Charon networking

Overview

This document describes Charon's networking model which can be divided into two parts: the internal validator stack and the external p2p network .

Internal Validator Stack

Charon is a middleware DVT client and is therefore connected to an upstream beacon node and a downstream validator client is connected to it. Each operator should run the whole validator stack (all 4 client software types), either on the same machine or on different machines. The networking between the nodes should be private and not exposed to the public internet.

Related Charon configuration flags:

- --beacon-node-endpoints
- : Connects Charon to one or more beacon nodes.
- --validator-api-address
- : Address for Charon to listen on and serve requests from the validator client.

External P2P Network

The Charon clients in a DV cluster are connected to each other via a small p2p network consisting of only the clients in the cluster. Peer IP addresses are discovered via an external "relay" server. The p2p connections are over the public internet so the charon p2p port must be publicly accessible. Charon leverages the popular libp2p protocol.

Related Charon configuration flags :

- --p2p-tcp-addresses
- : Addresses for Charon to listen on and serve p2p requests.
- --p2p-relays
- : Connect charon to one or more relay servers.
- --private-key-file
- : Private key identifying the charon client.

LibP2P Authentication and Security

Each charon client has a secp256k1 private key. The associated public key is encoded into the cluster lock file to identify the nodes in the cluster. For ease of use and to align with the Ethereum ecosystem, Charon encodes these public keys in the ENR format , not in libp2p's Peer ID format .

caution Each Charon node's secp256k1 private key is critical for authentication and must be kept secure to prevent cluster compromise.

Do not use the same key across multiple clusters, as this can lead to security issues.

For more on p2p security, refer to libp2p's article . Charon currently only supports libp2p tcp connections with noise security and only accepts incoming libp2p connections from peers defined in the cluster lock.

LibP2P Relays and Peer Discovery

Relays are simple libp2p servers that are publicly accessible supporting the circuit-relay protocol. Circuit-relay is a libp2p transport protocol that routes traffic between two peers over a third-party "relay" peer.

Obol hosts a publicly accessible relay at https://0.relay.obol.tech and will work with other organisations in the community to host alternatives Anyone can host their own relay server for their DV cluster.

Each charon node knows which peers are in the cluster from the ENRs in the cluster lock file, but their IP addresses are unknown. By connecting to the same relay, nodes establish "relay connections" to each other. Once connected via relay they exchange their known public addresses via libp2p's identify protocol. The relay connection is then upgraded to a direct connection. If a node's public IP changes, nodes once again connect via relay, exchange the new IP, and then connect directly once again.

Note that in order for two peers to discover each other, they must connect to the same relay. Cluster operators should therefore coordinate which relays to use.

Libp2p'sidentify protocol attempts to automatically detect the public IP address of a charon client without the need to explicitly configure it. If this however fails, the following two configuration flags can be used to explicitly set the publicly advertised address:

- --p2p-external-ip
- : Explicitly sets the external IP address.
- --p2p-external-hostname
- : Explicitly sets the external DNS host name.

caution If a pair of charon clients are not publicly accessible, due to being behind a NAT, they will not be able to upgrade their relay connections to a direct connection. Even though this is supported, it isn't recommended as relay connections introduce additional latency and reduced throughput and will result in decreased validator effectiveness and possible missed block proposals and attestations. Libp2p's circuit-relay connections are end-to-end encrypted, even though relay servers accept connections between nodes from multiple different clusters, relays are merely routing opaque connections. And since Charon only accepts incoming connections from other peers in its cluster, the use of a relay doesn't allow connections between clusters.

Only the following three libp2p protocols are established between a charon node and a relay itself:

- circuit-relay
- : To establish relay e2e encrypted connections between two peers in a cluster.
- identify
- : Auto-detection of public IP addresses to share with other peers in the cluster.
- peerinfo
- : Exchanges basic applicationmetadata
- for improved operational metrics and observability.

All other charon protocols are only established between nodes in the same cluster.

Scalable Relay Clusters

In order for a charon client to connect to a relay, it needs the relay's multiaddr (containing its public key and IP address). But a single multiaddr can only point to a single relay server which can easily be overloaded if too many clusters connect to it. Charon therefore supports resolving a relay's multiaddr via HTTP GET request. Since charon also includes the uniquecluster-hash header in this request, the relay provider can useconsistent header-based load-balancing to map clusters to one of many relays using a single HTTP address.

The relay supports serving its runtime public multiaddrs via its--http-address flag.

E.g.,https://0.relay.obol.tech is actually a load-balancer that routes HTTP requests to one of many relays based on thecluster-hash header returning the target relay's multiaddr which the charon client then uses to connect to that relay.

The charon--p2p-relays flag therefore supports both multiaddrs as well as HTTP URIs.Edit this page Previous Cluster configuration Next CLI reference