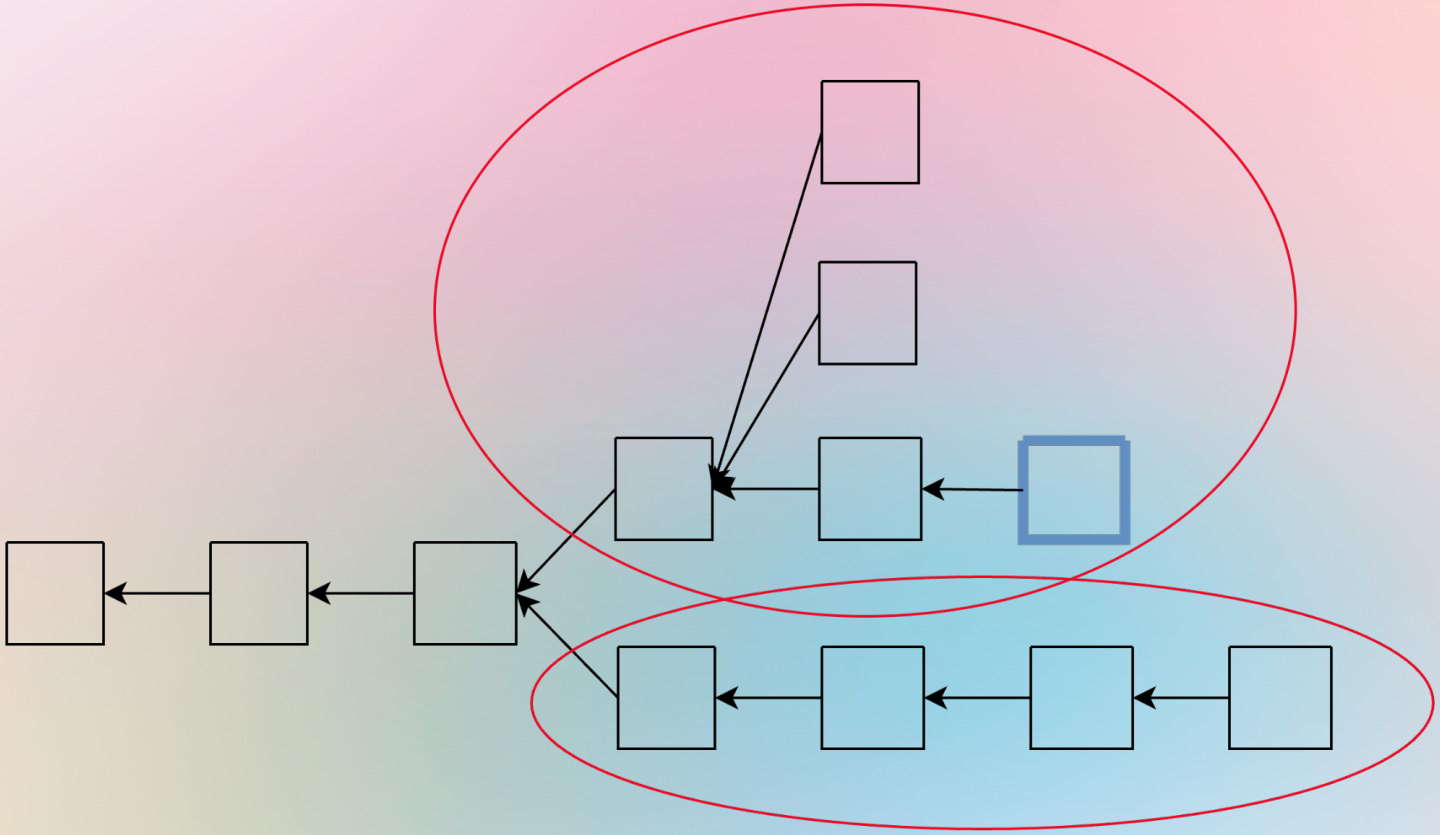
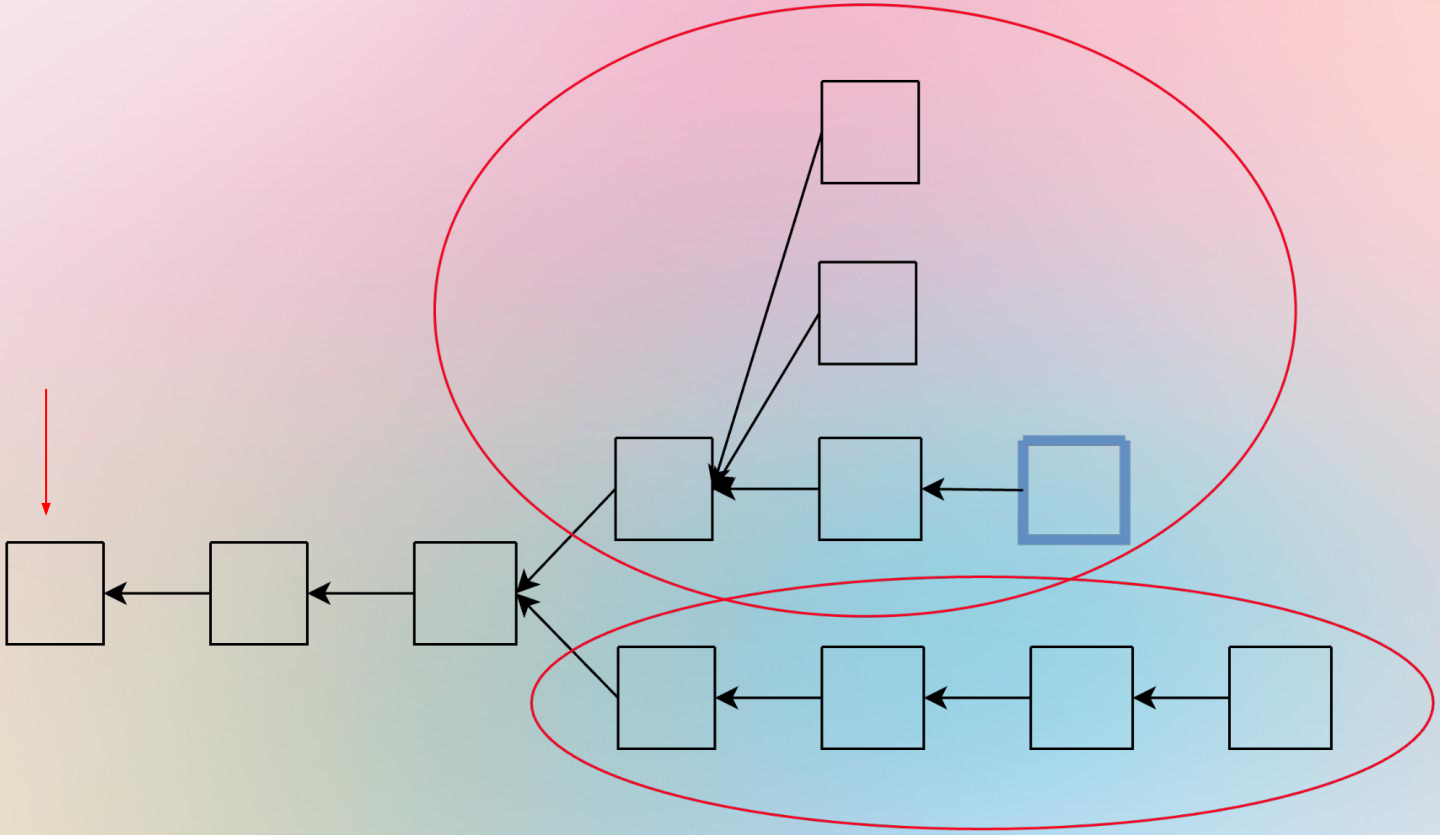
Gasper recap

Gasper = LMD-GHOST + Casper-FFG

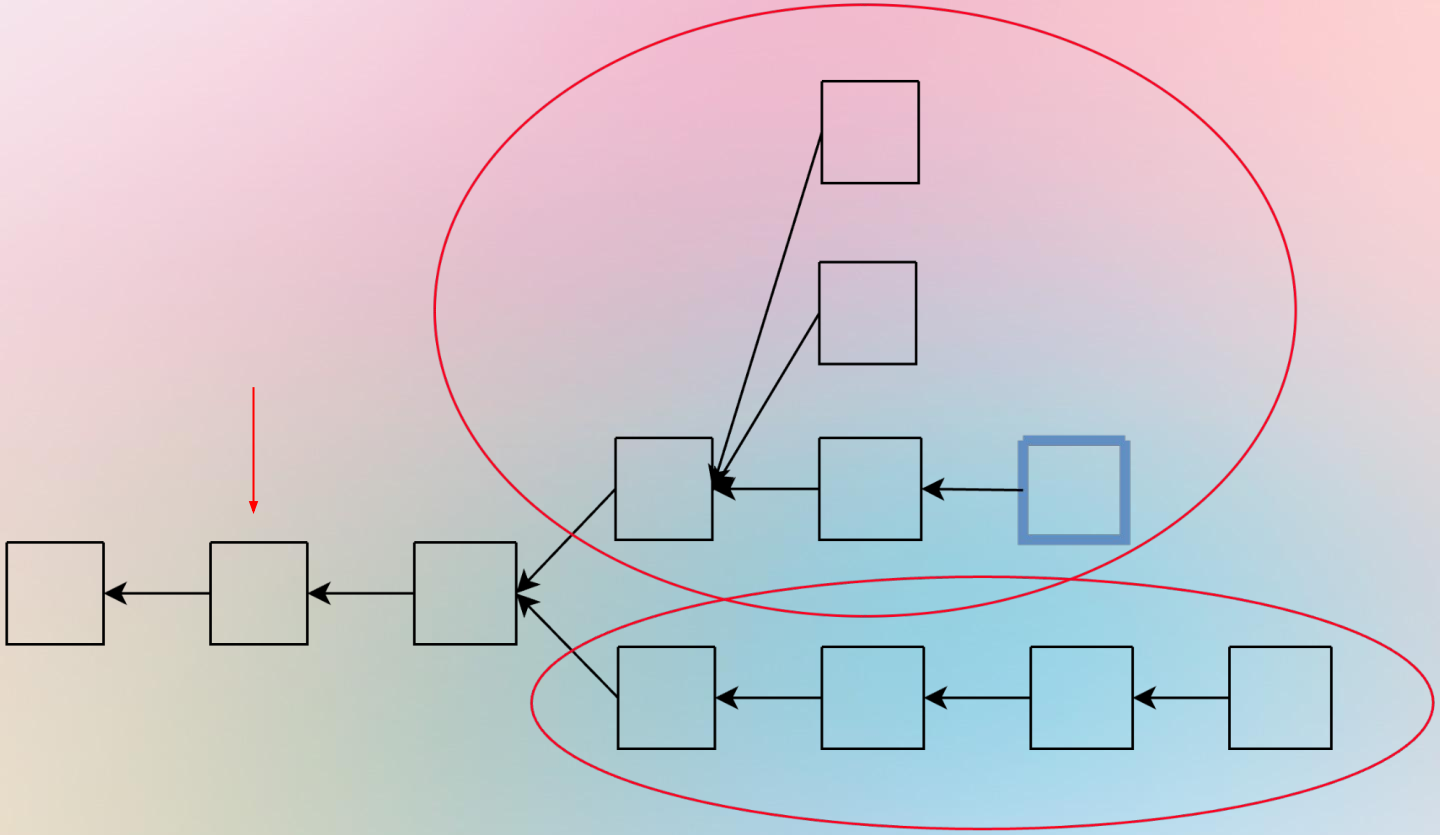
GHOST = Greediest Heaviest Observed SubTree



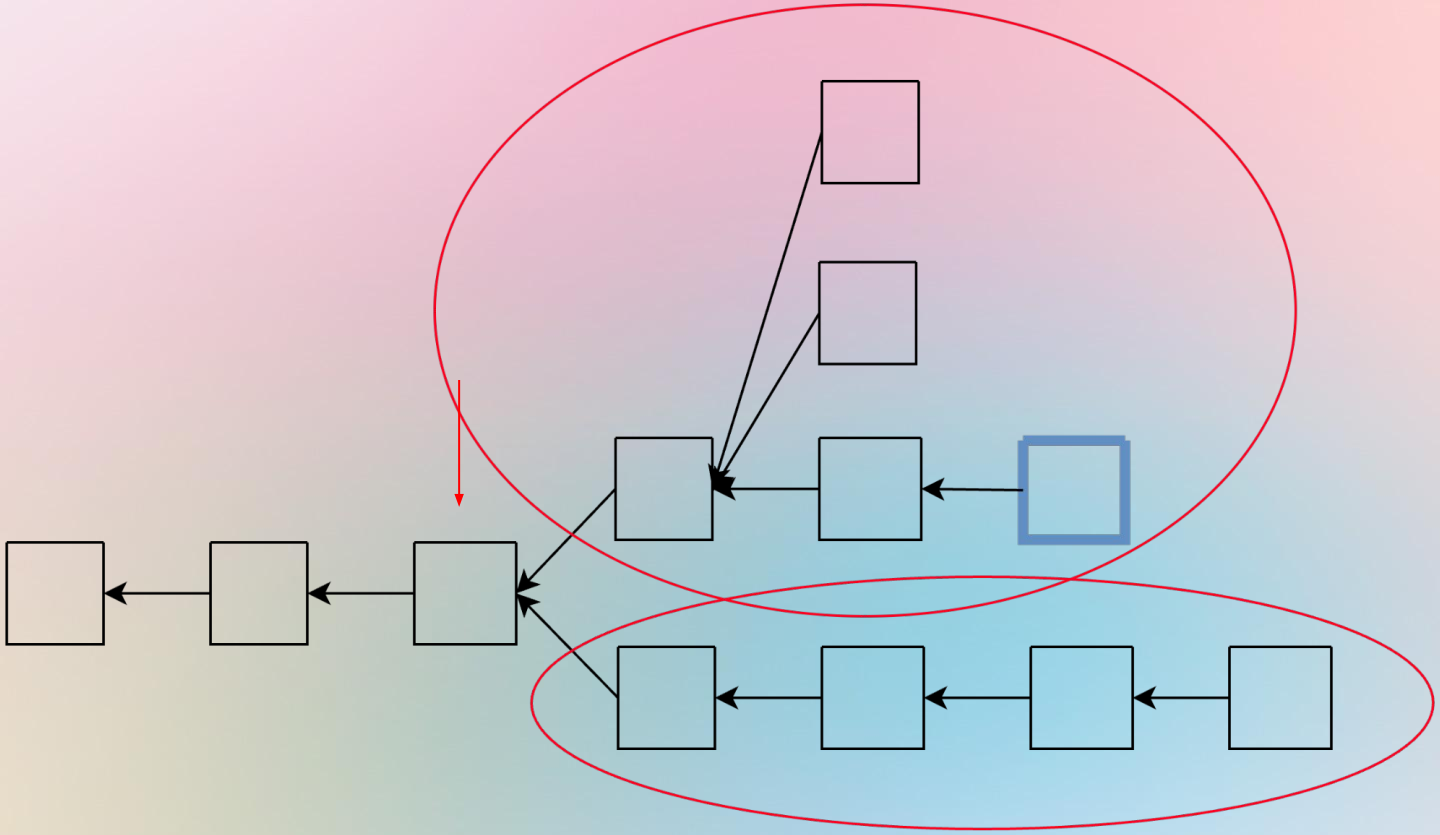
GHOST = Greediest Heaviest Observed SubTree



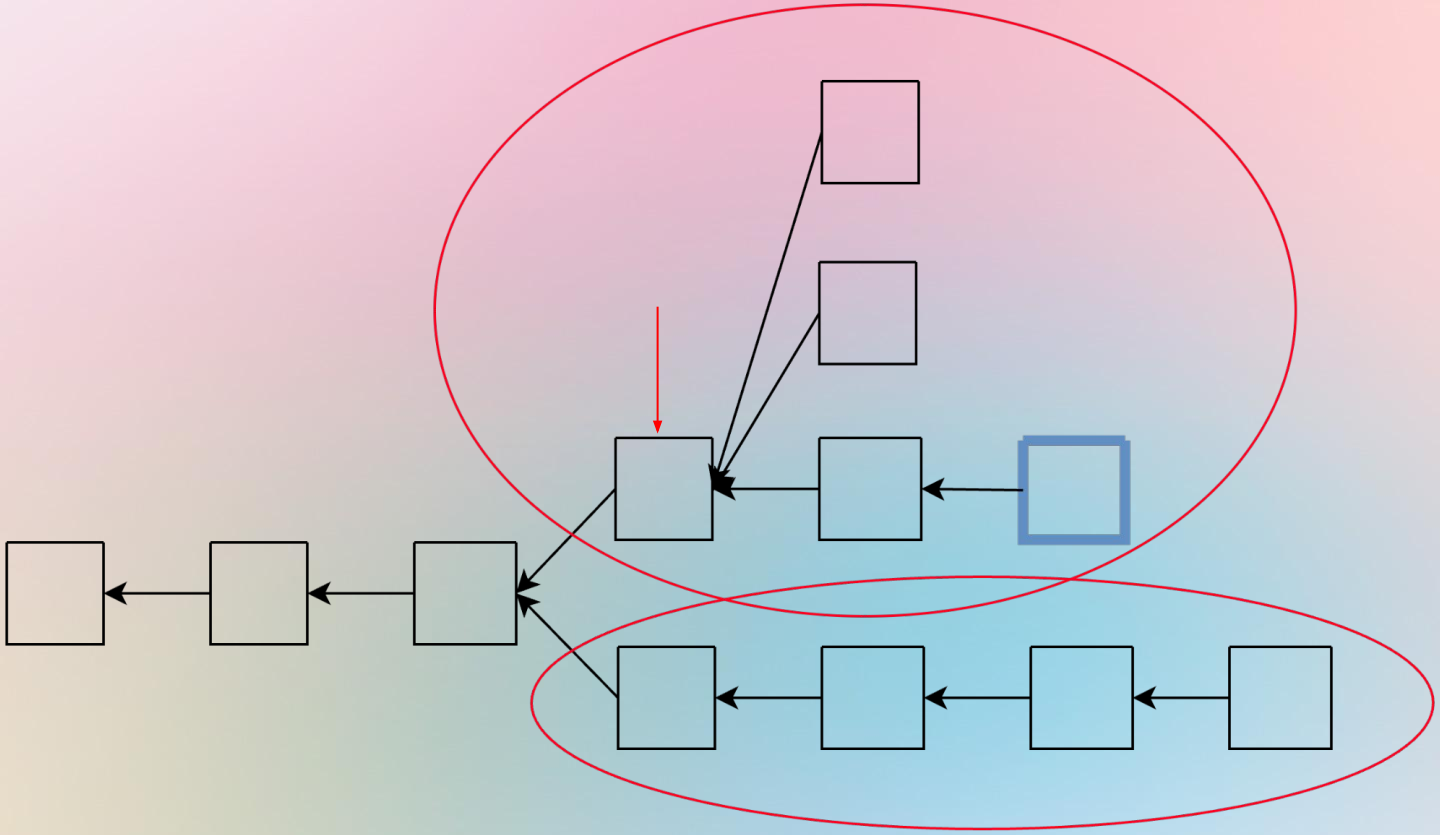
GHOST = Greediest Heaviest Observed SubTree



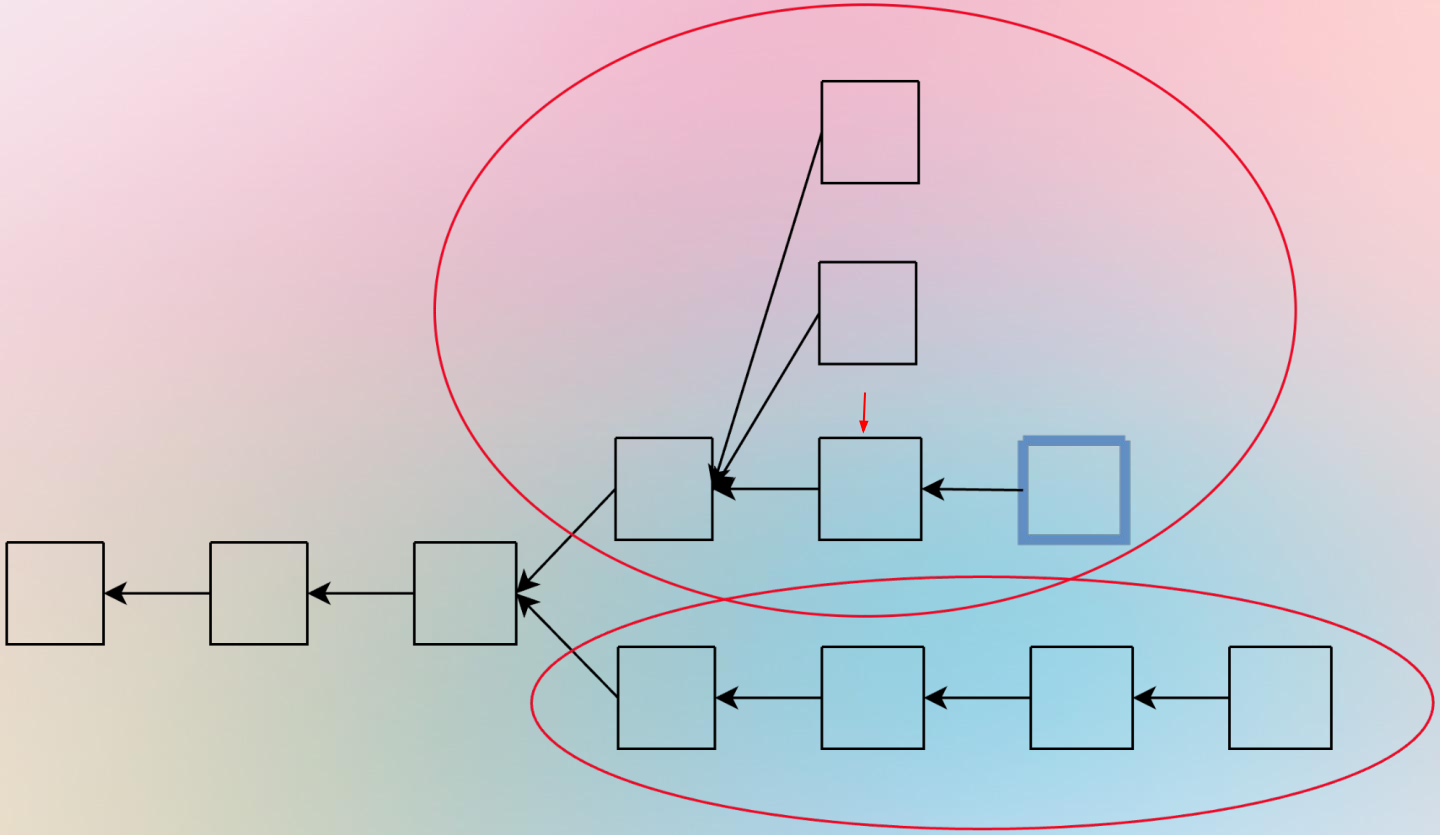
GHOST = Greediest Heaviest Observed SubTree



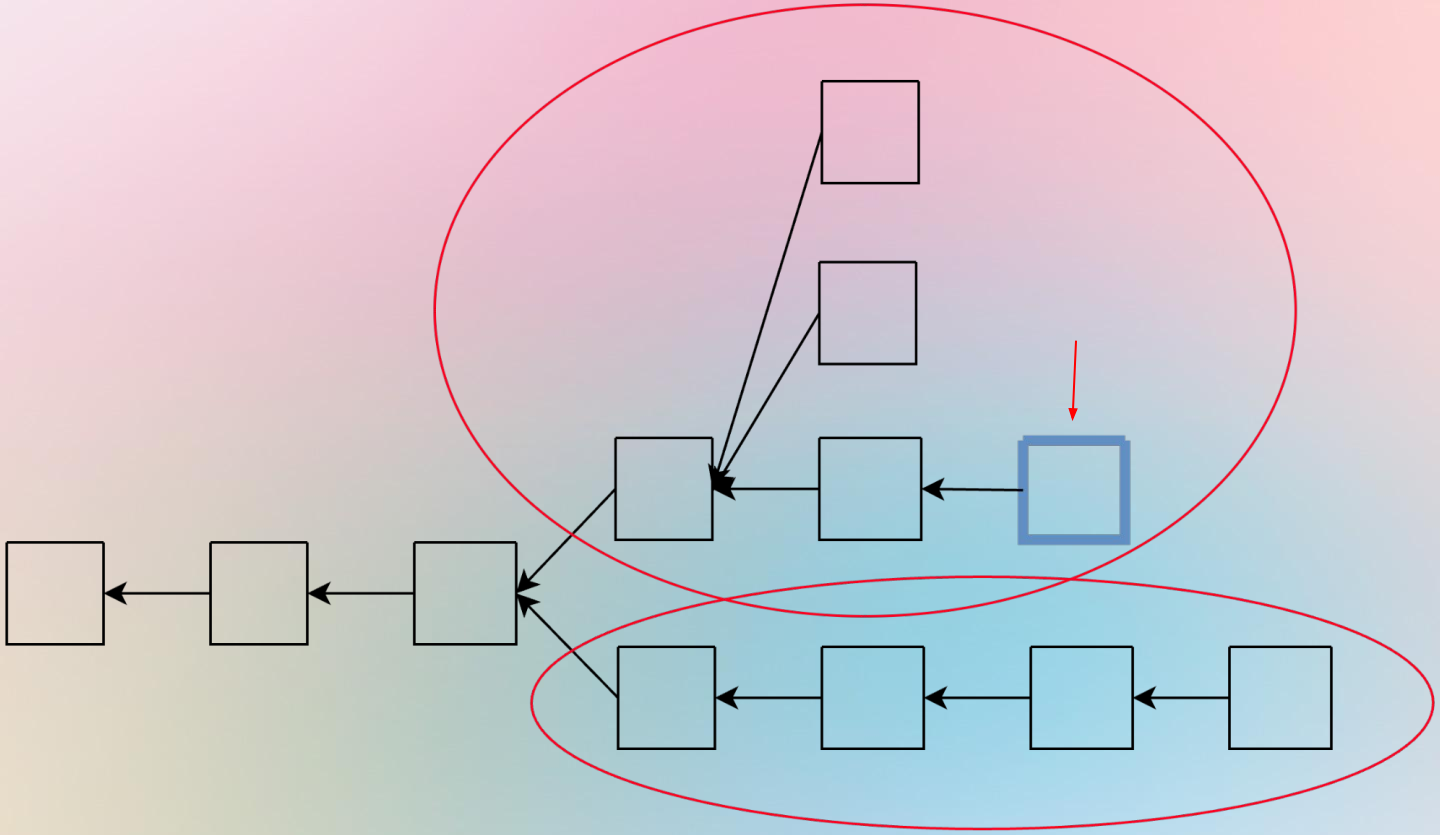
GHOST = Greediest Heaviest Observed SubTree



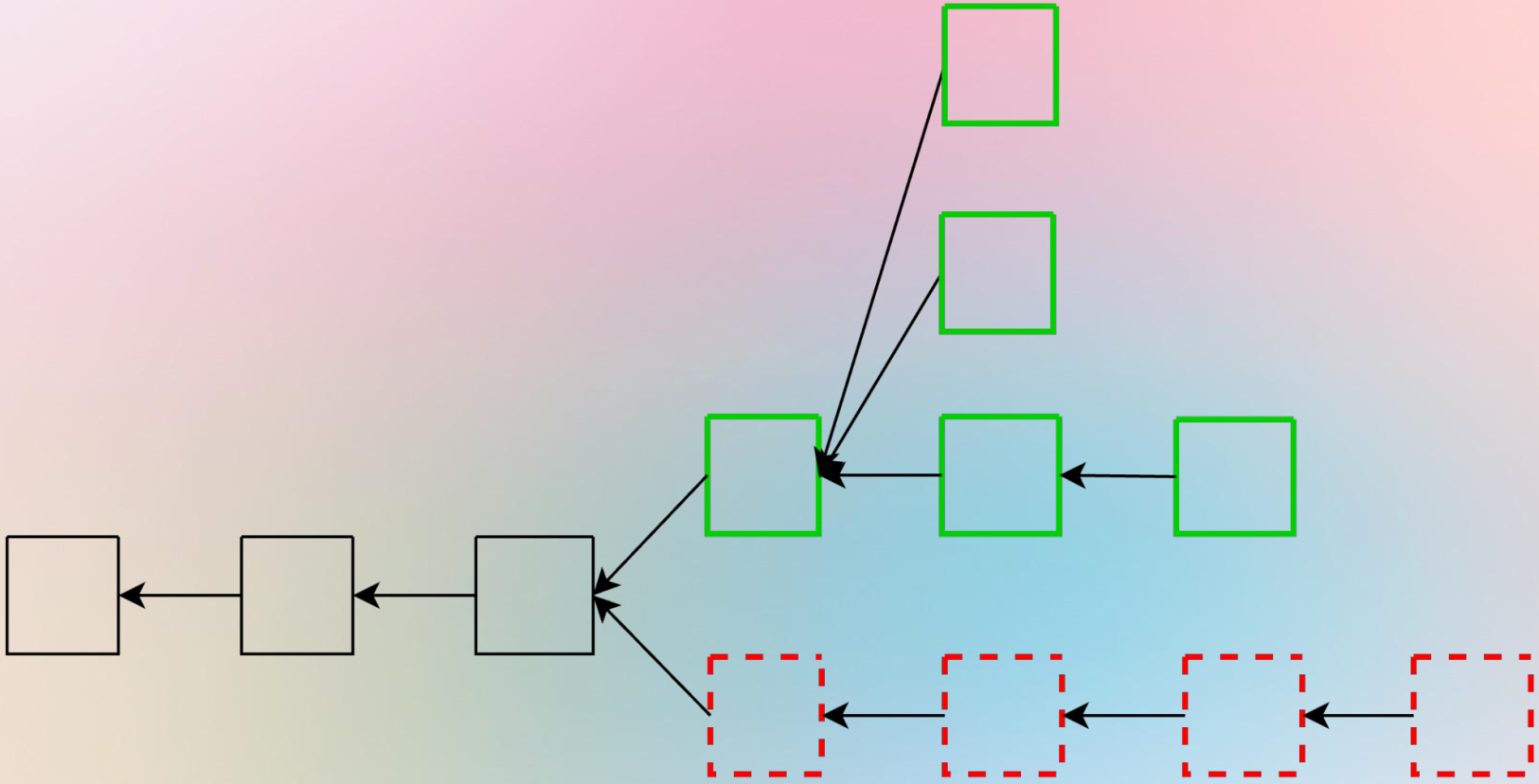
GHOST = Greediest Heaviest Observed SubTree



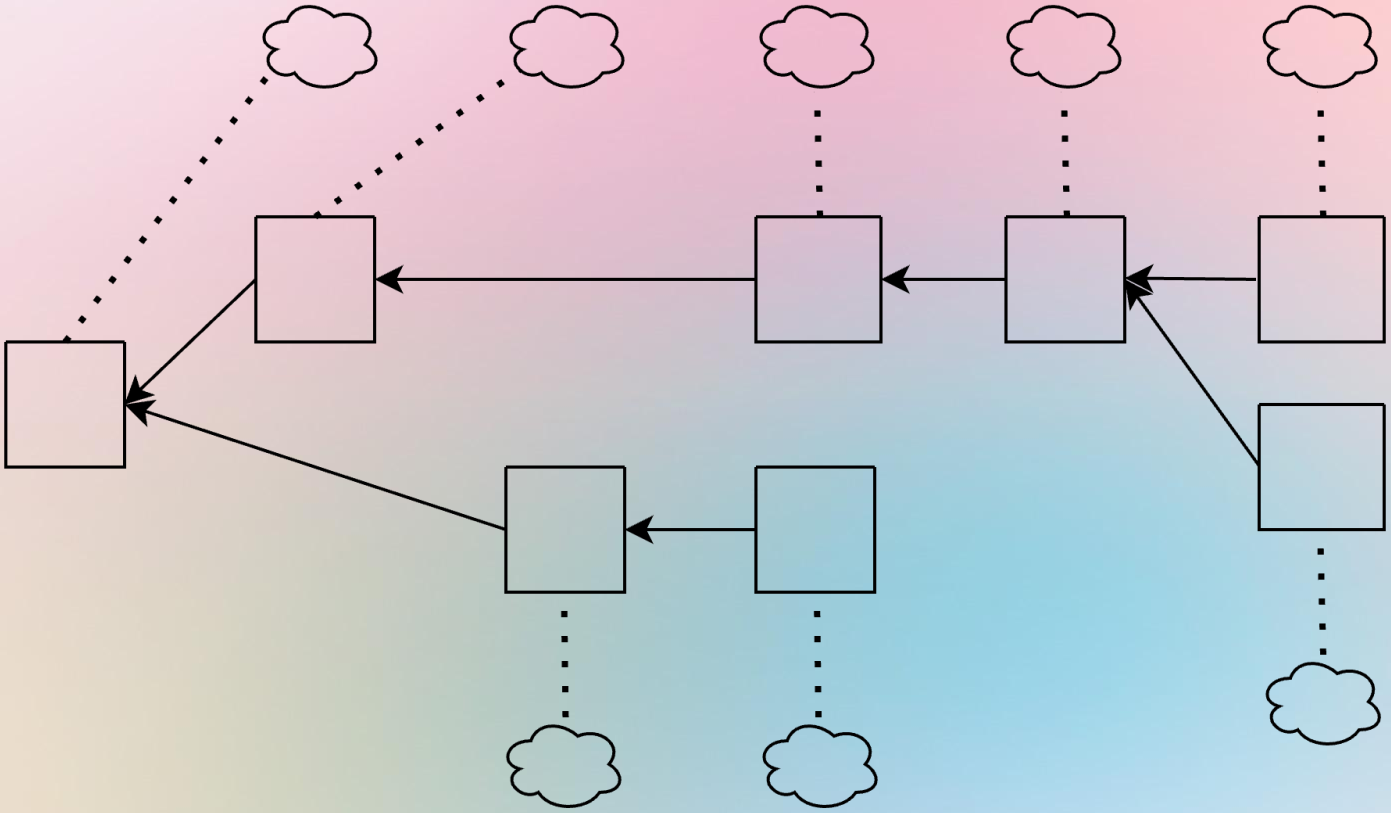
GHOST = Greediest Heaviest Observed SubTree



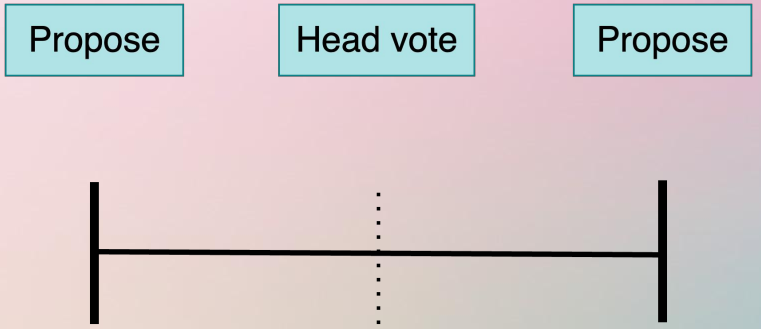
GHOST = Greediest Heaviest Observed SubTree



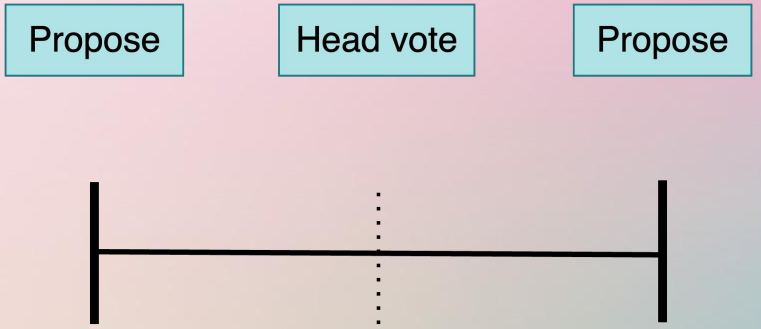
LMD-GHOST = Latest Message Driven GHOST

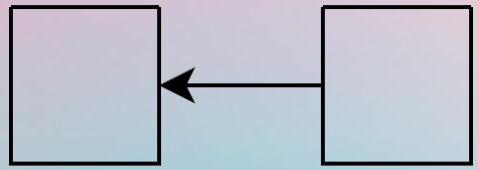


LMD-GHOST as a protocol

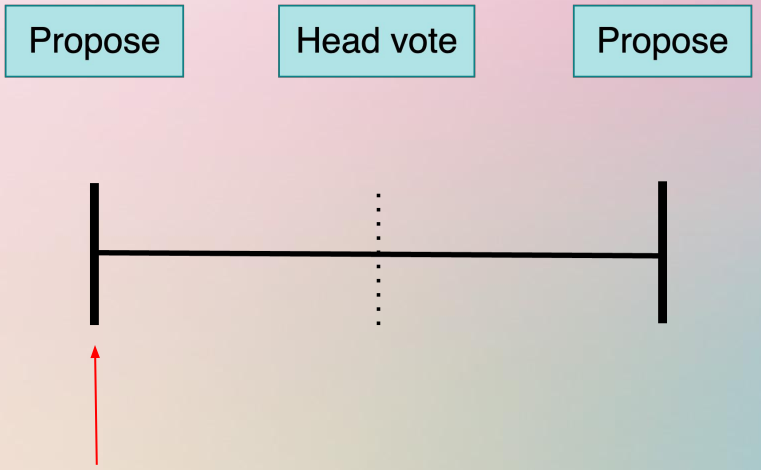


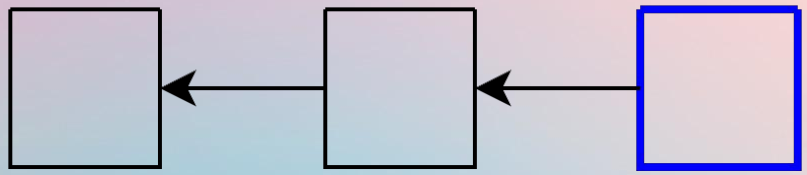
LMD-GHOST as a protocol



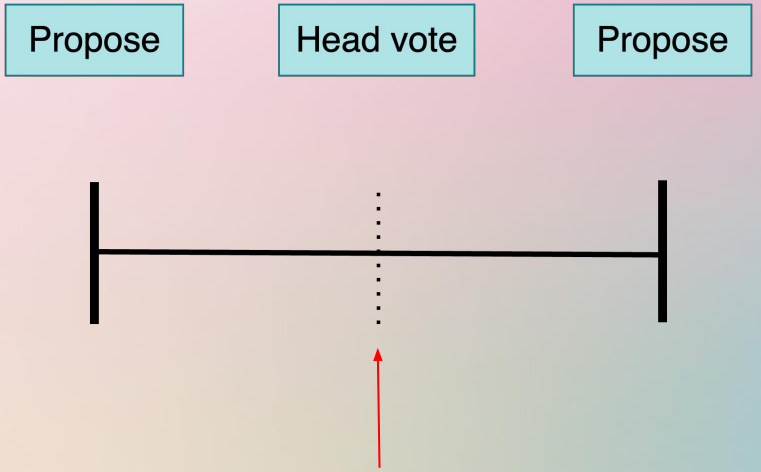


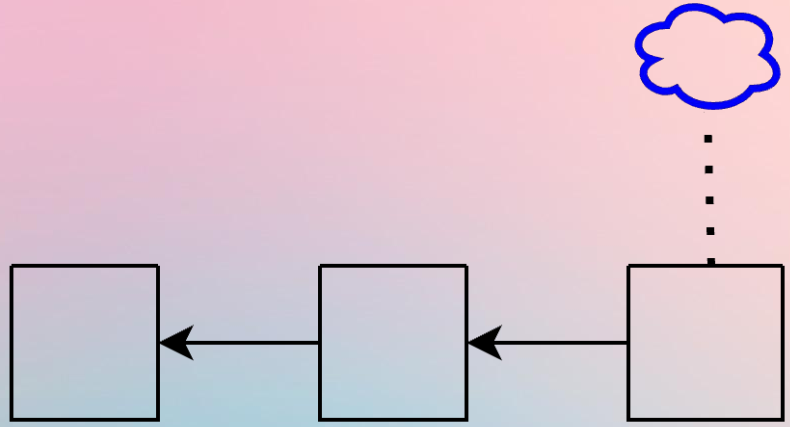
LMD-GHOST as a protocol



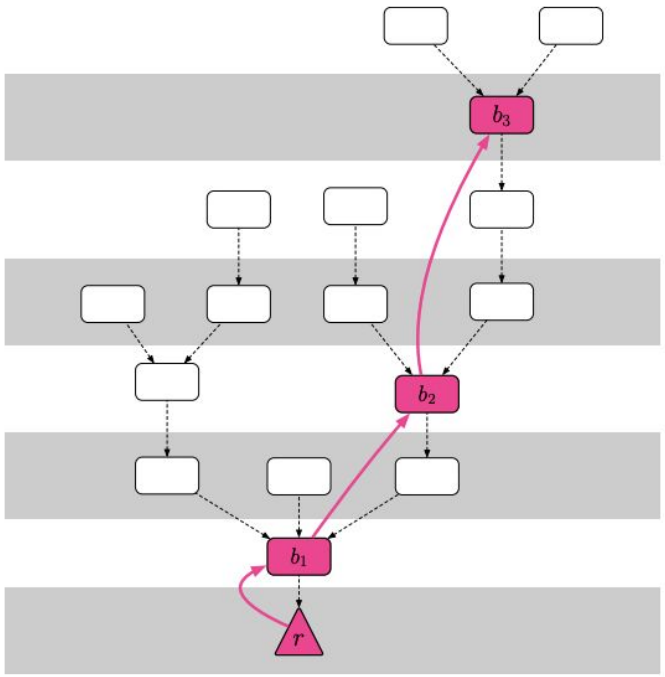


LMD-GHOST as a protocol





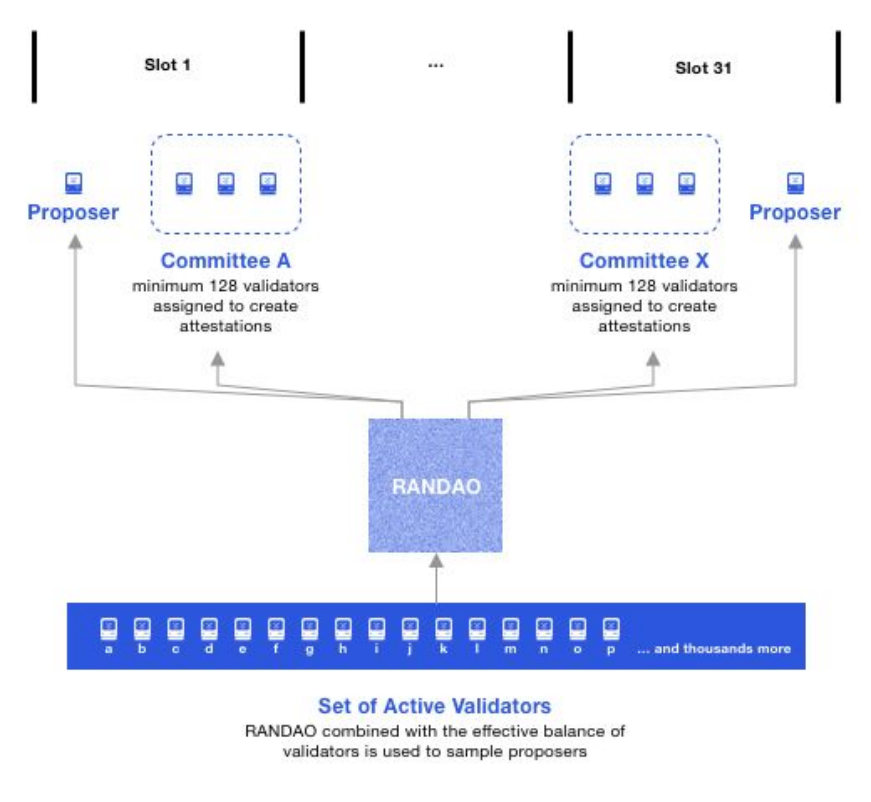
Casper FFG (Friendly Finality Gadget)



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

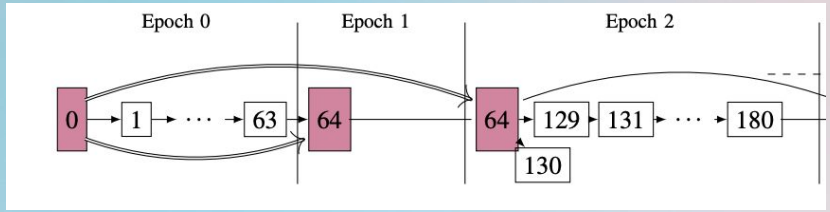
Casper the Friendly Finality Gadget
https://arxiv.org/pdf/1710.09437.pdf

Ethereum today



In each slot, a pseudorandomly elected committee votes for LMD-GHOST and for CASPER-FFG

Over 32 slots (an epoch) all validators have voted, concluding one round of voting of CASPER-FFG



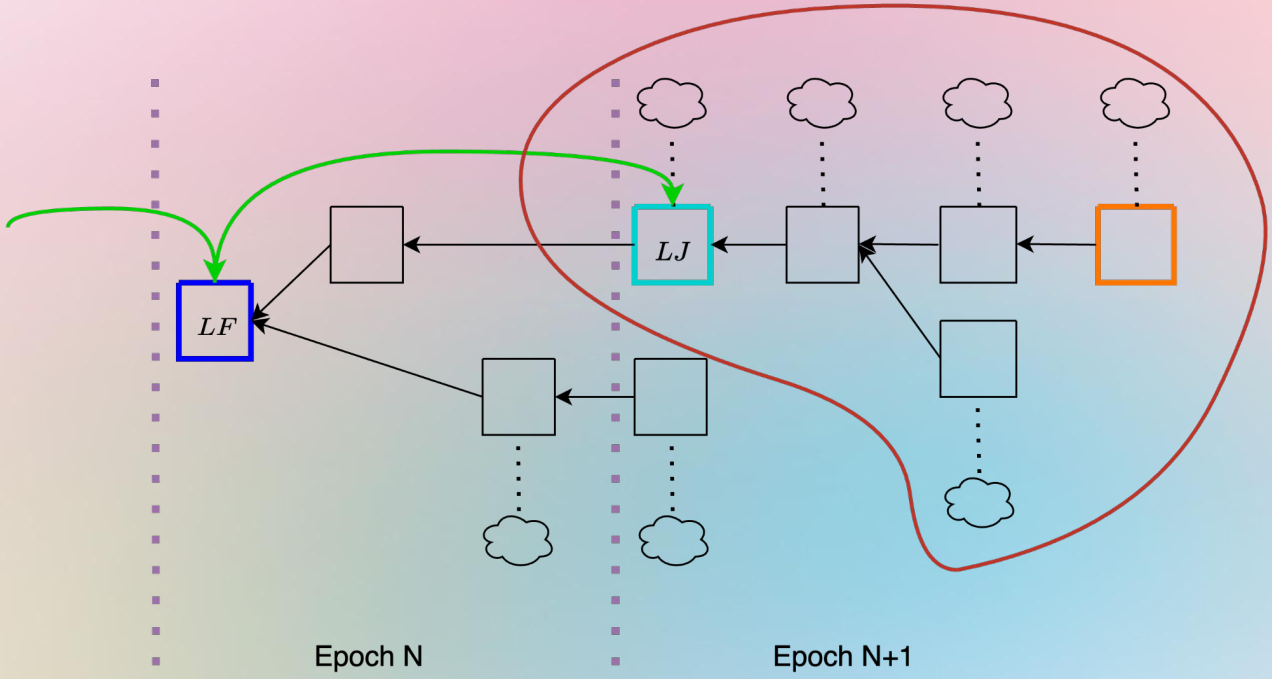
[The Beacon Chain explainer you need to read first, https://ethos.dev/beacon-chain]

[Combining GHOST and Casper, https://arxiv.org/pdf/2003.03052.pdf]

Modelled as an Ebb-and-flow protocol

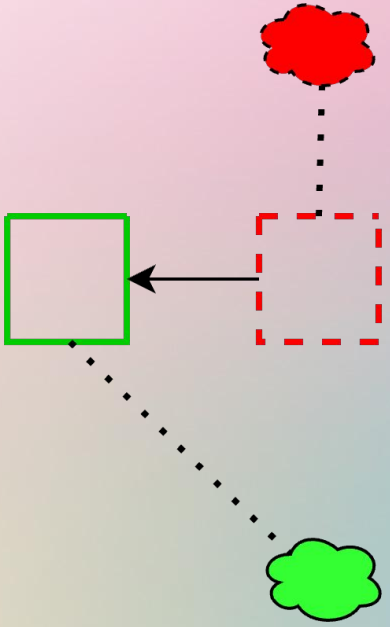
- Network model: partially synchronous network with GST, time where synchrony begins
- Participation model: dynamic until a time GAT, where it stabilizes
- Want:
  - **Dynamic availability**: available chain to be safe and live under network synchrony and dynamic participation (GST = 0, GAT = ∞)
  - **Finality**: Finalized chain to be always safe and live after max(GST, GAT)
  - **Prefix**: finalized chain is a prefix of the available chain

Hybrid fork-choice:
Start from the latest justified checkpoint, then run LMD-GHOST
=> Prefix



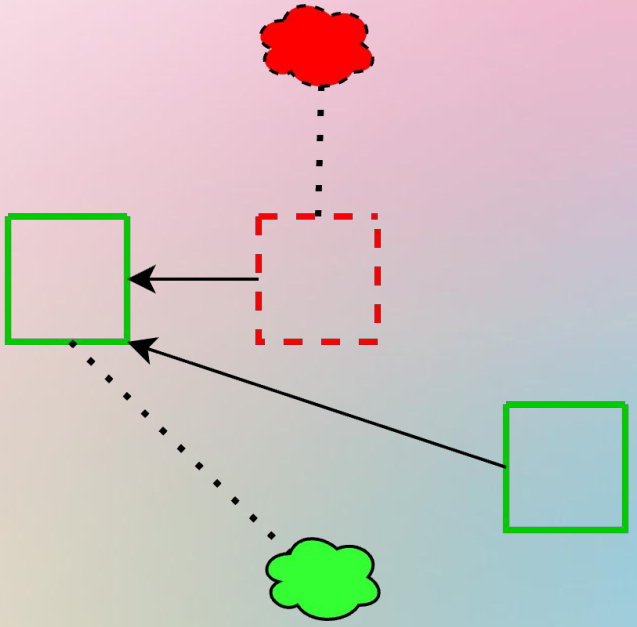
Problems of LMD-GHOST

Simple ex-ante reorg



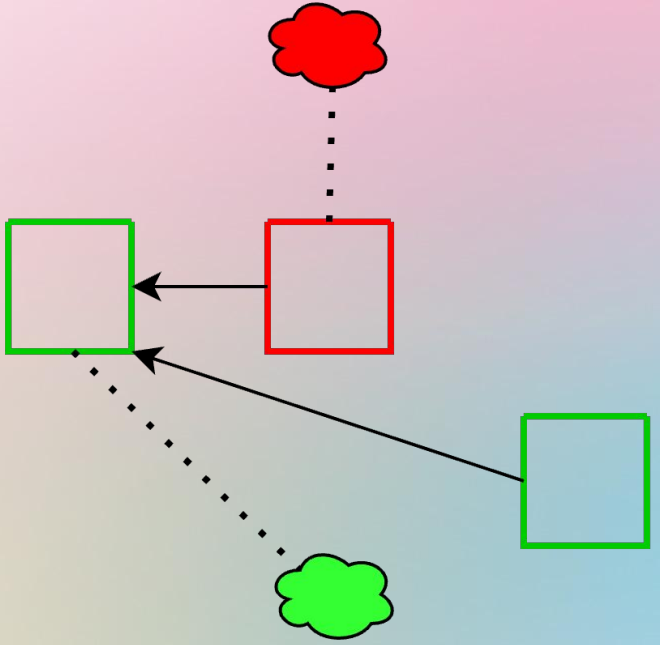
Adversary withholds a block and even just a single attestation to it

Simple ex-ante reorg



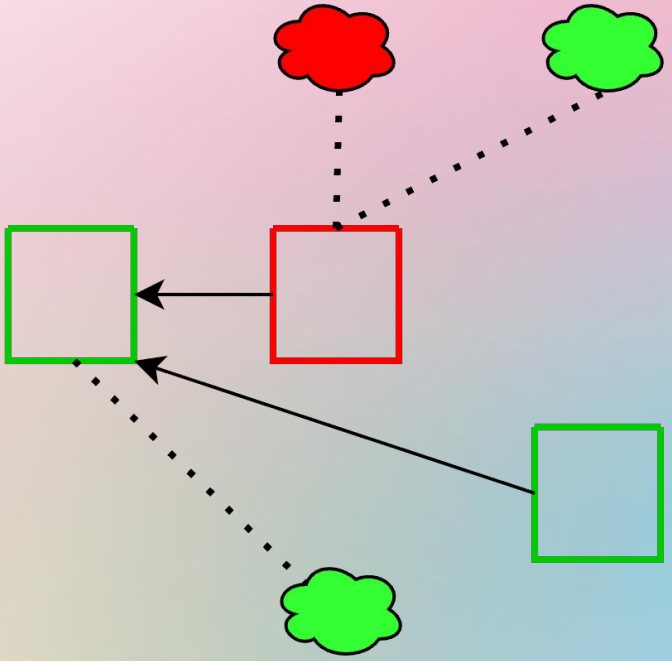
An honest proposer builds a block while unaware of the withheld one

Simple ex-ante reorg



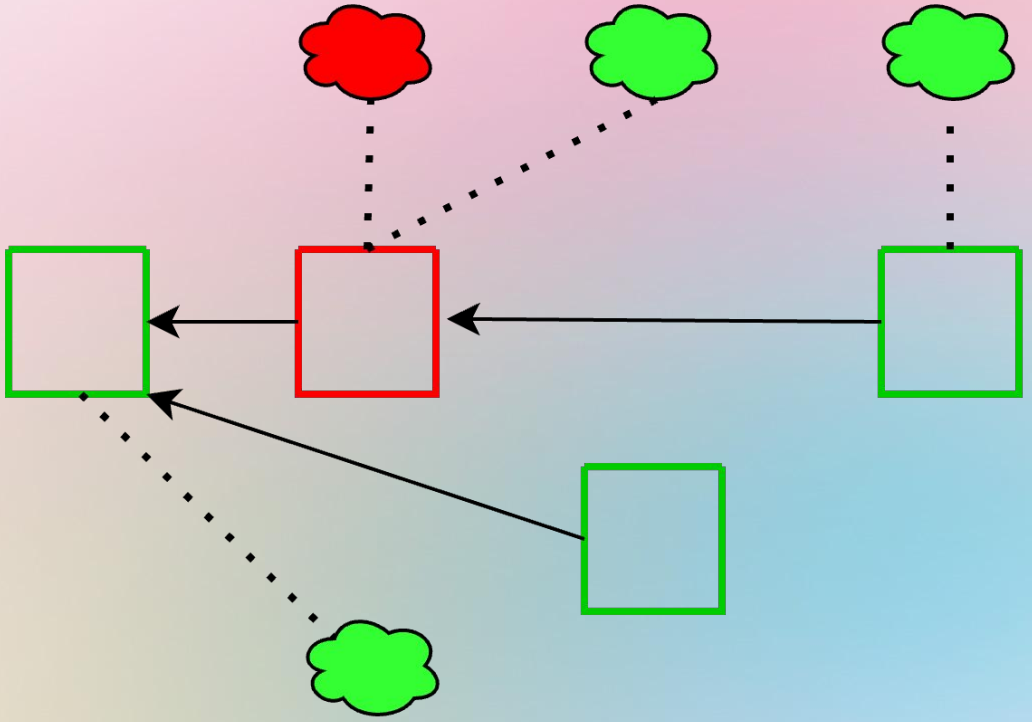
Adversary reveals its block and attestation

Simple ex-ante reorg



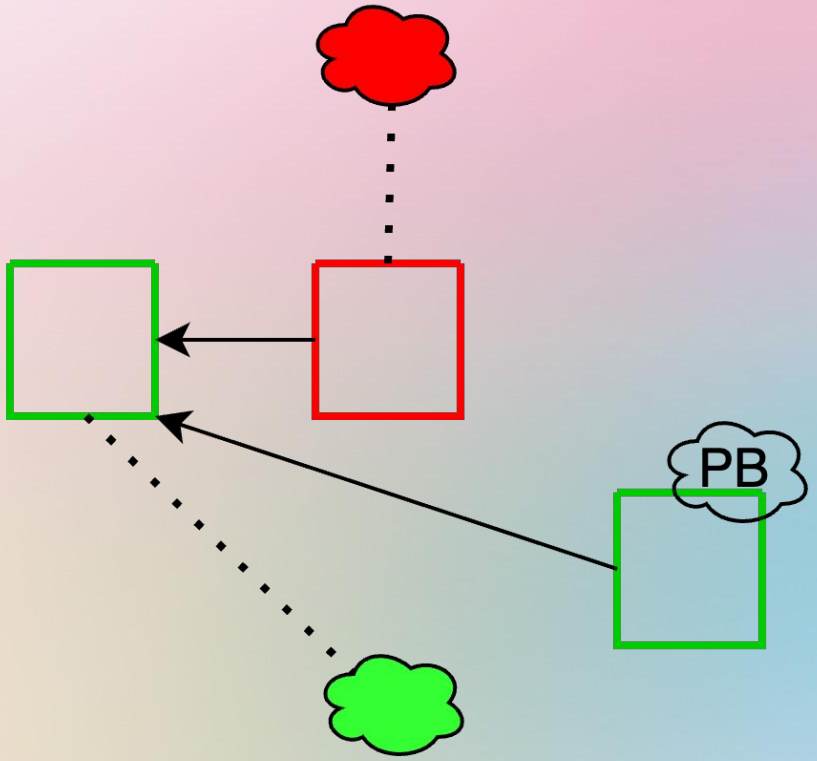
Attesters see the red block and attestation and attest to it

Simple ex-ante reorg



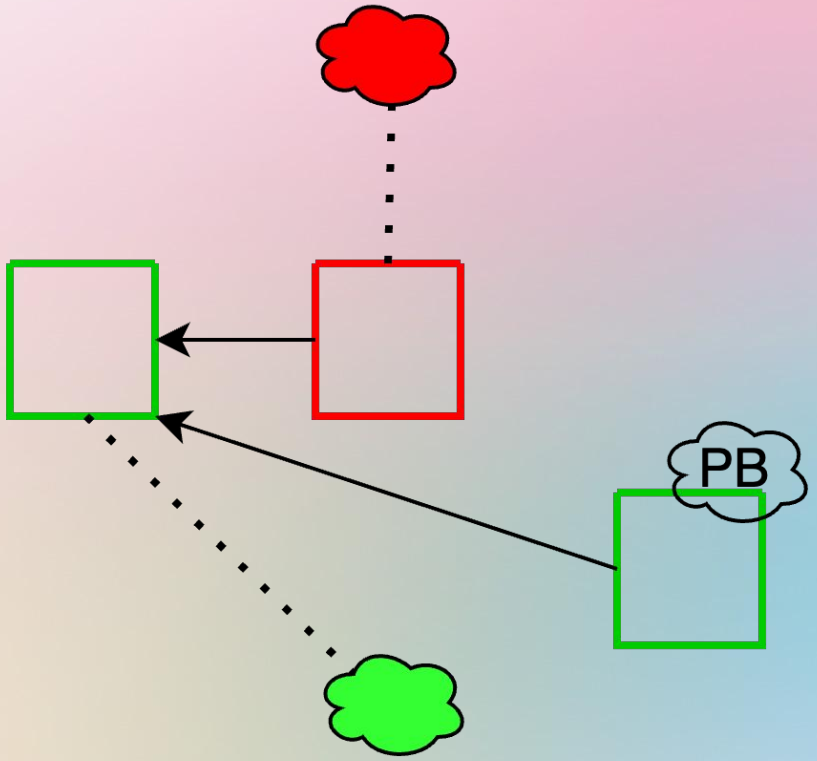
The honest proposal is forked out

Proposer boost



New block proposals have a temporary “weight boost” *during their slot.* In practice, set to 40%

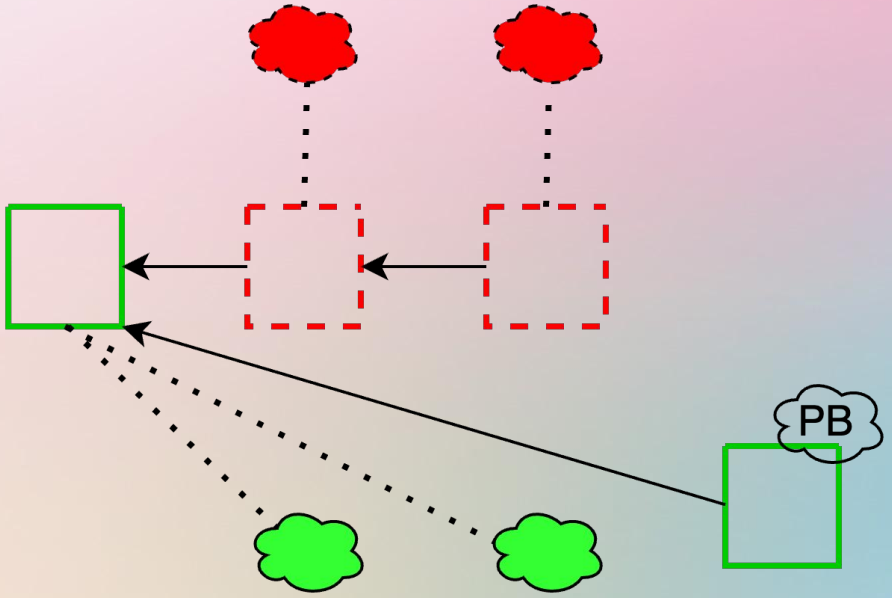
Proposer boost



New block proposals have a temporary “weight boost” *during their slot.* In practice, set to 40%

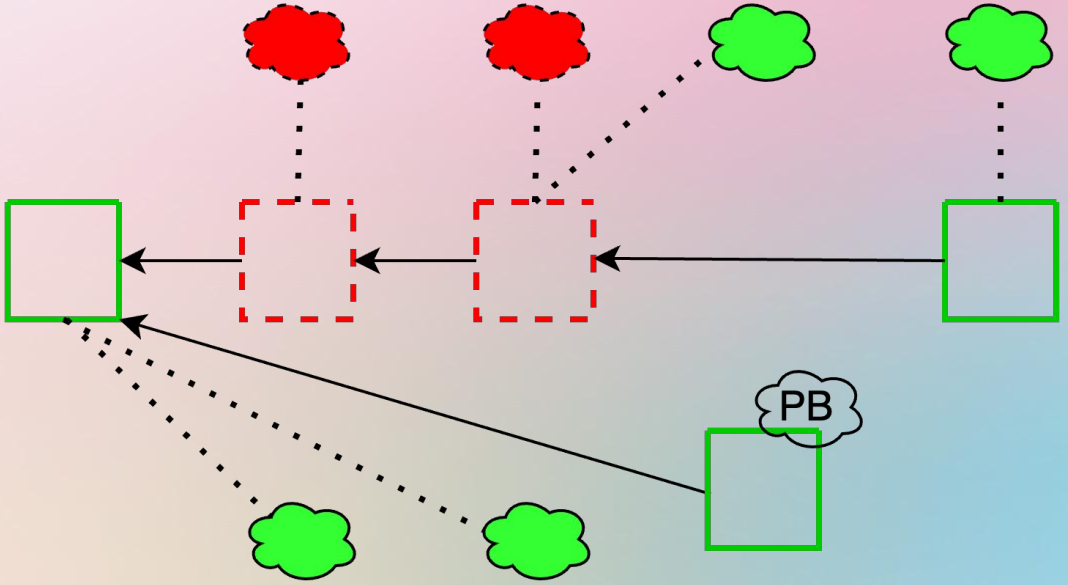
Can think of it as an hybrid of GHOST (weight = blocks) and LMD-GHOST (weight = attestations)

Ex-ante reorgs are still possible if controlling enough validators



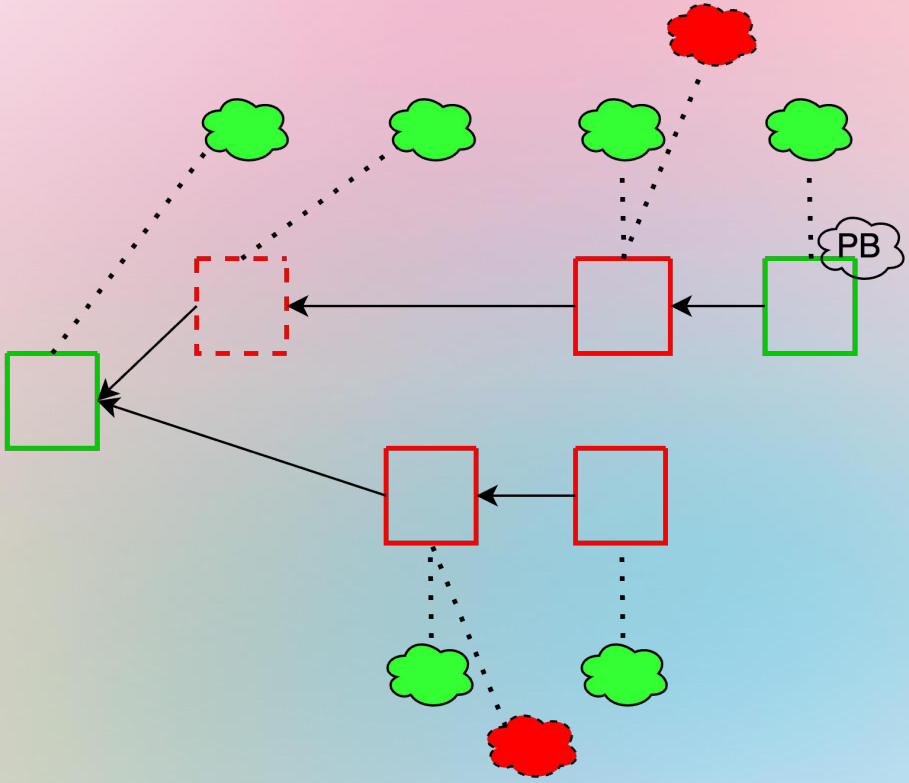
Adversary controls a few proposers in a row, withholds the block proposals and votes and reveals them to overcome the proposer boost.

Ex-ante reorgs are still possible if controlling enough validators



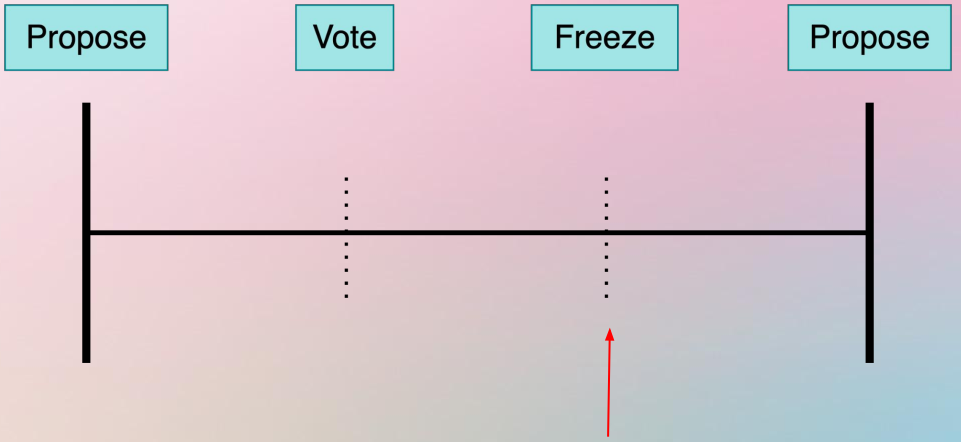
Adversary controls a few proposers in a row, withholds the block proposals and votes and reveals them to overcome the proposer boost. Honest attesters move to the adversarial branch.

Balancing attacks



Designing a theoretically secure available chain

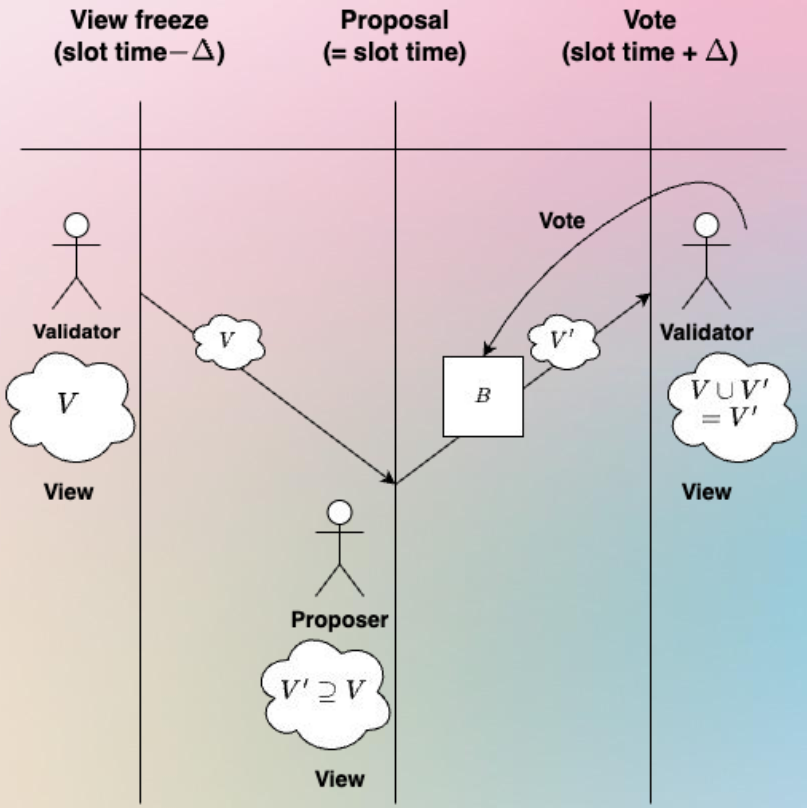
Improving on proposer boost: view-merge



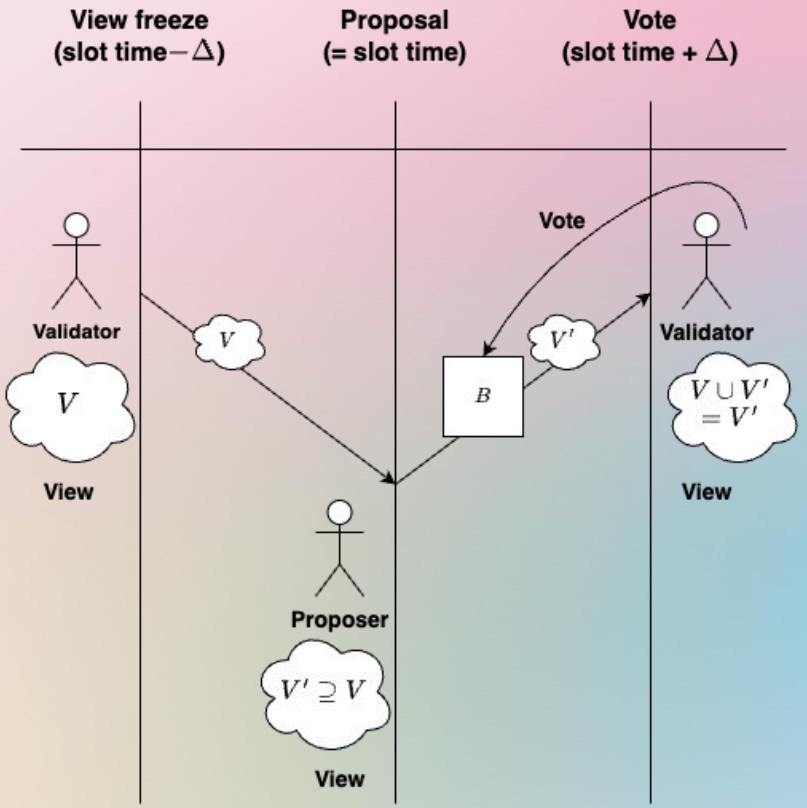
Goal: whenever an honest proposer shows up, their proposal is voted by all honest attesters

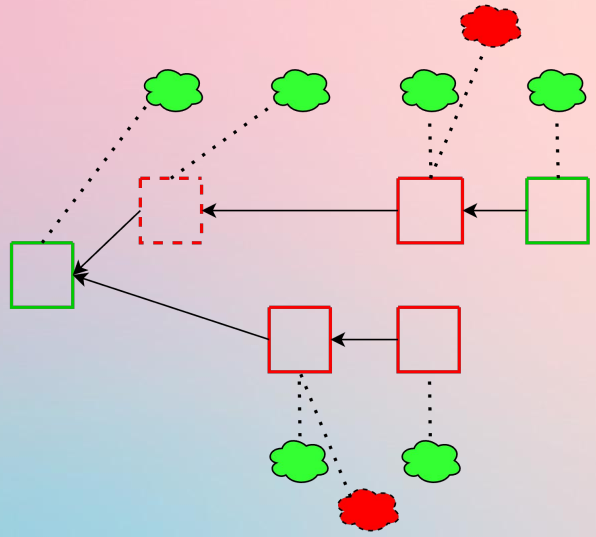
Introduce a new phase, where validators “freeze their view”: they buffer new attestations until after the next round of voting *unless the proposer says otherwise*

View-merge synchronization mechanism

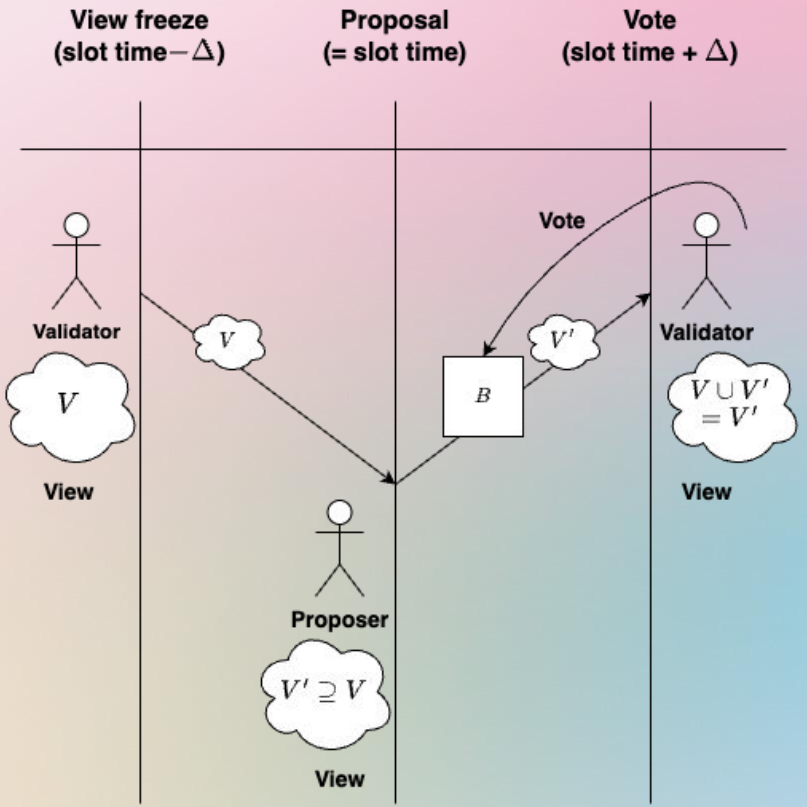


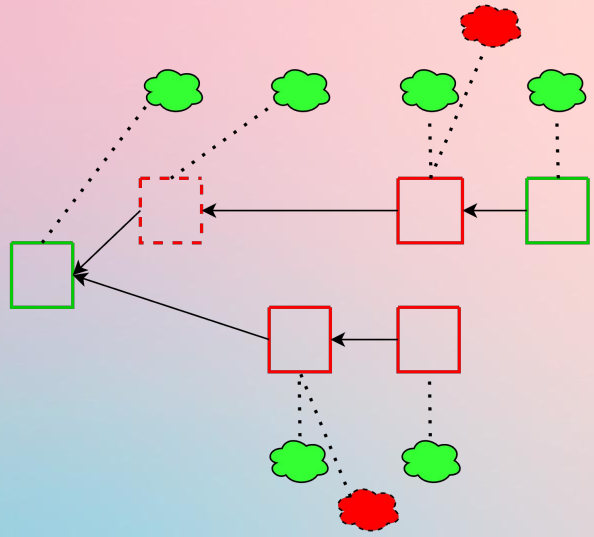
View-merge synchronization mechanism





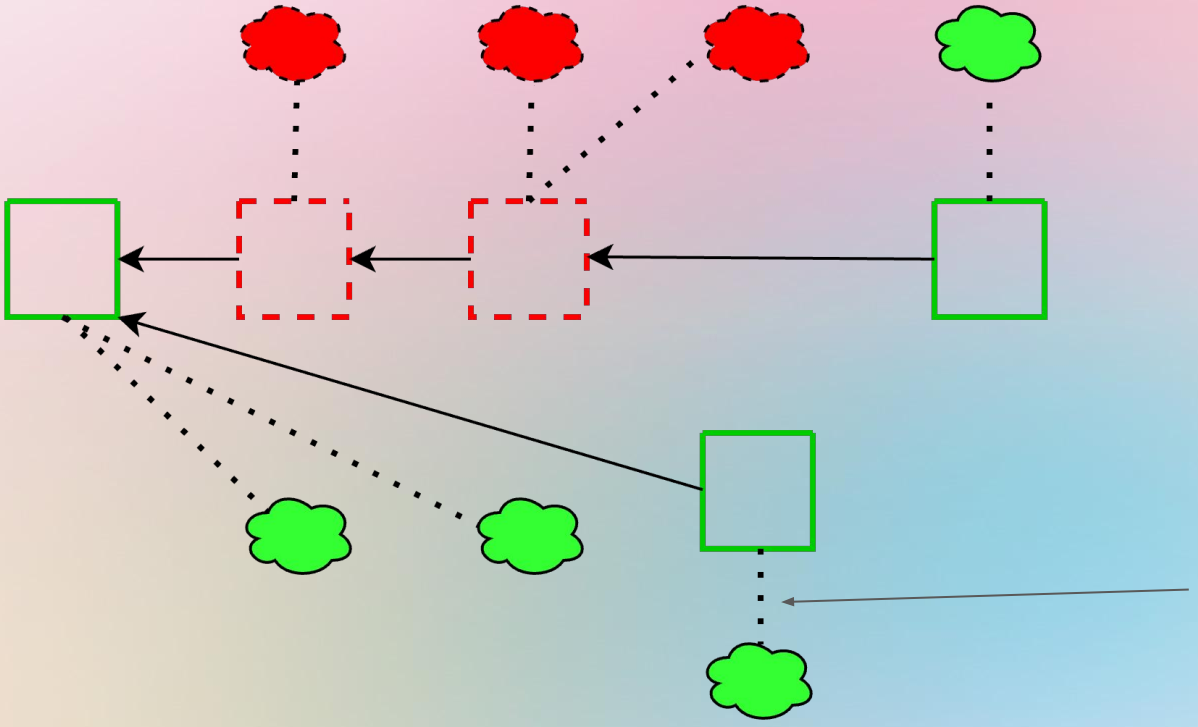
View-merge synchronization mechanism





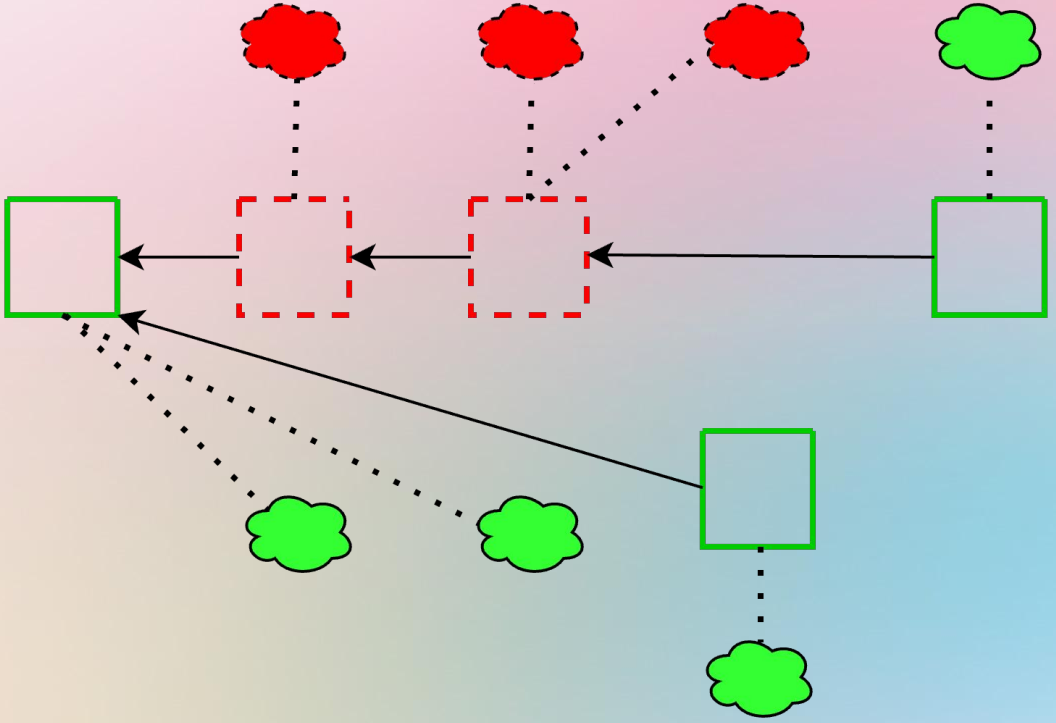
Regardless of how powerful the adversary is, honest attesters vote for honest proposals

… but ex-ante reorgs are still possible



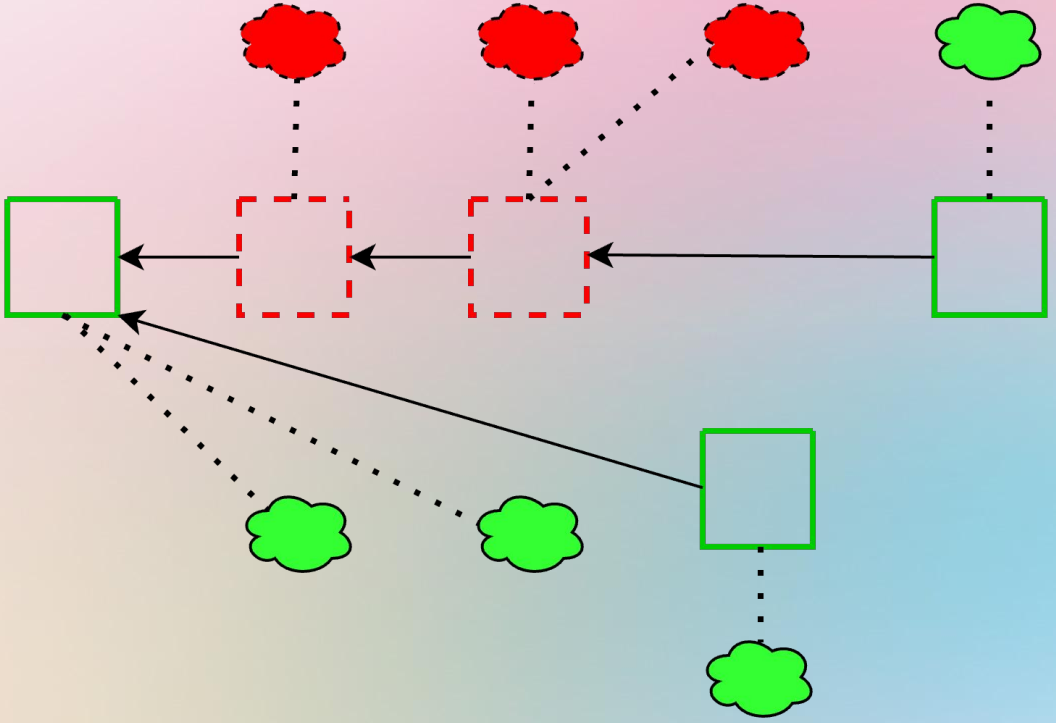
Honest attesters vote for the honest proposal, but it's not enough to overcome the adversarial votes

… but ex-ante reorgs are still possible



The root cause is that *committees* allow for weight accumulation over multiple slots, even if LMD counts only one vote per validator

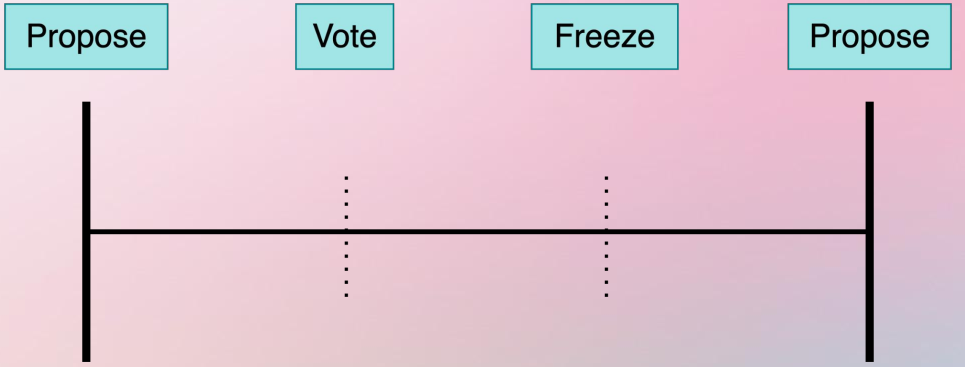
… but ex-ante reorgs are still possible



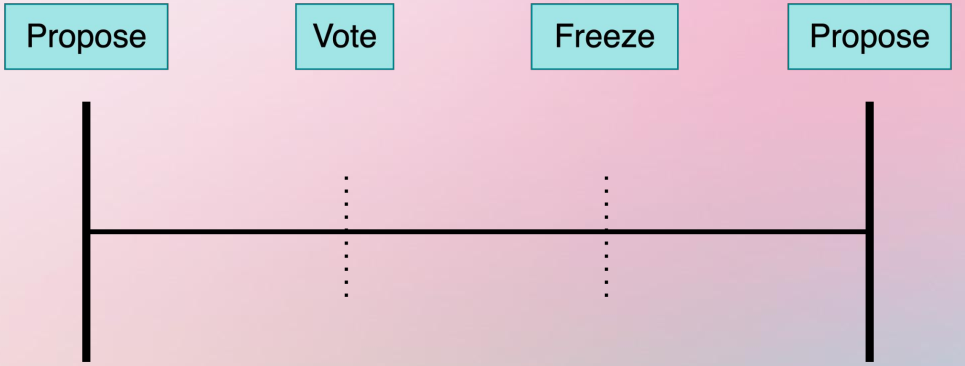
The root cause is that *committees* allow for weight accumulation over multiple slots, even if LMD counts only one vote per validator

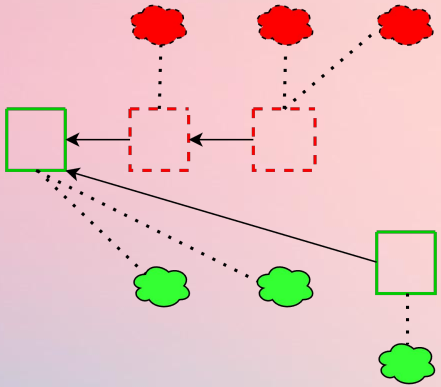
"Simple" solution: do away with committees. Everyone votes every slot

RLMD-GHOST



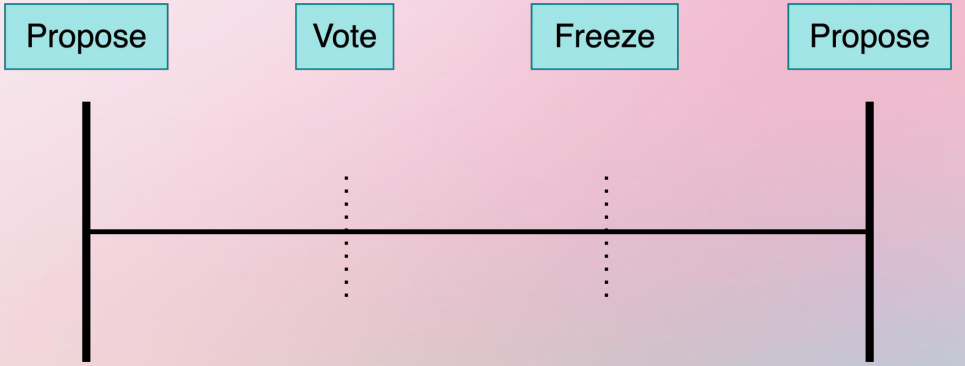
RLMD-GHOST

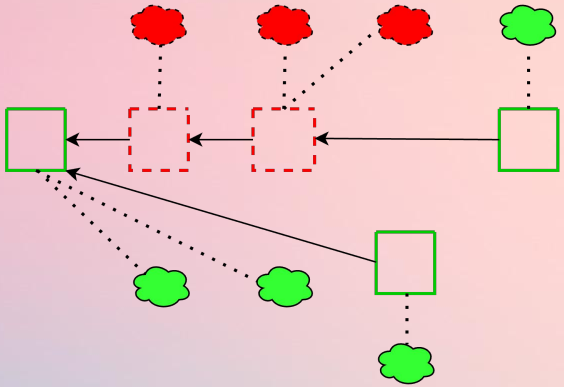




- View-merge => honest proposals are voted by all honest validators

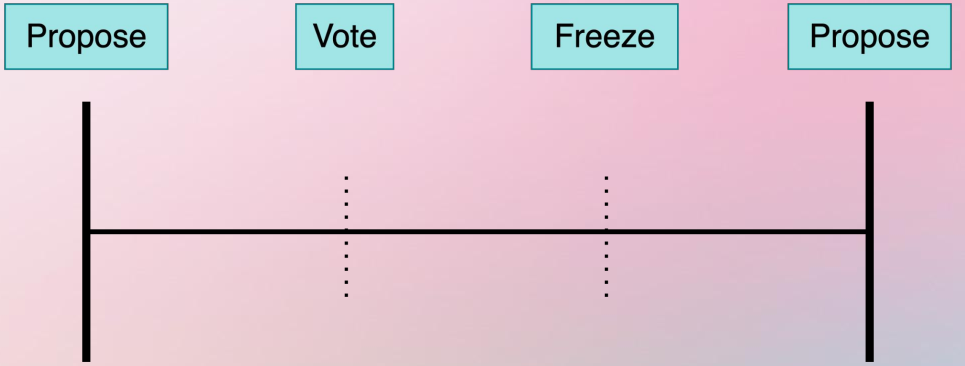
RLMD-GHOST

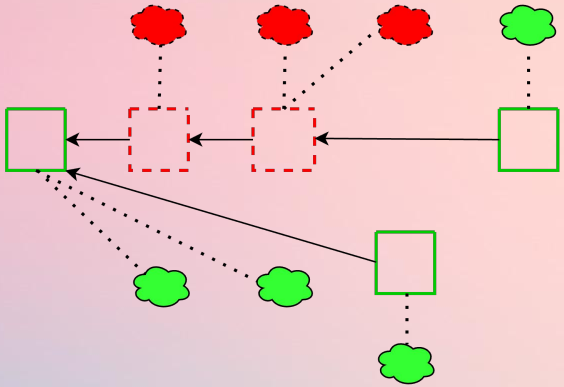




- View-merge => honest proposals are voted by all honest validators
- No committees => if all honest validators vote for something, it stays in the canonical chain forever

RLMD-GHOST

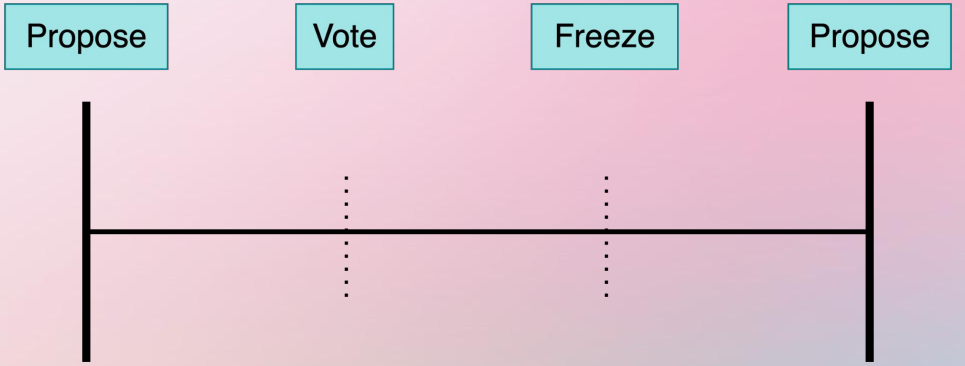


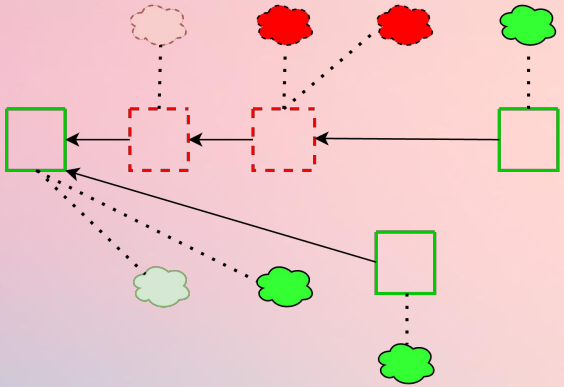


- View-merge => honest proposals are voted by all honest validators
- No committees => if all honest validators vote for something, it stays in the canonical chain forever

Together =>
Reorg resilience: honest proposals are never reorged

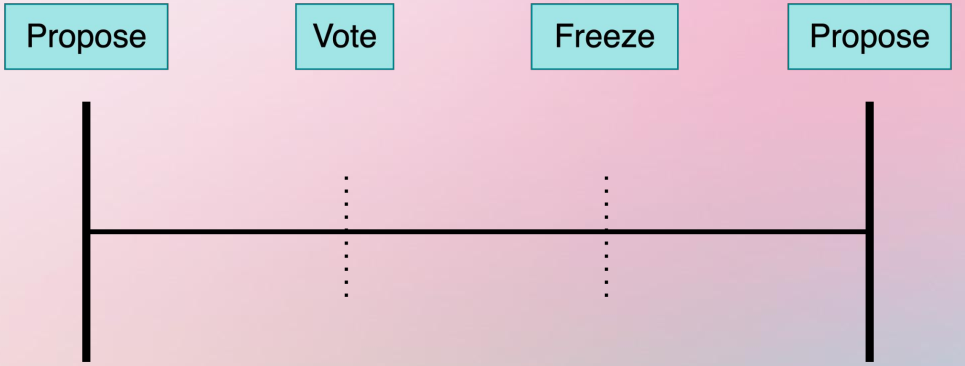
RLMD-GHOST

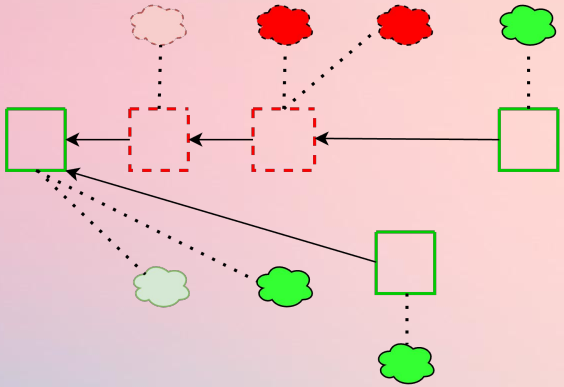




Vote expiry (**R** = recent): attestations don't contribute fork-choice weight after some number of slots

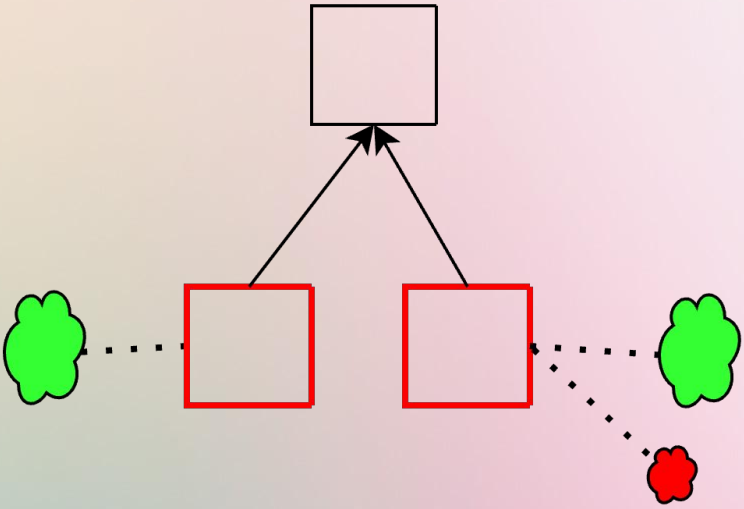
RLMD-GHOST

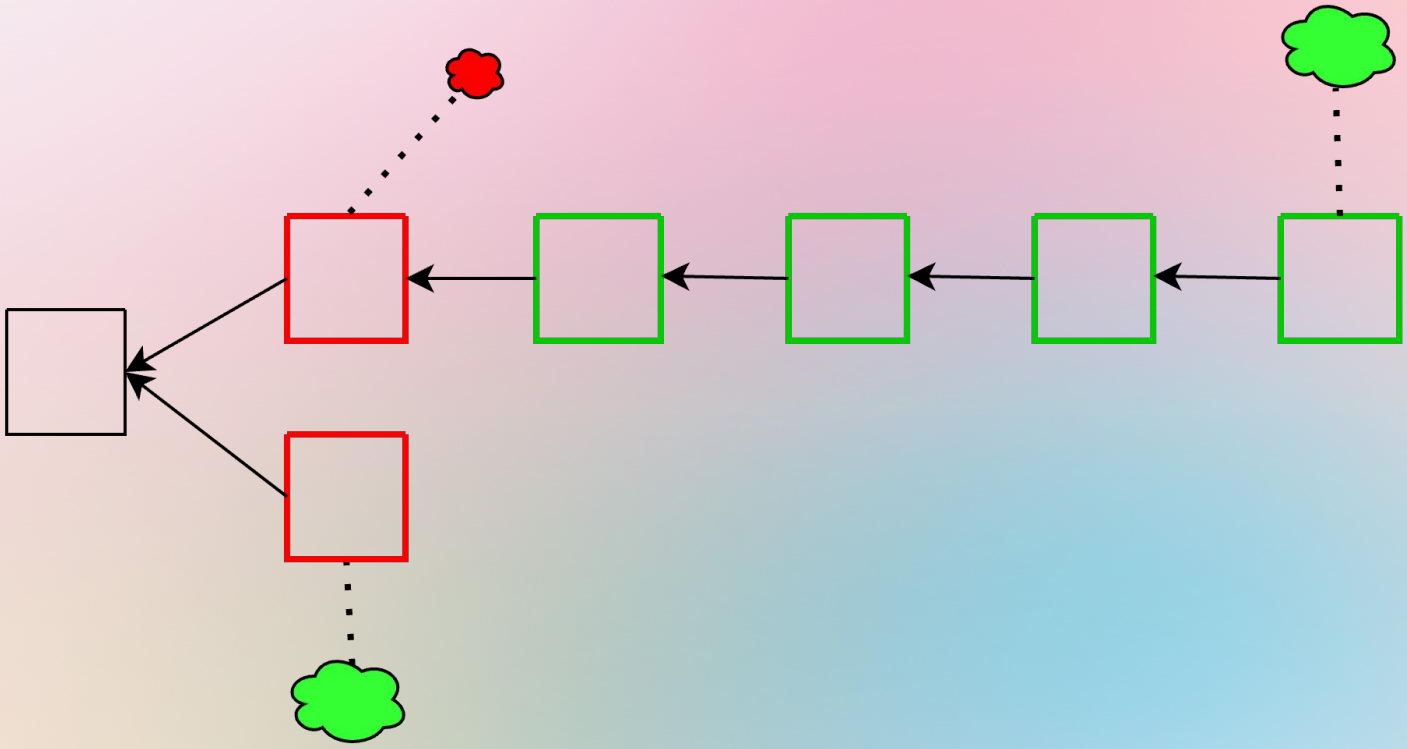


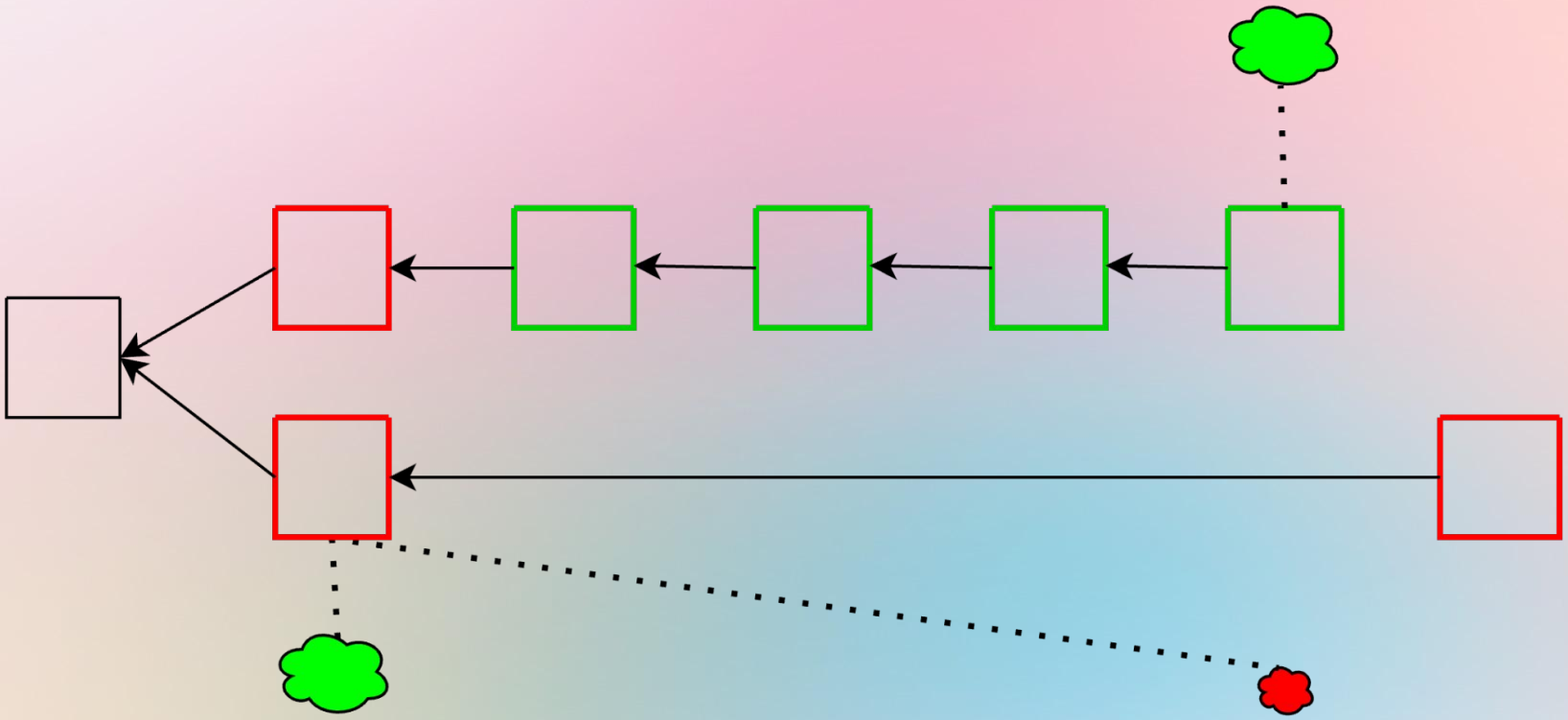


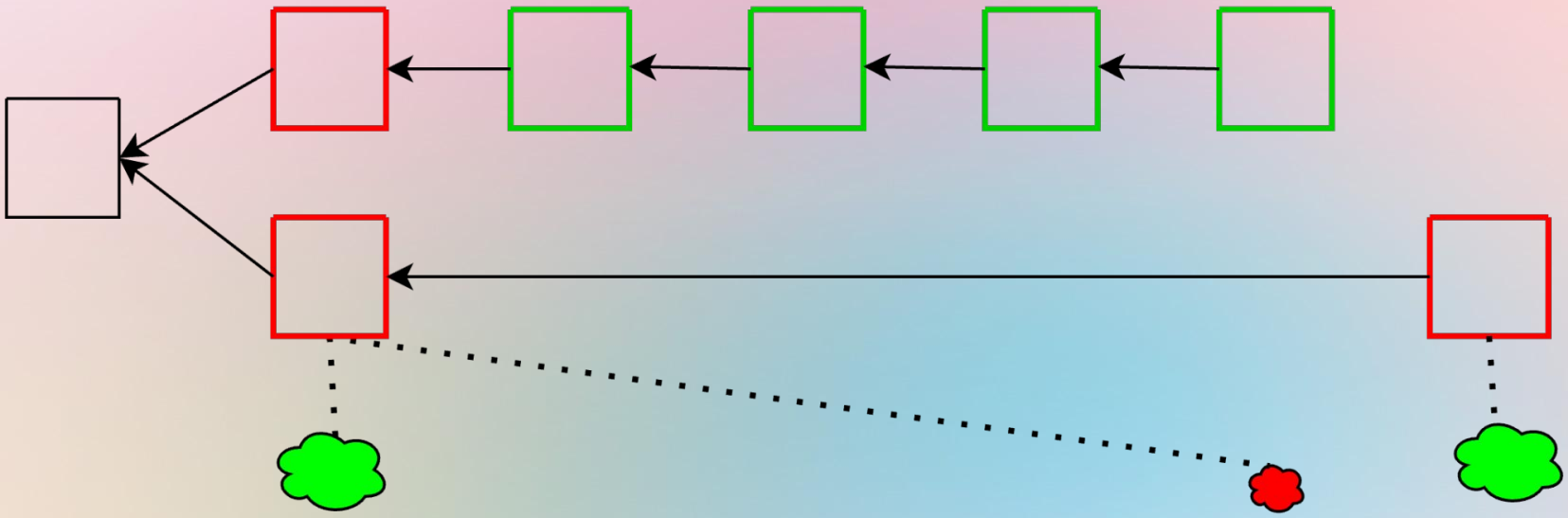
Vote expiry (**R** = recent): attestations don't contribute fork-choice weight after some number of slots

=> can always recover from a large portion of the validator set going offline

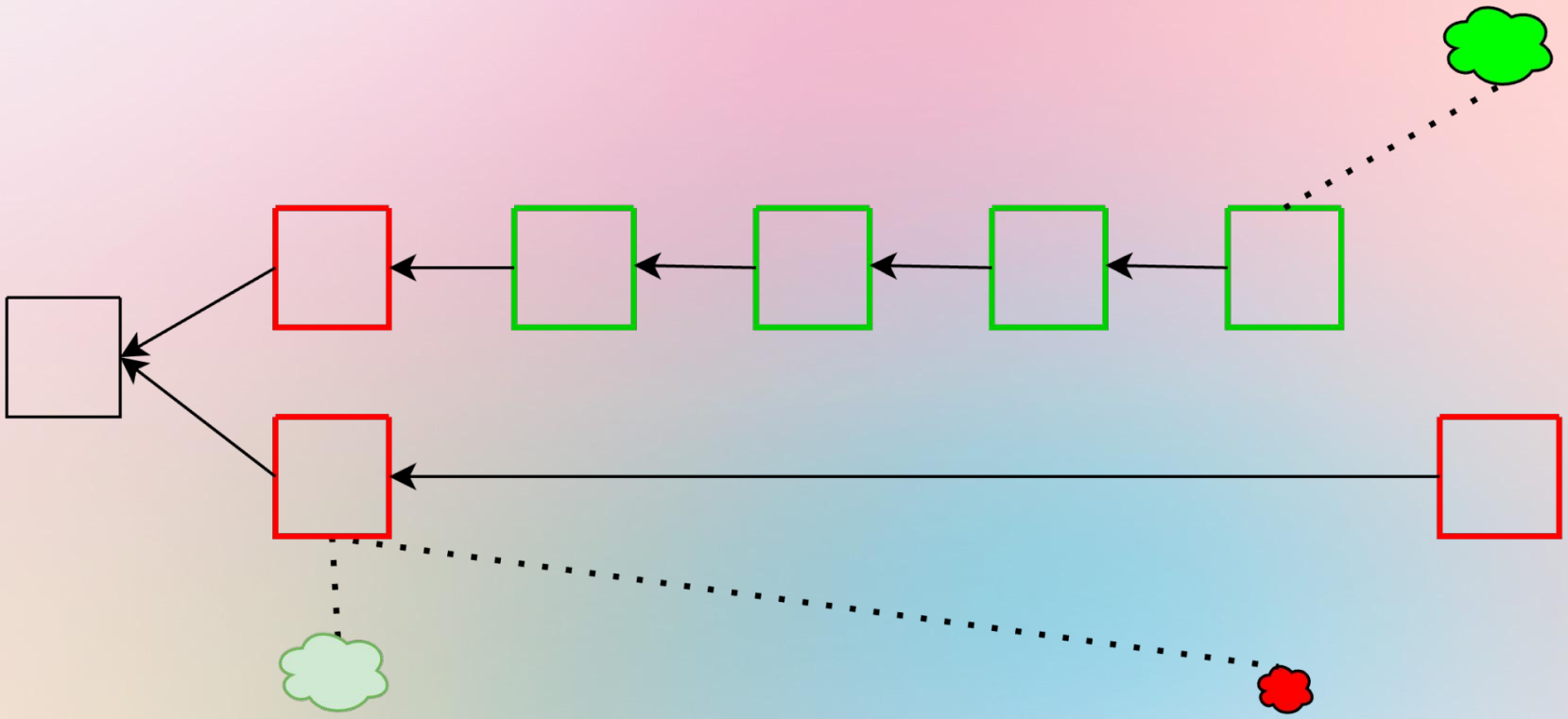




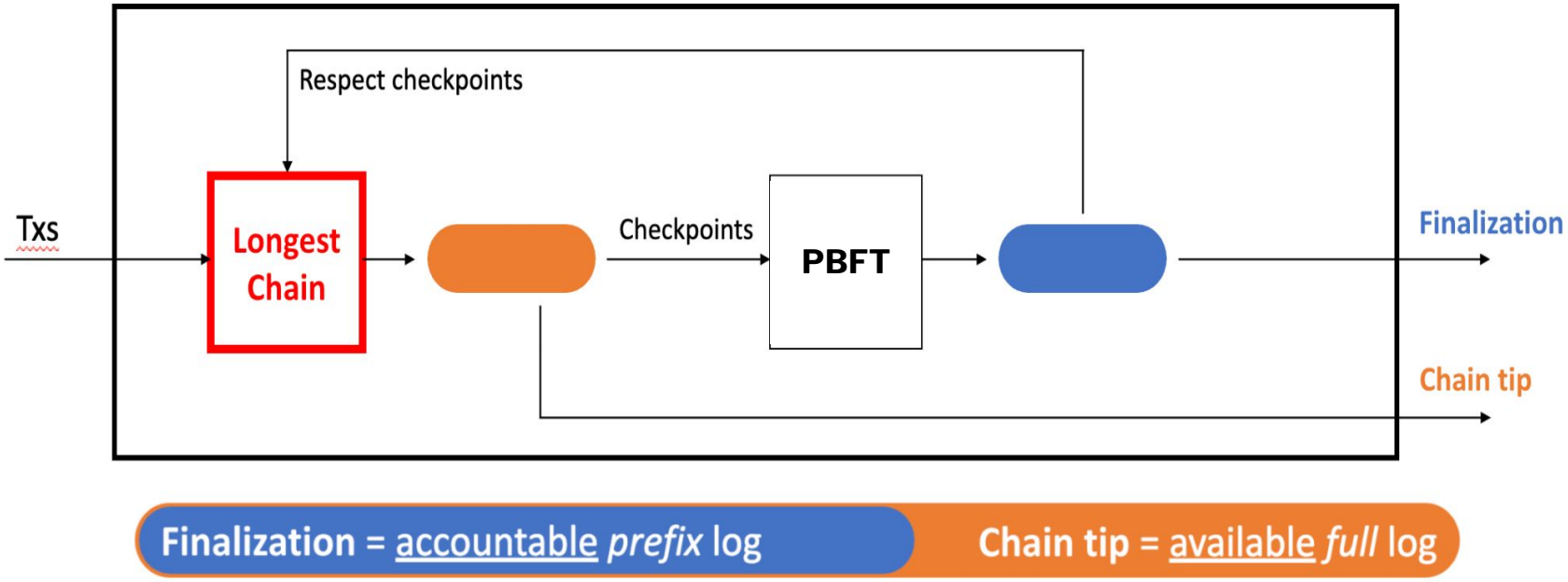


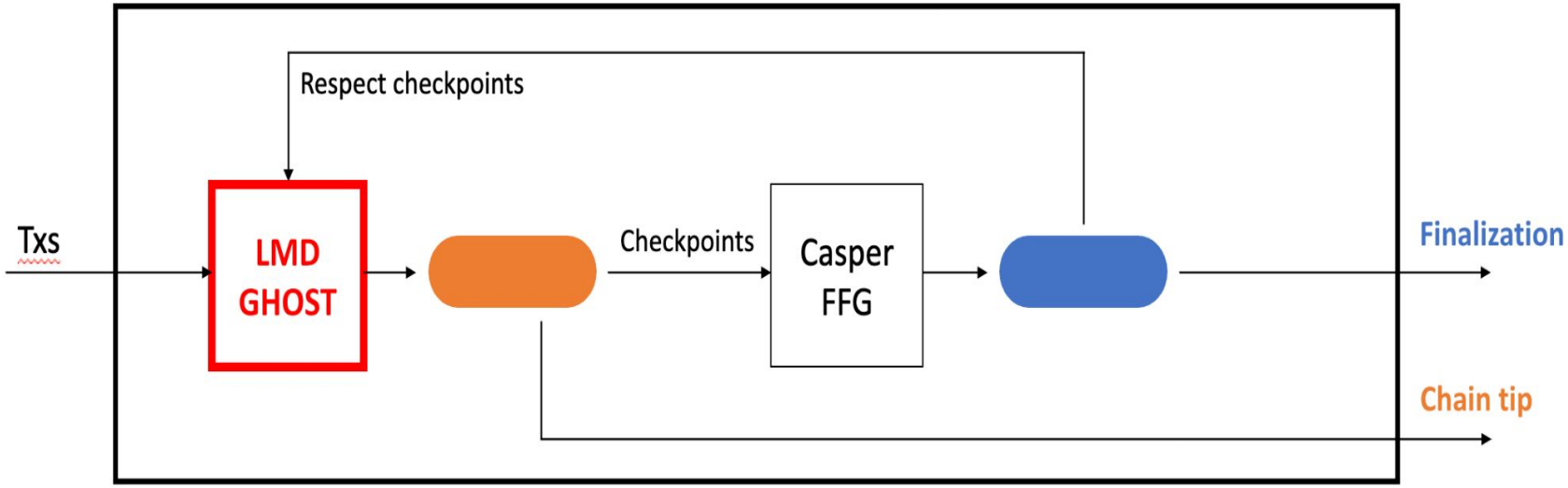


... while with expiry



SSF protocol


Respect checkpoints
Txs
Longest Chain
Checkpoints
PBFT
Finalization
Chain tip
Finalization = accountable prefix log
Chain tip = available full log



Finalization = accountable *prefix* log

Chain tip = available *full* log

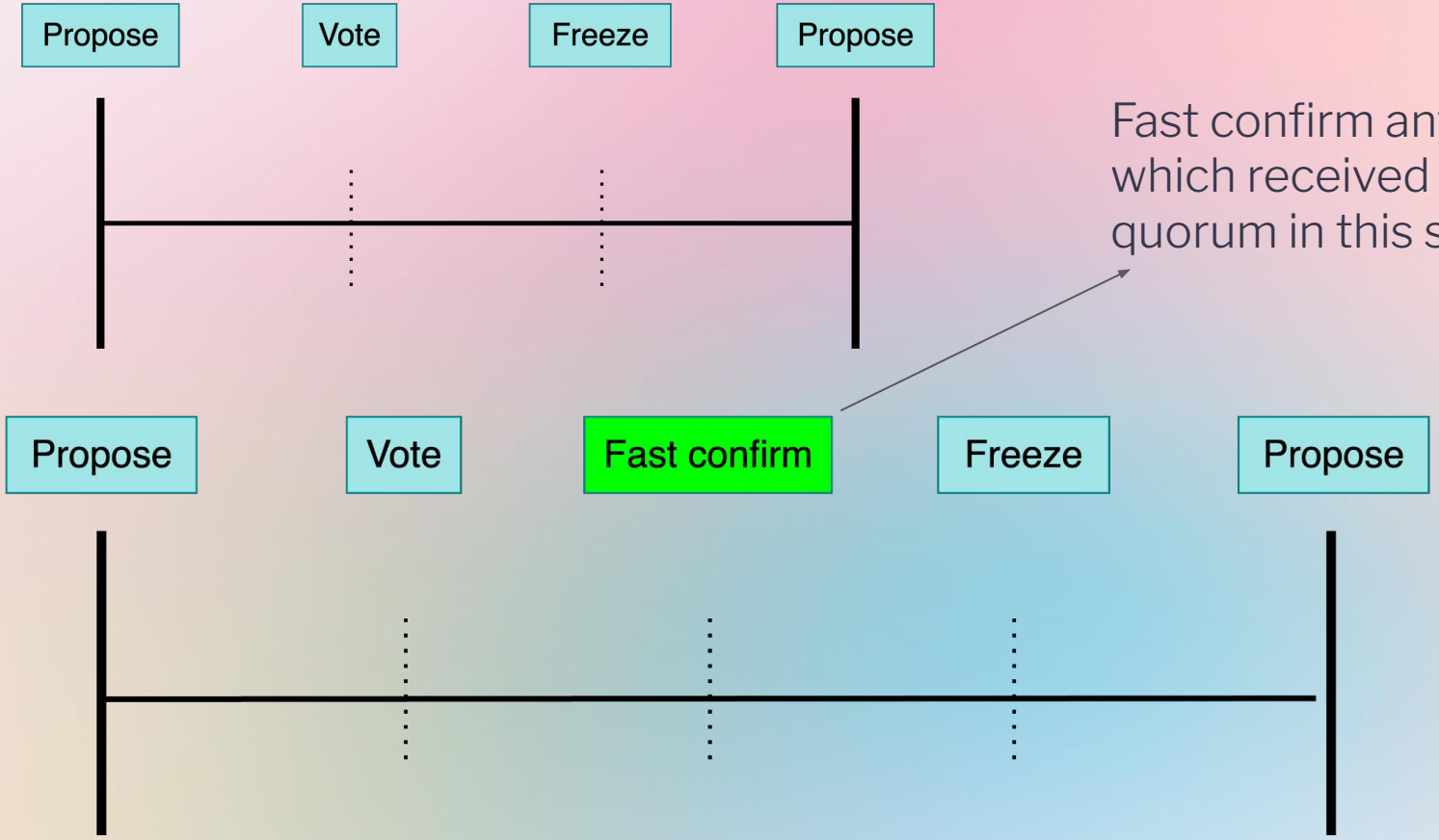
A key idea

Only attempt to finalize blocks which are **already confirmed by the underlying available protocol** (LMD-GHOST for us)

=> Finality gadget does not interfere with available protocol under *permanent* network synchrony

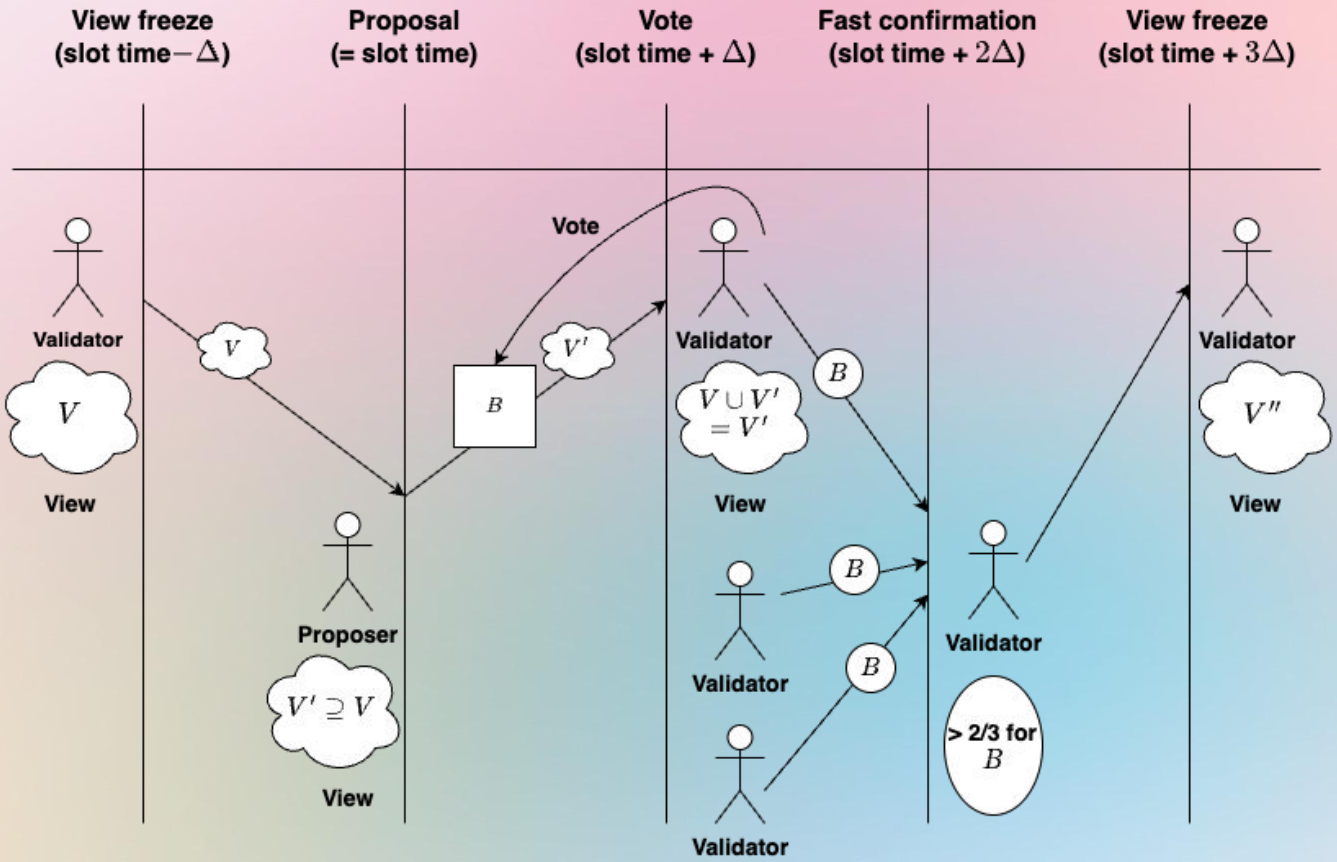
=> “all” we have to worry about is behavior when *recovering from periods of asynchrony*, when the gadget might interfere (tricky)

Add fast confirmation to RLMD-GHOST

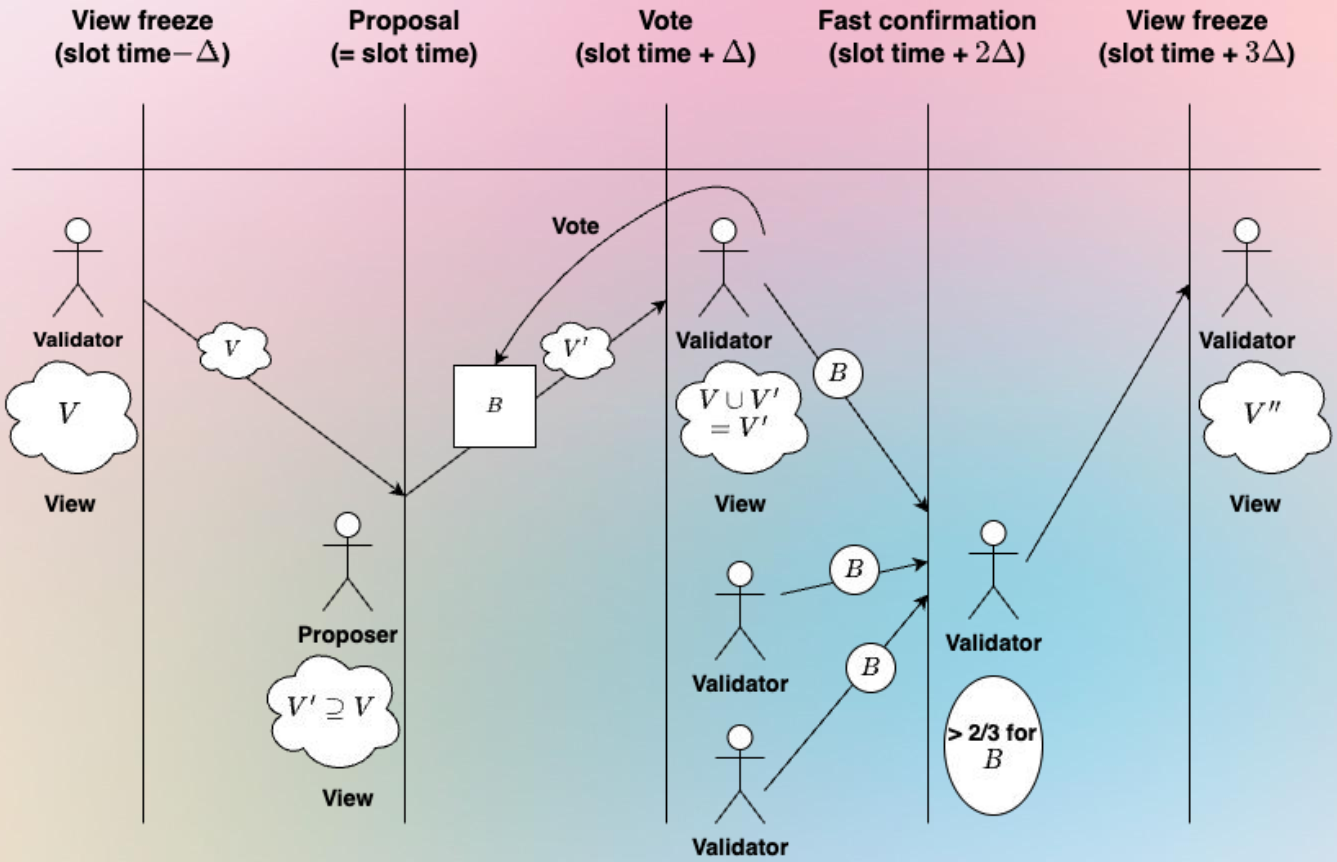


Fast confirm any block which received a ⅔ quorum in this slot

Add fast confirmation to RLMD-GHOST



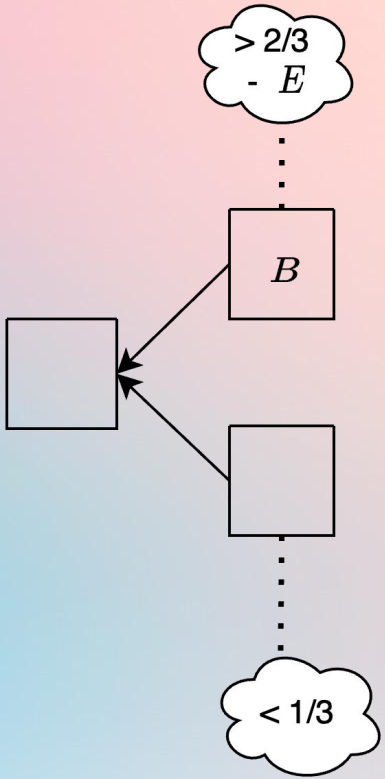
Add fast confirmation to RLMD-GHOST



Security guarantees under synchrony and 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

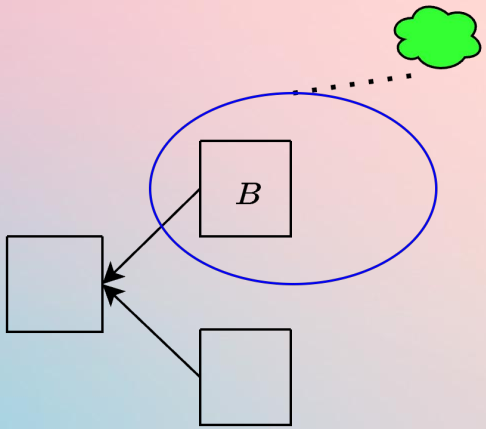


Security guarantees under synchrony and honest majority

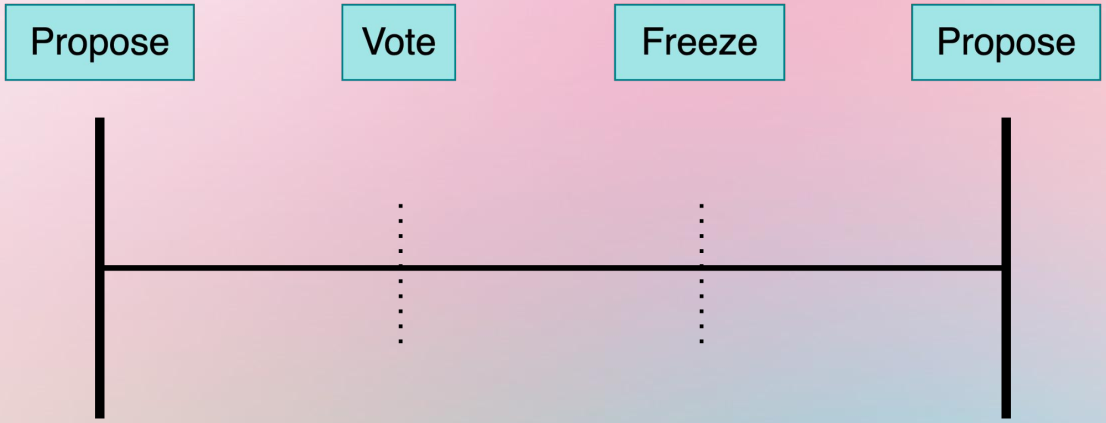
If an honest validator fast confirms B, then every honest validator sees the ⅔ quorum for B before freezing its view.

=> *Unless there are at least ⅓ equivocations (E),* every honest validator sees B as canonical in the next slot and votes for it

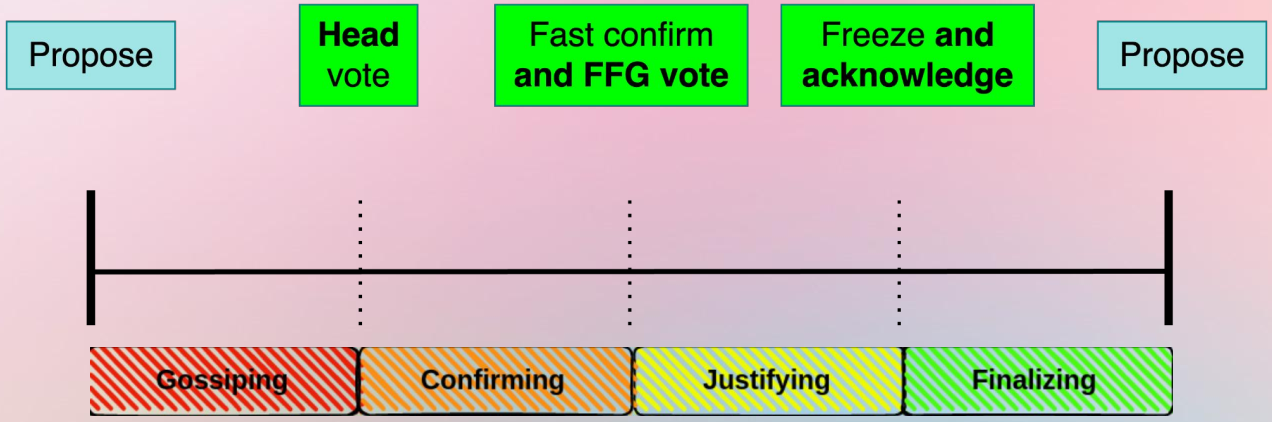
=> Every honest validator votes on the subtree of B in the next slot. B remains in the canonical chain forever *if there is an honest majority*



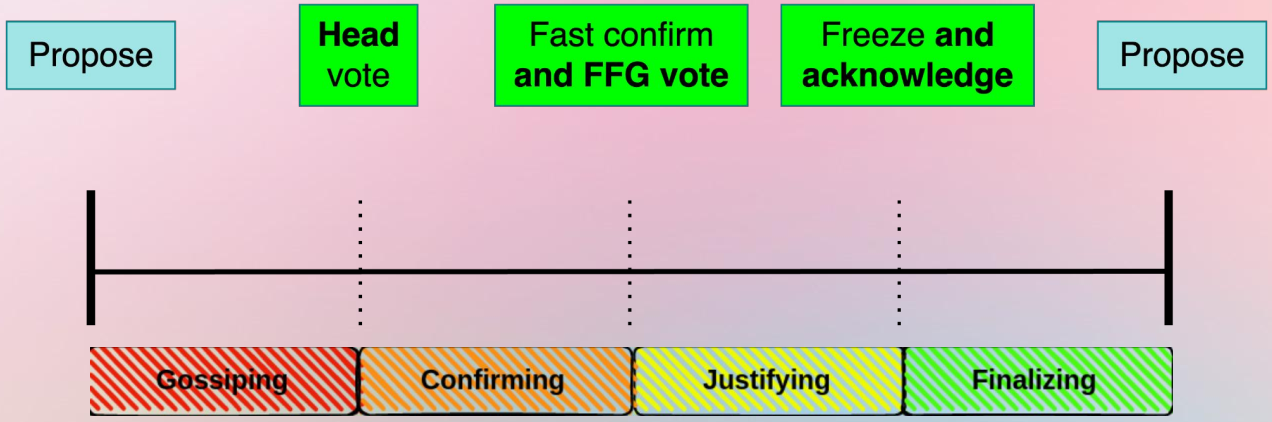
SSF protocol



SSF protocol



SSF protocol

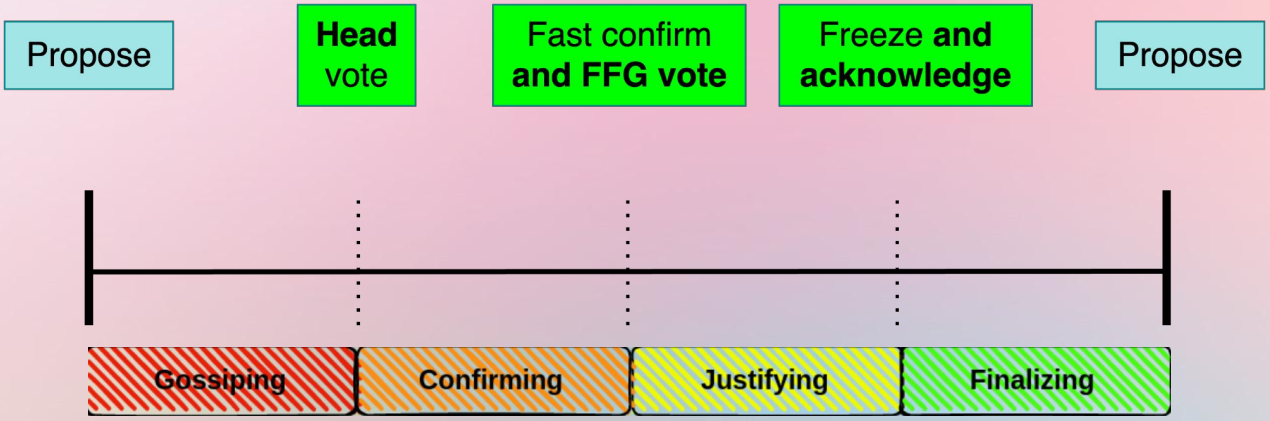


Fork-choice: start from latest justified, run RLMD-GHOST.

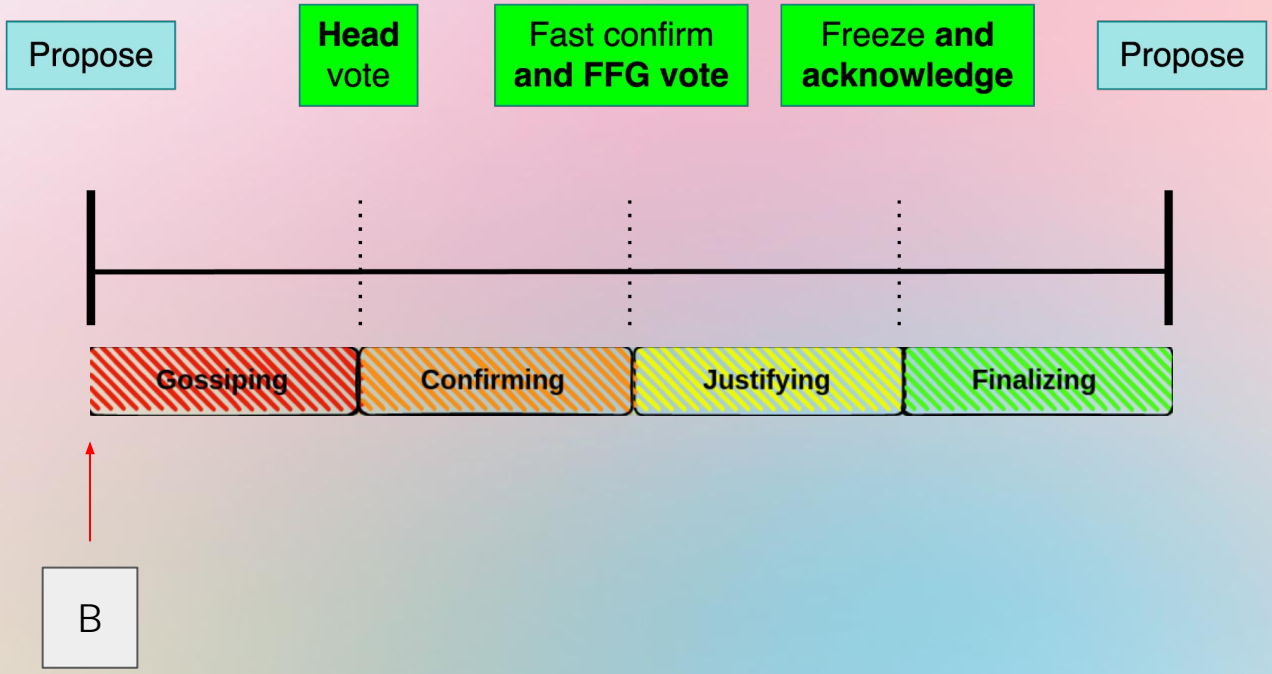
FFG votes: source = latest justified, target block = highest confirmed block descending from justified (justified is always confirmed)

Acknowledgment: if there is a new justified checkpoint, acknowledge it, committing to start your fork-choice from it. A quorum of acks = finality

Progression of honest slot t

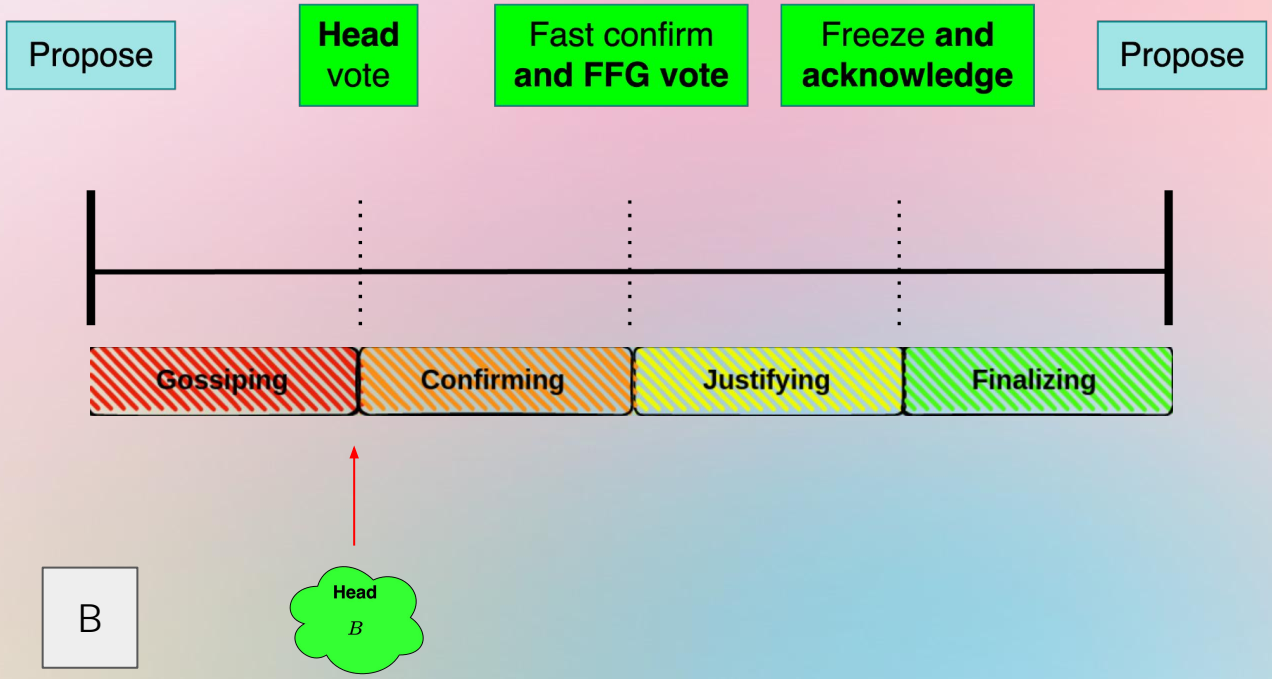


Progression of honest slot t



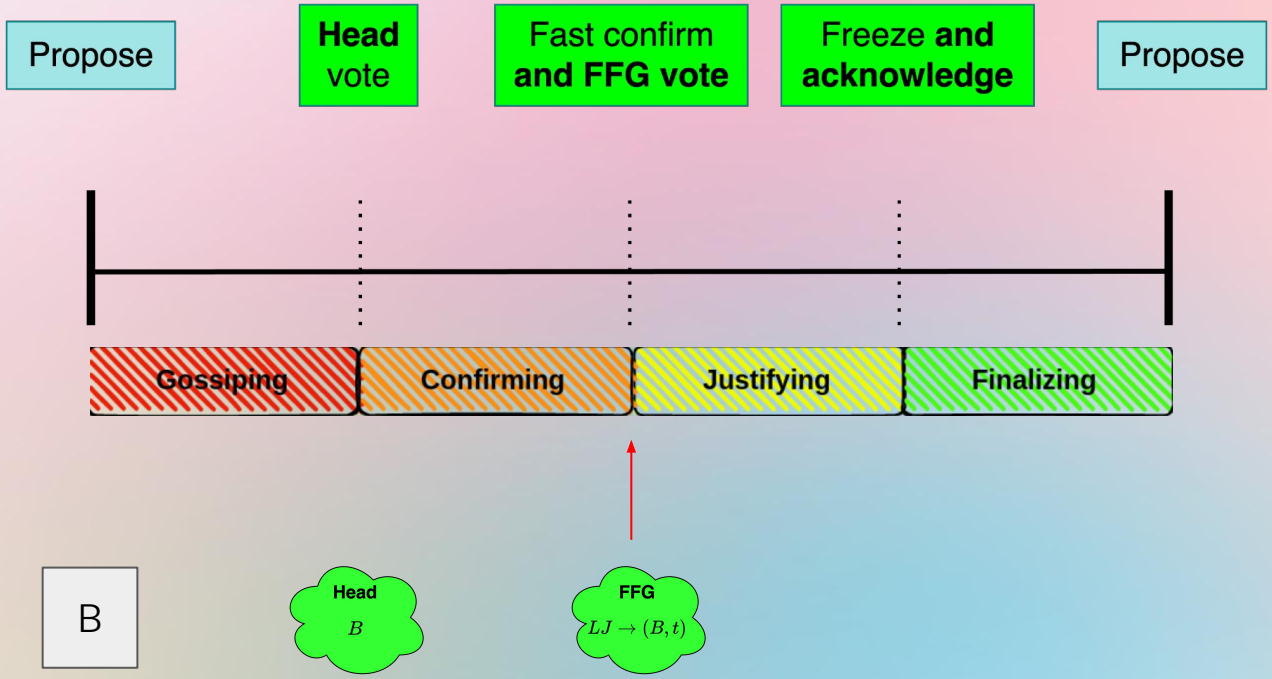
Honest block proposal B is made timely

Progression of honest slot t



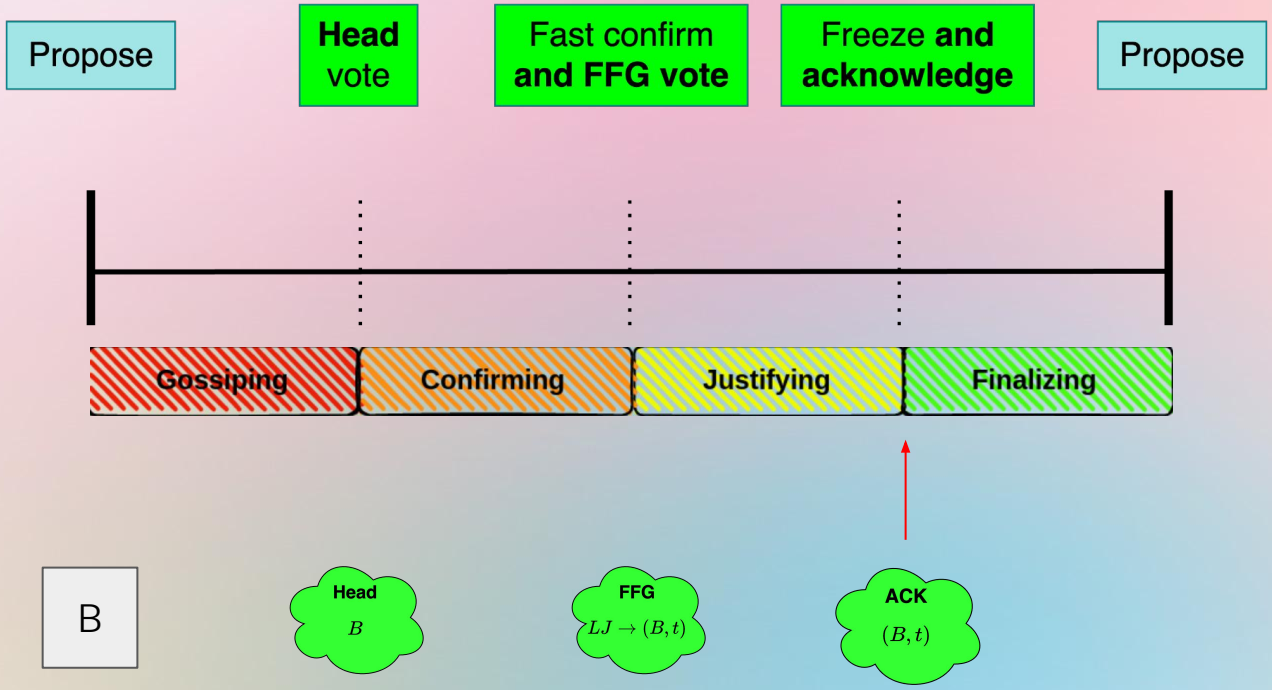
All honest validators vote for B (*view-merge*), forming a ⅔ quorum of head votes for it

Progression of honest slot t



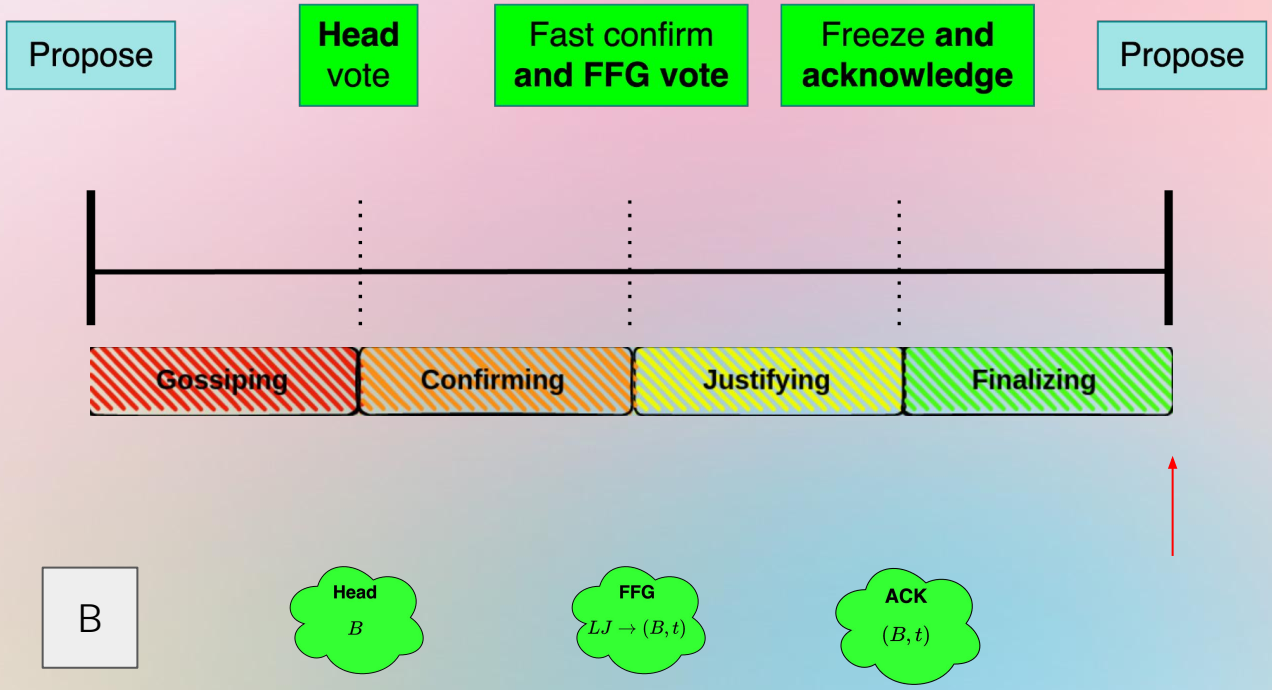
All honest validators get the quorum for B, fast confirm B and use it as their FFG target.
A supermajority link LJ -> (B,t) forms

Progression of honest slot t



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.

Progression of honest slot t



Everyone sees B as finalized.

- Acknowledgments do not affect the fork-choice:

- Acknowledgments do not affect the fork-choice:
  - It is ok to freeze views before their propagation, no need to add another Δ

- Acknowledgments do not affect the fork-choice:
  - It is ok to freeze views before their propagation, no need to add another Δ
  - They do not need to be propagated and aggregated before the next slot, unless we want to incentivize timely acknowledgments through their immediate inclusion on-chain

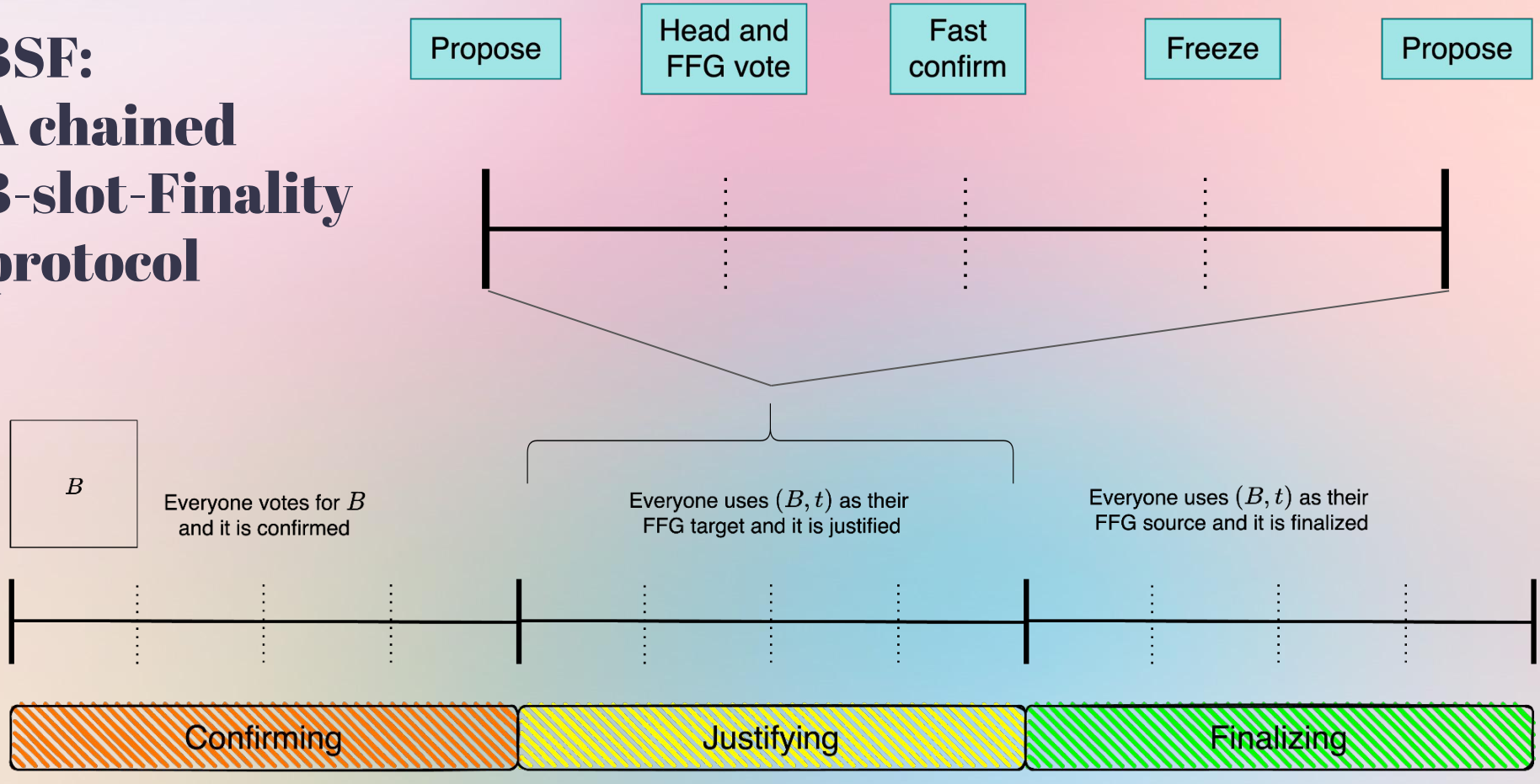
- Acknowledgments do not affect the fork-choice:
  - It is ok to freeze views before their propagation, no need to add another Δ
  - They do not need to be propagated and aggregated before the next slot, unless we want to incentivize timely acknowledgments through their immediate inclusion on-chain
  - An honest proposal will be seen by everyone as the latest justified at the end of its slot (before freezing of views), so there is no chance to reorg honest proposals by revealing new justified checkpoints. Acknowledgments are irrelevant to this dynamic, they only let finality be detected and make it accountable

- Acknowledgments do not affect the fork-choice:
  - It is ok to freeze views before their propagation, no need to add another Δ
  - They do not need to be propagated and aggregated before the next slot, unless we want to incentivize timely acknowledgments through their immediate inclusion on-chain
  - An honest proposal will be seen by everyone as the latest justified at the end of its slot (before freezing of views), so there is no chance to reorg honest proposals by revealing new justified checkpoints. Acknowledgments are irrelevant to this dynamic, they only let finality be detected and make it accountable
- We need at least the latest *off-chain* justification to be taken into account by the fork-choice

Removing filtering

- A justification is processed against *the state of the target* (not source, so inactivity leak works automatically). In particular we care about:
  - Effective balances
  - Active and slashed status
- For on-chain processing over a short period of time (e.g. target at slot 100, justification processed at slot 110), we can keep small diffs in the state.
- Exceptionally, any chain containing a justified checkpoint can process its justification by being provided the relevant attestations + a witness encoding the necessary state.
- This kind of “chain agnostic” justification certificate is also used for off-chain justification

3SF: A chained 3-slot-Finality protocol

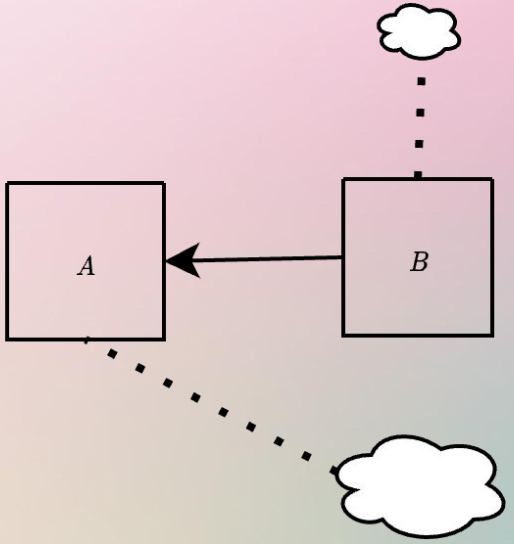


Fork-choice in the Ethereum Roadmap

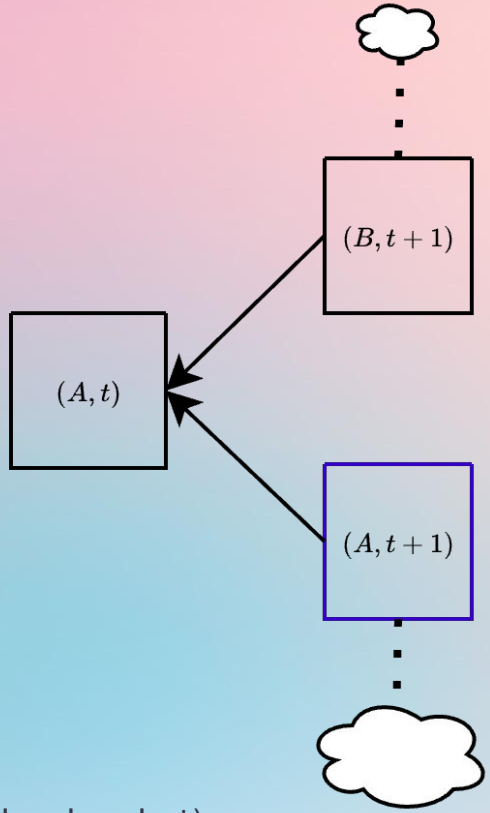
Fork-choice in the Ethereum Roadmap

- SSF (or fast finality):
  - Optimally secure consensus protocol
  - Fast confirmation of high value transactions, e.g. fast deposits on exchanges
  - Faster bridging
  - Better L2 interoperability
- ePBS
- DAS

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

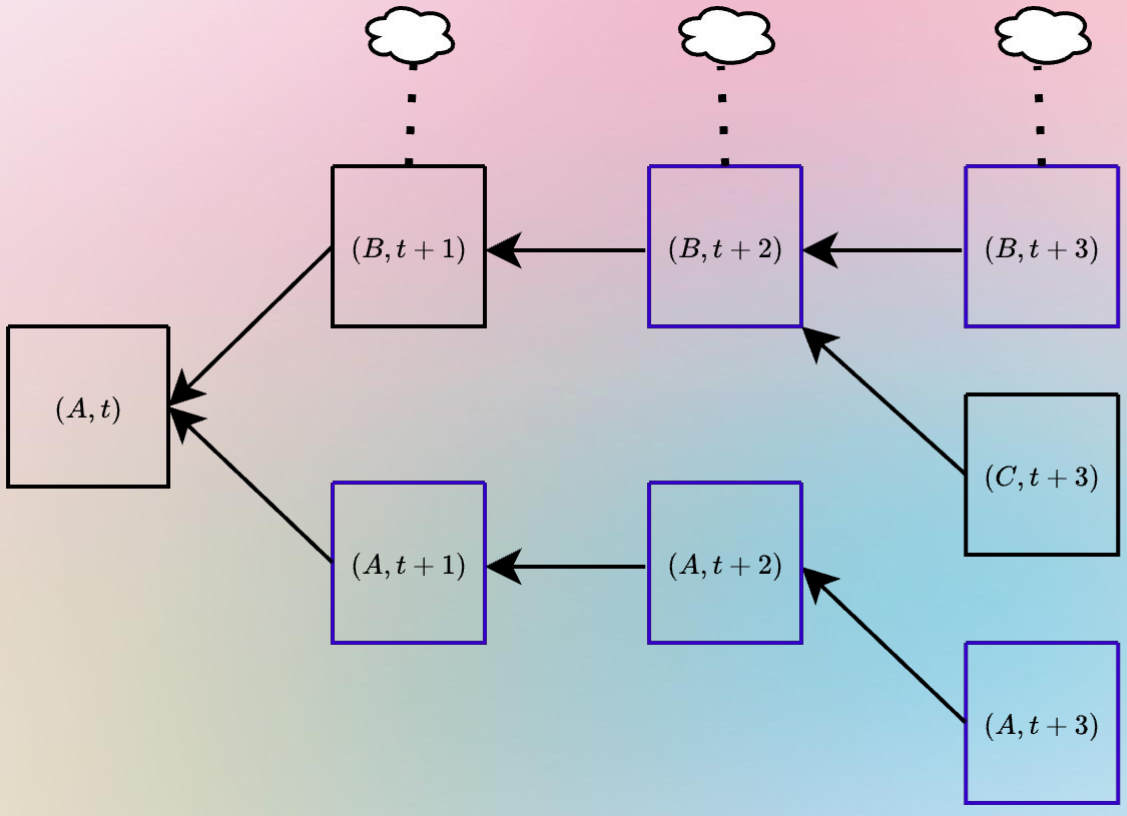


Status quo

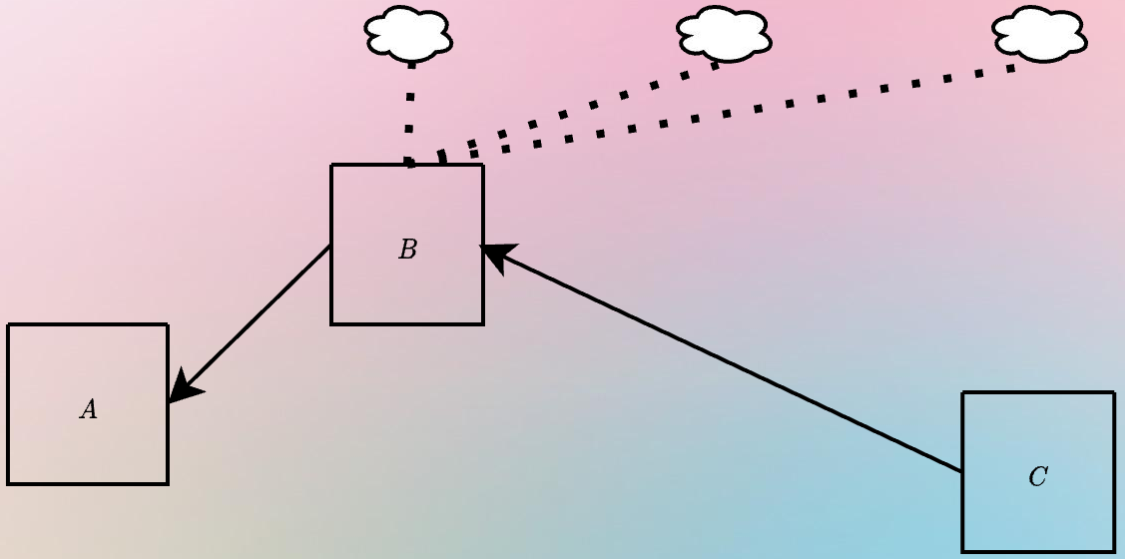


(Block, slot)

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



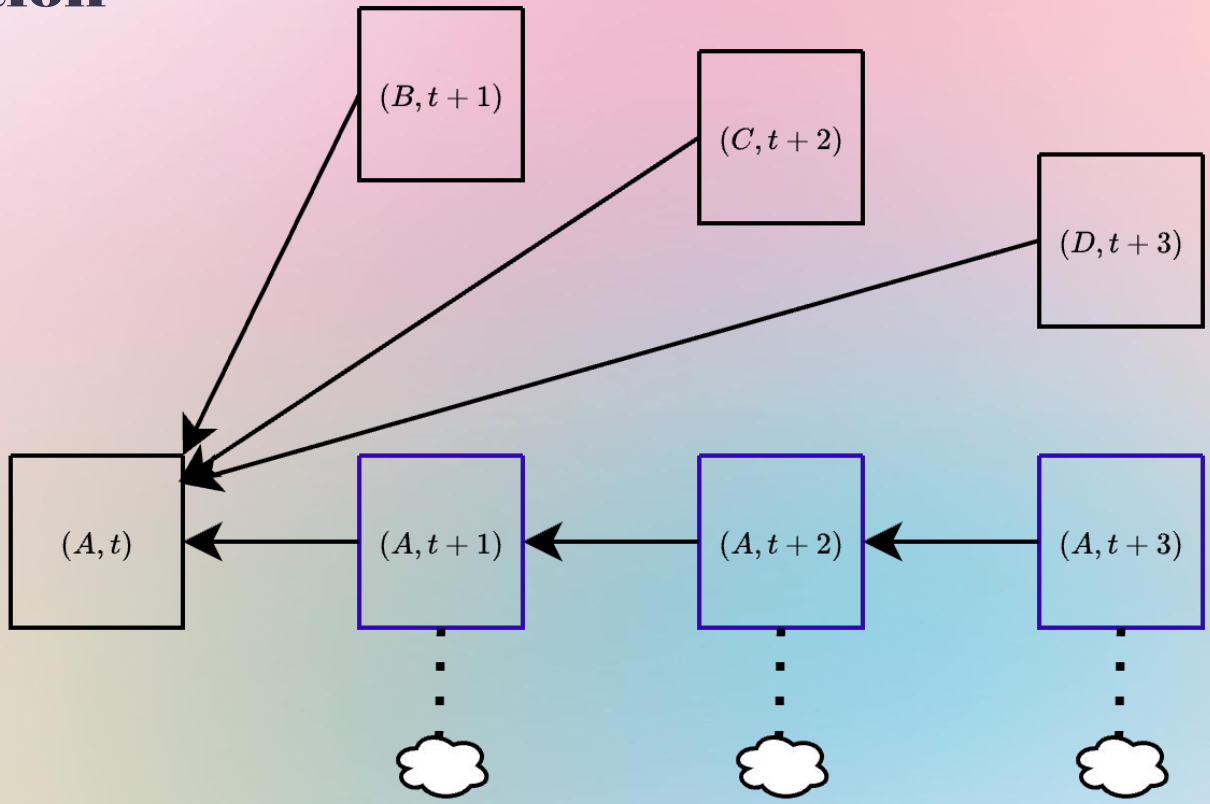
Block-slot fork-choice: enshrining empty slots



Applications

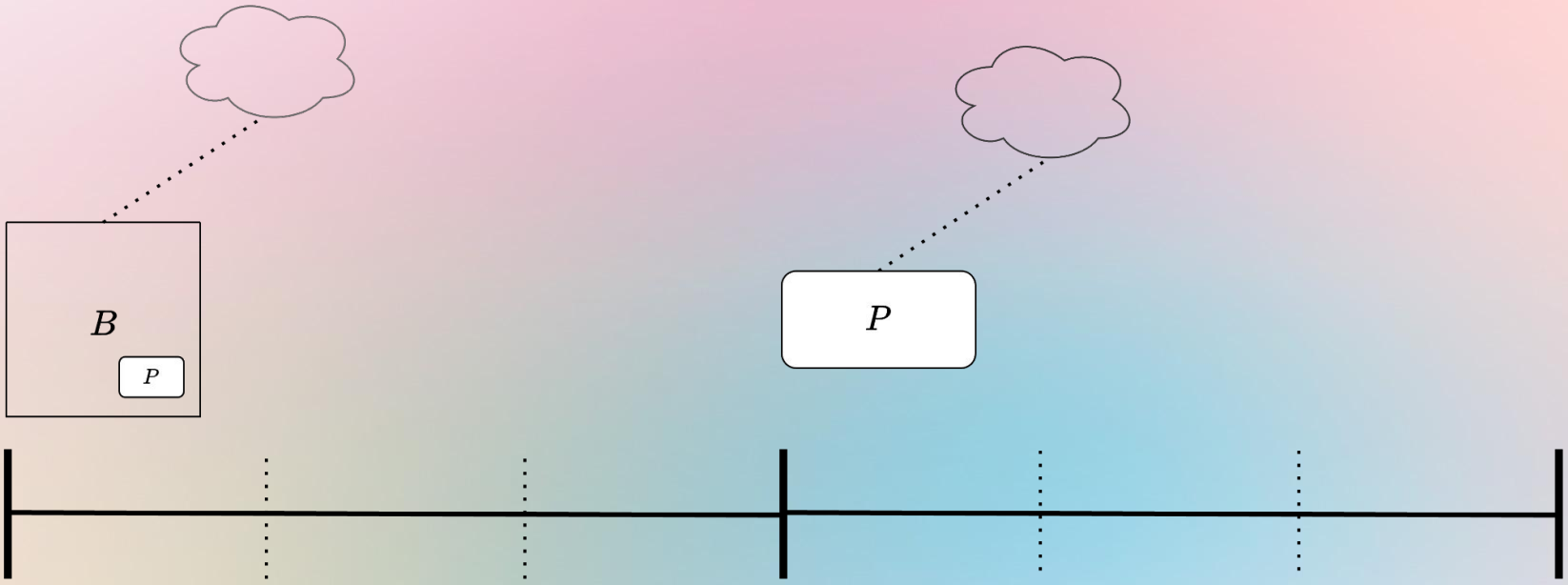
- Committee-enforceable properties: if a proposal does fulfill a certain property, the attesting committee votes “against it” (for the empty block)
  - ILs satisfaction (one possible way of doing this)
  - Bid maximization (MEV burn)
- ePBS fork-choice: proposers accept bids from builders. If they don’t 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

Challenge: higher latency halts block production

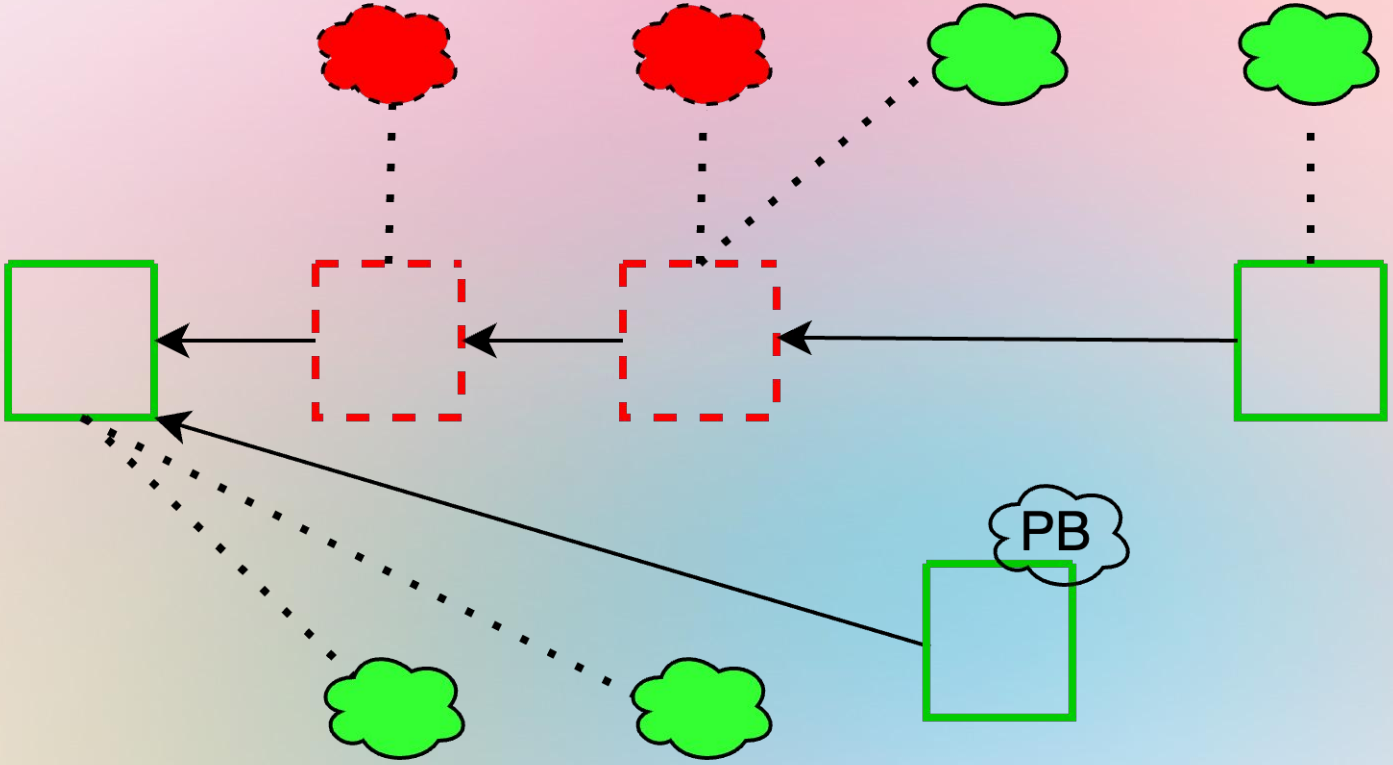


Fork-choice in the Ethereum Roadmap: ePBS

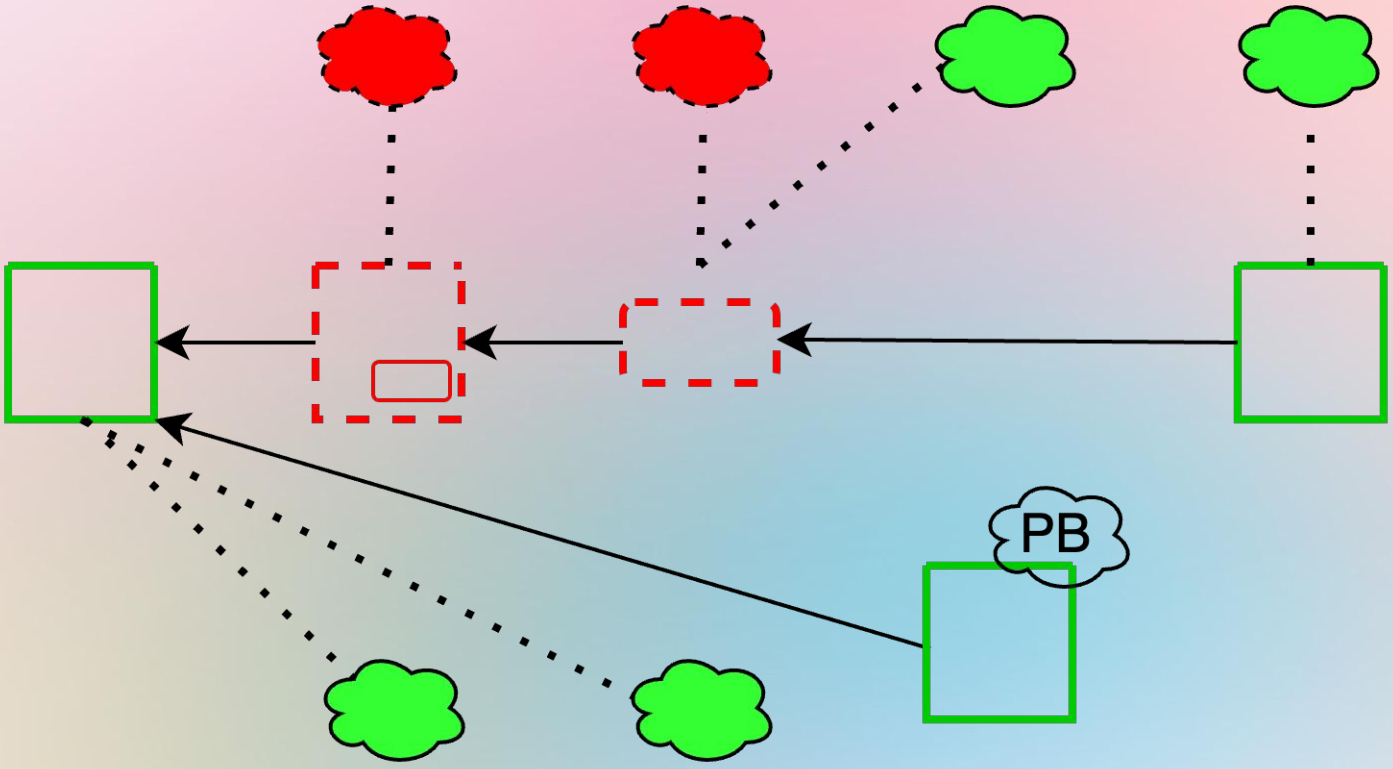
Two-slot ePBS



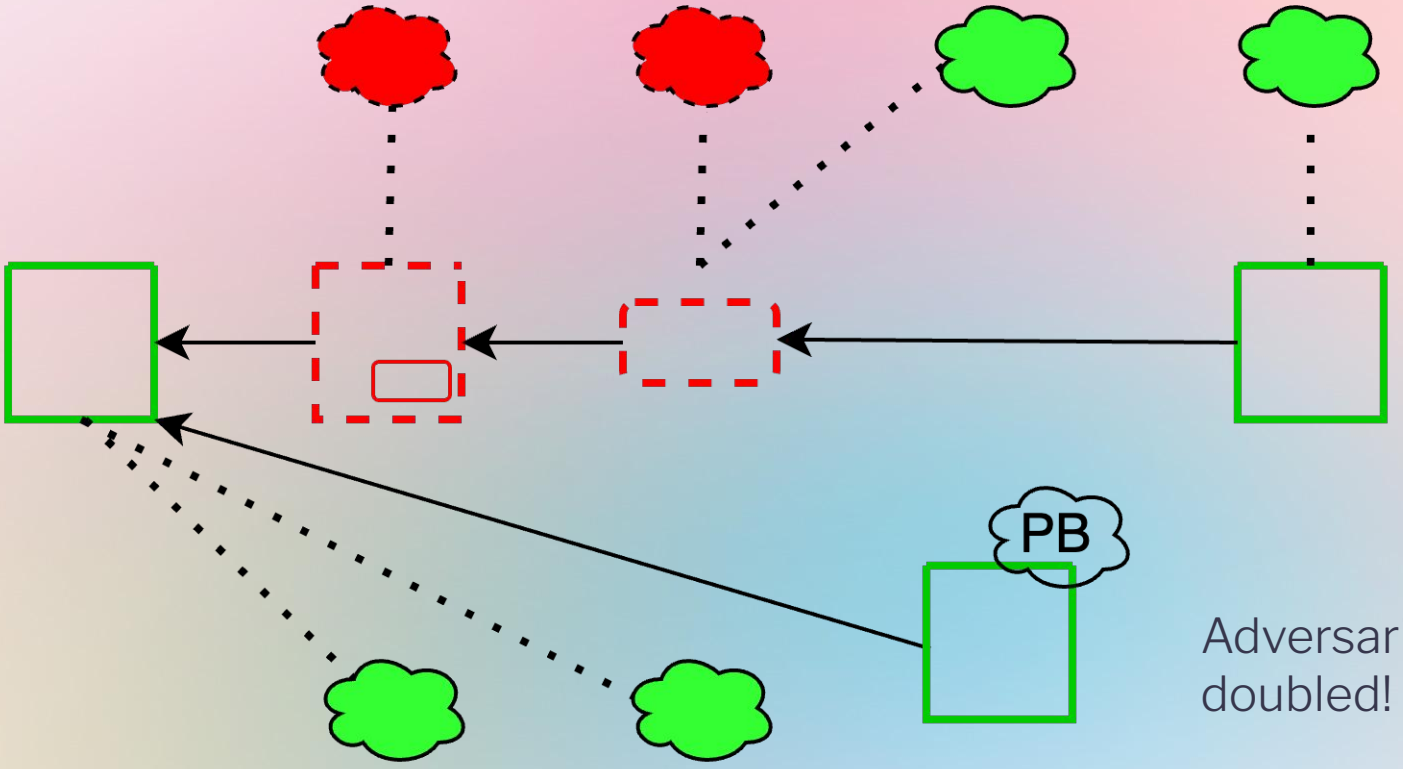
Ex-ante reorgs, ePBS version



Ex-ante reorgs, ePBS version

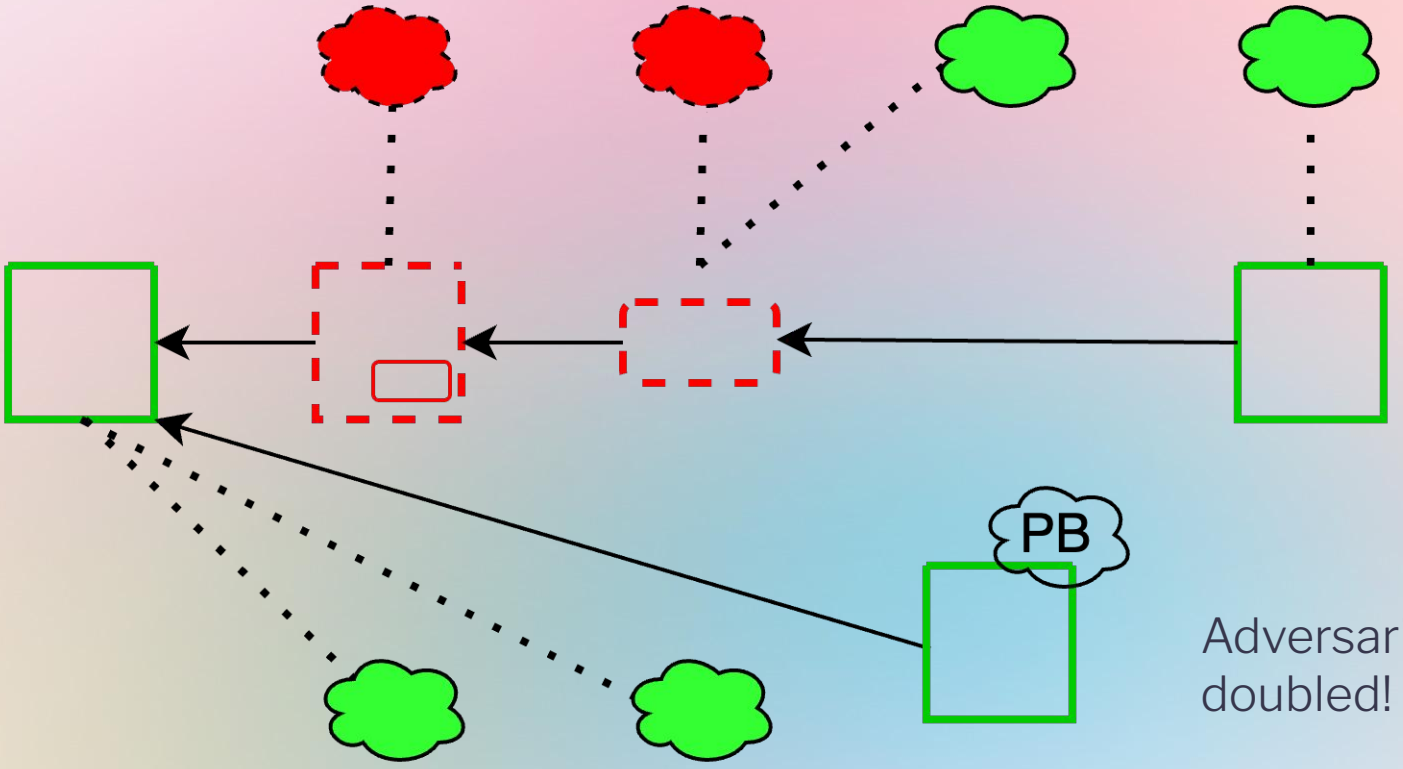


Ex-ante reorgs, ePBS version



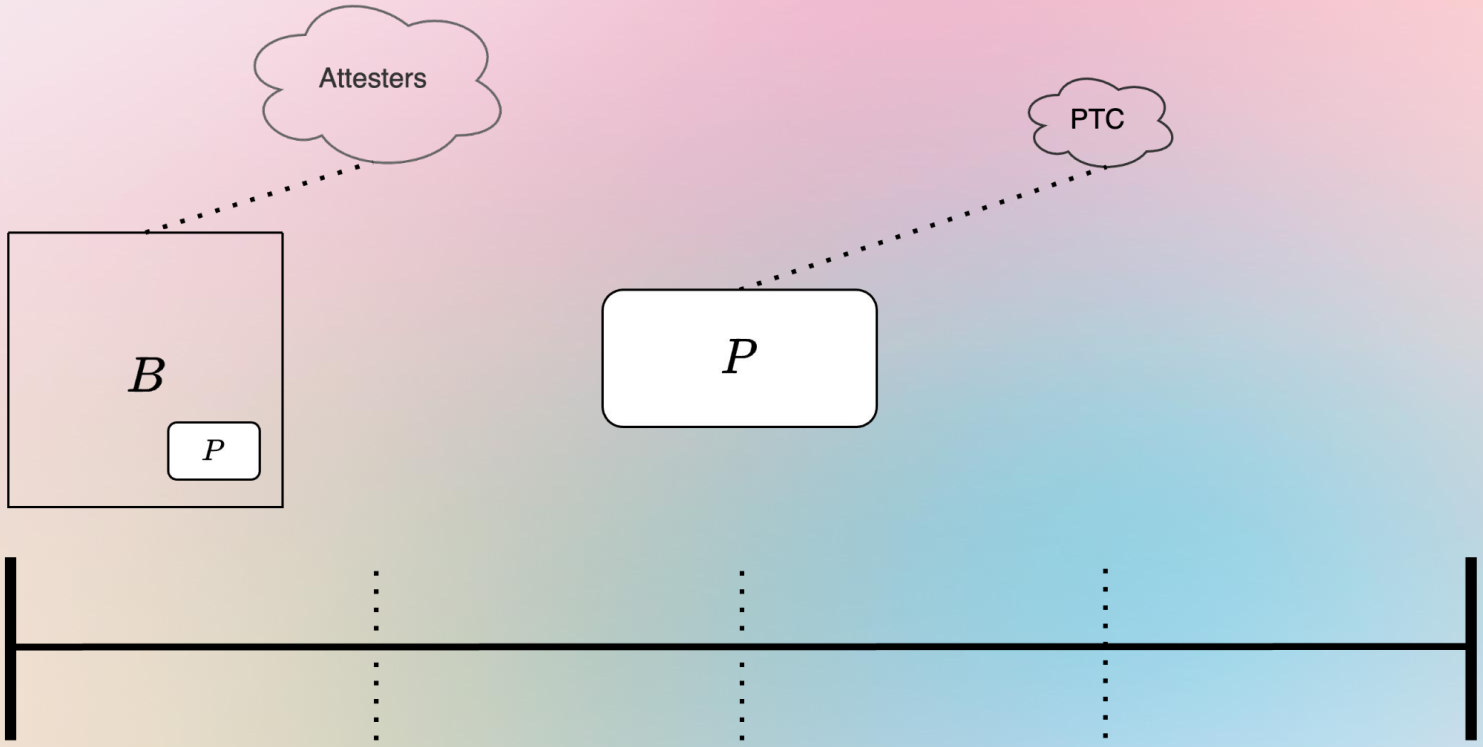
Adversarial power doubled!

Ex-ante reorgs, ePBS version

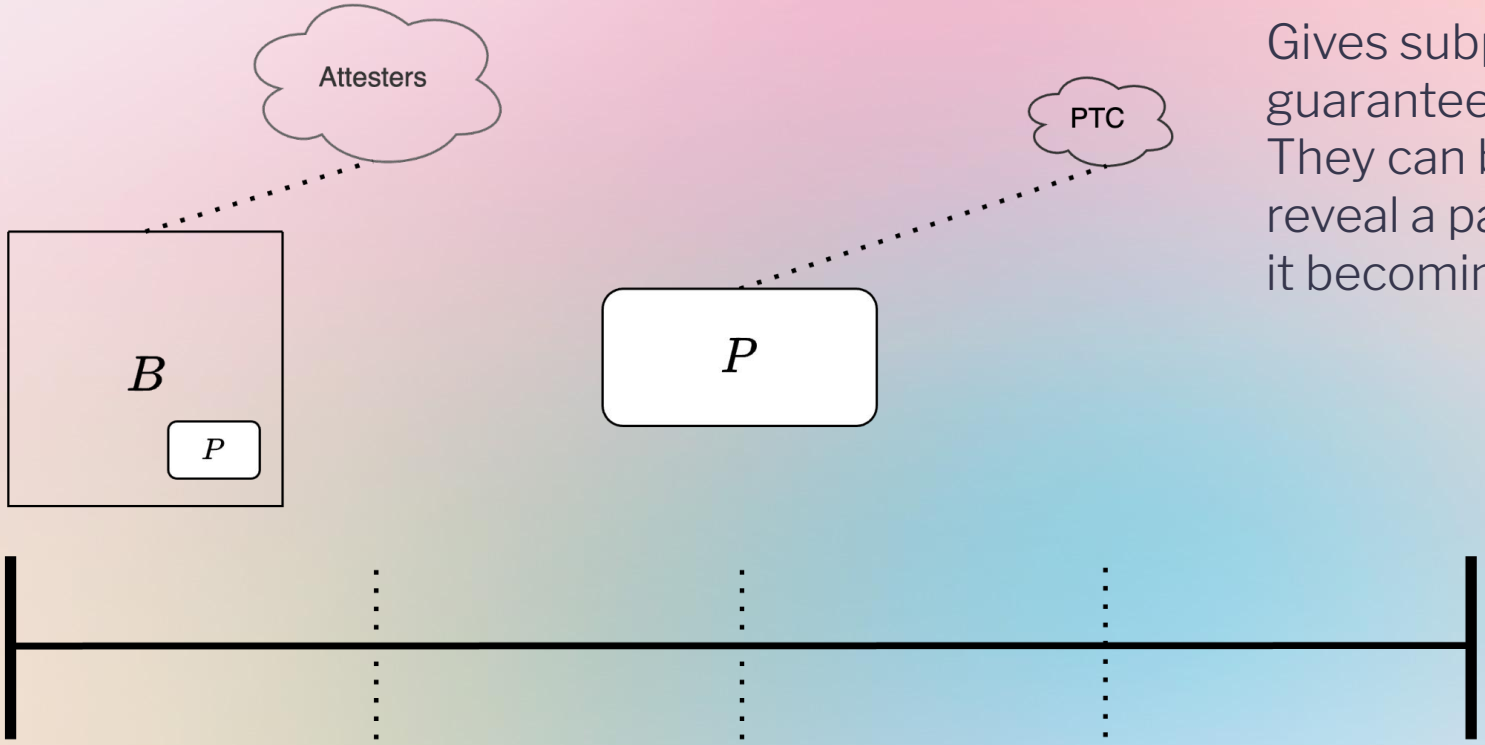


Adversarial power doubled!

One-slot ePBS with PTC (Payload Timeliness Committee)

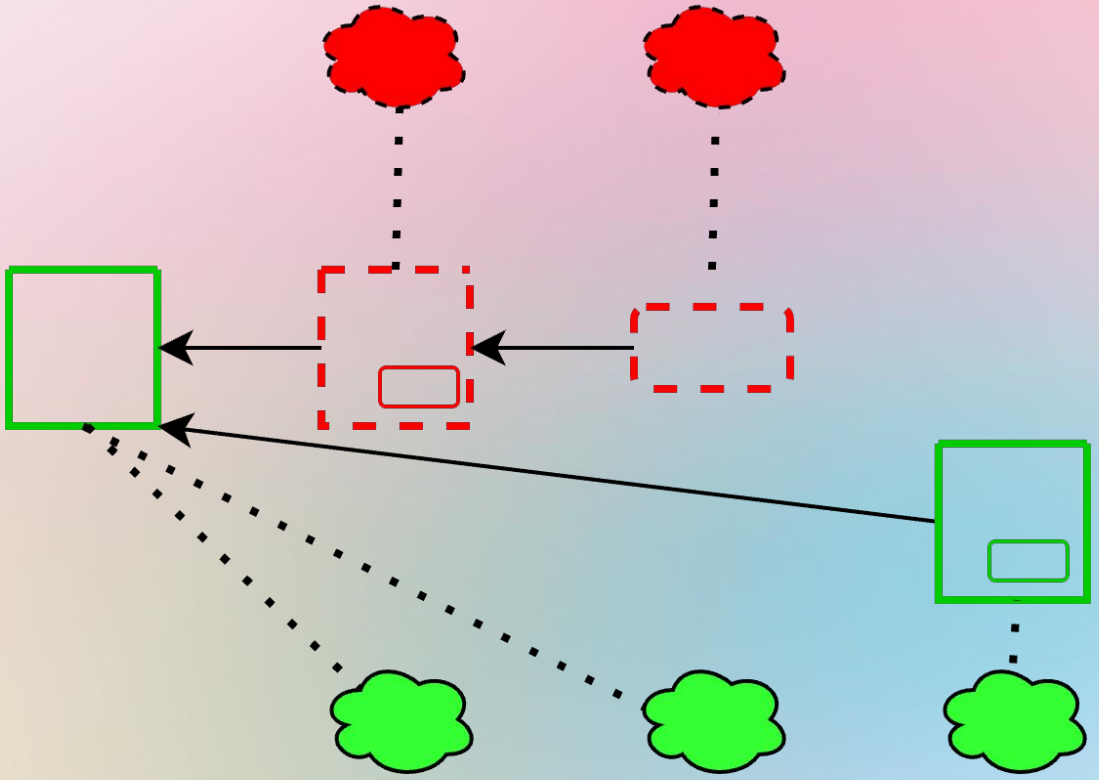


One-slot ePBS with PTC (Payload Timeliness Committee)

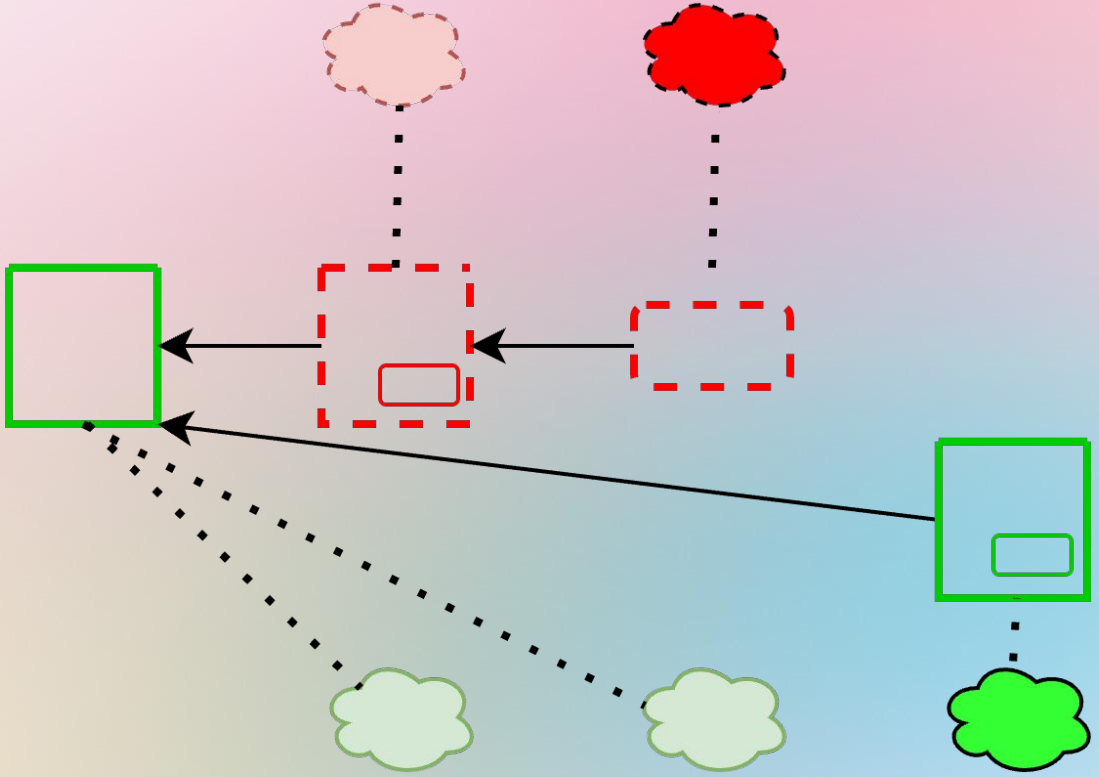


Gives subpar guarantees to builders… They can be forced to reveal a payload without it becoming canonical

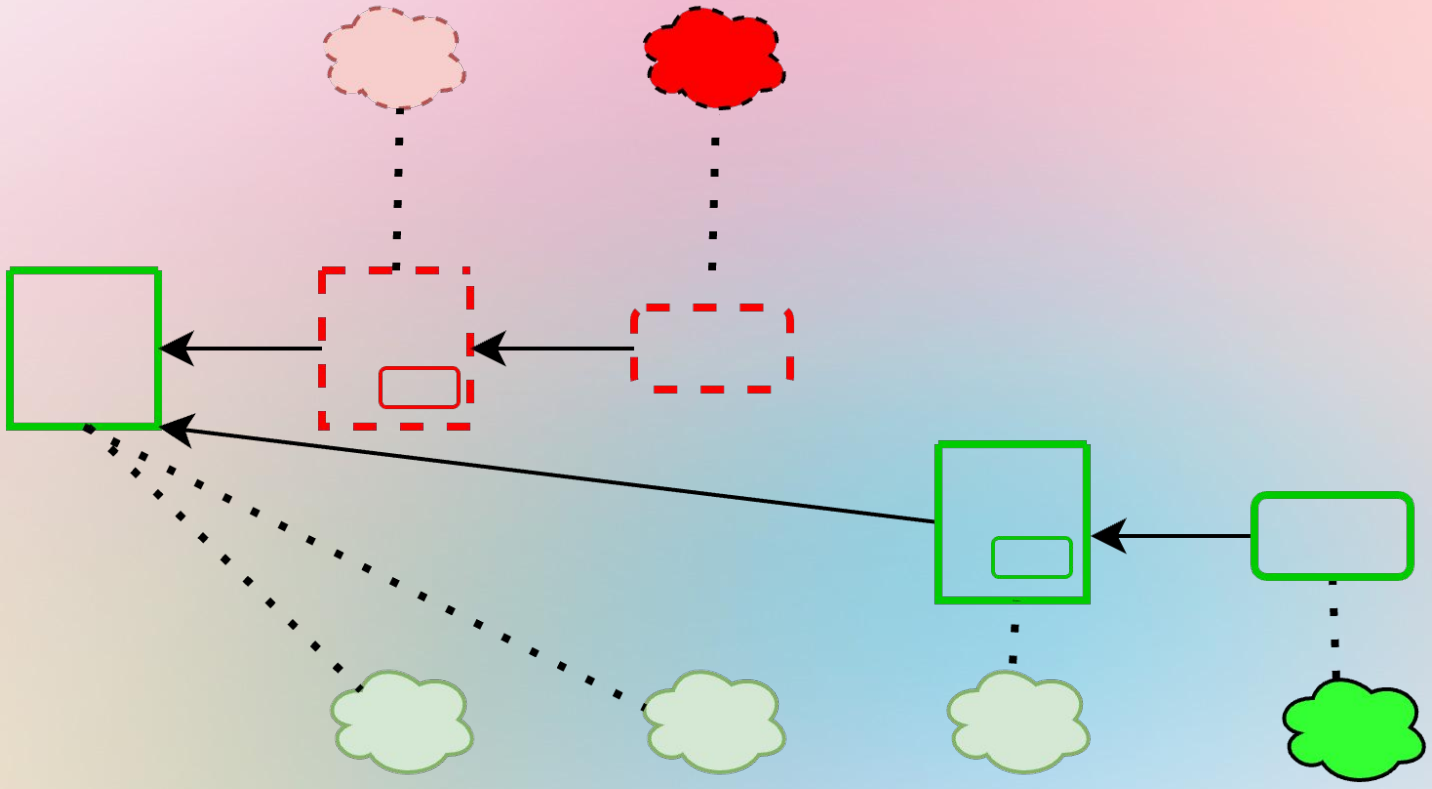
RLMD-GHOST with ePBS



RLMD-GHOST with ePBS

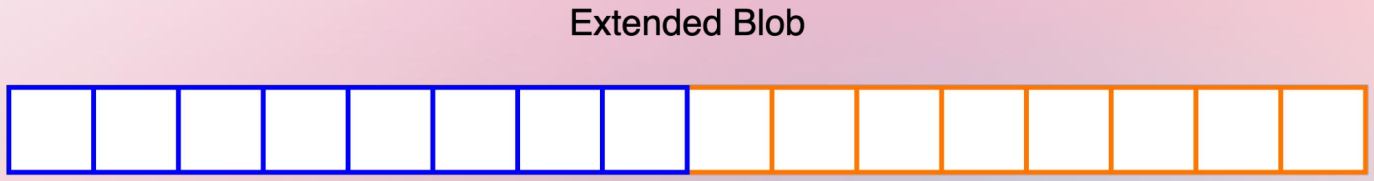


RLMD-GHOST with ePBS



Fork-choice in the Ethereum Roadmap: PeerDAS

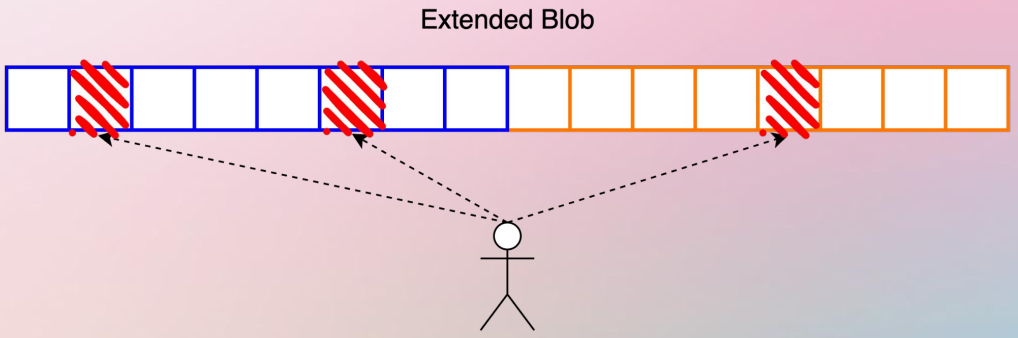
Data Availability Sampling



We *extend* blobs, introducing redundancy which allows **reconstruction** of the full data whenever having at least 50% of it

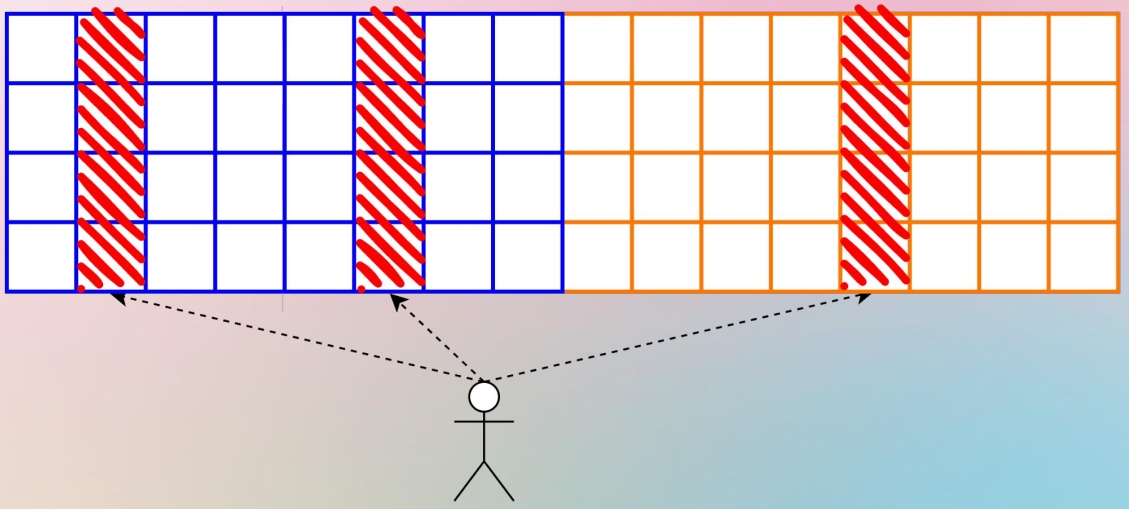
=> Checking availability of a blob only requires making sure that 50% of it is available

Data Availability Sampling



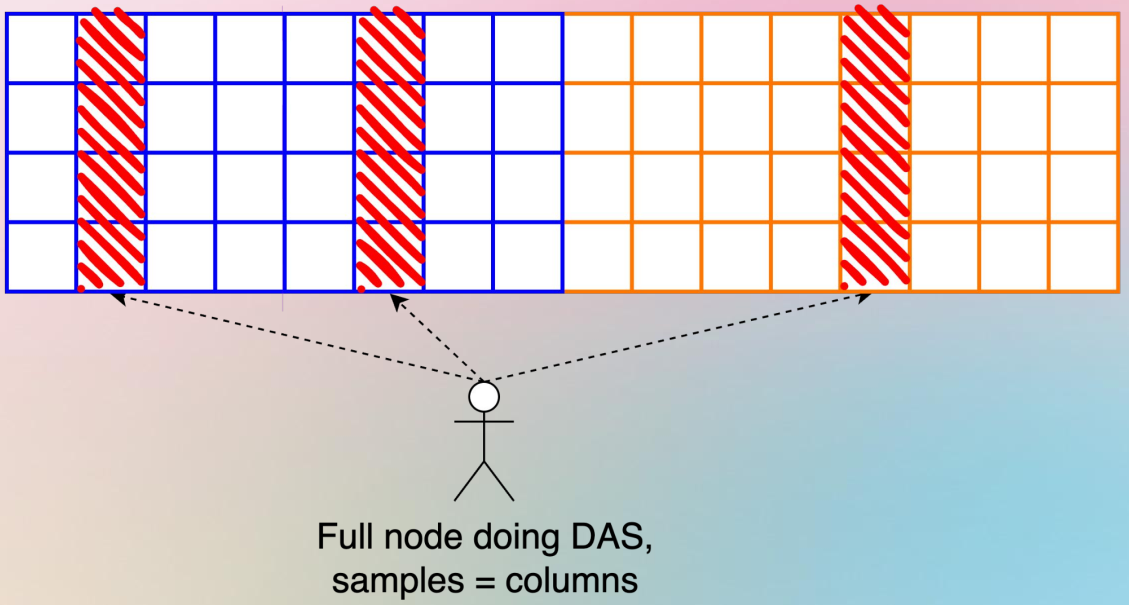
Ultimately, we want full nodes to only download a few chunks of data of their choice (**samples**) to verify that the data is available, instead of the whole data

Data Availability Sampling



Full node doing DAS,
samples = columns

Data Availability Sampling



Like sampling for each blob, but using the same choice of indices

Fork-choice implications

If we do enough sampling *before voting*, we get a great global property: Only at most a small percentage **δ** of the honest validators will see unavailable data as available

Fork-choice implications

If we do enough sampling *before voting*, we get a great global property: Only at most a small percentage **δ** of the honest validators will see unavailable data as available

=> if we have > ½ + δ honest validators,
a majority always votes *against* unavailable blocks

Fork-choice implications

If we do enough sampling *before voting*, we get a great global property: Only at most a small percentage **δ** of the honest validators will see unavailable data as available

=> if we have > ½ + **δ** 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

Fork-choice implications

If we do enough sampling *before voting*, we get a great global property: Only at most a small percentage **δ** of the honest validators will see unavailable data as available

=> if we have > ½ + **δ** 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

The fork-choice essentially works as if we didn't have DAS at all!

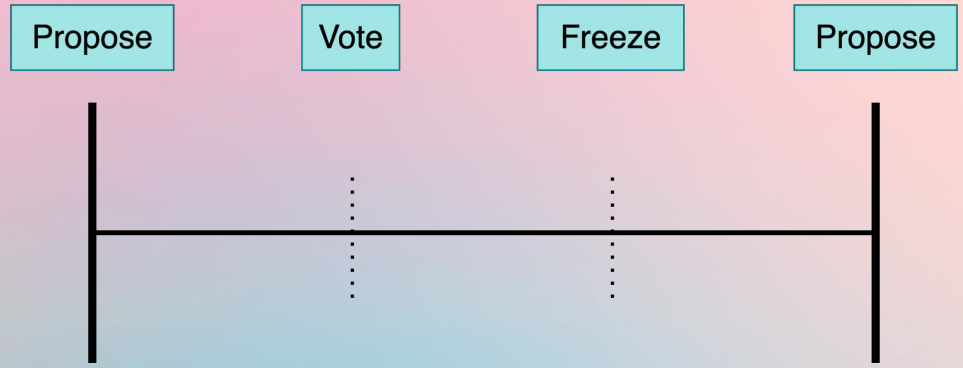
Thank you!

Alternative available chain

Merge-less available chain

Same structure as before, but:

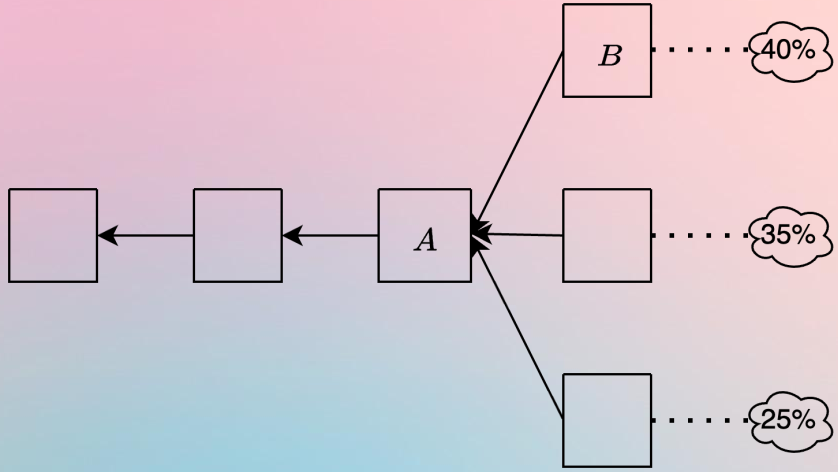
- Fork-choice change: head is the highest block supported by a majority of the voting weight
- No view-merge message necessary.



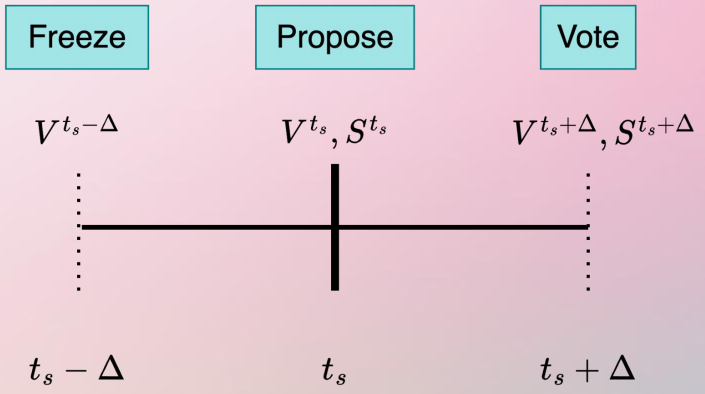
Merge-less available chain

Same structure as before, but:

- Fork-choice change: head is the highest block supported by a majority of the voting weight
- No view-merge message necessary.



Synchronization mechanism: time-shifted quorum

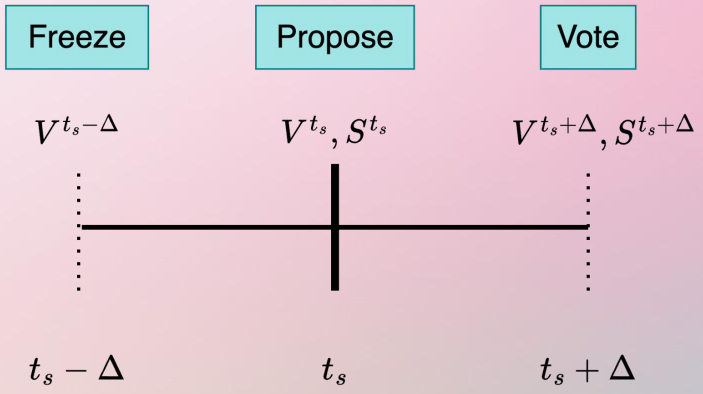


V^t = latest unexpired votes from non-equivocators received by time t

S^t = voters by time t

t_s = time of slot s

Synchronization mechanism: time-shifted quorum



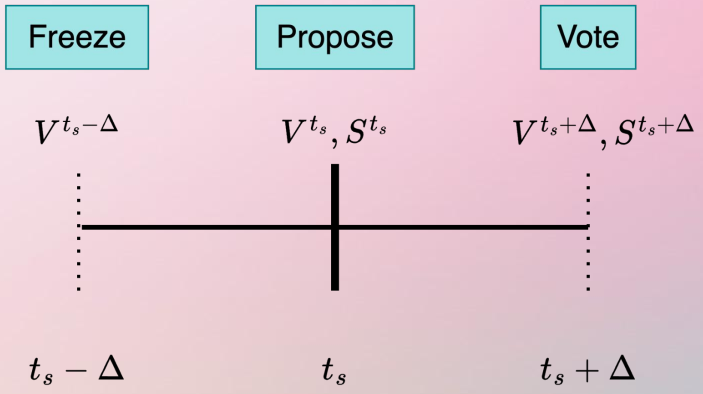
V^t = latest unexpired votes from non-equivocators received by time t

S^t = voters by time t

t_s = time of slot s

$V^{t_s-\Delta} \subseteq V^{t_s}$ and $S^{t_s} \subseteq S^{t_s+\Delta}$

Synchronization mechanism: time-shifted quorum



V^t = latest unexpired votes from non-equivocators received by time t
S^t = voters by time t
t_s = time of slot s

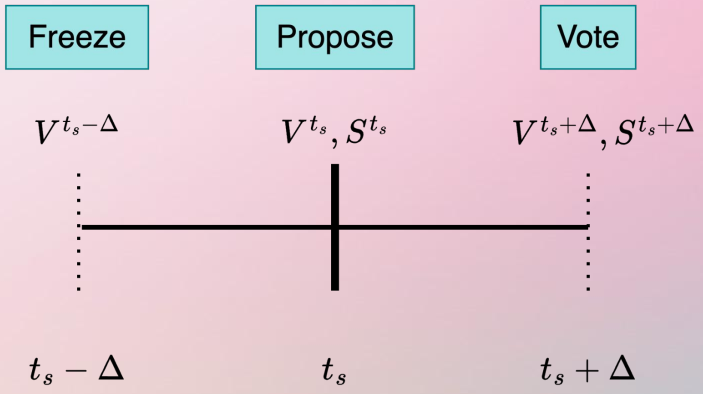
$V^{t_s-\Delta} \subseteq V^{t_s}$ and $S^{t_s} \subseteq S^{t_s+\Delta}$

B is canonical when voting at slot s if

$$|V_B^{t_s-\Delta}| > |S^{t_s+\Delta}|/2$$
$$\implies |V_B^{t_s}| \geq |V_B^{t_s-\Delta}| > |S^{t_s+\Delta}|/2 \geq |S^{t_s}|/2$$
$$\implies |V_B^{t_s}| > |S^{t_s}|/2$$

$\implies$ B is also canonical for the proposer

Synchronization mechanism: time-shifted quorum



V^t = latest unexpired votes from non-equivocators received by time t

S^t = voters by time t

t_s = time of slot s

~~$V^{t_s-\Delta} \subseteq V^{t_s}$ and $S^{t_s} \subseteq S^{t_s+\Delta}$~~

B is canonical when voting at slot s if

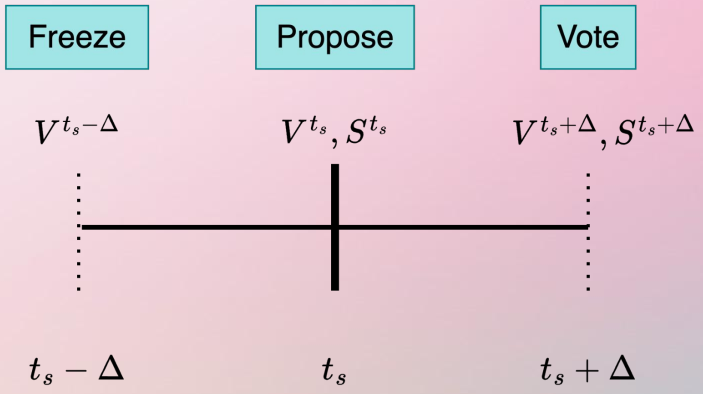
$$|V_B^{t_s-\Delta}| > |S^{t_s+\Delta}|/2$$

$$\implies |V_B^{t_s}| \geq |V_B^{t_s-\Delta}| > |S^{t_s+\Delta}|/2 \geq |S^{t_s}|/2$$

$$\implies |V_B^{t_s}| > |S^{t_s}|/2$$

$\implies$ B is also canonical for the proposer

Synchronization mechanism: time-shifted quorum



V^t = latest unexpired votes from non-equivocators received by time t

S^t = voters by time t

t_s = time of slot s

$V^{t_s-\Delta} \subseteq V^{t_s}$ and $S^{t_s} \subseteq S^{t_s+\Delta}$

B is canonical when voting at slot s if

$$|V_B^{t_s-\Delta} \cap V^{t_s+\Delta}| > |S^{t_s+\Delta}|/2$$

If an equivocation or a later vote is seen by the proposer, it will be seen by the voter as well, and discarded by the intersection with its current view