Three dichotomies in ePBS

It has come to my attention that there are very much contending forces trying to shape the future, or even the existence, of ePBS. As such I want to collect here some thoughts that are purely my personal opinion and do not reflect any agreement neither from the Prysmaticlabs team nor any group I interact with. However, since I think some abstractions are useful to frame the problems I decided to write these thoughts in the form of an ethresear.ch quick note. This note highlights three dichotomies that are contending forces in the design space for ePBS, they are inherent to the problem and independent of design choices as long as some very broad assumptions on forkchoice are kept. None of the concepts in this lists are mine and they are all well known to people in the field. Below I will clearly highlight what I believe to be an abstract and formal aspect of the problem vs what is my personal opinion on the subject.

Background

For the purposes of this note I will call validator

the entity that currently resides in the BeaconState

, it has a BLS key that it can use to sign and broadcast attestations and from time to time it is assigned a slot to broadcast a full

SignedBeaconBlock

. By full

here I mean a consensus layer block that includes an ExecutionPayload

. We can abstractly think of a full block as a union of a SignedBlindedBeaconBlock

(that is one that includes only the ExecutionPayloadHeader

instead of the full payload) and an ExecutionPayload

. Validators are staked, they have typically 32ETH captive in the Beacon chain and they can be slashed if they misbehave. They are slashed automatically for certain offenses that are predetermined in the protocol, and they are always subject to after-the-fact slashing by the community if we, as a group, so decide. We will refer to the validator as a proposer

when it's it turn to propose a block.

A builder

is an entity that produces an ExecutionPayload

to be included by the proposer in its beacon block. This is loosely defined at this moment since it is not an entity that is controlled by the protocol in any way. The proposer can be itself a builder, producing its own execution payload, or it may rely on other entities to produce it for him. In the current status of the Ethereum network, builders are a reduced group of entities that are not controlled by the protocol, they are not staked in any way and may act in an anonymous manner. According to rated network, at the time of writing this, 90% of builder's blocks are produced by just 5 different entities

, while the top 3 share over 62% of the blocks built. In total there are 40 active builders.

A full beacon block is obtained by combining the product of the proposer and the builder, this can be any mechanism, from the naive one of both freely broadcasting them to some more complicated mechanisms either involving a trusted intermediary or trustlessly using the Ethereum network to be this intermediary. Currently, proposers and builders use relays

as intermediary. According to the same network above, there are currently 10 active relays and they broadcast 90% of the total

current beacon blocks in the network.

Enshrined proposer builder separation aims to remove the trust assumption on the relay, as a single point of failure on the network, by using the protocol itself as intermediary. For this, builders become entities that are accounted for in the beacon chain, like validators are, they can be staked or not, they can be subject to slashings or rewards/penalties, etc, just like validators are.

Why ePBS?

There are many reasons why the network may want to 1) avoid using relays and 2) register builders in the protocol. In the first case, relays are a single point of failure for the network, in the event that major relays are down (or there is a bug in the single software currently in use to communicate with them) blocks may be lost. There are safety valves built-in clients that

check for this situation and revert to local building of payloads, but still this leads in the best of cases to a poor user experience. This was the situation for example right after the Capella fork, had it not been for this built-in valve, many more blocks would have been missed. In the worst case of relays actively being malicious, they could cause reorgs by selectively deciding when to broadcast blocks and to whom. They could grieve both proposers and builders, etc. There are economics incentives for them not to do this, but the fact is that relays can act however they see it fit, without any penalties imposed by the protocol.

For 2) there are varying reasons and they depend on the technical implementations of ePBS. As a form of example, proposers have direct impact in what the rest of the network sees as the current chain tip. If they decide to submit their block at particular times to different peers, they can cause a split view between their own block and the previous head. A more naive way of trying to produce this split view is simply producing two different valid blocks. This is already accounted for in the protocol and it is a slashable offense (the reason is slashable is not because of the possible split view which is an annoyance, but rather to guarantee formal properties of FFG+LMD). When proposers are slashed, we wait an extended amount of time to check whether other validators are being slashed at the same time (pointing to either a bad bug in clients or an active attack by malicious validators) and the amount they are slashed increases with this number. Another reason for the wait time is to detect if there are other factors that are not accounted for in the protocol, that may lead the community to decide on a harsher penalty for these validators (or even others that may not have committed one of the slashable offenses). There are different designs for ePBS, all of them give more or less power to builders to produce such split views (between payload present or not for example) and some may give them powers to split views between contending forks. It is because of this reason that in all designs for ePBS (that I am aware of) builders are staked, ie. they maintain a balance captive in the beacon chain that can be reduced by automatic detection of certain offenses or by the community deciding so.

In the case that builders are staked, what mechanism of slashings for offenses do we have? in particular, what form of correlation should we apply with other slashings? This again will vary from design to design, but if we go with a system that mimics the current situation, in which 5 builders produce 90% of blocks, then correlating by number of builders slashed does not seem meaningful.

My personal opinion

is that builders should be staked, and because of the previous paragraph, since it seems reasonable to expect that the set of builders is and will always be a reduced set, that they should be very heavily staked, orders of magnitude more than what validators are. This unfortunately comes with a high price as we see below.

Three dichotomies

There are several different ePBS proposals out there, all with their own benefits and pitfalls. Without getting into the technical differences between them, and never entering in engineering decisions (that are very relevant) like P2P network stress considerations, I would like to stress some inherent dichotomies that follow from basic and broad assumptions.

Home stakers building vs relays bypassing ePBS

Since one of the main design choices in ePBS is how will the entity of a builder

be accounted for in the protocol, we are at the liberty of allowing or not validators to also be builders. If we allow validators to be builders, nothing prevents them from still using the current mechanism of relays, that is regardless of economical or rational reasons, proposers can always chose to use relays and build their full blocks without using the ePBS mechanism.

If we allow every validator (eg. home stakers) to be a builder then we need mechanisms to disincentivize the use of relays bypassing the protocol safeguards. My personal opinion

is that there should be a strong commitments of clients to remove any support for software like MeV-Boost or anything that facilitates the use of a relay if ePBS is in-place. Defaults are sticky and it seems very unlikely that large operators like Lido will send their node operators to use unaudited forks of clients instead of the protocol designed builder separation. There is currently a school of thought that clients should maintain MeV-Boost support to allow validators fast communication with a relay instead of the slower P2P network.

The other option is bleak in the eye of the public: the set of builders are not validators and if a home staker wants to produce its payload locally, then it needs to register as a builder. We will get to this option in the next dichotomy

Home stakers building vs Builders heavily staked

We have already argued that every proposal for ePBS has staked builders, the question is for how much? If we require a heavy stake on builders, then we are not allowing every validator to be a builder. Both sides of this dichotomy seem very bleak to me. There are several mitigations that can be placed still in this problem: proposers may be allowed arbitrarily large and strong inclusion lists for example, forcing their preferred transactions to be included, and in a prescribed order if they decide so. Proposers may still use a side channel like relays and searchers to construct these lists, or simply arranging with a builder to propose the payload, but whatever the system they chose, they cannot bypass the fact that the payload has to come from a heavily staked builder and if we find that builders are acting against the network, we can consequently heavily slash them. Validators may also keep their entire control over forkchoice this way, not guaranteeing any LMD weight to the

payload vote. This is the content of the PTC design of ePBS, which comes with less guarantees to builders as we will discuss below.

As I stated above, My personal opinion

is that builders should be heavily staked and therefore this rules out home stakers from building their own payloads. I believe with strong inclusion lists this solves any censoring concerns, with ePBS we can force builders to include these transactions or the network stalls. Attacks like the proposer splitting views as mentioned in the article above are heavily mitigated by the fact that a) searchers can always protect themselves from these attacks and b) the set of validators is independent of the set of builders which is expected to be reduced as it is today. In any case, the network only needs a single honest builder to be present to work well in a trustless manner, while this is not a new trust assumption (we have this assumption today on the set of validators=builders) we are strengthening it by putting this assumption in a reduced set.

Builders' safety vs Builder's forkchoice weight

The following is a property that a proposer building its local payload has in the current system: If I submit my block on time, extending the canonical head, and honest validators see it on time, it becomes the canonical head

. Builders also have this property as long as the relay does not grief them, that is, if they reveal their payload on time and the relay broadcasts the full block and honest validators see it on time, their payload will become the canonical head. This is a property that would be desirable for both builders and proposers to keep.

This property is a consequence of the way honest validators work, if they see a block arriving before a deadline (4 seconds into the slot) they attest for it. In fact we even give this block an artificial boost

in case there is a competing fork. Thus this guarantee comes from the fact that validators attest to blocks they see. Let us make the following assumption about any hypothetical ePBS system:

- proposers broadcast signed blinded blocks and validators attest to them
- builders broadcast signed execution payload and some staked entity

attest to them

- validators count these attestations to decide their view of the chain.

All ePBS proposals satisfy the above broad assumption. Let us call the first type of attestations CL attestations and the second payload attestations. For proposers to keep the above property we need therefore to make sure that if a proposer broadcasts its block and is seen by honest validators on time then its block becomes canonical. This means that CL attesters have forkchoice weight

in the sense that when given two contending options to vote for, their attestation will favor one or the other fork. This is currently enforced by the LMD-GHOST rules, but even if we move away from this system, the above remains true: in order for proposers to keep the desired property, CL attesters have forkchoice weight

in the sense described here. The same reasoning applies for builders, in order to maintain their guarantee the payload attesters need to have forkchoice weight

.

Now an immediate consequence of the above is that if the proposer (resp. the builder) keeps its guarantee, then itself has forkchoice weight, since it can influence directly the CL attesters (resp. payload attesters) by releasing its block (resp. payload) at particular times and to particular peers.

My personal opinion

is that we should never give forkchoice weight to builders, we see currently 5 builders responsible for 90% of blocks, their bearing on forkchoice should be zero. Therefore I believe that it is impossible to guarantee this desired property to builders.

This personal belief does not come from anything but the above described impossibility under so very lax and general assumptions. We should analyze however what are the consequences of the loss of this property. This means that a builder may reveal a payload, on time and acting in honest fashion, but its payload is not canonical. However, we can still keep the following formal property that is independent of the above:

If the builder reveals its payload for slot N and is seen by honest validators, then no other payload for slot N can be canonical

As long as this property is maintained, then builders can be assured that their transactions cannot be replayed in other blocks, and attacks like the low carb crusader are not viable. This of course may still not satisfy builders, they could be revealing information that otherwise would be kept private.

This last dichotomy is not such, at least it is not black or white as the previous ones in which if we require a higher stake from a builder, then by definition not all validators can be builders. But in this case we can give some

forkchoice weight to builders by regulating a parameter like proposer boost

to a minimum amount. My personal opinion

by the description above is that this parameter should be zero since builders, from the point of view of forkchoice security analysis, can be thought of as attackers controlling a large stake (they are guaranteed to get several blocks in a row statistically).

Conclusion

I believe that these three dichotomies are present regardless of any ePBS implementation and that we need to make a stand in all of them. The rest is obviously my personal belief:

1. Status quo or ePBS:

definitely ePBS, even with allowing validators to be builders and the relay to bypass the system, in the worst case we are where we are now. And if CL clients agree to remove MeV-Boost support we will default many blocks to either local execution or ePBS blocks by staked builders.

1. Home stakers building vs heavily staked builders:

I believe builders will always be centralized as they are now, therefore they pose a unique threat to Ethereum, as such I would want them to be heavily staked and invested in the protocol. I believe we need to give up local building in exchange for strong assurances that proposers will be guaranteed some control over the payload like strong inclusion lists and similar constructions.

1. Home stakers building vs bypassing ePBS:

By the above, by requiring builders to be heavily staked, we give up on single validator building and therefore we can guarantee that ePBS is not bypassed. Relays may still function and sidechannels may be set up to agree on a particular builder, but whatever is done, this builder will still be registered in the protocol and be heavily staked (and punished if it misbehaves).

1. Builders' safety vs Builders' forkchoice weight

: I believe a set of 5 entities producing all blocks should never have any bearing on forkchoice and we should rely on the already established, mostly decentralized, set of validators to decide the canonical head. Therefore I believe the property listed in the last section should be relaxed for builders.

The risk is that no builder may want to participate in such a network but, given that the network only needs a single honest builder, there is zero risk since any leak of information is irrelevant in a single-builder network.