
Decoupled Consensus
Specifications

Prereqs

- Create an EIP for Decoupled Consensus
  - The consensus-specs require an EIP number for features
  - The EIP can be a basic overview of the idea, no code
  - For example, see PeerDAS (EIP-7594)
- Finish/prove specifications for Goldilocks in Lean
  - I believe there is still some work to do on this
- Assemble team for writing Python specifications
  - Mikhail/Francesco/Justin

Python or Lean?

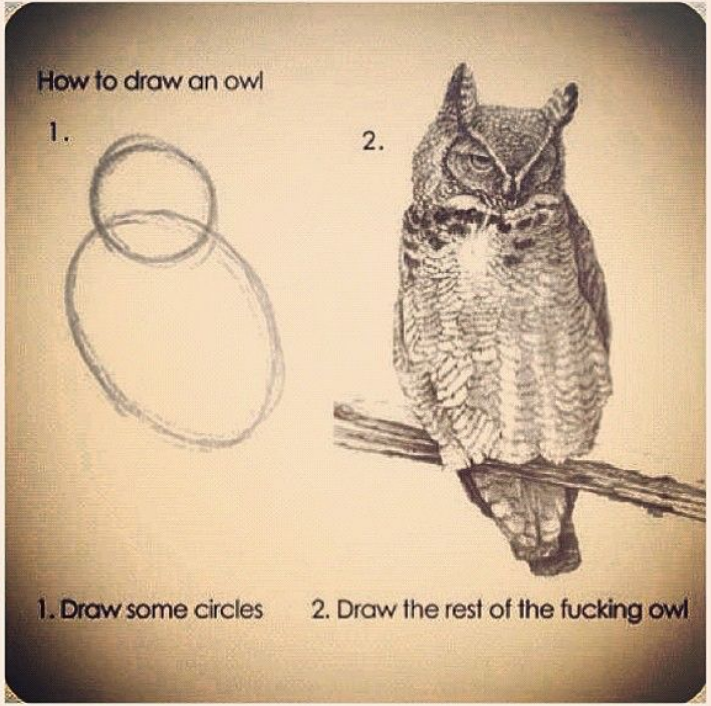
- Let’s do both!
- Manage Lean specs & proofs in a separate repo
- For now, keep consensus-specs as Python only
- Prove equivalence of Python spec to Lean spec
- Mixing both languages in consensus-specs will be hacky

Progress through iteration

- The initial specs do not need to be perfect
- Get a decent-enough feature merged asap
- Small pull requests are easier/faster

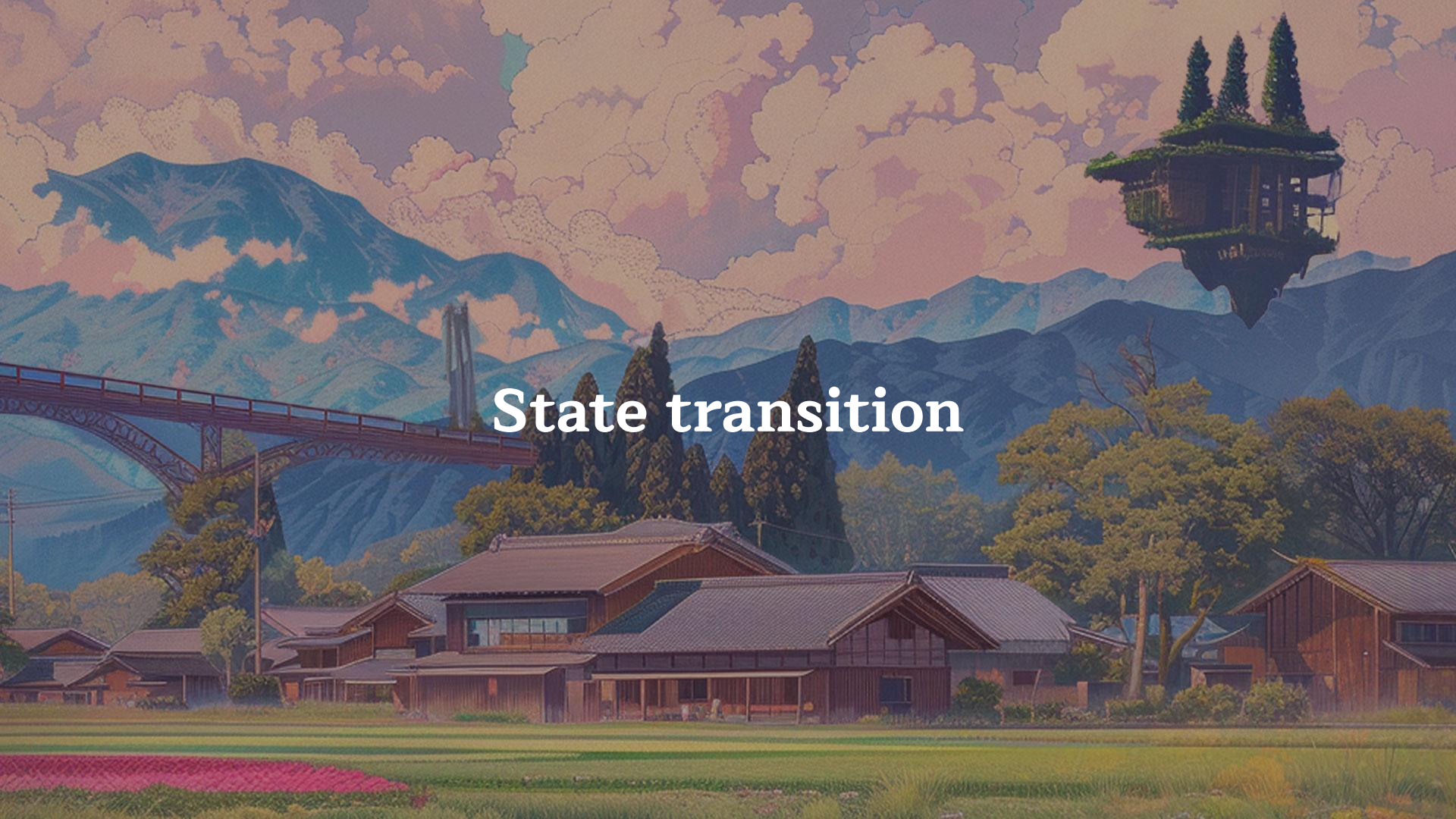
How to organize it

- Create specs/_features/eipXXXX
- Define specs for FG in finality-gadget.md
- Define specs for AC in available-chain.md
- Update many functions in other components
- ...



Testing

- Start by simply disabling everything that fails
  - This will probably be a lot
- Define new test formats instead of forcing a square into a circle
  - Otherwise, it complicates things in Hegota
- Take advantage of compliance tests
  - Make passing these a requirement
  - Include them in the standard test bundle
  - Provide nightly runs with other seeds for the latest release


State transition

Epochs and rounds

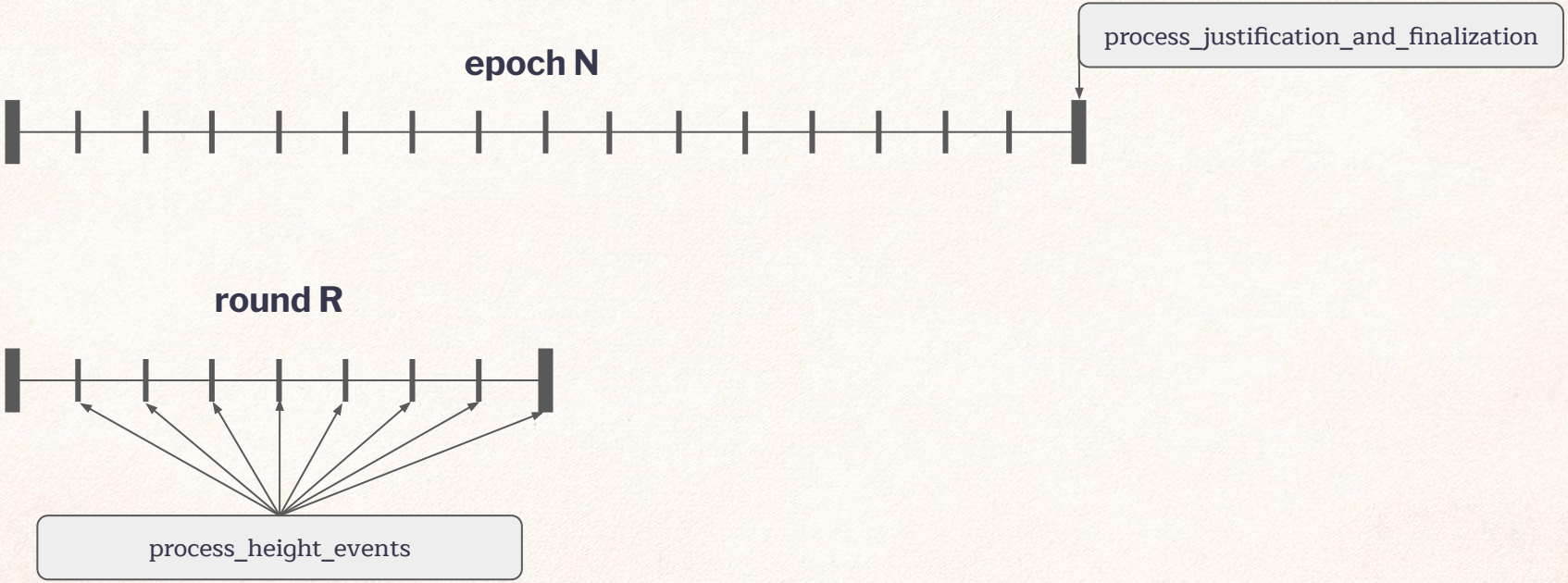
SLOTS_PER_EPOCH = 32

- Validator management
  - activations, exits, consolidations
  - effective balance updates
- Randao
- Historical summaries
- Correlation penalties
- Proposer lookahead
- Builder payments
- PTC computation

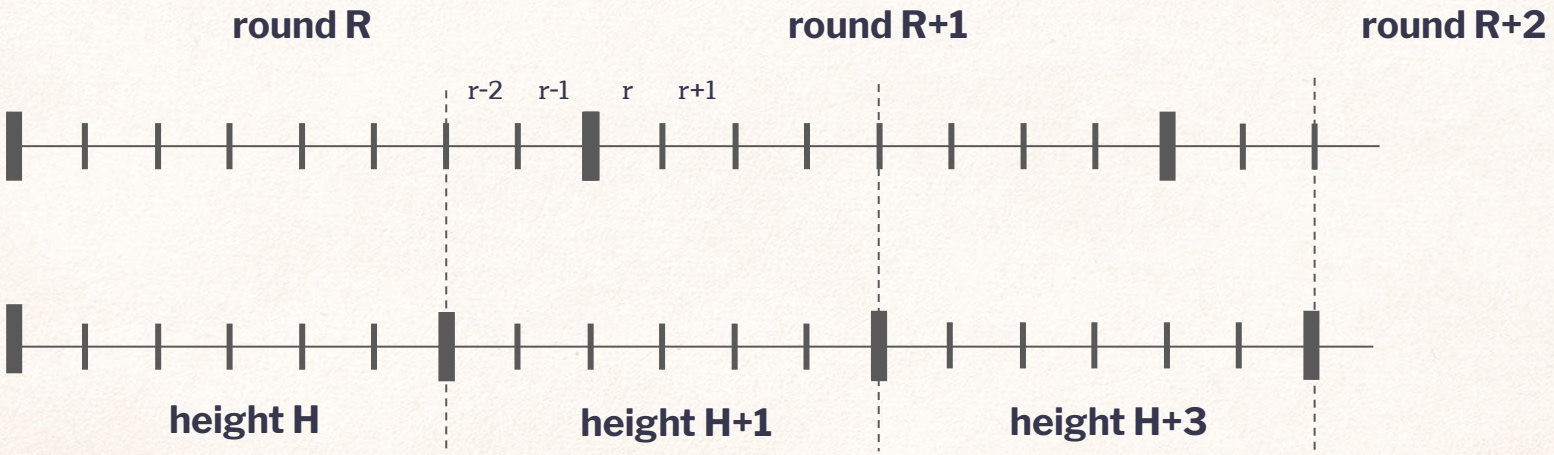
SLOTS_PER_ROUND = 8

- Attestations
- Rewards and penalties
- Inactivity leak

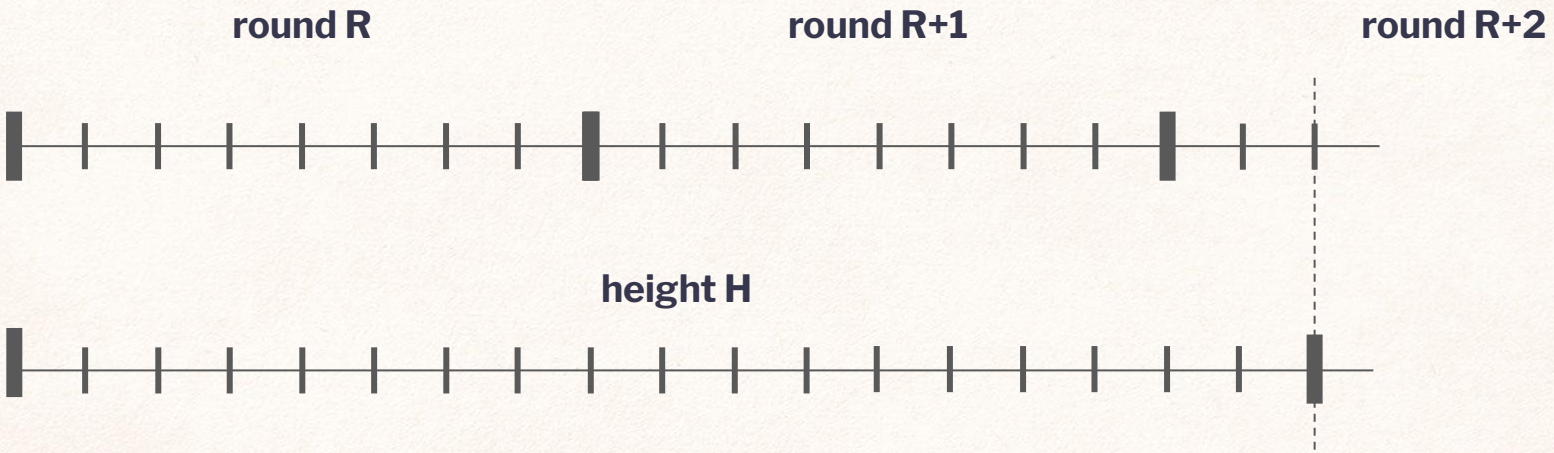
Justification and finalization



Rounds and heights



Rounds and heights. Unhappy case



What is needed?

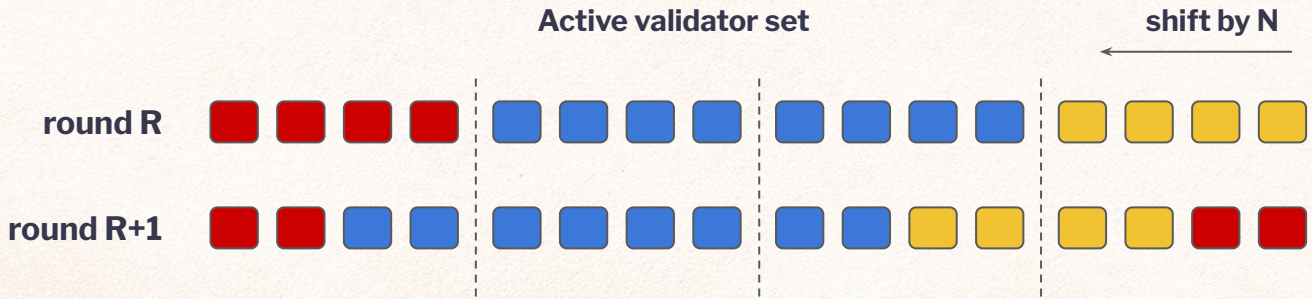
- Strong source of a new target
- Smart committee rotation
- Extended participation tracking
  - count votes supporting previous target and finalize target in the round participation
  - **previous_round_participation**
    - track participation in previous round
    - rewards, penalties and leak processing
  - **current_round_participation**
    - track participation in current round
  - **current_height_participation**
    - process justification (progress) and finalization
    - avoid leaking and penalizing validators that has performed their duties for the height
    - lifetime: between a sub-round and a span of rounds

Committee shuffling

- Ensures that each committee is pseudo-randomly selected
- Splits validator set into subsets within an epoch
- Assigns the same validator to different slots in different epochs
- Aids encoding aggregation indices as a bitfield

FG committee rotation

- Splits validator set into subsets of similar size within a round
- Assigns the same validator to different slots in different rounds
- Validator position within the entire set depends on a round number



Observation: large staking entity may always be in the same committee or a span of committees

Aggregation indices

- indices = decode(state, attestation)
- state = dependent_state(att.slot, att.beacon_block_root),
  where **dependent_state** can be different for any two branches that diverged two epoch boundaries ago
- implementation and huge computational complexity during periods of non-finality

Stable indices encoding

- indices = decode(state, attestation)
- state = head_state | finalized_state
- epoch = epoch(att.target_root) | epoch(att.finalize_root) | epoch(att) - N
- sample a set of validators not yet exited at a the epoch:
  - state and epoch shouldn't be too old to allow inclusion of recently activated validators
  - the combination of epoch and state should provide a stable validator set across different branches
- given round(attestation) apply rotation mechanism to the sampled set

Builder payments

- ePBS keeps track of head votes included on chain during the last two epochs
- Include Goldfish votes on chain for the last two epochs
- Track participation in the BeaconState
    - 512 bits per slot, 4 kB in BeaconState

Builder payments. VRF selection

- Validator indices instead of bitfields
  - 512 bytes per slot, 32 kB in BeaconState
- Include Goldfish votes on chain for the last two epochs
  - Include selection proofs into blocks, 48 kB in block
  - Keep selection proofs cached and influence block processing for the last two epochs
- Change the builder payment logic

upgrade_to_dc

On DC_FORK_EPOCH:

- bootstrap a new FG from the outcomes of the FFG
- stop including old attestations on chain
- keep including old slashings on chain to preserve accountable safety

Alternatively:

- activate new fork logic and keep including old attestations on chain
- wait till TRANSITION_EPOCH with TRANSITION_EPOCH = finalized_epoch + 2
- bootstrap a new FG
- stop including old attestations and old slashings on chain

Next

- fork choice and its switch at the transition
- honest validator behaviour
- p2p interface
- fast confirmation
- optimistic sync
- checkpoint sync
- slasher
- beacon API
- light client and sync committees
- etc