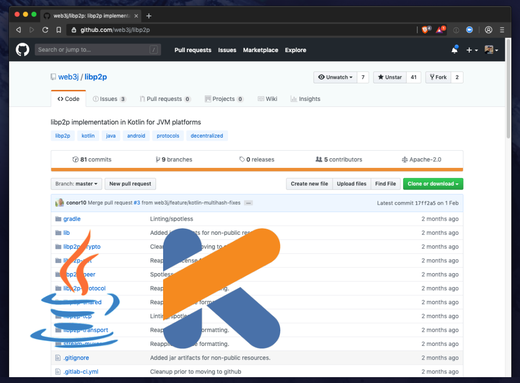
Web3 Foundation

py-libp2p



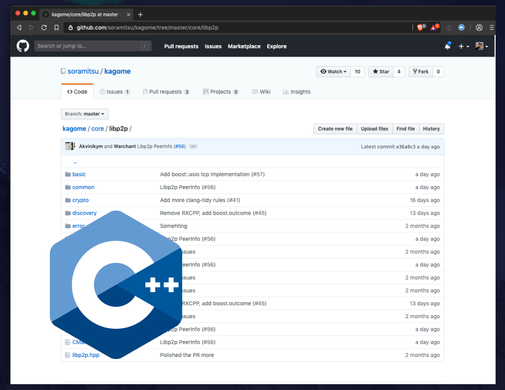
Ethereum Foundation

jvm-libp2p



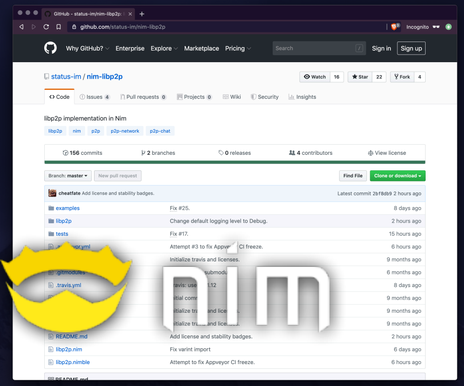
Harmony, PegaSys, Web3 Labs

cpp-libp2p



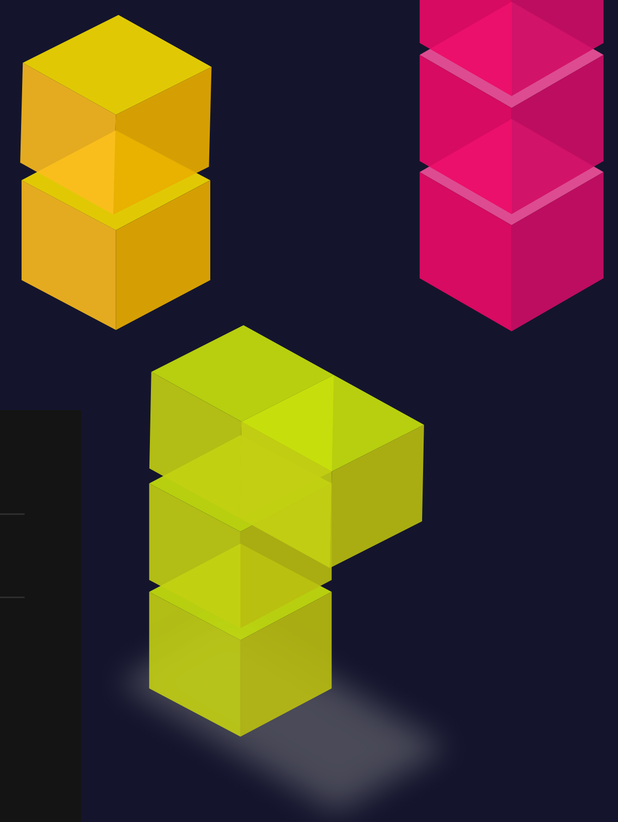
Web3 Foundation, Soramitsu

nim-libp2p



Status

ETH2 NETWORKING

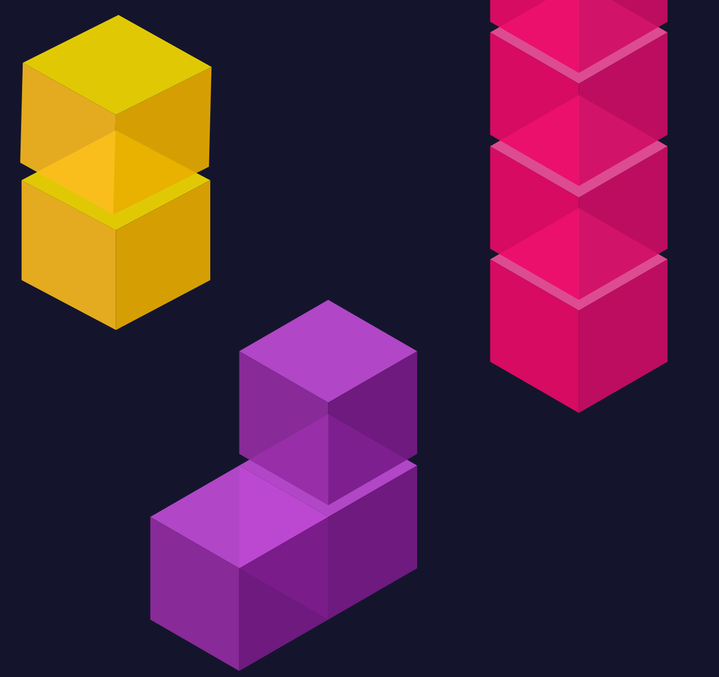


Ethereum 2.0 networking specification

Table of contents

DECENTRALIZED PROCESS ADDRESSING

libp2p's raison d'être is the ability to **locate**, **connect**, **authenticate**, **negotiate** and **interact** efficiently with any process in the world, ***no matter the runtime*** (server, browser, IoT, embedded, etc.) so long as its ***identity is cryptographically derived from its public key***; and have all of that happen in a ***seamless manner*** (e.g. NAT, relay, packet switching), even as those processes **relocate, roam, evolve and mutate *over time***. It is ***juxtaposed to endpoint addressing*** (e.g. IP networks).



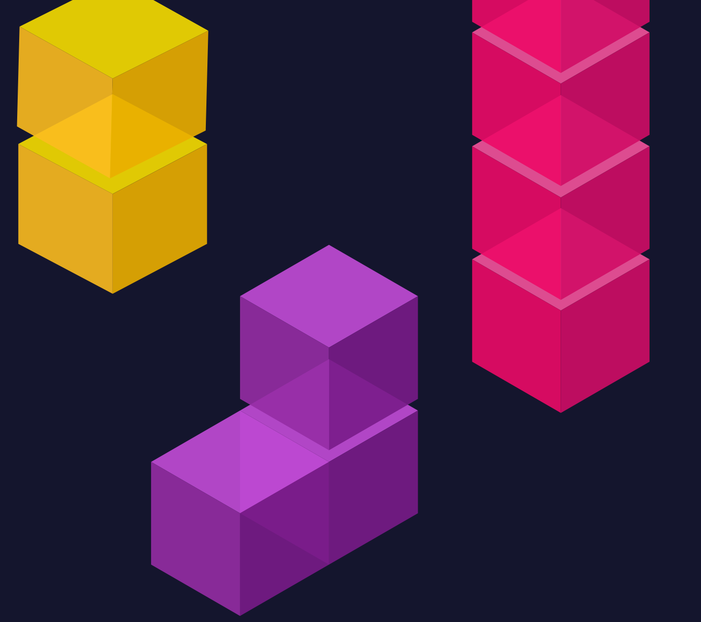
transport-agnostic addresses

MULTIADDRS

`/ip4/104.236.179.241/tcp/4001/p2p/QmPeer…`

`0x0468ecb3f1060fa1 (omitting the id)`

- Composable, future-proof, upgradeable transport-agnostic addresses.
- binary byte-encoded packed format, with a canonical string representation.
- encodes addressing, transport, routing, identity, encryption (future) of a peer.
- flexible varint-based byte[] format vs. hardcoding assumptions.
- what if...?



multiaddrs are inspired by the

PLAN9 UBIQUITOUS FILESYSTEM

everything is a file, including remote network processes

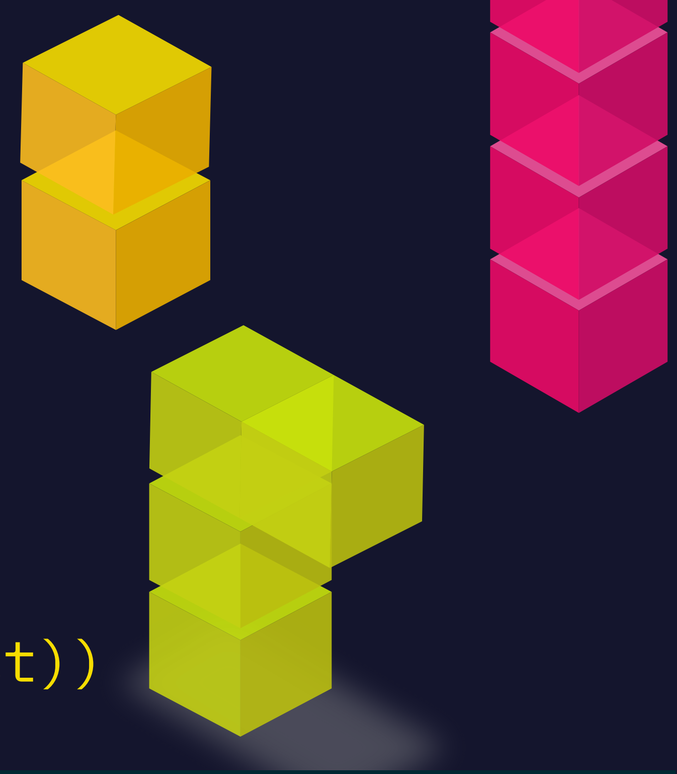
```
/
|_ ip4                  # all processes addressable via IPv4
    |_ 7.7.7.7          # the process lives in network address 7.7.7.7
       |_ udp           # all processes exposing udp sockets
       |_ tcp           # all processes exposing tcp sockets
          |_ 1111       # the process sitting in port 1111
          |_ 1234       # the process sitting in port 1234
             |_ p2p     # all identities exposed by this process
                |_ QmPeerA     # we want to talk to identity QmPeerA
                |_ QmPeerB     # we want to talk to identity QmPeerB
                   |_ p2p-circuit  # relay
                      |_ ... you get it ...
```

http://doc.cat-v.org/plan_9/misc/ubiquitous_fileserver/

PEER IDENTITY

`QmSoLPppuBtQSGwKDZT2M73ULpjvfd3aZ6ha4oFGL1KrGM`

```
pubkey-struct := { <key-type>, <pubkey-bytes> }
peer-id := multihash(SHA-256, protobuf(pubkey-struct))
```



- Peer IDs: derived from the public key.
- Four key types: RSA 👎, Ed25519 👍, SECP256k1, ECDSA.
- Serialize into pb, then multihash with SHA-256.
- String representation: base58btc.
- multiaddrs: `/ip4/1.2.3.4/tcp/5001/p2p/<peer-id>`

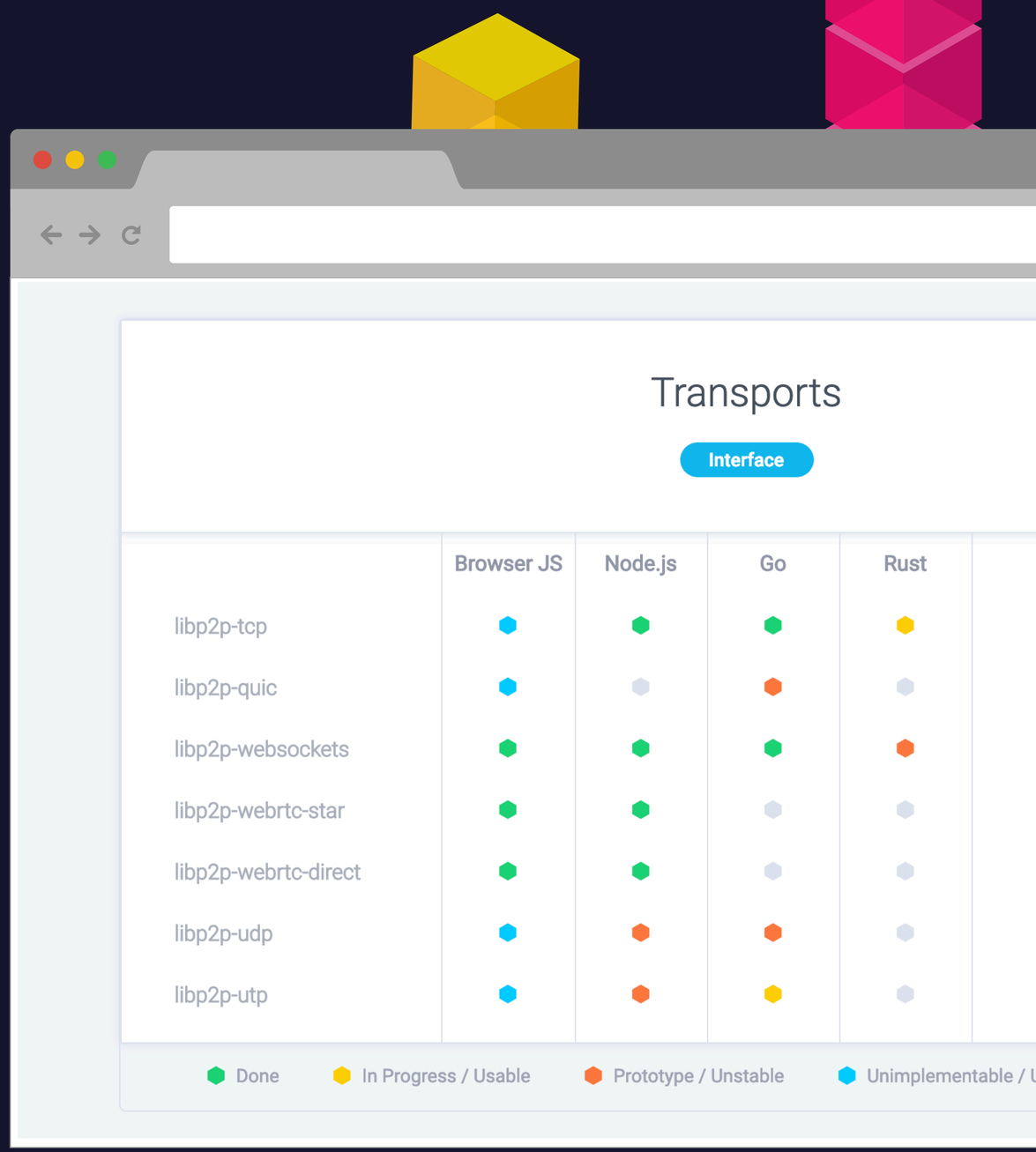
```
enum KeyType {
    RSA = 0;
    Ed25519 = 1;
    Secp256k1 = 2;
    ECDSA = 3;
}

message PublicKey {
    required KeyType Type = 1;
    required bytes Data = 2;
}
```

TRANSPORTS

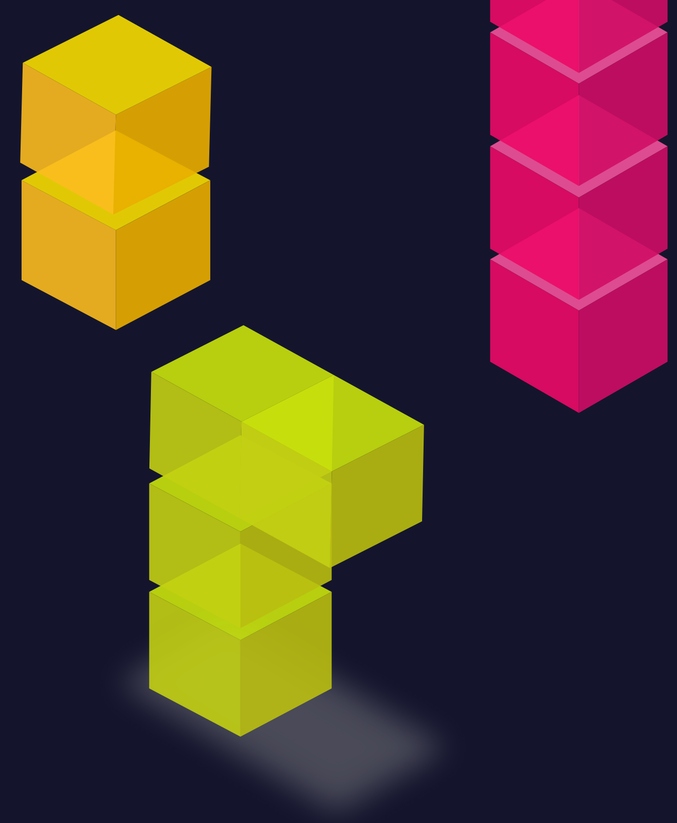
- ***transports*** are a core abstraction of libp2p.
  - Think of transports as *connection factories*.
  - Dialing and listening.
  - They produce raw connections/sockets/sessions – “the L4” of libp2p.

- Current:
  - TCP
  - QUIC
  - WebSockets
  - WebRTC

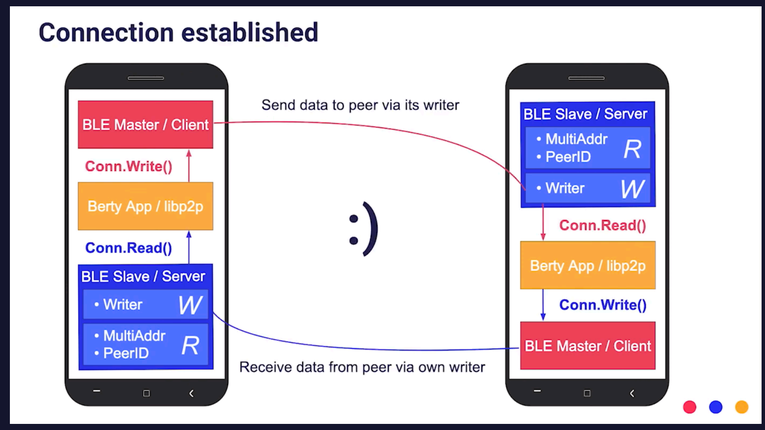


coming soon, to a node near you…

TRANSPORTS



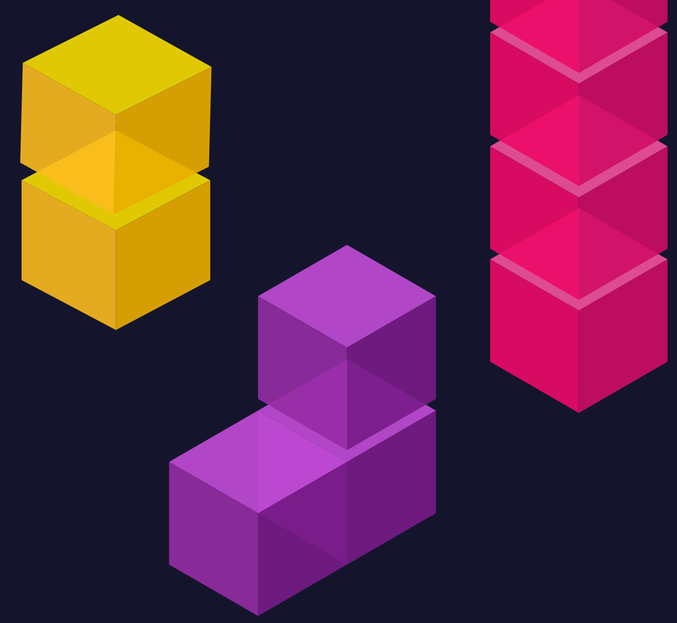
- **Stable, ubiquitous QUIC.**
- **Bluetooth (go, via our friends @ Berty).**
- µTP.
- **Message orientation: UDP.**
- new transports => new multiaddrs
  - graceful upgrade path & backwards compatibility.



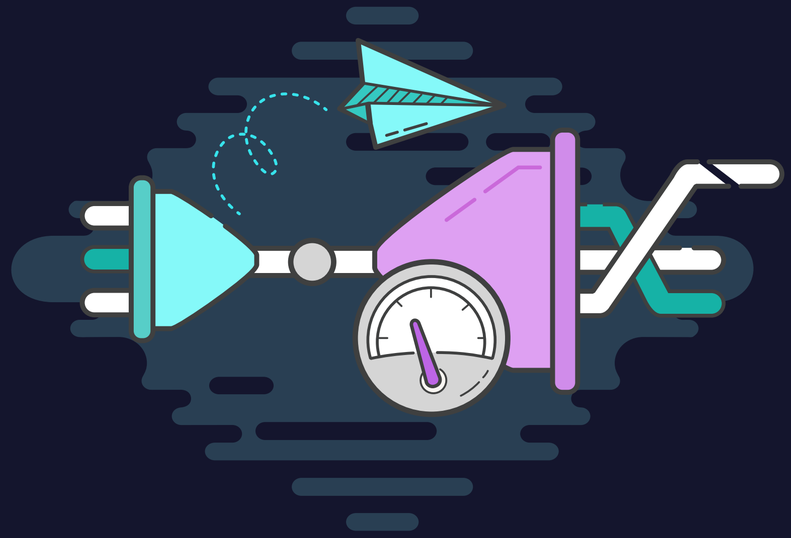
DIALLING...

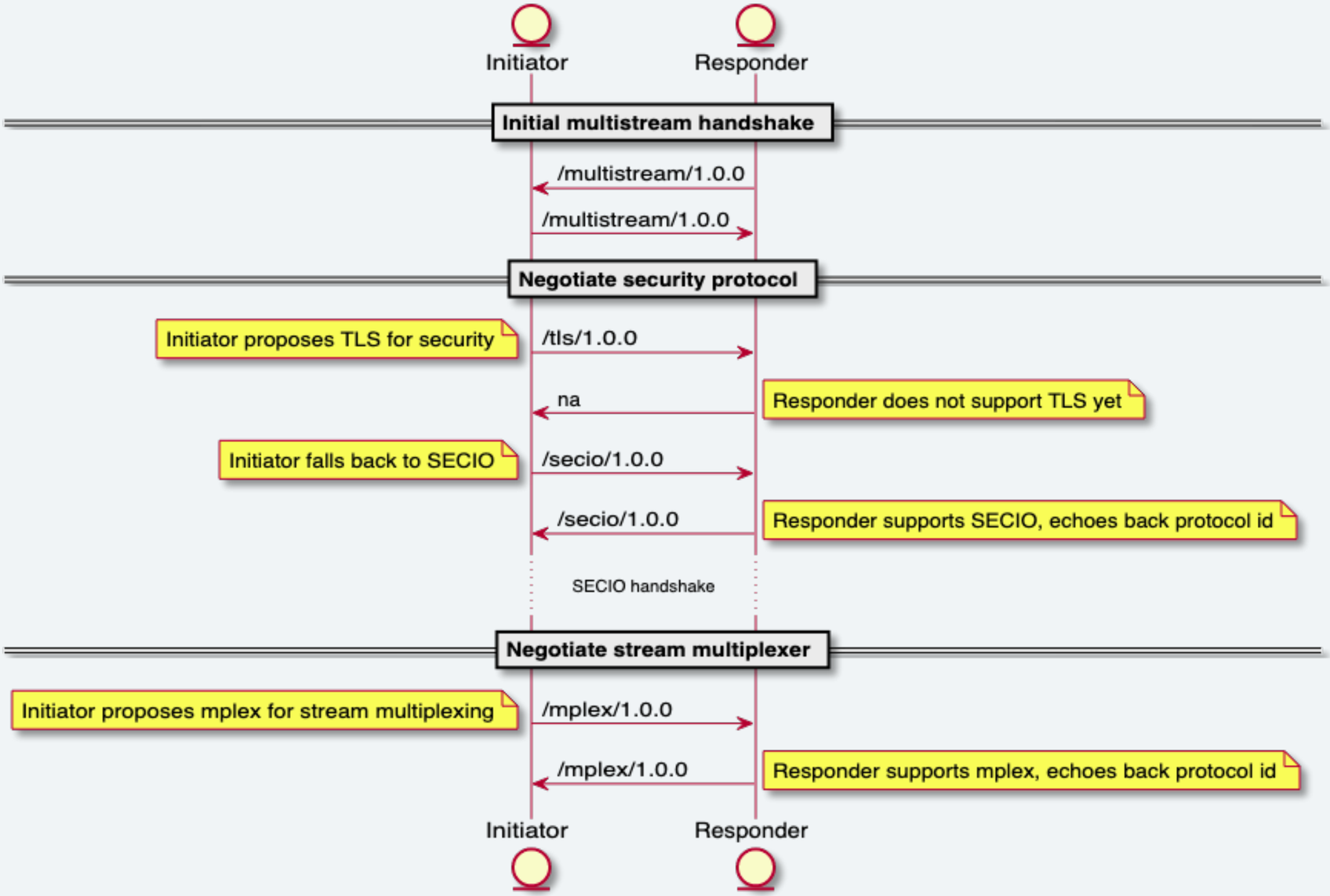
/ip4/1.2.3.4/tcp/4545/ws/id/QmSoLV6aBf…

CONNECTION BOOTSTRAPPING



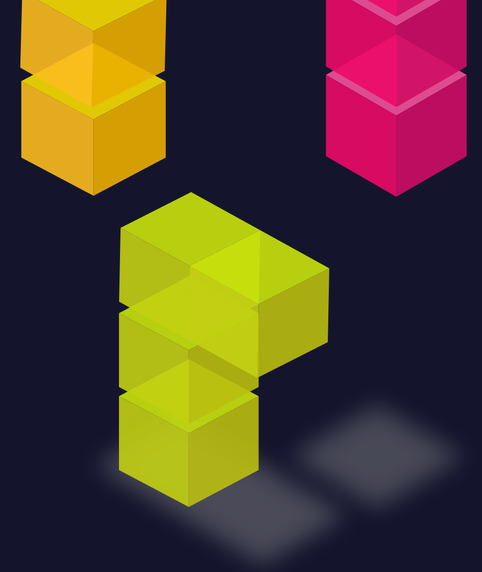
- Connection bootstrapping takes a raw connection, created by a transport, and “*upgrades*” it to a...
- ***Capable connection:*** authenticated, encrypted and multiplexed.
- **We negotiate the secure channel and the multiplexer dynamically.**
  - multistream-select 1.0; more on this later.
- Some transports don’t need *upgrading*.
  - They’re natively capable of encryption and multiplexing.
  - QUIC is an example.



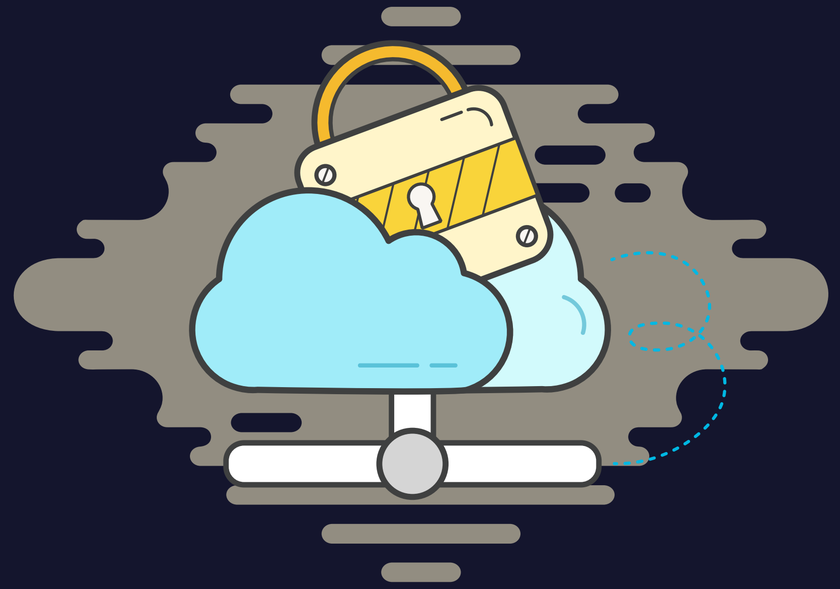


https://github.com/libp2p/specs/blob/master/connections/README.md

SECURE CHANNELS

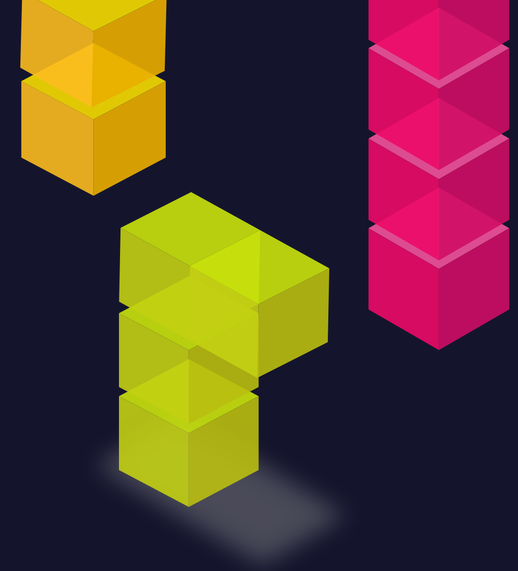


- Peer authentication.
- Transport encryption.
  - Does not preclude you from encrypting/signing application data.
- Transmission integrity. Non-repudiation. HMAC.
- All implementations: SECIO.
  - Some: TLS 1.3.
- For testing:
  - /plaintext/2.0.0. Try it out with Wireshark! 👀



coming soon, to a node near you…

SECURE CHANNELS



- **new secure channels: TLS 1.3 & Noise handshakes.**

- **Noise Handshakes.**
  - Noise Pipes: IK for optimistic scenario, falling back to XX.
  - Encrypted early data to expedite multiplexer negotiation.
  - Required by Ethereum 2.0 mainnet.

- **TLS 1.3 in all languages.**
  - Available in go-libp2p. Can enable as an experiment in IPFS.
  - Roadblock in rust-libp2p, **devgrant** available for js-libp2p!

```
XX:
  -> e
  <- e, ee, s, es
  -> s, se

IK:
  <- s
  ...
  -> e, es, s, ss
  <- e, ee, se

XXfallback:
  -> e
  ...
  <- e, ee, s, es
  -> s, se
```

holding many conversations at once

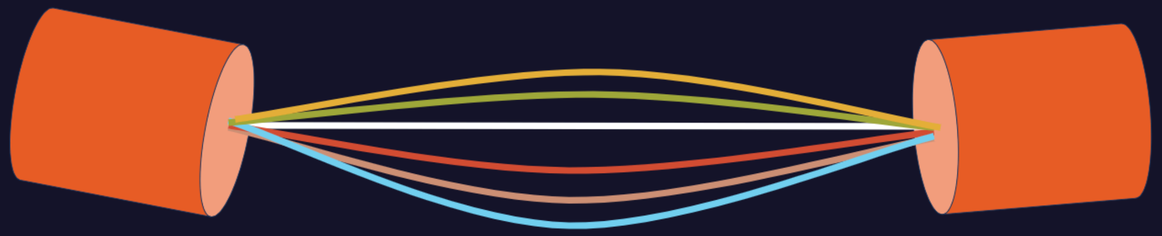
MULTIPLEXING

Establishing a p2p connection may not be cheap or easy (e.g. hole punching, negotiation, handshake, etc.)

Let’s amortise it.

holding many conversations at once

MULTIPLEXING



- yamux (hashicorp)
- mplex (libp2p)
- spdystream
- quic native multiplexing

streams are identified by a number.

how do we contextualise the stream to a particular protocol?

🔮 **future: monoplexing**

multistream-select 1.0

PROTOCOL NEGOTIATION

- multistream-select 1.0 is our current protocol negotiation scheme.
- interactive, chatty, speculative, and wasteful in pessimistic scenarios.

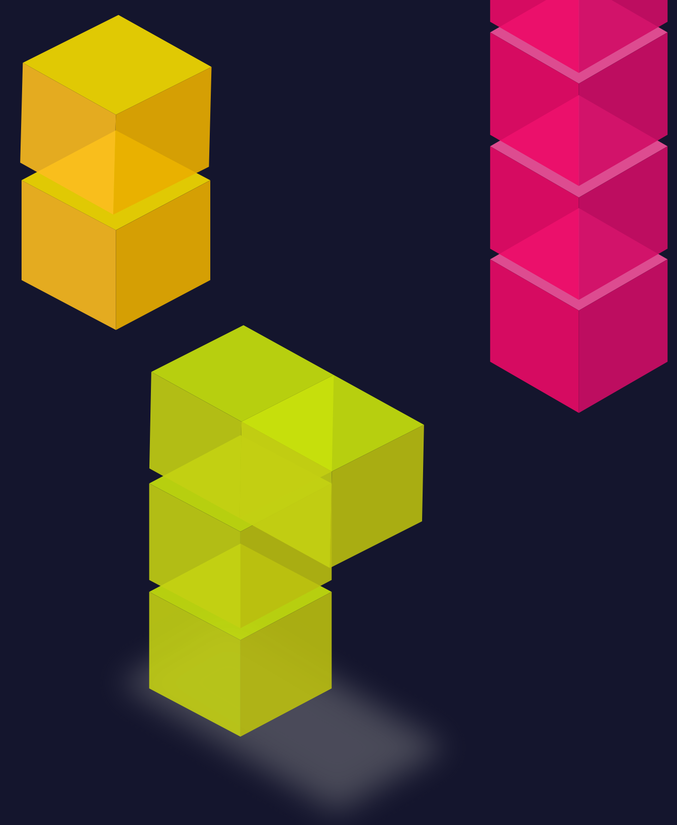
```
>>> /multistream/1.0.0     (1)
<<< /multistream/1.0.0     (2)
>>> /foo/A                 (3)
<<< na                     (4)
>>> /foo/B                 (5)
<<< /foo/B                 (6)
```

towards multiselect 2.0

PROTOCOL NEGOTIATION

- **multiselect 2.0 for efficient protocol negotiation.**
  - https://github.com/libp2p/specs/pull/205, features:
    - 1-RTT connection bootstrapping.
    - cheap repetitive protocol selection.
    - TCP simultaneous connect support.
    - session resumption.
    - upfront protocol tables.
    - confident selection. interoperable with deterministic or probabilistic discovery-level advertisements, e.g. bloom filters.

PROTOCOL HANDLING



- **Protocol IDs:**
  - `/floodsub/1.0.0`
  - `/ipfs/bitswap/1.0.0`
  - `/eth2/beacon_chain/req/...`

- analogous to REST endpoints, they serve for routing and handling.
  - can represent an entire protocol, or a message type.
  - to attach a new protocol, you mount a handler.

- semver under question — we'll probably migrate away.

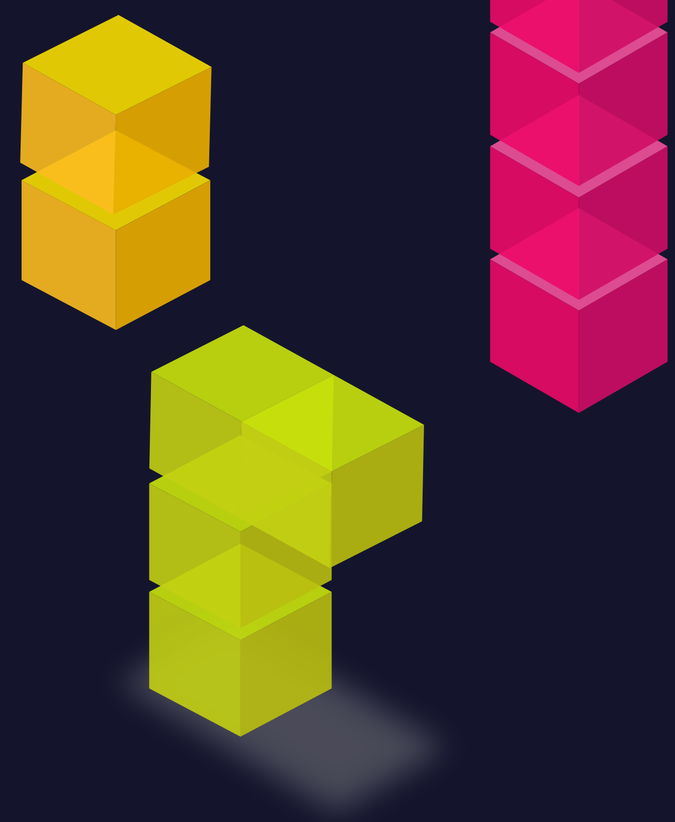
PEER DISCOVERY

```go
// Advertiser is an interface for advertising services
type Advertiser interface {
    // Advertise advertises a service
    Advertise(ctx context.Context, ns string, opts ...Option) (time.Duration, error)
}

// Discoverer is an interface for peer discovery
type Discoverer interface {
    // FindPeers discovers peers providing a service
    FindPeers(ctx context.Context, ns string, opts ...Option) (<-chan peer.AddrInfo, error)
}

// Discovery is an interface that combines service advertisement and peer discovery
type Discovery interface {
    Advertiser
    Discoverer
}
```

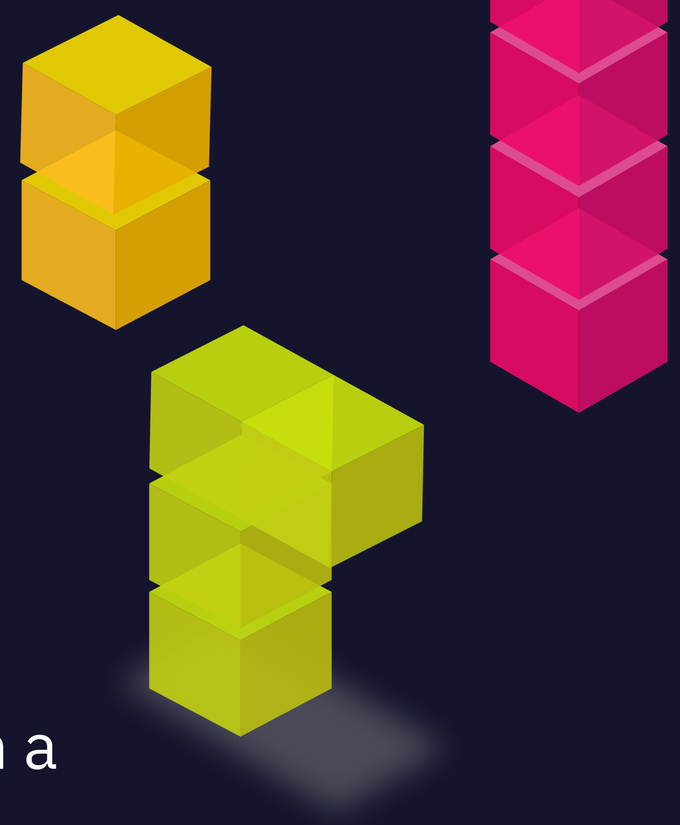
PEER DISCOVERY



- **backends:**
  - DHT.
  - Rendezvous nodes / signalling servers.
  - mDNS for local peer discovery.
  - future: Bluetooth, NFC, etc.
  - future: Brahms (membership sampling via pubsub).
  - anything, really: matrix channel, IRC, etc.

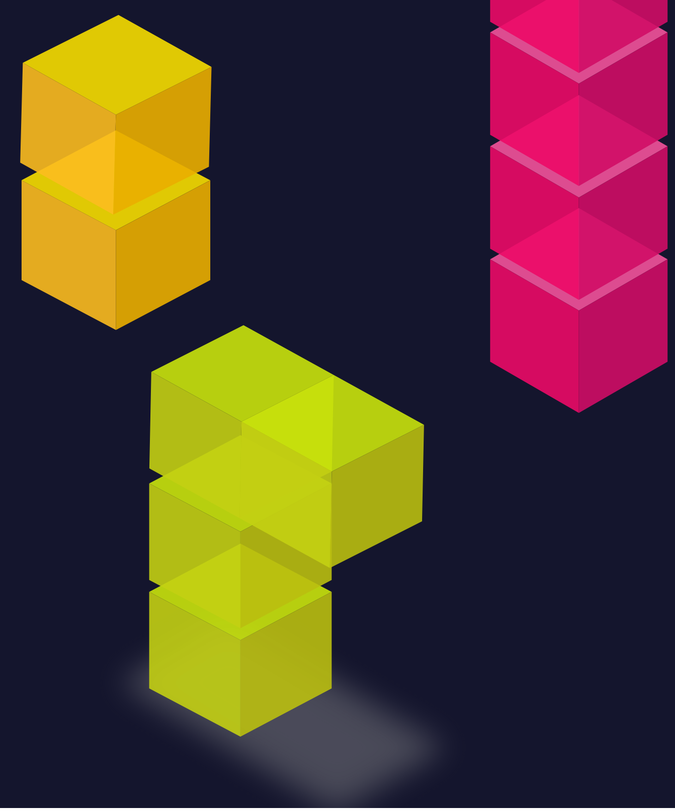
- **the bootstrapping/seeding issue:** hybrid approaches?

CONTENT ROUTING

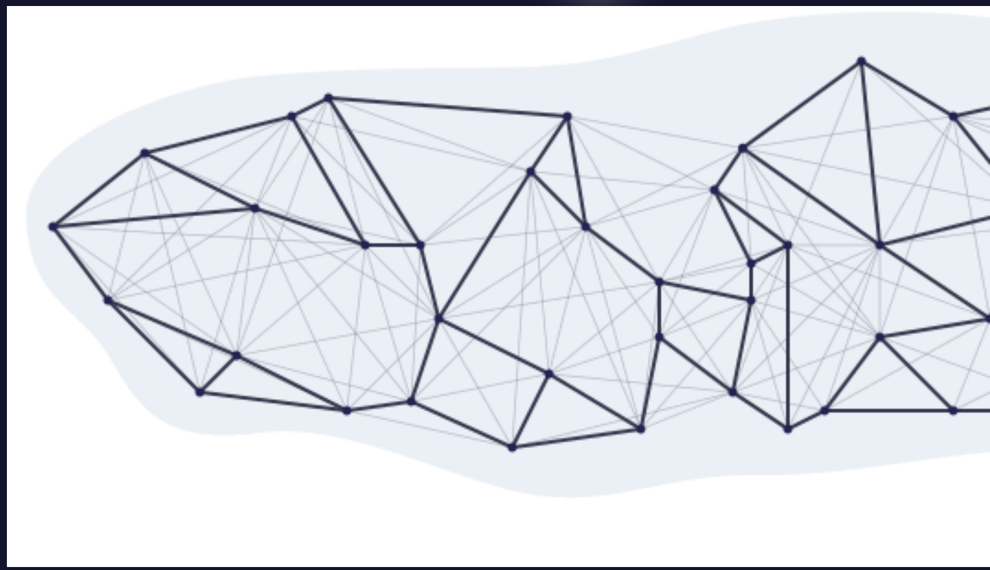


- Store, find, advertise content-addressed chunks of data in a decentralised p2p network.
- Also an interface, mostly fulfilled by the DHT.
- Alternatives implementations can query DNS, use pubsub, etc.

PUBSUB / GOSSIP



- Publish/subscribe (pubsub) is a message-oriented communication pattern.
- The interaction model of pubsub is M:N.
  - This is contrast to RPC or request/reply patterns which are 1:1.
- **Pubsub communication is asynchronous and decoupled in nature.**
- Widely popular in enterprise software (Apache ActiveMQ, Apache Kafka, RabbitMQ).
  - **Centralized around brokers in private networks. Not suitable for public message fabrics.**

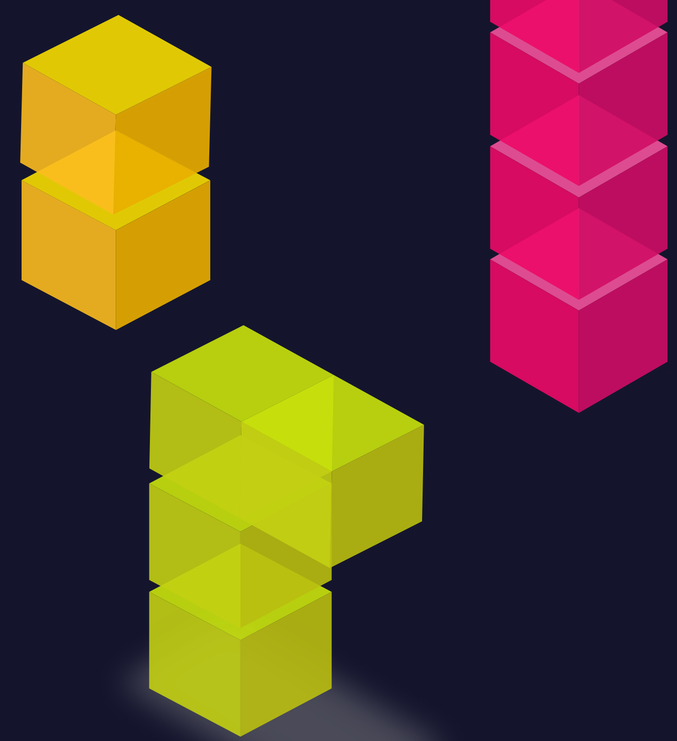


DESIRABLE PROPERTIES IN P2P PUBSUB

1. ***Reliability:*** all messages get delivered to all peers subscribed to the topic.
2. ***Resilience:*** peers can join and leave topics or the network without disruption under realistic churn scenarios, with **fast recovery** to mend the overlay to restore 100% hit.
3. ***Efficient dissemination:*** fast delivery, low write amplification, fair distribution of load.
4. ***Altruism-free routing:*** only nodes invested in a topic participate in its dissemination.
5. ***Scalability:*** enormous number of nodes, topics, messages, subscriptions, subscribers.
6. ***Low overhead*** of overlay maintenance: nodes track as little state as possible.
7. ***Simplicity:*** the system is simple to understand and implement.

Adapted from: PolderCast: Fast, Robust, and Scalable Architecture for P2P Topic-Based Pub/Sub

DECENTRALIZED PUBSUB IN LIBP2P

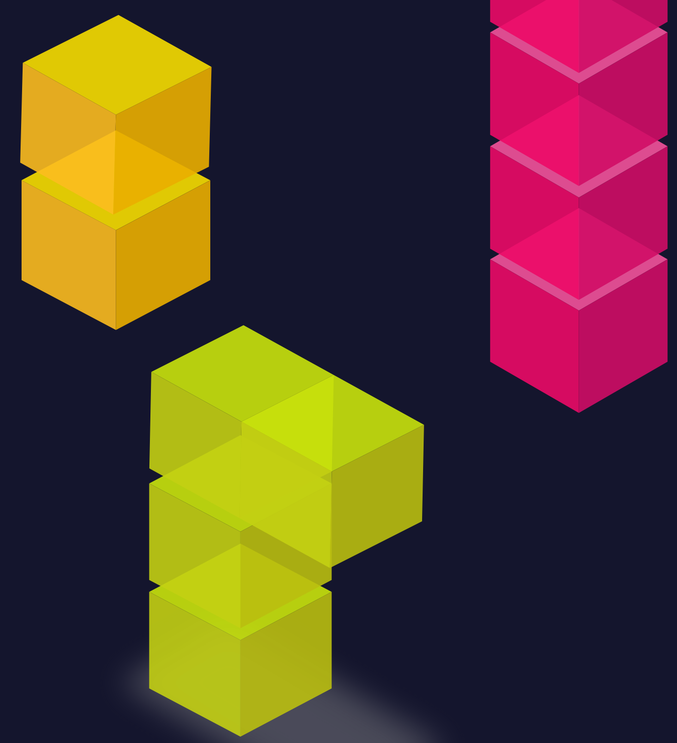


- various routing algorithms available:
  - floodsub (initial) / randomsub (for fun).
  - **gossipsub.**
    - Provides base layer and primitives to build adaptive, self-optimizing routing algorithms capable of repair and rearrangement, in the presence of churn, latency and failures.
    - forms stable reciprocal topic-based meshes to exchange messages.
    - with some gossip to detect message delivery failures and enable routing decisions.
    - control message piggybacking.
    - go, js, rust, jvm implementations

Episub: Plumtree + HyParView + GoCast.

CIRCUIT RELAY (TURN)

AUTONAT AND AUTORELAY

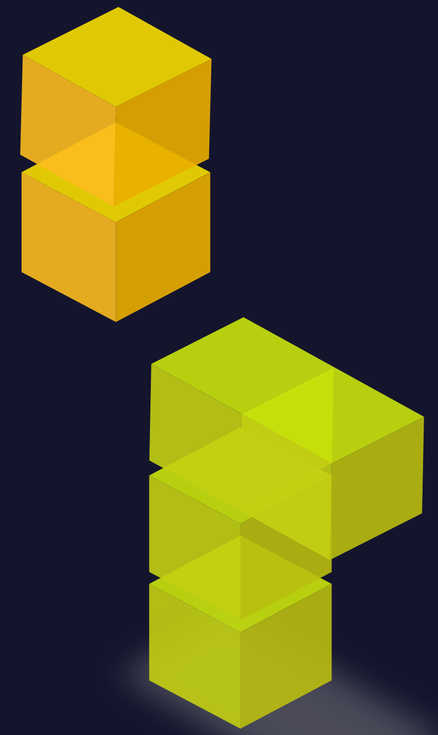


- Via the identify protocol, we get decentralised “STUN” for free.
- We also use uPNP and NAT-PMP to create router mappings *automagically*.
- **autonat** is an ambient service that requests dialbacks on addresses we observe of ourselves.
  - it determines if we’re actually diallable and reports a routability: PUBLIC, PRIVATE, UNKNOWN.
- **autorelay** can kick in when routability is PRIVATE.
  - discovers relays automatically, connects to them, and advertises relay addrs. we are now connectable.
  - does not scale well.

🔮 **next: decentralized hole punching.**

coming soon, to a node near you…

NAT HOLE PUNCHING

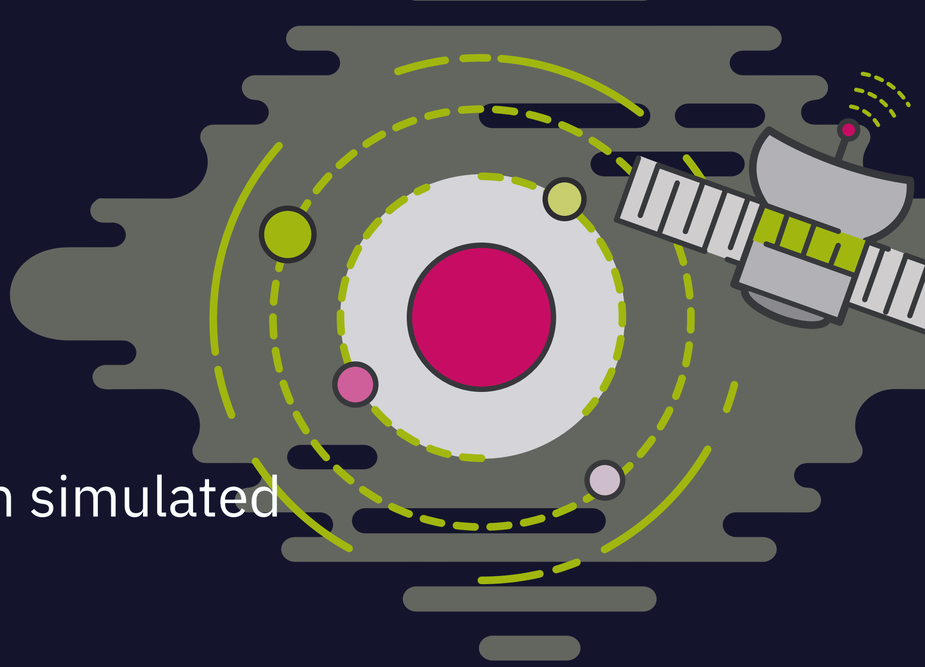


- **relaying traffic is inefficient.**
  - latency, centralisation, and expensive.

- **instead: use relay servers as a conduit for hole punching signalling and sync.**
  - establish an embryonic relayed connection.
  - immediately try to upgrade to direct conn
    - then transplant connection state (non-trivial).
  - tcp success rate is 60%; udp is 90%. that’s also why we want QUIC.
  - inspired by ICE.
  - careful with tcp simultaneous open; need to adapt multistream-select v1.

(besides everything I've mentioned)

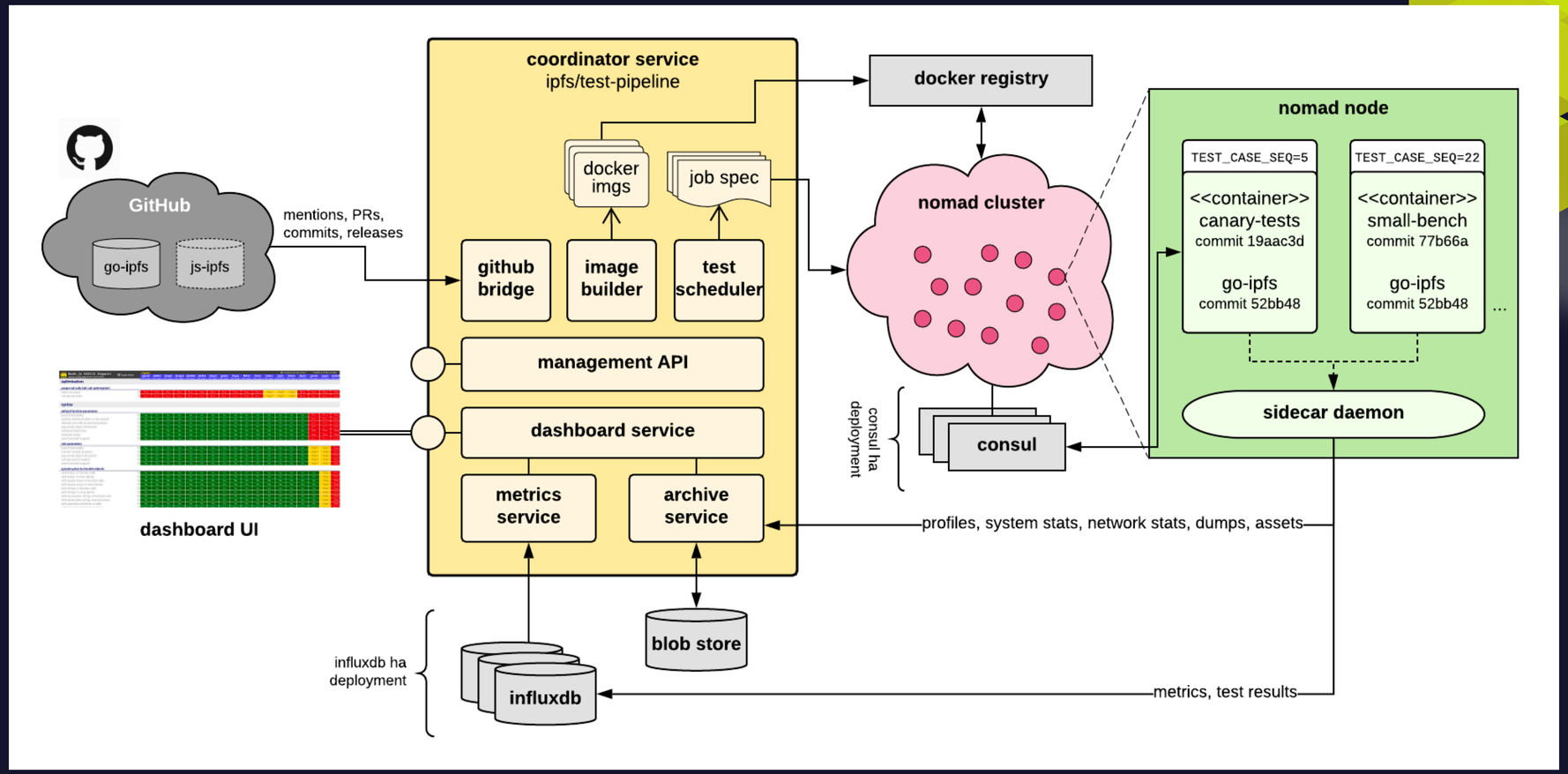
ON THE HORIZON



- distributed, reproduceable testing across 100k nodes with simulated network conditions.
- CTK: conformance test kit.
- even better specs: goal 100% spec'ed.
- Phantom{Drift} (network visualisation).
- DHT 2.0.
- NAT hole punching via relays. Direct connection upgrade.
- episub and gossipsub formalisation.
- spec completeness and accuracy; target: 100%.
- wireshark dissectors.

on the horizon

INTERPLANETARY TESTGROUND



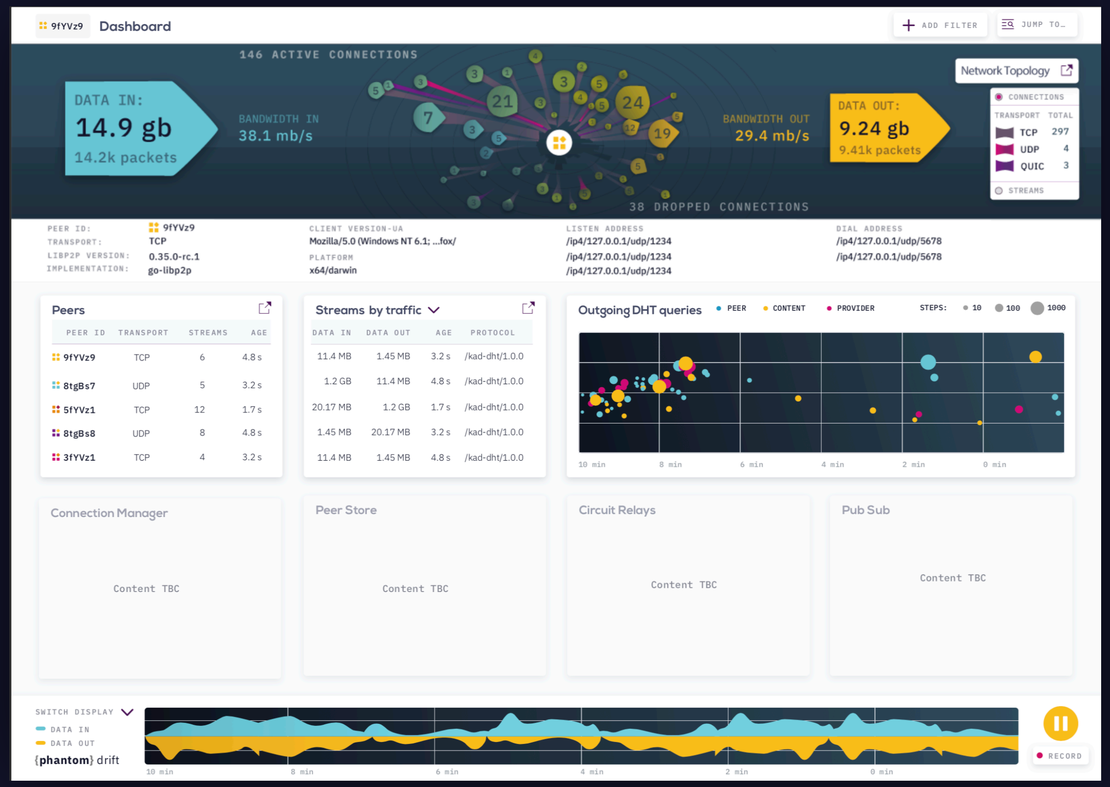
https://github.com/ipfs/testground

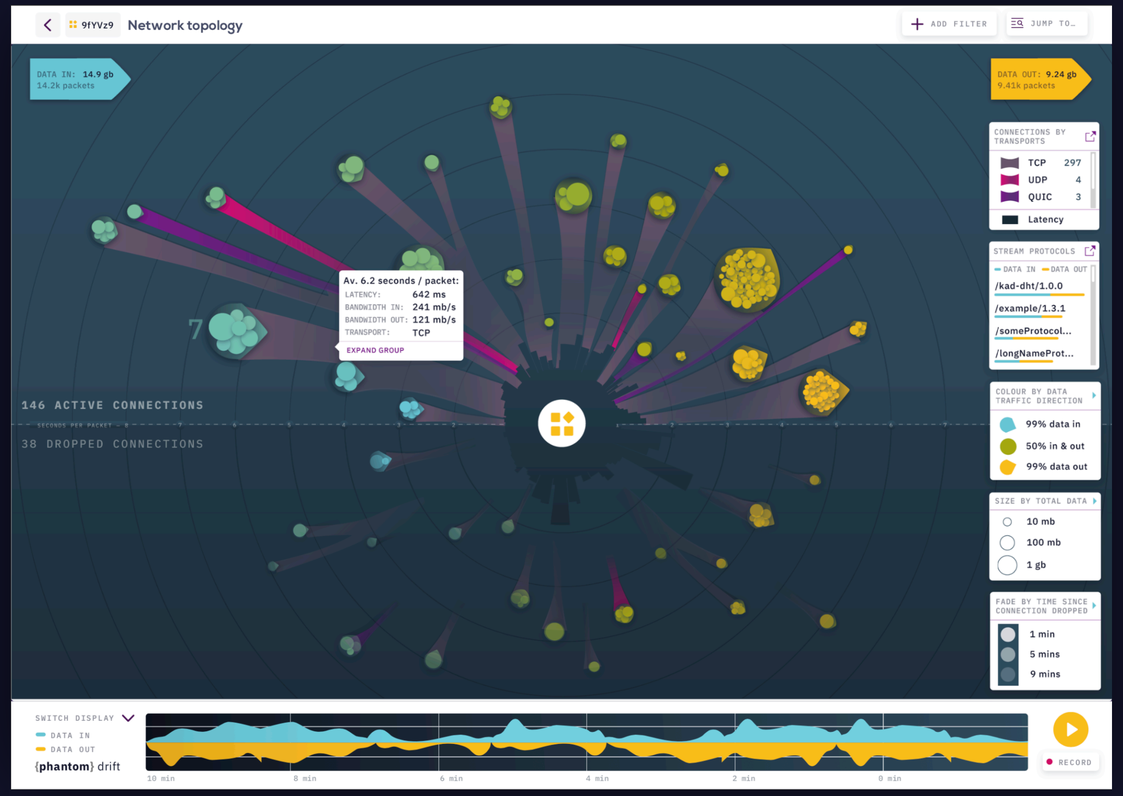
future

on the horizon

INSTRUMENTATION

standard introspection protocol and d3-powered visualizations.





WE  YOU

get involved through GitHub!

shape the future

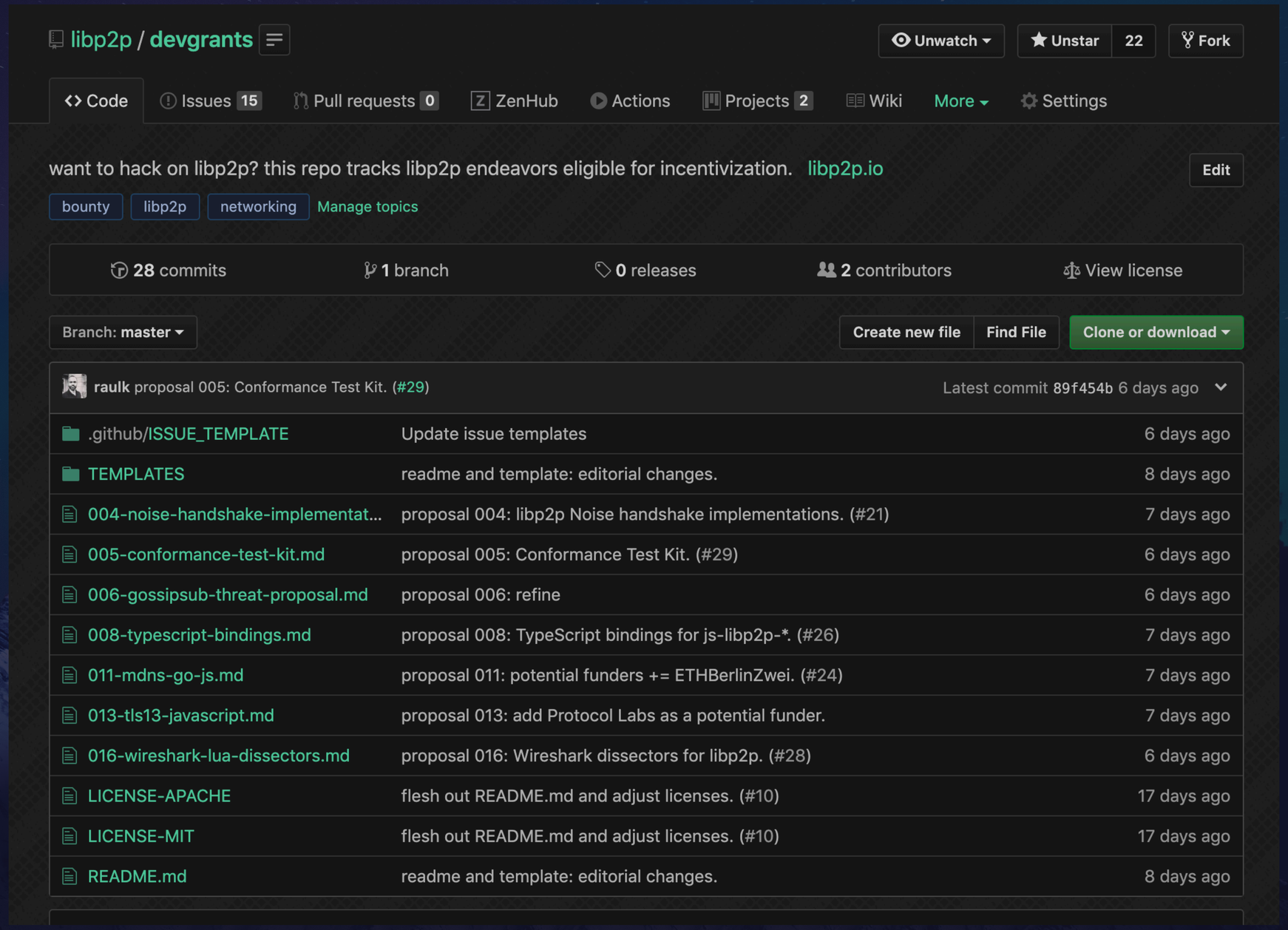
libp2p/specs

hack on meaty things

libp2p/devgrants

contribute

project repos



https://github.com/libp2p/devgrants/

libp2p

questions?

LET'S HACK ON THE FUTURE OF P2P NETWORKING TOGETHER.

RAÚL KRIPALANI

GITHUB
https://github.com/raulk

EMAIL
raul@protocol.ai

TWITTER
@raulvk