Week 10 Research - Fork Choice

Guest speaker

Francesco, Research at Ethereum Foundation

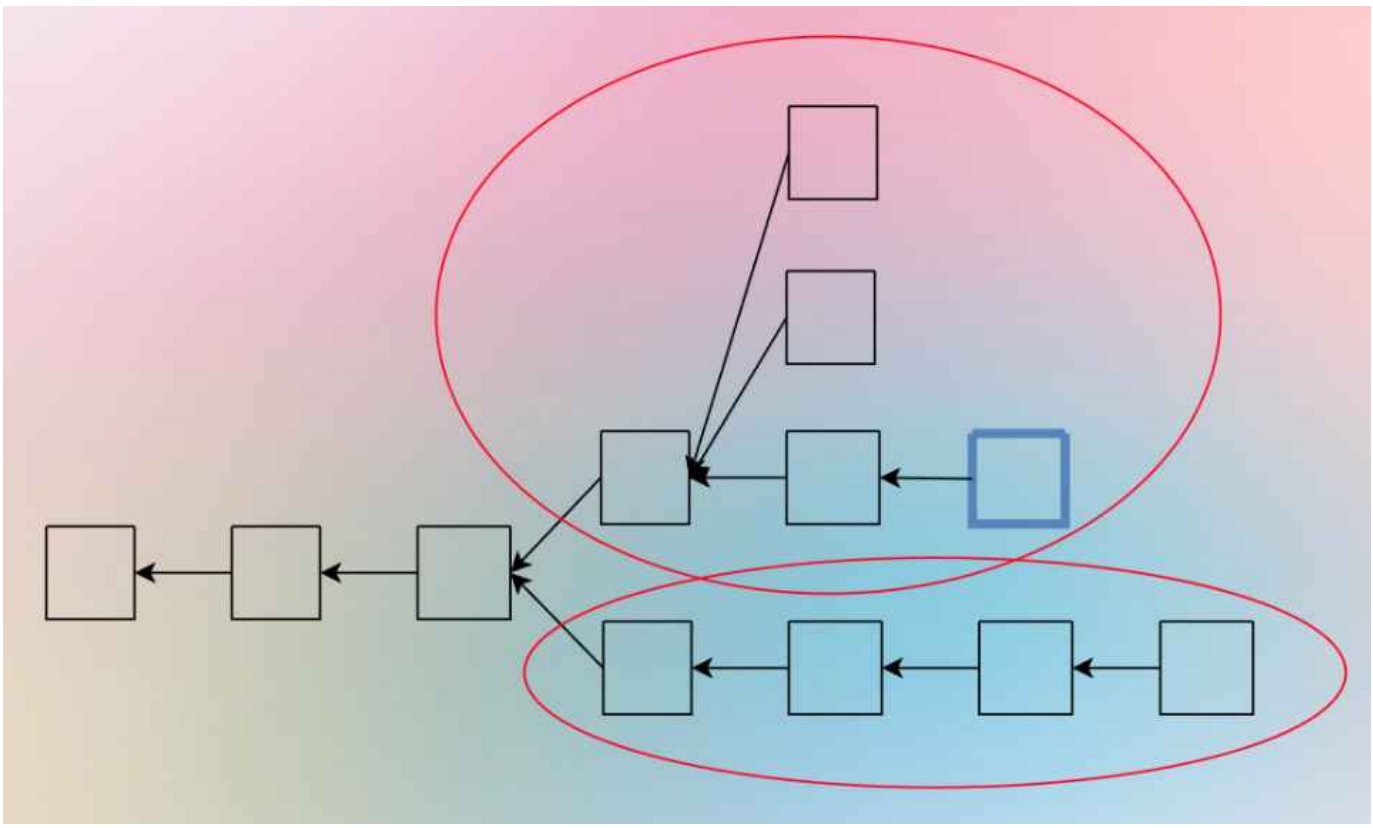
Slides link: https://docs.google.com/presentation/d/1Hrk-0x7N18qHwy9d7DeONdOVpA_6GPOqG7xxf6TtaGw/edit#slide=id.g2ab6d86c14d_0_208

Summary notes

- Edited by Chloe Zhu
- Online version: https://ab9jvcjkej.feishu.cn/docx/AMtAdcMOsoemsLx6txicLzOBnbg

Gasper recap: LMD-GHOST + Casper-FFG

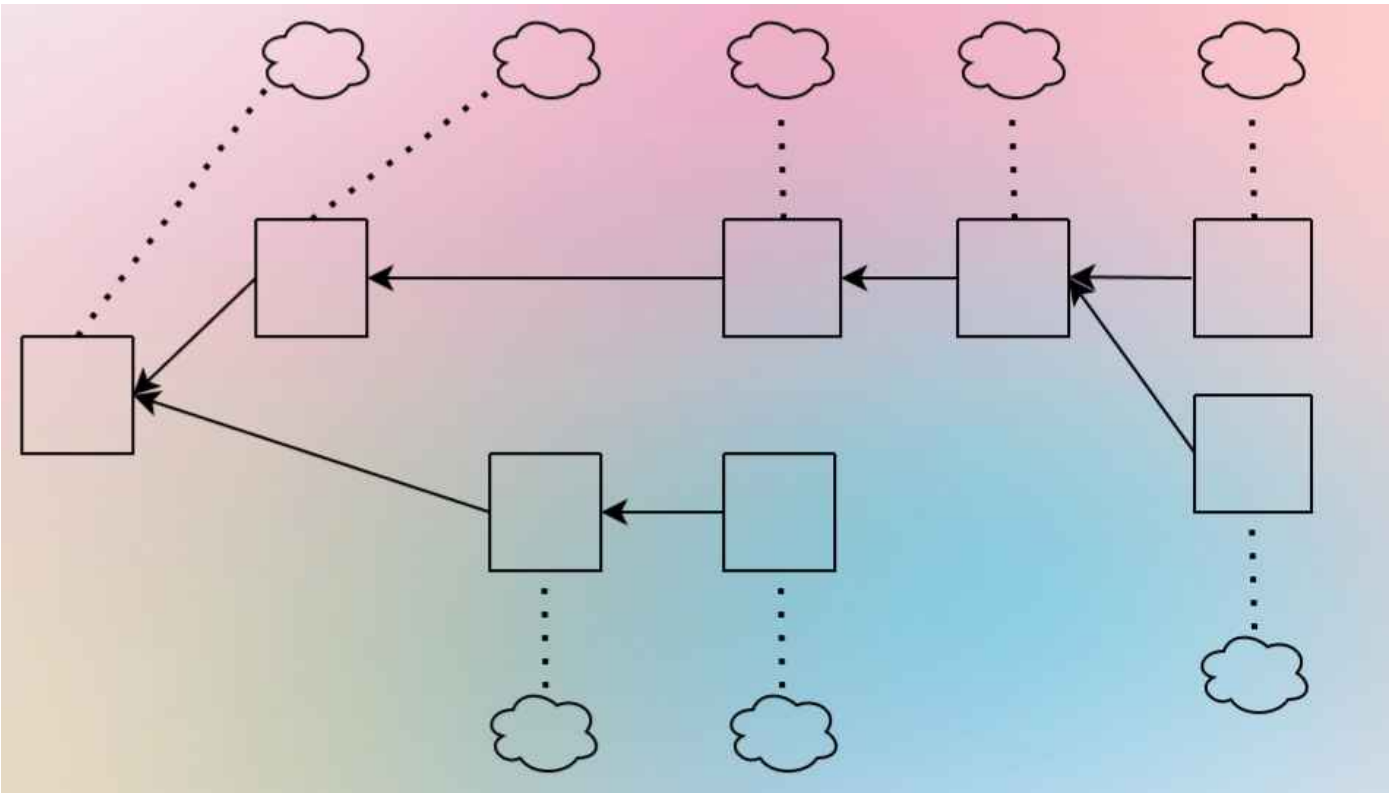
GHOST (Greediest Heaviest Observed SubTree)



- Overview
  - A fork choice algorithm to find the head of the chain, i.e. the canonical chain, given a block tree

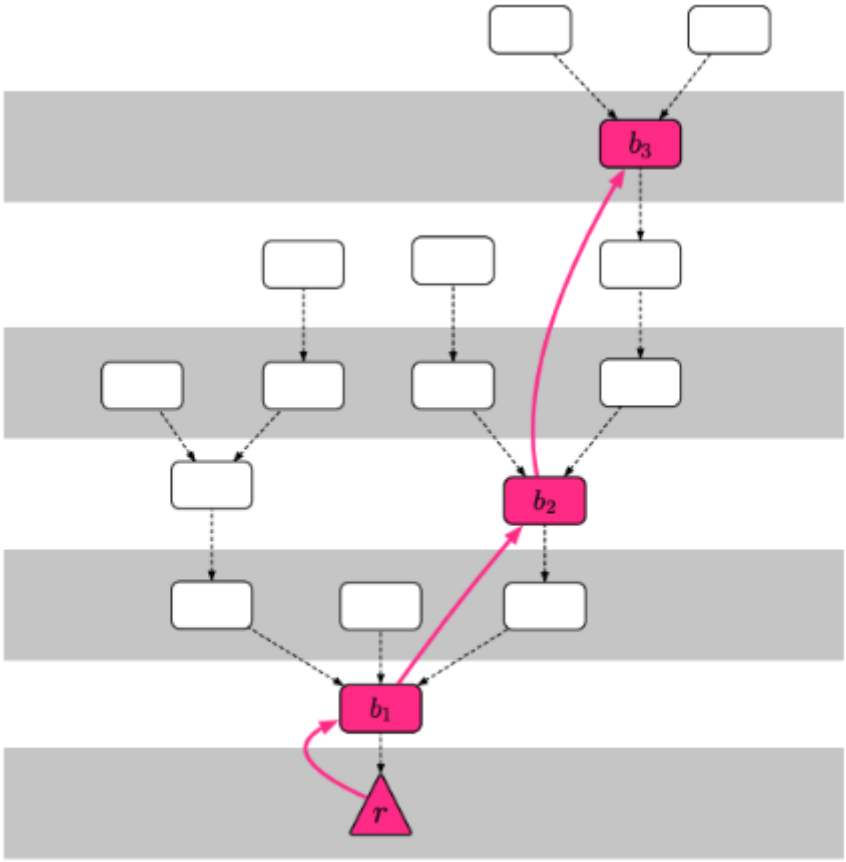
    - GHOST originally only operates on blocks, without votes etc.
- How does it work
    - When there is a fork on the block tree, the algo will choose the heaviest sub-tree as the chain head. Eg. In the above case, the upper tree has 5 blocks vs. the lower 4 blocks, therefore the upper sub-tree will be the canonical chain

LMD-GHOST (Latest Message Driven GHOST)



- Overview: Votes-driven GHOST
- How does it work
    - Instead of using block producing difficulty, LMD-GHOST uses votes & stake to measure weight

Casper FFG



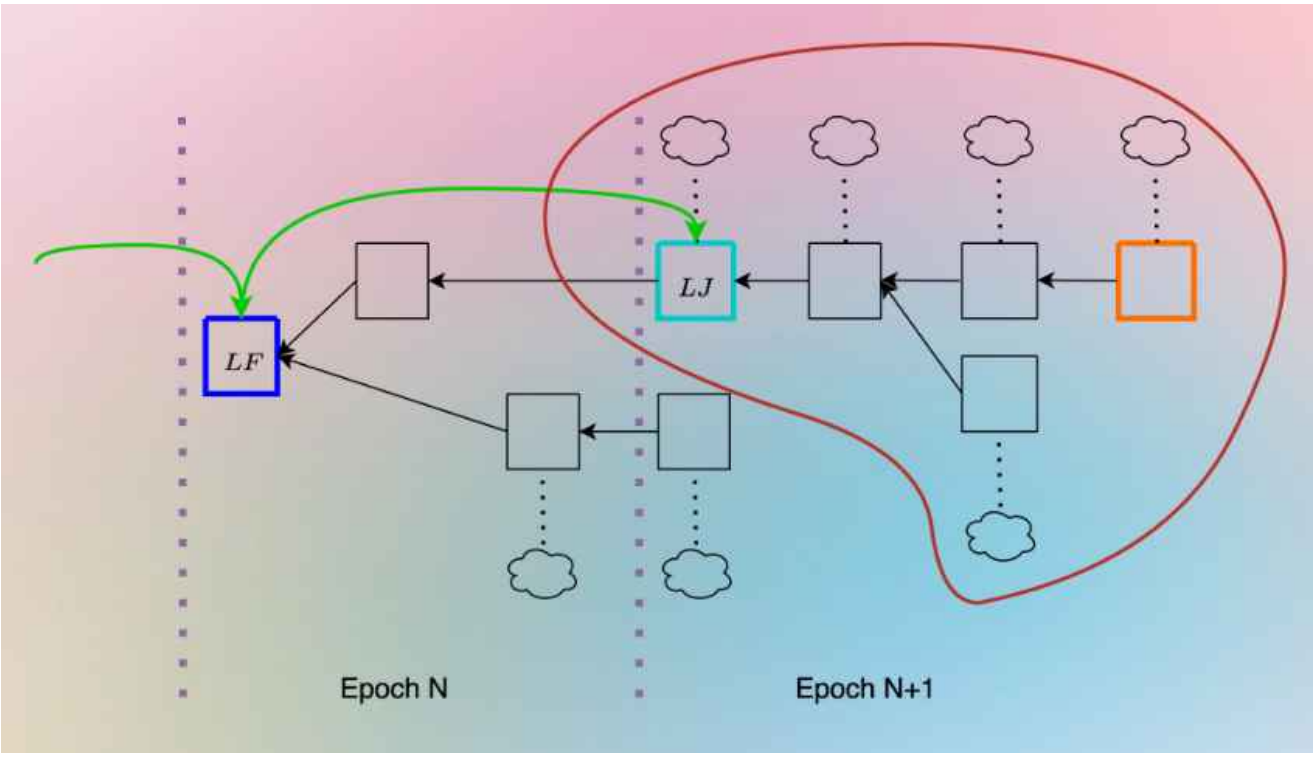
(c) The justified chain $r \rightarrow b_1 \rightarrow b_2 \rightarrow b_3$

- Overview
  - A finality gadget that works on top of LMD-GHOST
- How does it work
  - Start with the block tree, and introduce the concept and mechanism of checkpoints to finalize the block, in order to avoid reorgs
- Relevant paper
  - Casper the Friendly Finality Gadget: https://arxiv.org/abs/1710.09437

Ethereum today

- The consensus mechanism
  - 32 slots in an epoch
  - For each slot, a pseudorandomly elected committee votes is chosen from the validator set for LMD-GHOST and for Casper-FFG
  - Over an epoch, all validators have voted, concluding one round of voting of Casper-FFG
  - Relevant blogs/ paper
    - The beacon chain explainer you need to read first: https://ethos.dev/beacon-chain
    - Combining GHOST and Casper: https://arxiv.org/abs/2003.03052

- Modelled as an Ebb-and-flow protocol
  - Network model: Partially synchronous network with GST (global stabilization time), where synchrony begins
  - Participation model: Dynamic until a time GAT (global awake time), where it stabilizes
  - Goal
    - Dynamic availability: available chain to be safe and live under network synchrony and dynamic participation (GST = 0, GAT = ∞)
    - Finality: Finalized chain to be always safe and live after max(GST, GAT)
    - Prefix: finalized chain is a prefix of the available chain
      - Finalization = accountable prefix log
      - Chain tip = available full log
- Hybrid fork-choice: LMD-GHOST + Casper FFG



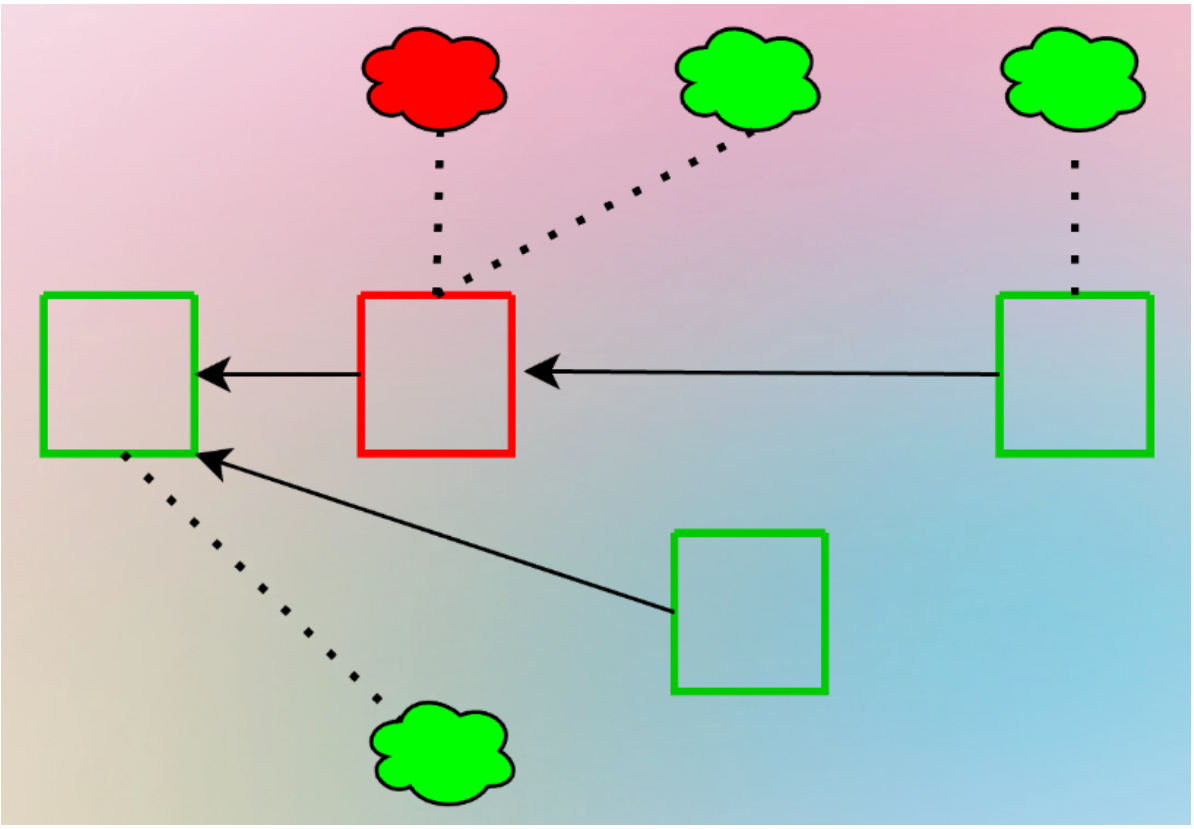
  - Goal: To ensure the property of prefix, the protocol needs to combine LMD-GHOST and Casper FFG
  - How does it work
    - Run LMD-GHOST from the latest justified (LJ) checkpoint

Problems of LMD-GHOST

Problem: Simple ex-ante reorg

- Overview: A reorg attack which is setup by an adversary in advance

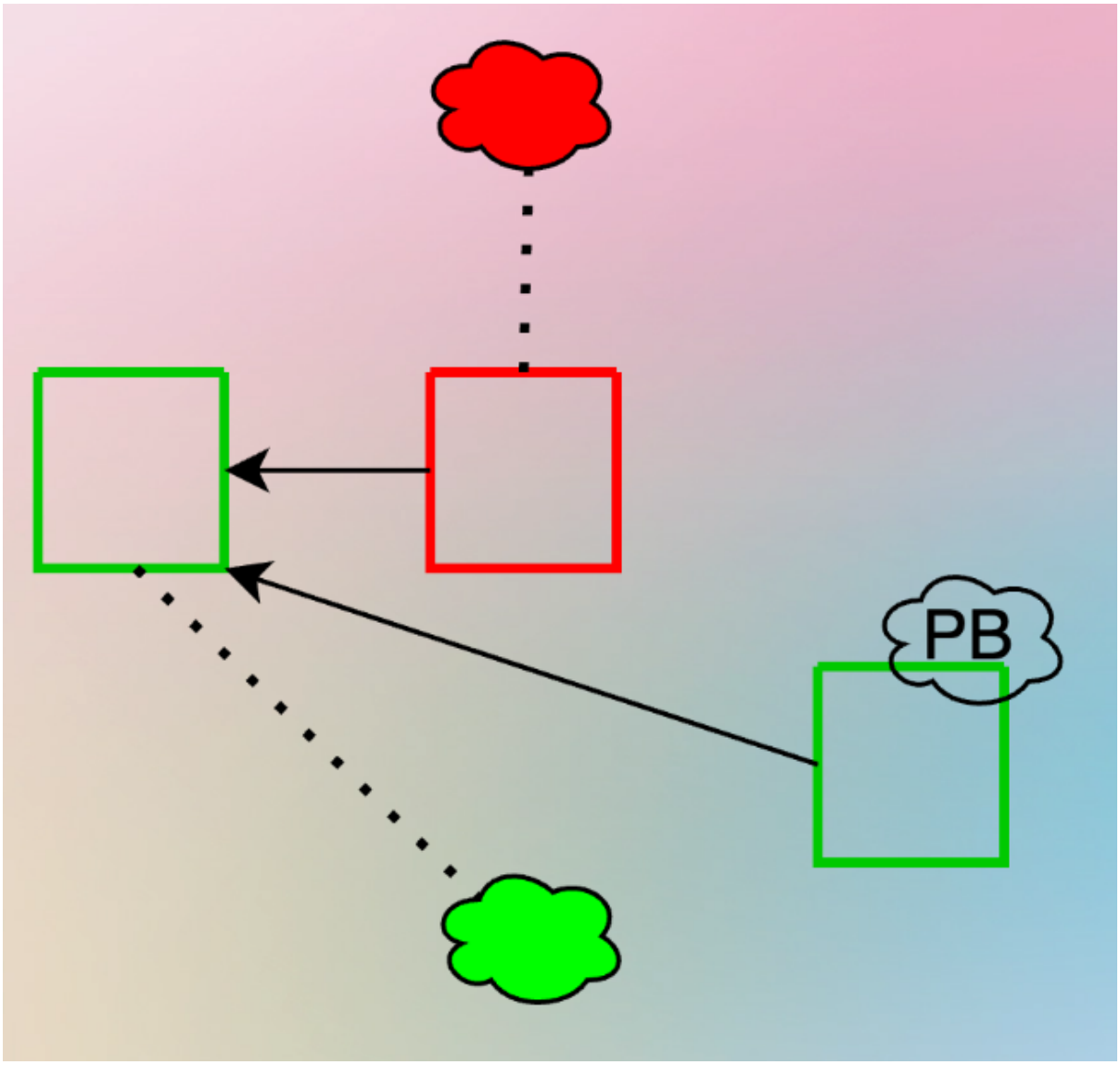
- How it happens:



  - An adversary withholds a block (red block) and can have just a single attestation to it
  - An honest proposer would build a block (gree block) while unaware of the withheld block, with no attestation yet
  - Then the adversary will reveal its block and attestation
  - Attesters see the red block & its attestation, so they start attest to that block
  - Thus, the honest proposal is forked out

Solution: Proposer boost

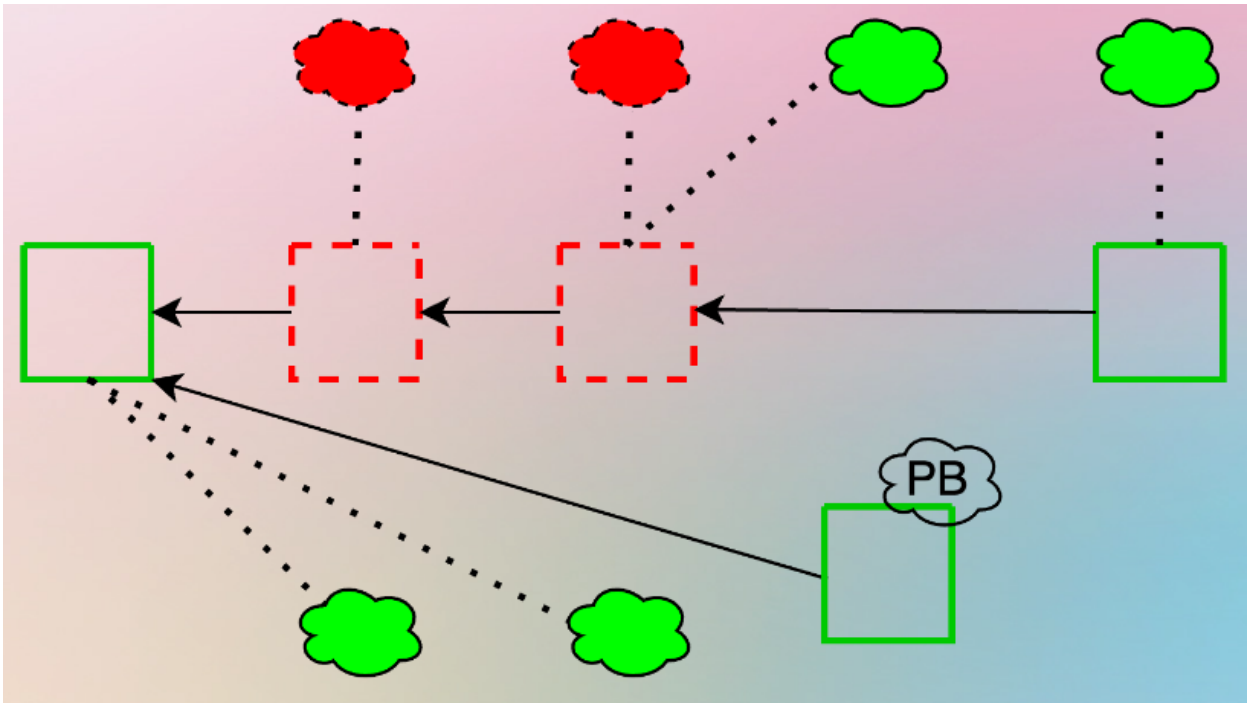
- How it works



- New block proposals have a temporary "weight boost" during their slot
- In practice, set the proposer boost to 40% of one committee's weight
- A hybrid way of combining GHOST (weight = blocks) and LMD-GHOST (weight = attestations)

Problem: ex-ante reorgs are still possible if controlling enough validators

- How it happens



- Adversary could control a few proposers in a row, withhold the block proposals & votes, and reveal them to overcome the proposer boost
- Then honest attester will move to the adversarial branch

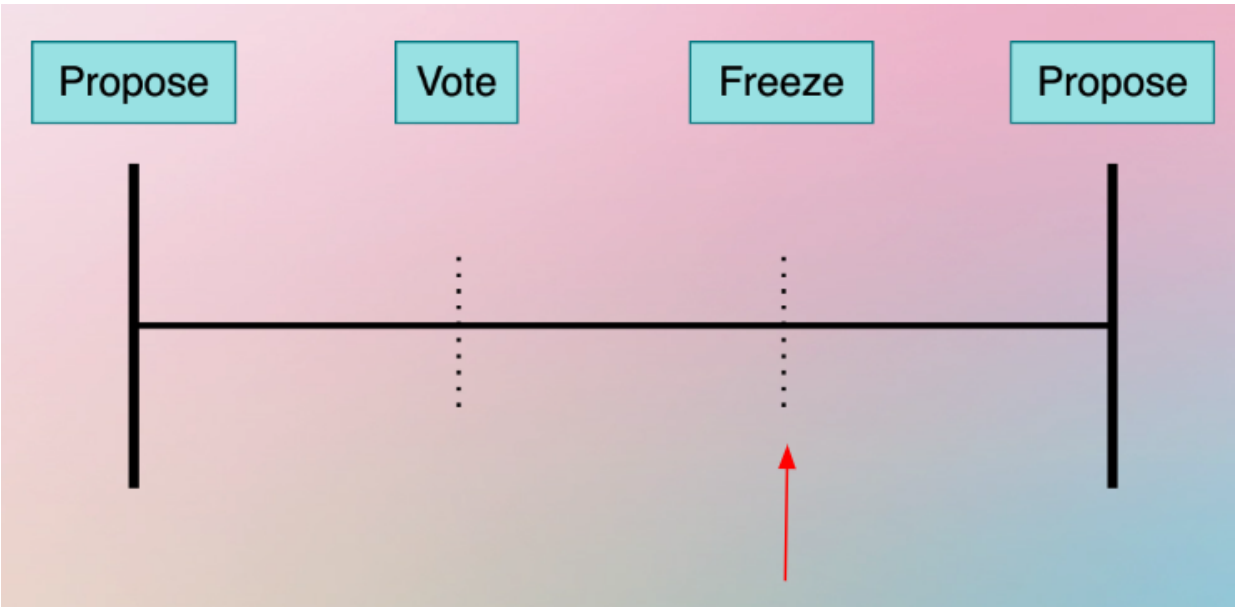
Problem: Balancing attacks

- How it happens
  - The honest attesters flip flop between two branches under the adversary attestation setup
- Solution: Proposer boost still works in this case

Designing a theoretically secure available chain

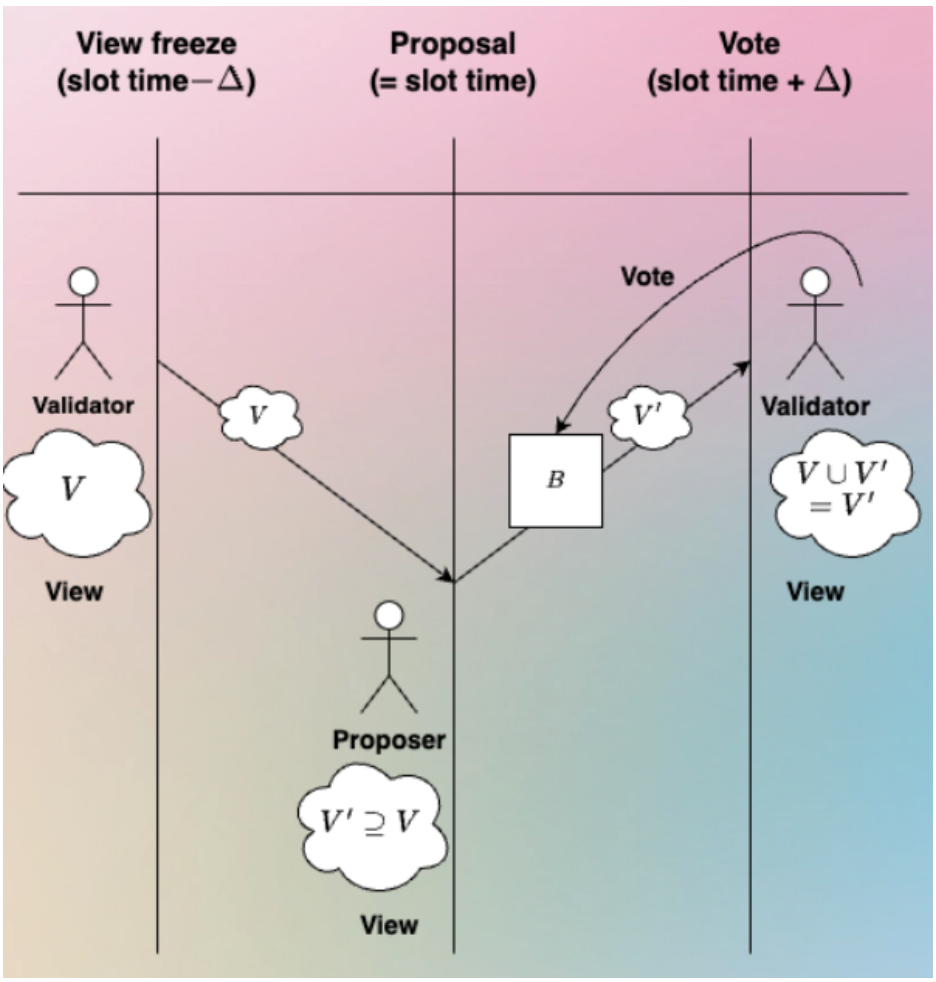
Improving on proposer boost: View-merge

- Goal
  - Whenever an honest proposer shows up, their proposal is voted by all honest attesters
- Introducing Freeze



    - Introduce a new phase in the slot, called "Freeze", where validators freeze their view
    - They buffer new attestations until after the next round of voting unless the proposer says otherwise

- View-merge synchronization mechanism
    - Mechanism



        - View freeze time

            - Validator will gossip its view V to the network
        - Proposal time
            - Proposer will get the validator's view V
            - The proposer should have a bigger view V' than V
            - The proposer will propose the block based on its view V', and send V' to validators
        - Vote time
            - Validator will merge its view V with the proposer's view V'
            - Since V' scope is bigger than V, then the final view would be V'
    - Result
        - No matter how powerful the adversary is, all the honest attesters will attest to an honest proposal, because all the honest attesters now see the same block tree as the proposer due to view merge.
- Problem: ex-ante reorgs still exist
    - Honest attesters vote for the honest proposal, but it's not enough to overcome the adversarial votes
    - The main reason is that **committees allow for weight accumulation over multiple slots**, even if LMD counts only one vote per validator
    - "Simple" solution is to let everyone vote for every slot, so that there is no possbility of cumulating over slots. However, the execution could be complicated as the number of validators keeps growing.

RLMD-GHOST

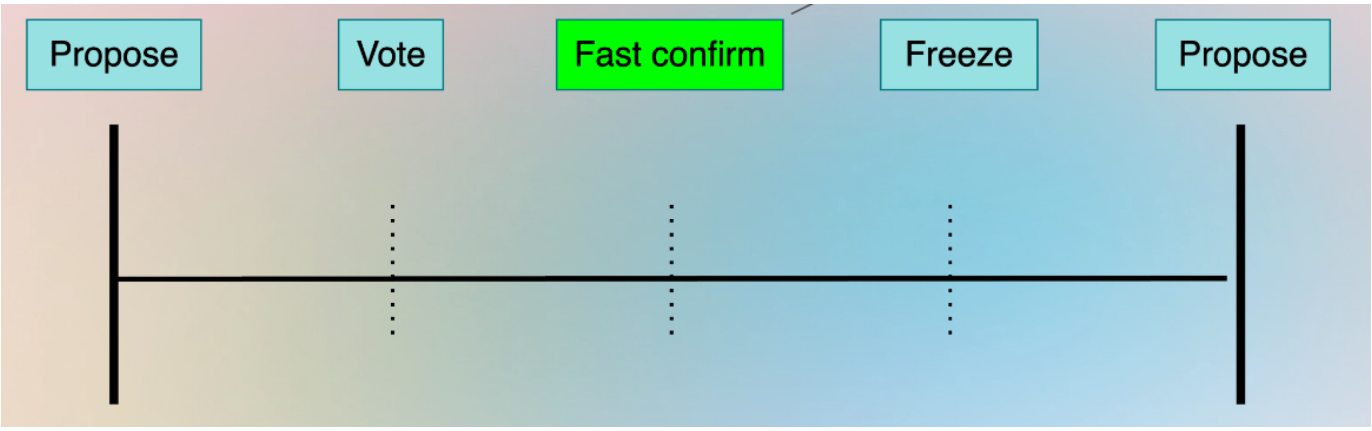
- The idea: Reorg resilience
    - View merge: Honest proposals are voted by all honest valdiators
    - No committees: If all honest validators vote for something, it stays in the canonical chain forever
    - With the above two, **honest proposals will never be reorged**
- Vote expiry (R = recent)
    - Attestations don't contribute fork-choice weight after some number of slots, i.e. expired
    - But they can always recover from a large portion of the validator set going offline

SSF protocol

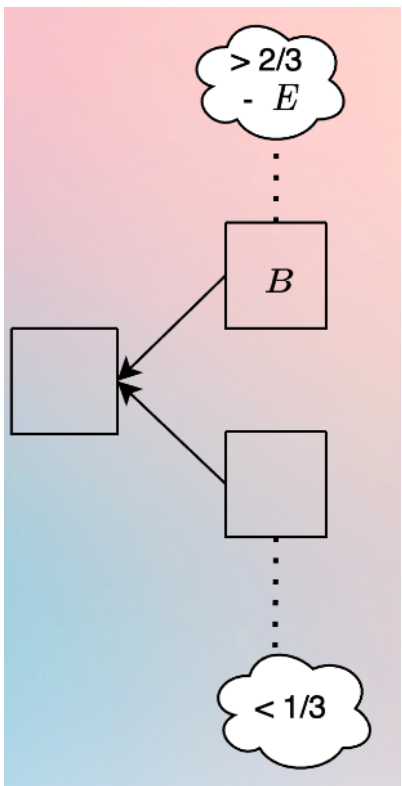
Key idea

- Only attempt to finalize blocks which are already confirmed by the underlying available protocol (i.e. LMD-GHOST)
- Finality gadget does not interfere with available protocol under permanent network synchrony
- The tricky thing is when recovering from periods of asynchrony, when the gadget might interfere

Add fast confirmation to RLMD-GHOST



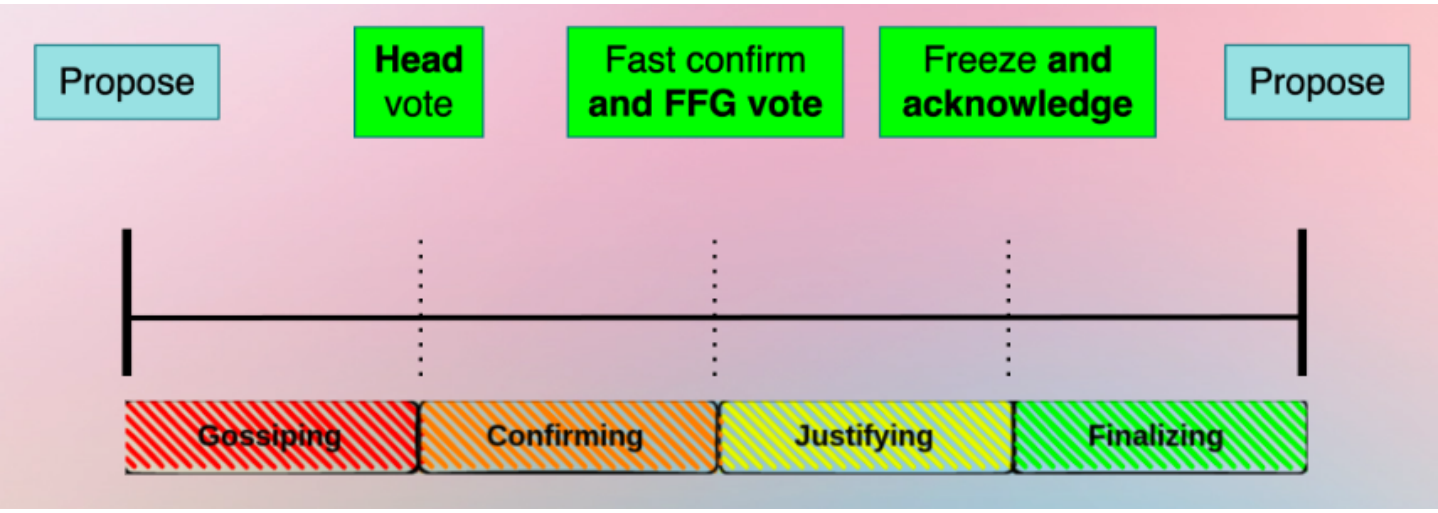
- Idea: Fast confirm any block which received a 2/3 quaorum in its slot, before view-freezing stage
- Security guarantees under synchrony & honest majority



  - If an honest validator fast confirms B, then every honest validator sees the ⅔ quorum for B before freezing its view

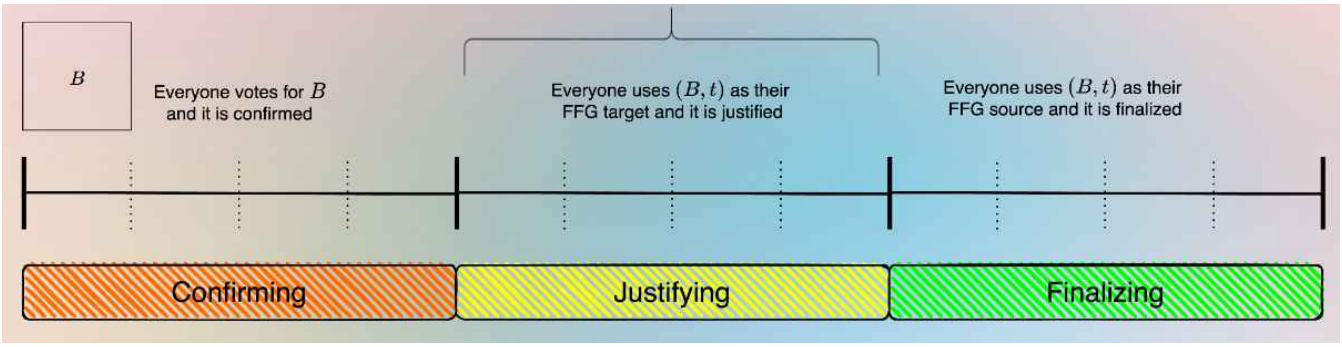
- Unless there are at least ⅓ equivocations (E), every honest validator sees B as canonical in the next slot and votes for it
- Every honest validator votes on the subtree of B in the next slot. B remains in the canonical chain forever if there is an honest majority

SSF protocol



- Stages within a slot
  - Fork choice:
    - Start from the latest justified checkpoint
    - Then run RLMD-GHOST
  - FFG votes:
    - Source of the vote = latest justified checkpoint
    - Target block = Highest confirmed block descending from justified (justified is always confirmed)
  - Acknowledgement:
    - If there is a new justified checkpoint, acknowledge it, committing to start your fork-choice from it
    - A quorum of acknowledgement = Finality
- Progression of honest slot t
  - Propose:
    - Honest block proposal B is made timely
  - Head vote:
    - All honest validators vote for B through view-merge, forming a 2/3 quorum of head vote

    - Fast confirm & FFG vote: A
        - ll honest validators get the quorum for B, fast confirm B and use it as their FFG target
        - A supermajority link LJ -> (B,t) forms
    - Freeze and acknowledge:
        - All honest validators justify (B,t) before freezing their view
        - Their fork-choice starts from B
        - They acknowledge (B,t), and a supermajority acknowledgment of (B,t) forms
    - Propose:
        - Everyone sees B as finalized
- 3SF: a chained 3-slot-finality protocol
    - Idea: A chain protocol to let validators only vote once in a slot, rather than 3 times under the above SSF model
    - How it works



        - Time period: 3 slots
        - 1st slot: Honest block proposal B. Everyone votes for B, and it is confirmed.
        - 2nd slot: Everyone uses (B,t) as their FFG target, and it is justified.
        - 3rd slot: Everyone uses (B,t) as their FFG source, and it is finalized.
    - Status
        - This topic is still an ongoing project.

Fork choice in the Ethereum Roadmap

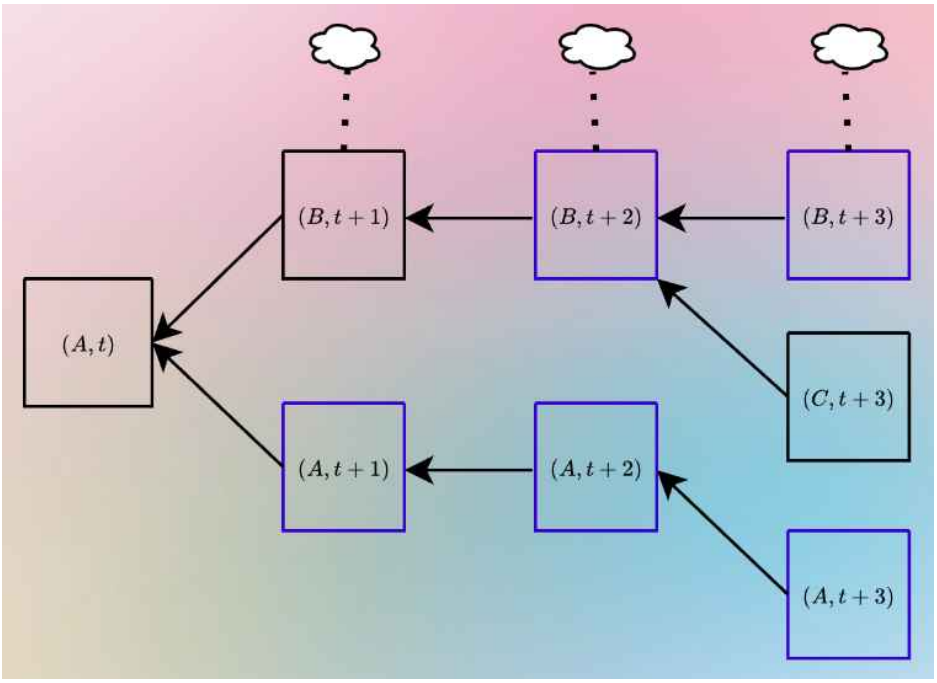
SSF (or fast finality) key features

- Optimally secure consensus protocol
- Fast confirmation of high value transactions, e.g. fast deposits on exchanges
- Faster br/idging

- Better L2 interoperability

Fork choice variant: (Block,slot) fork-choice : enshrining empty slots

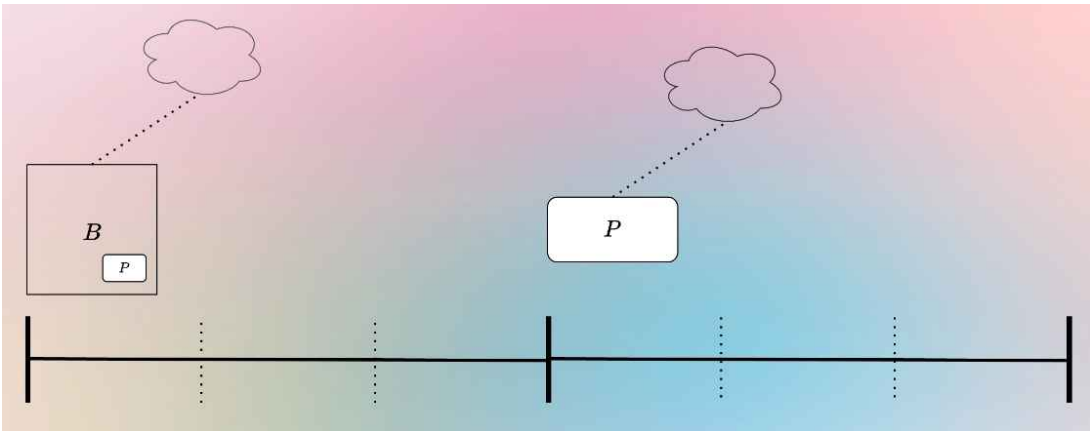
- Allow participants to vote against blocks through empty slot voting



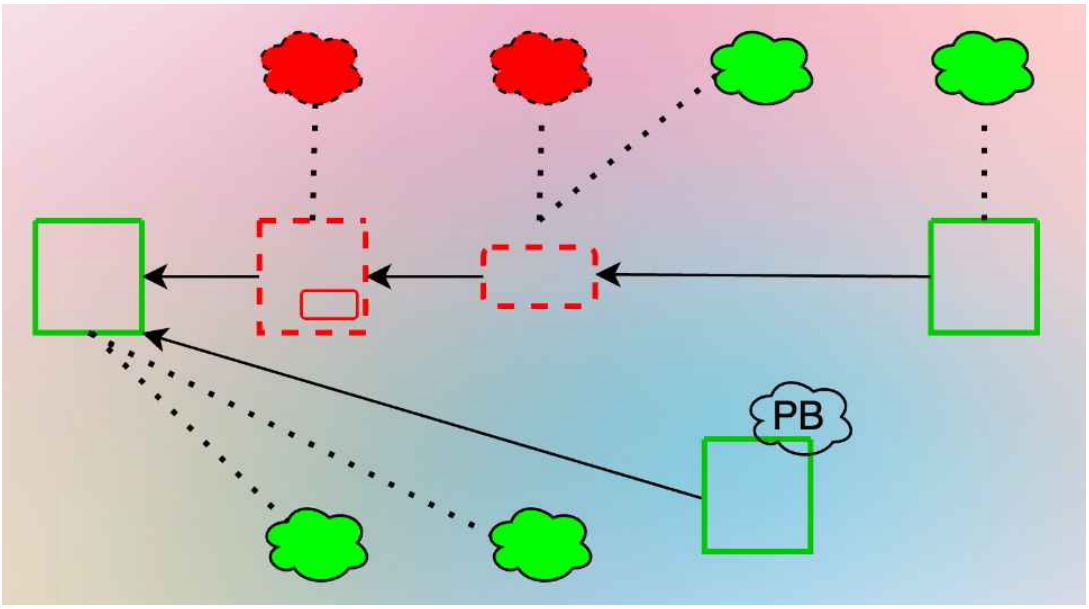
- Application
  - **Committee-enforceable properties**: If a proposal doesn't fulfill a certain property, the attesting committee could vote "against it" (for the empty block)
    - ILs satisfaction (one possible way of doing this)
    - Bid maximization (MEV burn)
  - **ePBS fork-choice**: Proposers accept bids from builders. If they don't publish them on time, the empty block wins and the builder is off the hook.
  - **DAS fork-choice**: If a block is unavailable, most honest validators will not vote for it, i.e. they will vote against it, for the empty block. Thus, no unavailable block should ever “look” canonical
- Main challenge
  - Higher latency could halt block production
  - To tackle this challenge, it probably needs a back off system so that the protocol can revert to producing blocks. This is still an open problem to work on.

Fork-choice in ePBS

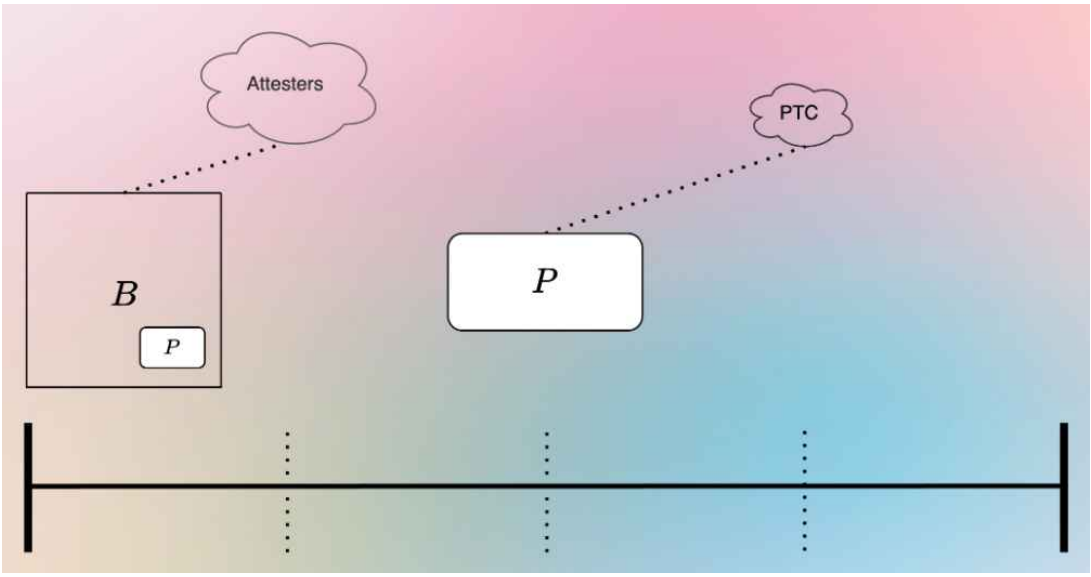
- Two-slot ePBS



  - The commitment part where the beacon proposer accepting the payload header happens in one slot
  - The reveal of the payload happens in another slot
- Problem: ex-ante reorgs in ePBS version



  - The adversial power is doubled in the ePBS version
  - As soon as the dishonest proposer controls one slot, it also controls the next slot
- Solution: One-slot ePBS with PTC (payload timeliness committee)



    - Introduce PTC to just vote on whether the payload is timely
    - However, this approach gives subpar guarantees to builders. Builders can be forced to reveal a payload without it becoming canonical.
    - Therefore, this is also an open question under exploration.
- Solution: RLMD-GHOST with ePBS
    - Due to vote expiry, there is no accumulation of votes. And all honest attesters will vote for the honest block.

Fork-choice in PeerDAS

- Data availability sampling (DAS)
    - Overview
        - We extend blobs, introducing redundancy which allows **reconstruction** of the full data whenever having 50%+ of the data
        - Checking availability of a blob only requires making sure that 50% of it is available
    - Ultimate goal
        - Full nodes only need to download a few chunks of data of their choice (samples) to verify that all the data is available
- Fork choice implications
    - If we do enough sampling before voting, we get a great global property
        - Only at most a small percentage ☒ of the honest validators will see unavailable data as available
        - If we have > ½ + ☒ honest validators, a majority always votes against unavailable blocks

- With the (block, slot) fork-choice, this means that no validator will ever see an unavailable block as canonical

Q&A

- If possible, can you elaborate on what's the BLS signature count limit / slot in SSF scenario?
  - Nowadays, the protocol aggregates c.30k votes per slot
  - The proposed idea is to have a 2-layer aggregation to have more committees operate in parallel. However, the main challenge is the verification of the the final huge BLS at the end, and the networking issue.
- Regarding the view-merge synchronization mechanism, what if the proposer is dishonest?
  - All honest votes will get to everyone before they freeze views. Then the proposer cannot really censor honest votes, it can only maybe confuse people if there are late votes. So the dishonest proposer couldn't have any more adversarial power.
- In the block-slot slide, does the ePBS refer the block/auction ePBS? How about slot auction ePBS?
  - Yes. It doesn't change much for other kinds of ePBS. For slot auction, it seems like it simplifies things but still have the same problem.
  - This can actually be an area to explore more.