Special thanks to Justin Drake and the Flashbots team for feedback and discussion.

A major risk threatening the ongoing decentralization of consensus networks is the economics around miner extractable value (MEV), sophisticated tricks to extract profit from the ability to choose the contents of the next block. A simple example of MEV is arbitraging all on-chain decentralized exchanges against price movements that have happened since the previous block. While normal PoS rewards are reasonably egalitarian, as single validators earn the same rate of return as powerful pools, there are significant economies of scale in finding sophisticated MEV extraction opportunities. A pool that is 10x bigger will have 10x more opportunities to extract MEV but it will also be able to spend much more effort on making proprietary optimizations to extract more out of each opportunity. In addition to this problem, MEV also complicates decentralized pooling, as in a decentralized pool there would still need to be one entity packaging and proposing the block, and they can easily secretly extract MEV without sharding that revenue with the pool itself.

The best-known solution is proposer/block-builder separation. Instead of the block proposer trying to produce a revenue-maximizing block by themselves, they rely on a market where outside actors that we call block-builders produce bundles consisting of complete block contents and a fee for the proposer, and the proposer chooses the bundle with the highest fee. The proposer's choice is reduced to picking the highest-fee bundle, an algorithm so simple that in a decentralized pool it can even be done inside an MPC to prevent cheating.

This post proposes some designs for how this could be done.

See also, ideas from 2018 that closely inspire the ideas here: Optimised proposal commitment scheme

Desired properties for a proposer/builder separated block proposal design

We will focus on five major desired properties:

- Untrusted proposer friendliness: there's minimal or no risk that a proposer will screw over a block builder, so block builders have no incentive to prefer proposers that have some off-chain reputation or personal connection to the builder (as that would favor large pools).
- Untrusted builder friendliness: there's minimal or no risk that a block builder will screw over a proposer, so proposers have no incentive to favor builders that have some off-chain reputation or personal connection to the proposer (as that would make it harder for new builders to enter the market). If deposits are needed to accomplish this, they should be maximally low.
- Weak proposer friendliness: the mechanism should not require proposers to have either (i) high bandwidth or other computational resources or (ii) high technical sophistication
- Bundle un-stealability: proposers should not be able to take bundles proposed by block builders and extract transactions from them to make their own bundles, preventing the block builder from earning a profit (and possibly harming them even further)
- Consensus-layer simplicity and safety: the mechanism should continue to be safe and ideally be covered by the same analysis as the existing block proposal mechanism from a consensus-layer perspective

Idea 1

1. Block builders make bundles and publish the headers of the bundles that they create. A bundle header contains a commitment to the bundle body (the intended block contents), the payment to the proposer, and a signature from the builder.
2. The proposer chooses the bundle header offering the highest payment (considering only bundles where the builder has enough balance to actually make that payment). They sign and publish a proposal containing that bundle header.
3. Upon seeing the signed proposal, the block builder that offered the included bundle header publishes the full bundle.

At this point, the fork choice rule has the ability to make one of three judgements (instead of the usual two, block present vs block absent):

- Block proposal absent
- Block proposal present but bundle body absent
- Block proposal present and bundle body present

Note that in the second case, the proposal still becomes part of the chain and, crucially, the block builder's payment to the proposer still processes (but the block builder does not get any fees or MEV themselves).

Analysis

Three of the five properties are fairly simple to show:

- The proposer receives the promised payment unconditionally, so bundles can't screw over proposers
- All three steps are very automated and low-bandwidth, so this satisfies weak-proposer-friendliness
- The proposer cannot see the contents of the bundles that they are signing, so this satisfies bundle un-stealability

Consensus-layer properties, and untrusted proposer friendliness, are more tricky. This design does change how the fork

choice works, increasing it to 3 options instead of 2, and it also means that the proposer is no longer the last actor in the game. Theoretically, one can reason that if fork choices are capable of making decisions, then this should be fine, but it's still a significant change with potential unknown-unknowns.

The proposer does not see bundle contents and cannot screw over block-builders by bundle-stealing, but they can use a much more subtle attack to grief block builders. They can publish their proposal near the end of a slot, ensuring that attesters (probably) see the proposal on time, but not giving the block-builder enough time to publish the body, so there would be a significant chance that the attesters do not see the body on time. This imposes a risk on block-builders, and gives them an incentive to favor trusted proposers. Additionally, it creates an opportunity by which a malicious majority can heavily penalize block-builders that it dislikes.

I see two families of approaches to mitigating this problem:

- Attesters have a 2s delay between the maximum time at which they accept a proposal and the maximum time at which they accept a body. This mostly solves the issue if you trust the attesters, though the fundamental issue that block builders have a risk of losing funds still remains. Additionally, it's not clear that it's incentive-compatible for attesters to vote in this way (though one could conceivably force them to wait by requiring them to attest to a 2-second-long VDF solution to the proposal)
- If a body does not get included, the proposer only gets half the payment (and the block builder only pays half). This makes griefing by the proposer costly, but it still ensures that griefing by the block builder continues to be costly (in both cases costly enough that you can generally trust even anonymous actors to not want to do it). For example, if a bundle has proposer fee 1 and block builder profit 1.05:

Honest behavior would lead to a (builder, proposer) payoff of (0.05, 1) Proposer or attester publishing to late, leading to a header-only block being accepted, would lead to a payoff of (-0.5, 0.5) * Honest behavior would lead to a (builder, proposer) payoff of (0.05, 1) * Proposer or attester publishing to late, leading to a header-only block being accepted, would lead to a payoff of (-0.5, 0.5)

Idea 2

1. Block builders make bundles and publish the headers of the bundles that they create. A bundle header contains a commitment to the contents, the payment to the proposer, and a signature from the builder.
2. The proposer chooses and signs a statement consisting of the list of bundle headers that they've seen.
3. Upon seeing that statement, the selected block builders publish their corresponding bodies.
4. The proposer chooses one of the bundle headers from the list they've pre-committed to, and publishes a proposal with it.

There is a new slashing condition that would eject and penalize any proposer who publishes a proposal that is not part of the list that they committed to in the same slot.

Note also that the list of bundle headers submitted by the proposer in step (2) could also instead be a list of encrypted hashes of bundle headers with each hash encrypted to the public key of the builder of that bundle, so that only the builder knows if they were accepted. This reduces DoS risks.

Analysis

Once again, three of the five properties are fairly simple to show:

- Proposers cannot steal bundles because they only see any bundle bodies when they've already restricted themselves to a finite set of existing bundle headers.
- There's no possibility of the builder-to-proposer payment happening without the full body being included, so proposers cannot cheat builders economically either.
- Consensus properties are the same as before, because the system is still a proposer-moves-last game and there's no change in what the consensus rules are deciding on

The two harder properties to ensure in this case are weak-proposer-friendliness and untrusted-block-builder-friendliness. The concern is that a malicious block builder can attack proposers by making a large number of proposals that all offer a very high fee, but never publish the body of any of them. If the proposer has a cap on how many bundles they accept, then this attack can price out all of the legitimate bundles, and leave the proposer with no bundles that they can legally include in their block. If there is no cap on how many bundles the proposer can accept, then this risks an unbounded number of full bundle bodies (think: 500 kB each) being sent to the proposer, an overwhelming amount of bandwidth requirement.

One solution to this conundrum is to rate-limit bundle header submission in some way that is not a hard limit.

- A fee for submitting bundles, which is adjusted through some EIP-1559-like mechanism to target some rate (eg. 8 bundles per slot)
- A deposit requirement for being a block builder (necessary anyway to ensure proposers get paid), together with a rule that if you publish a bundle that does not get included when a lower-priced bundle did get included, you cannot submit bundles for the next N slots

The fee itself could also be charged only in the case where your bundle does not get included but a lower-priced bundle does, as that's the specific situation in which you may have acted maliciously (or the proposer was malicious or the network was bad at the time).

There is some precedent for this; ENS auctions have a 0.5% loser fee to discourage people from making bids when they are clearly not going to win just to force up the amount that the winner has to pay.

However, these techniques risk introducing a trust requirement on the proposer, so they need to be done carefully and the penalty for failing to get a bundle included cannot be too high.

An alternative solution is to allow free and unlimited bundle body publication, but limit body propagation at network layer. One simple algorithm is:

- Add a slight delay for the minimum time at which bundle bodies can be propagated: 0s for the highest-paying bundle, 0.2s for the second-highest-paying bundle, 0.38s for the third-highest-paying bundle, and generally 2 * (1 - 0.9^{k-1}) seconds for the k'th highest paying bundle.
- Add a rule that a node does not propagate a bundle body if it has already propagated a higher-paying bundle body.

These two techniques can be combined together: you could have a slight fee to reduce the expected number of bundles to eg. 50 per slot, and then use network-layer mechanisms like this to reduce bandwidth requirements further.

Conclusions

So far I don't have a clear logical reason to believe that the above two approaches are the only families of solutions to the problem; there could be others. Out of these two approaches, idea (1) is conceptually simpler but it introduces risk to the block builder as well as more complex fork choice rule requirements. Idea (2) is simpler from the fork choice and consensus perspective, but it has challenges dealing with malicious block builder DoS and any solutions to this problem risk creating other problems as well, though this could conceivably be minimized. As of yet I'm still not sure which one is better.