Forward inclusion list

Resources:

- Vitalik's State of research post
- More extensive document about censorship-resistance, including some historical context, various crList alternatives and a different kind of censorship-resistance proposal

High-level idea:

We want to make it so proposers are able to combat censorship by forcing inclusion of some transactions, but we don't want to do it in a way which is bandwidth-intensive (i.e. can often result in redundant gossiping and/or block inclusion of transactions), which disadvantages altruistic proposers by requiring them to sacrifice MEV or which gives sophisticated proposers some power that they can exploit to extract more MEV than unsophisticated ones. When builders are not censoring, it would be ideal if the protocol would essentially behave as in the regular PBS proposal, with the proposers just choosing the most profitable block, and minimal overhead

The key to do this is to understand what censorship looks like after 1559, given a view of the mempool: we suspect a builder is censoring if there is blockspace which could be allocated to currently eligible mempool transactions and is instead left empty, i.e. if blocks are not as full as they could be given the current mempool. Obviously there is some degree of subjectivity given by the view of the mempool, but what we can do is let the proposer impose their view (or a slice of it) in advance, so that we can distinguish censorship from simple divergence of views of the mempool.

Rather than allowing proposers to directly force inclusion of their chosen transactions, we instead allow them to force builders to fully utilize the available blockspace: if they cannot do so, they must use the unused space for the proposer-selected transactions

Cost of censorship

For an overview about how the cost of censorship changes with PBS, if we don't assume the bribing model (where the whole validator set is bribable), see Vitalik's State of research post

crLists + EIP 1559 change this dramatically:

- On a timescale of a few slots, the cost of censorship is still linear in the number of slots for which you want to censor, but with a much better constant. Without crLists, a dominant builder only incurs the opportunity cost of not including the transaction, per slot. With crLists, they are forced to fill each block to its gas limit, which is many times more expensive
- On a longer timescale, the cost of censorship is exponential in the number of slots, because the basefee goes up and filling blocks becomes exponentially more expensive (and it prices out real transactions, leaving the attacker to fill entire blocks)

Design goals from Vitalik's post

- **DoS protection and no free data-availability**: the proposer (or the builder) should not be able to include data which no one pays for, as that could be abused by an attacker to bloat the chain or by rollups to cheat on transaction fees. For example, this means that if a block contains a transaction that turns out to be invalid because of insufficient balance or other transactions in the same block, the protocol must still charge the proposer base fees for it.
- **Minimal additional bandwidth consumption**: the mechanism should be efficient not just with on-chain data, but also with data in the p2p network. For example, having hundreds of redundant full bodies from different builders floating around the network is not realistic.
- **Does not re-introduce proposer centralization**: the whole point of PBS is that it doesn't require proposers to be sophisticated. We don't want to create a mechanism which reintroduces a benefit for proposers to be sophisticated and hence an incentive for proposers to enter into further extra-protocol auctioning relationships or join pools.
- **Ideally, allow proposers to be stateless**: validators being able to be fully stateless (onceVerkle trees are also included) would be a significant boon for further decentralization and scalability.
- **If we rely on altruism, don't make altruism expensive**: we generally can rely on assumptions that at least a few percent of proposers are altruistic and will ignore bribe offers and accept censored transactions. However, this assumption is less realistic if doing so is expensive in-protocol. Proposers should not have to sacrifice large amounts of revenue to help ensure censored transactions get in.

Natural language specification

Here we are assuming two-slot PBS, with even blocks being proposer's blocks and odd blocks being builder's block. A quick summary is that the proposer publishes a crList with their beacon block, and the list is relevant to the next builder's block, the one chosen by the following proposer. That this block is compliant is enforced by the attestors of that block, if crList has been timely published.

- The proposer at slot 2n publishes a list of currently valid transaction, crList, at the same time as the publication of their beacon block. The list is published to the p2p network, not in the beacon block.
- The builder's block in slot 2n+1 and the proposer's block in slot 2n+2 are in no way influenced by the crList at slot 2n, and neither is the prescribed attestors' behavior
- Attestors for slot 2n+3, the next builder's block, determine whether a crList is available at a deadline which is well in advance of the beginning of slot 2n+2, so that builders intending to make a bid for slot 2n+2/3 have ample time to see crList and make a compliant block. For example, we can set the deadline at the end of slot 2n, so builders have almost a whole slot to see it and make a compliant block.
- Builders for slot 2n+2/3 make their block compliant with crList if they have seen it, where compliant means that crList must not contain any transaction tx which is still valid after the block and such that$block.remaining_gas > tx.gas_limit$
- Attestors of slot 2n+3 vote against the builder's block if they have seen crList in time and the block does not comply with it.

Without PBS (e.g. in the post-merge status quo), we have an equivalent scheme without builders:

- Proposer of slot n publishes the list
- Attestors of slot n+1 enforce the list for the proposal of slot n+1.

Analysis

Consensus safety

Since we are adding an extra condition to some attestations (half of them, in fact), which introduces a new opportunity for views to diverge between attestors, we need to make sure that we are not weakening the stability of consensus. The impact here is thankfully minimal. Two attestors' views diverge because of this extra condition only in one case: a crList must have been published but only seen in time by one of them, and the next builder must have not made a compliant block. We can further identify two cases:

- Honest builder: the builder must have not seen the crList before making their bid, even though crList was published before the deadline (since some attestor saw it). Given good network conditions, this is highly unlikely. With network conditions bad enough for the builder (which we would even assume to be better connected) to not see the crList between the publication deadline and publishing a bid (almost a full slot if we set the deadline at the end of the proposer's slot), there's many simpler ways for views to diverge
- Dishonest builder: they could purposefully make a block which does not comply with crList after it has been published in a way which splits views. This would be effective at splitting the attestations to the builder's block, which could potentially result in the builder's block being excluded from the canonical chain. More importantly, a builder can always split views by publishing their block close to the publication deadline, so this does not give them any additional power.

| | >50% committee sees crList| >50% committee does not see crList|

| -------- | -------- | -------- |

| Builder sees crList| :slightly_smiling_face: | :slightly_smiling_face: |

| Builder does not see crList | :disappointed: | :slightly_smiling_face: |

Properties

More generally, this scheme has the following desirable properties:

- **Untrusted builder consensus safety**: a builder can already split views by publishing their block close to the publication deadline. crLists do not make it any easier
- **Untrusted proposer consensus safety**: publishing a crList late does not split attestations unless the builder participates
- **Untrusted proposer builder safety**: a malicious proposer cannot easily grief a builder:

    - if they publish later than the deadline, the builder only needs to rely on honest majority in their committee, which they anyway rely on.
    - if they publish before the deadline, the builder has almost a full slot to see crList. Furthermore, they are free to not publish if they don't see one and they have other reasons to believe that they are being attacked.

- **Minimal risk, and pushed to builders**: the risk is all on builders, and they can freely manage it by reducing their bids or not making any if they have reasons to believe that their body might be considered missing because of this extra attestation condition.
- **MEV Smoothing compatibility**: regardless of whether the crList has been published in time or not, all builders are able to make a compliant block. In the first case, they should all see the list on time, in the second case no condition is enforced.
- **SSLE Compatibility**: the proposer publishes crList and their block together, so they don't deanomize themselves before time
- **No free data availability**: the attestations to a builder's block cannot be used as a guarantee of availability of anything, as the block does not reference crList at all
- **Minimal additional bandwidth consumption**: the only added bandwidth is to propagate crList. This could even be further minimized by having the proposer publish a list of transaction hashes instead of the full list.
- **If we rely on altruism, don't make altruism expensive**: the proposer does not incur any cost for publishing crList, not even because of the slight added risk for builders and the possibility that they might claim a premium for that, simply because their crList comes after accepting a bid. The only caveat is that the proposers of two consecutive slots might be the same, but that will generally not be the case, and it especially won't be the case for home stakers, which are the primary "censorship resistance tools" we have.