Thanks to my collaborators @murat, @ckartik, @Evan-Kim2028, and @bemagri. Further thanks to the ETHGlobal London community for early feedback and in particular Nethermind for awarding a prize at the hackathon for our credible preconfirmation leader auction.

Introduction

In this post, we describe an out-of-protocol mechanism for blob inclusion preconfirmations. It allows preconfirmation providers to bid in an auction to become the leader for the subsequent slot. The auction winner can then accept bids on blob inclusions and issue preconfirmations to the bidders. The preconfirmed blobs then act as an inclusion list that needs to be respected by all participants. The goals of this design are to enable L2 sequencers and other entities relying on blob inclusion to gain certainty about their inclusion, prevent censorship, and possibly mitigate latency issues of blocks with many blobs in the future. Our out-of-protocol design can be implemented right now and serve as a starting point to measure and understand blob inclusion lists in practice until they are available natively in the Ethereum protocol.

Given recent issues with blob contention [1,2], we believe it is timely to implement an out-of-protocol solution as the one described here.

We next provide an overview of the entities involved in the protocol and then describe the protocol in more detail. We then provide more details on the payment and slashing mechanism. We conclude with discussions on the N+1

slot design, the leader auction, and how to handle premature blob inclusion.

Actors

We assume there is a sidechain (with a faster block time than L1) that is used to settle payments among the protocol participants. This chain in the following is called mev-commit chain

.

The following actors participate in the protocol:

- L1 proposers. We assume a subset of L1 validators opt-in to participate in the protocol and stake a collateral on the mev-commit chain.
- Relays. The opted-in proposers need to exclusively work with relays that participate in the protocol and only forward blocks from builders that also participate in the protocol.
- Preconfirmation providers. These are the actors bidding to become the blob preconfirmation leaders and issue the blob inclusion lists. They could be the proposers themselves, builders, or relays. For this write-up, we assume the relays play the role of preconfirmation providers.
- Preconfirmation bidders. The entities who want blobs to be included in a block, such as L2 sequencers.
- Builders. The builders need to be aware of the blob inclusion lists and include the corresponding blobs in their blocks.

Protocol Description

We next describe the protocol execution in more detail. Let N

be the current L1 slot and consider preconfirmation bidders want to include blobs in a block by slot N+1

(see below for a justification of the one-slot delay).

The protocol description is divided into three steps. The first is done during slot N

, the second at the beginning of slot N+1

, and the last one during the rest of slot N+1

. See also the figures below for graphical overviews of the corresponding steps.

Protocol Execution in Slot N

1. During L1‘s slot N

, preconfirmation bidders send their bids to the providers. A bid contains the hash of the KZG commitment of the blob, the bid amount, and the target slot N+1

, and is signed by the bidder.

1. Preconfirmation providers collect preconfirmation bids during L1 slot N

.

1. The relays further submit leader bids for slot N+1

to the leader auction.

[

Fig. 1: Protocol overview for Slot N.

1212×534 19.2 KB

](https://ethresear.ch/uploads/default/original/2X/7/78d6d4a75c61c8924502b41d72fa6adfa2853f0f.png)

Protocol Execution at the Beginning of Slot N+1

1. At the beginning of slot N+1

, the leader auction declares the provider with the highest bid as the leader for slot N+1

by publishing the leader together with the auction price on the mev-commit chain. The auction further settles the payment from the leader to the proposer.

1. The elected leader can now issue preconfirmations. Note that the leader has already received all preconfirmation bids for that slot and, therefore, can issue all preconfirmations immediately. They do so by publishing a blob inclusion list on the mev-commit chain. This list is final at this point and includes all bids including the blob KZG commitment hashes the leader commits to. To avoid timing issues in the next step, we can require the inclusion list to be published sufficiently early, e.g., within the first 6 seconds of the slot (and ignore lists published too late).

[

Fig. 2: Protocol overview for beginning of slot N+1.

849×599 45.9 KB

](https://ethresear.ch/uploads/default/original/2X/4/4ed8dfebb950c5c7877ade27b2d878d918a04cb0.png)

Protocol Execution During Slot N+1

1. The builders receive from the mev-commit chain the inclusion list of blobs that are mandated to be in the block. Then, the builders build a block containing those blobs (in an order of their choice) and send it to the relays. Note that the builder can also include additional blobs if they choose to do so.
2. The relays only forward to the L1 proposer the blocks from builders that opted-in to the protocol. The relays can also optionally verify that the blocks respect the blob inclusion list (or simply trust the builders in case of optimistic relays).
3. The proposer signs a block header received from a relay (this can be done optimistically without checking for blob inclusion).

[

Fig. 3: Protocol overview for slot N+1.

1153×553 41.6 KB

](https://ethresear.ch/uploads/default/original/2X/5/5498d249fa6e64846cf1c2a555e5b2c93a003873.png)

L1 Monitoring

1. An oracle monitors the L1 chain for blocks and reports to the mev-commit chain when a block is proposed by an opted-in proposer, who the proposer is, and whether this block (or a previous block) contained the blobs from the inclusion list. The oracle may want to wait for L1 finality to avoid reorg issues.
2. If the oracle reports a blob inclusion list violation, slashing is executed (see below for details).

Payments and Slashing

We consider the following rules for payments and slashing.

The following two types of payments are always executed (even if blobs are not included, slots get missed, etc.):

1. The preconfirmation leader pays the amount for winning the leader auction to the L1 proposer (regardless of whether any preconfirmations are issued; this is because they participated in the auction and won the rights to become the leader).

2. For all blobs in the inclusion list, the preconfirmation bidders pay the corresponding amount to the preconfirmation providers (regardless of whether the blob actually gets included; bidders may get reimbursed via slashing of proposers as described below).

If the oracle reports that a blob from an inclusion list is not included in an L1 block in slot N+1

or earlier, we need to execute slashing. There are different scenarios that can lead to a blob from the inclusion list not being included on L1:

1. An opted-in relay sends a block not containing the blob.

2. The proposer includes a block from an external relay.

3. The proposer proposes their own block.

4. The proposer misses the slot.

5. The block gets proposed but then orphaned in a reorg.

In the first case, the relay or the builder violated the protocol, while in cases 2, 3, and 4, it is the proposer's fault. Without additional mechanisms, the proposer cannot prove which relay has sent what block. We therefore always slash the proposer. If it was indeed the relay's fault, the relay's reputation with the proposer gets "slashed" in the same way as the relay sending invalid blocks. If in turn the relay has received that block from an opted-in builder, that builder's reputation with the relay is "slashed" and they can settle the dispute out-of-protocol with existing mechanisms.

The last case is special since it may not be the fault of anyone involved and constitutes a general reorg risk the proposers need to consider. The slashing amount therefore needs to be set such that proposers still can make profit overall.

The slashed amount is distributed to the preconfirmation bidders proportionally to their bid amounts since they are the ones harmed by the protocol violation.

Further Details and Discussions

N+1

Slot Design

In our protocol, preconfirmation bidders bid in slot N

for blob inclusion in slot N+1

. This means that there is an expected one-slot delay between bidding and blob inclusion. While this would be unacceptable for time-sensitive transactions, e.g., for mev extraction, blob inclusion is substantially less time-sensitive.

We further note that this next-slot inclusion list design is also used in EIP-7547 [3]. The lack of an out-of-protocol next-slot design and consequently the lack of real-world data from such designs has recently been criticized in the context of L1 inclusion lists [4]. Hence, the availability of this via mev-commit can be used to gather data about the efficacy of next-slot inclusion lists and can serve the Ethereum Foundation to make an effective decision based on the obtained empirical data.

The purpose of the delay is to ensure that at the time the block builders want to build the block, the blob inclusion list is available to them. Furthermore, this allows the preconfirmation leader of slot N+1

to preconfirm blobs at the beginning of the slot. By doing so, all participants know which blobs are going to be included in the block of slot N+1

way ahead of the end time of that slot. In particular, the L1 validators could in the future use this information to pre-fetch the blob data at this time, thereby mitigating reorg risks due to long blob propagation (see, e.g., [5]). Having a predetermined leader also means that once the leader issues a preconfirmation, all participants can immediately rely on the blob inclusion, which allows for synchronous composability.

Leader Auction

The auction to determine the preconfirmation leaders should ideally be credible, i.e., not require a trusted auctioneer. There are different ways to achieve this, and the precise implementation is orthogonal to our preconfirmation design. One option using SUAVE was explored by the Primev team, which has built a prototype auction at ETHGlobal London and was awarded a prize from Nethermind [6]. Another option is to use cryptographic tools such as timed commitments to realize the auction [7]. As an intermediate solution, the auction can also be run by a trusted actor.

Premature Inclusion of Blobs

Leaders can issue preconfirmations for blobs to be included in slot N+1

. However, before slot N+1

, some builder (possibly outside of our protocol) may have already included this blob in a block that is now part of L1. In such a case, all payments are executed as described above and no slashing takes place. The preconfirmation bidders got their blobs included even earlier than expected and nobody else is harmed in this scenario.

Conclusion

We have presented an out-of-protocol mechanism for blob inclusion lists. As long as inclusion lists are not available as part of the Ethereum protocol, we believe our solution offers a viable solution to the current issues with blob contention. Since it is a next-slot design, findings from this solution can further help to improve the understanding of such designs, which are also considered for L1 inclusion lists.

Currently, mev-commit for transaction preconfirmations is live on the Holesky testnet and we are excited to add an implementation of the design discussed here. We are looking forward to feedback to further refine and improve this system.

References

[1] https://twitter.com/bertkellerman/status/1773031698222989623?s=46

[2] https://twitter.com/mcutler/status/1773033173573628009

[3] EIPs/EIPS/eip-7547.md at 30fec793f3cb6769cb44d2d0daa5238451f67c48 · ethereum/EIPs · GitHub

[4] The Case for ILECTRA - HackMD

[5] Validator Timing Game Post EIP4844

[6] BlobPreconf Auction | ETHGlobal

[7] Riggs: Decentralized Sealed-Bid Auctions