The goal of this post will be to illustrate some of the challenges in making a two-way bridge between eth1 and eth2 (eg. to support two-way convertibility of ETH), and how it could be implemented.

An eth1 -> eth2 link already exists as part of the eth2 proposal, and is necessary to allow deposits to happen. This link is implemented using the [eth1data

voting mechanism](https://github.com/ethereum/eth2.0-specs/blob/fffdb247081b184a0f6c31b52bd35eacf3970021/specs/core/0_beacon-chain.md#eth1-data). Note that this mechanism assumes that PoS validators are an honest majority, and that the PoW chain does not get attacked (specifically, that it does not revert more than ~5 hours); if either assumption fails, then the two chains would no longer “agree” with each other. There is an implied “social contract” at least at the beginning that if either case happens this would be remedied, more likely via a soft-fork of the PoS chain, though if the PoW chain really does revert more than 5 hours then a community agreement that the attack chain is illegitimate is also quite likely. Note that, in either of these cases, it is not possible for a failure of the PoS chain to necessitate a soft-fork of the PoW chain.

If we want the eth1 chain to be aware of eth2 state (a prerequisite to allow ETH to move back from eth2 to eth1), there are two ways to do this. One is to have the PoW chain contain a light client of the PoS chain. The other is to have PoS finality also finalize the PoW chain. The latter could be done by adding a mechanism where if a PoS block B_S

includes a reference to a PoW block B_W

via eth1_data

voting, and B_S

is finalized, then B_W

is also treated as finalized. However, this implies that PoW miners (and clients) also need to be running an eth2 implementation so that they know what eth2 chains are finalized.

[

TwoWay2

1208×231 15 KB

](https://ethresear.ch/uploads/default/original/2X/1/17b77da4169642ffd40e156fe47966d1d6f5c7a1.png)

[

TwoWay

1088×309 15.7 KB

](https://ethresear.ch/uploads/default/original/2X/4/40bab01f90c5131362a26e702311fa982ef660b4.png)

The former requires an eth2 client implemented inside of eth1. This would require either webassembly or native support for BLS-12-381 verification, neither of which are currently expected to happen soon. Additionally, it only provides a light-client level of security.

The latter is more interesting, because it gives eth1 a “native” form of reversion limitation (this is often called the “finality gadget proposal”). Note that this proposal does something different from the first, in that while it does make the eth1 fork choice

aware of eth2, it does not immediately make eth1 aware of eth2 state

. For instance, note that it’s theoretically possible for two competing eth2 chains to finalize the same eth1 block (this would imply eth2 has broken, but it is still theoretically possible). More commonly, there could be two eth2 finalized blocks where one is a child of the other, both of which support the same eth1 block, and some miners could be aware of the more recent of the two eth2 blocks and the others not aware. This is not a problem for “eth2 as finality gadget” but it does mean we need more infrastructure to make eth1 explicitly aware of eth2 block state for the purposes of allowing withdrawals from the deposit contract.

One possibility is to simply create an eth2_data

voting mechanism inside eth1; essentially, replicate the same mechanism as used to make eth2 aware of eth1. This could be combined with the above to ensure consistency: eth1 miners would only vote for eth2_data

blocks if those blocks are (i) finalized and (ii) reference in their eth1_data

blocks that are ancestors of the eth1 block that the miner is building.

Challenges

Both of these proposals would require eth1-side changes. Currently, the eth2 roadmap has zero eth1-side changes before "the final transition". Both of these proposals would require emergency remedial action on the eth1 side if the eth2 side breaks. The latter proposal would require all eth1 miners to also be running an eth2 node. Hence, while both proposals are absolutely feasible, they are not something that should be implemented quickly.

However, as eth2 continues to run and proves its resilience, then there certainly comes a point at which implementing such a bridge makes sense. To reduce risks, there are a few things that could be done:

- Running eth2 voting on eth1 with a one-week voting period, to allow time for human intervention if things go wrong
- The eth1 chain becoming aware of eth2 finalized blocks via light client could also have a one-week delay before withdrawal for similar reasons
- Only turn the bridge on when deposited stake is high enough (eg. >5 million)
- Set the voting threshold higher than 50% (eg. 80%); bias the system toward not including any eth2 blocks unless there is strong agreement in their favor.