Just like we can have applications that run under alternative execution engines (cf@JustinDrake's recent ideas) that have different tradeoffs and properties from the main execution engine, we can also have applications individually choose their block proposal mechanisms.

Anyone can create a second-layer proposal mechanism, that consists of two parts. The first part is an off-chain layer consisting of various participants that actually performs block creation. The second is an on-chain contract which is capable of interpreting and verifying the output of the first layer, and then forwarding it - that is, the transactions sent to the off-chain layer would be of the form [to, data]

, and the contract would make a series of internal transactions (aka calls), sending the desired data to the desired recipient for each transaction that the proposal mechanism accepted, in the order that the proposal mechanism specifies.

The proposal mechanism would be able to cryptoeconomically commit to including transactions potentially much faster than the block time. Additionally, to preserve censorship resistance a mechanism could be added where a user can call the contract with a transaction, and the proposal mechanism would be required to include it within some number of blocks. Alternatively, commitments could be based on internal order: the proposal mechanism could safely make an absolute cryptoeconomic commitment to include some transaction T before including any other transactions outside of some list of transactions that have been committed to already.

Unlike Plasma, this does not

improve scalability, but it does improve latency, as well as front-running resistance. This could be useful for on-chain decentralized exchanges, auctions and other highly time-dependent systems. That said, Plasma itself can also be used as an engine for experimentation in alternative proposal mechanisms with similar properties.