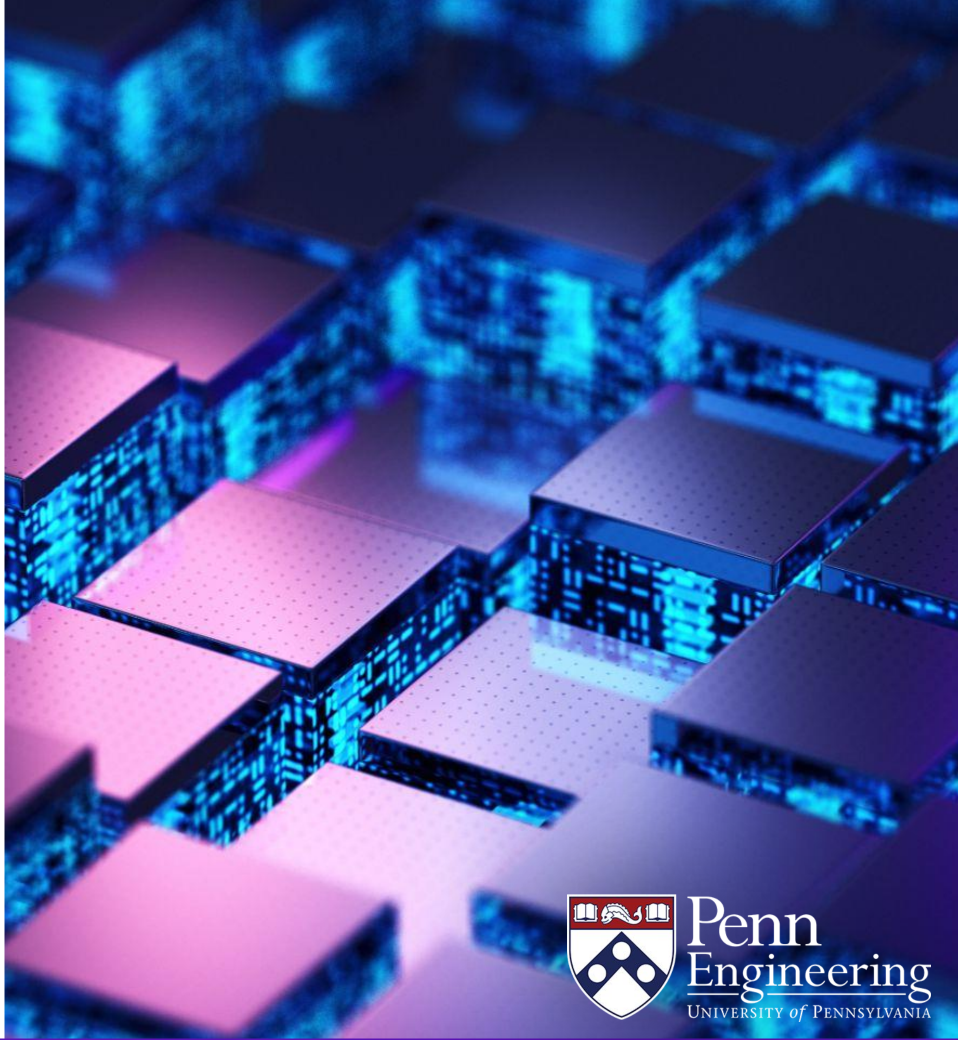
EAS 5830: BLOCKCHAINS

Ethereum's Consensus Mechanism

Dr. Brett Hemenway Falk



PoW Ethereum

- When Ethereum was created in 2013, it used Nakamoto Consensus
  - Similar to Bitcoin but using Ethash instead of SHA-256 for PoW

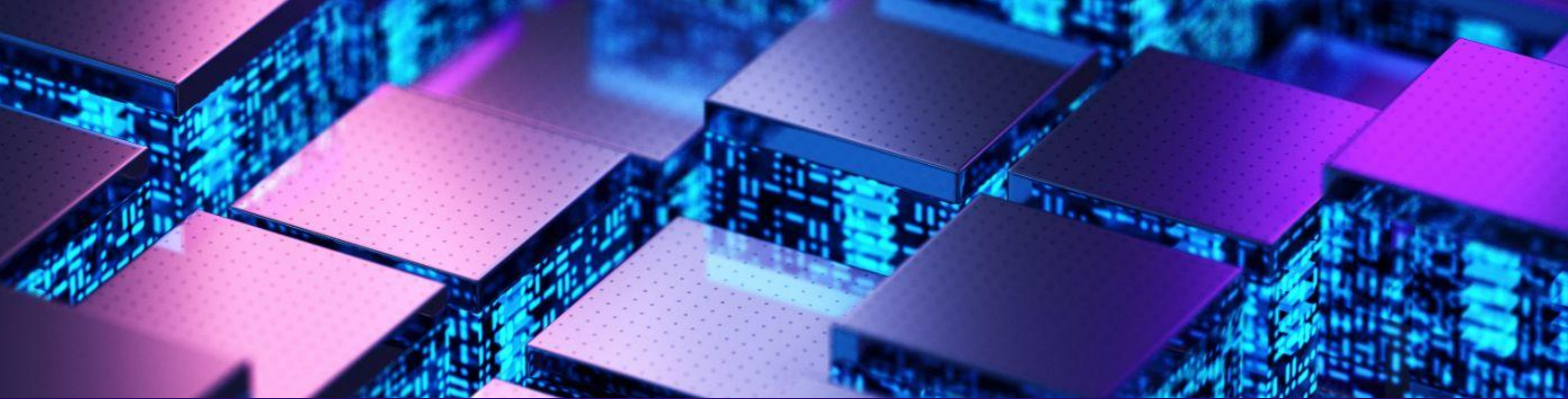
Vitalik always liked Proof of Stake

“there are substantial economic reasons to believe that proof of stake actually is much more economically efficient than proof of work….Now, it’s simply a matter of standardizing the algorithms, and giving blockchain developers the choice.”

-Vitalik Buterin 2014

Ethereum’s move to PoS

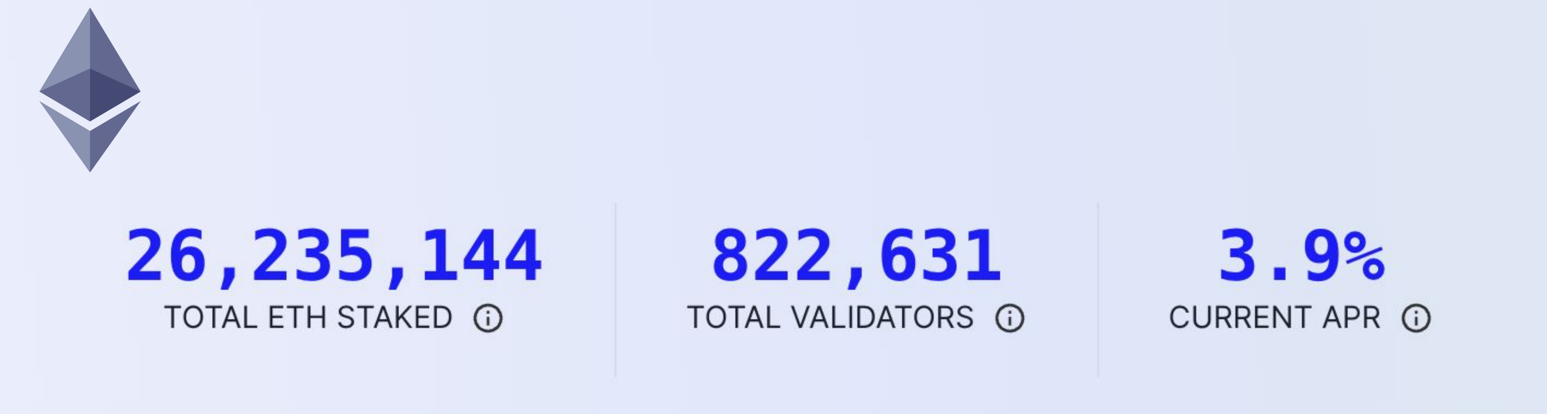
- 2017
  - “The ultimate goal of the Ethereum Foundation for 2017 is to follow the vision of Ethereum founder Vitalik Buterin and make a move from a proof of work to a proof of stake protocol.”
- 2020
  - “While the proof of stake Ethereum date was originally set for January 2020, this deadline has been missed and it isn’t clear when Ethereum’s PoS will launch now. Guesses vary from sometime in 2020 to sometime in 2021 to never (according to hardcore ETH haters)!”
- 2021
  - Phase 0: “Beacon Chain” launched December, 1st 2020
- 2022
  - The “Merge” – Ethereum becomes Proof-of-Stake
- 2023
  - Shapella upgrade – Stakers can un-stake



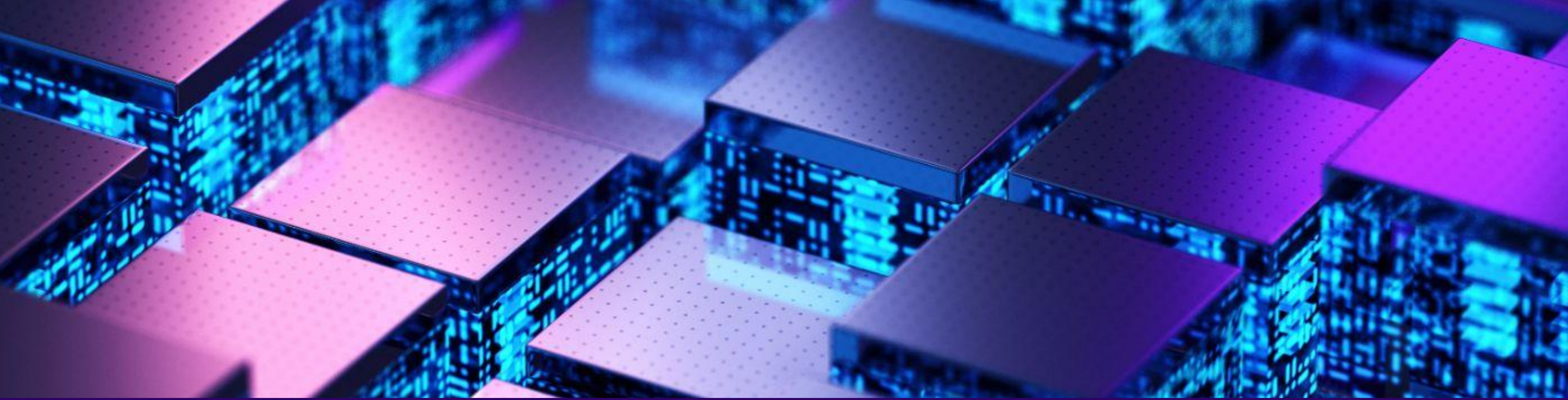
Staking

Staking

- Nodes stake exactly 32 ETH
  - Staking more requires generating separate validator keys for each 32 ETH
- Staked ETH is locked in a “deposit contract”
  - Subject to slashing conditions

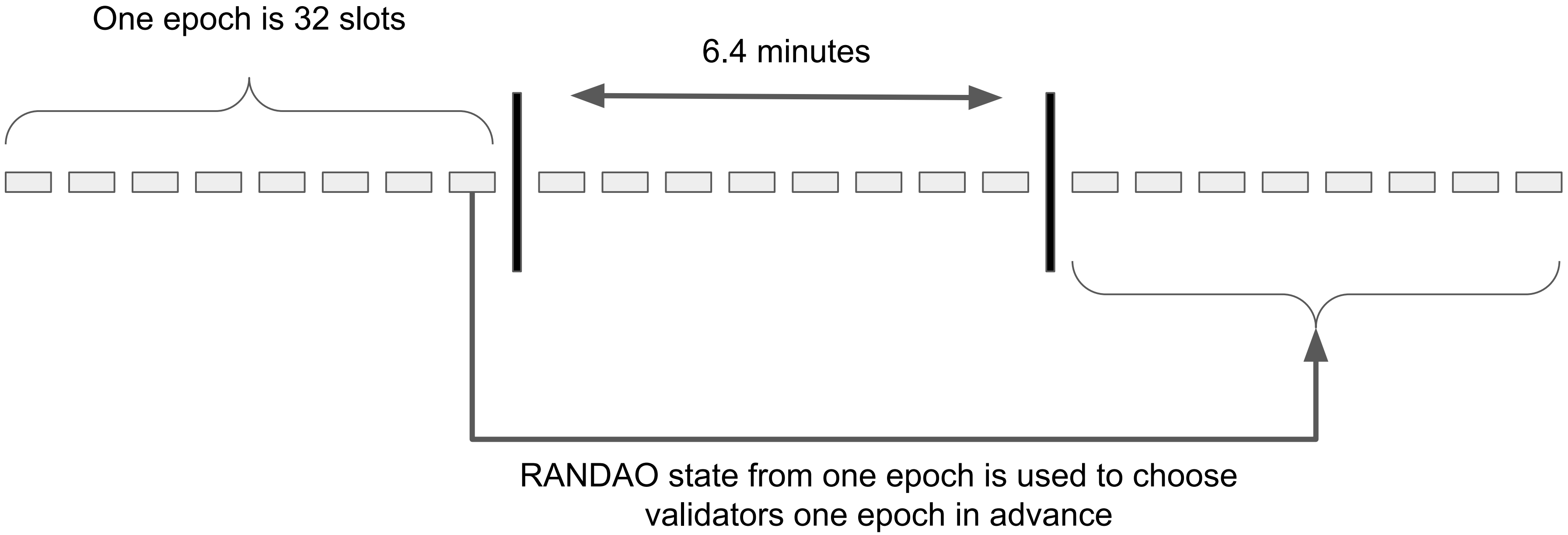


26,235,144	822,631	3.9%
TOTAL ETH STAKED	TOTAL VALIDATORS	CURRENT APR



Choosing Block Producers

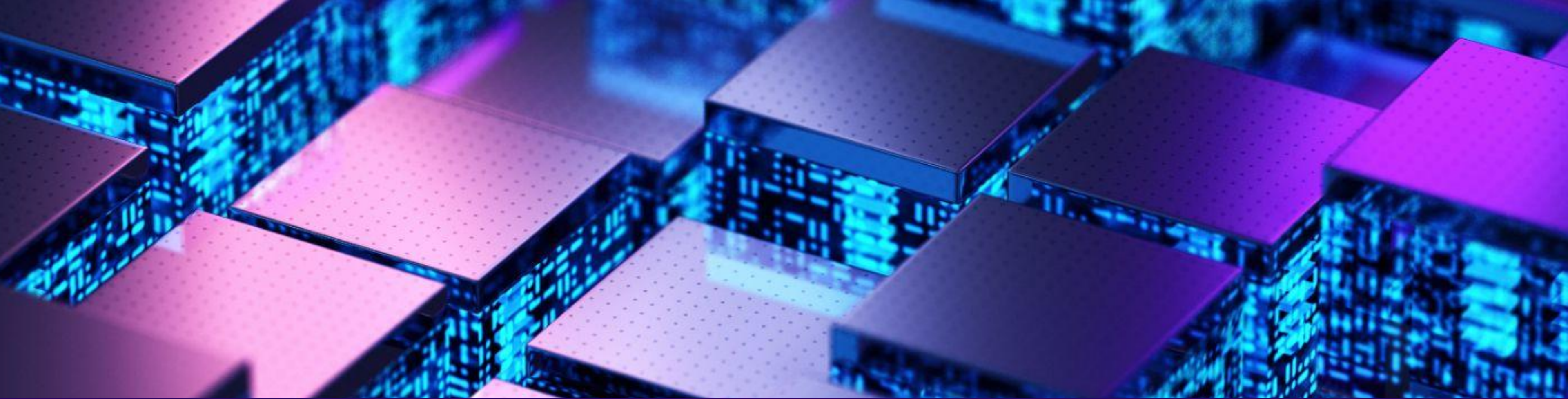
RANDAO



One epoch is 32 slots

6.4 minutes

RANDAO state from one epoch is used to choose validators one epoch in advance

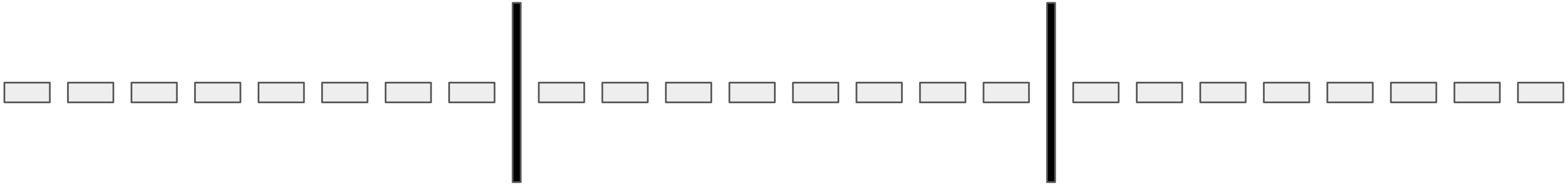


Finality

Finality

Validators "attest" to checkpoint blocks

Finality occurs at epoch level
(not block level)



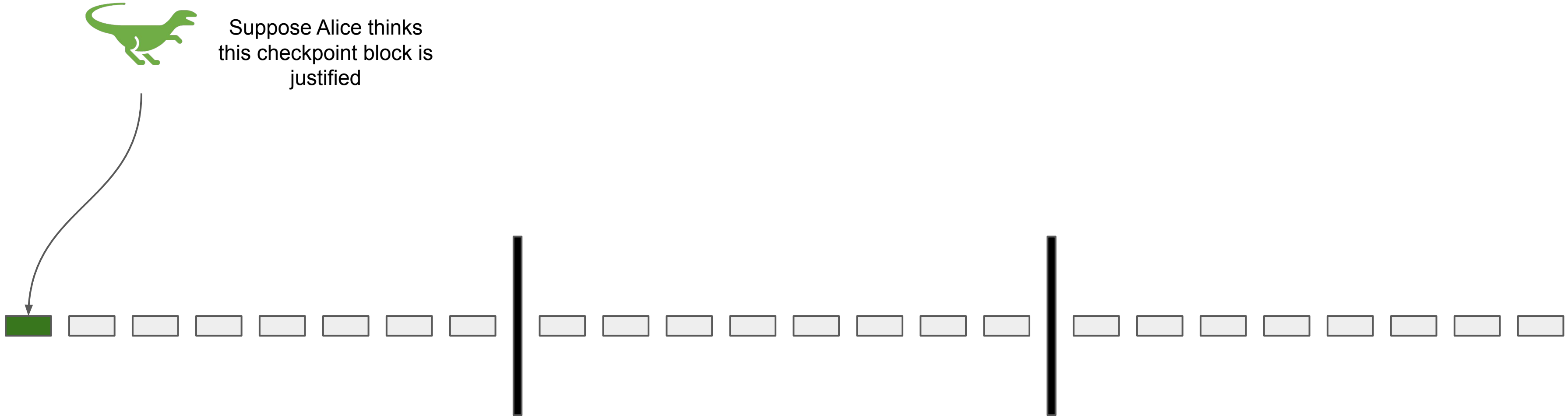
Justified - a validator calls a checkpoint justified if it has received attestations from ⅔ of the stake

Finalized - a validator calls a checkpoint finalized if it is justified and the parent of a justified checkpoint

Attestations

- Attestation includes
  - slot number
  - committee number (multiple committees per slot)
  - block hash
  - source: most recent “justified” block
  - target: first block in current epoch
- And the whole thing is digitally signed

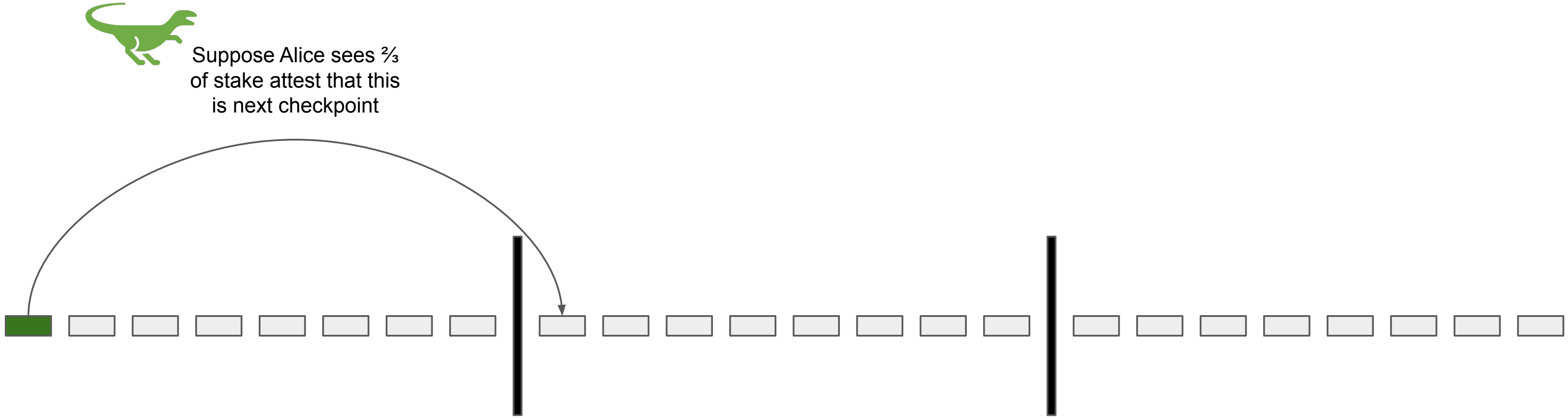
Finality



Suppose Alice thinks this checkpoint block is justified

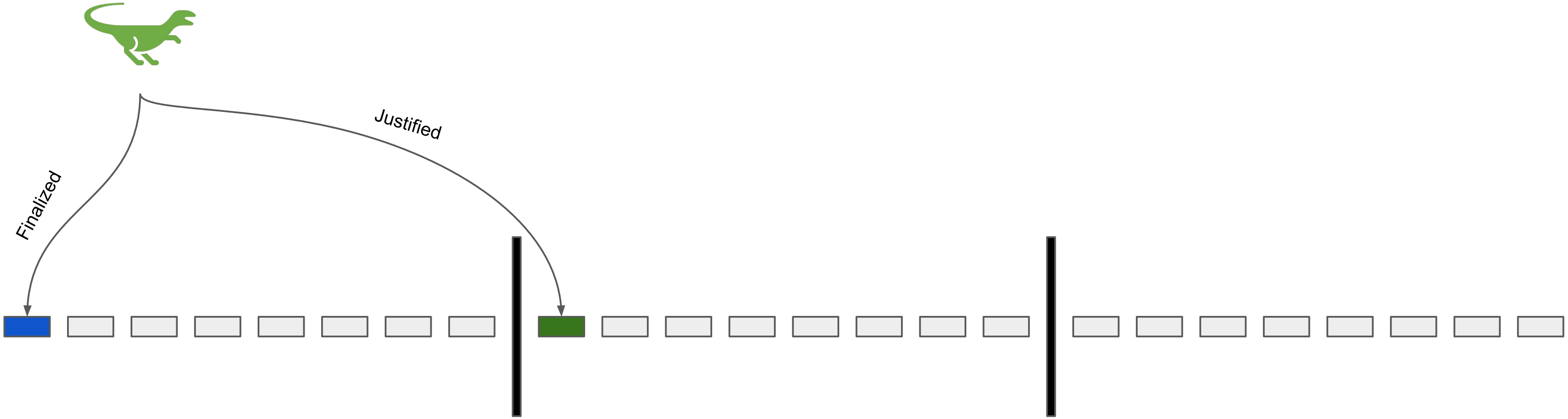
Justified - a validator calls an epoch justified if it has received attestations from ⅔ of the stake
Finalized - a validator calls a block finalized if it is justified and the parent of a justified epoch

Finality


Suppose Alice sees ⅔ of stake attest that this is next checkpoint

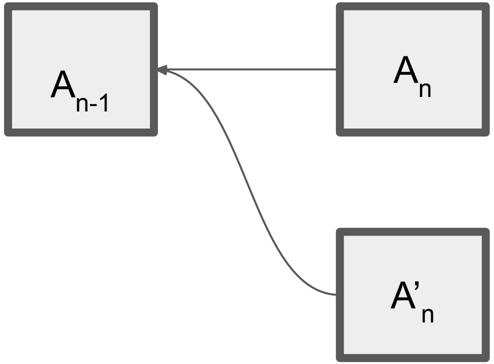
Justified - a validator calls an epoch justified if it has received attestations from ⅔ of the stake
Finalized - a validator calls a block finalized if it is justified and the parent of a justified epoch

Finality



Justified - a validator calls an epoch justified if it has received attestations from ⅔ of the stake
Finalized - a validator calls a block finalized if it is justified and the parent of a justified epoch

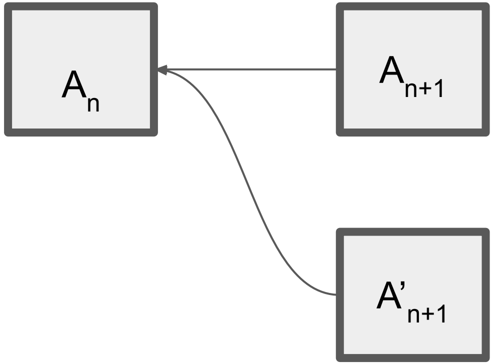
Why do we need both “finalized” and “justified”



Suppose:

- ⅔ of the stake attest $A_{n-1} \rightarrow A_n$
- ⅔ of the stake attest $A_{n-1} \rightarrow A'_n$
- then ⅓ must have attested to two conflicting blocks

Why do we need both “finalized” and “justified”



Suppose:

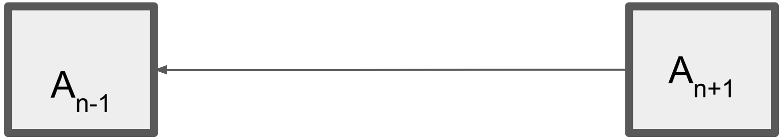
- ⅔ A_{n+1}

A_n → A'_{n+1}

ave attested to two conflicting blocks

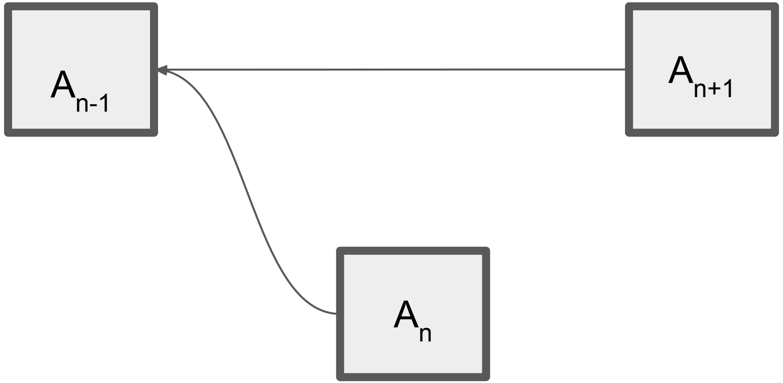
Slashable

Why do we need both “finalized” and “justified”



Suppose validators hear nothing from the block proposer in slot n. So they attest to block A_{n+1}

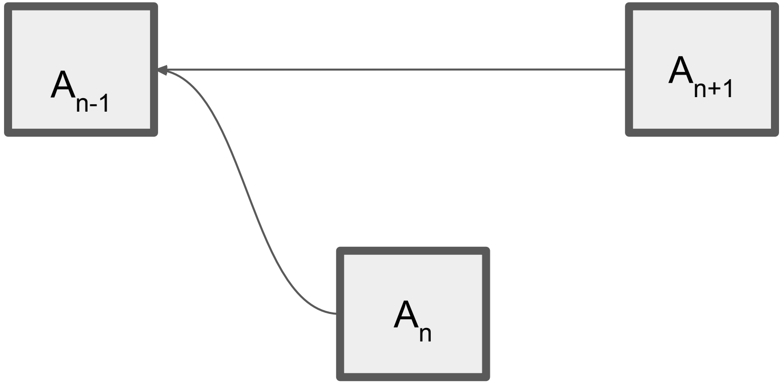
Why do we need both “finalized” and “justified”



Suppose validators hear nothing from the block proposer in slot n. So they attest to block A_{n+1}

Then, block n arrives, and they attest to that

Why do we need both “finalized” and “justified”



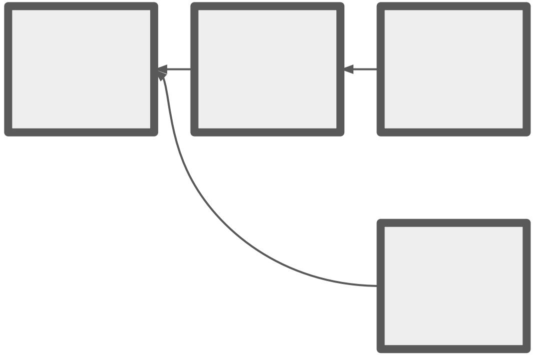
Suppose validators hear nothing from the block proposer in slot n. So they attest to block A_{n+1}

Then, block n arrives, and they attest to that

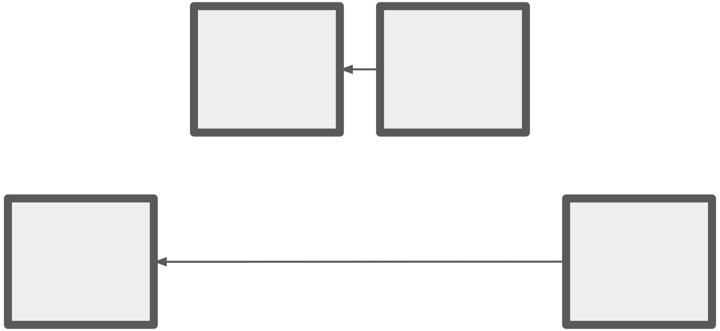
Blocks A_n and A_{n+1} are incompatible, and they’re both justified, and no one has committed a slashable offence

Slashing

Attesting to two checkpoints at the same height

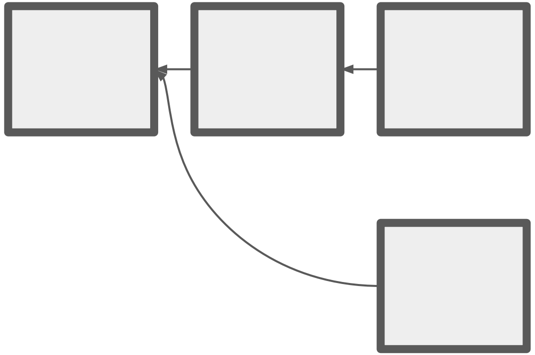


“Sandwiching” attestations

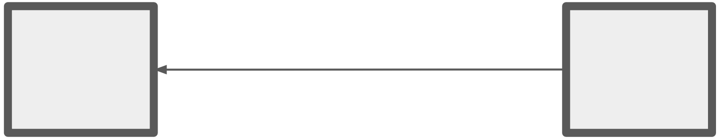


Slashing

Attesting to two checkpoints at the same height

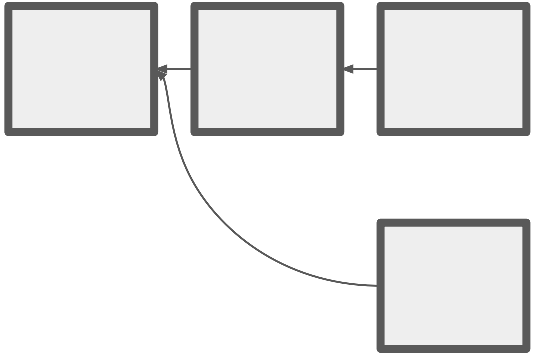


"Sandwiching" attestations

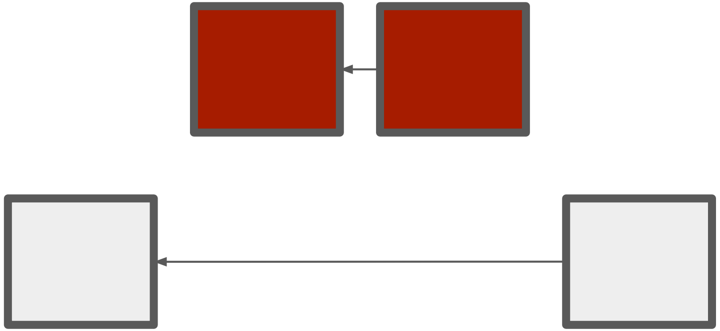


Slashing

Attesting to two checkpoints at the same height

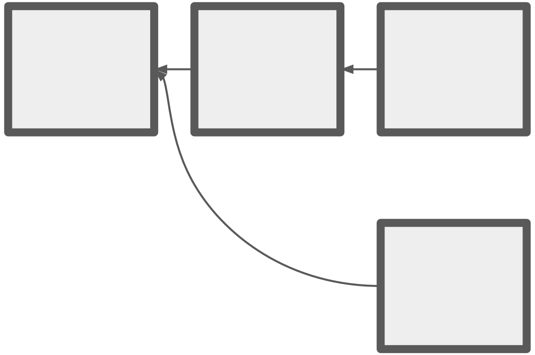


“Sandwiching” attestations

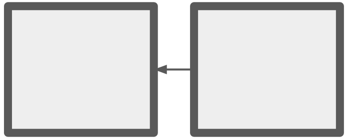


Slashing

Attesting to two checkpoints at the same height

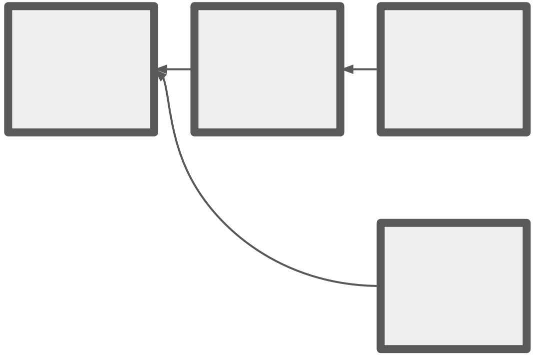


“Sandwiching” attestations

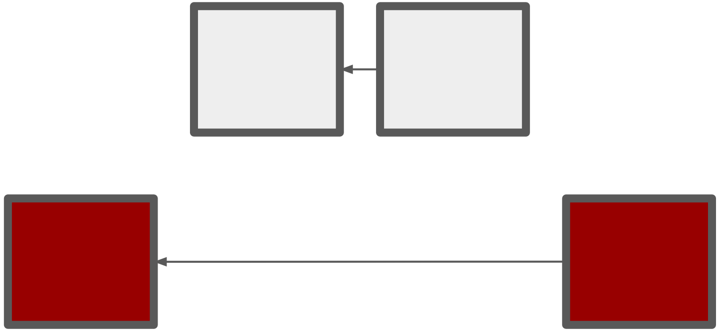


Slashing

Attesting to two checkpoints at the same height



“Sandwiching” attestations



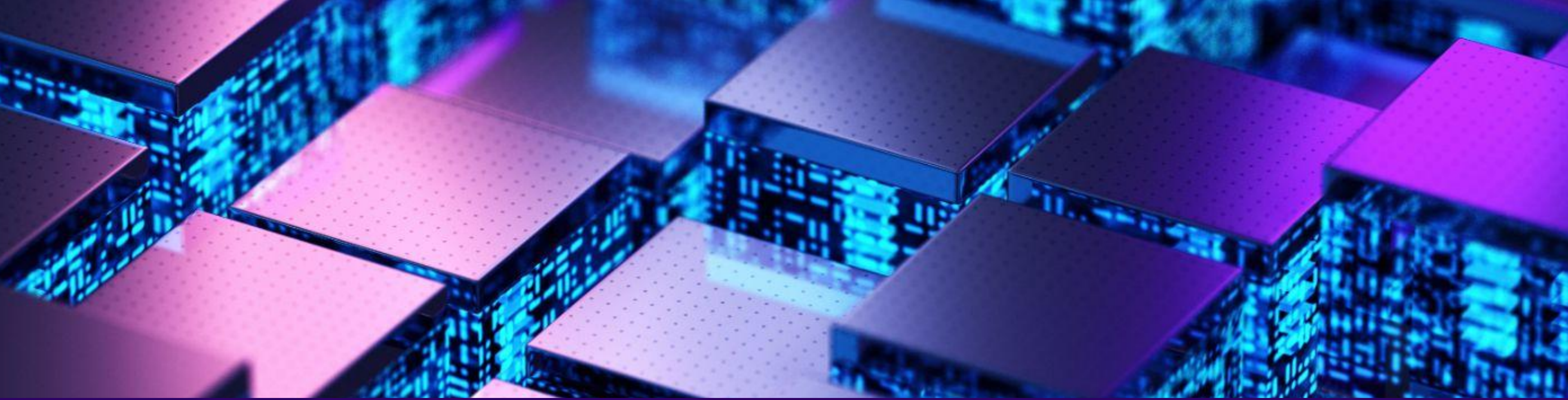
Safety

Theorem (Casper Theorem 1):

If there exist two checkpoints, not on the same branch of the chain, that have both been finalized (in the view of any nodes), then at least ⅓ of the stake has committed a slashable offence

Corollary:

If a checkpoint has been finalized in the view of any node, then that checkpoint cannot be reverted (“forked out”) without at least ⅓ of the stakers (by weight) committing a slashable offence



Aggregating attestations

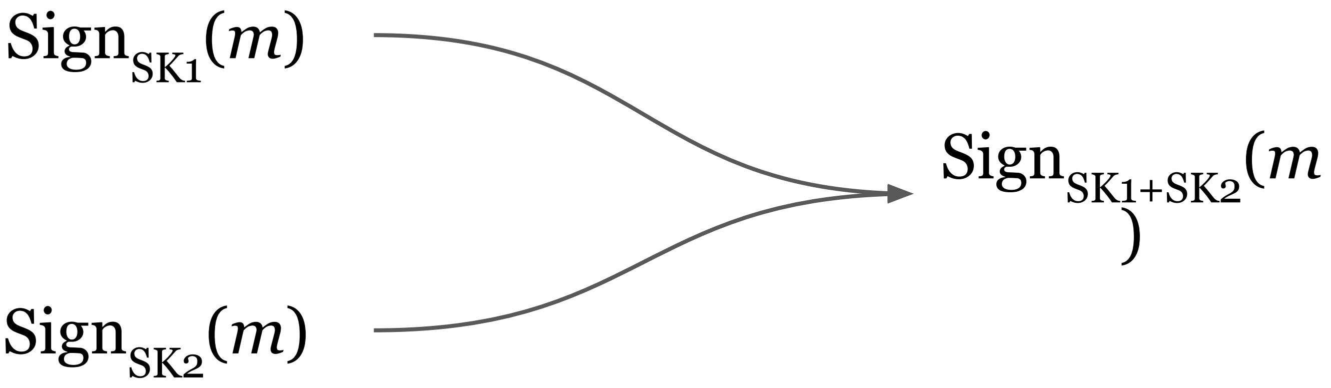
Aggregating attestations

- Problem: there are too many validators – can’t fit all their (signed) attestations into a block
- Solution:
  a. Each validator only attests to 1 block per epoch
     - Divides the number of signatures by 32
  b. Validators are divided into committees, each committee aggregates their signature into one signature
     - Can’t be done using ECDSA, so validators use BLS signature scheme

Signature aggregation

$$\begin{tikzcd}
\mathrm{Sign}_{\mathrm{SK1}}(m) \arrow[dr] & \\
& \mathrm{Sign}_{\mathrm{SK1+SK2}}(m) \\
\mathrm{Sign}_{\mathrm{SK2}}(m) \arrow[ur] &
\end{tikzcd}$$

Signature aggregation



$\mathrm{Sign}_{\mathrm{SK1}}(m)$

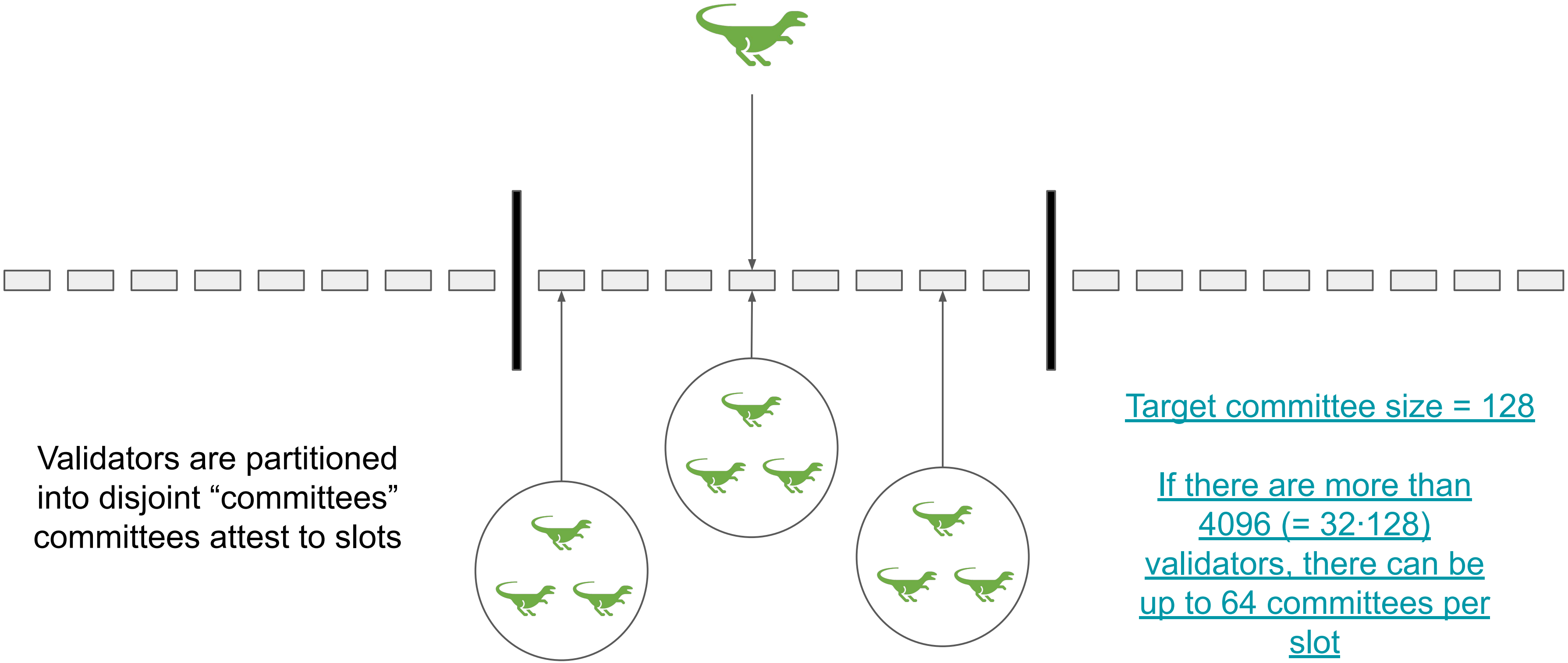
$\mathrm{Sign}_{\mathrm{SK2}}(m)$

$\mathrm{Sign}_{\mathrm{SK1+SK2}}(m)$

Valid signature under: VK1+VK2

Attestations

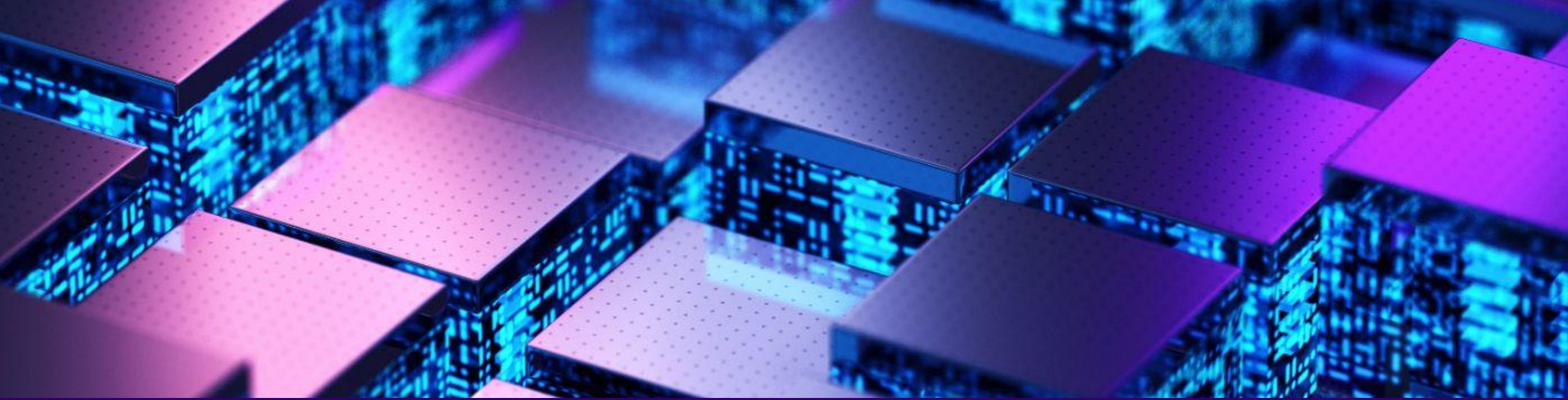
A single validator is chosen to produce a block in a given slot



Validators are partitioned into disjoint “committees” committees attest to slots

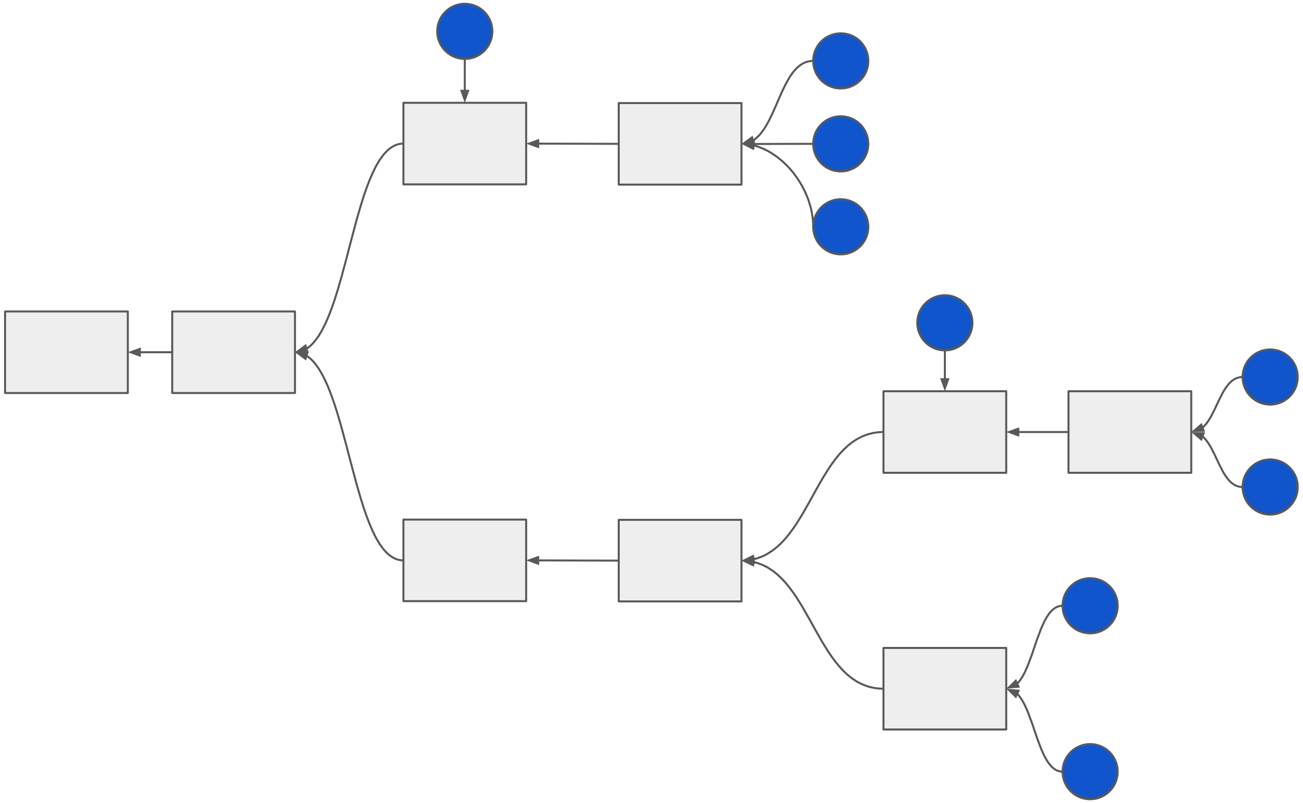
Target committee size = 128

If there are more than 4096 (= 32·128) validators, there can be up to 64 committees per slot

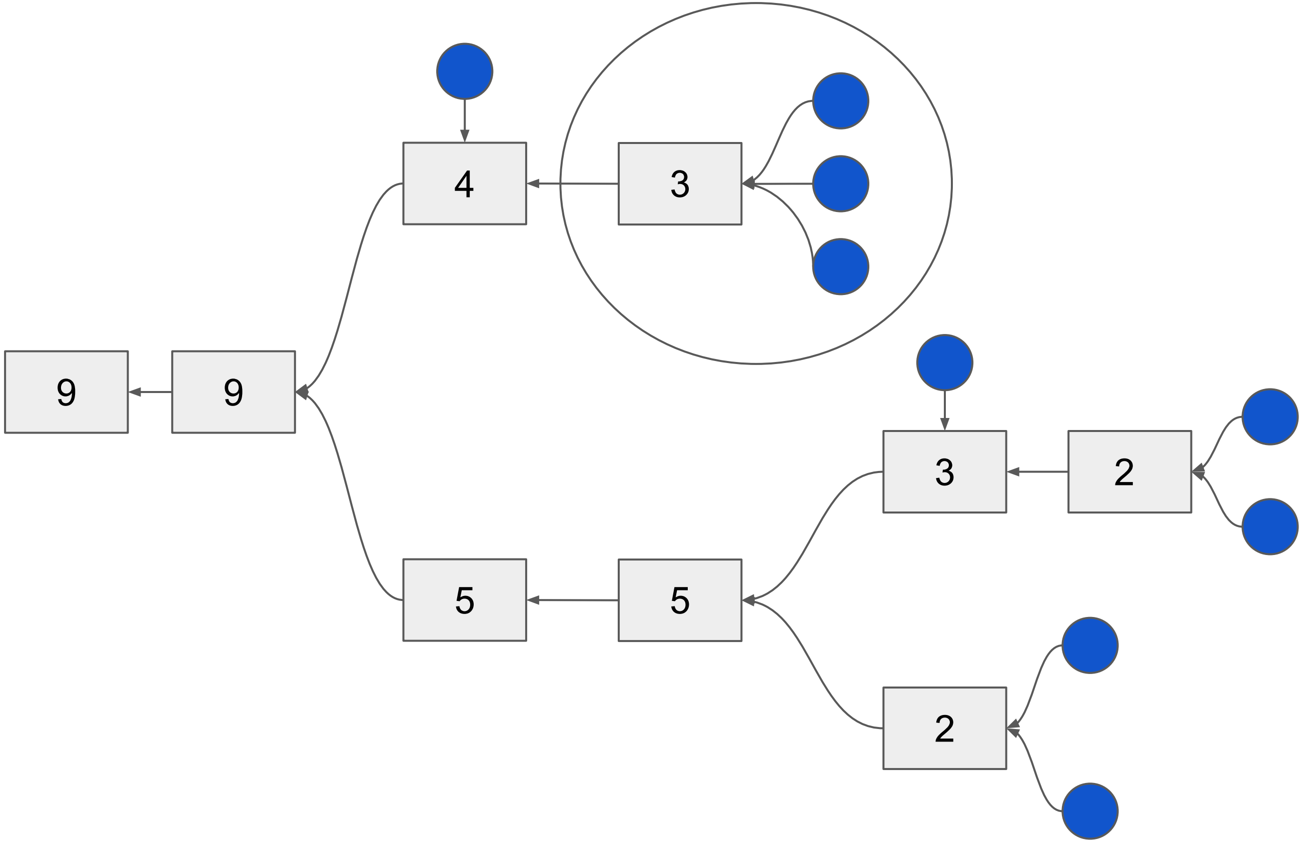


Fork choice

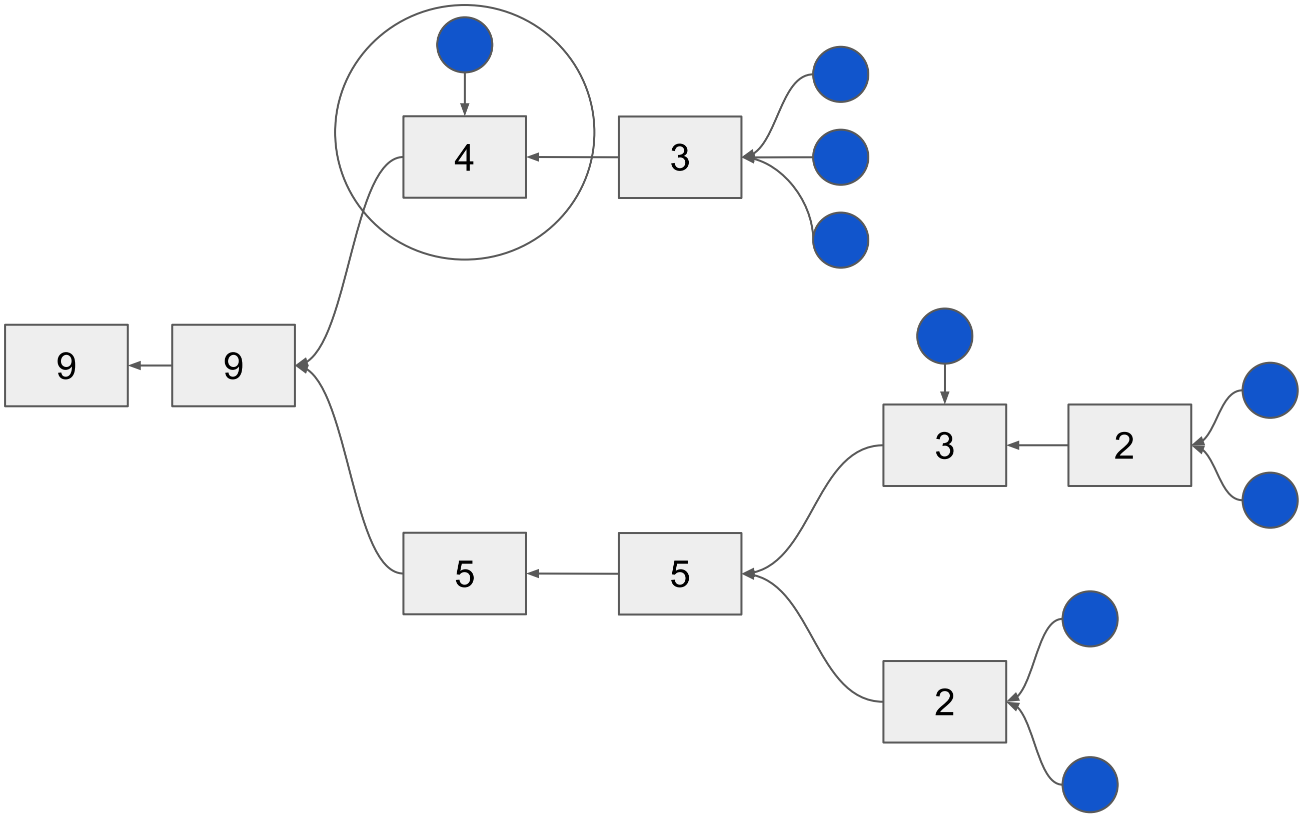
LMD-GHOST Fork Choice



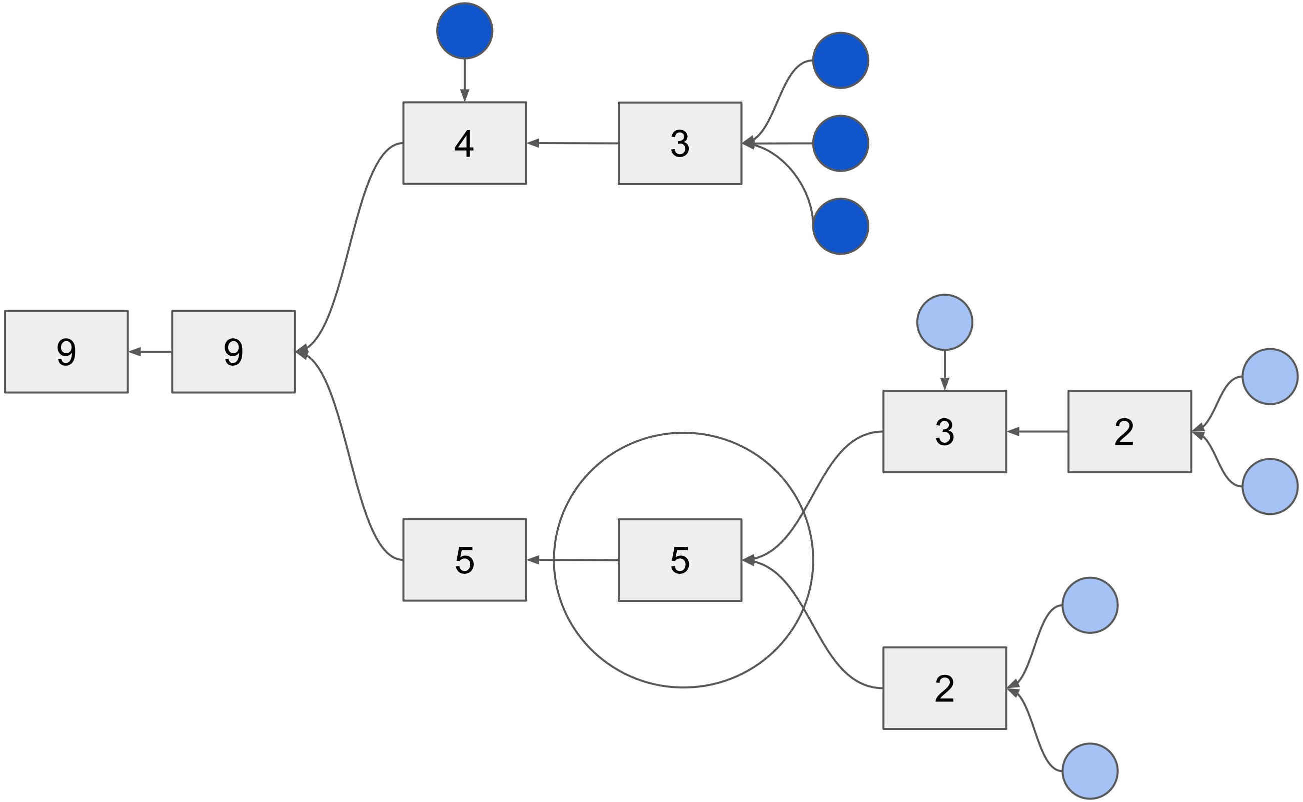
LMD-GHOST Fork Choice



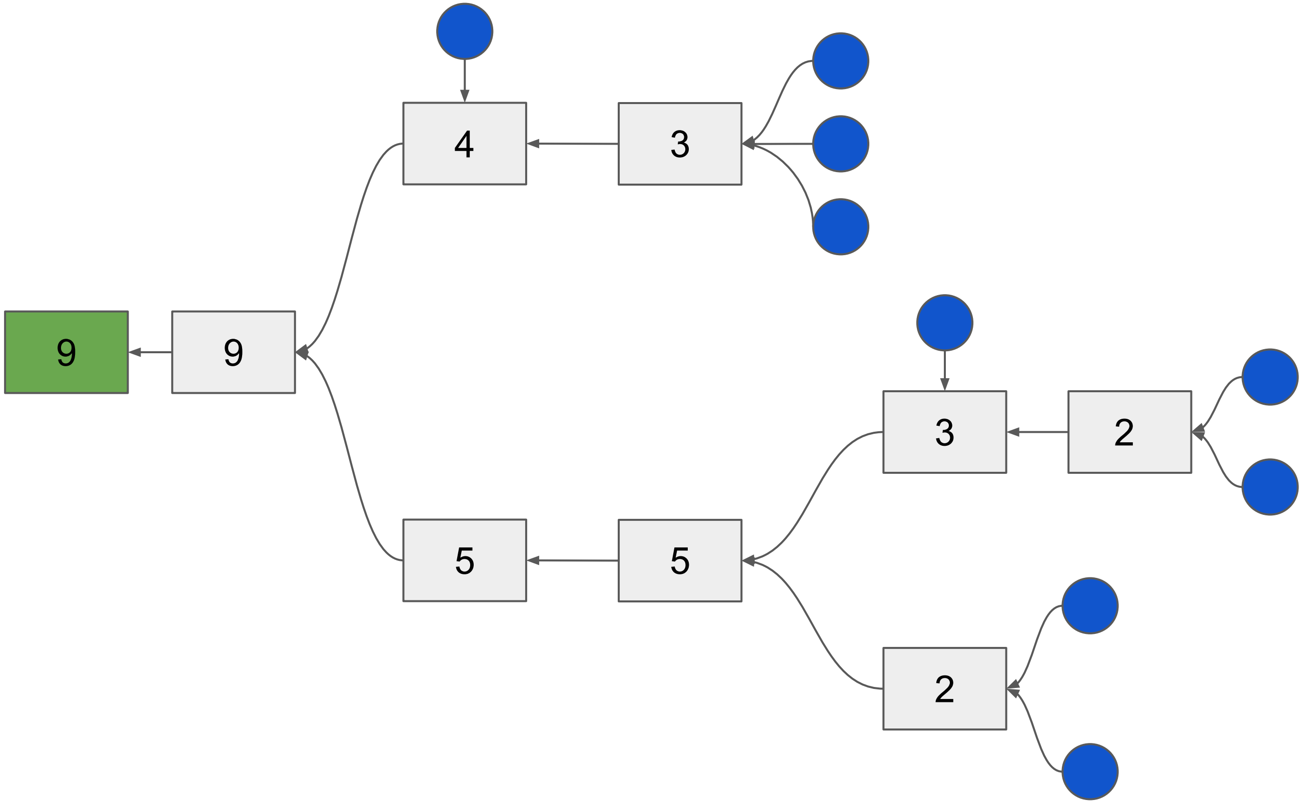
LMD-GHOST Fork Choice



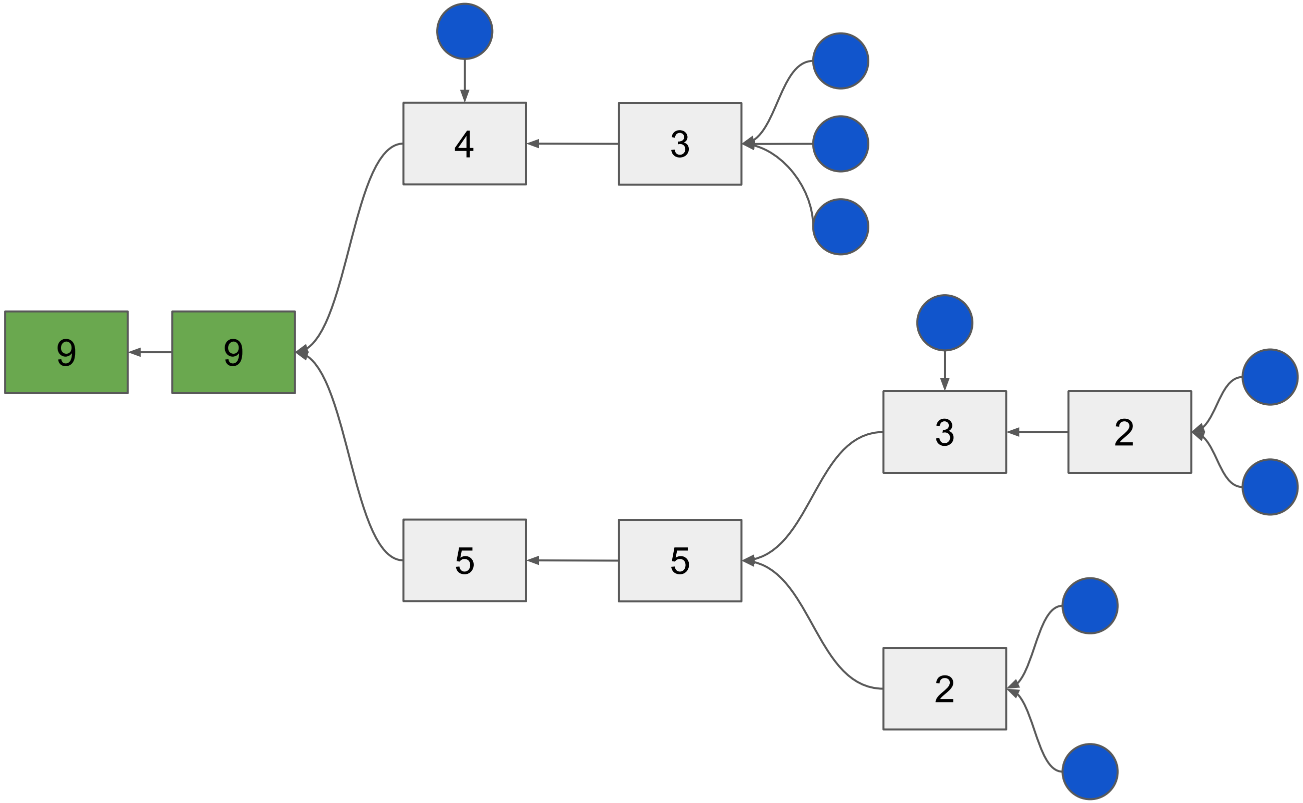
LMD-GHOST Fork Choice



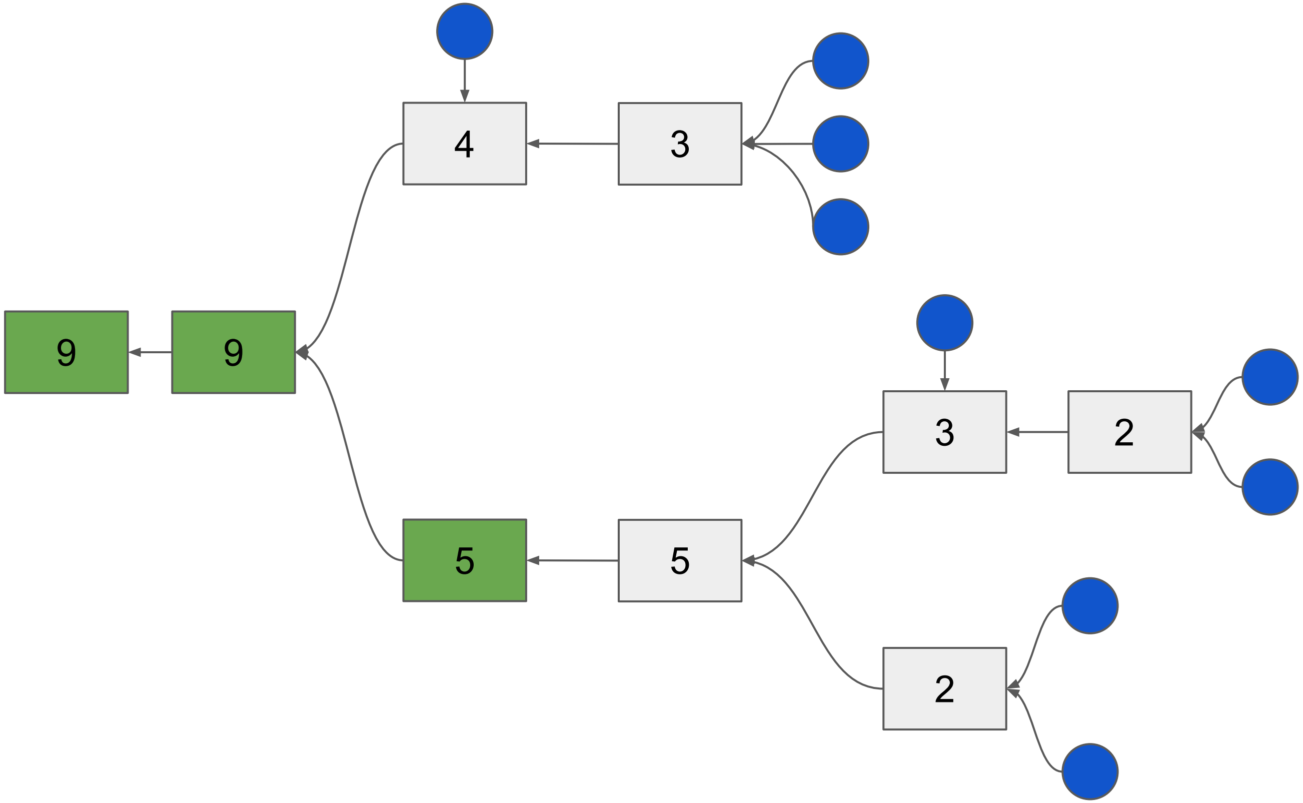
LMD-GHOST Fork Choice



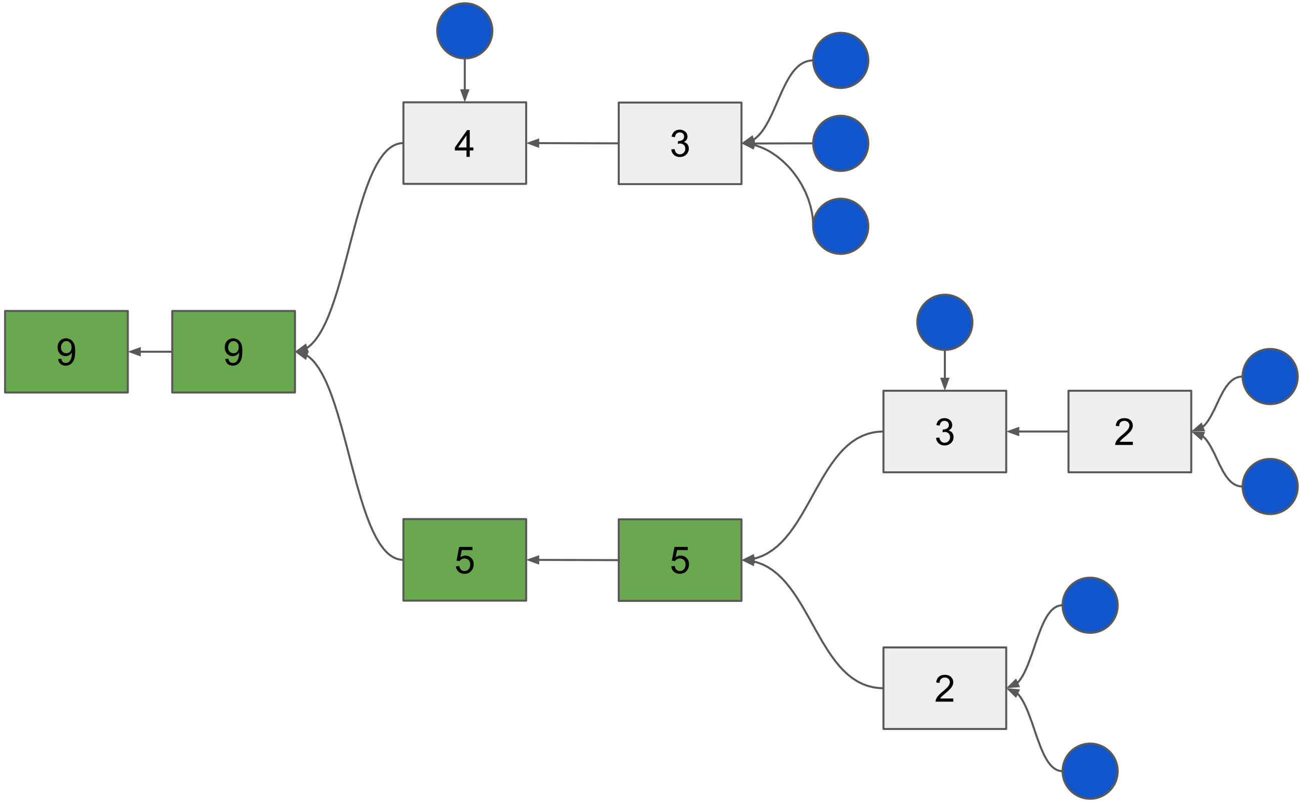
LMD-GHOST Fork Choice



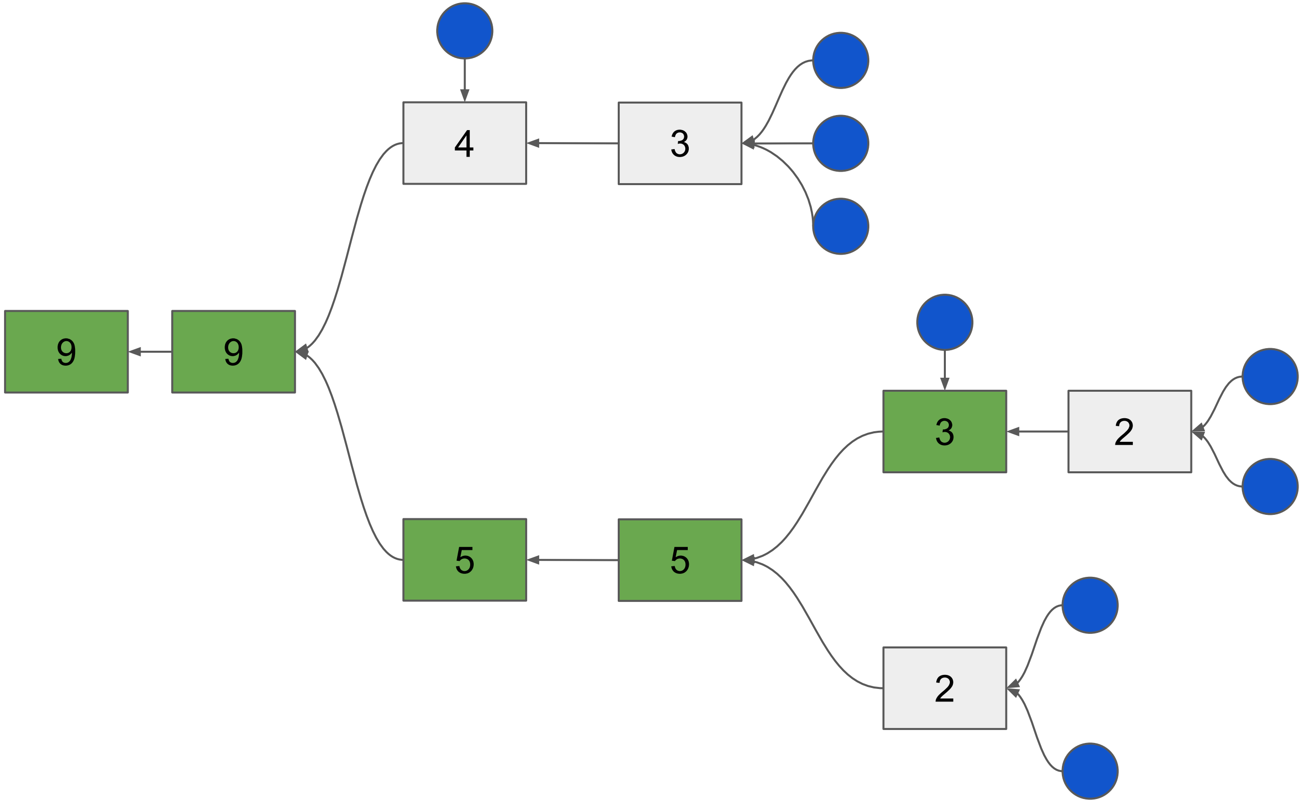
LMD-GHOST Fork Choice



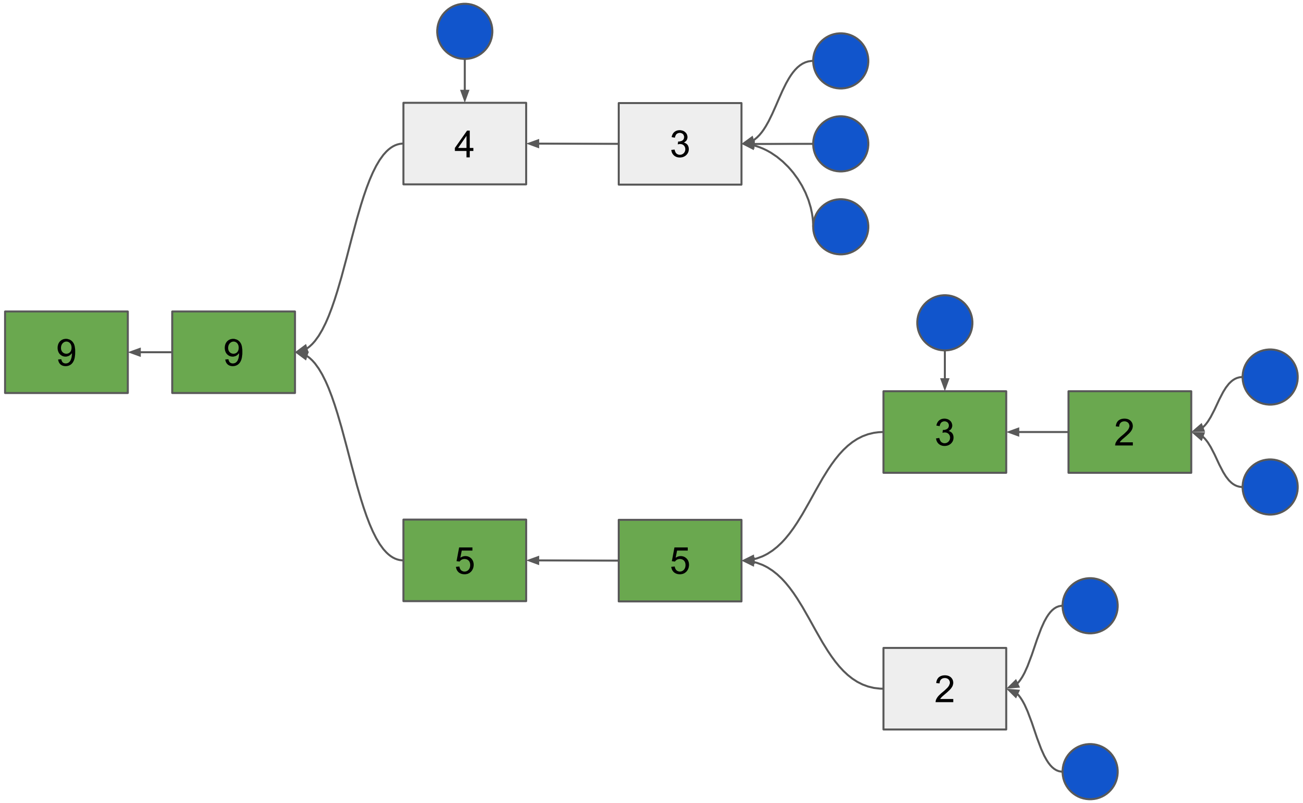
LMD-GHOST Fork Choice

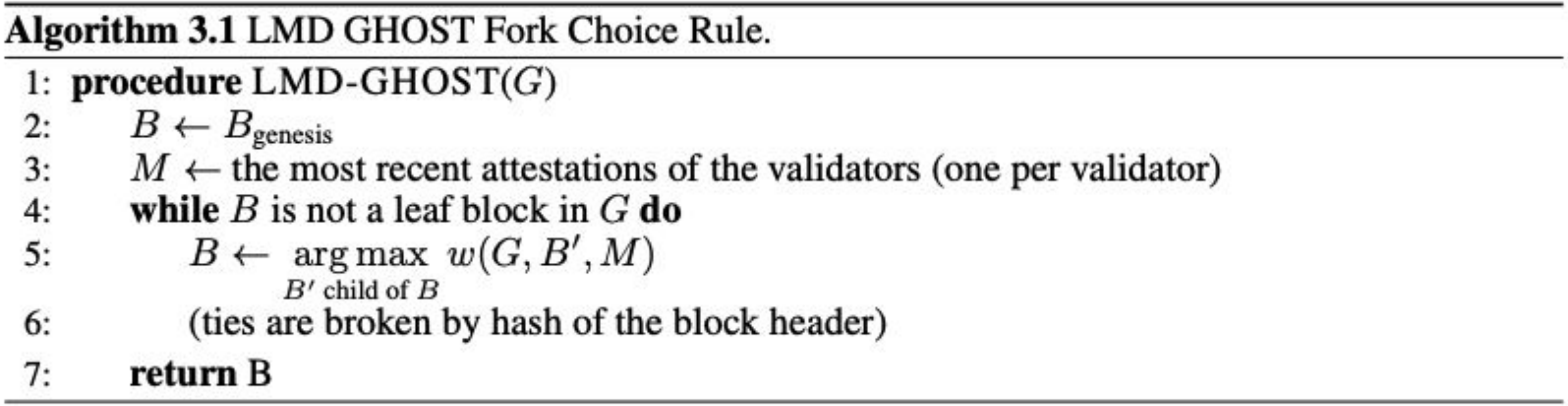


LMD-GHOST Fork Choice



LMD-GHOST Fork Choice





Algorithm 3.1 LMD GHOST Fork Choice Rule.

```
procedure LMD-GHOST($G$)
    $B \leftarrow B_{\text{genesis}}$
    $M \leftarrow$ the most recent attestations of the validators (one per validator)
    while $B$ is not a leaf block in $G$ do
        $B \leftarrow \underset{B' \text{ child of } B}{\arg\max}\ w(G, B', M)$
        (ties are broken by hash of the block header)
    return B
```