This summarises our current thinking on the use of consensus mechanisms. We are currently not focusing on consensus so the thoughts will be quite high level.

Old discussions happened in this thread

Why Do We Need Consensus At All?

- DA for CR:

Since brewers can do things like block network traffic, we need a way to attest to the “state of the mempool.” A clear use case that illustrates this need is brewers censoring bids in an auction

- MEVM state:

The contracts with which users interact must be made public and updated according to specific rules.

- Security Information:
- Information for validating attestiations (MRENCLAVE)
- enforcement of rules for updating the above information (e.g. voting/delays for MRENCLAVE switch)
- registry of valid kettle keys (and potentially endpoints)
- Information for validating attestiations (MRENCLAVE)
- enforcement of rules for updating the above information (e.g. voting/delays for MRENCLAVE switch)
- registry of valid kettle keys (and potentially endpoints)

We have typically thought of the DA use case as being served by (a) bulletin chain(s) and the MEVM and security information living on “SUAVE chain”

Desired Properties

Requirements for consensus performing DA

- The key property is that it must be hard to censor transactions. We primarily have thought of this in the honest-threshold setting: “A transaction is guaranteed to be included in the next block if at least \frac{1}{n}

validators receive the transaction before time T, given a byzantine threshold of f” * The aim is to maximise n

and f

- When we say “block” we do not require there to be consensus on a total ordering of transactions, only agreement on which transactions are in the “block”. Transactions don’t need to be executed.
- The aim is to maximise n

and f

- When we say “block” we do not require there to be consensus on a total ordering of transactions, only agreement on which transactions are in the “block”. Transactions don’t need to be executed.
- Economic designs that give a similar property to the one above in a rational setting are desirable, but seem less tractable and so are not a priority
- light-node friendliness (or very light full nodes)
- Single-slot finality
- Low latency

Requirements for MEVM State

- Single-slot finality
- Light clients
- …?

Requirements for Security Component

- Censorship resistance would be nice
- Definitely light-client friendly
- …?

Do We Actually Need Our Own Chain

DA:

for the DA use case the answer seems clearly yes, because there are no other existing chains that meet out requirements

Security module and MEVM:

its less obvious at this point of time whether we need our own chain or can use another.

- if we expect significant MEV accrual then it makes sense to run our own chain
- influence of another chain's governance over our application might not be acceptable
- means to monetise
- running a chain is expensive and requires more engineering effort

Implementing

Dan wrote this nice doc that starts exploring some implementation options for "bulletin chains" which touches:

- Narwhal
- Heterogeneous Narwhal
- Hierarchical Consensus

Other options we have yet to explore:

- CometBFT (multiplicity edition)
- Themis without the FCFS properties
- BigDipper (seems promising but unclear what the lift is to implement/what tooling we could tap into)