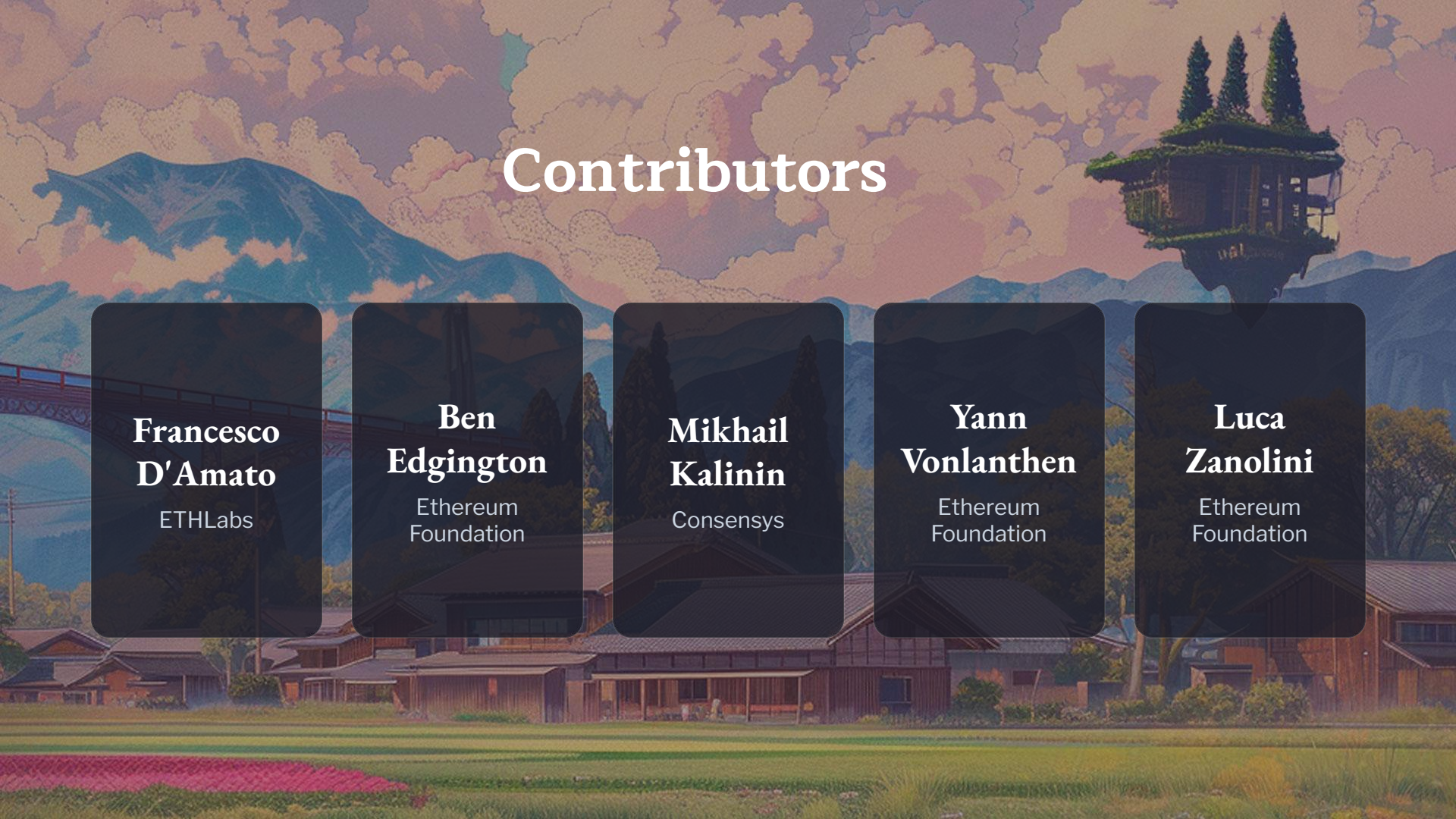

Decoupled Consensus



Contributors

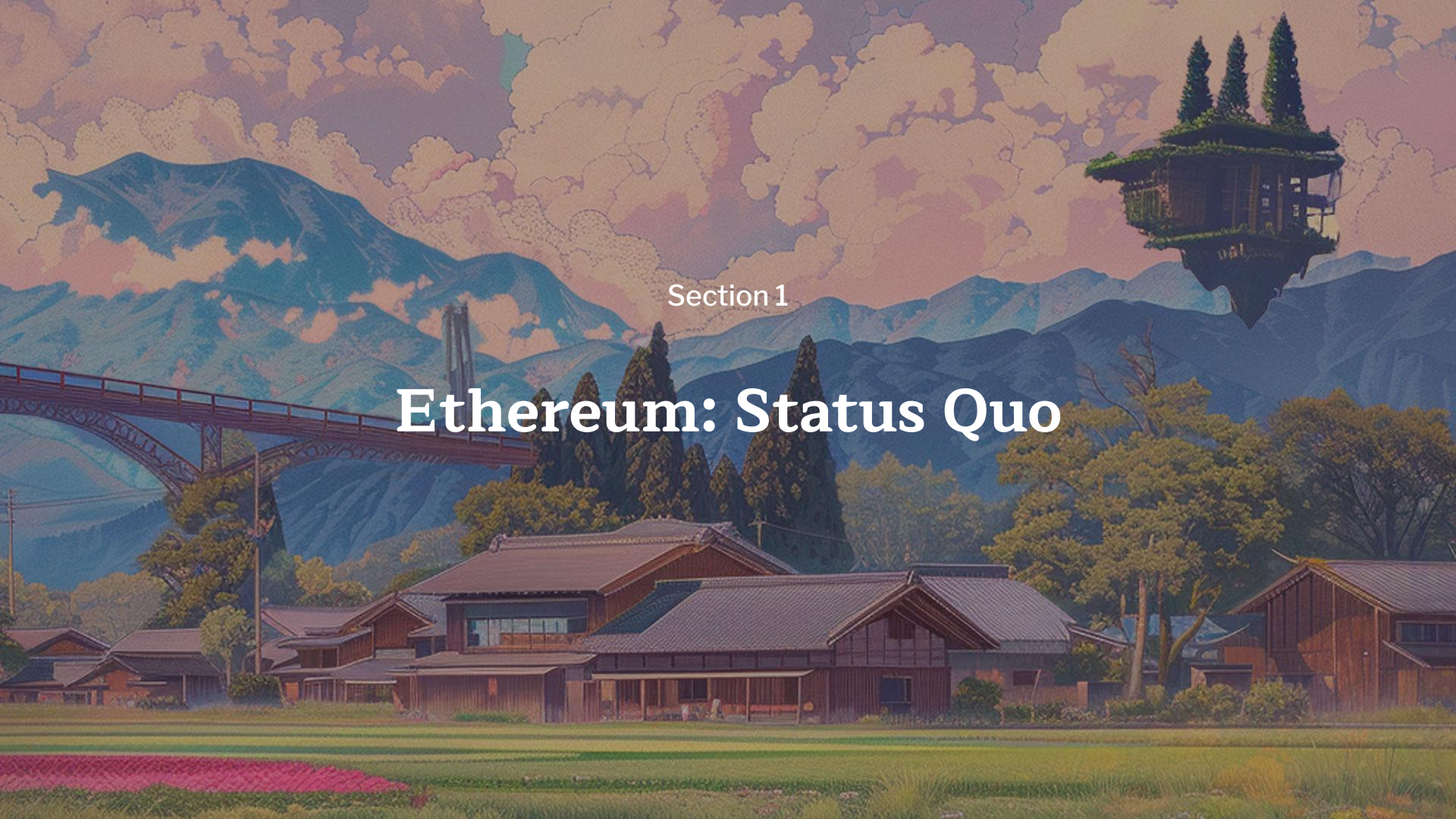
Francesco D'Amato
ETHLabs

Ben Edgington
Ethereum Foundation

Mikhail Kalinin
Consensys

Yann Vonlanthen
Ethereum Foundation

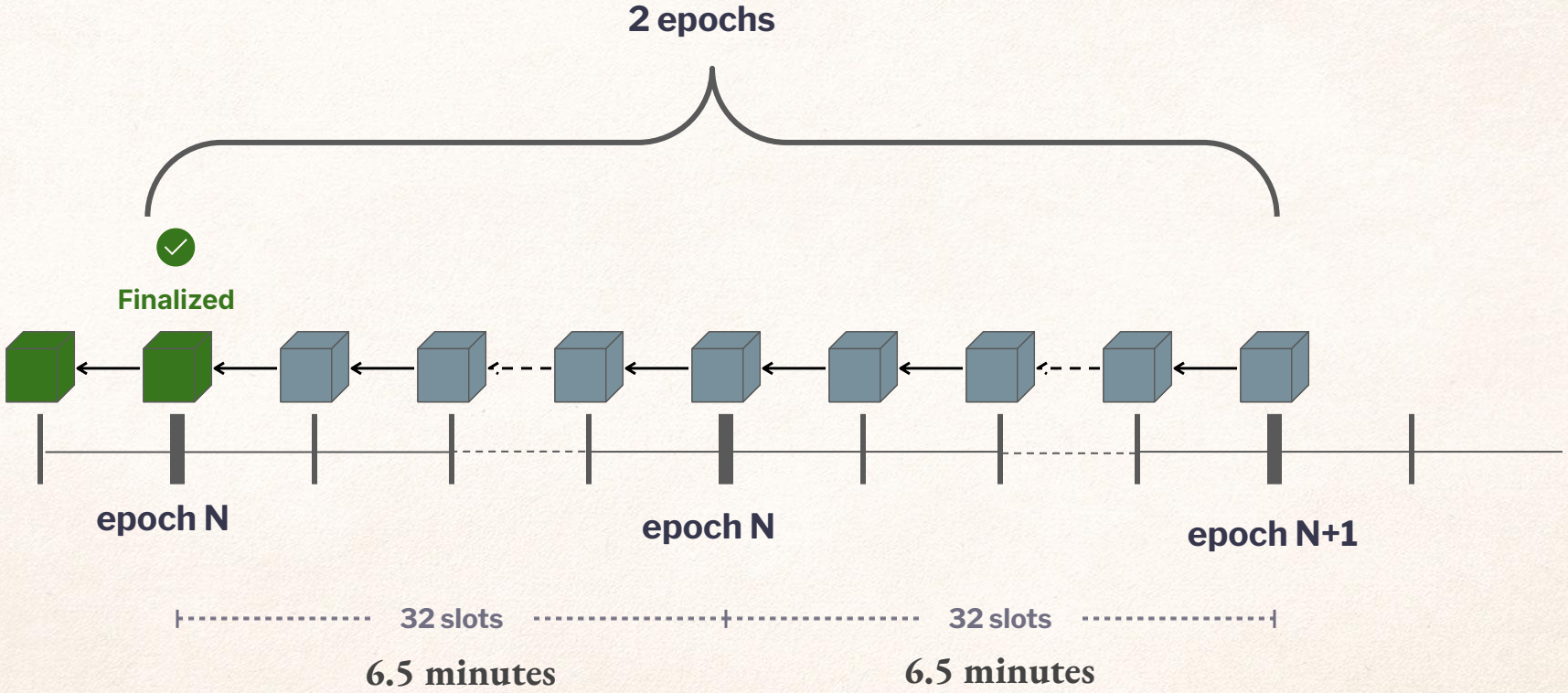
Luca Zanolini
Ethereum Foundation



Section 1

Ethereum: Status Quo

Today's Metrics



Today's Metrics

Block Time

12s

Time required to add a new block to the Ethereum blockchain.

Finality Time

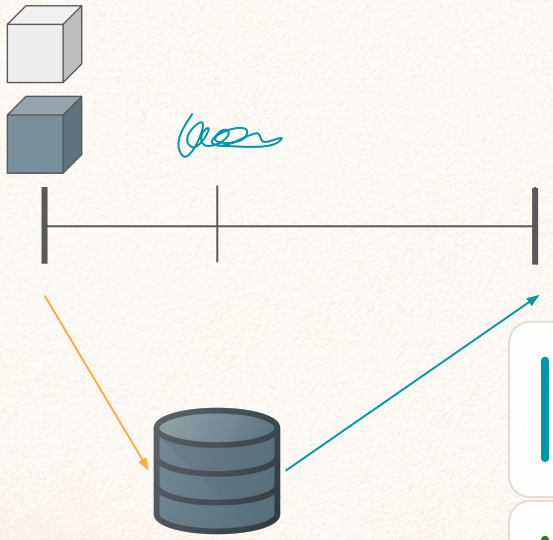
Best Case

64 blocks = 13 minutes

Average Case

80 blocks = 16 minutes

Coupled timelines in Gasper



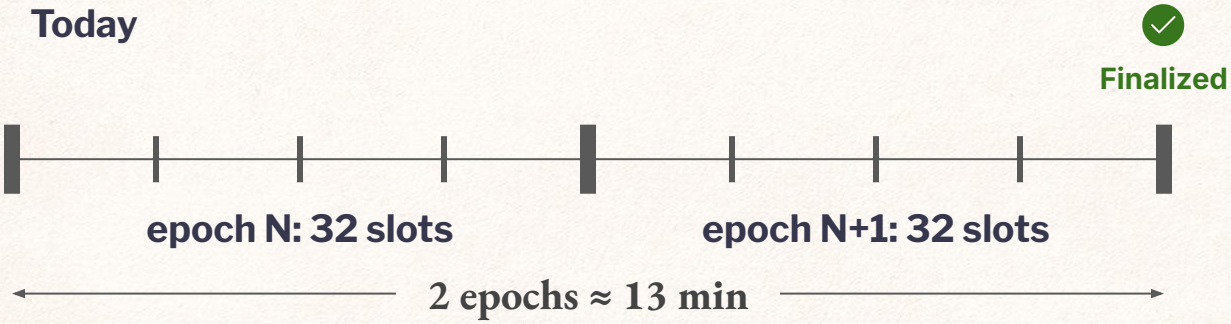
Head Vote
To progress the chain

Finality Vote
To finalize the chain

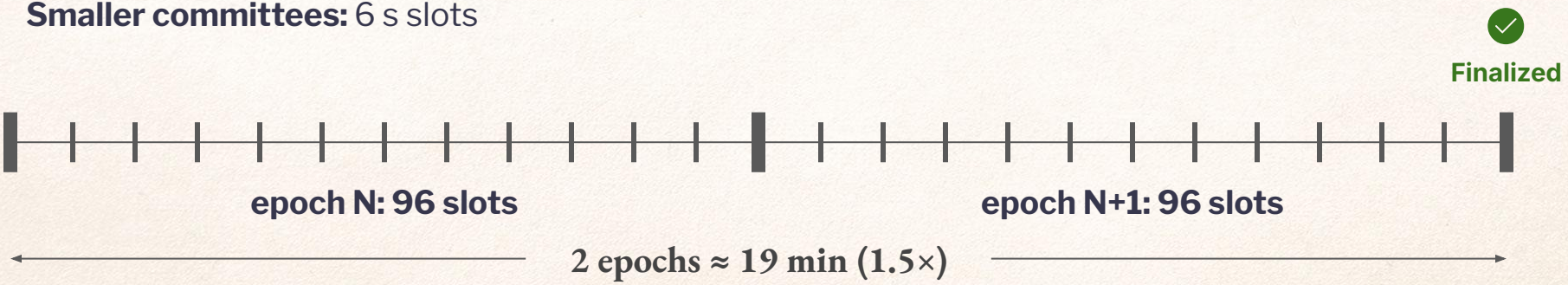
Coupled timelines in Gasper

Small committees
→ long epochs
→ slow finality

Today



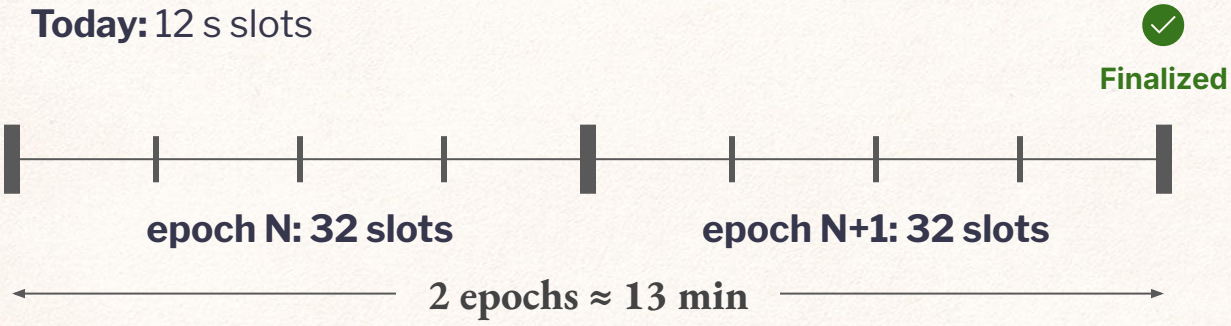
Smaller committees: 6 s slots



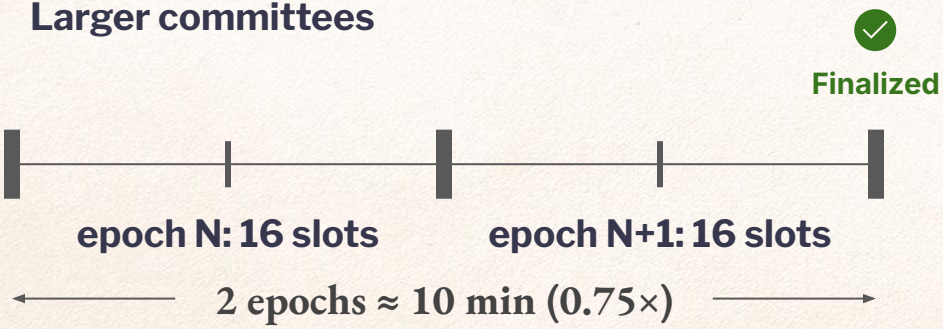
Coupled timelines in Gasper

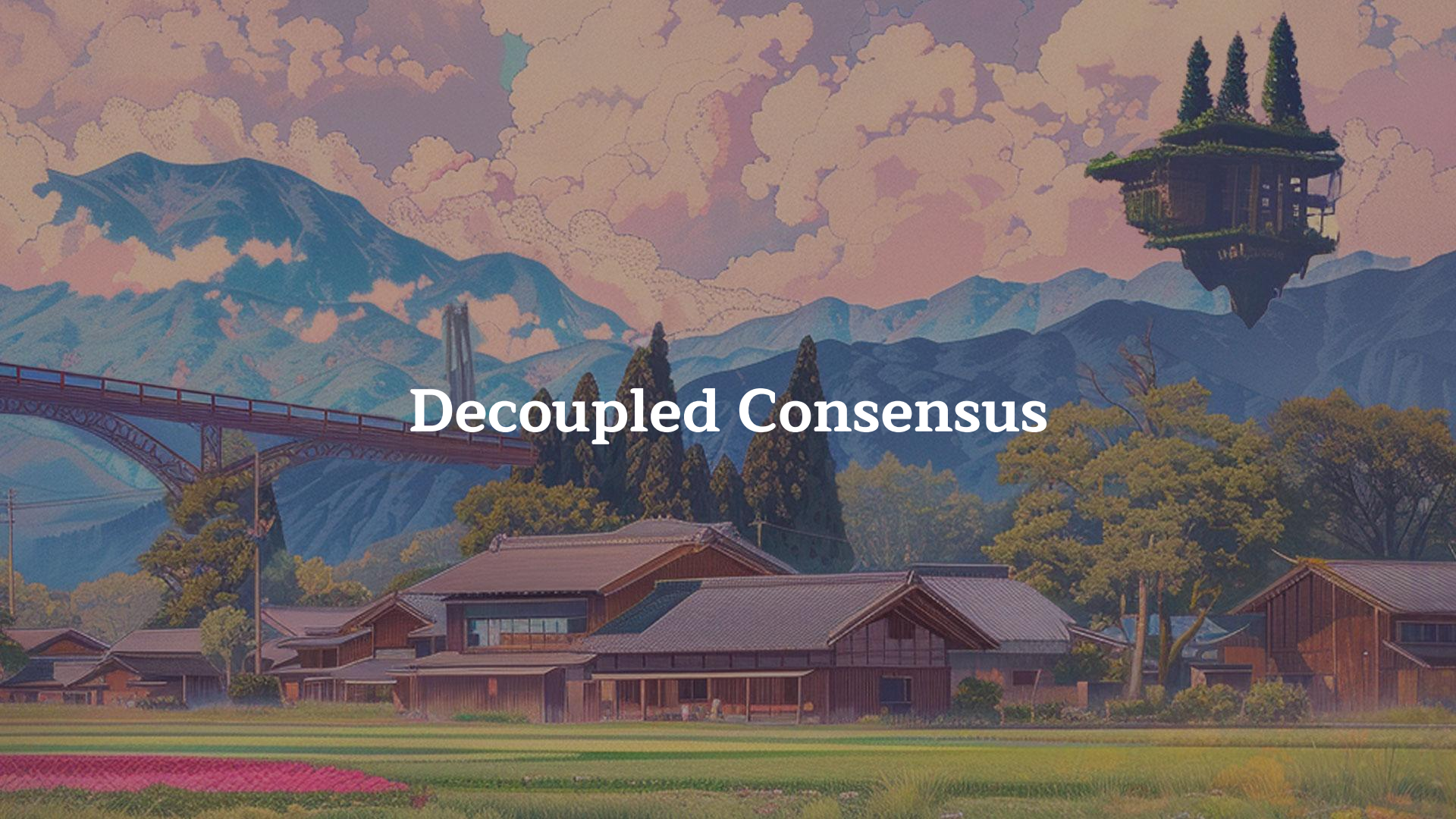
Large committees
→ long slots
→ slow block production

Today: 12 s slots

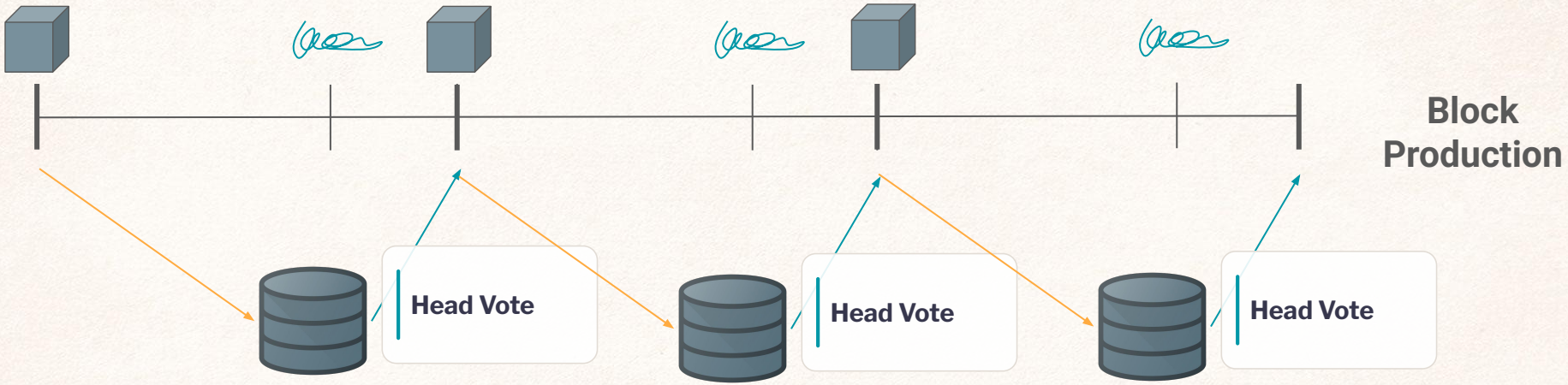


Larger committees

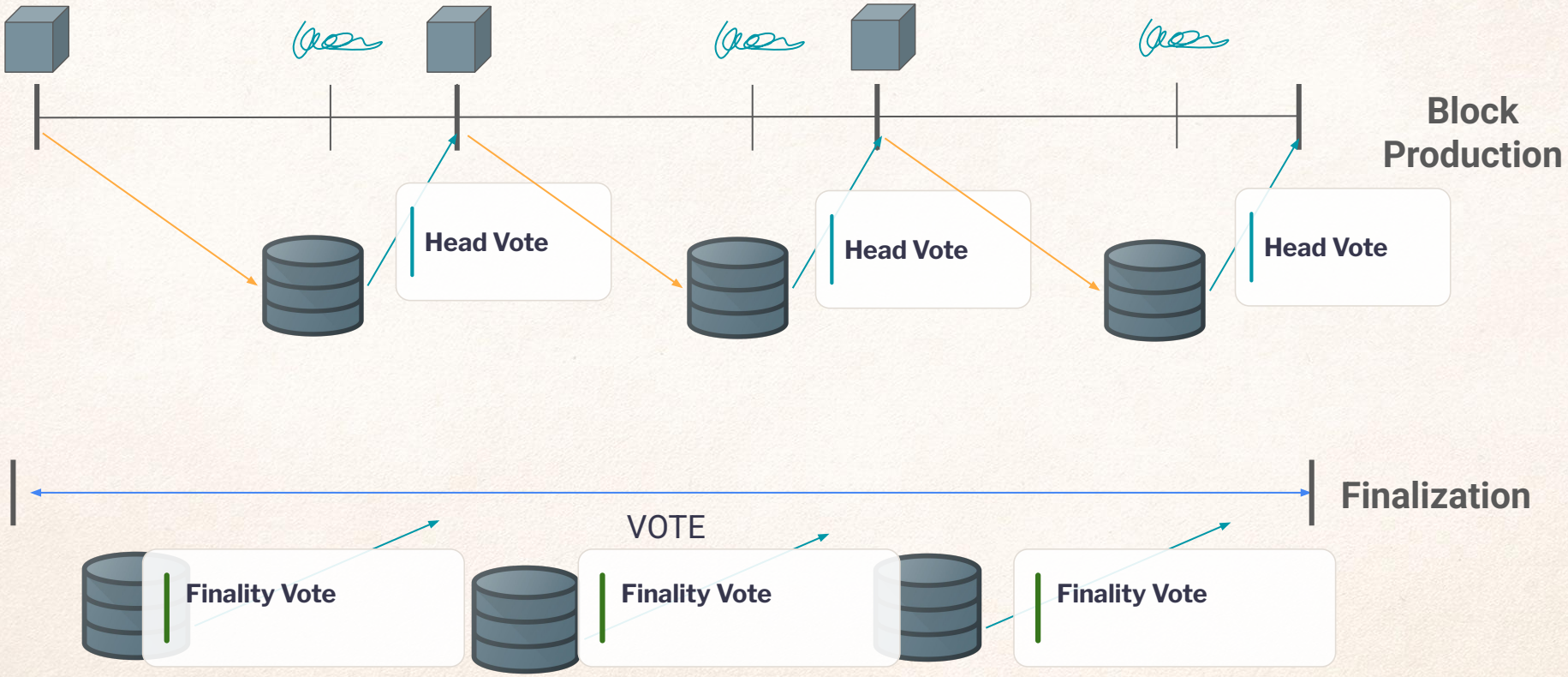



Decoupled Consensus

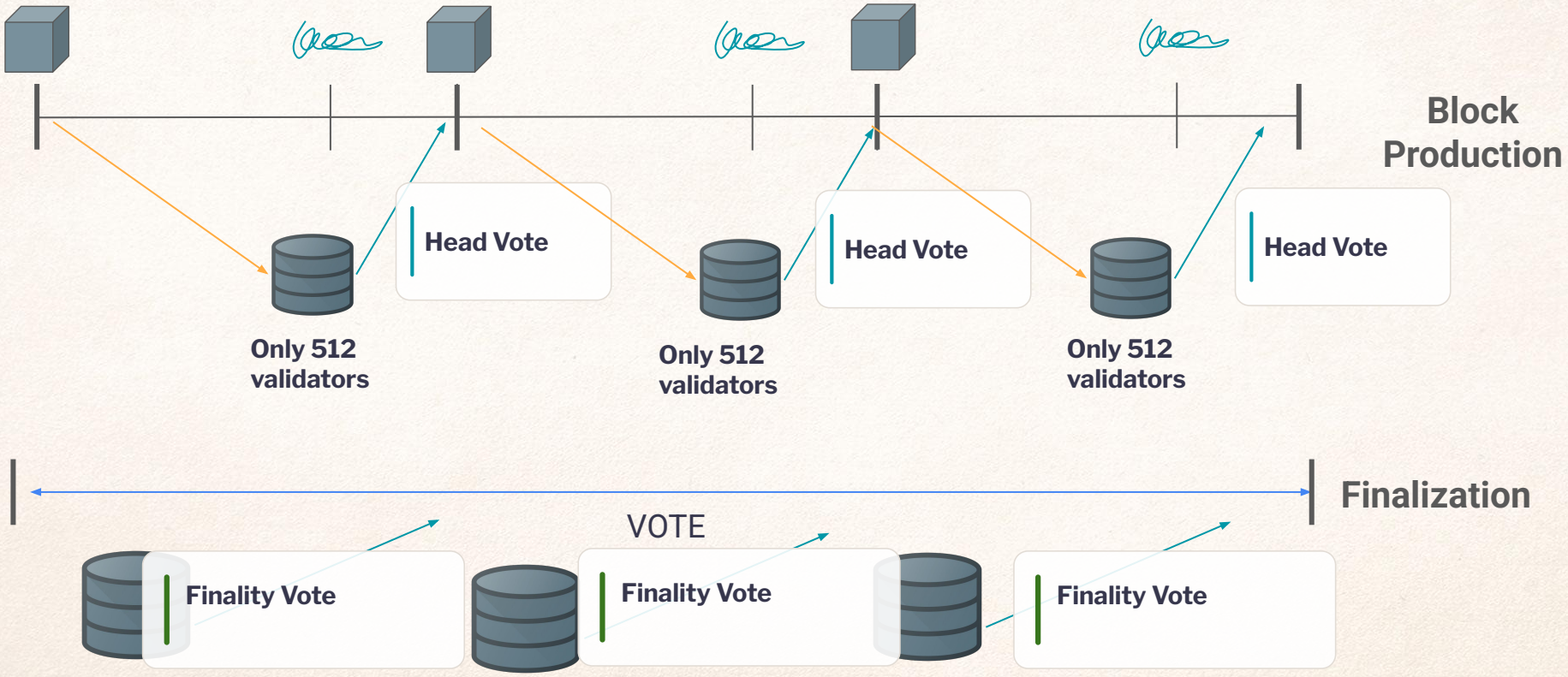
Decoupled Consensus



Decoupled Consensus

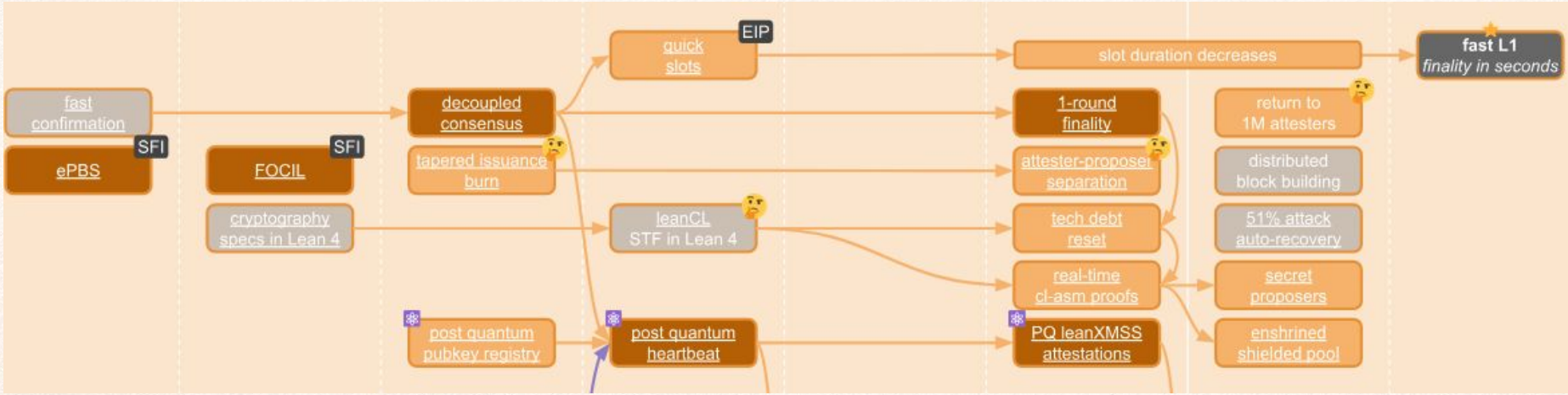


Decoupled Consensus




What we gain

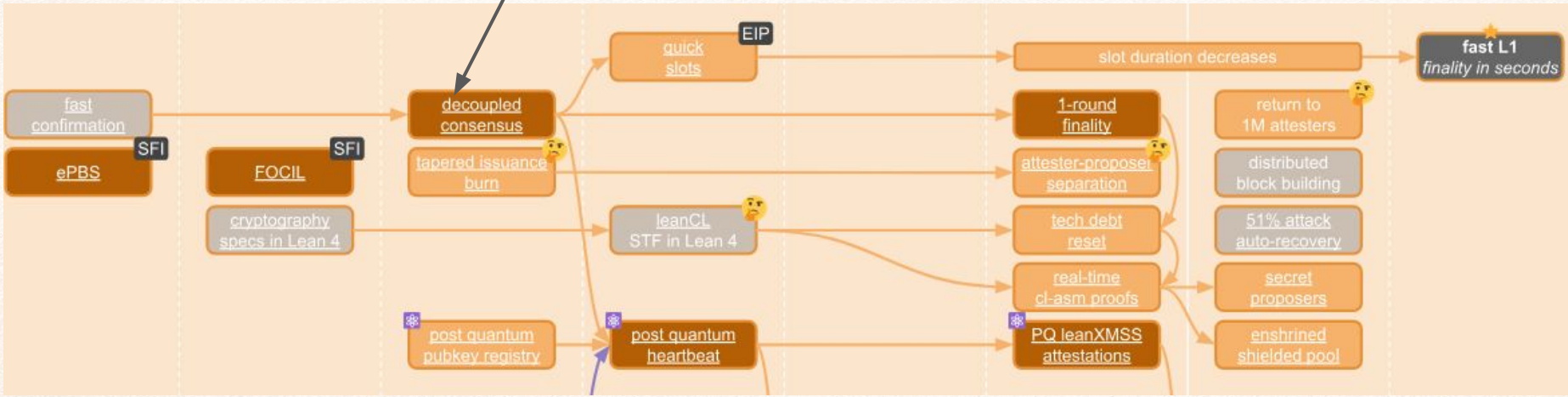
What we gain


fast confirmation
ePBS
SFI
FOCIL
SFI
cryptography specs in Lean 4
decoupled consensus
tapered issuance burn
quick slots
EIP
leanCL STF in Lean 4
post quantum pubkey registry
post quantum heartbeat
slot duration decreases
1-round finality
attester-proposer separation
tech debt reset
real-time cl-asm proofs
PQ leanXMSS attestations
return to 1M attesters
distributed block building
51% attack auto-recovery
secret proposers
enshrined shielded pool
fast L1
finality in seconds

What we gain

4x Faster Finality

Reduces finality time to under 4 minutes, representing a highly conservative speed up estimate.



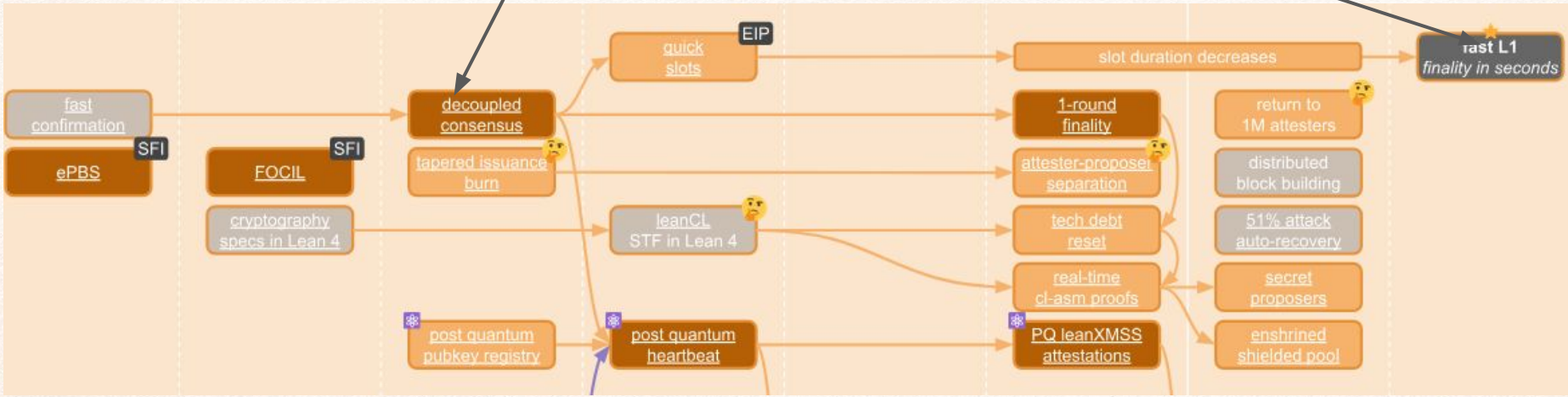
What we gain

4x Faster Finality

Reduces finality time to under 4 minutes, representing a highly conservative speed up estimate.

Unlock < 1min Finality

Unlocks the architectural path to rapid, sub-minute finality operations in seconds.



What we gain

Short Slots

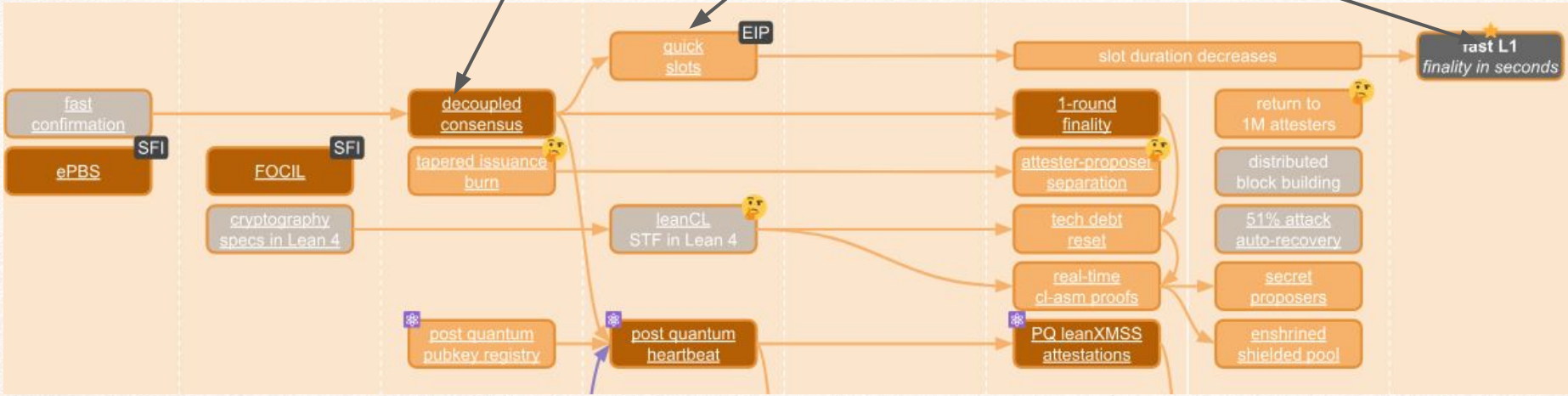
Decreases slot duration to achieve faster transaction processing and lower L1 latency.

4x Faster Finality

Reduces finality time to under 4 minutes, representing a highly conservative speed up estimate.

Unlock < 1min Finality

Unlocks the architectural path to rapid, sub-minute finality operations in seconds.



What we gain

Short Slots

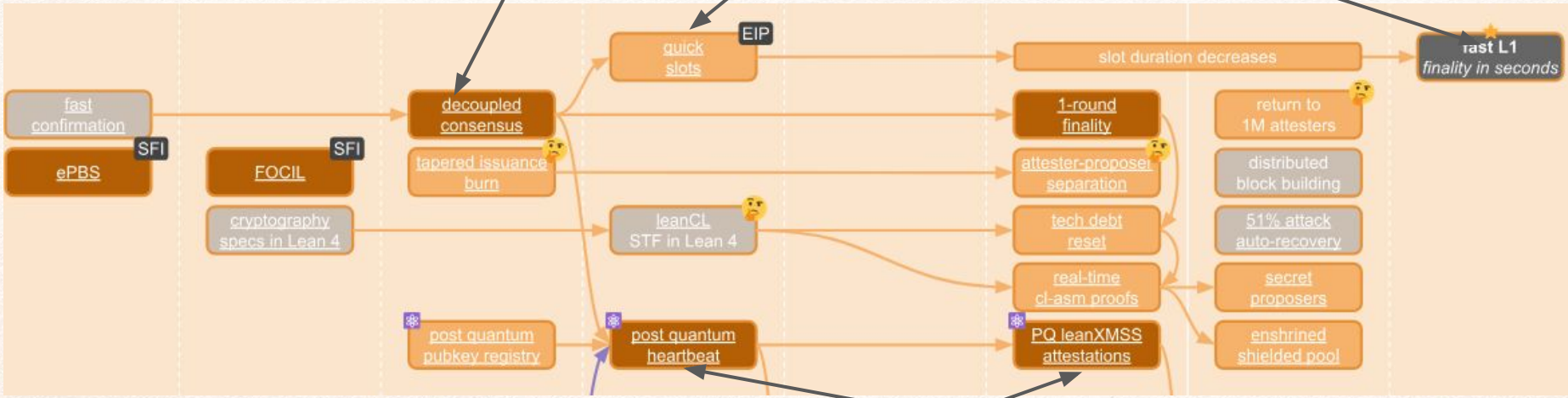
Decreases slot duration to achieve faster transaction processing and lower L1 latency.

4x Faster Finality

Reduces finality time to under 4 minutes, representing a highly conservative speed up estimate.

Unlock < 1min Finality

Unlocks the architectural path to rapid, sub-minute finality operations in seconds.

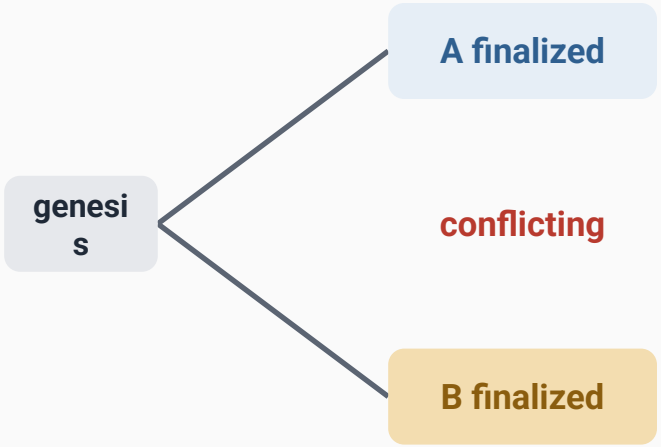


PQ unlocking

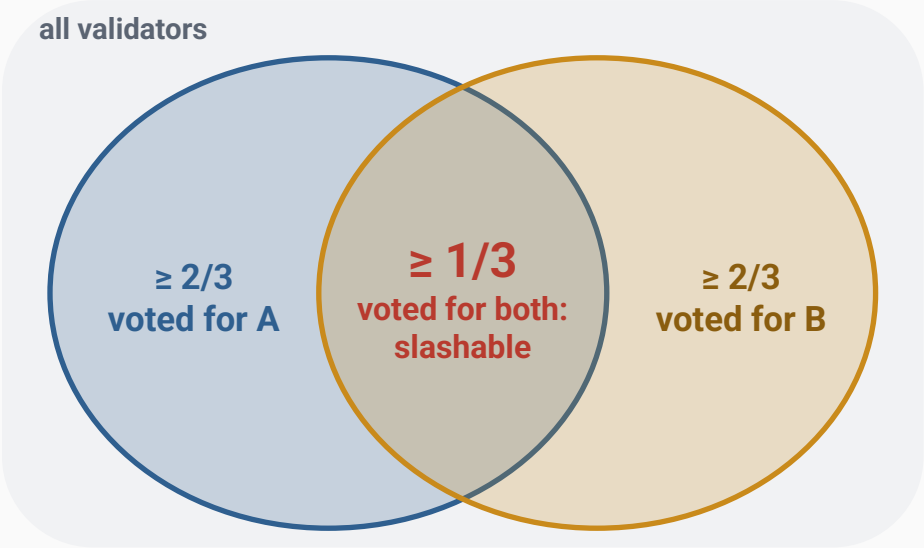
Reduce performance impact of transition to PQ crypto.


Properties we want

Accountable safety



Two conflicting blocks are finalized



The quorums must overlap: 2/3 + 2/3 − 1 = 1/3

Accountable safety: if two conflicting blocks are finalized, the signed votes alone identify at least 1/3 of validators who broke a voting rule, so they can be slashed.

Honest validators are never slashed

Honest validator

Signs one vote per slot and always follows the voting rules.



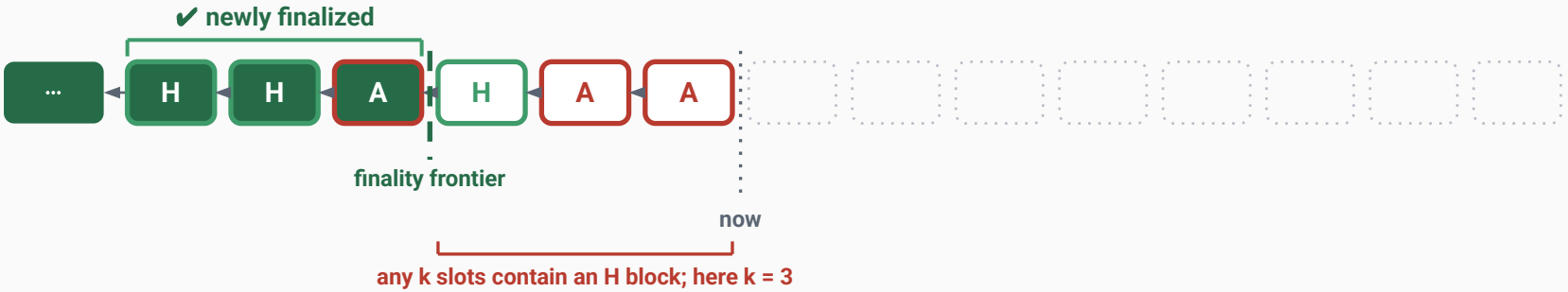
no two of its votes conflict

✔ never slashed

Slashing needs proof: two conflicting votes signed by the same validator. An honest validator never signs such a pair, so it is never slashed, whatever the network or the adversary does. The ≥ 1/3 exposed by accountable safety are all faulty.

Finality Liveness

everyone is awake + network is synchronous + honest proposer at least every k slots

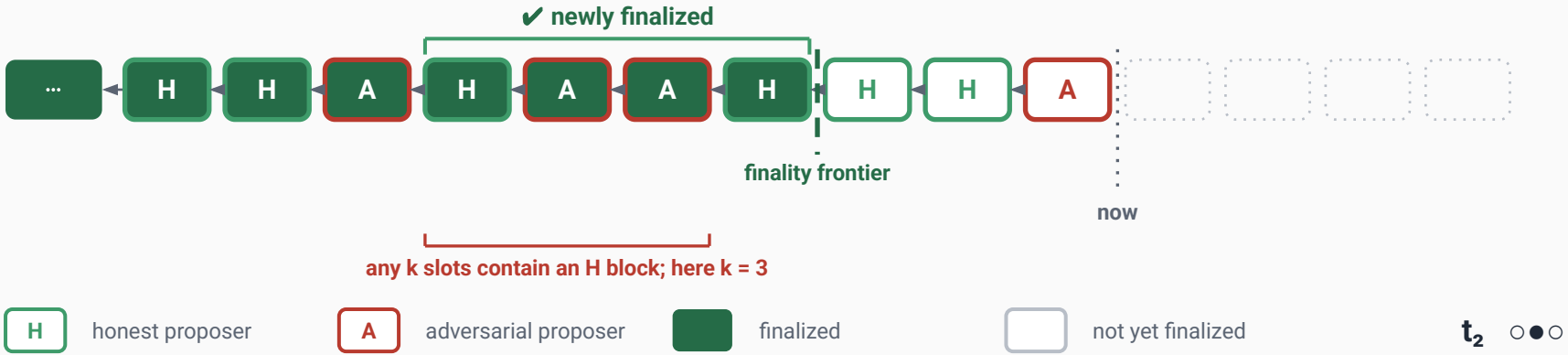


H honest proposer A adversarial proposer finalized not yet finalized t_1

At t_1 the blocks up to the frontier are finalized. Honest and adversarial proposers both add blocks to the chain.

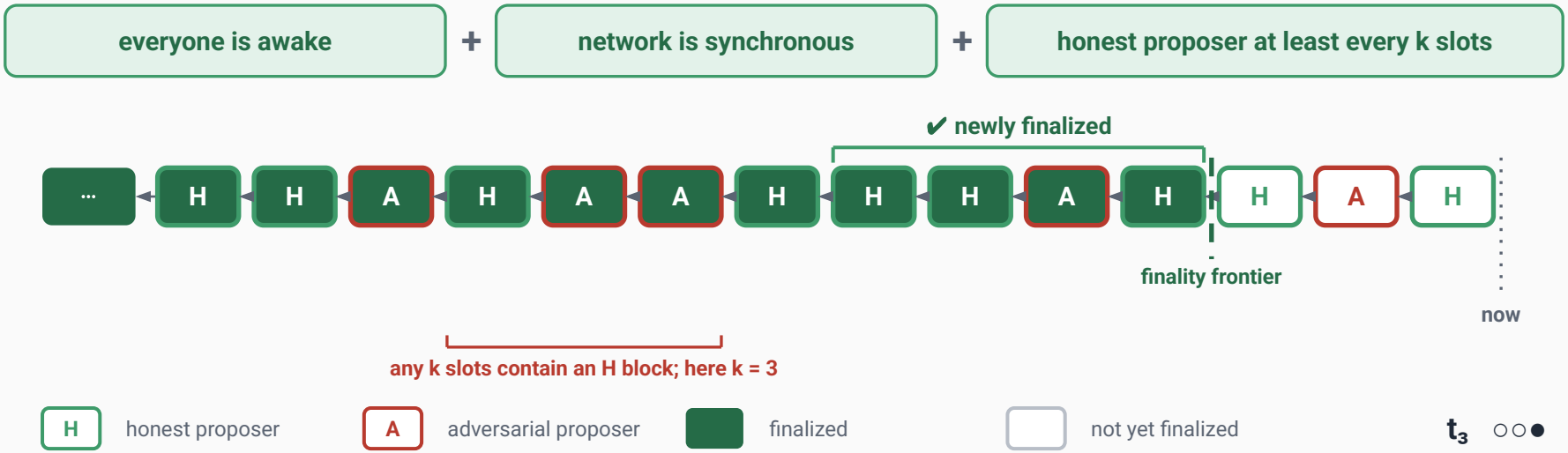
Finality Liveness

everyone is awake + network is synchronous + honest proposer at least every k slots



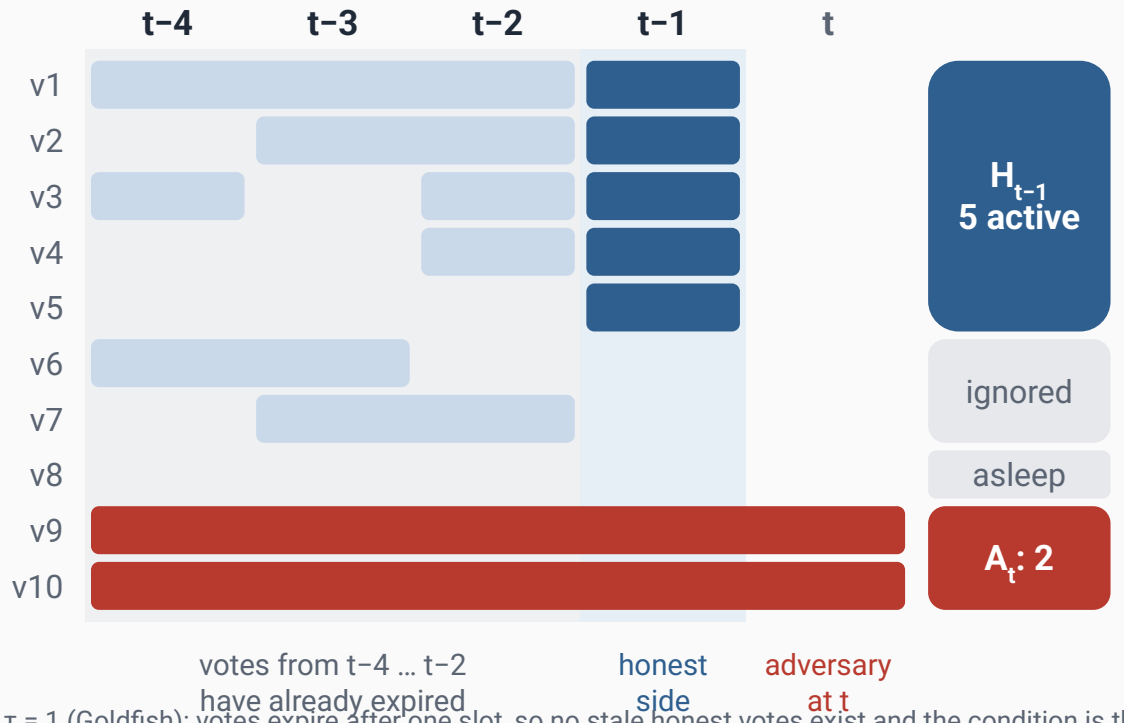
At t_2 the frontier has moved: more blocks are finalized, adversarial ones included.

Finality Liveness



At t_3 it moves again. With everyone awake, a synchronous network and an honest proposer in every k slots, new blocks keep getting finalized.

Dynamic Availability : Sleepiness



Condition at slot t (τ = 1)

$$|H_{t-1}| > |A_t|$$

5
honest, active in t−1

>

2
adversarial only

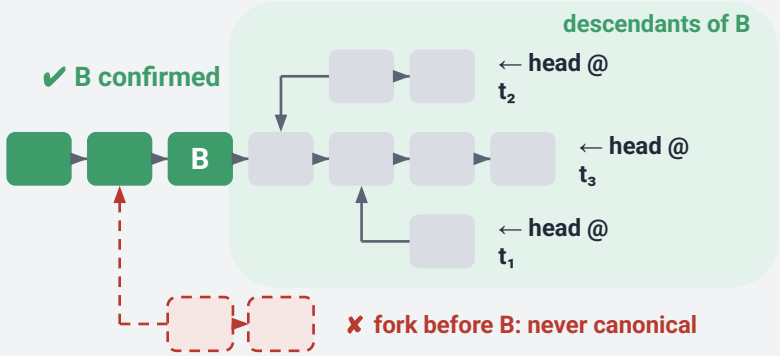
5 > 2: sleepiness holds

τ = 1 (Goldfish): votes expire after one slot, so no stale honest votes exist and the condition is the original sleepy model. Source: D'Amato & Zanolini, RLMD-GHOST (2023).

Dynamic Availability : Main Properties

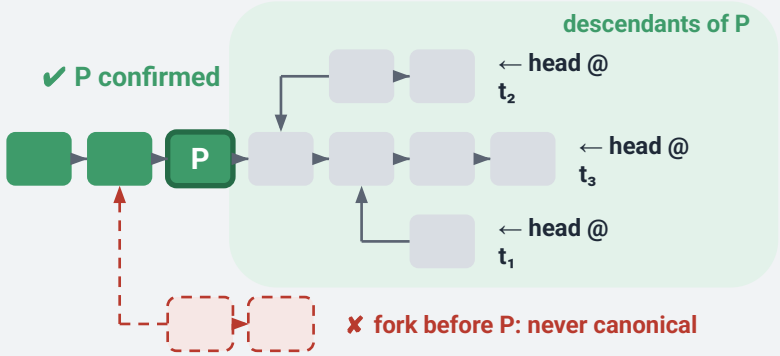
Safety of confirmations

A confirmed block is always part of the canonical chain.



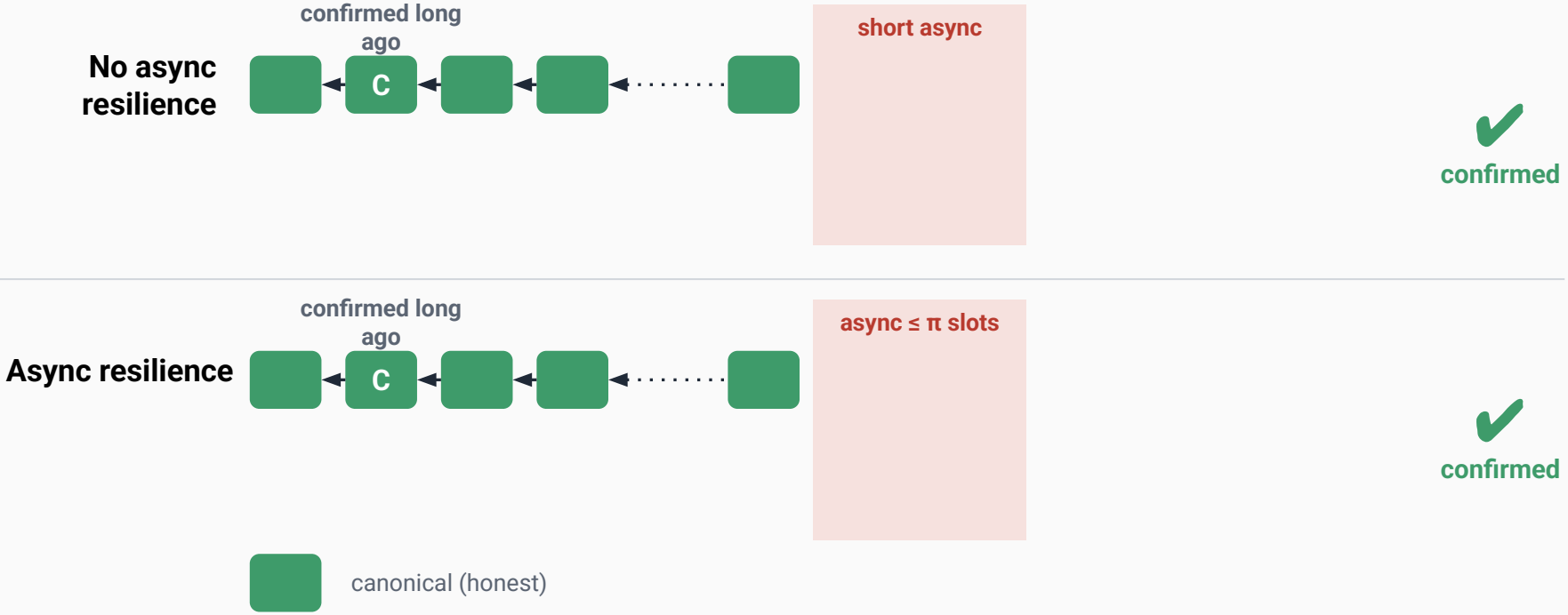
Safety of honest proposals

A block proposed by an honest validator gets confirmed, and is therefore safe.



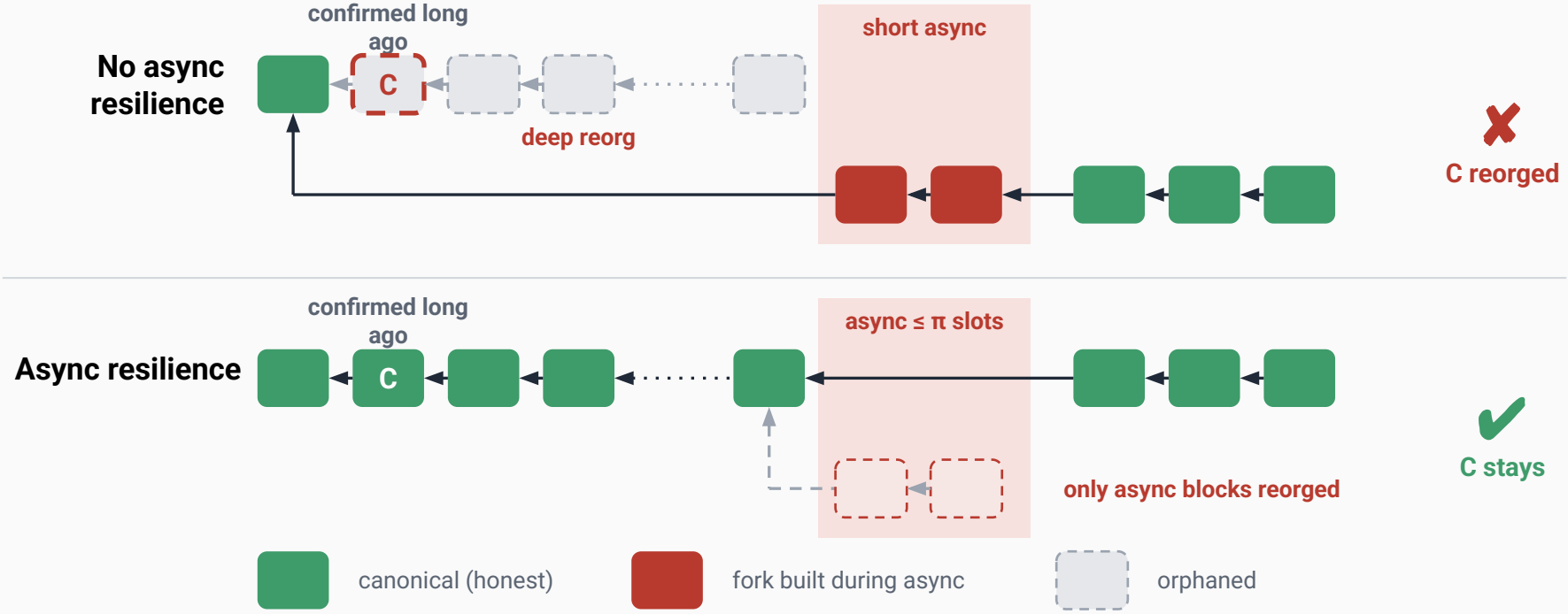
The head may move between forks, but every fork it can follow extends B (or P). Both should hold whenever the τ-sleepiness condition holds.

Dynamic Availability: Resilience to Bounded Asynchrony



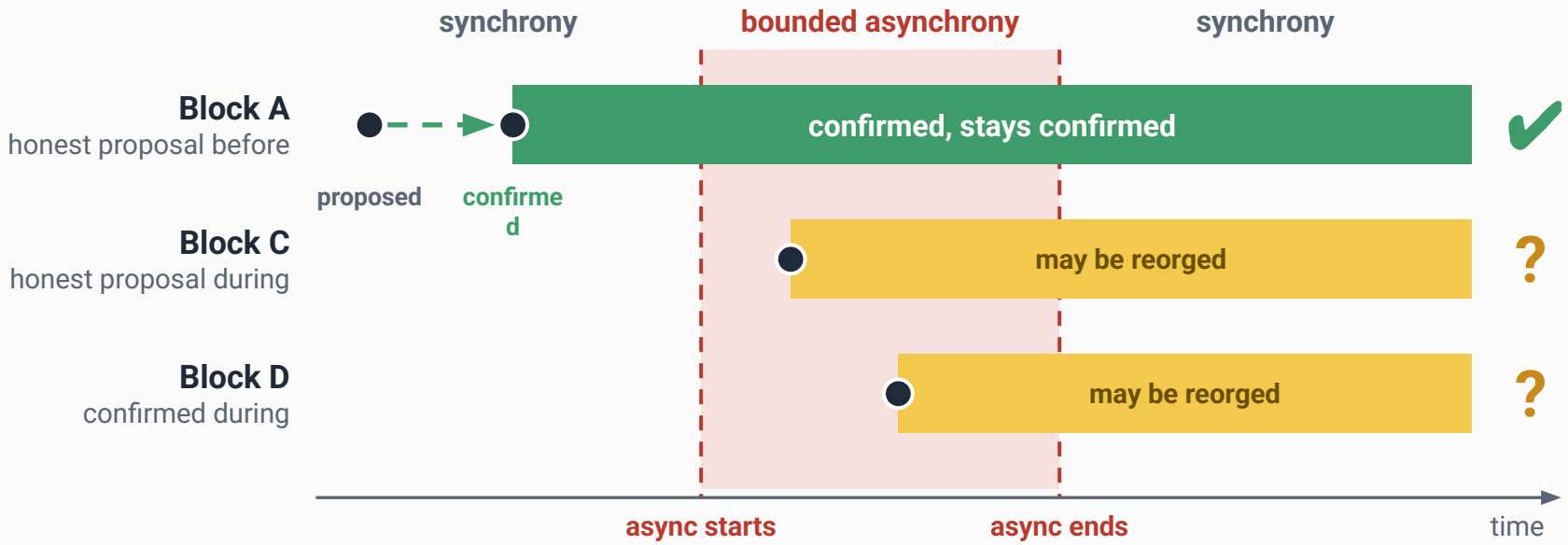
Before the asynchronous period, both chains look the same: every block up to the red band is confirmed, including C, which was confirmed long ago and is now deep in the chain.

Dynamic Availability: Resilience to Bounded Asynchrony



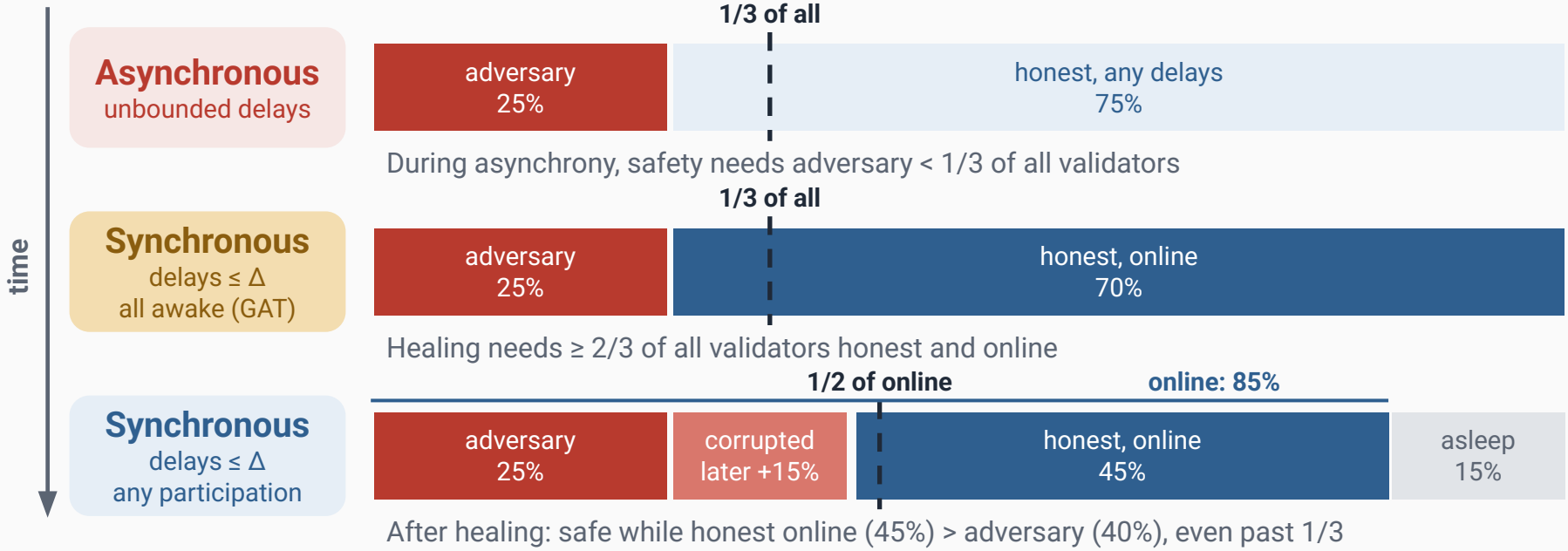
The async fork branches off before C and wins, a deep reorg of confirmed blocks. RLMD-GHOST: everything confirmed before async stays; the async blocks can only extend it, and at most they get reorged.

Dynamic Availability: Resilience to Bounded Asynchrony - Limitations



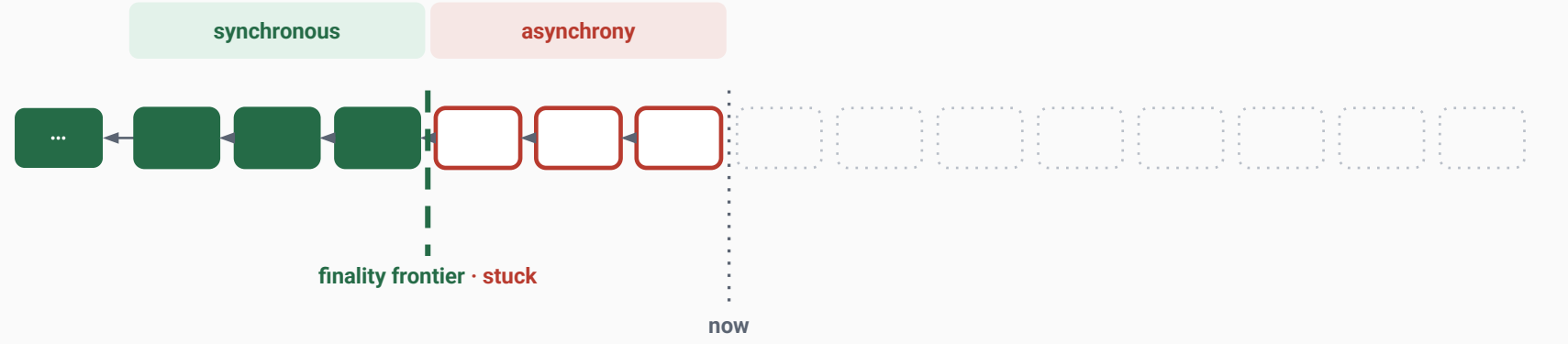
Bounded asynchrony: an honest proposal is confirmed, and any block confirmed before the asynchronous period stays confirmed through it and after. Blocks proposed or confirmed during it may be reorged.

Healing from Asynchrony



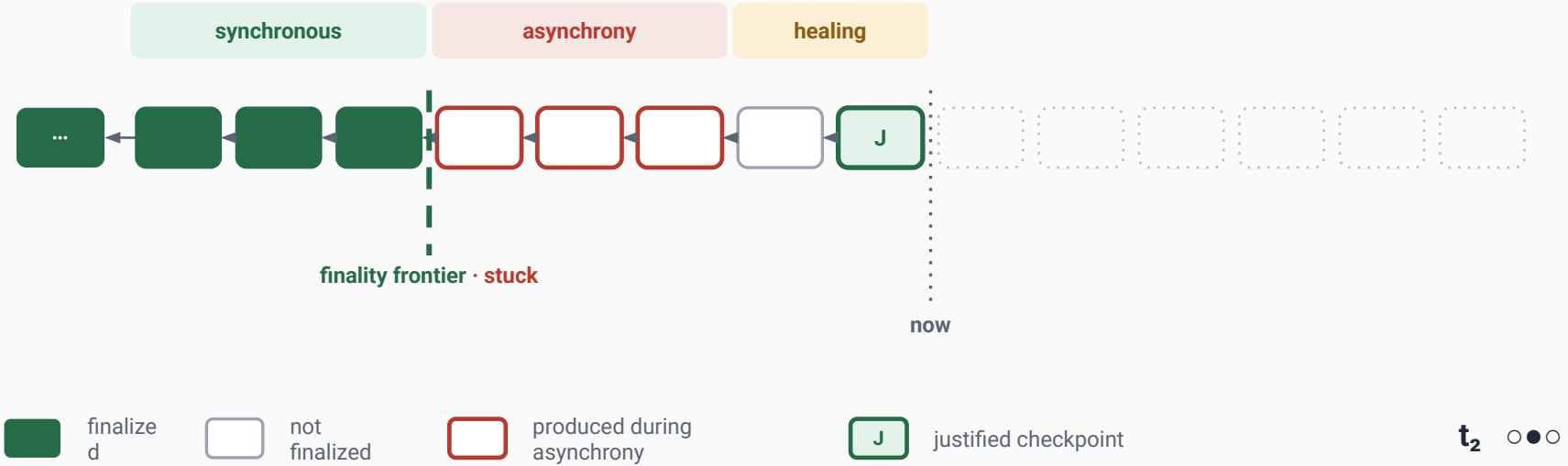
Percentages are illustrative. Corruption is permanent, so the adversary never shrinks, but after healing it may grow. Under τ-sleepiness, honest validators asleep at t−1 whose votes have not expired also count on the adversary's side.

Healing from Asynchrony: Finality Resumes



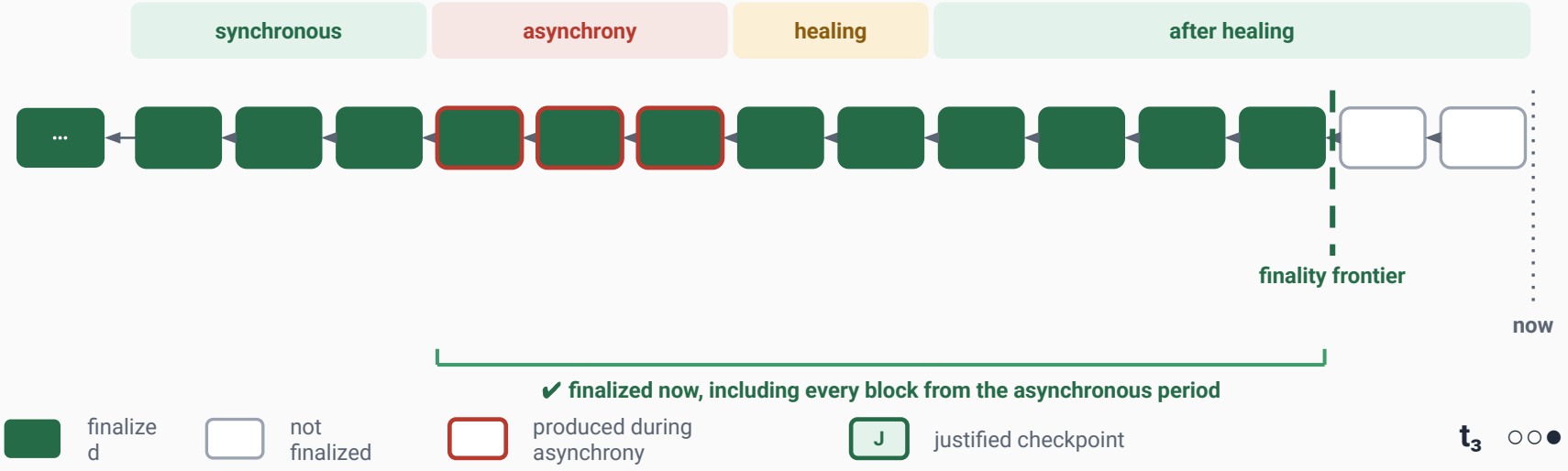
Asynchrony: blocks keep coming, but votes are delayed and no target reaches 2/3. The finality frontier is stuck.

Healing from Asynchrony: Finality Resumes



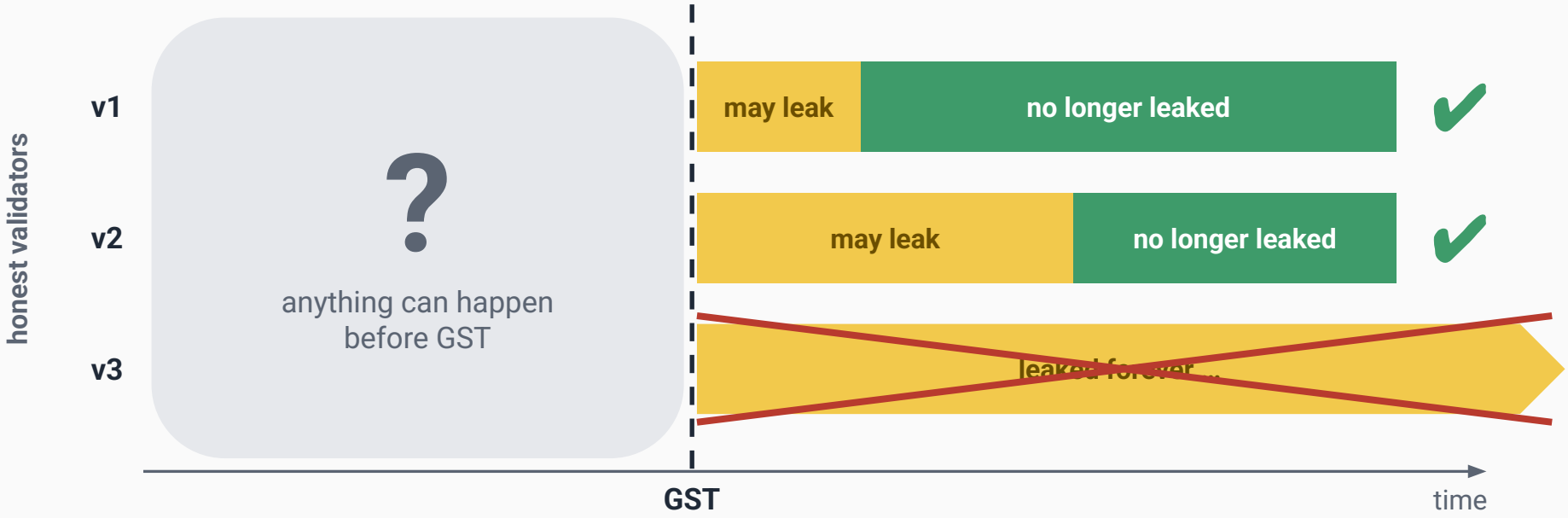
Healing: from max(GST, GAT), with ≥ 2/3 of validators honest and online, the first checkpoint after healing is justified.

Healing from Asynchrony: Finality Resumes



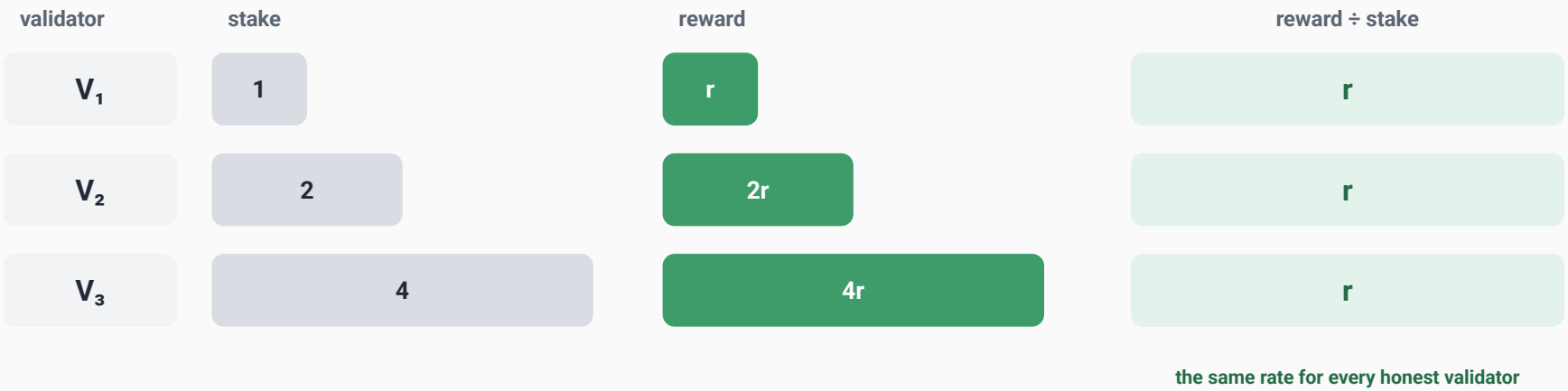
Finality resumes: the frontier jumps forward and the blocks from the asynchronous period are finalized with it. It keeps advancing while ≥ 2/3 are honest and online.

Leak Fairness



Leak fairness: no matter what happened before GST, every honest validator stops being leaked within finite time after GST.

Reward Fairness



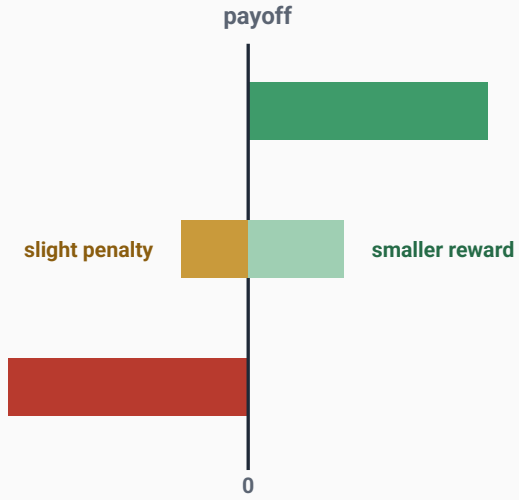
Same rate, any size
A small validator earns the same return on its stake as a large one.

Splitting or pooling changes nothing
Four validators with stake 1 earn 4r together, exactly what one validator with stake 4 earns.

Rewards come from timely, correct votes and from proposals. Every vote is weighted by stake, and proposers are chosen in proportion to stake, so proposal rewards are proportional in expectation.

Incentive compatibility

behaviour		outcome
Follow the protocol vote on time for the right head and target; propose when selected	→	✔ full reward, proportional to stake
Stay offline or vote late miss some votes or proposals	→	✘ smaller reward, or a slight penalty
Contradict your own votes two conflicting votes at one height	→	✘ slashed: a large share of stake is lost



Pay for useful work
Rewards go to timely, correct votes and proposals, the actions that move the chain forward.

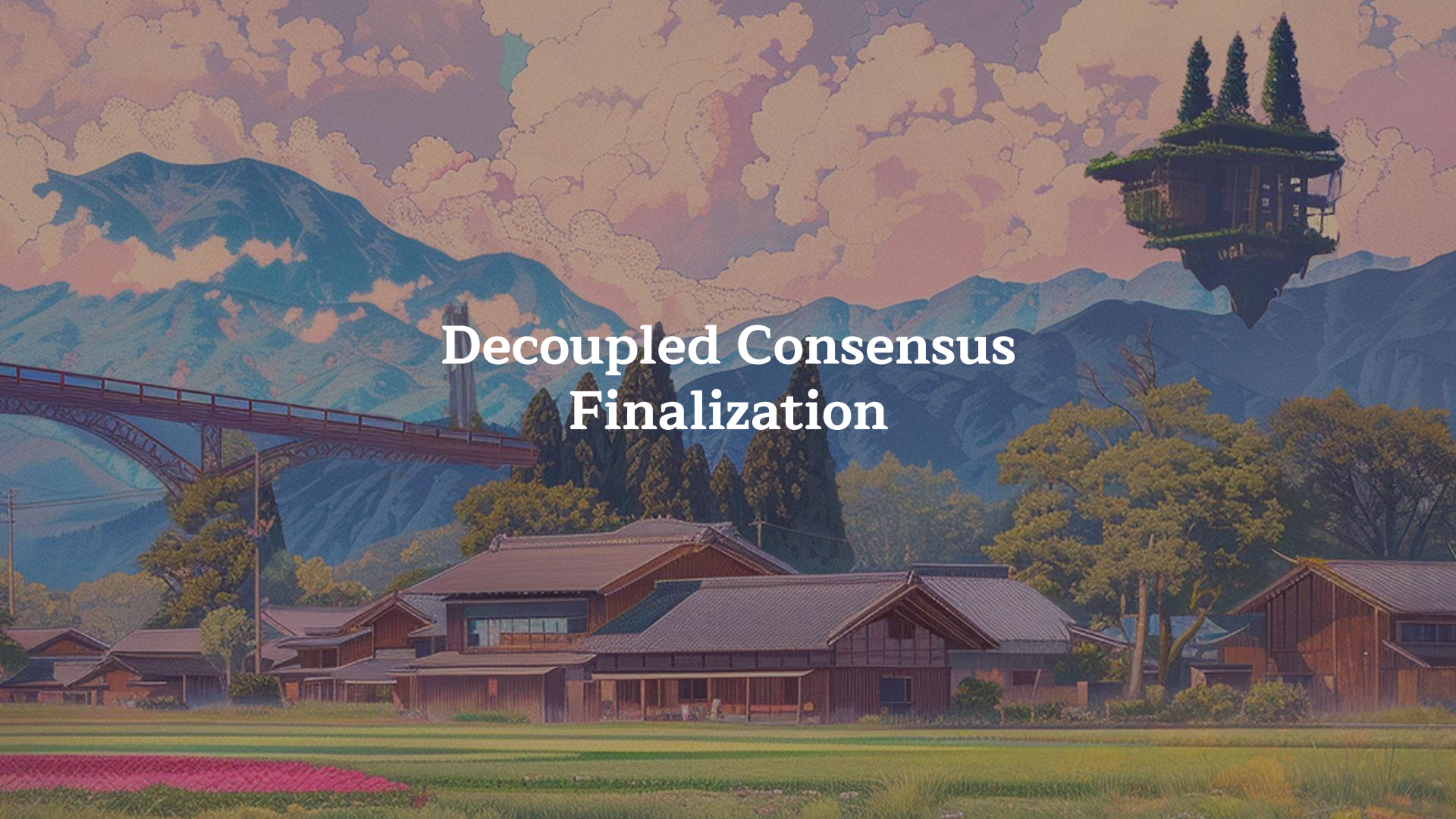
Make faults expensive
Missing duties costs rewards; contradicting yourself costs stake.

No gain from deviating
Withholding, delaying or equivocating should never earn more than following the protocol.

The goal: for a rational validator, following the protocol pays at least as much as any deviation, so honest behaviour is also the self-interested one.

Summary of Properties

Finality	**Accountable safety** If two conflicting blocks are finalized, at least 1/3 of the stake is provably at fault and can be slashed.	**Honest validators are never slashed** Slashing needs two conflicting votes signed by the same validator; an honest one never signs such a pair.	**Finality liveness** Everyone awake, a synchronous network and an honest proposer every k slots: new blocks keep getting finalized.
Dynamic availability	**Safety of confirmations** Under τ-sleepiness, a confirmed block always stays in the canonical chain.	**Safety of honest proposals** Under τ-sleepiness, a block proposed by an honest validator gets confirmed.	
Asynchrony	**Resilience to bounded asynchrony** Asynchrony lasting at most π slots cannot reorg blocks confirmed before it.	**Healing from asynchrony** From max(GST, GAT), with ≥ 2/3 honest and online, one justification restores safety with an online honest majority, and finality resumes.	
Incentives	**Leak fairness** Whatever happened before GST, every honest validator stops being leaked within finite time.	**Reward fairness** Honest validators earn the same reward per unit of stake, so rewards are proportional to stake.	**Incentive compatibility** Following the protocol should pay at least as much as any deviation.


Decoupled Consensus
Finalization

For finality, each vote carries two components: a height vote and a finality vote

Height vote: one of

justify (h, T)

justify T, the target of height h

timeout (h, T)

move past h without justifying T

none

nothing about heights

Finality vote: one of

(h_j, J)

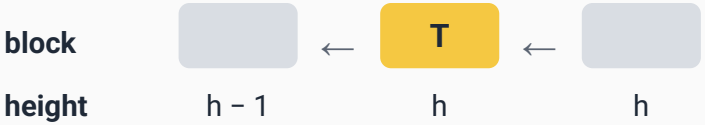
finalize J, justified at height h_j

none

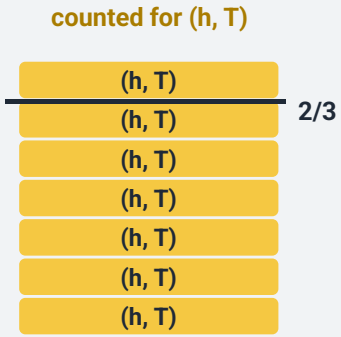
no finality vote this round

Each validator sends one attestation per round, carrying both votes. A vote is counted on a chain once a block of that chain includes the attestation.

A block is justified when 2/3 of the stake votes for it as the target of its height

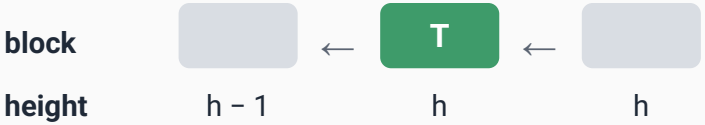


✔ **T is justified**

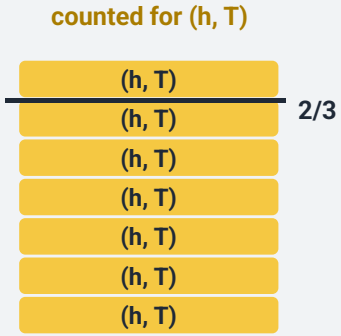


Timeout votes, and votes for another block at height h, never justify T. They can still let the chain move past h, as the slide on height progression shows.

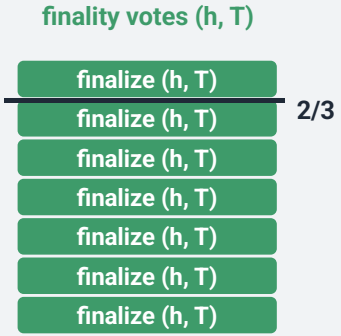
A justified block is finalized when 2/3 of the stake votes to finalize it



✔ T is justified

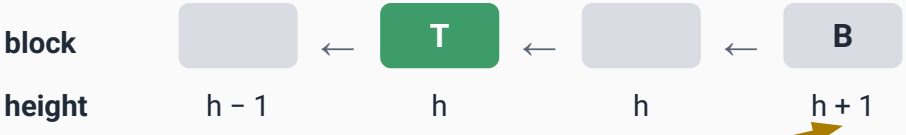


✔ T is finalized



Once T is justified at height h, attestations carry finality votes for (h, T). When the finality votes of 2/3 of the stake are included in the chain, T is finalized.

The block whose votes justify T moves the chain to height h + 1



votes included in B

✔ T is justified

counted for (h, T)

(h, T)
(h, T)
(h, T)
(h, T)
(h, T)
(h, T)
(h, T)

2/3

✔ T is finalized

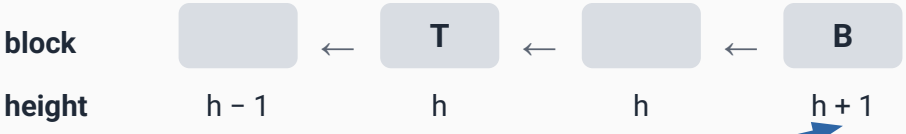
finality votes (h, T)

finalize (h, T)
finalize (h, T)
finalize (h, T)
finalize (h, T)
finalize (h, T)
finalize (h, T)
finalize (h, T)

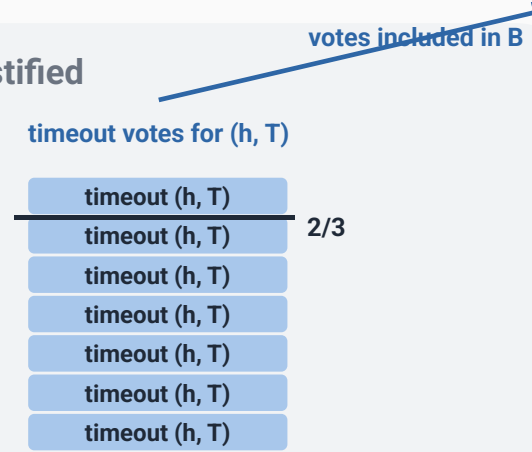
2/3

Votes count once a block of the chain includes them, and that block need not be T's child. When the votes included in B bring (h, T) to 2/3 of the stake, T is justified and B becomes the first block at height h + 1.

Timeout votes move the chain to height h + 1 without justifying T



✘ T is not justified



✔ B is the first block at h + 1

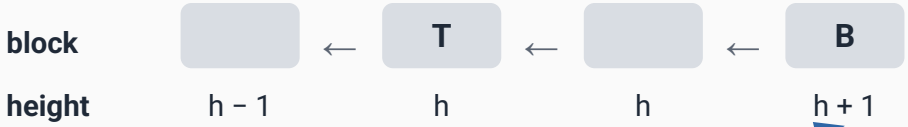
In B's chain state:

- target and timeout votes for (h, T) weigh 2/3 of the stake
- B's slot is at least T's slot plus the timeout delay

So B becomes the first block at height h + 1, and T stays unjustified.

A timeout vote for (h, T) asks to move past height h without justifying T. Timeout votes count toward moving the height only, and only once the timeout delay has passed since T's slot. The justified block does not change.

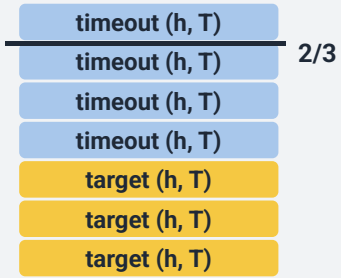
A mix of target and timeout votes also moves the chain to height h + 1



votes included in B

✘ T is not justified

target and timeout votes for (h, T)



✔ B is the first block at h + 1

In B's chain state:

- target and timeout votes for (h, T) weigh 2/3 of the stake
- B's slot is at least T's slot plus the timeout delay

So B becomes the first block at height h + 1, and T stays unjustified.

Target votes for (h, T) count toward moving the height too. Alone they weigh less than 2/3 of the stake, so T is not justified. With the timeout votes they reach 2/3, and after the timeout delay B becomes the first block at height h + 1.

Slashing rules

✘ **Slashable**

Two different targets at one height

justify (h, A)

+

justify (h, B)

✘ **Slashable**

Voting to finalize a target, and voting otherwise at that height

finalize (h A)

+

justify (h, B)
or
timeout(h, *)

✔ **Not slashable**

A target vote and a timeout vote at one height, with no vote to finalize there

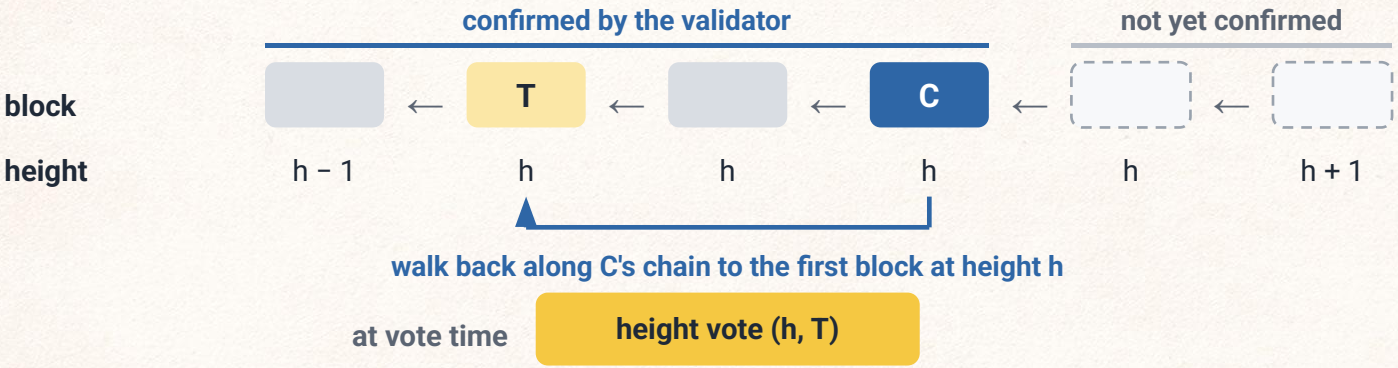
justify (h, A)

+

timeout (h, *)

The two votes can be in one attestation or in two. The rules on the previous slide are there so that an honest validator does neither.

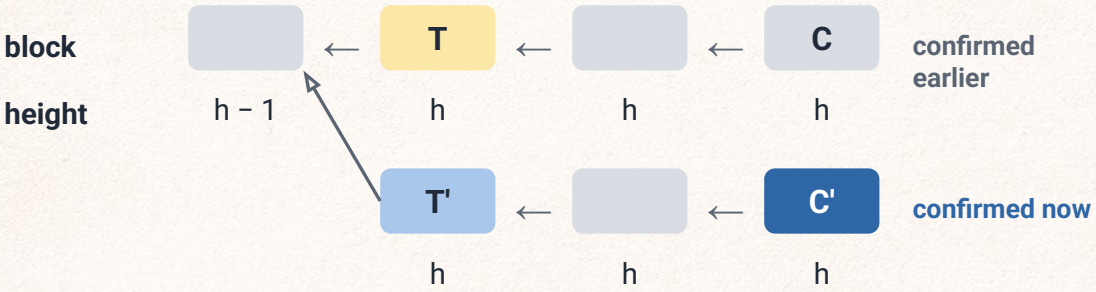
At vote time, a validator's height vote follows its latest confirmed block on the Goldfish chain



At vote time, the validator
1. takes C, the latest block it has confirmed on the Goldfish chain;
2. reads C's chain state: its height h and its target T, the first block at height h;
3. votes (h, T). The next slides say when it sends a timeout vote (h, T) instead.

Blocks the validator has not confirmed yet do not change its height vote. Here the tip of the chain is already at height h + 1, but the vote is for height h.

If the validator then confirms a fork that branches off before T, it sends a timeout vote



The validator's height votes at h

before	target (h, T)
now	timeout (h, C')

a target vote (h, T') would be slashable

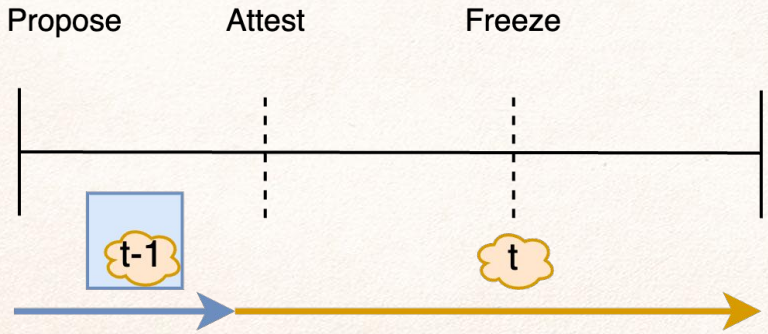
1. Earlier, the validator confirmed C and sent the target vote (h, T).
2. A fork that branches off before T is now confirmed instead: C', whose first block at height h is T'.
3. A target vote (h, T') would be a second target at height h, so it sends the timeout vote (h, T').

The timeout vote names T', so it still counts toward moving the chain of C' past height h. It never counts toward justifying T'.


Goldfish

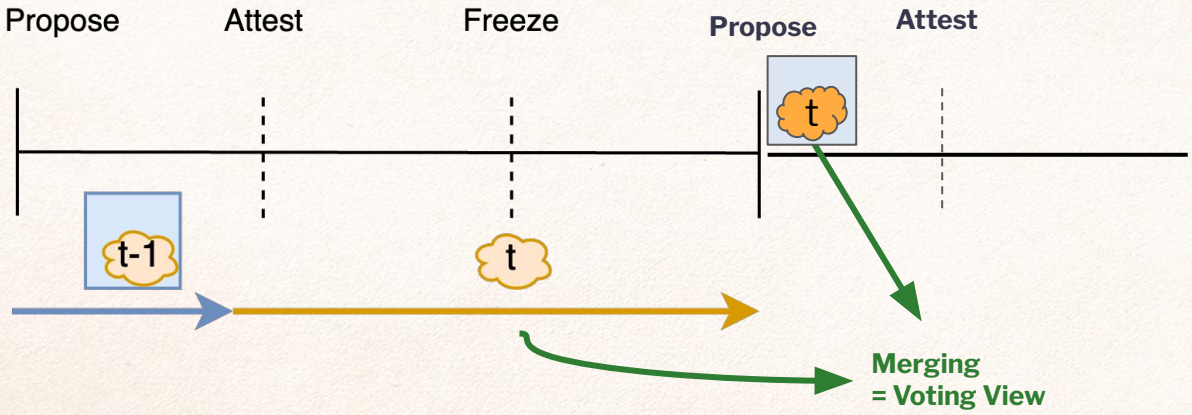
Goldfish

At slot t, only run the fork-choice (GHOST) with votes from slot t-1.
Old votes (previous committees, asleep validators…) have no influence.



Goldfish

At slot t, only run the fork-choice (GHOST) with votes from slot t-1.
Old votes (previous committees, asleep validators…) have no influence.



Goldfish - Dynamic Confirmation

At slot t, only run the fork-choice (GHOST) with votes from slot t-1.
Old votes (previous committees, asleep validators…) have no influence.

