This document marks the beginning of a series in which I will discuss ePBS and its essential components from the Proof of Stake beacon chain perspective, excluding detailed coverage of the fork choice, networking, validators, and builders, which will be addressed in future posts. We'll explore new additions and changes to the beacon chain, dive into individual components, and provide some design rationales. Finally, I will highlight two potential blockers that need to be addressed before we can proceed further.

Reference Documents

If you want to review the Python specifications in detail, check out these links:

(Some of these are outdated with the latest EIP-7547 design)

- ePBS beacon chain spec
- ePBS fork choice spec
- ePBS design rational doc
- Prysm reference implementation

High-level Changes

- For ePBS, we leverage the following consensus EIPs:
- EIP-7251: Increase the MAX_EFFECTIVE_BALANCE
- EIP-7002: Execution layer triggerable exits
- EIP-7547: Inclusion lists
- EIP-7251: Increase the MAX_EFFECTIVE_BALANCE
- EIP-7002: Execution layer triggerable exits
- EIP-7547: Inclusion lists
- We borrow concepts from the following research posts:
- Payload-timeliness committee (PTC) – an ePBS design
- Why staked builder?
- Two-slot proposer/builder separation
- Payload-timeliness committee (PTC) – an ePBS design
- Why staked builder?
- Two-slot proposer/builder separation
- New staked consensus participant in the beacon chain

: builder

- New honest validator duty

:

- Payload timeliness committee (PTC) to cast payload timeliness votes.
- Proposer to include aggregated PTC attestations in beacon block body.
- Payload timeliness committee (PTC) to cast payload timeliness votes.
- Proposer to include aggregated PTC attestations in beacon block body.
- The slot is divided into four intervals instead of three

.

1. Proposer proposes block
2. Attester attests block

3. Aggregator aggregates attestations / Builder reveals payload
4. PTC votes payload

Note: Timestamps are not specified because the duration of each interval can be adjusted

- Proposer proposes block
- Attester attests block
- Aggregator aggregates attestations / Builder reveals payload
- PTC votes payload
- New notation

: empty slot

. This implies that consensus block exists but is missing execution data. This is not a skipped slot!. A skipped slot does not contain consensus data.

- Skipped

slot

: Missing consensus block therefore no execution payload can be included.

- Empty

slot

: Consensus block is included, but the execution payload is not included. This means the builder did not reveal, or builder revealed late.

- Full

slot

: Consensus block and payload are both included on the chain.

- Skipped

slot

: Missing consensus block therefore no execution payload can be included.

- Empty

slot

: Consensus block is included, but the execution payload is not included. This means the builder did not reveal, or builder revealed late.

- Full

slot

: Consensus block and payload are both included on the chain.

[

Screenshot 2024-02-11 at 12.39.35 PM

2474×528 110 KB

](https://ethresear.ch/uploads/default/original/2X/e/ea1b25aa265609874794826b97bbabb47c620f61.png)

- Two new beacon chain signing domains
- DOMAIN_BEACON_BUILDER

: Builder to sign over the execution header and execution payload.

- DOMAIN_PTC_ATTESTER

: PTC to sign over payload timeliness attestations.

- DOMAIN_BEACON_BUILDER

: Builder to sign over the execution header and execution payload.

- DOMAIN_PTC_ATTESTER

: PTC to sign over payload timeliness attestations.

- Beacon state object changes
- Added new fields previous_inclusion_list_proposer

and latest_inclusion_list_proposer

to track inclusion list is covered under empty slot condition (EIP-7547 and ePBS).

- Added new field signed_execution_payload_header_envelope

to track proposer selected builder header data is cache (ePBS).

- Added new field last_withdrawals_root

to track withdrawals are sufficiently covered if payload fails to reveal in the current slot and next slot will use the same payload (ePBS).

- Added new fields deposit_balance_to_consume

and pending_balance_deposits

to track builder to validator bid transfers are under churn limit (ePBS).

- Added new fields exit_balance_to_consume

, earliest_exit_epoch

and pending_partial_withdrawals

to track execution layer withdrawal (EIP-7002).

- Added new fields previous_inclusion_list_proposer

and latest_inclusion_list_proposer

to track inclusion list is covered under empty slot condition (EIP-7547 and ePBS).

- Added new field signed_execution_payload_header_envelope

to track proposer selected builder header data is cache (ePBS).

- Added new field last_withdrawals_root

to track withdrawals are sufficiently covered if payload fails to reveal in the current slot and next slot will use the same payload (ePBS).

- Added new fields deposit_balance_to_consume

and pending_balance_deposits

to track builder to validator bid transfers are under churn limit (ePBS).

- Added new fields exit_balance_to_consume

, earliest_exit_epoch

and pending_partial_withdrawals

to track execution layer withdrawal (EIP-7002).

[

Screenshot 2024-02-11 at 12.49.14 PM

1468×422 33.2 KB

](https://ethresear.ch/uploads/default/original/2X/8/864e30a179a8d2d990b1e385e18d173f02a11441.png)

- Beacon block body object changes
- Removed fields execution_payload

(ePBS) and blob_kzg_commitments

(EIP-4844).

- Added new fields signed_execution_payload_header_envelope

(ePBS), payload_attestations

(PTC) and execution_payload_withdraw_requests

(EIP-7002) and inclusion_list_summary

(EIP-7547).

- Removed fields execution_payload

(ePBS) and blob_kzg_commitments

(EIP-4844).

- Added new fields signed_execution_payload_header_envelope

(ePBS), payload_attestations

(PTC) and execution_payload_withdraw_requests

(EIP-7002) and inclusion_list_summary

(EIP-7547).

[

Screenshot 2024-02-11 at 12.49.26 PM

1478×364 29.6 KB

](https://ethresear.ch/uploads/default/original/2X/9/97deaa8fcb867627f3fc6c1a8667528d0118f2d0.png)

- Execution payload header object changes
- New object wrapper SignedExecutionPayloadHeaderEnvelope

that wraps around ExecutionPayloadHeaderEnvelope

and ExecutionPayloadHeader

.

- Builder index and bid value (Gwai) are defined in ExecutionPayloadHeaderEnvelope

.

- Builder signature is defined in SignedExecutionPayloadHeaderEnvelope

.

- Builder signs over ExecutionPayloadHeaderEnvelope

with beacon builder domain and public key in the beacon state.

- SignedExecutionPayloadHeaderEnvelope

is gossiped as a p2p networking object. Gossip topic is execution_payload_header

.

- There’s no change to the ExecutionPayloadHeader

object.

- New object wrapper SignedExecutionPayloadHeaderEnvelope

that wraps around ExecutionPayloadHeaderEnvelope

and ExecutionPayloadHeader

.

- Builder index and bid value (Gwai) are defined in ExecutionPayloadHeaderEnvelope

.

- Builder signature is defined in SignedExecutionPayloadHeaderEnvelope

.

- Builder signs over ExecutionPayloadHeaderEnvelope

with beacon builder domain and public key in the beacon state.

- SignedExecutionPayloadHeaderEnvelope

is gossiped as a p2p networking object. Gossip topic is execution_payload_header

.

- There’s no change to the ExecutionPayloadHeader

object.

- Execution payload object changes
- New object SignedExecutionPayloadEnvelope

that warps around ExecutionPayloadEnvelope

and ExecutionPayload

.

- Builder index, blob kzg commitments (EIP-4844), beacon block/state root, and inclusion list’s proposer/signature (EIP-7547) are defined in ExecutionPayloadEnvelope

.

- Builder signature is defined in SignedExecutionPayloadEnvelope

.

- Builder signs over ExecutionPayloadEnvelope

the way it signs over the header envelope.

- SignedExecutionPayloadEnvelope

is gossiped as a p2p networking object. Gossip topic is execution_payload

.

- There’s no change to the ExecutionPayload

object.

- New object SignedExecutionPayloadEnvelope

that warps around ExecutionPayloadEnvelope

and ExecutionPayload

.

- Builder index, blob kzg commitments (EIP-4844), beacon block/state root, and inclusion list's proposer/signature (EIP-7547) are defined in ExecutionPayloadEnvelope

.

- Builder signature is defined in SignedExecutionPayloadEnvelope

.

- Builder signs over ExecutionPayloadEnvelope

the way it signs over the header envelope.

- SignedExecutionPayloadEnvelope

is gossiped as a p2p networking object. Gossip topic is execution_payload

.

- There's no change to the ExecutionPayload

object.

[

Screenshot 2024-02-11 at 12.50.51 PM

1650×646 47 KB

](https://ethresear.ch/uploads/default/original/2X/f/fda0d1dff6e0c4e8ff91f6d73cd2083b813e277b.png)

- Payload attestation changes
- PayloadAttestation

wraps PayloadAttestationMessage

and PayloadAttestationData

.

- PayloadAttestationData

signals the head block root, the slot and whether the payload is revealed for block root.

- PayloadAttestationMessage

wraps PayloadAttestationData

with PTC validator index and signature.

- PayloadAttestation

aggregates PayloadAttestationMessage

with PTC bit vector and aggregated signature.

- PayloadAttestation

wraps PayloadAttestationMessage

and PayloadAttestationData

.

- PayloadAttestationData

signals the head block root, the slot and whether the payload is revealed for block root.

- PayloadAttestationMessage

wraps PayloadAttestationData

with PTC validator index and signature.

- PayloadAttestation

aggregates PayloadAttestationMessage

with PTC bit vector and aggregated signature.

[

Screenshot 2024-02-11 at 12.50.02 PM

766×566 22.3 KB

](https://ethresear.ch/uploads/default/original/2X/4/44759f9beadb483cd94bd3f6dd5372374cd40f01.png)

- Block processing function changes
- process_block_header

is modified to account for the validator’s new slashed field (EIP-7251).

- process_withdrawals

is modified to account for the empty previous slot condition. Beacon chain will not process new withdrawals if the previous slot already exists consensus changes, but execution changes failed to follow through (ePBS).

- process_execution_payload_header

is added, and process_execution_payload

removed (ePBS). Processing header replaces the processing payload validation steps (ie. withdrawals root, RANDAO, timestamp). Additionally, the processing header verifies the builder signature, the builder has enough balance to pay the proposer, and then the builder’s bid value is deducted at this function. For the inclusion list, It checks the summary’s proposer index matches the block proposer if the parent block is full (EIP-7547). Finally, it caches the payload header envelope to verify the execution payload after the builder reveals it.

- process_proposer_slashing

and process_attester_slashing

are modified to account for new slashing status for validator object (EIP-7251).

- process_payload_attestation

is added to account for processing of aggregated payload attestation messages in the beacon block body (ePBS). The attestations need to be valid for the block to be valid. PTC members and proposer are properly rewarded and penalized for the correct messages in PTC.

- process_execution_layer_withdraw_request

is added to process execution layer withdrawals (EIP-7002).

- process_block_header

is modified to account for the validator’s new slashed field (EIP-7251).

- process_withdrawals

is modified to account for the empty previous slot condition. Beacon chain will not process new withdrawals if the previous slot already exists consensus changes, but execution changes failed to follow through (ePBS).

- process_execution_payload_header

is added, and process_execution_payload

removed (ePBS). Processing header replaces the processing payload validation steps (ie. withdrawals root, RANDAO, timestamp). Additionally, the processing header verifies the builder signature, the builder has enough balance to pay the proposer, and then the builder’s bid value is deducted at this function. For the inclusion list, It checks the summary’s proposer index matches the block proposer if the parent block is full (EIP-7547). Finally, it caches the payload header envelope to verify the execution payload after the builder reveals it.

- process_proposer_slashing

and process_attester_slashing

are modified to account for new slashing status for validator object (EIP-7251).

- process_payload_attestation

is added to account for processing of aggregated payload attestation messages in the beacon block body (ePBS). The attestations need to be valid for the block to be valid. PTC members and proposer are properly rewarded and penalized for the correct messages in PTC.

- process_execution_layer_withdraw_request

is added to process execution layer withdrawals (EIP-7002).

[

Screenshot 2024-02-11 at 12.41.00 PM

3180×854 255 KB

](https://ethresear.ch/uploads/default/original/2X/e/eab326bbe9582a89aa8784acb3f99baecf91389c.png)

- Epoch processing function changes
- process_registry_updates

is modified to use the new churn limit (EIP-7002).

- process_pending_balance_deposits

is added to process builder to validator bid transfer under churn limit (ePBS).

- process_effective_balance_updates

is modified for new effective balance accounting (EIP-7251)

- process_registry_updates

is modified to use the new churn limit (EIP-7002).

- process_pending_balance_deposits

is added to process builder to validator bid transfer under churn limit (ePBS).

- process_effective_balance_updates

is modified for new effective balance accounting (EIP-7251)

[

Screenshot 2024-02-11 at 12.41.09 PM

3196×768 265 KB

](https://ethresear.ch/uploads/default/original/2X/0/02d009265ab313d19e99848221fe3682cfd502fe.png)

- Execution payload processing additions
- process_execution_payload

is now an independent check for the state transition after the builder reveals the payload before the cutoff. It verifies: * Valid builder signature

- The same proposer index as the consensus block
- Valid inclusion list summary signature
- Consistent with the header
- Consistent with the builder index
- Blob kzg commitments are valid and available
- Valid post-state root
- Valid builder signature

- The same proposer index as the consensus block
- Valid inclusion list summary signature
- Consistent with the header
- Consistent with the builder index
- Blob kzg commitments are valid and available
- Valid post-state root
- process_execution_payload

caches latest_execution_payload_header

and previous_inclusion_list_proposer

for subsequent slot accounting

- process_execution_payload

is now an independent check for the state transition after the builder reveals the payload before the cutoff. It verifies: * Valid builder signature

- The same proposer index as the consensus block
- Valid inclusion list summary signature
- Consistent with the header
- Consistent with the builder index
- Blob kzg commitments are valid and available
- Valid post-state root
- Valid builder signature
- The same proposer index as the consensus block
- Valid inclusion list summary signature
- Consistent with the header
- Consistent with the builder index
- Blob kzg commitments are valid and available
- Valid post-state root
- process_execution_payload

caches latest_execution_payload_header

and previous_inclusion_list_proposer

for subsequent slot accounting

End to end flow

1. Builders for slot n

prepare execution blocks (payloads) and submit blind execution blocks (header) with bids and signatures to the header subnet. The bid is an amount to be paid to the proposer regardless of whether the builder reveals the payload as long as the block becomes canonical. The execution block for n

must satisfy the inclusion list condition for n-1

1. Proposer for slot n

broadcasts consensus block with the best header and an inclusion list sidecar for n+1

builder to build on mandatory.

1. Attesters for slot n

cast votes on the head of the chain, if the head is n

block that implies that they verified the consensus block and the inclusion list sidecar.

1. Aggregators aggregate attestations. In parallel, the winning builder for slot n

reveals the execution payload by broadcasting the payload to the payload subnet.

1. Payload timeliness committee for slot n

cast votes on whether it has seen and verified the correct payload revealed from the same builder.

Expanding more:

- The proposer builds and broadcasts a beacon block at the second 0 of the slot. This beacon block includes an execution header comprising a transaction root, a builder index, the builder's signature, and a bid transfer value. This differs from today, where a beacon block includes a full execution payload. The proposer also builds an inclusion list transaction sidecar and broadcasts it alongside the beacon block. The inclusion list transaction sidecar only contains the transactions. The inclusion list summary is included in the block body.
- All nodes verify the validity of the beacon block, inclusion summaries and transaction list using the consensus and execution client through execution-API. The node inserts the block into the fork choice tree to update the chain's head. Attesters cast their votes for the head of the block tree before the attestation cut off time.
- The builder reveals its payload at the same time as aggregator releases aggregated attestations. Builder will reveal the payload if it sees its header included in the beacon block with sufficient attester votes.
- The Payload Timeliness Committee (PTC) votes for the builder's reveal and correctness and broadcasts the PTC message before the PTC attestation cut-off time.
- The next slot n+1

's proposer and attesters run fork choice compute head to determine the head of the chain based on the weight of the fork choice branch and the timeliness of the builder's reveal. Only proposers and attesters can impose weight on the fork choice tree through out time. The PTC assesses the timeliness of the builder's payload reveal, determining whether the slot is full or empty, affect what is head for the subsequent slot.

[

Screenshot 2024-02-11 at 12.44.23 PM

3538×784 365 KB

](https://ethresear.ch/uploads/default/original/2X/6/6f613a2c0d67a28d2478eae821473ac8ed2ada45.png)

Now we will go through each component section by section

In-protocol staked builder

ePBS adds a new staked consensus participant builder

. The builder is identical to the validator. It performs consensus duties. The main differences between builder and validator are:

- Withdrawal prefix: BUILDER_WITHDRAWAL_PREFIX
- Staking requirement: BUILDER_MIN_BALANCE
- Builder prepares and broadcasts the execution payload header with bid and signature. Builder then reveals the execution payload once the header is included in the block when it's safe to do so.
- Builder also performs validator duties. Given the different staking requirement, EIP-7251 Max Effective Balance is a prerequisite for ePBS. To become a builder, anyone can manually send a deposit with sufficient amount and the builder prefix. The protocol may also automatically migrate validator to builder during the fork transition if the validator's balance surpasses BUILDER_MIN_BALANCE

.

Why staked builder?

Builder is be staked to be part of the consensus. If everyone is allowed to sign a header and a payload, the system is then bypassed, which means we are back to mev-boost. Rest is a tradeoff question: how much should the builder be staked? The payment can be guaranteed on the consensus layer, and with any staking, the protocol should provide them rewards, while the validator duties are performed.

Inclusion List

Given staked builder, solo validators can no longer self-build without their balances exceeding BUILDER_MIN_BALANCE

. To ensure censorship resistance of the network, the inclusion list has become a mandatory component for ePBS. Lately the design of the inclusion list has gained significant momentum. One notable version is EIP-7547, which is expected to be implemented in future hard forks.

The inclusion list is forward-looking and with its dedicated gas limit. Here's a summary of its functionality. More details can be found by reading the EIP, consensus specs and more info here:

- At slot n

: The proposer of block n

broadcasts an inclusion_list_transactions

sidecar along with the signed_beacon_block

. This sidecar contains inclusionl list transactions with a proposer signature for gossip verification. The proposer inserts inclusion_list_summaries

in the block body. A beacon block n

without a valid inclusion_list_transactions

sidecar that satisfied the summarizes in the block body will not be imported into the fork choice by honest nodes. To verify inclusion list validity:

- The lengths of summaries

and transactions

must match and be less than MAX_TRANSACTIONS_PER_INCLUSION_LIST

.

- The total gas limit of the transactions

must be less than MAX_GAS_PER_INCLUSION_LIST

.

- The execution layer client must verify the inclusion list, ensuring that the transactions are executable at the start of slot n

and have a maxFeePerGas

at least 12.5% higher than that of the current slot's maxFeePerGas

.

- The lengths of summaries

and transactions

must match and be less than MAX_TRANSACTIONS_PER_INCLUSION_LIST

.

- The total gas limit of the transactions

must be less than MAX_GAS_PER_INCLUSION_LIST

.

- The execution layer client must verify the inclusion list, ensuring that the transactions are executable at the start of slot

n

and have a maxFeePerGas

at least 12.5% higher than that of the current slot's maxFeePerGas

.

- At slot n+1

: The proposer or builder of n+1

will build the block using the inclusion list of n

, and the transactions will be in the execution payload. The criteria for execution payload validity are:

- For each index i

in n

's inclusion_list

, the tx[i]

is either included in slot n

or the current execution payload with increasing order.

- The execution payload is valid from the execution layer client's perspective.

- For each index i

in n

's inclusion_list

, the tx[i]

is either included in slot n

or the current execution payload with increasing order.

- The execution payload is valid from the execution layer client's perspective.

What happens if there's an empty slot?

If the payload for the canonical block in slot n

was not revealed, then the summaries and transactions list for slot n-1

remain valid. The honest proposer for slot n+1

should not submit a new IL sidecar. If the proposer submits inclusion list on top of an empty slot, the inclusion list will be ignored. The builder for n+1

is required to satisfy the summary of n-1

. If there are k

slots in a row that are missing payloads, the next full slot still must satisfy the inclusion list for n-1

.

It's essential to prove that a valid payload can always be produced, satisfying the inclusion list even if the slots are skipped or empty. It is also important to prove that builders cannot manipulate the inclusion list to force transaction reversion due to gas limits. Transactions in the exclusion list must have lower nonces, preventing builders from invalidly altering the transaction order in the exclusion list. Thus, valid transactions from n

remain valid for the next block.

Payload Timeliness Committee (PTC)

Inspired by the PTC an ePBS design, ePBS bootstraps a new, small, honest validator committee for casting payload

timeliness votes called PTC in our design. Unlike the beacon attestation committee, the PTC does not have explicit fork choice weight in determining the block tree that nodes follow. Instead, it votes on the timeliness of the payload revealed by the builder. Nodes will use PTC votes to determine whether a slot is full or empty. In the current design, members of the PTC Committee are selected as a subset of the beacon slot's attestation committee. For instance, they can be the first member of each Beacon Committee who is not a builder. The normal beacon attestations submitted from the PTC Committee are ignored. The PTC votes on (head_root, payload_present)

, indicating whether before PTC attestation cut-off time, the validator has seen a payload.

PTC Rewards And Penalty

The PTC members assigned to PTC duty receive a full attestation reward if they correctly vote for head root and payload present values. Otherwise, they incur a penalty equivalent to that for a missed attestation. The PTC members do not receive additional rewards for performing beacon attestation duty.

Consensus Block Body Includes PTC attestations

PTC attesters broadcast PayloadAttestationMessage

over the p2p beacon payload_attestation_message

subnet. These messages can be aggregated, and the honest proposer of the next slot should include the aggregated PayloadAttestation

s in its block body. The same beacon attestation rewards apply to the proposer by including the correct PTC attestations. The current design allows the proposer to include one or two aggregated attestations covering both revealed statuses.

PTC Committee Duty

Validators are selected for PTC duty with an epoch lookahead using the helper function similar to beacon attestation duty. PTC committee has the same look ahead and using the exact shuffling mechanism as the beacon attestation committee.

PTC Optimal Committee Size

PTC committee size is currently set at 512. We want a size that's small enough that we don't need another aggregation interval, which could prolong the existing 12s slot time. 512 is a reasonable size for gossip, the sync committee is a good reference. 512 also feels "safe," given that a 35% attester will need 20,000 years to take over 51% of a committee using binomial with continuity correction.

Builder Payment To Proposer

The builder to proposer's bid transfer amount is in Gwai

and included in the ExecutionPayloadHeaderEnvelope

object with the builder's index. The SignedExecutionPayloadHeaderEnvelope

object contains the builder signature. The bid is not automatically added to the proposer when the block with the header becomes canonical. Two-step process: 1. The builder's balance is deducted by the bid amount once the block is processed, then the bid is transferred as PendingBalanceDeposit

object cached in the beacon state 2. It then subsequently added to the proposer's balance at the epoch boundary, the amount of transfer allowed in a given epoch is subjected to a churn limit. The beacon state tracks all the bid transfers from builders to proposers and the maximum transferable amount of a given epoch, which is calculated by get_validator_churn_limit

. If the churn limit is not sufficient, then the pending bids will be transferred to the future epoch boundaries.

Why not same-slot payment as long as the block is canonical?

Same slot transfer may trigger an attack vector where a builder can transfer large amounts to proposers by placing a large bid. This poses a fork choice risk, as it can instantaneously shift large weights from one block branch to another block branch and potentially circumvent significant penalties in slashing events. To mitigate this, bid transfers are churned by borrowing the deposit design from Max EB EIP. This smoothens the increase in the proposer's balance suddenly.

Withdrawal Consideration

Validator withdrawal works differently under ePBS. Today, withdrawals are deterministic and originate from the consensus layer, then span to the execution layer. Both layers must process withdrawals synchronously, or we will get a consensus

failure. In ePBS, withdrawals are cached in the beacon state after processing the consensus block. During the execution phase, the execution block payload withdrawals will be verified against the cached beacon state withdrawals.

With an empty slot, the withdrawal comes incomplete despite the consensus data that exists. In this case, the protocol will require the next slot's n+1

builders to process withdrawals from n

in their payloads. Or else the payload is invalid. To summarize, if the payload is absent, the cached withdrawals in state persist for inclusion in future execution payloads.

EIP-4844 Consideration

Today, blob transactions are tightly coupled with the execution payload. The consensus clients call execution-API NewPayload

to request execution clients to verify blob sidecars alignments. ePBS shifts KZG commitment fields from the beacon block body to the execution payload. When the proposer selects the best header, it will be unaware of the blob content, which is only known during the execution stage when the payload is revealed and then can be verified. The data availability check is also shifted from the consensus phase to the execution phase's payload processing. An execution payload is valid only if it passes the data availability check and KZG proofs and KZG commitments are consistent across blob sidecars referenced in the execution payload. Builders may broadcast blob sidecars earlier as they seem safe after seeing the beacon block, which reduces last second propagation of blob sidecars, which often take longer to propagate than the execution payload itself. The inclusion list design does not accommodate blob transactions, which should be considered in future research.

Proposer Splitting Attack

What is the proposer splitting? At slot n

, the proposer reveals the block close to the attestation deadline, which splits the view of the attesters. Half of the validators voted for the n

block, and half voted for a different block. The votes for n

bock are weak. As a builder with a header in the proposer block. I now face a dilemma before the payload deadline: to reveal or not. What are the potential outcomes?

Reveal:

- Happy case: the n

block is canonical, the builder pays the proposer, and gets its payload included

- Unhappy case: the n

block is not canonical, the builder does not have to pay the proposer, but everyone, including the n+1

proposer learns about the payload. This is not the same as the same slot unbundling vulnerability. The downside is information leakage, which could be used as beneficial for n+1

proposer

Not reveal:

- Happy case: the n

block is non-canonical. The builder protected itself and won. It doesn't have to pay the proposer and none of it's payload got leaked

- Unhappy case: the n

block is canonical. The builder has to pay the proposer but never gets its payload included

What are some potential solutions?

- Proof-based: Allow the builder to prove it's "honest" that it indeed revealed the payload "on-time" based on the proposer block. Allow the builder to prove that the proposer is "dishonest". How to prove it is hard.
- Fork choice based: Give the builder some weights in the fork choice. Give the builder some power to determine the head of the chain. But this also opens up the builder griefing the proposer

- Strong finality based: If today the proposer block “cant be forked” then the problem no longer exists. Many solution spaces have single-slot finality or strong BFT consensus for the proposer block. But these solutions prolong the development cycle or beacon chain slot time from 12s to 24s.

Proposer Equivocation Attack

One of the open questions in any ePBS design space is proposer equivocation to unbundle the builder payload so points here:

- The current design protects builder from same slot unbundling, it does not protect builder from next slot re-orging and builder leaking information for next slot proposer and builders.
- A builder must first reveal the payload of a proposed block before a proposer can attempt to unbundle. This means the builder is fully protected against proposer equivocation until the payload is revealed. The crucial monitor for a builder before the reveal phase is not the number of blocks a proposer released but the votes that the header accumulated.
- If the block initially committed to by the builder is eventually included in the blockchain, the builder doesn’t incur any risk. Builder pays proposer the bid amount This outcome aligns with the builder’s original intention.
- After revealing the payload, the builder is safe from being unbundled unless next slot’s beacon committee is predominantly composed of malicious validators that vote for an empty block over the original block. If the attesters’ committee is corrupted, we are powerless in this situation, and the same issue persists currently.

Forkchoice Changes

We will have a separate post to cover fork choice changes because it’s more nuanced. In this post, I will summarize the important concepts introduced in ePBS because fork choice & beacon chain are closely coupled

- One fundamental change is the head block could be with or without

execution data. Then what should get_head()

return? First point is the head returns the block with the strongest LMD weight disregarding its payload status for future slot. There will be a helper is_payload_present(block_root)

, which returns true whether the slot exists an execution data.

- Checkpoint is still the same as today which a (epoch, root)

pair. Given payload status does not change any consensus head, to compute a new head, a node still start from a latest justified checkpoint and then traverse down the block tree to get its head.

- To account for LMD weights in the block tree, the accounting scheme is identical to today under one exception: direct votes are only counted in chains supporting distinct payload status based on PTC votes. This means you have (n-1, full)

parent and two children (n, empty)

and (n, fill)

. Both children’s weights would be counted towards (n-1, full)

. Now let’s say (n+1, full)

builds on top of (n, full)

, not (n, empty)

. In this case, the weight would only count towards (n, full)

.

We’ll do a comprehensive post covering fork choice changes in further details

Roadblocks

As of today, the most significant road blocks to ePBS are proposer splitting and fitting entire end to end flow within a 12-second timeframe. The proposer splitting attack is particularly challenging to address without necessitating additional rounds of fork choice or achieving single slot finality. Fitting everything within 12 seconds is challenging, but optimistic, given that consensus and execution are divided between two virtual slots, reducing the comparison required during peak periods. We

will share more information as we continue to make progress