MEV-Boost: Merge ready Flashbots Architecture

This architecture is a work in progress - the final design will reflect the feedback collected and lessons learned through implementation and can be found on the mev-boost github repository

This document outlines the design for a marketplace for block construction (often referred to as block proposer / block builder separation or PBS) compatible with the upcoming Ethereum merge fork. This trust based solution closely resembles the current Flashbots auction design with modifications to enable solo staker participation without introducing changes to Ethereum consensus. This solution aims to bridge the gap to a permissionless PBS design which should be strongly considered for the cleanup fork in order to improve decentralization.

Thank you to the Flashbots team, EF research team, client implementation teams, and staking service providers for the feedback and countless hours of discussion which went into this document.

terminology

- consensus client

: the client in charge of ethereum proof of stake consensus

- execution client

: the client in charge of ethereum virtual machine execution

- execution payload

: a message containing the complete content of an unsigned execution payload to be added into a beacon block

- mev-boost middleware

: a new piece of software that sits between the consensus client and the execution client to outsource block construction

- proposer

: party which signs and submits a beacon block to the network, often referred to as the validator

- user

: a normal ethereum user who sends transactions

- searcher

: advanced ethereum user specialized in finding MEV opportunities and sending advanced transaction types like bundles

- builder

: party specialized in the construction of ethereum execution payloads using transactions received from users and searchers (trusted by searchers and users for fair inclusion)

- relay

: party specialized in DoS protection and networking who validates and routes execution payloads to proposers (trusted by builders for fair routing and by proposers for block validity, accuracy, and data availability)

- escrow

: party tasked with providing redundant data availability for proposed execution payloads (trusted by relay for data privacy)

architecture

This proposed architecture allows validators to outsource the task of block construction to a network of third party block builders. While the validators have the ability to include any payload into the chain, we believe network neutrality and validator revenues are maximized when the validator's job is limited to selecting the payload which pays them the most ETH.

[

image

2002×628 33.2 KB

](https://ethresear.ch/uploads/default/original/2X/c/c9fa06a58cabde886291b28dedb643abd968623c.png)

The block construction process involves the following steps:

1. Users and searchers send transactions to block builders through the public p2p txpool or through direct channels.
2. Builders construct execution payloads using these transactions as well as header parameters provided by validators. Builders may directly set the validator's feeRecipient

address as the coinbase address of the payload, or the builders may set their own address and include a transaction which transfers to the feeRecipient

in the payload.

1. Relays receive execution payloads from builders and verify the validity of the payloads as well as calculate the payload value (amount of ETH paid to the feeRecipient

).

1. Escrows receive the full execution payloads from relays to provide data availability.
2. Validators receive execution payload headers from relays (execution payloads stripped of the transaction content). The relay must attach an indication of the payload value to each header. The validator selects the most valuable header, signs the payload, and returns it to the relay and escrow to be propagated to the network.

Note that a single entity may play multiple roles in this system. The roles are labeled separately as they each lend themselves to some level of specialization. For redundancy and decentralization, there should be multiple independent parties competing in each role. In the future, a permissionless PBS design at the consensus level will remove the need for relays and escrows.

On the validator side, this architecture makes use of an independent middleware called MEV-Boost which sits between the consensus client and the execution client. This middleware handles communication with the relays, the profit switching logic for selecting the most valuable payload, and a fallback mechanism to a local execution client if relays get disconnected. Using a middleware instead of direct modification of the consensus clients allows for maintaining each component independently and provide cross client compatibility with minimal changes to the engine api.

[

image

1368×834 32.9 KB

](https://ethresear.ch/uploads/default/original/2X/f/f0ef8cdd3c5b3f28bc7267aa920aac486338c273.png)

It's important to note that relays should only be used for payloads during block construction - relays are not meant to be used for other roles performed by the execution clients like verifying blocks for attestations. Validators should always fall back to local payload construction if relays are unavailable.

Since the execution payload header does not include the transaction content, the risk of validators frontrunning is eliminated. This design allows solo stakers to participate in MEV thus reducing the incentive for economic centralization. However, this

design also requires validators to trust the relay to filter invalid payloads, accurately report payload value, and reveal the transaction content once the header is signed. Escrows are introduced to improve data availability by duplicating the transaction content across multiple providers.

Summary:

- Validator trust relay for data availability, payload validity, and payload value accuracy
- Users and searchers trust builders, relays, and escrow for frontrunning protection
- Minimal changes to consensus clients due to middleware architecture
- Adds additional round trip communication between relay and validator to block proposal latency

discussion points

client modifications

Flashbots will provide a formal specification similar to thecurrent MEV-Geth specification and will support client implementation teams to add MEV compatibility for the merge.

The current design requires the following modifications to the consensus client interface:

- enabling the validator to include and signexecution payload header in beacon blocks instead of full execution payloads
- enabling the validator to sign prefixed messages using the staking keys for authentication purposes
- enabling the consensus client to route a signed beacon block back to the middleware instead of attempting to send to the network
- enabling the consensus clients to fall back to a local or remote execution client if there is a fault in the middleware

There are no execution client changes currently required.

malicious relay

The most important security mechanism of this design is the ability for all validators to identify if a relay has turned malicious.

Lets assume the worse case scenario: there is a single relay which is connected to 100% of the validators. The relay begins proposing invalid blocks and lies about the value of the payloads such that the validators always selects them.

Under this scenario, a naive MEV-Boost implementation could cause the chain to halt as validators are no longer proposing valid blocks.

There are three ways for the relay to propose a bad payload:

1. invalid payload: the payload breaks some consensus rules
2. inacurate value: the payload proposed has a different value than claimed once executed
3. missing data: the payload body was never revealed by the relay

To mitigate this, it must be possible to either 1) pre validate payload between multiple parties ahead of sending to validators or 2) for all validators in the network to identify when a relay has proposed a bad payload and automatically fallback to a safe alternative. Critically, this security mechanism must be robust in the face of relays who are adversarial towards each other.

communication between MEV-Boost and relays

There are multiple alternatives for communication between MEV-Boost and the relays: push vs pull, direct vs p2p.

Care must be taken to ensure the following requirements are met in order to select the best implementation:

1. it must protect validator privacy by not associating validator key and IP address
2. it must have lowest possible latency
3. it must be robust against adversarial relays / validators

risk of missed slots

Missed slots can occur if the block proposer fails to propagate a block in time for sufficient attestations to be collected. In this case, the block proposer does not receive the base reward for the block (approx 0.025 eth at 10m eth staked) nor the transaction and MEV rewards. Since this architecture requires sending signed headers back to the relays / escrows before propagation to the attestors, it will be important to understanding what is an acceptable rate of missed slots and how to minimize it under normal network operating conditions.

header parameters

In order to produce valid blocks, builders and relayers must know all of the attributes of the execution payload header. It's in the interest of validators to provide all the inputs required for block construction as early as possible. This allows builders to maximize the accuracy and compute time available for finding an optimal block construction.

All header attributes except from coinbase

can be derived by the builder based on their observed state of the network. It is up to the middleware to filter payloads built on incorrect states.

validator authentication and feeRecipient

address

The validators needs to communicate to builders and relays the address they wish to use as the feeRecipient

. This address is necessary to accurately measure the value of the proposed payload. Since feeRecipient

can be any address, the validator must authenticate themselves when publishing it.

There does not currently exist a way for validators to authenticate and publish their feeRecipient

. The best way to do this would likely be to add a signing domain which allows validators to safely sign arbitrary messages with their staking key.

partial blocks vs full blocks

This specification deprecates the use of "Flashbots Bundles" in the communication between builders and block proposers in favor of execution payloads which represent the full ordering and content of a block. While bundles have provided an efficient auction mechanism for inclusion of transaction at the top of blocks, they are not expressive enough for all order preference. In particular, bundle auctions are not well suited for large transactions or frontrunning protection use cases. Moving to full block proposals means the full range of preference on ordering can be expressed.

out of band bribes for priority and MEV smoothing / burning

It is possible some validators will accept out of band bribes to prioritize certain payloads or attempt to perform time bandit attacks through attestation bribery. It is not possible to protect against this behavior without consensus modifications. In theory, consensus could enforce that the block builder which pays the most to the block proposer in ETH must be included and that the payment is spread or burned, though the mechanism is out of scope of this proposal requires further research.

decentralization

As mentioned in the architecture section above, relays are trusted providers of frontrunning and DoS protection. While the system enables having multiple relays competing with eachother permissionlessly, it is possible there will only be a handful

of successful relays. We suggest a permissionless design be strongly considered by core devs for the cleanup fork in order to eliminate the need for relays which should improve network decentralization.

client optimizations

Although not recommended, a validator who is relying exclusively on third party block builders for block construction may run a stripped down version of an execution client which removes the transaction pool and the block construction logic thus reducing the hardware and bandwidth requirement of their client.