Tl;dr

We argue that the dominant strategy encouraged by the proposer/collator distinction will simply be collators making self-proposals. Given the substantial additional protocol, engineering and security complexity introduced by the proposer/collator split, it would be expedient to abandon this distinction (for now, at least) and revise the proposed sharding roadmap accordingly.

Background

For definitions of the roles of proposer and collator, and also the roadmap outlining the phases mentioned below, see Sharding Phase 1 spec.

A key motivation for dividing the roles of proposer and collator is that,

the nodes participating in the transaction ordering process still need to have access to the state, and still need to perform state executions, because they need to know whether or not the transactions they accept will pay for gas.

The idea is that collators will not invest in maintaining state since they will be shuffled rapidly between shards. Instead, each shard will have proposers who maintain the state and use that information to construct collation proposals in which they "know whether or not the transactions they accept will pay for gas."

Economic model

Based on the quality of their proposals (i.e. the transaction fee revenue for the collation), and possibly other factors, proposers will offer their collations to the collator along with a bid amount. A four step "proposer–collator" game follows, both to protect the proposers from the theft of their proposals, and to protect the collator against data withholding by proposers. The likelihood is that the collator will accept the proposal that offers the highest bid, although it is not bound to do so, and may even choose to use its own proposal (a self-proposal).

From each proposer, n

, we have the equality,

$B_n = T_n - c_n - ε_n$

where

- B_n

is the bid amount for this proposal,

- T_n

is the anticipated transaction fee revenue to the proposer,

- c_n

is the cost to the proposer of maintaining state, finding the optimal transaction ordering, participating in the bidding process, etc. including the proportional costs of unsuccessful bids, and

- $ε_n$

is the proposer's profit.

The collator is likely to select the proposal from n

with the greatest B_n

.

On the face of it, this looks good: use of collation space is optimised by ensuring that it is filled with high value, non-invalid transactions (maximise T_n

); inefficient proposers are discouraged (minimise c_n

); and, assuming the market is competitive, proposers' profits are held in check (minimise $ε_n$

).

One potential issue is that, since $ε_n$

may be negative (a proposer may subsidise a proposal), it is possible for a proposer to censor transactions on a shard for arbitrary lengths of time. Eventually, the sender of a censored transaction may be able to increase the transaction fee so that a different, non-censoring proposer can win, but this is undesirable.

Asymmetries

A collator making self-proposals has two significant advantages over the proposers.

1. A proposer needs to bid an amount B

to the collator. To minimise the risk of losing funds it needs to ensure that (on average) $B < T - c$

. Thus proposers need to evaluate/estimate T

, the transaction fee revenue, in order to safely make an offer B

. For the collator making a self-proposal, the only risk is opportunity cost (missing out on potential revenue): the risk profile is not symmetrical; the collator is not at risk of losing funds.

1. The costs for proposers include the costs of preparing failed bids. If there are N

proposers on a shard, with cost c'

per period, all other things being equal the average cost of any successful proposal is $c = Nc'$

. A collator, on the other hand, can ensure that all of its self-proposals are accepted. If its costs are similar to the proposers' costs, c'

, and there is more than one proposer on a shard, then the collator has a significant cost advantage in always making self-proposals. This is relevant to the discussion of stateless clients below.

Potentially, on point 2, a proposer has an opportunity to recoup its costs for maintaining state by also acting as an executor, and charging fees for providing state to clients. It is hard to reason about this in the absence of an economic model for executors, which in any case are not planned until Phase 3 sharding, at the same time as stateless clients become available.

Challenging the assumption

The stated assumption in the above is that nodes ordering transactions need to know whether or not the transactions they accept will pay for gas. This is so that the transaction revenue T_n

can be maximised by proposer n

. The transaction revenue is maximised in two ways: by including transactions that offer the highest fees, and by ordering the transactions so that none are invalidated by being executed out of order.

However, consider situations in which c_n

can be minimised with little impact on T_n

.

Assuming a non-stateless client environment for the time-being, the main contributor to c_n

is likely to be the cost of maintaining state, which is an expensive operation, with costs for storage, computation and network bandwidth, not to mention administration time.

The non-full shard

In a situation where the shard is not full (the pending transaction queue is smaller than the collation size, which is the long-term desirable state), the proposer has an easy strategy: don't maintain state, simply include all the available transactions from the queue without checking them. (Note that part of the collator's work is to check the availability of previous collations; it is easy for the collator to eliminate transactions that have already been committed.)

The cost for this strategy, c_n

, is negligible. But how does the proposer know what to offer as the bid amount, B_n

? In general, this would be a problem. However, if the collator is the proposer

the bid amount does not matter. This is called a self-proposal. The collator can take all of $T - c$

as profit, with c

expected to be negligible.

Proposers cannot safely adopt this strategy

since without examining the state they cannot accurately estimate the transaction revenue and therefore the bid amount, B

. Thus the costs for proposers remain c_n

, remembering that this includes also the costs for failed proposals. Therefore, whenever $T > T_n - c_n - \varepsilon_n$

it is more profitable for the collator to make a self-proposal. In the non-full shard regime, the collator can assume that $T \approx Tn$

(the transactions are the same, only the ordering may differ), so it is always more profitable to make a self-proposal. This also has the benefit of side-stepping the complex auction process, while guaranteeing to the collator that the collation is available.

With regard to the ordering of transactions to maximise transaction fees, if it were found to be a significant factor, the collator could approximate the optimal ordering in its self-proposal by, for example, using the order in which transactions appeared in the shard's transaction pool.

The full shard

What about when shards are full? That is, there is a queue of pending transactions that is larger than the available collation space.

Let $T_{proposer}$

be transaction fees that can be earned by including transactions from the queue. The idea is that a proposer can maximise this by choosing to include the highest value transactions which are likely to be valid (and will therefore pay out). This maximisation process requires (approximate) knowledge of state, and comes at a cost c

. We will ignore the proposer's desired profit, ε

, since it doesn't affect the argument except in the case of malicious behaviour (negative ε

).

Let $T_{collator}$

be the expected transaction fees a collator can earn just by taking a heuristically sampled set of data from the transaction pool at small or negligible cost. Strategies for this which do not depend on state are outlined below.

If $T_{collator} > T_{proposer} - c$

then it always makes sense for the collator to make self-proposals.

Is it possible for a proposer to make $T_{proposer} - T_{collator} > c$

?

This seems doubtful (but see discussion on spam below). The cost c

of maintaining state is likely high, and also includes the costs of bidding for collations in which the proposer is not the winner.

In contrast, the collator has a number of possible approaches for selecting profitable transactions that do not depend on knowledge of state and have negligible cost. For example,

- Simply grabbing a collation's worth of transactions from the transaction pool.
- Using heuristics based on the kinds of transactions that have paid out in the past. The collator does not need to maintain state for this, it can consult executors to check past outcomes on a particular shard.
- Light-weight machine learning based on the above as a negative filter for spam transactions and positive filter for high-value transactions.

It is not at all clear that the state-maintaining proposer can maintain any long-term net advantage over collator self-proposals.

The one guaranteed way for a proposer to be able to make a bid that is more attractive than a self-proposal is for it to subsidise the bid. It may do this for legitimate reasons (e.g. to guarantee participation in an ICO) or illegitimate (censorship of transactions). The latter is undesirable; the former introduces a level of complexity and non-transparency by having two economic factors that affect transaction inclusion: transaction fees and proposal fees.

High-spam regimes and stateless clients

A danger of being undiscerning in what is included in collations is that it could effectively become free to add data to a shard's blockchain, which may end up seeing shards significantly abused. The proposer/collator model allows spam transactions to be filtered by the proposer. Without knowledge of the state it is difficult for a collator to reliably identify and remove spam transactions.

This may be the most significant argument in favour of the proposer/collator split. At a certain level of spam transactions it becomes hard for the collator to maintain its economic advantage over the proposer, and a market for proposers can then develop.

However, this is only likely to be an issue in Phase 2 sharding. In Phase 1 sharding, there is no concept of a "spam" or invalid transaction, and in Phase 3 onwards as currently conceived there is a proposal for a "stateless client". In fact, an original motivation for introducing stateless clients was to allow rapidly shuffled collators to be proposers (see point #6

of The Stateless Client Concept).

In the stateless client model, we can assume that the costs to proposers and the costs to collators of preparing high-value, low-spam collations is roughly the same, c'

, except that proposers have to compete with other proposers, so that their costs are Nc'

on average where there are N

proposers per shard. This model degenerates either to there being only one super-efficient (or malicious) proposer per shard, or to the collator making self-proposals and keeping the marginal proposer profit ε

for itself.

As an alternative to implementing the proposer/collator split, it may be better to bring forward the development of the stateless client.

Hidden transactions

Apart from censorship of transactions, there is another scenario in which proposers could maintain an advantage over the collator (and other proposers). This is where a proposer includes (temporarily) hidden transactions that are not broadcast to the network. It is not clear that this behaviour should be encouraged.

Conclusion

We have argued that asymmetries exist between the risks and costs to proposers and the risks and costs to collators making self-collations. These lead to the proposer/collator model degenerating to collators simply making self-proposals in all scenarios expect for stage 2 sharding in the case that shards are over-capacity with many low-value/spam transactions.

In view of this, we would recommend removing the proposer/collator distinction and making all proposals self-proposals by the collator.

Benefits of this are,

- Substantially reduced protocol complexity through removal of the four step bidding process.
- Reduction in the number of actors, leading to
- Vastly reduced development, testing and implementation effort
- Reduced crypto-economic "attack surface"
- Simpler protocol analysis.
- Vastly reduced development, testing and implementation effort
- Reduced crypto-economic "attack surface"
- Simpler protocol analysis.
- Elimination of the opportunity for proposers to censor or hide transactions.

- Elimination of the wasteful use of resources by proposers all independently creating proposals of which only one is selected.
- Reduction the number of actors who need to have their private keys available (and therefore vulnerable) to sign proposals and collations.

It is our hypothesis that the network will degenerate to this proposer-less situation in any case.

Ben Edgington, Nicolas Liochon - ConsenSys/PegaSys