# Blockchain and Distributed Ledger technologies

SAPIENZA
UNIVERSITÀ DI ROMA

Massimo La Morgia
massimo.lamorgia@uniroma1.it

# Ethereum nodes

In Ethereum a node is made of two clients:

The execution client (also known as the Execution Engine, EL client or formerly the Eth1 client) listens to new transactions broadcasted in the network, executes them in EVM, and holds the latest state and database of all current Ethereum data.

The consensus client (also known as the Beacon Node, CL client or formerly the Eth2 client) They are responsible for all consensus-related logic including the fork-choice algorithm, processing attestations and managing proof-of-stake rewards and penalties.

There is also a third piece of software, known as a 'validator' that can be added to the consensus client, allowing a node to actively participate to the proof of stake, staking 32 ETH and becoming a validator.

# Ethereum nodes

In Ethereum there are three different kind of nodes:

## Full node

Full nodes do a block-by-block validation of the blockchain, including downloading and verifying the block body and state data for each block. full nodes only keep a local copy of relatively recent data (typically the most recent 128 blocks), allowing older data to be deleted to save disk space.

- It stores full blockchain data, but not all state data (typically only those related to the most recent 128 blocks).
- Participates in block validation, verifies all blocks and states.
- All states can be either retrieved from local storage or regenerated from 'snapshots' by a full node.
- Serves the network and provides data on request.

## Archive node

Archive nodes are full nodes that verify every block from genesis and never delete any of the downloaded data. It is needed if you want to query something like an account balance at block #4,000,000. This kind of node requires a lot of storage.

# Ethereum nodes

## Light node

Light nodes only download block headers.

These headers contain summary information about the contents of the blocks. Any other information the light node requires gets requested from a full node.

The light node can then independently verify the data they receive against the state roots in the block headers.

Light nodes enable users to participate in the Ethereum network without the powerful hardware or high bandwidth required to run full nodes. Eventually, light nodes might run on mobile phones or embedded devices.

The light nodes cannot be validators, but they can access the Ethereum blockchain with the same functionality and security guarantees as a full node.

# Ethereum nodes

## Validators

To participate as a validator, a user must deposit **32 ETH** into the **deposit contract** and run three separate pieces of software:

- the execution client
- the consensus client
- the validator client.

Validators are the nodes that form the consensus. Differently from the meaning of their name, they don't really validate anything. (Validation is done by standard nodes).

Each validator has a maximum balance of 32 ETH, but stakers can stake all their ETH. For every 32 ETH staked, one validator is activated.

# Ethereum Time

Time in proof-of-stake Ethereum is divided into:

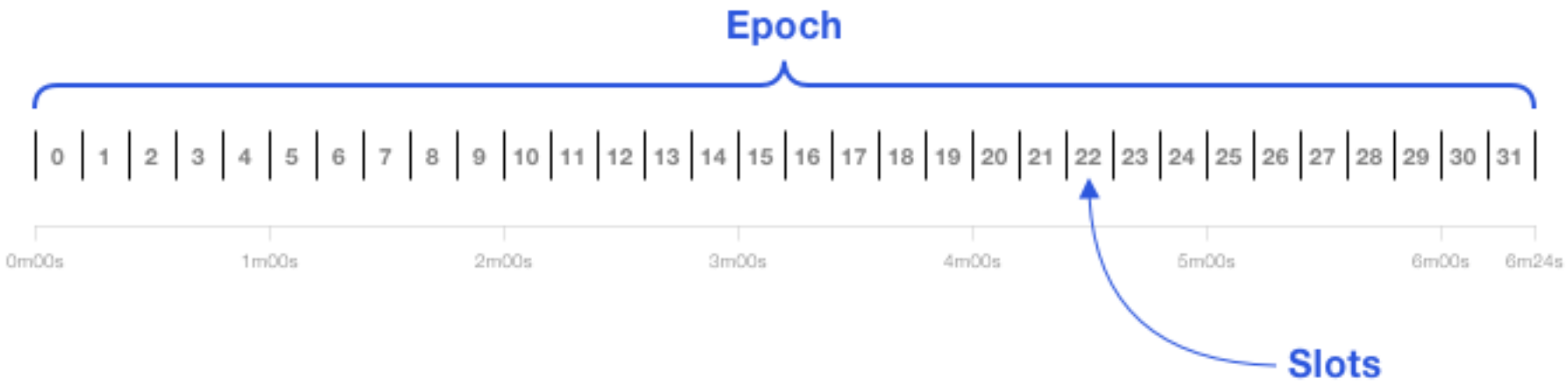
- slots (12 seconds)
- epochs (32 slots, 6.4 minutes)

Slots are the heartbeat of the Ethereum blockchain.

A slot is a chance for a block to be added to the Beacon Chain.

A slot can be empty. I.E. no block is added at that slot.

Slots and epochs progress regularly and relentlessly, whatever else may be happening on the network.



# Proposer

A block proposer is a validator selected to build a block.

Proposers are selected pseudo-randomly by RANDAO with a weighting on the validator's balance.

Every slot, exactly one validator is selected to propose a block.

The proposer shares its block with the whole network via a gossip protocol.

A slot can be empty: a block proposer might be offline, or propose an invalid block, or have its block subsequently reorged out of the chain.

# Validators and attestations

**Every epoch**, every validator gets to share its view of the world exactly once, in the form of an ***attestation***.

An attestation is a validator's vote, weighted by the validator's balance.

Attestations are broadcasted by validators in addition to blocks.

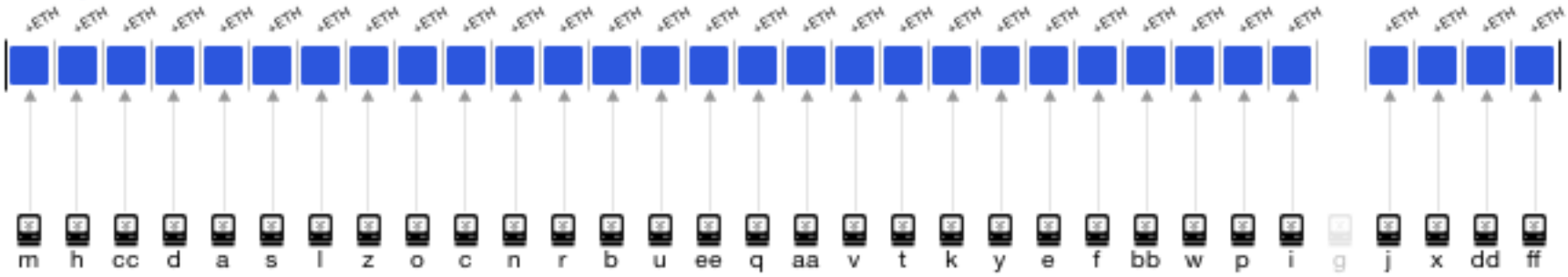
An attestation contains votes for the **head** of the chain that will be used by the **LMD GHOST** protocol, and votes for checkpoints that will be used by the **Casper FFG** protocol.

Like blocks, attestations can be missing for all sorts of reasons, and the protocol can tolerate this to various extents

# Block proposer

A random Validator proposes a Block for each Slot.

The Validator receives a reward if other Validators vote for this proposed Block.




Slots may be missing a Block.

This happens when the assigned Validator did not propose a Block, for reasons such as being offline or out of sync with the rest of the network.

The Validator misses out on the reward.

# Committee

In every slot, a committee of validators is randomly chosen, whose votes are used to determine the validity of the block being proposed. A **committee** is a group of validators.

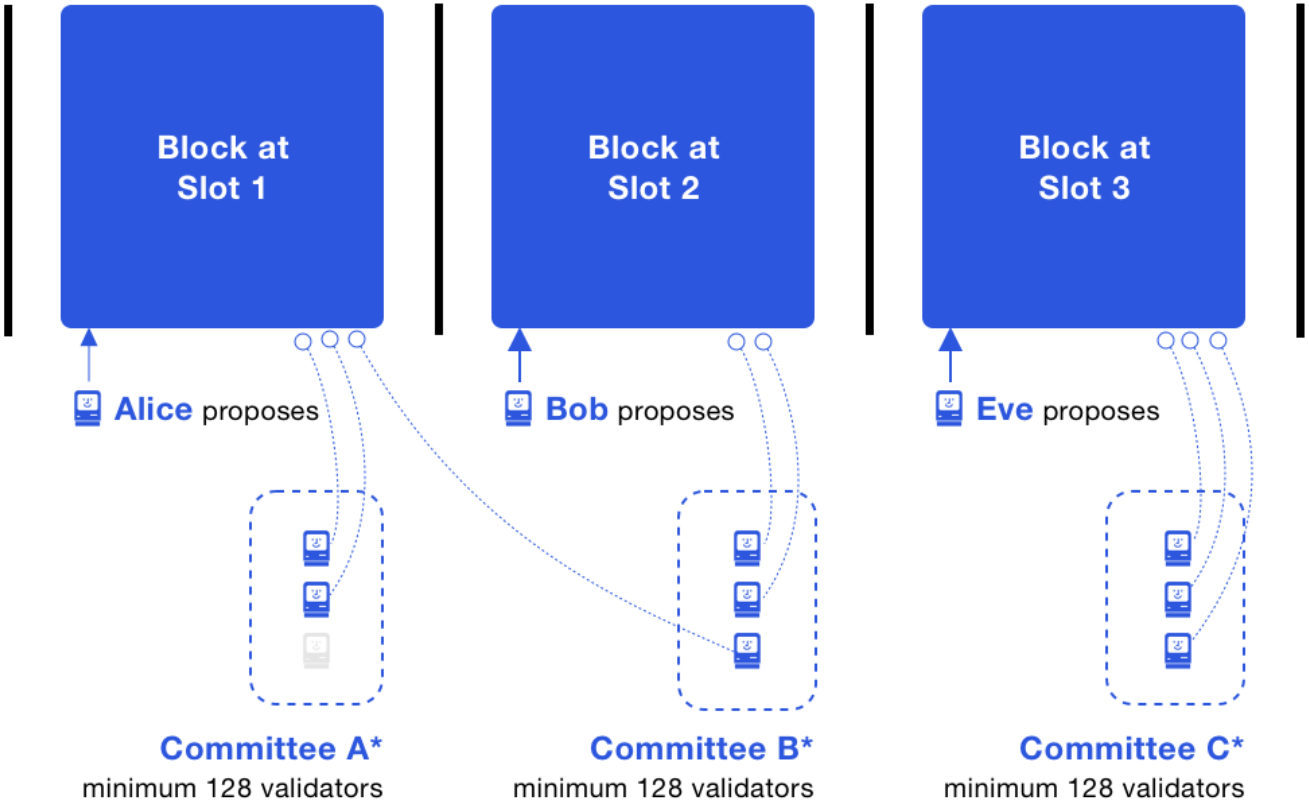
Each slot has committees of at least 128 validators.

Dividing the validator set up into committees is important for keeping the network load manageable.

Committees divide up the validator set so that every active validator attests in every epoch, but not in every slot.

There are 1 committee per slot, If the the network is made of less than 8'192 validators. Otherwise there would be at least two committees per slot.

# Committee



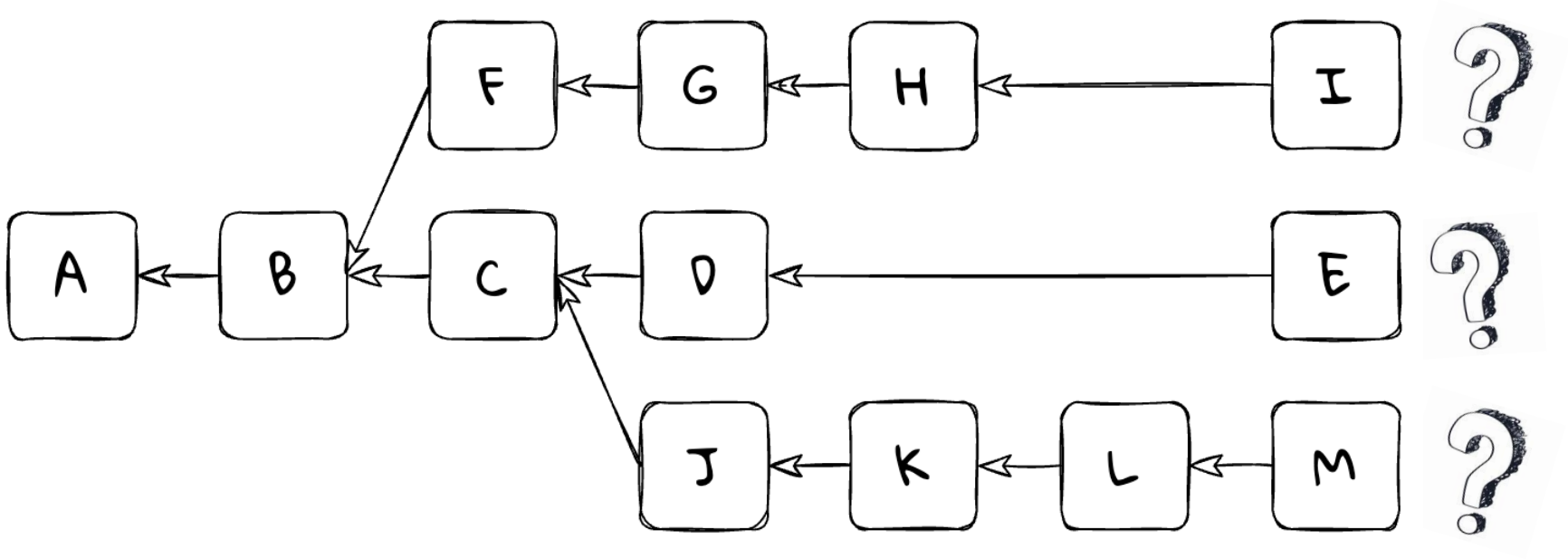

**Validators in the committees are supposed to attest to what they believe the head of the blockchain is**

*Note there can be more than one committee per slot.

# LMD GHOST

LMD : "Latest Message Driven”
GHOST: "Greedy Heaviest-Observed Sub-Tree”

It is it is a fork choice rule.



# LMD GHOST

Given a tree of blocks and a collection of votes, LMD GHOST will tell which block should be considered to be the best head.

Thereby as a result LMD GHOST provides a linear view of history from that head all the way back to genesis

## LMD

Nodes receive attestations both directly, via attestation gossip, and indirectly, contained in blocks.

On receiving an attestation, the node performs some basic validity checks on the attestation:

- It must be from the current or previous epoch.
- It must not be related to a future slot
- Validators sign attestations and are accountable for them.
- Ignore attestations that conflict with other attestations.

After passing these and some other checks, the attestation is considered for insertion and the view of the node is updated accordingly.

If the node don't already have a head block vote for the validator, then this one is stored for it. If we have a head block vote for the validator, then this one replaces it if it is more recent. I.E. only the last vote for each validator is stored.

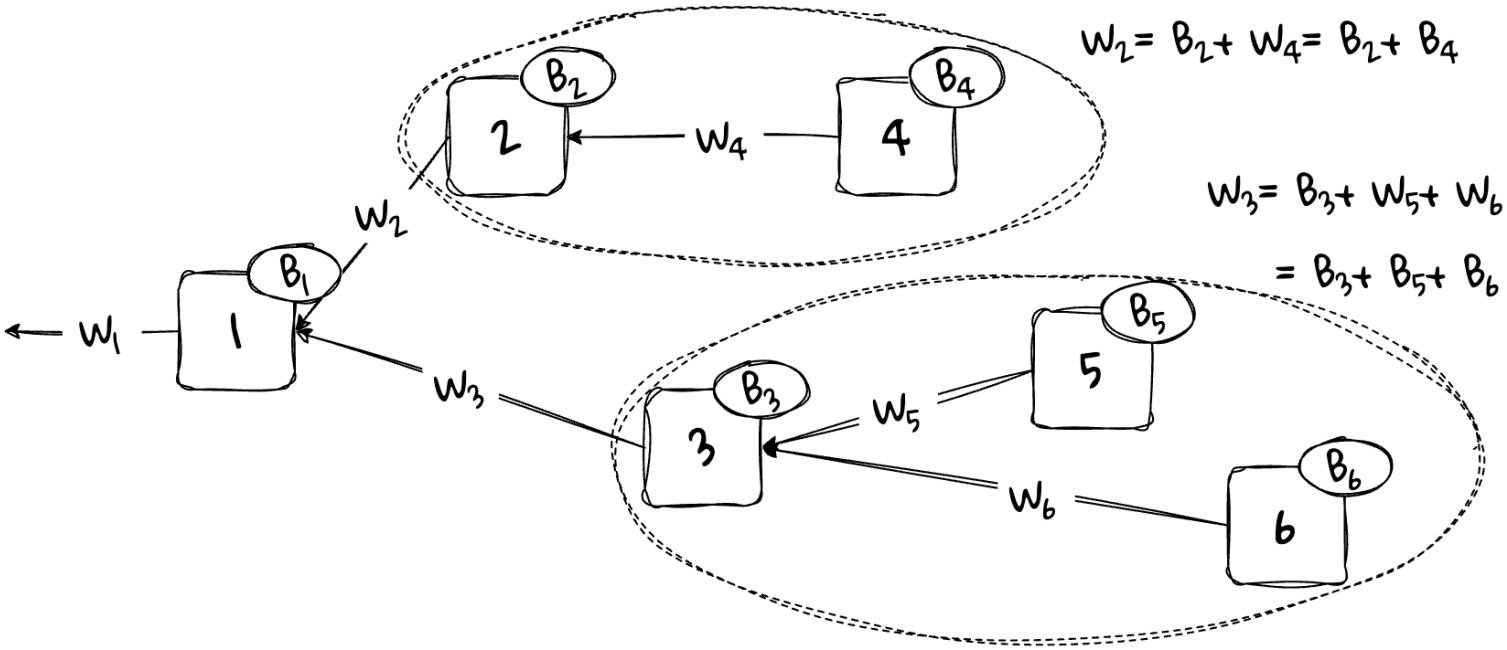
Once it is in the store it remains there indefinitely, and continues to contribute to the fork choice until it is updated with a more recent vote

# LMD GHOST

Thus, the view of a node is made of:

- The block tree, which is just a list of blocks. The blocks' parent links join them logically into a tree.
- The list of latest messages (votes) from validators.
- The validators' **effective balances** (based on some state) as these provide the weights used in the algorithm.

The weight of a branch is the weight of the votes for the block that roots it, plus the weights of that block's child branches. B.



# LMD GHOST

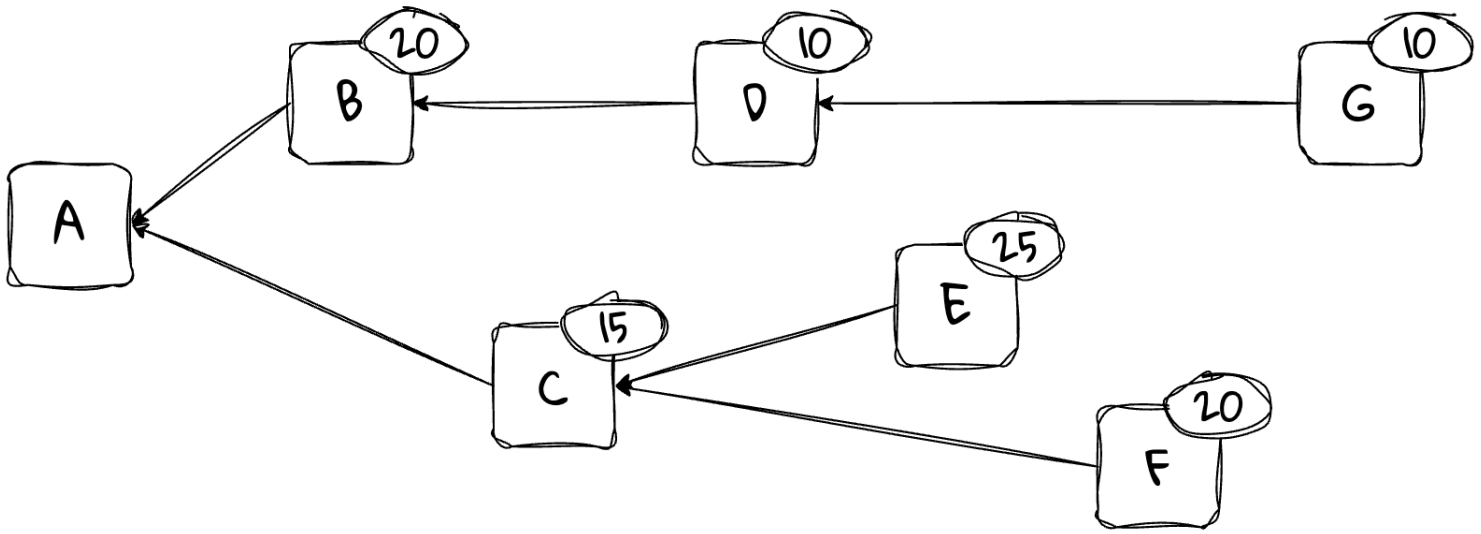
Given a block, we select the heaviest branch descending from it.

If a block has only one child, then the choice is obvious and there is no work to do.

If two or more branches have equal weight, we arbitrarily choose the branch rooted at the child block with the highest block hash value.

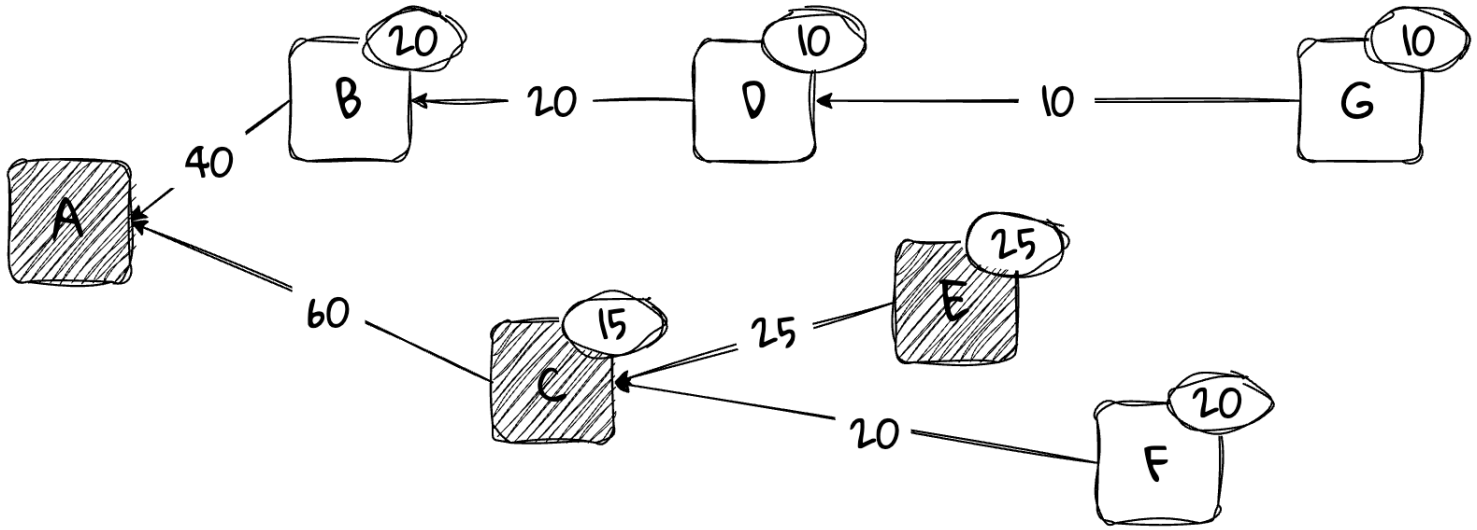
Eventually we will reach a block with no children.

This is a leaf block and will be the output of the algorithm.



# LMD GHOST

E is the output.



# CASPER FFG

Recall that **finality** is the property that there are blocks in the chain that are guaranteed never to be reverted: they will be part of the chain forever.

LMD GHOST does not provide finality, indeed there is always a chance that validators might decide to build a competing chain, and there is no real penalty for doing so.

Casper FFG functions as a "finality gadget", and we use it to add finality to LMD GHOST.

We operate on an asynchronous network, the Internet, which means that we can tolerate at most $\frac{1}{3}$ of the validators being adversarial (or faulty) if we want to achieve both safety and liveness.

.

# Why $^1/_3$

**The goal is to use vote counting to judge when we have seen a majority of the votes of honest validators.**

**N** *validators*

**f** *faulty or adversarial*

To preserve liveness, we need to be able to make a decision after hearing from only **n− f** validators, since the f faulty ones might withhold their votes.

However, the **f** non-responders may simply be delayed, and not faulty at all.

Therefore, we must account for up to f of the n−f responses we have received being from faulty or adversarial validators.

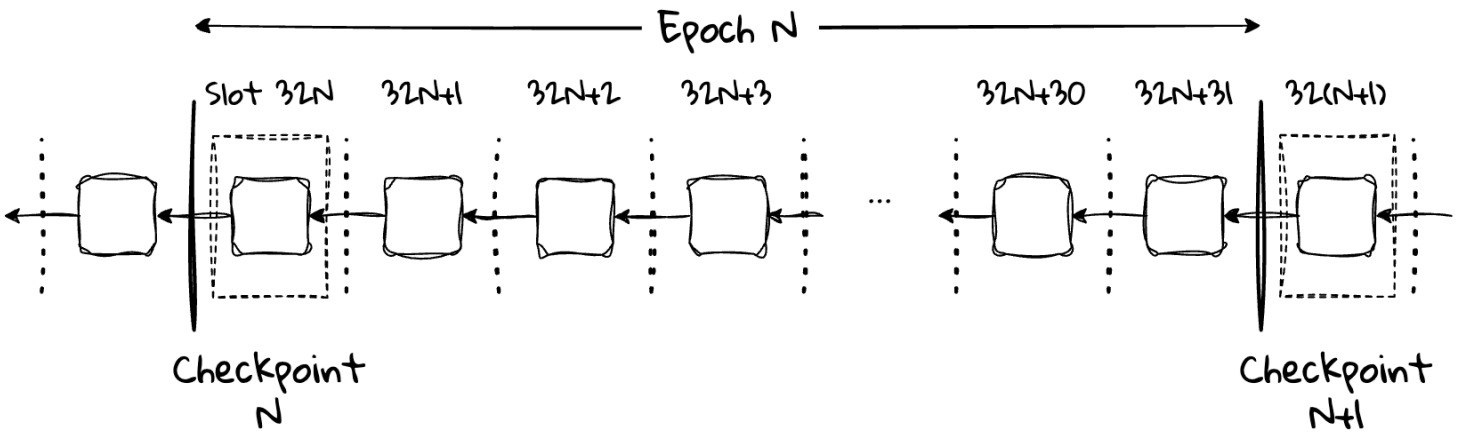
To guarantee that we can always achieve a **simple majority of honest validators** after hearing from n−f validators, we require that

$\frac{(n-f)}{2} > f$. That is $n > 3f$.

That means that we can support at maximum $\frac{1}{3}n$ faulty nodes.

# Epochs and Checkpoint

A checkpoint is a block in the first slot of an epoch.



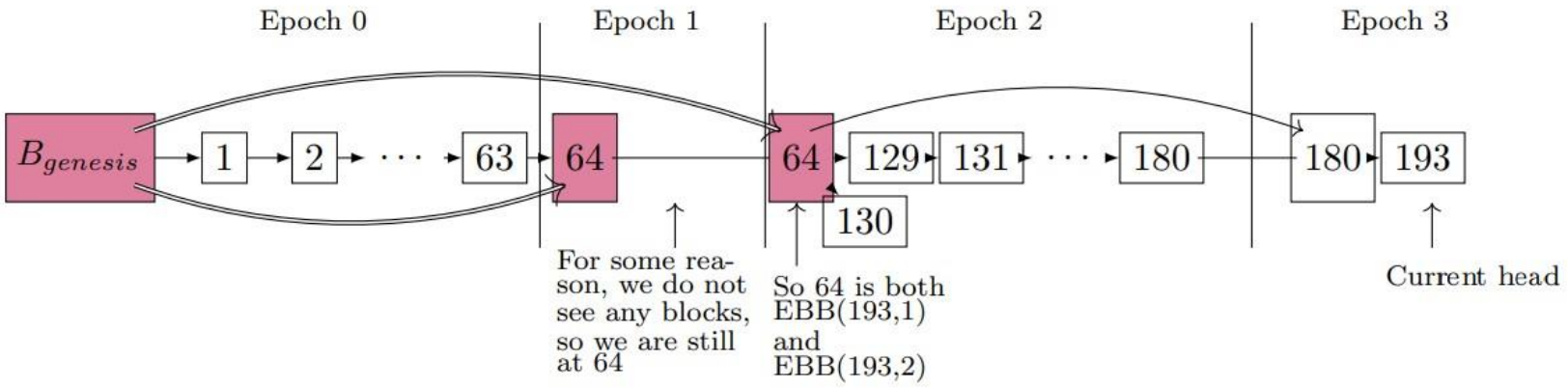

At every epoch, all the validators vote for a checkpoint.

# Epoch and Checkpoint

However, there is not a block for every slot.

So:
A checkpoint is a block in the first slot of an epoch.

- If there is no such block, then the checkpoint is the preceding most recent block.
- There is always one checkpoint block per epoch.
- A block can be the checkpoint for multiple epochs.



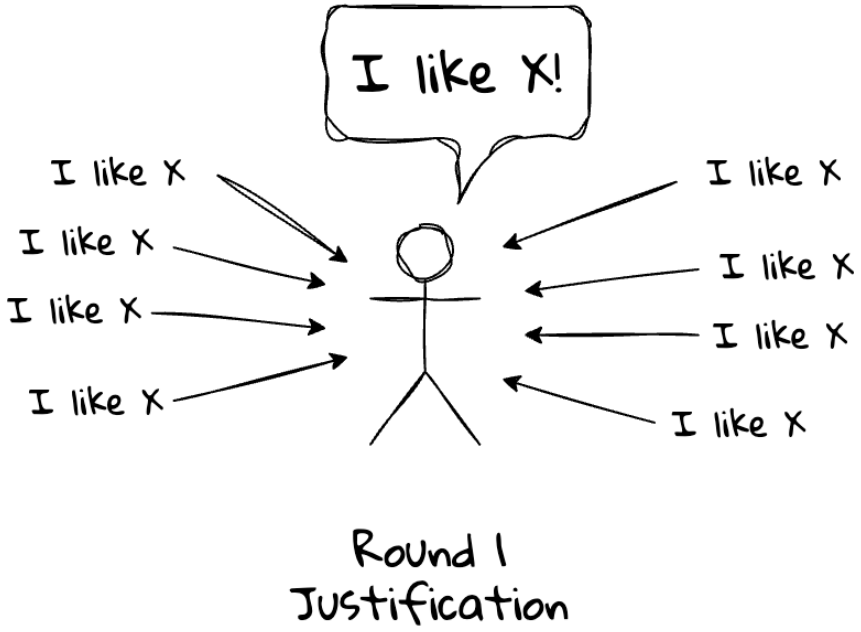

In the figure, Slot 65 to Slot 128 are empty. The Epoch 2 checkpoint would have been the block at Slot 128. Since the slot is missing, the Epoch 2 checkpoint is the previous block at Slot 64.

Epoch 3 is similar: Slot 192 is empty, thus the previous block at Slot 180 is the Epoch 3 checkpoint.

# Justification

Round 1 (ideally leading to justification):

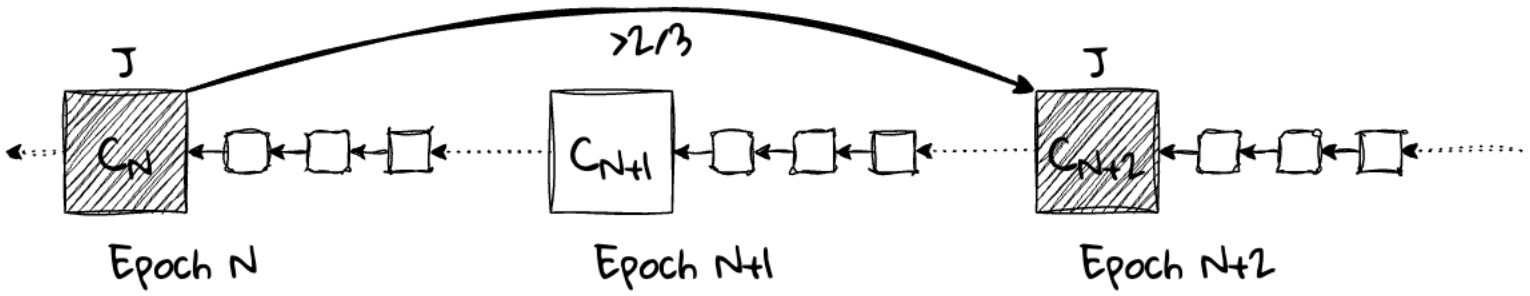
- I tell the network what I think is the best checkpoint.
- I hear from the network what all the other validators think is the best checkpoint.
- If I hear that $\frac{2}{3}$ of the validators agree with me, I justify the checkpoint.




Justification means that the node have seen commitments from over $\frac{2}{3}$ of the validator set that they will not revert checkpoint $C_2$ as long as they get confirmation from at least $\frac{2}{3}$ of validators that also will not revert $C_2$.

# Justification

When a node sees a supermajority link from justified checkpoint $C_1$ to checkpoint $C_2$, then it considers checkpoint $C_2$ to be justified.

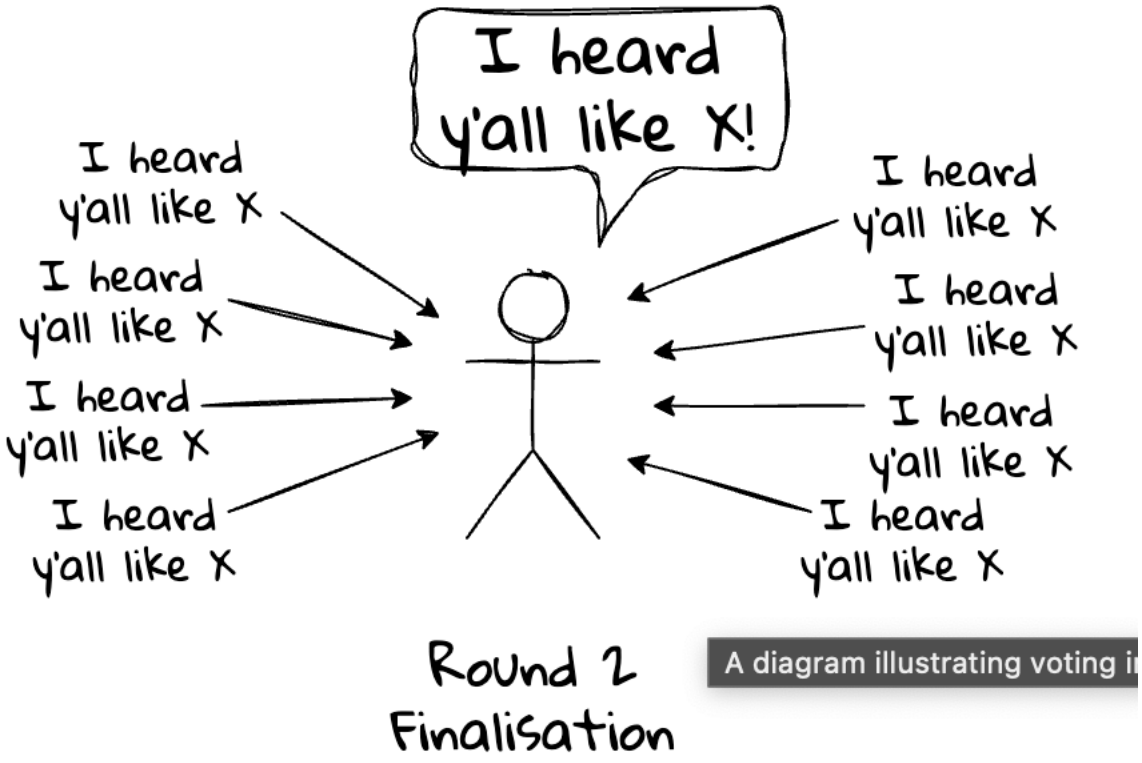



Justification means that the node have seen commitments from over $\frac{2}{3}$ of the validator set that they will not revert checkpoint $C_2$ as long as they get confirmation from at least $\frac{2}{3}$ of validators that also will not revert checkpoint $C_2$..

# Finalization

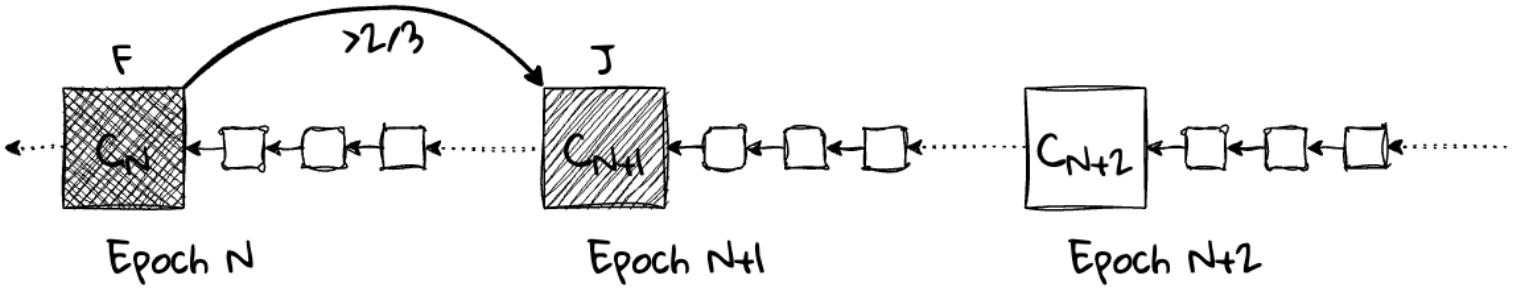
Round 2 (ideally leading to finalization):

- I tell the network my justified checkpoint, the collective view I gained from round 1.
- I hear from the network what all the other validators think the collective view is, their justified checkpoints.
- If I hear that $\frac{2}{3}$ of the validators agree with me, I finalize the checkpoint.



# Finalization

When a node sees a supermajority link from justified checkpoint $c_1$ to checkpoint $c_2$, and checkpoint $c_2$ is a direct child checkpoint of $c_1$, then it considers checkpoint $c_1$ to be ***finalised***. In other words, a finalised checkpoint is a justified checkpoint whose direct child is justified.




Finalization means that I have seen confirmation from over $\frac{2}{3}$ of the validators that they have seen commitments from over $\frac{2}{3}$ of the validators that they will not revert checkpoint $c_1$. Checkpoint $c_1$ cannot now be reverted without at least $\frac{1}{3}$ of validators provably changing their minds, and therefore getting **slashed**.

# Attestation

```
class AttestationData(Container):
    slot: Slot
    index: CommitteeIndex
    # LMD GHOST vote
    beacon_block_root: Root
    # FFG vote
    source: Checkpoint
    target: Checkpoint
```

# Finalization

The role of the **target** vote is to broadcast the view of what the the node thinks should be the next checkpoint to be justified. It is a **soft (conditional) commitment** not to revert that checkpoint as long as I hear from $\frac{2}{3}$ of validators that they also commit to that checkpoint.

The role of the **source** vote is to broadcast that the node have seen support from $\frac{2}{3}$ of the network for checkpoint , and that it is the most recent such checkpoint that the node knows about. By making this source vote the node upgrades its soft commitment not to revert the checkpoint to a **hard (unconditional) commitment** never to revert it.

# Supermajority

A link is a Casper FFG vote pair that links the source and target checkpoints, **s→t**.

A link s→t is a supermajority link when over $\frac{2}{3}$ of the validators (weighted by stake) have published the same link (and had their votes included in **blocks** in a timely way).

# GASPER

Casper FFG's role is to finalise a checkpoint.
History prior to the finalised checkpoint is a linear chain of blocks with all branches **pruned away**.
LMD GHOST is used to select the best head block at any time. LMD GHOST is constrained by Casper FFG in that it operates on the block tree only after the finalized checkpoint.