I'd like to propose a thought experiment/protocol, based on proposed data availability schemes such asthis. Some have previously argued that a blockchain is primarily a proof-of-publication system; i.e. once you publish some data on it, you as a full node know that it is available, at least to the participants of that system following the protocol rules correctly. However, if it's possible to convince yourself that data is available without running a full node, then this leads to some interesting consequences, such as creating a cryptocurrency where full nodes only need to do O(log(n)) operations per block, where n is the number of transactions in a block.

Blockchains that push all the transaction validity consensus rules to clients that care about specific transactions, rather than all the full nodes, have been proposed before, but here we propose one where full nodes only need to do O(log(n)) operations per block, with the help of data availability proofs.

Suppose we have a UTXO-based blockchain where full nodes do no validation on transactions whatsoever; the blockchain is purely there to order transactions. Additionally, this blockchain has the structure of an Certificate Transparency style Merkle tree (aka Merkle Mountain Ranges). This means given two trees A and B, you can prove that the two trees are consistent (i.e. tree B has all of the elements of tree A, plus some additional elements, and thus proving the append-only property of the tree) in O(log(n)) time where n is the number of elements added to tree A to make tree B. Suppose that each block simply contains a) a Certificate Transparency style Merkle tree containing every single transaction in the blockchain, b) a merkle consistency proof to show that transactions were only added to the tree in the last block, not removed and c) the data of the new transactions in the block.

The last one, c), does not need to be downloaded by full nodes, and that node will still be as secure as any other node, in that they cannot be made to accept blocks that are invalid according to the consensus rules through a 51% attack, because the transaction data is not a part of the consensus rules.

Because invalid transactions and double-spends are allowed on this blockchain, we have an implicit, client-side, rule for determining which transactions are valid; and any other client following the same rule will consider the same set of transactions to be valid. We say that a given transaction on the blockchain is valid if a) it meets all the standard transaction validity rules (good signature, enough balance, etc) and b) it is the first transaction on the blockchain to spend all the UTXOs it references as inputs.

One of the properties of a full node is that it does not need to trust anyone to determine the network's valid chain according to the protocol rules (trustless), but traditionally they need to process and verify all the transactions, an O(n) operation for n transactions per block. However, here we have proposed a type of full node that can still trustlessly determine the network's valid chain, but with only O(log(n)) operations for n transactions per block. Note we can still enforce blocksize limits if we want to (in the form of number of transaction per block), as we can calculate the number of transactions added to each block from the merkle consistency proofs.

Of course, the system is only useful if we can actually use this as a cryptocurrency, which means we need a way for nodes to figure out which transaction was the first to spend a particular UTXO, to prove that a payment was successful. But this is only possible in the first place if the data in each block is actually available, because a miner can mine a block, withhold the block's transaction data, and release the data later, and that block may contain transactions that spent certain UTXOs before transactions in future blocks. So nodes have to rely on data availability proofs to make sure that block transaction data is available, before accepting each block.

For nodes to figure out with transactions were first to spend certain UTXOs, we can have a new type of node, an archival node, which nodes can query to ask "Which was the first valid transaction to spend this UTXO?", and the archival node responds with a signed reponse.

To disincentivise archival nodes from lying, they can deposit some money in a contract, such that they lose their deposit, upon proof that the archival node responded with a certain transaction, when there is evidence that there is in fact an earlier transaction spending a UTXO, and the fraud prover is rewarded with some money. This proof itself should also be backed by a deposit, in case there is an even earlier transaction than specified in the proof. This deposit should be equal to the amount of money transfered by the transaction being referenced, plus some more for the fraud proof reward. Archival nodes can also be further positively incentivised to run such nodes in the first place, by people paying for their service. Therefore we end up with a cryptoeconomic incentive model where it is never profitable for an archival node to lie about what transaction was first, because if they try to do so to trick a user that a money has been transfered when it hasn't, then the archival node will lose the same amount of money that they lied about anyway. If the transaction is transferred too much money such that it is unlikely for any archival node to have that amount of money, the user can probably afford to run their own archival node.

The cryptoeconomic model proposed is just one of several models to let full nodes discover valid transactions. Other possibilities include solutions such as VerSum, where a full node asks many archival nodes, and only one of them need to be honest.

However, the key to all of this of course, relies on how good the underlying data availability scheme is. If you download the data yourself (i.e. traditional full nodes), then you're 100% sure the data is available. If you use a data availability scheme based on i.e. erasure coding, it may be more like 99.99%. Does the 0.001% make all the difference in the confidence of a node to trust that a transaction is valid?