TL;DR

We propose hierarchical finality gadgets

, a framework to construct a finality gadget. For a given Nakamoto-style chain (target chain

), which we want to provide finality, we use another blockchain as the finality gadget (FG) chain

. Our framework makes it possible to construct a finality gadget with the desired property (e.g., responsiveness, flexible commit rules, etc.) depending on the use-case.

Background: Finality Gadget

“Finality gadget” is a component that overlays any Nakamoto-inspired block proposal mechanism and provides deterministic finality with safety robust against network failure (asynchronous safety, Footnote 1). This idea is initially proposed by Casper FFG and now adopted by Polkadot (GRANDPA), NEAR (Nightshade Finality Gadget), and Concordium (Afgjort).

The benefit of finality gadgets is the decoupling of the finality cycle and block intervals. Unlike the protocols like Tendermint, Hotstuff, where all the consensus nodes vote every time a block is proposed, the chain continues to grow while consensus nodes are voting. Therefore, a blockchain with a finality gadget can handle a larger number of consensus nodes at the cost of longer time to finality.

Target chain

In this post, we call the blockchain target chain

, which our finality gadget overlays. The protocol of the target chain can be PoW + longest-chain, PoS + LMD GHOST, Ouroboros, etc. Such a protocol is required to produce blockchains, and the chains must eventually converge to a single chain by its fork-choice rule.

Proposal: Hierarchical Finality Gadget

We propose a design framework of finality gadgets called hierarchical finality gadgets

.

For a given target chain (e.g., eth2’s beacon chain, Eth1 chain, etc.), we introduce another blockchain protocol as a finality gadget (FG) chain

. (We call the blocks of the FG chain FG blocks

.) The FG chain only exists for finality. Specifically, we construct the FG chain as follows:

- There is a set of nodes called validators

that continuously perform consensus to grow the FG chain. * We assume a static validator set here but will discuss validator rotation later.

- We assume a static validator set here but will discuss validator rotation later.
- Validators receive blocks of the target chain.
- Each FG block specifies a checkpoint

, i.e., the hash of a specific block in the target chain. * A block proposer of the FG chain selects the latest block in the target chain, which is (i) favored by its fork-choice rule and (ii) received more than \Delta

seconds ago as the checkpoint (Footnote 2).

- A FG block is valid only if the checkpoint it specifies is available and extends the checkpoint specified by its parent FG block.
- A block proposer of the FG chain selects the latest block in the target chain, which is (i) favored by its fork-choice rule and (ii) received more than \Delta

seconds ago as the checkpoint (Footnote 2).

- A FG block is valid only if the checkpoint it specifies is available and extends the checkpoint specified by its parent FG block.

The FG chain and the target chain comprise a two-layer hierarchy of chains.

[

image

498×649 18.5 KB

](https://ethresear.ch/uploads/default/original/2X/0/0ca47ee36987fbd0de27a85a075d4c462f853197.png)

The finality condition and its relation to the fork-choice rule are:

- When a block B

in the FG chain gets finalized, the checkpoint specified by B

(and all of its ancestors in the target chain) gets finalized.

- The fork-choice rule of the target chain must start from the latest finalized block, instead of the genesis block.

Choice of finality gadget chain

The advantage of our proposal is the flexibility of the choice of the FG chain; we can use any blockchain/state machine replication protocol for the FG chain depending on the needs. For example, Hotstuff for optimistic responsiveness & linear view-change, Flexible BFT for diverse commit rules, CBC Casper for the win, etc.

Safety and liveness

We can adopt a blockchain protocol for the FG chain in a composable way because the modification is only about the application-level validity condition of blocks. Notably, the “checkpoint must be received \Delta

ago” rule guarantees that the timing of accepting a FG block is not affected assuming \Delta

synchrony bound.

Therefore, the safety of the FG chain and the validity condition of FG blocks guarantees that the checkpoints specified by finalized FG blocks are consistent, and hence the target chain has safety. Also, by the liveness of the FG chain, new blocks in the target chain are continuously being specified as checkpoints and finalized.

Embedding the Finality Gadget

In the construction of a hierarchical finality gadget, a few on-chain mechanisms of the original protocol adopted for the FG chain must be “embedded” in the target chain.

Validator rotation

Any blockchain protocol with a (usually PoS-based) on-chain validator rotation mechanism can be used as the FG chain with a dynamic validator set, by simply moving the rotation mechanism into the target chain. Specifically, participants submit an entry (resp. exit) transaction on the target chain to become (resp. resign as) a validator. The validator set at a block B

in the FG chain can be calculated by the state of the target chain at the checkpoint specified by B

.

It might be necessary for the target chain to have on-chain finality

, i.e., the info about up to which block is finalized for validator rotation. For example, eth2 has on-chain finality for inactivity leak. Similarly, we can implement on-chain finality by requiring the target chain to include FG blocks and votes (Footnote 3).

Slashing

One approach to slash equivocating validators on the target chain efficiently is to support FFG-style stateless slashing condition so that it becomes sufficient to submit a pair of equivocating votes on-chain to prove equivocations. We conjecture that Chained-Tendermint/Hotstuff can support this by requiring votes to have a few additional fields like FFG. Also, an instantiation of CBC Casper has a stateless slashing condition.

Related Work

Casper FFG

We can consider the “checkpoints tree” in FFG (See Figure 1 in the original paper) is a virtually

constructed FG chain. The safety mechanism of the virtual FG chain is fundamentally chained-Tendermint (Footnote 4). Note that unlike FFG, the checkpoints in our approach are not pre-defined.

Hierarchical sharding

The hierarchical finality gadget is similar to the hierarchical structure of Eth2 and CBC Casper sharding, where shard blocks are referenced by the parent chain and finalized if the referrer is finalized. (The actual finality condition is more complicated because of sharding specific reasons e.g., fraud proofs.)

Application for Ethereum 2.0

Benefits

Casper FFG is a bit hard to guarantee liveness (at least in the traditional network model, due to the bouncing attack). Our framework potentially enables us to design a finality gadget in a way the safety/liveness proof becomes more intuitive.

Moreover, if we adopt a responsive FG chain (Footnote 5), the latency to finality only depends on the network speed, rather than an in-protocol time parameter like epochs. This is similar to GRANDPA.

Merging two layers

Eth2 “merges” the finality gadget and the target chain; votes (called attestations) are used for both Casper FFG’s finality and the fork-choice rule (LMD GHOST), reducing the overhead. To merge two layers in our proposal, we choose a lock-step FG chain which proceeds in epochs and replaces the data fields of attestations used for FFG with the ones for the FG chain (Footnote 6), sacrificing the responsiveness.

Footnotes

1. The synchrony assumption for eth2’s safety is weaker than complete asynchrony due to inactivity leak (see thread), but this is a trade-off about the design of validator rotation, not the finality gadget approach.
2. Considering the motivation of finality gadgets to handle a large validator set, we assume the block production speed of a FG chain is slower than the target chain. Nonetheless, if there is no new FG block later than the parent’s checkpoint due to an exceptional case, the checkpoint can be the same as the one in the parent.
3. The size of a FG block is assumed to be smaller than ordinal blockchains because it does not include transactions.
4. However, the liveness strategy of the virtual FG chain in FFG is different from chained-Tendermint; its lock-step execution (i.e., validators proceed in epochs) and leaderless, relying on the fork-choice rule.
5. There are various levels of responsiveness. For example, Tendermint can create a new block without waiting for a pre-defined period only if the previous round succeeds. Hotstuff guarantees responsiveness even with a view-change. FFG has neither of them.
6. The checkpoint specified by the head of the FG chain can conflict with the head of the LMD GHOST. For this, we can (i) allow attestations to support a block in LMD GHOST, which conflicts with the checkpoint it votes for, or (ii) modify the definition of the fork-choice rule to start from the checkpoint specified by the head of the FG chain.