TLDR

—We suggest adding to the beacon chain a simple price feed service tracking a small set of key assets. The service allows building fully decentralised oracles that produce a price for every tracked asset at every epoch boundary, i.e. with a granularity of 6.4 minutes.

Thanks to @albert, @benjaminion, @dankrad, @danrobinson, @DCinvestor, @djrtwo, Eric Conner, Evan Van Ness,@karl, @khovratovich, Mehdi Zerouali, @mkoeppelmann, @paulhauner, @protolambda, @Robert, @ryanseanadams, @sassal, @scott_lew_is, @vbuterin for feedback and discussions

Construction

A price_data: PriceData

field is added to [BeaconBlockBody

](https://github.com/ethereum/eth2.0-specs/blob/6474218fb162424a9c362de7d114e68a24059877/specs/phase0/beacon-chain.md#beaconblockbody) where

PRICES_PER_ASSET = 8

Type aliases

Price = uint64 AssetPrices = Vector[Price, PRICES_PER_ASSET]

class PriceData(Container): # Currencies in the IMF SDR basket usd: AssetPrices # 0.01 USD price denomination eur: AssetPrices # 0.01 EUR price denomination cny: AssetPrices # 0.01 CNY price denomination jpy: AssetPrices # 1.00 JPY price denomination gbp: AssetPrices # 0.01 GBP price denomination

From the point of view of Eth2 consensus the price_data

object is treated similarly to the graffiti

string. That is, price_data

can be arbitrary data whose value is ignored by the beacon chain state transition function and fork choice rule.

Honest beacon block proposers must populate best-effort prices for every tracked asset in PriceData

. The prices must:

- baseline

—represent 1 ETH of the tracked asset

- snapshots

—be snapshotted at the latest PRICES_PER_ASSET

epoch boundaries (ordered increasing by age in the AssetPrices

vector)

- denomination

—be denominated in the smallest ISO 4217 currency unit (e.g. if 1 ETH is 213.09 USD set to Price(21309)

)

- rounding

—be rounded with half down rounding (e.g. if 1 ETH is 22,732.66 JPY set to Price(22733)

)

- overflow

—be set to Price(2**64 - 1)

in case of uint64

overflow

- underflow

—be set to Price(0)

in case of negative prices

Example—honest-majority ETH-USD oracle

One can build an honest-majority ETH-USD oracle with a price at every epoch boundary. Consider a smart contract that takes the median of all prices corresponding to a given epoch boundary (up to SLOTS_PER_EPOCH * PRICES_PER_ASSET = 256

price points).

If more than half the corresponding beacon blocks are produced by honest validators the median price is secure, i.e. not manipulatable by dishonest validators.

Basic features

Below are basic features of the price feed service:

- consensus minimalism

—By treating price_data

the same as graffiti

both the beacon chain state transition function and fork choice rule are minimally impacted.

- subsidised delivery

—Every beacon block contains a dedicated PriceData

object, bypassing application-layer block space constraints and gas markets.

- raw prices

—The price feed is a low-level service that exposes raw price points. Those can be mixed and matched with other price sources to create hybrid oracles with custom logic.

- bandwidth efficiency

—Each tracked asset serialises to type_byte_length(uint64) * PRICES_PER_ASSET = 64

bytes for an overhead of 320 bytes per [BeaconBlock

](https://github.com/ethereum/eth2.0-specs/blob/6474218fb162424a9c362de7d114e68a24059877/specs/phase0/beacon-chain.md#beaconblock). The bandwidth overhead for block propagation is marginal, especially with network-level compression.

- public good

—The service is an unopinionated public good which can be used by Eth1, Eth2, as well as other blockchains. It can also be used offchain, e.g. by fully decentralised wallets to estimate fiat-denominated gas costs.

- initial deployment

—The price feed service is optional Eth2 infrastructure which can be deployed when ready. Eth1's DeFi ecosystem could benefit from an deployment, e.g. in phase 1.

Feed governance

Maintenance of the set of tracked assets is a governance issue for which we suggest the following social norms:

- minimum viability

—The set of tracked assets should be kept to a minimum viable, i.e. only include assets important to Ethereum's long-term health and success. The five initial assets USD, EUR, CNY, JPY, GBP reflect three key principles: * credible neutrality

—they are the currencies from the IMF'sspecial drawing rights (SDR) basket

- high liquidity

—they are the fiat currencies most traded with ETH

- broad utility

—they are major fiat currencies natural for DeFi (e.g. for decentralised stablecoins to peg against)

- credible neutrality

—they are the currencies from the IMF's special drawing rights (SDR) basket

- high liquidity

—they are the fiat currencies most traded with ETH

- broad utility

—they are major fiat currencies natural for DeFi (e.g. for decentralised stablecoins to peg against)

- rough consensus

—New assets (such as gold, oil, Bitcoin) could be added to the tracked set through hard forks driven by rough consensus.

- backward compatibility

—Once added assets must not be removed from the tracked set to maintain backward compatibility.

Operator behaviour

We discuss the expected behaviour of various types of validator operators:

1. honest operators

—By definition honest operators run validators that follow the consensus rules and report correct price data.

1. lazy operators

—The laziest behaviour for a lazy operator (e.g. a busy amateur staker) is to run the default validator client settings. As such lazy operators should run validators that report correct price data. (Computational laziness is not a concern since the computational overhead—in bandwidth, CPU time, RAM, disk—is marginal.)

1. rational operators

—Rational operators seek to maximise revenue and will consider price data manipulation to maximise Validator Extractable Value (VEV). However, since PriceData

objects are both public and cryptographically signed, dishonest validators are identifiable by the social layer. Potential punishments include: * social shaming

—Public staking-as-a-service operators (e.g. Coinbase, Binance) that run validators reporting incorrect price data would get socially flagged (e.g. on Twitter, Reddit) and risk losing reputation as well as business from disgruntled customers.

- social slashing

—If a price manipulation attack is attempted, and especially if the attack succeeds or almost succeeds, the community can coordinate a hard fork to slash the balance of dishonest validators. The balance of dishonest validators can be fully slashed (more so than enshrined slashing which by default partially slashes balances).

- social orphaning

—If dishonest price reporting is observed the community could socially orphan block proposals and attestations from dishonest validators to deprive them from rewards and inflict inactivity penalties.

1. social shaming

—Public staking-as-a-service operators (e.g. Coinbase, Binance) that run validators reporting incorrect price data would get socially flagged (e.g. on Twitter, Reddit) and risk losing reputation as well as business from disgruntled customers.

1. social slashing

—If a price manipulation attack is attempted, and especially if the attack succeeds or almost succeeds, the community can coordinate a hard fork to slash the balance of dishonest validators. The balance of dishonest validators can be fully slashed (more so than enshrined slashing which by default partially slashes balances).

1. social orphaning

—If dishonest price reporting is observed the community could socially orphan block proposals and attestations from dishonest validators to deprive them from rewards and inflict inactivity penalties.

Validator client guidelines

Validator clients must be able to query external sources to derive prices at epoch boundaries. Below are client implementation guidelines:

- diversity

—several diverse price sources must be queried for robustness, including onchain sources (e.g. Uniswap V2, MakerDAO, Chainlink—see oracles.club) and offchain sources (e.g. Coinbase, CoinMarketCap)

- security

—attack vectors arising from querying untrusted APIs must be carefully addressed

- sanitisation

—outlier sources must be handled adequately (e.g. with sanity checks and weighted medians)

- customisability

—client operators should have the option to easily change the default price derivation logic and customise their price sources

- decorrelation

—implementations should strive to maximise decorrelation in the price derivation logic (e.g. across implementations, releases, slots)

- standardisation

—over time price derivation best practices must be formalised and standardised

Guidelines for dApps

- historical accumulator

—Every PriceData

object can be authenticated against the historical_roots

in the [BeaconState

](https://github.com/ethereum/eth2.0-specs/blob/6474218fb162424a9c362de7d114e68a24059877/specs/phase0/beacon-chain.md#beaconstate). For gas efficiency the SHA256 Merkle witnesses can be efficiently compressed to a SNARK (with optimisations such as plookup).

- risk analysis

—The data feeds are provided “as is”. dApps must carefully evaluate price manipulation risk from dishonest validators and make informed security assumptions (e.g. validator honesty assumptions).

- hybrid oracles

—dApps should consider combining the enshrined feed service with other price sources available to them. The feed service is a piece of infrastructure designed to increase choice and robustness for the Ethereum ecosystem.