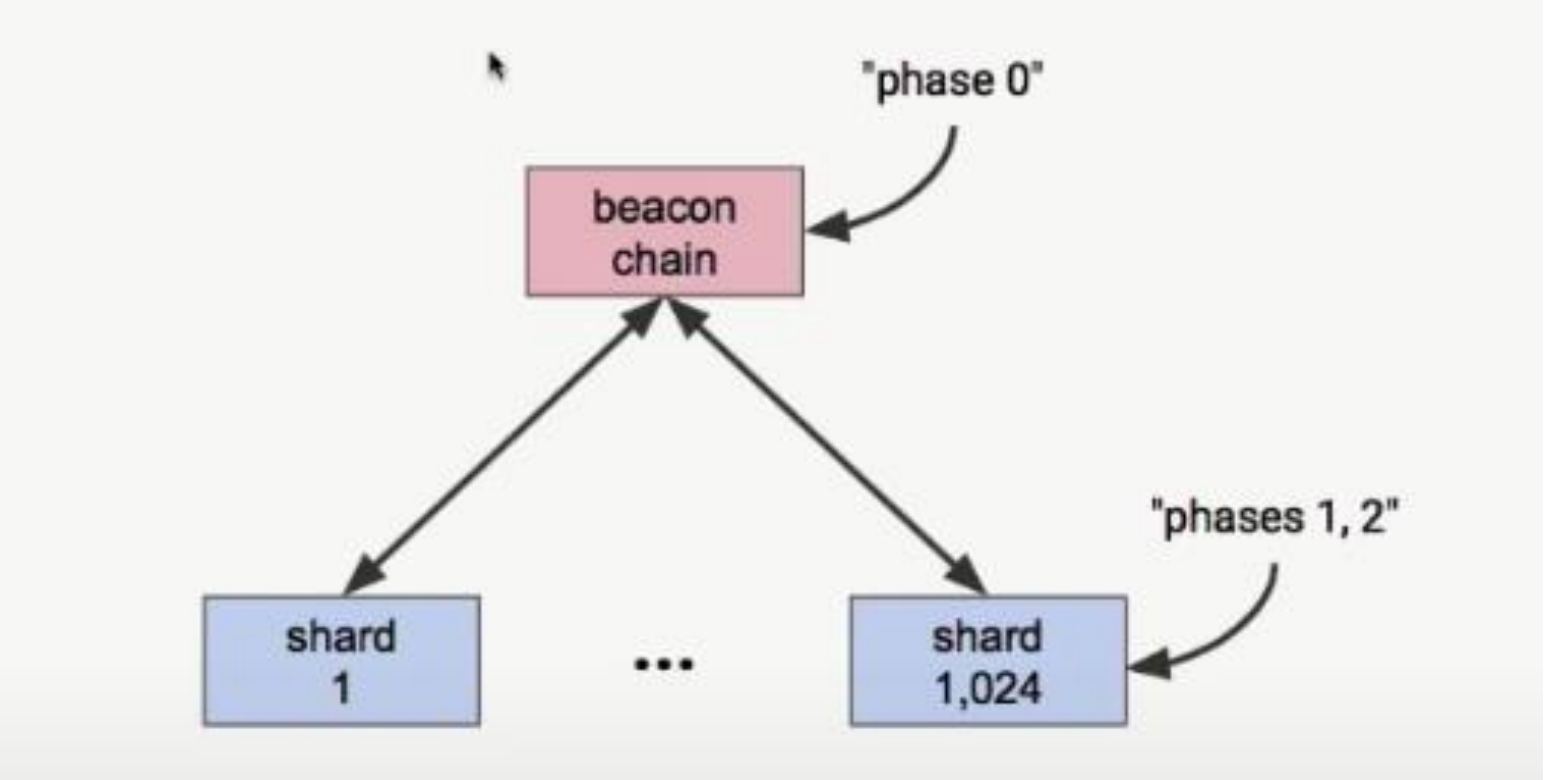
BEACON CHAIN

INTRODUCTION ON STATE OBJECT

BY ~ DEVRAJ SINGH

HUB AND SPOKE MODEL

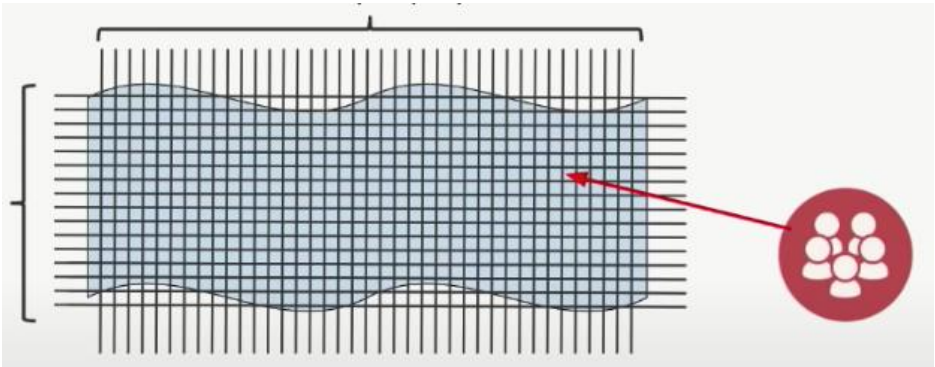


BEACON CHAIN

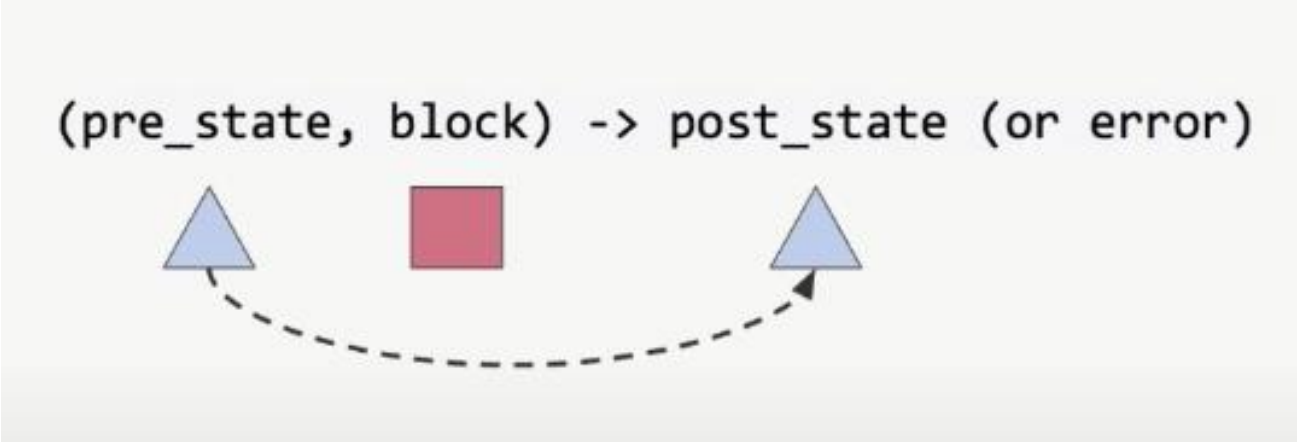
- Works more like a Voting Machine
- It is the place where we Organize the Voters (Validators) and Process the votes called attestations
- Rewards and Penalties are applied at this layer
- It does not have any user Transaction

BASICS ABOUT BEACON CHAIN

- Slots are Discrete time for 12 sec each
- Each Slot can generate one Block
- One Block Proposer is elected for a slot, It is possible that there is not block produced in any slot
  - 32 Slots - 1 epoch
  - 64 committees per slot
  - 32 * 64 = 2048 committees
  - One committee per Shard - One Committee is assigned a task
  - Each Single validator will get one vote per Epoch
- Is that mean one committee has more than one validator and one of them will be selected for block production in that slot?
- One Committee will have minimum 128 validator



STATE TRANSITION



- Most part of the spec talks about state transition
- State transition invloves with state object and block object

BASIC TYPES OF COMPONENT TO BUILD OBJECT

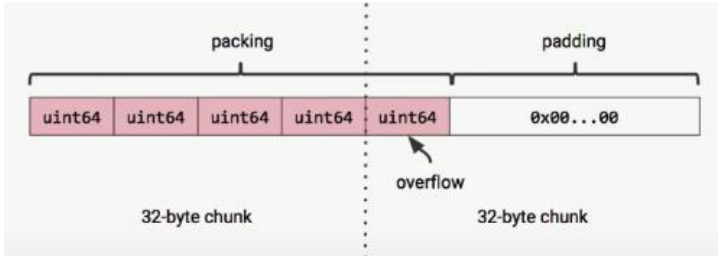
- Uint64 (8 bytes) - Serialization 8 bytes (little endian)
  Ex :- Slot, Epoch (32 slots), Shard, Index (validator index), Amount, Timestamp
- Byte - Ex:- Hash, State roots (32 bytes), Pub Key (48 bytes), Signatures (96 bytes), Bitfield
- Bool - Flag

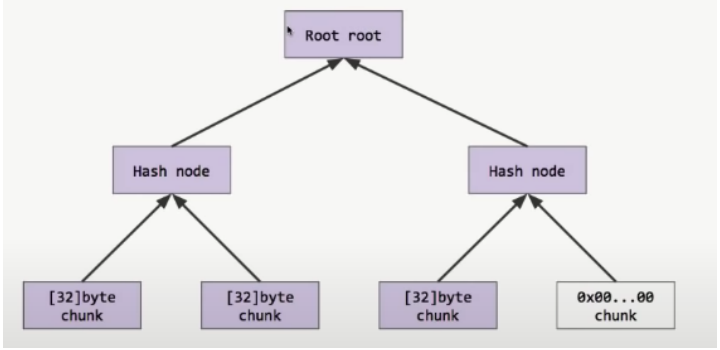
COMPOSITE TYPE

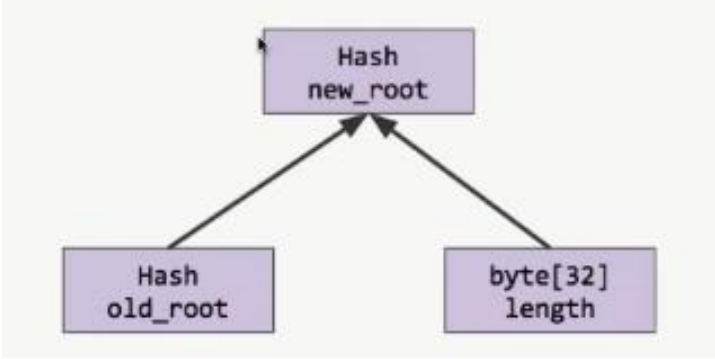
- Containers - Heterogeneous Object

```
{
    slot                Slot (8 bytes) ,
    prev_b_root         Root (32 bytes),
    state_root          Root,
    block_body_root     Root,
    signature           Signature (96 bytes),
}
```

- Tuples - fixed array. [uint64, uint64, uint64, uint64, uint64]]
- Lists - variable array [uint64, uint64, uint64 .....]







OBJECT MARKETISATION

- Simple Serialized Spec for Client communication
- Basic Objects, basic tuples - Pack them into 32 byte Chunks and Merkleize
- basic list - Pack them into 32 byte Chunks and Merkleize, Mix the Length
- containers, composite tuples - Recursive Merkleize, Merkleize root

STATE OBJECT

VERSIONING

- genesis_time type uint64 : Burner contract acts as bridge between Eth1 and Beacon chain, receipts of deposit receives by this contract is consumed by Eth2 chain and once we have enough of these receipts then the staking process could be started. So the Genesis can be initiated after 24 hours of receiving 2 million ether deposit receipts (Just to make sure to have enough confirmation & a clean genesis start)
- slot type Slot (uint64) : duration of slot is 6 sec but could vary from 5 to 7 due to leap second in Unix time, Epoch = slots / 64
- fork type Fork

{

prev_version uint64,

current_version uint64,

epoch Epoch(uint64)

} : This field of to provide a native support for hard forking

ROOT

- latest_block_roots type [8192] Root (32 bytes) : This is expose to shards, calculated whenever a block is generated
- latest_state_root type [8192] Root : calculated at every slot [8192] Root
- historical_roots type [] Root : Merkleise block root and stateroot for last Every 8000 slots, so a new value is added only after 8k slots, with this we can prove that some block root or state root was a given value in the past?
- latest_block_header type BlockHeader

{

slot Slot (8 bytes) ,

Prev_b_root Root (32 bytes),

State_root Root,

Block_body_root Root,

Sig Signature (96 bytes),

}

ETH1

- Eth1 chain is used to bootstrap Eth2 chain and eth2 chain contain data about Eth1 chain in this section.
- latest_eth1_data Eth1Data

{ deposit_root type Root : Merkle root of all the deposits up to the point in the deposit contract

block_hash type Root } : Recent Block hash in Eth1 Chain

- Eth1_data_votes []Eth1DataVote { eth1_data type Eth1Data, vote_count type Unit64 }
- When ever a new Eth1 data wants to persist on Eth2 chain, Eth2 selects a committe from block proposer and when more than 50% vote for Eth1 data then it is accepted.

REGISTRY

- Validator_registry type []Validator

{

pubkey type PubKey,

withdrawal_credentials type Root,

Every validator will have two pubkey, one which is his day to day siging key and another is for used for withdraw money from the system, withdrawal_credentials is the has of that key.these two fields are specified in the deposit transaction which we do at Eth1 to become a validator.

activation_epoch type Epoch, : Once deposit is >= 32 eth, validator is activated and activation should be >= Current epoch

exit_epoch type Epoch,

There are ways of Exit a) if your balance is significantly lower b) if you attests wrong blocks (Forced exit) c) voluntary exit, you can raise a voluntary exit message, This field also helps to check the active validators in the system, if you check active validator for a epoch the scan the whole registry and get the validators for whom correct epoch falls in between activation epoch and exit epoch.

withdrawal_epoch type Epoch,

voluntary_exit flag, : This flag is to make the exit validator push into exit queue and make sure that there should be a constant number of validator in system

slashed flag,

}}

REGISTRY

- validator_balances type []Amount

This field is intentionally kept out from validator structure as if you observe all of validator fields will not going to change in their life span But validator balance will keep changing every epoch due to micro payment/slashing, so it avoid hashing overhead

- validator_reg_update_epoch type Epoch
- latest_active_index_root type [8192] Root

This fields is mostly used for light client operation, in a certain time period system make a checkpoint for the active validator by voting process (voted by the last checkpoint active validators) and save the root of the result in this field, these checkpoint is useful for light clients

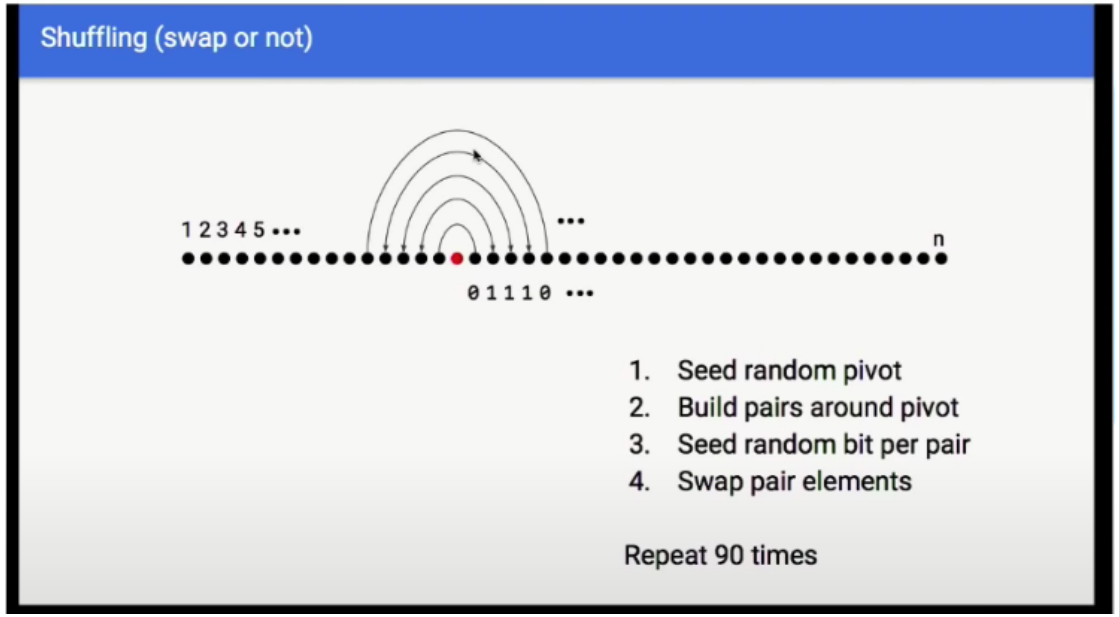
SHUFFLING

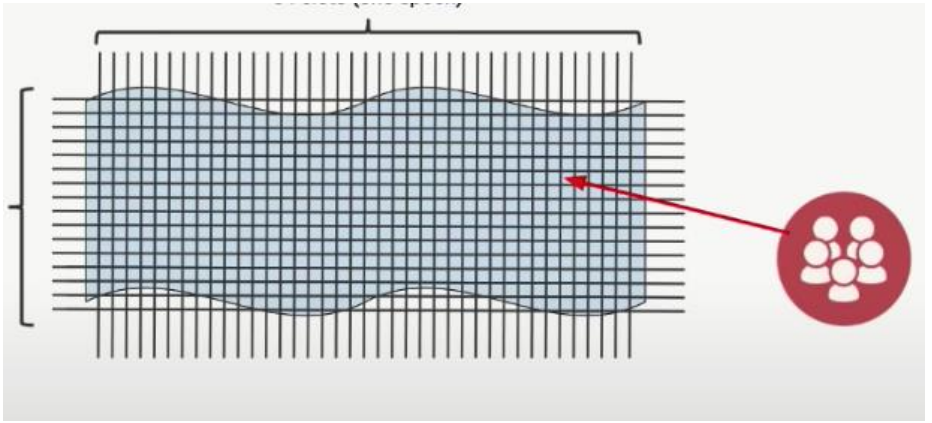
- This section is imp to maintain high level of security for beacon chain by shuffling the validator sets,
- It takes the validators + Seed (RanDOA Mix) Passed into Shuffle Function and spit Shuffled Validators
- This Shuffle Function should have two basic properties
  - It should provide Pseudo random permutation - This means every validator should assigned to one unique place
  - Locally computable - It we take a specific validator, and want to compute where it has been assigned to, without having to compute the permutation on the whole validator set.

- Suffling function
  1. Seed which is a random number on validator index line
  2. Build the pairs around that pivot
  3. Seed random bit for each pair (either 0 or 1). take the position of the bit and hash it with seed rB = Hash(Bindex + Seed)
  4. If the Seed bit is 1 then you swap the pair
  5. Repeat this process 90 times

- After getting the shuffled validators we can create the 1024 committee from this set which will be assigned to 1024 shards

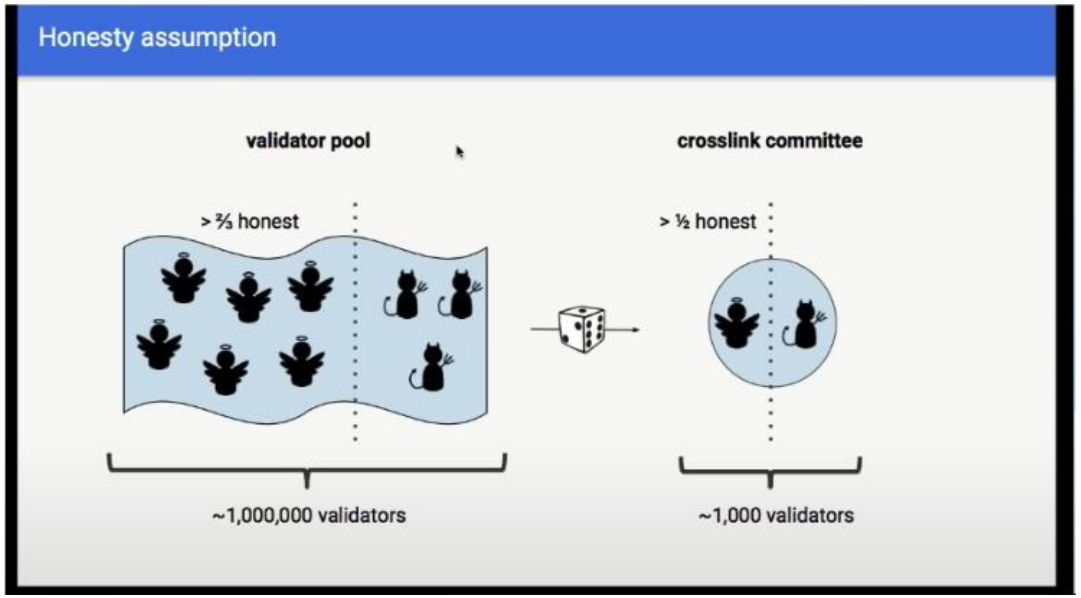
SHUFFLING FUNCTION EXAMPLE - (SWAP OR NOT)





- 64 Slots - 1 epoch
- 16 committees per slot
- 64 * 16 = 1024 committees
- One committee per Shard - One Committee is assigned a task
- Each Single validator will get one vote per Epoch

HONESTY ASSUMPTION



- 2/3 of the validator are honest?
- After shuffling function at least ½ of the selected validator are honest?

- I don’t understand how we have made the 1st and 2nd assumption

SHUFFLING

- latest_randao_mixes [8192]Bytes32

Below fields are used to recompute the set of active validators relative to two different shuffling from the recent past

- previous_shuffling_seed Bytes32
- previous_shuffling_epoch Epoch
- previous_shuffling_start_shard Shard
- current_shuffling_seed Bytes32
- current_shuffling_epoch Epoch
- current_shuffling_start_shard Shard

FINALITY

- This section gives us notion of economic finality .i.e. if something goes wrong like we have two cross-links (cross voting on two different block on same height) or we can say two finalized checkpoint which are not in same fork then we have a guarantee that at least 1/3 of active validators money will be burned
- Finality is done using attestations
- Here we have imp role of BLS signature which can aggregate multiple signature done by the validator who voted for same block and end result will be a single aggregated signature which can be validated as a whole. It is one of the imp notion in scalability.

FINALITY

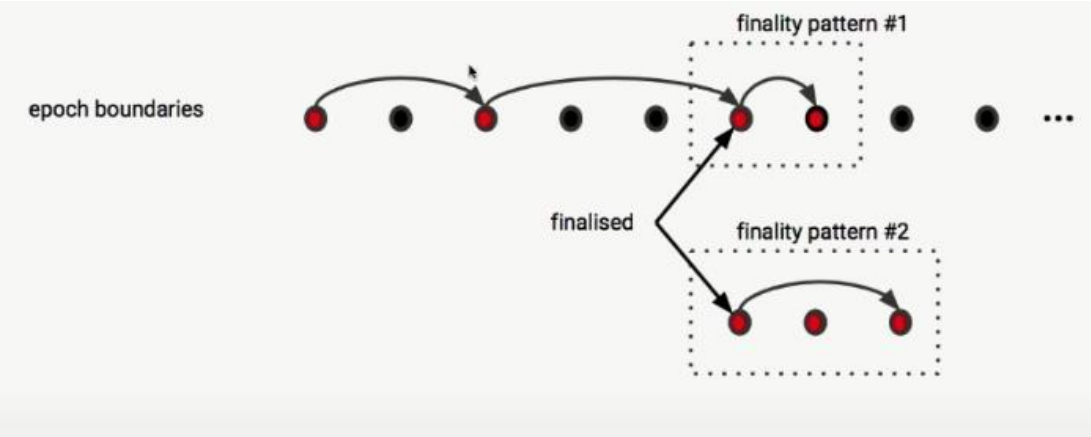
```
// Finality
previous_epoch_attestations  []PendingAttestation
current_epoch_attestations   []PendingAttestation
previous_justified_epoch     Epoch
current_justified_epoch      Epoch
justification_bitfield       uint64
finalized_epoch              Epoch
latest_crosslinks            [1024]Crosslink
```

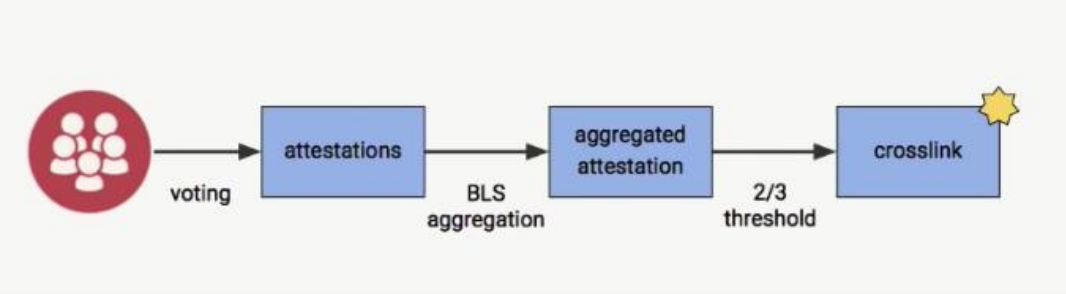
```
type PendingAttestation {
    aggregation_bitfield Bitfield
    data AttestationData
    custody_bitfield Bitfield
    inclusion_slot Slot
}
```

```
type AttestationData {
    // LMD GHOST vote
    slot         Slot
  ① block_root Root

    // FFG vote
    source_epoch
  ② source_root
  ③ target_root

    // Crosslink vote
    shard                Shard
    previous_crosslink   Crosslink
  ④ crosslink_data_root Root
}
```

- There are 3 kind of attestation (Sub Vote) a Validator does during the attestation process
  - The 1st one is for the single block root in the beacon chain, this is imp for the Fork choice rule, which assigns the weight to every single block
  - FFG Vote is used for finality
  - Crosslink votes are used for Sharding, crosslinks are the communication mechanism between the shard, Beacon chain is aware of every shard as a light Client, Crosslink are used a shard checkpoint that get included into the beacon chain
    - To become a crosslink, you need to have the committee assigned to a shard agree on the cross linked data root

FINALITY – CROSSLINKS





- Every shard is assigned to a small committee relative to total validator pool, they make the individual attestations
- These attestations are aggregated using BLS aggregation (If same)
- If we reach 2/3 threshold then it becomes a crosslink

PARTIAL SLASHING

- // Partial slashing
  - latest_slashed_balances [8192] Amount
  - This field keeps track of how many slashing events has been happened so that slashing can be done proportional

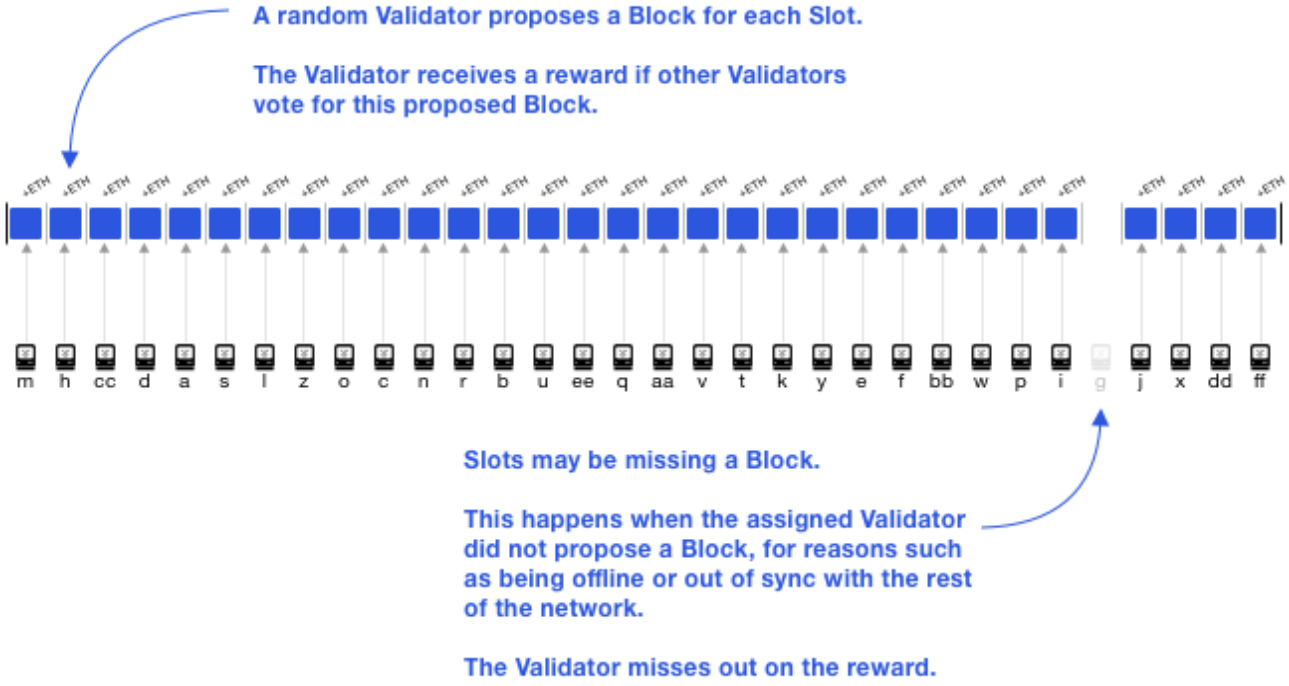
BLOCK OBJECT

EPOCH AND SLOTS

- A slot is a chance for a block to be added to the Beacon Chain and shards. You can imagine that the Beacon Chain and shard chains are choreographed in lockstep. Every 12 seconds, one beacon (chain) block and 64 shard blocks are added when the system is running optimally.

- At every epoch, a validator is pseudo randomly assigned to a slot and shard. The validator is participating in the consensus of that assigned shard so that it can vote for that shard's head. The validator links the shard head to the beacon block for a slot.
- An **attestation** is a validator's vote, weighted by the validator's balance. Attestations are broadcasted by validators in addition to blocks.

BLOCK PROPOSER



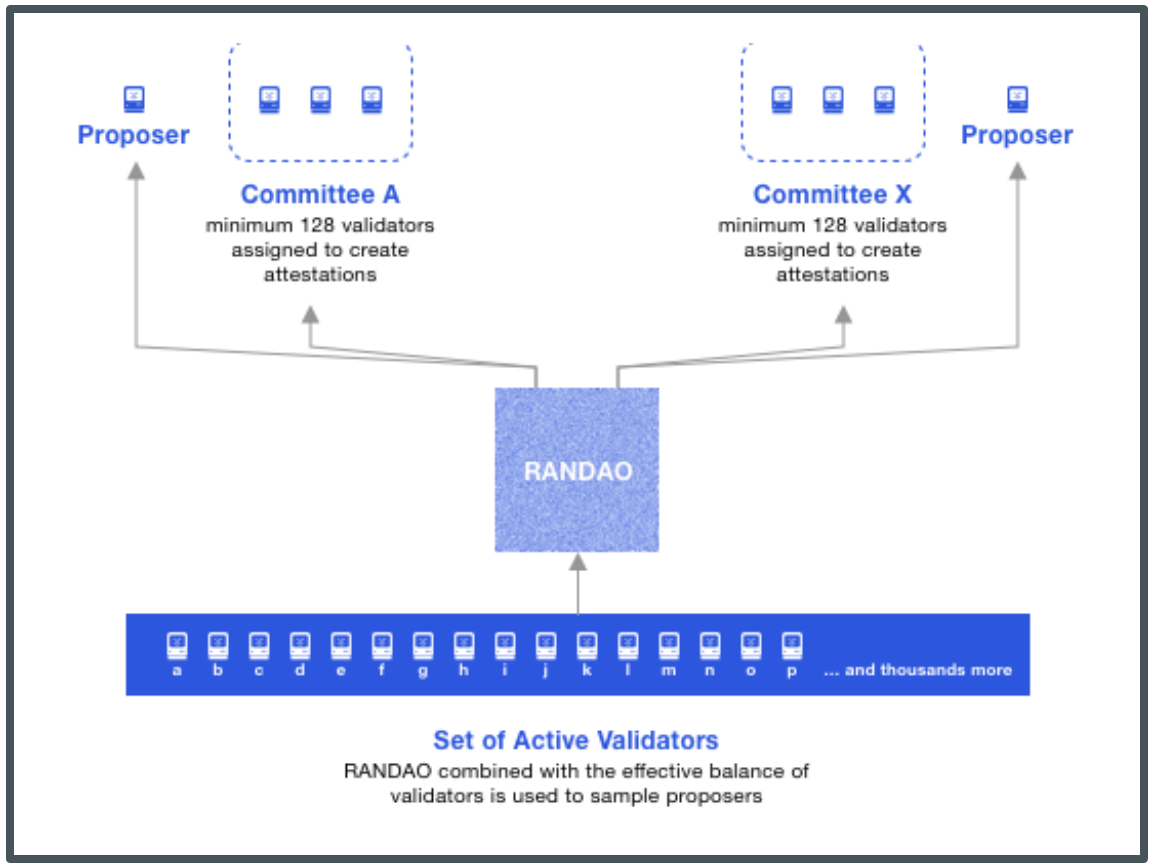
- A block **proposer** is a validator that has been pseudo randomly selected to build a block.
- Most of the time, validators are **attesters** that vote on beacon blocks and shard blocks. These votes are recorded in the Beacon Chain. The votes determine the head of the Beacon Chain, and the heads of shards.

STAKING VALIDATORS: SEMANTICS

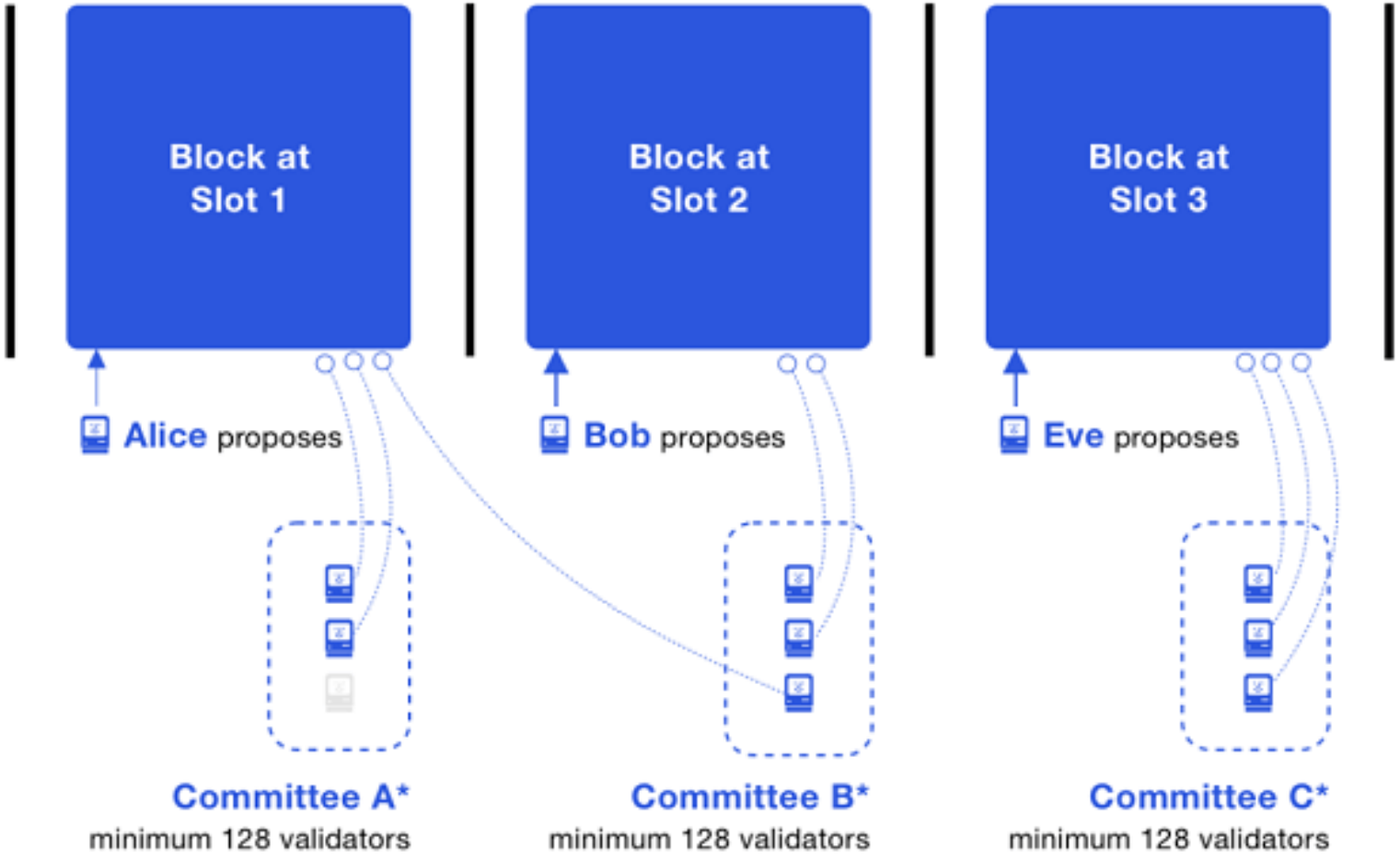
- Validators are virtual and are activated by stakers. In PoW, users buy hardware to become miners. In Ethereum 2.0, users stake ETH to activate and control validators.
- It is clearer to associate stakers with a stake, and validators with a balance. Each validator has a maximum balance of 32 ETH, but stakers can stake all their ETH. For every 32 ETH staked, one validator is activated.

- Validators are executed by ***validator clients*** that make use of a beacon (chain) node. A **beacon node** has the functionality of following and reading the Beacon Chain. A validator client can implement beacon node functionality or make calls into beacon nodes. One validator client can execute one or more validators.

COMMITTEES: INTRODUCTION



- A committee is a group of validators. For security, each slot (in the Beacon Chain and each shard) has committees of at least 128 validators. An attacker has less than a one in a trillion probability of controlling ⅔ of a committee
- At every epoch, a pseudorandom process RANDAO selects proposers for each slot, and shuffles validators to committees.



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.

The diagram is a combined depiction of what happened in three slots.

- In Slot 1, a block is proposed and then attested to by two validators; one validator in Committee A was offline. The attestations and block at Slot 1 propagate the network and reach many validators.
- In Slot 2, a block is proposed and a validator in Committee B does not see it, thus it attests that the Beacon Chain head is the block at Slot 1. Note this validator is different from the offline validator from Slot 1. **Attesting to the Beacon Chain head is called an LMD GHOST vote.**
- In Slot 3, all validators in Committee C run the LMD GHOST fork choice rule, and independently attest to the same head.
- A validator can only be in one committee per epoch. Typically, there are more than 8,192 validators: meaning more than one committee per slot. All committees are the same size, and have at least 128 validators.

CROSSLINKS: ROOTING SHARDS TO THE BEACON CHAIN

- A crosslink is a reference in a beacon block to a shard block. A crosslink is how the Beacon Chain follows the head of a shard chain.
- As there are 64 shards, each beacon block can contain up to 64 crosslinks (Shared Blocks).
- A beacon block might only have one crosslink, if at that slot, there were no proposed blocks for 63 of the shards.
- WHY Do We need Cross Links in a Beacon Block
  - Crosslinks are planned for eth2 Phase 1 to root the shard chains into the Beacon Chain,
  - Serving as the base of the shard fork choice
  - Shard chain finality (A crosslink by itself is insufficient to finalize a shard block, but contributes to the shard chain's fork choice. In other words, a shard block is finalized when it is crosslinked into a beacon block that is finalized.)
  - Cross shard communication.

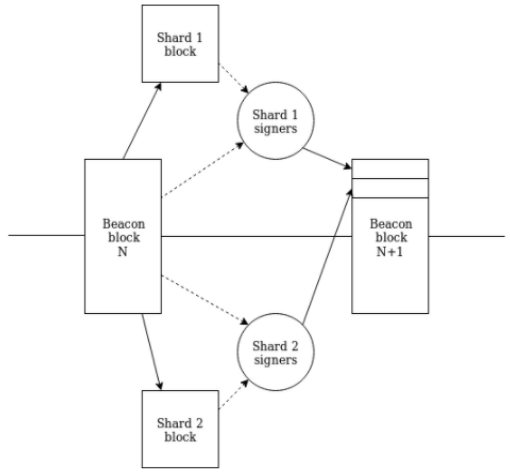
SHARDS AND CROSS LINK

We reduce SHARD_COUNT from 1024 to 64, and increase the maximum number of shards per slot from 16 to 64. This means that the “optimal” workflow is now that during every beacon chain block, there is a crosslink published for every shard (for clarity, let us retire the word “crosslink” as we’re not “linking” to a shard chain and use the word “shard block” directly)

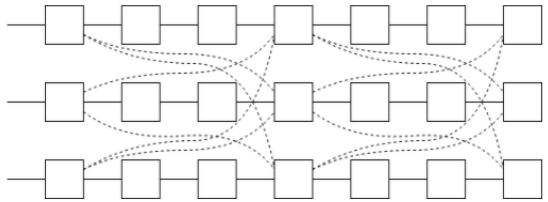
Notice a key detail: there is now a pathway by which a slot-N+1 block of *any* shard is aware of *all* slot-N blocks of *all* shards . Hence, we now have first-class single-slot cross-shard communication (through Merkle receipts).

Details

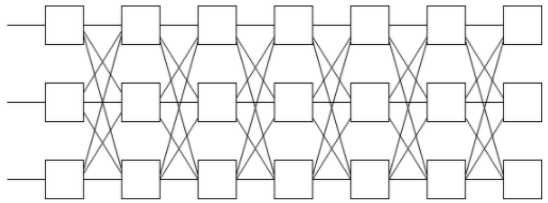
We reduce `SHARD_COUNT` from 1024 to 64, and increase the maximum number of shards per slot from 16 to 64. This means that the “optimal” workflow is now that during every beacon chain block, there is a crosslink published for every shard (for clarity, let us retire the word “crosslink” as we’re not “linking” to a shard chain and use the word “shard block” directly).



Notice a key detail: there is now a pathway by which a slot-N+1 block of *any* shard is aware of *all* slot-N blocks of *all* shards . Hence, we now have first-class single-slot cross-shard communication (through Merkle receipts).



Status quo (approximate)



Proposal

BEACON CHAIN AND SHARDS

We change the structure of what attestations link to: instead of containing a "crosslink" including a "data root" representing many shard blocks in some complex serialized form, it just includes a data root representing the contents of a single block

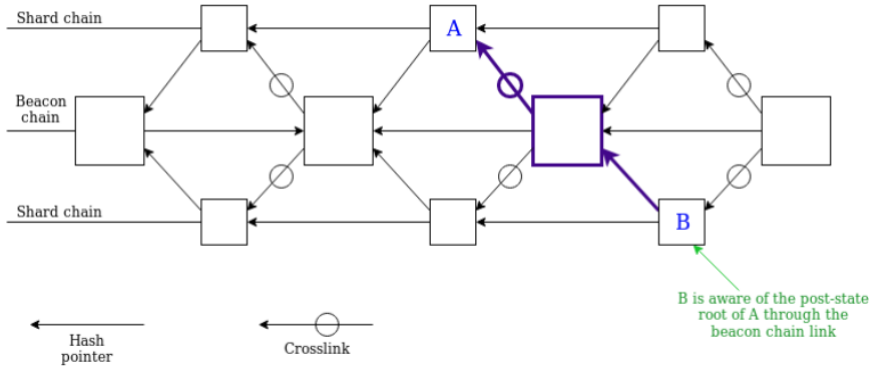
Shard blocks are still required to point to a "previous shard block" and we still enforce consistency, so the protocol requires such multi-slot attestations to be consistent. We expect committees to use the following "fork choice rule":

- For each valid+available shard block B (the block's ancestors must also be valid+available), compute the total weight of validators whose most recent messages support B or a descendant of B; call this the "score" of B. Empty shard blocks can also have scores.

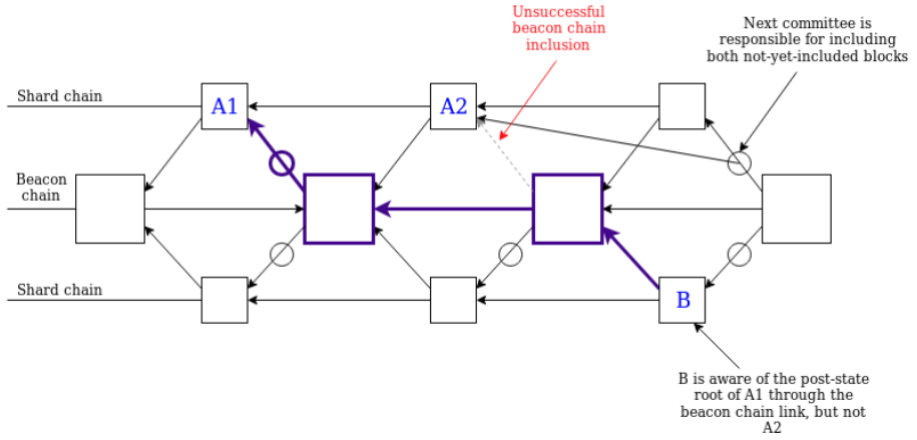
- Pick the shard block for latest_shard_blocks[i].slot + 1 with the highest score

- Pick the shard block for latest_shard_blocks[i].slot + k for k > 1 with the highest score, only considering blocks whose previous-block pointer points to the choice already made for latest_shard_blocks[i].slot + (k-1)

Best case



Exceptional case: failed beacon chain inclusion

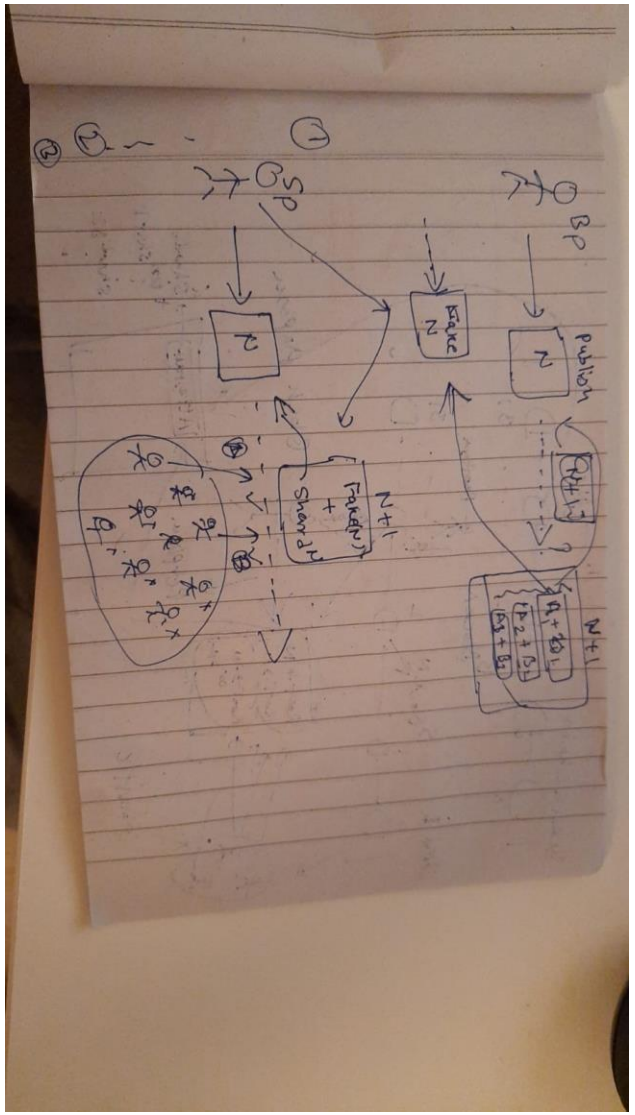


BLOCK PROGRESSION

Recap

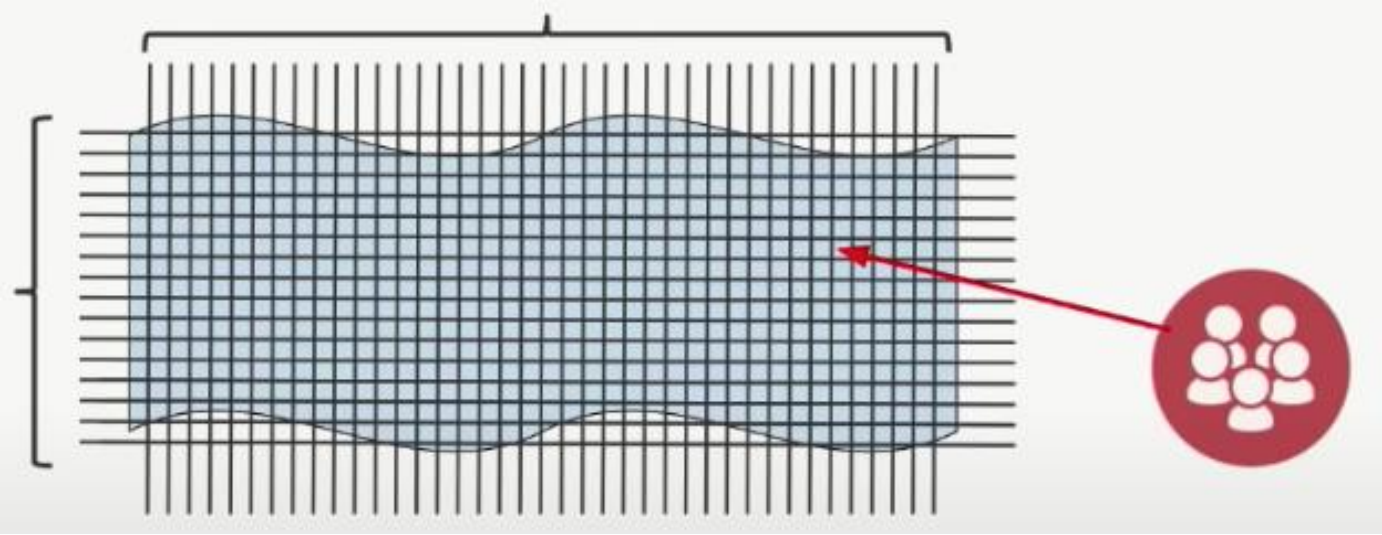
The process between publication of beacon block N and beacon block N+1 would look as follows:

1. Beacon block N is published
2. For any given shard `i`, the proposer of shard `i` proposes a shard block. The execution in this block can see the root of beacon block N and older blocks (if desired, we can reduce visibility to just block N-1 and older; this would allow beacon blocks and shard blocks to be proposed in parallel).
3. Attesters mapped to shard `i` make attestations, which include opinions about the slot N beacon block and the slot N shard block on shard `i` (and in exceptional cases also older shard blocks on shard `i`)
4. Beacon block N+1 is published, which includes these attestations for all shards. The state transition function of block N+1 processes these attestations, and updates the "latest states" of all shards.



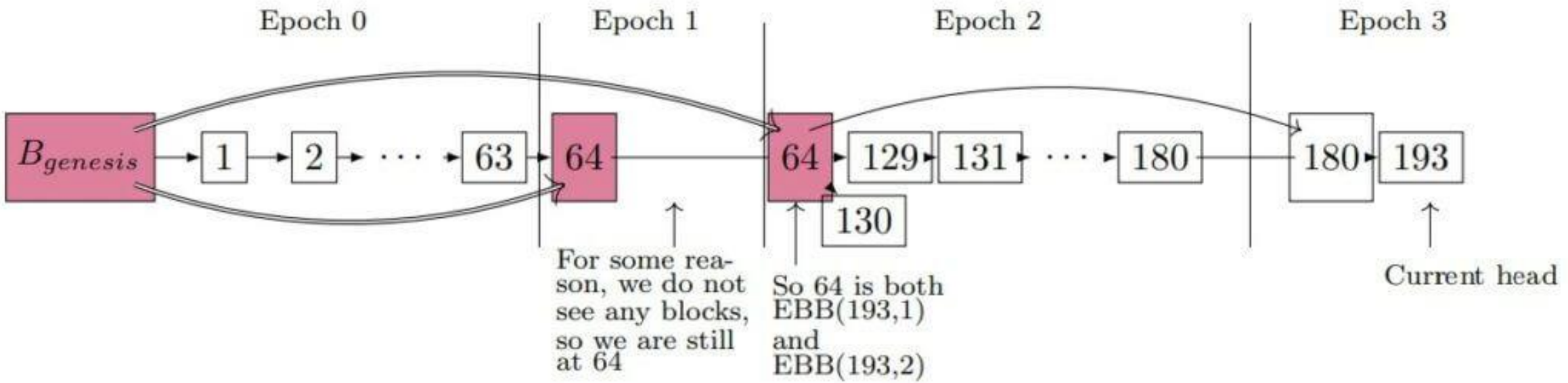
OVERHEAD ANALYSIS

- Note that no actors need to constantly be actively downloading shard block data. Instead, proposers only need to upload up to 512 kB in < 3 seconds when publishing their proposal (assuming 4m validators, each proposer will on average do this once every 128k slots), and then committee members only need to download up to 512 kB in < 3 seconds to validate a proposal (each validator will be called upon to do this once per epoch, as we are retaining the property that each validator is assigned to a particular crosslink in a particular slot in any given epoch).
- Note that this is lower than the current long-run per-validator load, which is ~2MB per epoch. However, the "burst" load is higher: up to 512 kB in < 3 seconds instead of up to 64 kB in < 3 seconds.



BEACON CHAIN CHECKPOINTS

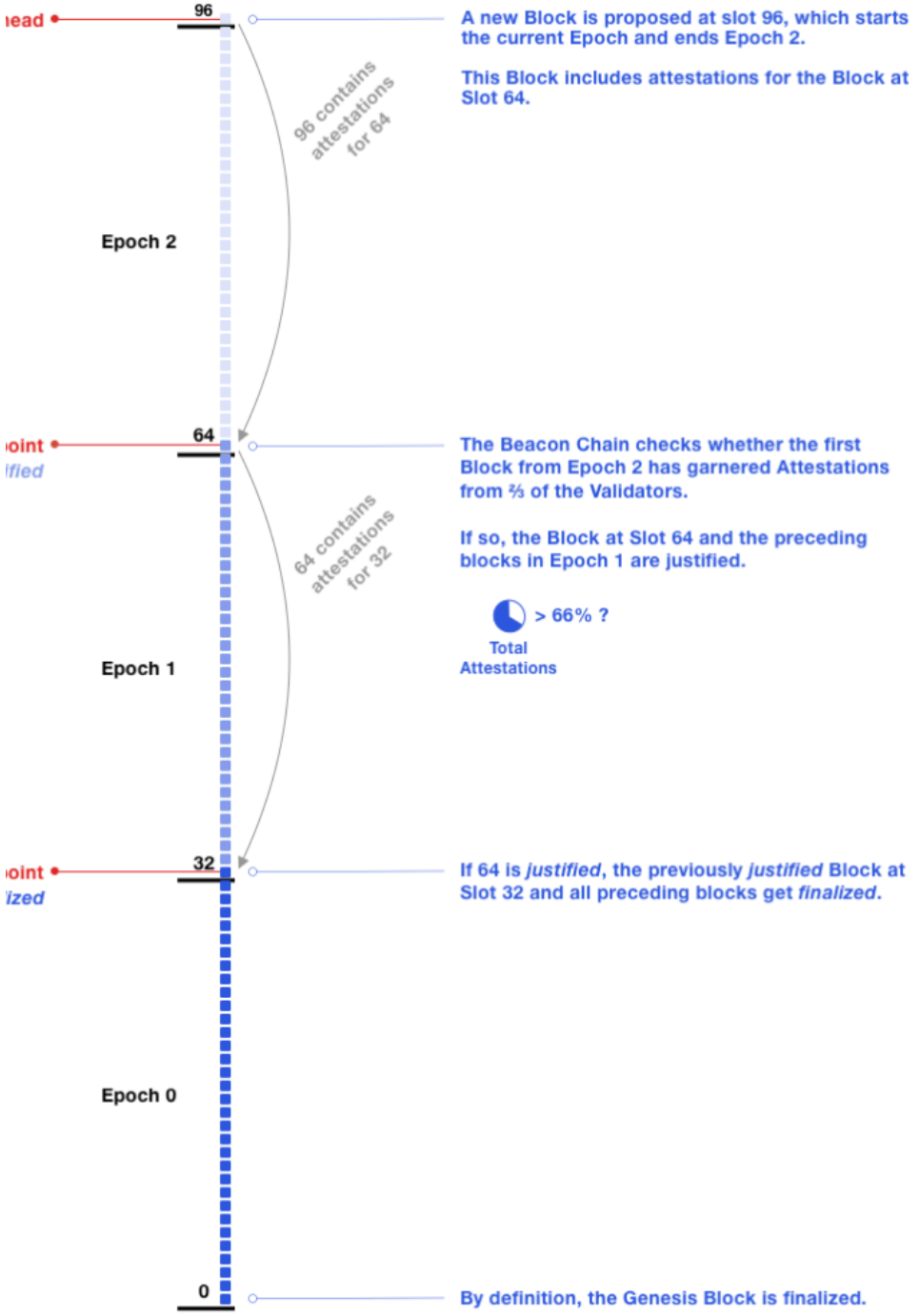
- A checkpoint is a block in the first slot of an epoch. If there is no such block, then the checkpoint is the preceding most recent block. There is always one checkpoint block per epoch. A block can be the checkpoint for multiple epochs.
- Note Slot 65 to Slot 128 are empty. The Epoch 2 checkpoint would 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.
- When casting an LMD GHOST vote, a validator also votes for the checkpoint in its current epoch, called the *target*. This vote is called a Casper **FFG vote**, and also includes a prior checkpoint, called the *source*. In the diagram, a validator in Epoch 1 voted for a source checkpoint of the genesis block, and a target checkpoint of the block at Slot 64. In Epoch 2, the validator voted for the same checkpoints.
- Only validators assigned to a slot cast an LMD GHOST vote for that slot. However, all validators cast FFG votes for each epoch checkpoint.



FINALITY

- A vote that is made by **⅔ of the total balance of all active validators**, is deemed a supermajority.
- When an epoch ends, if its checkpoint has garnered a ⅔ supermajority, the checkpoint gets justified.
- If a checkpoint B is justified and the checkpoint in the immediate next epoch becomes justified, then B becomes finalized.
- Use cases can decide whether they need finality or an earlier safety threshold is sufficient.

- Image is an example of one checkpoint getting justified (Slot 64) and finalizing a prior checkpoint (Slot 32).



OPEN QUESTION

- For a shard's crosslink to be included in the beacon chain, 2/3 of the relevant validators must vote for it (more precisely 2/3 of the stake). If this doesn't happen, a crosslink will not be created for that shard in that slot - this is the case "failed beacon chain inclusion".
- But as mentioned in the screenshot .. Even in case of "failed beacon chain inclusion". Shard Block is being included in the Shard Chain .. Correct?

- If The block with less than 2/3 or may be just few attestations are being included in the shard chain then will this not lead to inclusion of wrong shard data proposed by block proposer?
- That what I was trying to explain in my question..
- - Wrong Block data is proposed Proposer (i.e. wrong data about previous shard block)
- - As the data is wrong in the Block so it will get very less or no attestations (if validators are honest)
- - If still the proposed Block is included in the shard chain then how to maintain the data validity in shard Chain
- Ah, Ok. I see. The failed beacon chain inclusion only applies if the block later gets enough votes - that is, if it just missed its slot but is otherwise good.
- I think that if the next proposer doesn't like the block, it will build on the previous block instead and the bad block will be orphaned. It's as if the block didn't exist. But I haven't studied this in detail yet, but it should be in here somewhere: https://github.com/ethereum/eth2.0-specs/blob/dev/specs/phase1/shard-transition.md

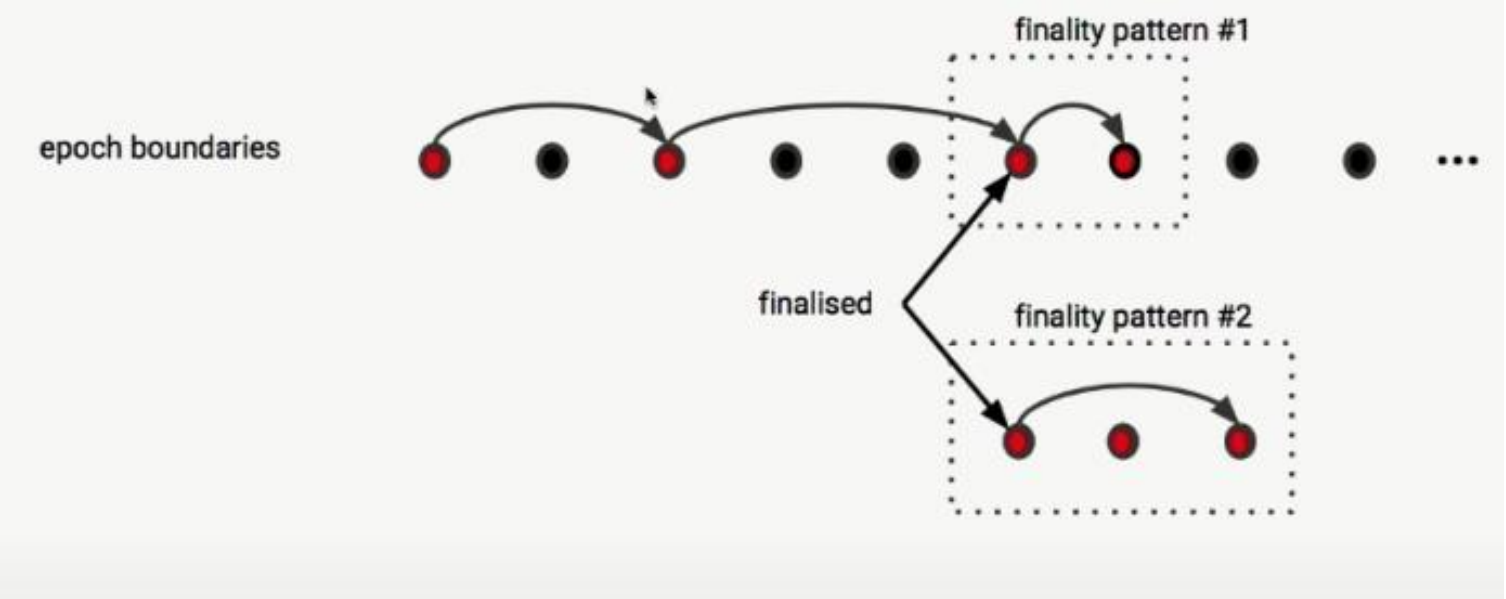
OPEN POINTS

Q: When it is said in spec that every validator has one Vote per Epoch, What does that Vote is used for?

- In phase 0, signaling fork choice vote and FFG finality vote
- Additionally In phase 1, the validity and availability of a shard block(s)

OPEN QUESTION

- What is the Finality pattern #2 which we had discussed in past?



JUSTIFICATION SCENARIO

Here is a storyline that could have been observed from genesis. All the proposers from Slot 1 until Slot 63 propose a block, and these appear on-chain. With each block in Epoch 1, its checkpoint (block at Slot 32) accumulates attestations from 55% of validators. The block at Slot 64 is proposed and it includes attestations for the Epoch 1 checkpoint. Now, 70% of validators have attested to the Epoch 1 checkpoint: this causes its justification. The Epoch 2 checkpoint (Slot 64) accumulates attestations throughout Epoch 2 but does not reach the ⅔ supermajority. The block at Slot 96 is proposed and it includes attestations for the Epoch 2 checkpoint. This leads to reaching the ⅔ supermajority and the justification of the Epoch 2 checkpoint. Justifying the Epoch 2 checkpoint finalizes the Epoch 1 checkpoint and all prior blocks.

- Q: When Check point 32 did not get 2/3 of votes that mean it will not be justified at Slot 64 .. Then in next epoch .i.e. Epoch 2.. Validators will vote for check point 64 or check point 32???

ATTESTATIONS: A CLOSER LOOK

An attestation contains both an LMD GHOST vote and an FFG vote. Optimally, all validators submit one attestation per epoch. An attestation has 32 slot chances for inclusion on-chain.

Q: Does that mean that even if there are less then 2/3 attestations for shard 1 block at **slot K** , it still have a chance until the epoch ends? .i.e. if any further committee for the **shard 1** also votes for the same LMD Ghost (.i.e. agrees on the same previous head for shard chain) BUT as we know that a validator also cast an FFG votes and LMD Ghost for Beacon chain in a slot.

BUT What if the previous block validators had voted for diff FFG Vote (.i.e. Different checkpoint source and target for Beacon chain) and well LMD Ghost for Beacon chain could be for different as the beacon chain head changes along with the slot propogates. So that mean there is chance that Beacon LDM could also be different.

Now there is another scenario how Beacon LDM Vote by validators could be different or same

A) If there are less then 2/3 LMD votes for beacon chain only at Slot Level

- In this case will the beacon chain propogate further? If yes then LMD vote for Beacon chain will be for sure different in each slot

B) If there are less then 2/3 LMD votes for beacon chain only at our mentioned shard 1 Level

- In this case Beacon chain should move ahead based on the overall attestations made by all validators in previous slot

- In this case my assumptions fits well (the beacon chain head changes along with the slot propogates.) and in such case will still there be a chance for our Slot 1 Shard 1 attestations to be included?

- This means a validator may have two attestations included on-chain in a single epoch. Validators are rewarded the most when their attestation is included on-chain at their assigned slot; later inclusion is a decaying reward.

This is mentioned in the blog BUT I am not sure how validator can make to two attestation when he only get single chance and that only in his slot.

OPEN POINTS

Proposers are selected by RANDAO with a weighting on the validator's balance. It's possible a validator is a proposer and committee member for the same slot, but it's not the norm. The probability of this happening is 1/32 so we'll see it about once per epoch.

- Does that mean a highest weight committee member can be a block proposer?
- Why the probability is 1/32?

Typically, there are more than 8,192 validators: meaning more than one committee per slot (.i.e. 8192 / 32 = 256 and one committee needs at least 128 validators). All committees are the same size.

- Does he mean that in a given epoch the number of shards will be functional will depend upon numbr of committee formed?
- Do we have Committee at Beacon chain level too?

OPEN POINTS

- What could be the reasons when a validator will attest a wrong block?
- A validator that is always offline or always votes on blocks that do not get finalised, will be penalised ¾ the amount that a validator would be rewarded for making punctual attestations that are finalised.
  - In this sentence, when we say always vote on blocks that do no get finalised, Does that mean such case could only happened when there is a bug in client or this could also happen in general protocol operation

IMP LINKS

- https://notes.ethereum.org/@vbuterin/HkiULaluS
- https://ethos.dev/beacon-chain/
- https://github.com/protolambda/eth2-docs
- https://github.com/ethereum/eth2.0-specs/blob/dev/specs/phase1/shard-transition.md
- Know more about *Secret Shared Validator clients for Eth2 trustless staking*

https://www.youtube.com/watch?v=Jtz9b7yWbLo