Bitcoin Cash: a short-term data availability layer for ethereum?

Sharding

vbuterin 1 Jul '19

Conceptual background: Layer 2 state schemes 195

Summary of conceptual background: there's a large space of quite powerful and effective scalability solutions that rely on a non-scalable *computation layer* (ie. the current ethereum chain suffices) plus a scalable data layer. The general principle of these techniques is that they use interactive computation techniques (eg. Truebit 82) to compute the state on the ethereum side, crucially relying on the data availability verification guaranteed by the data layer to make sure that fraudulent submissions can be detected and heavily penalized. One could use techniques like this to make a highly scalable general-purpose EVM-like system.

In the longer term (1+ year out) the scalable data layer is going to be ethereum 2.0, because its planned 10 MB/sec data throughput is much higher than that of any existing blockchain. In the shorter term, however, we can start working on these techniques *immediately* by using existing blockchains, particularly those that have lower transaction fees per byte than ethereum, as the data layer. Bitcoin Cash arguably fits the bill perfectly for a few reasons:

- High data throughput (32 MB per 600 sec = 53333 bytes per sec, compared to ethereum ~8kb per sec which is already being used by applications)
- Very low fees (whereas BTC would be prohibitively expensive)
- We already have all the machinery we need to verify Bitcoin Cash blocks inside of ethereum thanks to http://btcrelay.org/ 146 ; we just need to repoint it to the BCH chain and turn it back on. Verifying BCH blocks is also quite cheap compared to eg. ETC blocks
- The BCH community seems to be friendly to people using their chain for whatever they want as long as they pay the tx fees (eg. https://memo.cash 118)

The main weakness of the BCH chain is its 10 minute block time. This seems unlikely to change unfortunately. However, there is active interest in the BCH community on strengthening zero-confirmation payments using techniques like Avalanche pre-consensus 120 . If these techniques become robust for the use case of preventing double-spends, we could piggy back off of them to achieve shorter finality times as follows. On the Ethereum chain, we randomly select N "proposers". We incentivize people to send small BCH transactions to these proposers. We require in our layer-2 protocols that for data on the BCH chain to be valid, it must include as one of its inputs a particular UTXO that sends a small BCH payment to them. This way, once a proposer publishes a transaction, if BCH's anti-double-spend machinery works it will prevent that transaction from being replaced. Though this technique may be too complex to implement in practice, and we may want to just settle for being okay with 10-minute block times for a full general-purpose VM until eth2 comes out.

Skip to main content