IPLD Resolver

IPC Spec - IPLD Resolver

The IPLD Resolver can facilitate IPC in multiple ways:

- connecting all participants of an IPC hierarchy into a P2P network
- advertising via GossipSub which subnets a particular node can serve data for (the primary motivation being sending bottom-up checkpoints containing CIDs, which the nodes on the parent subnet procure with the resolver)
- gossiping votes about application specific things in subnet-specific topics, e.g. evidence, observations, attestations
- resolving CIDs into content via BitSwap
- pre-emptively push data to the parent subnet to circulate it via GossipSub, instead of waiting until the request arrives via BitSwap (e.g. the contents of a bottom-up checkpoint)
-

The resolver used to be a standalone library before it was migrated to the IPC monorepo and upgraded to use a newer version of libp2p . Since then the BitSwap unit tests show that there is a bug with larger data structures, which we haven't had time to investigate. This function isn't used at the moment, but if it were, the problem had to be fixed first.

Use Cases

The docs have a fairly good overview of what this component does, so here we'll just concentrate on how it is used in the context of Fendermint:

- gossiping votes about which blocks are final on the parent subnet
- resolving bottom-up checkpoints from the child subnet (not used at the moment)
-

The resolver is instantiated in the run CLI command if the node is configured with both:

- an IPC subnet (can be root), and
- a multiaddress where it will listen to incoming requests
-

If enabled, the application will be started with:

- checkpoint resolver pool
- a finality vote publisher
- a finality vote subscriber
- the IPLD resolver service itself, which discovers peers, manages subscriptions, publishes memberships, etc.
-

Parent Finality Vote Gossip

The voting module in the topdown crate has a generic STM enabled VoteTally component which has the following components:

- chain
- contains contains block hashes that our node sees as final on the parent subnet at each block height
- votes
- contains votes that any particular block hash received at any height from validators
- power_table
- contains the public keys of the validators who are currently eligible to vote
-

With these the VoteTally can be used to register votes coming in over a gossip channel, and to look for a finalized block height that our node knows of where there is also a quorum, treating a vote on a block as an implicit vote on all its known ancestors as well.

The VoteTally is part of the ChainEnv and consulted by the ChainMessageInterpreter during block proposals. The goal is that we only make proposals on parent subnet finalities when the tally indicates that there is already a quorum. Since the voters are the same validators who will vote about the proposal, the presence of the quorum should be enough for the proposal to pass as well, preventing any liveness issues with the consensus.

The votes are being fed to the tally by the dispatch_resolver_events function.

BottomUp Checkpoint Resolution

The resolver crate under vm is a generic component which consists of two parts:

- The [pool](#)
- module contains theResolvePool
- which is an [STM](#)
- enabled component where we can submit items to be resolved, and monitor their status, collecting. The pool is generic in the items it can resolve, as long as they can be mapped to aCid
- and aSubnetId
- . The pool is the shared memory which is used by the interpreters to add items and inquire about their status during the block execution.
- The [ipld](#)
- module contains theIpldResolver
- which is runs in the background to execute tasks sent to theResolvePool
- by sending them to actual IPLDService
- .
-

Currently theChainEnv requires a pool working with [CheckpointPoolItem](#) .

Ultimately this is not currently in use because checkpoint submissions ended up containing all the bottom-up messages.