The write up focuses on one of the approaches to two-way bridge. Vitalik mentions that a replication of Eth1 data voting maintained by miners could be a source of eth2_data

for Eth1.

This article investigates the major properties of two-way bridged system driven by miners voting rather than in-depth technical details like a particular approach to Eth1-side storage of eth2_data

. It also assumes Eth1 clients use a full version of beacon chain client.

For the sake of simplicity, beacon chain data placed on Eth1 is denoted as eth2_data

whatever structure and fields it would have.

Prerequisites

- On the way to Eth1 finality: On the way to Eth1 finality

Copy of Eth1Data poll

Miner maintains a beacon chain client and queries it for eth2_data

upon block creation. A reference point for the query could be timestamp of a block at the beginning of the voting period. An eth2_data

sample that collects BLOCKS_PER_ETH2_VOTING_PERIOD * VOTING_MULTIPLIER

votes is getting published on Eth1.

Miners are allowed to vote for eth2_data

published previously to cope with failures of various kinds, whether they are caused by malicious behavior or by software errors.

Picking up parameter values

Voting threshold set to BLOCKS_PER_ETH2_VOTING_PERIOD/2 + 1

wouldn’t work for the period of any length. As it would reduce an adversary power limitation by up to 10%

and would make bridge vulnerable to 41%

attacks (exact value depends on the length of the period but never reaches desirable 51%

).

To make this solution viable, we have to favor safety over liveness and pick satisfying VOTING_MULTIPLIER > 1/2

. Parameters calculation comes up with the following values (link to the calculation sheet).

Parameter	Value
BLOCKS_PER_ETH2_VOTING_PERIOD	1024
VOTING_MULTIPLIER	5/8
VOTING_THRESHOLD	640

Given parameter values, we can outline meanings of properties that Eth2 voting process got if it would follow this scheme.

Property	Value

Safety Fault Tolerance	50%
Liveness Fault Tolerance	30%
Chance of Safety Violation	5.73*10-16
Chance of Liveness Violation	1.19*10-7
Blocks to finality	1069..2953 blocks* (4.2 to 11.5 hours)

*

Blocks to finality is a sum of BLOCKS_PER_ETH2_VOTING_PERIOD

with values from the previouswrite up where 45

and 1929

are evaluations of the new proposal and the status quo, respectively.

To reduce risks, Vitalik suggests setting the length of the voting period to one week. The table below shows the meanings of properties for the one-week voting period.

Property	Value*
Safety Fault Tolerance	50%
Liveness Fault Tolerance	45%
Chance of Safety Violation	~0.0
Chance of Liveness Violation	~0.0
Blocks to finality	1 week

*

Values are based on 1 week tab ofthe calculation sheet.

Drawbacks

- Time to finality limitation.

A thousand blocks to finality seems to be the best that this solution can give. The reduction of this number either reduces safety and liveness fault tolerance or increases the chance of failure of one or the other. You may check the calculation sheet for details.

- Liveness weakening.

Liveness fault tolerance of the voting process is less than Eth1 fault tolerance.

Instant finality

Initially proposed by @nrryuya here.

Instead of passing through a voting process Eth1 starts to invalidate blocks containing invalid eth2_data

. Validity conditions for the data are as follow:

1. The checkpoint must be finalized and belong to the canonical view of beacon chain.
2. Eth1 block must belong to the canonical view of Eth1 chain.
3. The checkpoint must be either equal to or descendant of the previous one.
4. Eth1 block must be either equal to or descendant of the previous one.
5. Beacon chain validators must vote for Eth1 block only if it contains eth2_data

satisfying condition 1.

It worth noting that condition 1 opens a room for racing where Eth1 block could be deemed invalid due to Eth2 network latency. It will likely mean that Eth2 experiences networking issues. Possible mitigation would be to take a distance from beacon chain head.

Safety and Liveness

For the rest of this section, we assume safety and liveness of Eth1Data

voting are tight with corresponding beacon chain properties. As we'vediscussed previously, the status quo doesn't hold this. But there is a solution addressing that problem.

If a client follows the protocol supporting the above conditions, then the following statements are true:

1. Neither finalized checkpoint nor finalized block can't be reverted on Eth1 chain.
2. There is no possibility to finalize two conflicting Eth1 blocks as long as beacon chain has no safety failure.
3. With honest majority of miners it is possible to keep eth2_data

up-to-date as long as beacon chain has no liveness failure.

Statement 1 directly follows from conditions 3 and 4. To prove statement 2, suppose two conflicting Eth1 blocks were finalized. Two cases are making it possible:

- beacon chain validators polled a pair of inconsistent Eth1 blocks (Figure 1a)
- blocks are finalized by controversial beacon chain forks (Figure 1b)

The first case means the broken Eth1Data

voting process. The second implies that the canonical view of beacon chain contains two conflicting finalized checkpoints. Either of these two is a safety failure in beacon chain and contradicts the second part of statement 2.

While the first case can be remedied by condition 4, the second one looks more severe since it could cause a split in Eth1.

[

3312×1200 191 KB

](https://ethresear.ch/uploads/default/original/2X/3/3115165bfddec0f9d574904619374ce97821bfa9.png)

Liveness statement (statement 3) can be proved in a similar way to statement 2. If eth2_data

hadn't been updated for a significant amount of time, it would mean that either Eth1Data

voting was stuck or Eth1 liveness was violated. Both the former and the latter contradict statement 3.

Instant finality approach has two options affecting the security of the bridged system:

- Only miners run beacon chain clients.

In this case, regular users can’t check conditions 1 and 3 and are sharing security between honest majority of miners and beacon chain. Therefore, making issuance reduction unattainable.

- All Eth1 users run beacon chain clients.

Users running beacon chain clients can’t be forced through incorrect beacon chain forks if they check the conditions. According to this, beacon chain becomes the only supplier of Eth1 security.

These two options are not mutually exclusive and could become a pair of subsequent milestones within the same roadmap.

Inconsistent reads

Suppose there is a split in beacon chain, and there are two forks in Eth1 supporting a pair of corresponding beacon chain forks. Then the following scenario is possible:

1. Beacon chain fork BC_1

processes a portion of deposits from Eth1_1

.

1. BC_1

switches to Eth1_2

.

1. One may do a sequence of 1 ETH

deposits on Eth1_2

to increment the deposit counter.

1. After that, deposits from step 1 can be replayed on BC_1

via Eth1_2

fork.

[

2856×1324 175 KB

](https://ethresear.ch/uploads/default/original/2X/8/850cf7fdb4569b34ac084a33603fc6a952a1ce99.png)

Conditions 2 and 5 establish a one-to-one relationship between particular Eth1 and beacon chain forks, preventing inconsistent read and bounce attack scenarios.

Software errors

One of the potential problems with this scheme lies in the field of software errors. Suppose there is a consensus break between two major beacon chain clients, which means that the canonical view of beacon chain differs for those two. Then miners relying on this pair of clients would disagree about their view of Eth1 chain. Which, in turn, reduces the pace of honest blocks in Eth1 and increases its vulnerability to 51% attacks.

One could describe a simple model representing the influence of beacon chain software errors on Eth1 safety and liveness. Eth1 guarantees safety and liveness if $(1 - \gamma)(1 - \delta)\alpha > \beta$

, where α

and β

are total mining rates of honest and malicious miners, respectively, $0 < \delta < 1$

is a parameter proportional to network latency. And $0 < \gamma < 1$

is the probability of invalidating Eth1 block due to the failure of beacon chain software, γ

is proportional to the number of different clients used by beacon chain validators and inversely proportional to their maturity.

One of the approaches to reduce γ

is to set up a cluster of beacon chain clients and follow a conservative strategy of updating eth2_data

only when the majority of cluster participants agree on the new portion of data. This approach will work with honest miners. But in a more realistic rationale majority model, there is a centralization risk caused by replacing locally maintained cluster by a public provider of eth2_data

.

Drawbacks

- Cost of issuance reduction.

The cost of the desired issuance reduction is to obligate regular users to run beacon chain client software.

- Centralization risk.

Rationale miners and regular users tend to use public services to query beacon chain data.