Designing and testing an application layer communication protocol for decentralized peer to peer networks, based on libp2p

Master Thesis Proposal; by Tim Schupp ([redacted]*)*

This document outlines my idea for reasearching, developing and testing a new communication protocol and VPN. Using a decentralized peer to peer approach for node and network mangement implemented on the application layer on-top of libp2p. It should allow the creation of seperated peer VPN networks; from a single binary without requiring central control or global configuration.

Libp2p is a peer to peer networking stack. A specification of several approaches and protocols used in the development of the Interplanetary File System (IPFS). Libp2p implements many networking protocols and interconnects them. This is achieved by defining a spec that independently manages peers, addressing and transport selection.

Motivation

There are many VPN implementations; like openvpn and wireguard, one integrated at kernel level, the other in user-space. These generally require to include routing information during configuration, so they mainly allow creating VPN networks across static nodes. Serveral tools build on top of these VPN implementations, tailscale and netbird for example allow building dynamic peer networks and mange routing information, to achieve dynamic VPN systems, *but these systems require a central control node (and additional binaries).* New approaches like edgevpn that allow building peer to peer based VPN networks. Edgevpn is one of the inspirations to my tests and this proposal - it is also build on top of libp2p - it allows for dynamic VPN networks without central control nodes. Edgevpn allows creating networks in a self managed peer to peer systems. But it lacks per-node access control mechanisms, and uses a distributed-ledger to synchronize a global configuration to nodes. It doesn't allow to configure nodes independently, nor does it allow for seperate networks. Based on one distributed Ledger all peers are discovered and all network state is managed. *These projects motivated the Idea of creating and testing a more flexible libp2p based communication protocol that:*

1. is an application layer protocol, based on libp2p
2. same logic on every node, no central control node
3. automatic peer discovery and address management
4. multi network system with authentication based network join procedure
5. fully separated per-node and per-network access control and configuration
6. ability to exclude a malicious peer from a network
7. is as simple as possible while maintaining (1-6)

Such a protocol could allow creation global VPN networks that work with currently available transports, are fully encrypted, and mobile across any libp2p-supported transports.

Previous Work & Proof of Concept

I've studied multiple implementations of libp2p, and libs using it. Experimented a-lot with go-libp2p and implemented several proof of concepts and a bigger implementation (open-chat)

including a working mvp for some points listed before. I've created the basic data-structures and a sqlite based system for managing node state and persistance across re-starts. Developed experimental APIs to enable network management and joining. *From my initial tests, the minimal setup seem to require:* (a) Some way to authorize a user. (b) Some way to request nodes on a default network, without being authorized (for authorization and network joining). (c) Some way to bootstrap the network (find the first peer). (d) Some way to synchronize network peers across network nodes. (e) Some way to manage per-node routing tables and VPN configuration. These components I've developed in a very basic fashion in open-chat. You can find my basic federation API spec here, but this implementation also includes several APIs that are not required for this proposal. And here I wrote a blog post demonstrating the basic functionality so-far, this demonstrates a working experimental implementation for some of points mentioned below, with significant overhead and at-lot of simplification required.

Work on this proposal could start with; (1) simplifying and condensing it to the minimally required endpoints. (2) create a formal and detailed Protocol spec, defining endpoints, logic and data-types. (3) produce a minimal working prototype that ensure functions (a-d) having as little apis and code as possible. (4) Implement tests for most common network setups, authentication and some network topologies. (5) Test investigate and analyze the system, conduction the actual research and addressing questions posed below, doing an in-depth analysis, simulation and evaluation. A-long the way integrating and improving on the spec and implementation.

Research Objectives / Questions

(a) Develop and document the basic protocol spec and implement functioning prototype
   (i) deeply understand the inner working of libp2p and outline it in the thesis introduction
   (ii) implement a stable prototype that allows creating reliable VPN networks based on go-libp2p

(b) testing reliability, peer discovery and bootstrapping; implementing test-workflows covering:
   (i) Can the system bootstrap it's network, will it discover all peers, which circumstance can influence connectivity and discovery (firewalls, middleboxes, NAT)
   (ii) Can the system handle handover between different networks, what are handover times, can the same connection be kept alive. What are application effects of handover.

(c) measuring scalability and network size overhead
   (i) create a simulation setup that scales to a high number of nodes
   (ii) measure the influence of network size on synchronization speed and path finding
   (iii) analyze and highlight possible scaling bottlenecks and suggest optimizations

(d) What is the latency and trough-put of a peer connections in different scenarios, conduct large scale measurements for different setups:
   (i) same network, different networks, behind NAT trough relay-node
   (ii) Across regions using a Relay between two nodes behind NAT
   (iii) use tools to deeply investigate what effects contribute here; e.g.: system networking, implementation details, bandwidth etc...

(e) Review what are security implications and caveats of this approach
   (i) compare it to similar protocols, highlight differences
   (ii) what are possible incidents and how can malicious actors influence the network

Appendix

Some additional information. Like the minimal endpoints and data-structures such an implementation needs. On request I can prepare examples and demonstrations of the things things I've already implemented. This e.g.: could be showing that peer discovery, network joining and creating of the vpn configuration have successfully been achieved.

Basic Protocol Overview

The minimal system desined for this Thesis would require at-least the development of these APIs:

- `GET /peer_info`: Endpoint to access node id peer information
- `GET /get_federation_node/<peer-id>/`
- `POST /relay_network_request/<network>/<peer-id>/`
- `POST /register_network_peer/<network>/<peer-id>/`
- `POST /authorize/`: Used for user / network authorization
- `POST /synchronize_network/`: Synchronized Peer state for authorized network users
- `POST /add_route_rule/`: Allows defining VPN / routing rules. This is what creates and manages network (tun) devices and proxies routing rules across nodes

Each node needs some basic data structures for:

- `Users` (Admin and Network Users, hold authorization information)
- `Networks` (Registered Networks)
- `Peers` (libp2p manages an internal peer table, that is not persisted across restarts)
- `Routing Rules` (Hold per-node information for VPN routing)

Context & Acknowledgements

This document was send to [redacted] ([redacted]) - it is a proposal for a master thesis topic - at the [redacted]. So this is merely and idea / suggestion of a topic I'm highly interested in and would love to write a thesis on. In my opinion, the proposal fits the chair, as-well as the lecture [redacted] very well. The development and testing of the system proposed here would allow me to explore many communication protocols, understand there implementation details, and apply all this to create and rigorously test a real system, and detail my findings in a scientific context. All implementations, simulation setups and results would be open-source, and contributions to the libp2p spec or implementation, should also be made if required.

No LLMs have been used for this documents texts; its hand-typed and based on my own thoughts!