Ethereum Proof-of-Stake Protocol

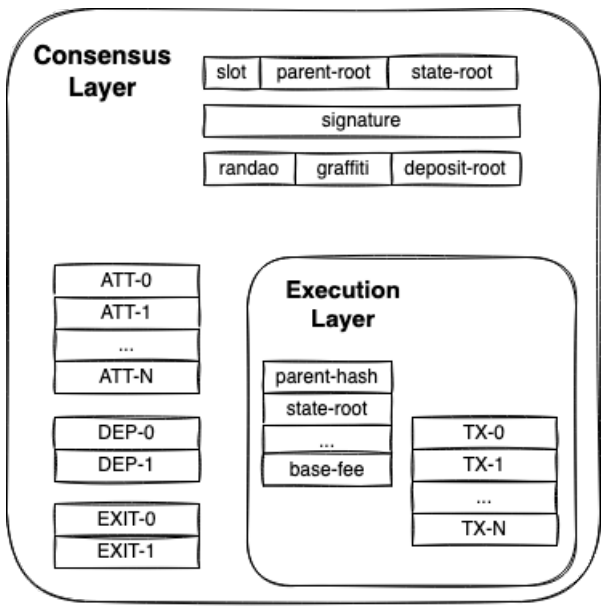
Saravanan Vijayakumaran

Associate Professor
Department of Electrical Engineering
Indian Institute of Technology Bombay

March 28, 2024

Ethereum Proof-of-Stake Protocol

- On Sept 15, 2022, Ethereum moved to a proof-of-stake consensus protocol
- Power required reduced by 99.95% from 5.13 GW to 2.62 MW
- Ethereum node components
  - **Execution client**: Executes transactions and updates world state
  - **Beacon chain client:** Implements the PoS algorithm to achieve consensus on the execution client blocks
- Ethereum blocks



Source: Ethereum Blog

Deposits and Withdrawals

- Validators = Nodes which participate in the consensus protocol
- Deposit contract = A standard Ethereum contract
- Validators send 32 ETH to the deposit contract
  - In March 2024, there are 980k validators
- Deposit and accrued rewards can be withdrawn later

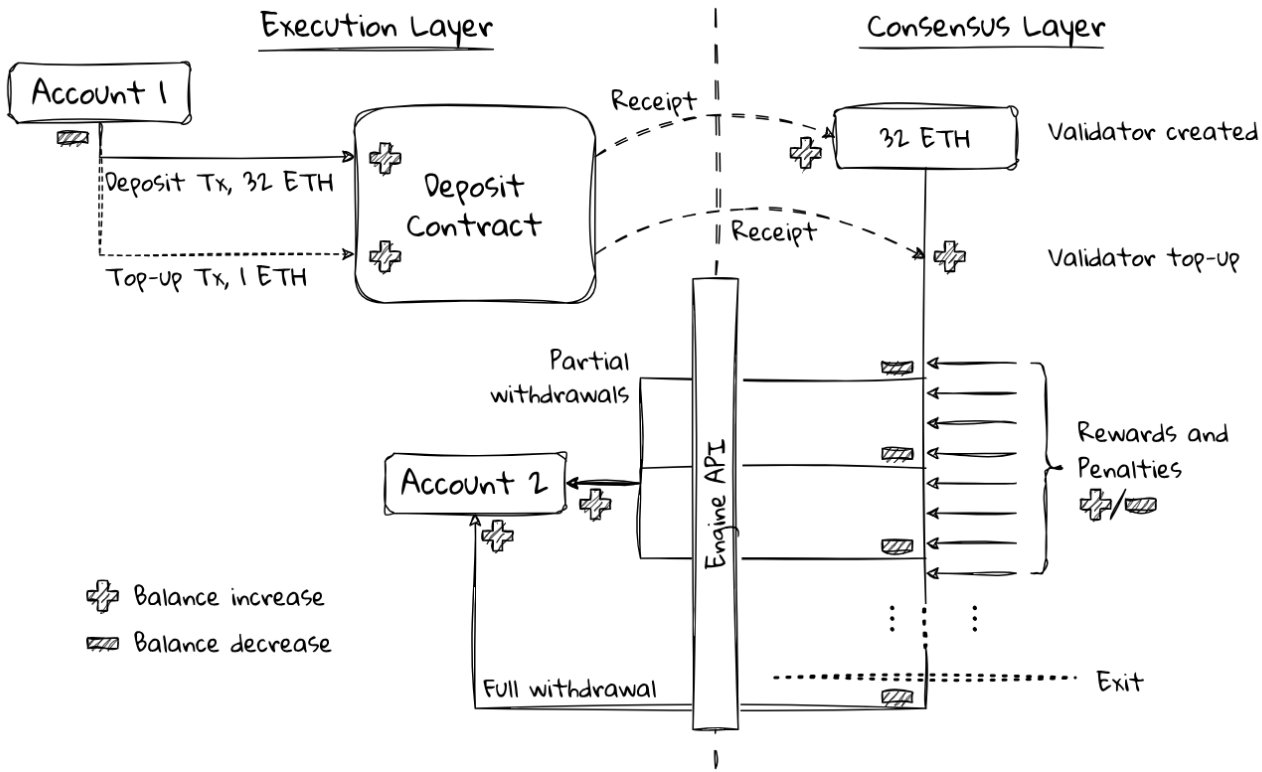


Image source: Eth2book.info

Committees

- Slots and Epochs
  - A **slot** is 12 seconds long (a new block is proposed in each slot)
  - An **epoch** consists of 32 slots, and is 6.4 minutes long
- In each slot, only $\frac{1}{32}$ of the validators vote to reduce communication overhead
- One of the active validators in a slot is selected as the block proposer
- Committees
  - The validators assigned to a slot are further divided into disjoint **committees**

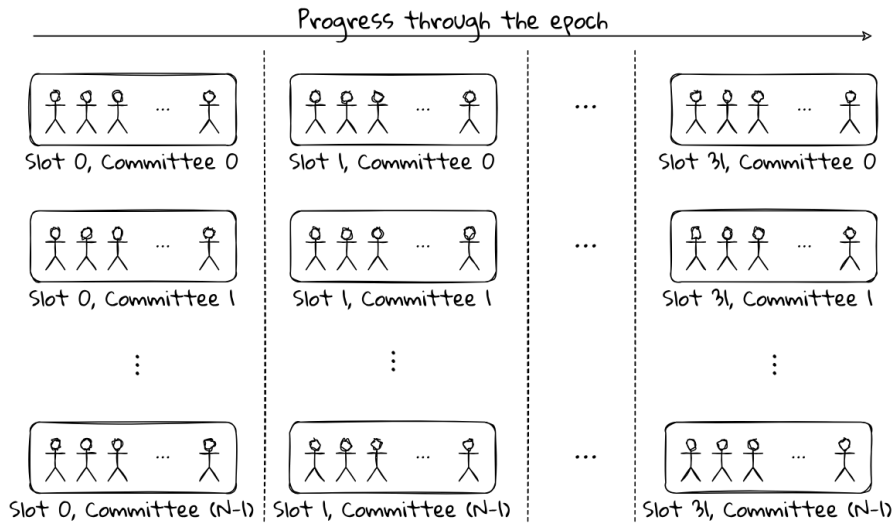


Image source: Eth2book.info

- $128 \leq$ Committee size ≤ 2048
- Maximum number of committees = 64

Aggregators

- Validators create and share **attestations** (signed votes)
- A subset of a committee is selected to be aggregators
- Aggregators aggregate attestations from committee members and forward them

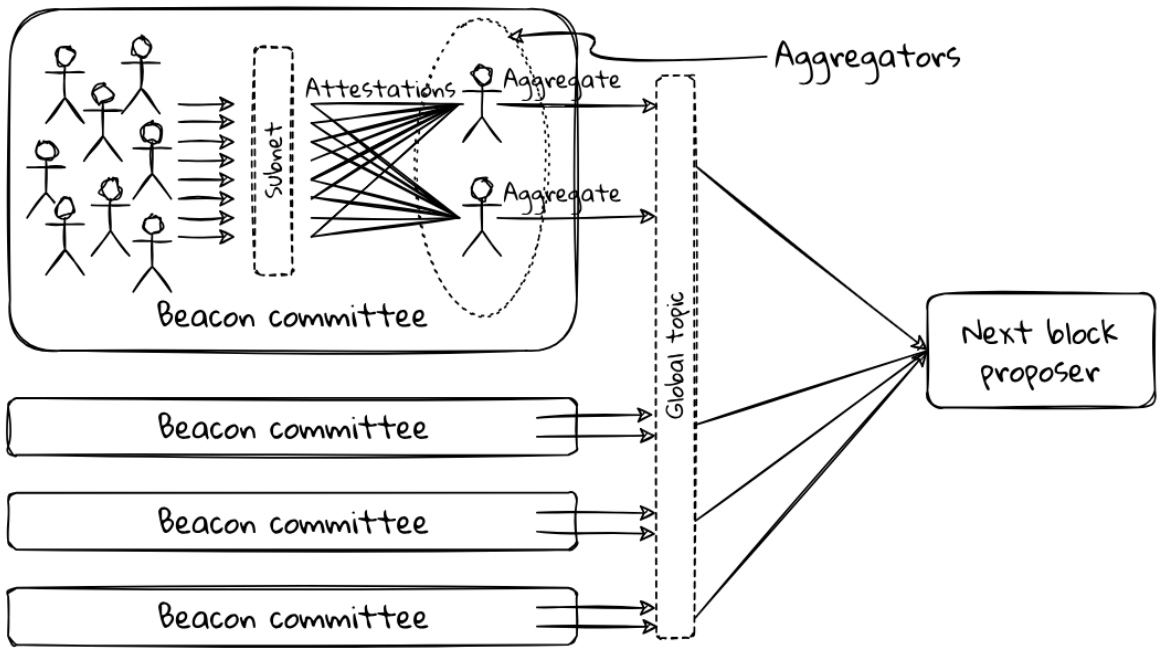


Image source: Eth2book.info

- BLS signatures
  - Signatures on the same message using different private keys can be combined
  - Combined signature has the same size as a single signature

BLS Signature Scheme

- Let G_1, G_2 be an elliptic curves of prime order p with generators g_1, g_2
- Let G_T be an order p multiplicative subgroup of a finite field
- A **pairing** is a map $e : G_1 \times G_2 \mapsto G_T$ satisfying
  1. **Bilinearity**: $\forall \alpha, \beta \in \mathbb{Z}_p$, we have $e(g_1^\alpha, g_2^\beta) = e(g_1, g_2)^{\alpha\beta}$
  2. **Non-degeneracy**: $e(g_1, g_2)$ is not the identity in G_T
- BLS Signature Scheme
  - Suppose $H : \{0,1\}^* \mapsto G_2$ is a hash function
  - Let (x, g_1^x) be a private-public key pair
  - BLS signature on message m is $\sigma = (H(m))^x$
  - Verifier checks that $e(g_1, \sigma) = e(g_1^x, H(m))$
- Aggregating BLS signatures on the same message m
  - Let $(pk_i, sk_i), i = 1, 2, \ldots, n$ be public-private key pairs
  - Suppose we have n BLS signatures $\sigma_1, \sigma_2, \ldots, \sigma_n$ on message m created using the n private keys
  - Then $\sigma_{agg} = \prod_{i=1}^n \sigma_i$ is the aggregate signature verifiable by the aggregate public key $pk_{agg} = \prod_{i=1}^n pk_i$
- Validators submit signatures verifiable by their individual public keys to prevent rogue key attacks

LMD GHOST and Casper FFG

- Components of the Ethereum consensus protocol
  - LMD GHOST = Latest Message Driven Greedy Heaviest Observed SubTree
  - Casper Friendly Finality Gadget (FFG)
- LMD GHOST is a fork-choice rule
  - Given a tree of blocks and votes, LMD GHOST suggests the best block for the head of the chain
- Casper FFG helps finalize checkpoints (blocks at regular intervals)
  - Once a checkpoint is finalized, it will not be reverted as long as $< \frac{1}{3}$ of the stake is Byzantine
  - If two conflicting checkpoints are finalized, then at least $\frac{1}{3}$ of the staked ETH will be forfeited (slashed)
  - Accountable safety

Attestations

- A validator signs the following data to create a vote for both LMD GHOST and Casper FFG

```
class AttestationData():
    slot: uint64
    committee_index: uint64

    # LMD GHOST vote
    beacon_block_root: bytes32

    # Casper FFG vote
    source: Checkpoint
    target: Checkpoint
```

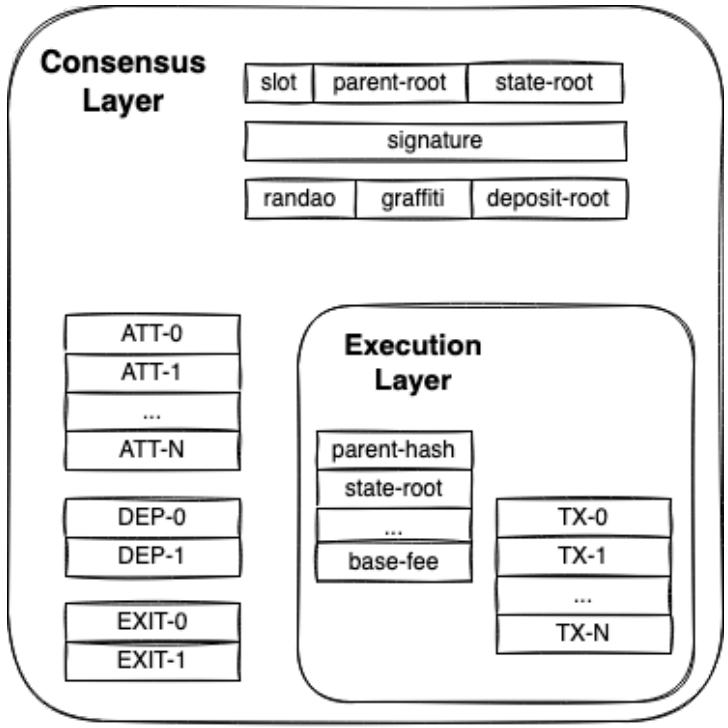
- The attestation itself is an aggregation of votes from a committee

```
class Attestation():
    aggregation_bits: Bitlist[MAX_VALIDATORS_PER_COMMITTEE]
    data: AttestationData
    signature: BLSSignature
```

- A validator sets a single bit in the `aggregation_bits` field to indicate its position in the committee
- An aggregator will set the bits corresponding to the signatures it is combining

Attestations

- Ethereum blocks



Source: Ethereum Blog

- Attestations are included in blocks
- Ensures that consensus decisions are based on the same data

LMD GHOST Fork Choice Rule

- **Message Driven**: Fork choice is decided by messages (attestations) from validators, not by blocks added by proposers
- **Latest**: Takes into account only the latest message from validators
- Each attestation for a block identifies the validators that have voted for it
- Using the total stake supporting a block as it weight, we choose the chain which is the heaviest

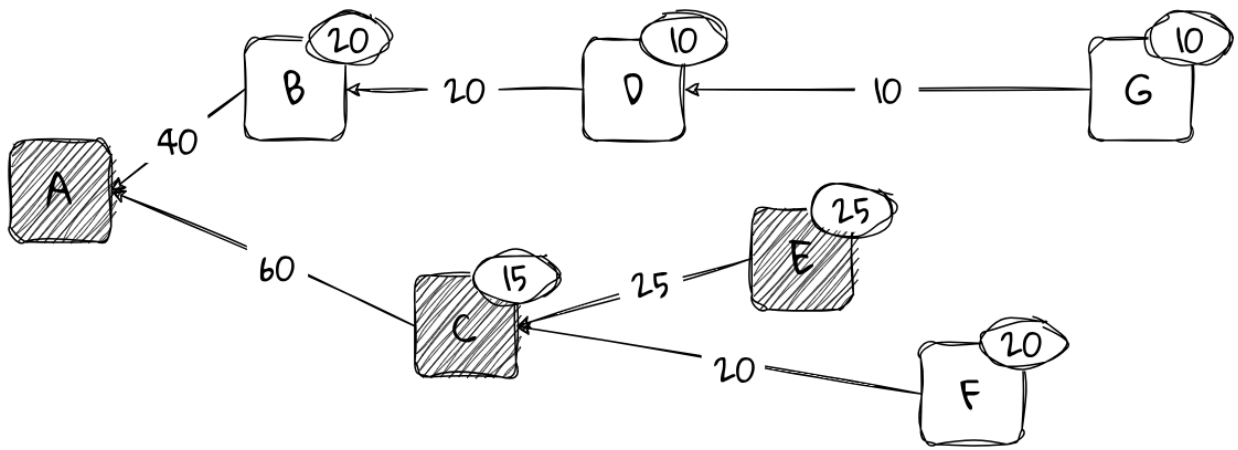


Image source: Eth2book.info

Checkpoints

- Casper FFG requires votes from at least $\frac{2}{3}$ of the validator set
- Each validator votes exactly once per epoch (32 slots)
- Checkpoint = First slot of an epoch

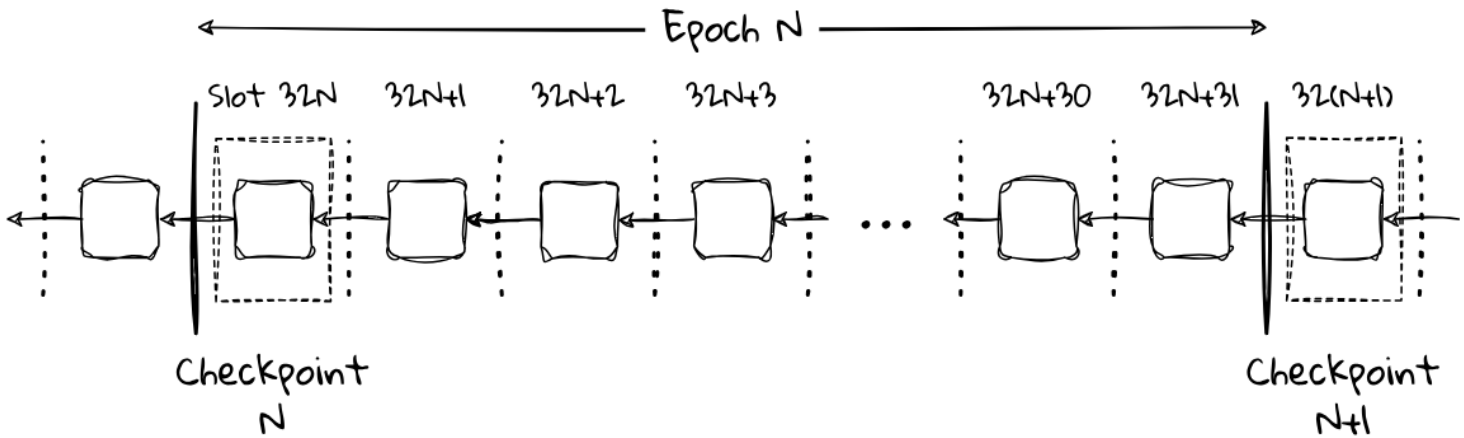


Image source: Eth2book.info

- Validators vote for checkpoints

```
class Checkpoint():
    epoch: uint64
    root: bytes32
```

- Once a checkpoint is finalized, the block in its slot and all predecessor blocks are finalized

Casper FFG

- Mechanism for finalizing checkpoints uses two rounds of voting
- Round 1 (ideally resulting in **justification**)
  - Validators tell the network what they think is the best checkpoint
- Round 2 (ideally resulting in **finalization**)
  - Validators tell the network about their highest justified checkpoint
- For efficiency, both rounds are combined into a single FFG vote

```
# Casper FFG vote
source: Checkpoint
target: Checkpoint
```

- An FFG vote is a link $s \rightarrow t$, where s is the source checkpoint and t is the target checkpoint
  - s is the highest justified checkpoint (which might be finalized)
  - t is the current epoch checkpoint (which might be justified)
- s must be an ancestor of t but need not be the immediate parent

Casper FFG Justification

- A link $s \rightarrow t$ is a **supermajority link** when $\frac{2}{3}$ of the validators have voted for it
- If a validator sees a supermajority link from **justified** checkpoint c_1 to checkpoint c_2, it considers c_2 as **justified**
- Example

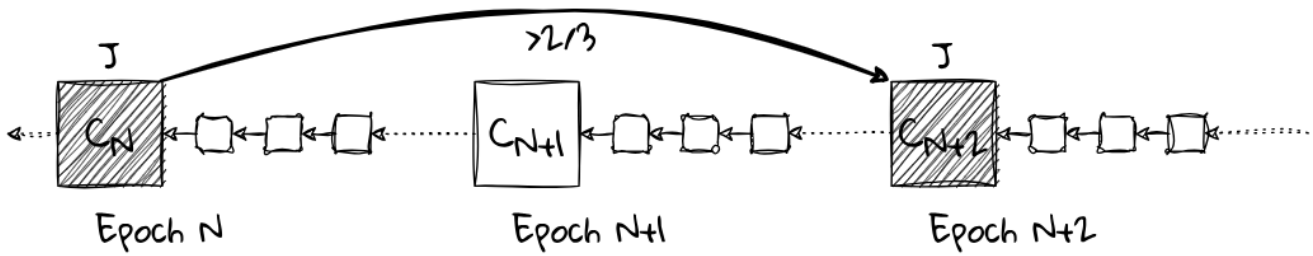


Image source: Eth2book.info

  - Checkpoint C_N from epoch N was already justified
  - A validator receives a supermajority link $C_N \rightarrow C_{N+2}$
  - Checkpoint C_{N+2} from epoch $N + 2$ is now justified
- Source and target checkpoints in a link $s \rightarrow t$ need not be consecutive
  - In the above example, suppose the current epoch is $N + 2$
  - t has to equal C_{N+2}
  - If C_{N+1} was not justified due to network delays, then s has to be a previously justified checkpoint like C_N

Casper FFG Finalization

- If a validator sees a supermajority link from **justified** checkpoint c_1 to checkpoint c_2 which is a direct child of c_1, it considers c_1 as **finalized**
- Example

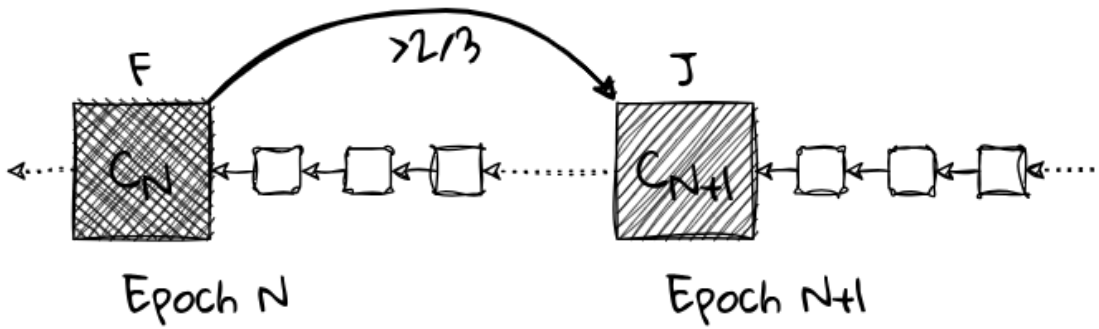


Image source: Eth2book.info

  - Checkpoint C_N from epoch N was already justified
  - A validator receives a supermajority link $C_N \rightarrow C_{N+1}$, where C_{N+1} is the epoch $N + 1$ checkpoint
  - Checkpoint C_N from epoch N is now finalized
  - Checkpoint C_{N+1} from epoch $N + 1$ is now justified
- For finalizing s, source and target checkpoints in the supermajority link $s \rightarrow t$ must be consecutive
  - Can be relaxed to links of the form $C_N \rightarrow C_{N+k}$ if the intermediate checkpoints are justified

Casper FFG Rules

- For a checkpoint c in slot $32N$, let $h(c) = N$ denote its height (epoch number)
- Rule 1: **No double vote**
  - A validator must not publish distinct votes $s_1 \rightarrow t_1$ and $s_2 \rightarrow t_2$ such that $h(t_1) = h(t_2)$
- Rule 1 could be violated using different source checkpoints

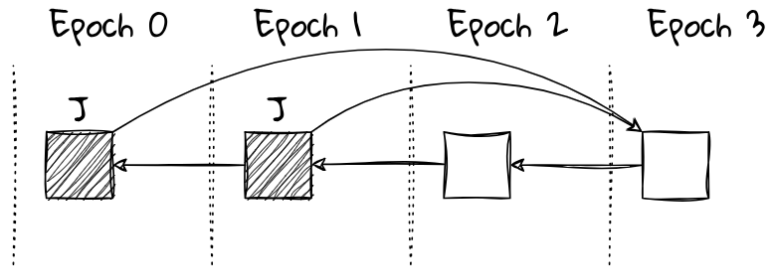


Image source: Eth2book.info

- Rule 1 could be violated using different target checkpoints

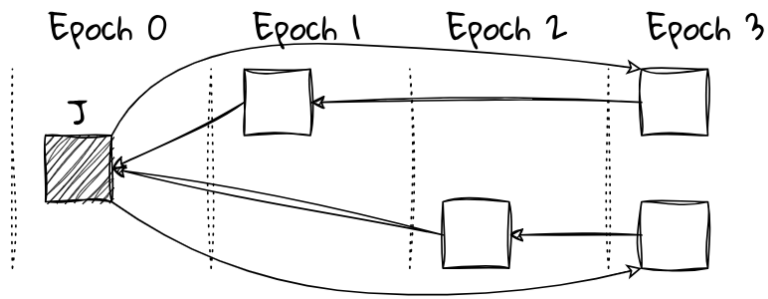


Image source: Eth2book.info

Casper FFG Rules

- Rule 2: **No surround vote**
  - A validator must not publish distinct votes $s_1 \rightarrow t_1$ and $s_2 \rightarrow t_2$ such that $h(s_1) < h(s_2) < h(t_2) < h(t_1)$
- Rule 2 could be violated on the same branch

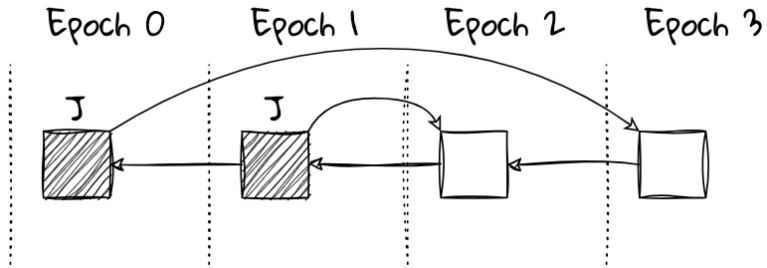


Image source: Eth2book.info

- Rule 2 could be violated using different branches

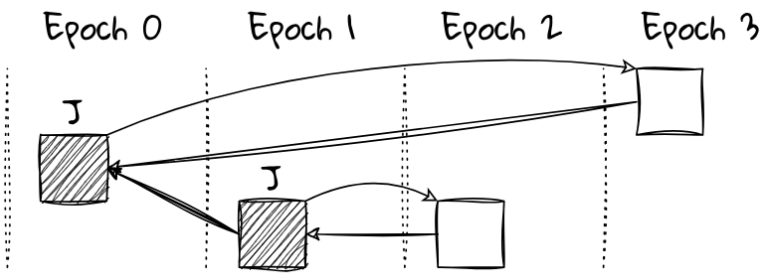


Image source: Eth2book.info

Accountable Safety in Casper FFG

- **Conflicting Checkpoints**: Two checkpoints c_1 and c_2 are conflicting if neither is an ancestor or descendant of the other
- Example: *B* and *C* are conflicting checkpoints (need not have the same height)

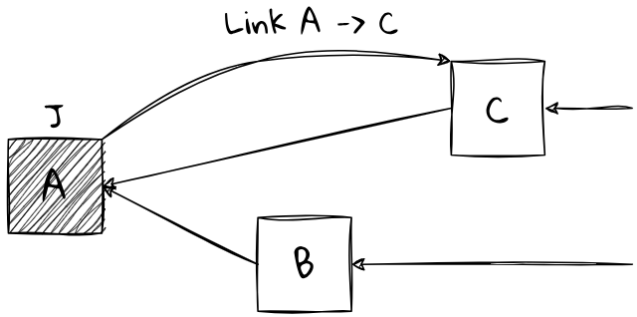


Image source: Eth2book.info

- **Accountable Safety**: If two conflicting checkpoints are finalized, then validators representing at least $\frac{1}{3}$ of the total stake will be slashed
- If conflicting checkpoints get finalized, we can identify exactly which validators violated one of the two Casper FFG rules

Proof of Accountable Safety

- Suppose that $< \frac{1}{3}$ of the validators are Byzantine
- Suppose two conflicting checkpoints a_m and b_n in epochs m and n are finalized
- By rule 1, $m \neq n$. Assume that $m < n$.
- Then there exists a series of supermajority links from the genesis block r to b_n

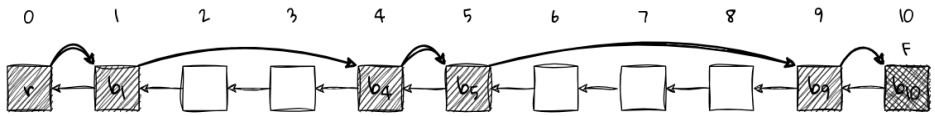


Image source: Eth2book.info

- Let these links be $\{r \rightarrow b_{i_1}, b_{i_1} \rightarrow b_{i_2}, \ldots, b_{i_{k-1}} \rightarrow b_n\}$ and let $\mathcal{B} = \{r, b_{i_1}, b_{i_2}, \ldots, b_{i_{k-1}}, b_n\}$
- By definition of finalization, there is a supermajority link $a_m \rightarrow a_{m+1}$
- Both $a_m \notin \mathcal{B}$ and $a_{m+1} \notin \mathcal{B}$, as it would make a_m an ancestor of b_n
- Also $b_m, b_{m+1} \notin \mathcal{B}$, as each epoch can have at most one justified checkpoint
- The pair (a_m, a_{m+1}) must fall between the epochs of consecutive elements $b_{i_{j-1}}$ and b_{i_j} in $\mathcal{B}$

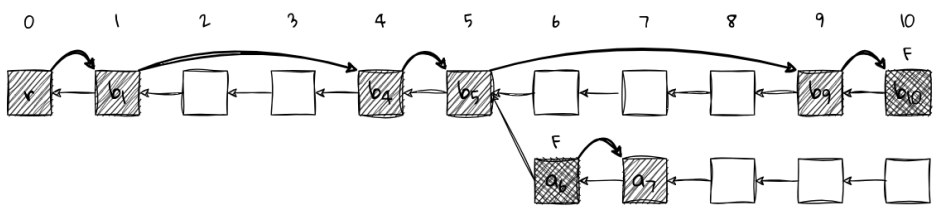


Image source: Eth2book.info

- This contradicts rule 2. Hence conflicting checkpoints cannot be finalized.

References

- Upgrading Ethereum, Ben Edgington `https://eth2book.info/capella/`
- Ethereum 1.0 PoW vs Ethereum 2.0 PoS power consumption comparison `https://blog.ethereum.org/2021/05/18/country-power-no-more`
- Validator withdrawals `https://eips.ethereum.org/EIPS/eip-4895`