Top-down Finality

Topdown Finality

The topdown finality helps propagating the important states in the parent subnet to the child blockchain such as fund transfer and validator updates.

The following data is passed:

- Parent block height + block hash
- Transactions from the parent to the child subnet
- Validator changes from parent to the child subnet
-

The finality is represented in contract in the form of:

```

Copy // Link: https://github.com/consensus-shipyard/ipc/blob/7af25c4c860f5ab828e8177927a0f8b6b7a7cc74/contracts/src/structs/CrossNet.sol#L11
structParentFinality{ uint256height; bytes32blockHash; }

```

The validator changes are represented as an array of Changes while topdown messages are an array of IPCEnvelope .

Once enabled, the topdown finality runs in the background. From a high level point of view, the topdown finality works as follows:

- Topdown finality will first fetch the last committedParentFinality
- . If there is no previous committed parent finality, the genesis block is used
- .
- There is aParentSyncer
- that constantly polls the parent states through RPC calls, either fetching events emitted or through getters. TheParentSyncer
- stores the pulled data in cache and publishes a vote on the latest block seen.
- Once a quorum is formed on the blocks seen, a topdown finality proposal will be added
- to the cometbft proposal. The proposal is just theParentFinality
- struct shown above, i.e. a block height and the corresponding block hash.
- Once a topdown proposal is received by each node, it will be validated against the topdown syncer cache and the RPC node if there is a cache miss. If the checks do not pass, the proposal will be rejected. The checks include:
-
    - The height proposed is present in the cache or exists in the RPC node
-
    - The hash matches the corresponding height's block hash
- *
- Once the proposal is accepted, it will be executed. This means the topdown messages will be executed and validator changes will be stored.
-

All nodes must agree on the parent state, even though they may be using different RPC endpoints. Those RPC endpoints may have different views of the parent state. At the moment the nodes don't have any way to verify the correctness ofParentFinality received from the RPC endpoints. That's why the nodes have to reach consensus on the parent finality through a voting mechanism.

Parent Syncer

The parent syncer polls the parent to obtain the "finalized" state in the parent. The "finalized" view in the parent is by delaying a fixed number of blocks from the chain head. This ensures the probability of a parent reorg is small.

There are currently two implementations of parent syncerLotusParentSyncer andTendermintAwareSyncer . TheTendermintAwareSyncer is a wrapper ofLotusParentSyncer , which stops the polling if the node is currently syncing with its peers, because if it is, then it won't be proposing anything until it's finished, and it will receive the parent finalities from the CometBFT blocks instead.

TheLotusParentSyncer will actually constantly poll the next height. The next height is either the last committed parent finality height plus 1 or the latest height seen in cache plus 1. If the next height is not more than the delayed chain head, parent syncer will skip this round and sleep for some fixed time.

Once the data is pulled, it will be committed to a in-memory cache. This cache is indexed by block height. It forces

sequential insertion the blocks inserted aresequential in block height.

At the same time, the observed parent block will be added to theVoteTally .

In Lotus, there is a concept of null block. When this happens, there is no data in the block. In the parent syncer, this block will be skipped, i.e. a None is inserted.

To ensure the parent blocks are chained, parent syncer also makes sure the block hash of the parent blockmatches the latest non-null parent hash. Otherwise a warning will be logged.

Do note that the currentimplementation of parent syncer relies heavily on solidity events. Make sure the RPC node used gives full access to historical events, otherwise not all events will be correctly returned and result in data loss.

Vote Tally

TheVoteTally is a component used to keep track of votes cast on parent finalities by the current child validators, and supports looking for the highest parent block height and hash which received a quorum. The votes are published over a GossipSub channel, once a node observes a finalized block through the syncer described above.

A quorum detected by theVoteTally is used a pre-condition for finality proposals being added to the CometBFT block proposals, to avoid any liveness issues which could arise if the other validators were to reject the proposal. If a premature finality causes the block propsal to fail, it means in that round CometBFT cannot make progress, it cannot finalize a block, potentially causing the subnet blockchain to stall. By requiring a quorum, we avoid this issue by only proposing when we have high confidence that the proposal will be accepted.

SeeIPC Spec - IPLD Resolver for a more detailed discussion of theVoteTally .

Topdown Finality Proposal

Once theVoteTally has detected a quorum, a parent finality proposal will be made. The entrypoint is thislink . The high level idea for proposal creation is that it will first get the latest height in cache. However, do note that some fendermint nodes might be syncing faster than other fendermint nodes, if the proposed height is too large, other nodes will reject the proposal because they have yet to “see” it. As such, there is aMaxProposalRange parameter that controls how far the proposed height can be greater than the last committed block height. Once the height is set, the corresponding block hash is added to the proposal. The proposed topdown finality is added to the list of transactions sending to the cometbft.

Topdown Finality Proposal Validation

Once a topdown finality proposal is received by a node, it’schecked to ensure its validity. The checking performs:

- The proposed height either exists in topdown syncer
- The block hash of the proposed height is correct.
-

If the above two checks are valid, the proposal is accepted.

Topdown Finality Proposal Execution

Once a proposal is accepted, it will beexecuted together with the batch of transactions. The execution of the proposal happens in three stages:

- Commit new topdown finality
- Fetch validator changes and store
- Fetch topdown messages and execute
-

The validator changes and topdown messages are first fetched from the parent syncer cache. If they are not present, then it will fallback to the RPC node. There is a simple exponential backoffretry . If the retry fails, which means the node cannot be reached to obtain the validator changes or topdown messages, an error is thrown. The reason for this is that the cluster has decided on the topdown finality, if one cannot obtain the corresponding changes and message, then it should stop.

As topdown finality relies heavily on RPC node for querying, make sure the RPC node is reliable and returns enough historical data.

See alsoIPS Spec - Executions for a description of how proposals and executions are implemented.

Previous Materializer Next Networks Last updated5 days ago On this page *