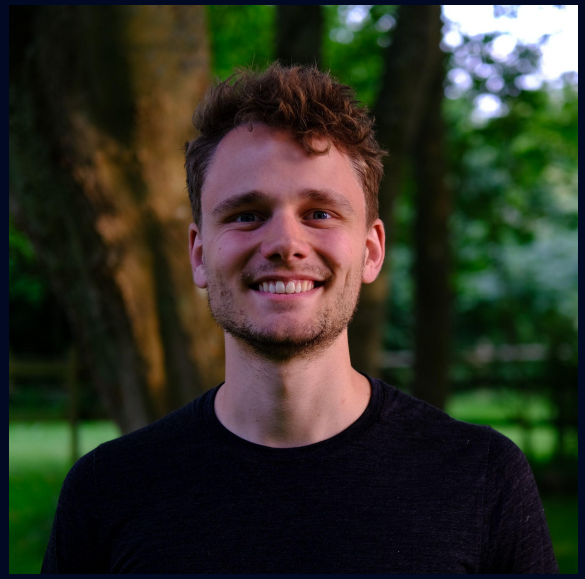
Introduction to libp2p

LIBP2P

Max Inden

Software Developer at Protocol Labs,
stewarding the libp2p project.

Maintainer of the libp2p Rust implementation.



max.inden@protocol.ai
@mxinden on GitHub / Twitter / ...

A modular peer-to-peer networking stack

- Composable building blocks based on a shared core to assemble future-proof p2p networking layers.
- All you need to build peer-to-peer applications.
- Runs on many runtimes: browser, mobile, embedded.
- Implemented in 7+ languages
- Powers the IPFS, Ethereum 2, Filecoin and Polkadot network

LIBP2P

Where?

Where does libp2p live?

L7 **Peer to Peer Application**

L3 / L4 **Transport**

L2 **Data-link Layer**

L1 **Physical Layer**

Let’s build a p2p chat application

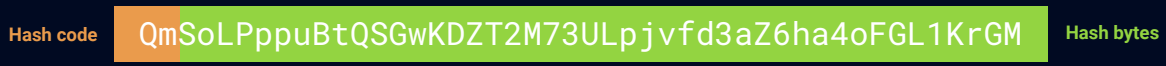
> All you need to build peer-to-peer applications.

To get familiar with the many building blocks of libp2p, let’s design a basic peer-to-peer chat application based on libp2p throughout the rest of the presentation.

- No centralized infrastructure (apart from bootnodes)
- Support for direct messages and group conversations
- Ephemeral messages (for now)
- Message authentication

Decentralized Process Addressing

Peer Identity



- Uniquely identifies peers
- IDs derived from their public key
    - Four key types: RSA, Ed25519, SEC256k1, ECDSA
    - Multihash representation
        - Identity multihash (i.e. no hash) if public key < 48 bytes
        - SHA-256 of multihash if > 48 bytes

Multiaddress

`/ip4/1.2.3.4/tcp/5001/p2p/<peer_id>`

- Network processes are named as filepaths.
- Specify available processes in a peer in a single address
- Used for dialing peer

Why?

Decentralized Process Addressing

- Ability to locate, connect, authenticate, negotiate and interact efficiently with any process in the world
  - Independent of the runtime
  - In a seamless manner (NAT-traversal, heterogeneous networks, relays).
  - Fully operational even in a case of mobility, relocation, roam, etc.
  - Every peer is uniquely identified.
- Network- and topology-independent alternative to endpoint addressing (e.g. IP networks)

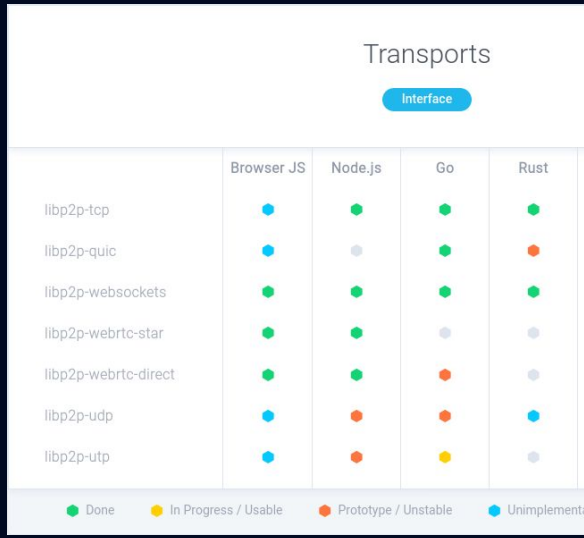

Transports
Multiplexers
Secure Channels
Peer Discovery
LIBP2P
Peer Routing
Content Routing
NAT Traversal
Pubsub


Transports
Multiplexers
Secure Channels
Peer Discovery
Peer Routing
Content Routing
NAT Traversal
Pubsub
LIBP2P

Transports

- Transports are core abstractions of libp2p
  - Enable connection establishment
  - Dialing and listening

- Current transports:
  - TCP, QUIC, WebSockets, WebRTC, Bluetooth, ...




Transports
Multiplexers
Secure Channels
Peer Discovery
LIBP2P
Peer Routing
Content Routing
NAT Traversal
Pubsub

Secure Channels

- Peer authentication and transport encryption.
- Transmission integrity. Non repudiation. Message authentication
  - Does not preclude from encrypting/signing application data
- Several security protocols supported:
  - Noise: Security framework. Handshake and negotiate primitives.
  - TLS 1.3

noise-libp2p - Secure Channel Handshake

> A libp2p transport secure channel handshake built with the Noise Protocol Framework.

Lifecycle Stage	Maturity	Status	Latest Revision
3A	Recommendation	Active	r2, 2020-03-30

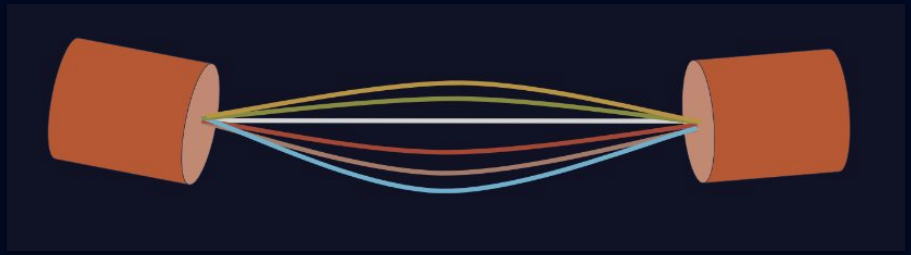
libp2p TLS Handshake

Lifecycle Stage	Maturity	Status	Latest Revision
2A	Candidate Recommendation	Active	r0, 2019-03-23


Transports
Multiplexers
Secure Channels
Peer Discovery
LIBP2P
Peer Routing
Content Routing
NAT Traversal
Pubsub

Multiplexing

- Establishing a P2P connection may not be cheap or easy (e.g. hole punching, negotiation, handshake, etc.)
- Re-use established connections for several protocols.
  - Applications can leverage already established connections.
- Streams are uniquely identified based on its multiplexer specification.
- Several implementations of multiplexers available:
  - Language specific libraries for stream multiplex (Yamux, Mplex)
  - Transport protocol native multiplexing capabilities (QUIC)



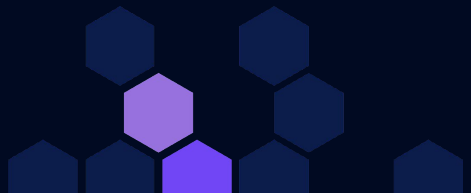

Transports
Multiplexers
Secure Channels
Peer Discovery
LIBP2P
Peer Routing
Content Routing
NAT Traversal
Pubsub

NAT Traversal

Motivation: *IPFS DHT crawl measurements (Nov 22nd 2019) showed that out of 4344 peers, 2754 were undialable (~63%).*

Goal:

- Achieve direct global connectivity in heterogeneous networks.
- No dependency on centralized infrastructure.

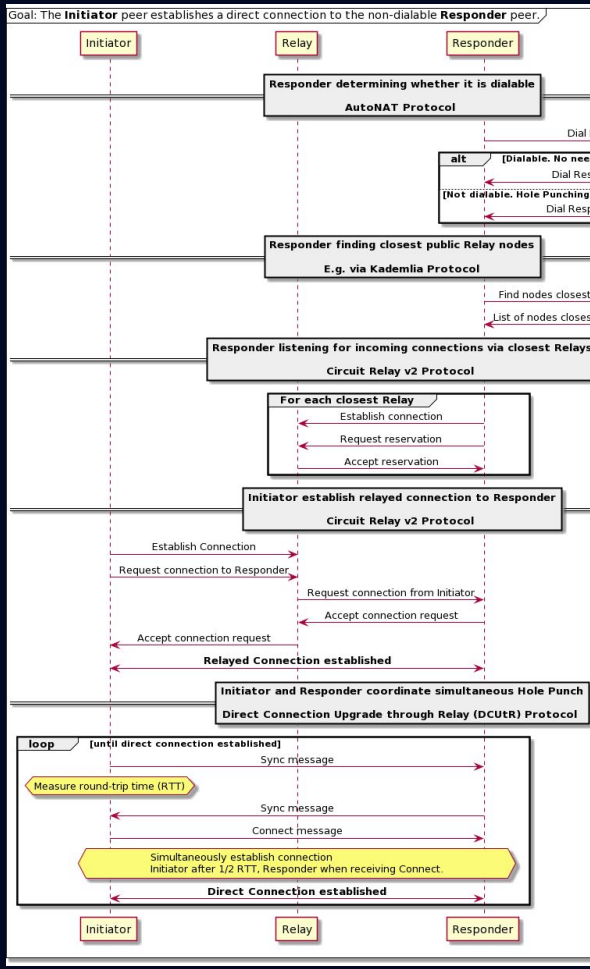


NAT Traversal

Short term - *Project Flare*

- Transport Protocols: TCP, QUIC
- Relay Protocol (TURN-like): Circuit Relay v2
- Signaling Protocol: Direct Connection Upgrade through Relay (DCUtR)
- STUN Protocol: AutoNAT

Long term - WebRTC




Transports
Multiplexers
Secure Channels
Peer Discovery
LIBP2P
Peer Routing
Content Routing
NAT Traversal
Pubsub

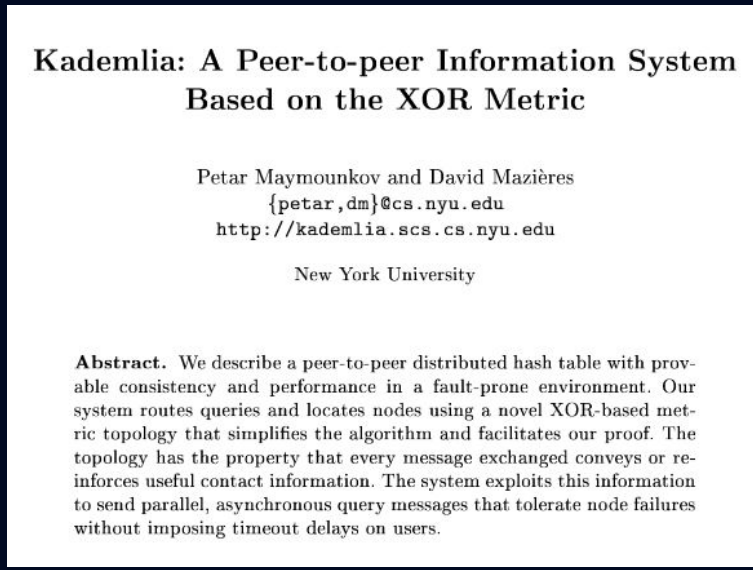
Peer Discovery / Routing

- Announce services to other peers
- Discover peers supporting certain services
- Find specific peers by peer ID

- Implementations
  - MDNS
  - Kademlia DHT

Kademlia DHT

- Distributed hash table
- Based on the Kademlia paper
- Operations:
  - FIND_NODE
  - GET_VALUE and PUT_VALUE
  - GET_PROVIDER and PUT_PROVIDER



Kademlia: A Peer-to-peer Information System Based on the XOR Metric

Petar Maymounkov and David Mazières
{petar,dm}@cs.nyu.edu
http://kademlia.scs.cs.nyu.edu

New York University

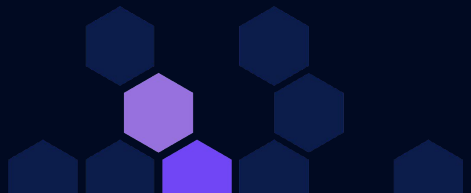
Abstract. We describe a peer-to-peer distributed hash table with provable consistency and performance in a fault-prone environment. Our system routes queries and locates nodes using a novel XOR-based metric topology that simplifies the algorithm and facilitates our proof. The topology has the property that every message exchanged conveys or reinforces useful contact information. The system exploits this information to send parallel, asynchronous query messages that tolerate node failures without imposing timeout delays on users.

https://pdos.csail.mit.edu/~petar/papers/maymounkov-kademlia-lncs.pdf

Content Routing

- Find and advertise Content-addressed chunks of data

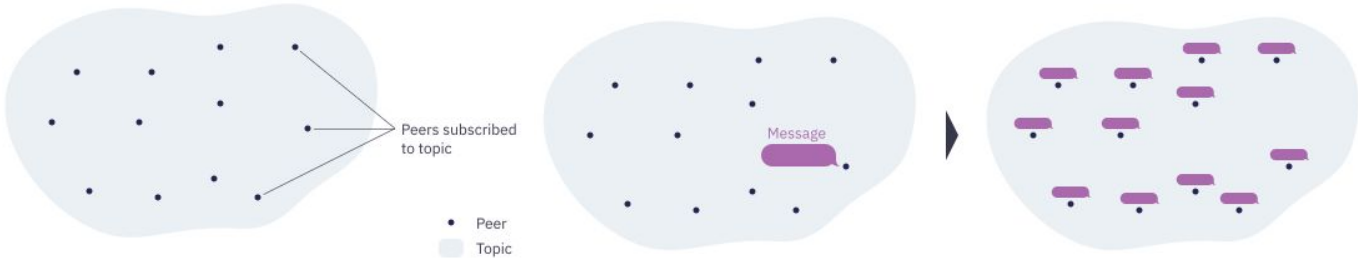
- Implementations
  - Kademlia DHT
  - DNS
  - BitTorrent trackers
  - Any other providing subsystems




Transports
Multiplexers
Secure Channels
Peer Discovery
LIBP2P
Peer Routing
Content Routing
NAT Traversal
Pubsub

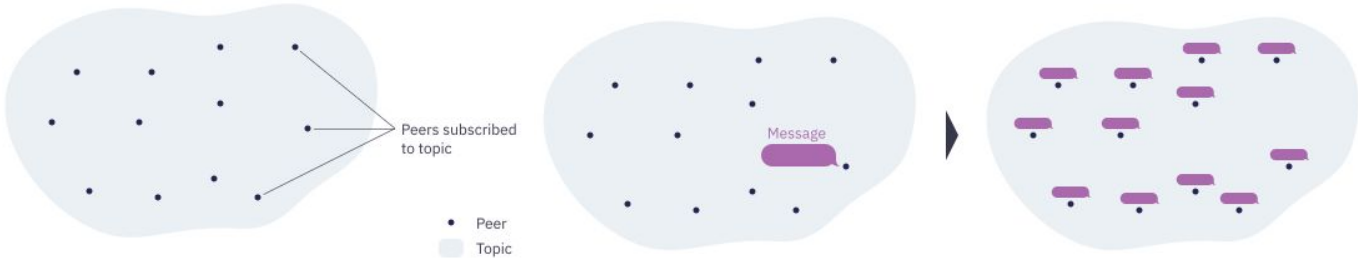
PubSub Interface

- PubSub is a message-oriented communication pattern.
- M:N interaction model. Asynchronous communication.
- Peers congregate around topics. Some processes publish messages and others listen to them. Common in enterprise software with decentralized brokers.



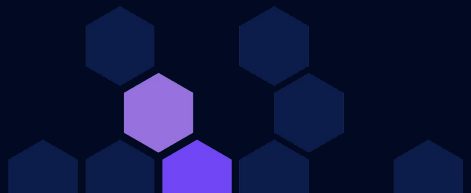
PubSub Interface

- Decentralized P2P PubSub.
  - Brokerless, self-regulating, no global knowledge
  - Constructed as overlay networks collaborating for message deliverability
- Several available protocols
  - GossipSub
  - FloodSub
- Use cases: IPNS, content-addressing, blockchain consensus, message dissemination, etc.



Summary

- Peer IDs and Multiaddr
- Transports
- Connection upgrades for security and mulitplexing
- NAT traversal for full connectivity
- Peer and content discovery / routing
- PubSub message communication



Roadmap

- Long-term
  - □ Decentralizing networks
  - 🤫 A spyproof libp2p
  - 📱 libp2p in mobile devices
  - 💡 libp2p in IoT
  - 🎓 libp2p as a platform for Networks Research & Innovation
  - 🤖 libp2p as a WASM library

Roadmap

- Short-term
  - 🤝 Low latency, efficient connection handshake
  - 🕸️ Unprecedented global connectivity
  - ...

Where to go from here

- **Documentation** - docs.libp2p.io/
- **Forum** - discuss.libp2p.io/
- **Specification** - github.com/libp2p/specs/
- **Implementations**
  - github.com/libp2p/go-libp2p
  - github.com/libp2p/rust-libp2p/
  - github.com/libp2p/js-libp2p
  - ...

Transports

Multiplexers

Secure Channels

Peer Discovery

Peer Routing

Content Routing

NAT Traversal

Pubsub

LIBP2P

Thank you for joining

Questions?

max.inden@protocol.ai
@mxinden on GitHub / Twitter / ...