Overview

This project is aimed at proposing a peer-to-peer (p2p) network solution to support Data Availability Sampling (DAS) for the Ethereum blockchain.

DAS is critical for any blockchain design that provides data availability guarantees beyond the resources of any standard node on the network – e.g., Layer-1 comes to consensus on 1MB of data per second, but any standard node on the network only has the resources (bandwidth, storage, etc) to validate/download 50kb per second.
In a DAS model, nodes randomly sample data that has been Erasure-coded to ensure data is available and (in the event of data being missing or withheld) reconstruct portions of the data.

DAS can be divided into a number of problems:

1. Disseminate small amounts of data to all nodes (samples or sets of samples) to support sampling from other nodes, in a way that supports random sampling (see 2)
2. Support queries for random samples in an efficient and safe way
3. Identify and reconstruct missing data (to then disseminate, 1 & 2)

Objectives: Our goal is to design and analyze a scalable, secure and efficient p2p network solution that is able to fulfill Ethereum Data Availability Sampling requirements. The project will include the state-of-the-art analysis, the p2p network design and performance evaluation through network simulations and security analysis and countermeasures design, following the main objectives:

We aim to build a scalable mechanism to providing on-demand access to Erasure-coded block data samples in a p2p network that is:

- Low-latency
- Efficient
- resilient to attacks (Sybil, Denial-of-Service, spamming the network with requests etc.)
- evenly distributes the load across the p2p network consisting of resource-constrained nodes.

Our solution:

1. ensures that certain meta-data for all the samples (e.g., block headers) are reliably broadcasted to verifiers in order for them to authenticate each sample upon retrieval,
2. enables verifiers to efficiently retrieve the samples of their choice.

Assumptions: We assume the existence of *block builders* with plenty of upload bandwidth (on the order of gbit/s upload bandwidth). A block builder produces the coded data samples and makes them available for verifiers to perform random sampling.

We assume a p2p network with thousands of peers (i.e., verifiers), each querying a very small subset (~75 or so [6]) of a large number of (~262,000) coded samples in order to retrieve and verify their availability as part of *random sampling*. The random sampling must be completed in

less than a single block production period (i.e., within a time slot of duration approximately 16 seconds [6]). Unlike the block builder, the p2p network consists of resource-constrained peers with limited bandwidth resources (upto tens of KB/s for download and upload) to allocate for retrieving and serving samples.

Performance Metrics: The main networking challenges of DAS are the timely retrieval of a large number of samples without introducing bottlenecks in the network and securing random sampling in the presence of (possibly massive) Sybil attacks. We will evaluate several candidate solutions using the following performance metrics:

- Latency
    - Completion time of sample retrievals by the verifiers.
- Security metrics:
    - Percentage of eclipsed (i.e., inaccessible) samples due to Sybils,
    - Percentage of failed sample requests.
- Overhead:
    - Distribution of bandwidth usage across the p2p network peers (i.e., load-balancing),
    - Bandwidth usage by the block builder.

Challenges: One potential concern is that the DHTs are known to be occasionally slow in locating peers and retrieving data. We plan to tackle this problem by using larger routing tables at the DHT nodes and have them cache a large number of known peers. We will also investigate a hybrid of *iterative* and *recursive* DHT routing strategies to leverage caching of samples at intermediate nodes. Recursive routing can reduce the number of Round-Trip Times (RTT) incurred.

Another potential concern is the possibility of massive Sybil attacks. We plan to tackle this issue by leveraging the verifiers' *ability to authenticate samples* retrieved from the network, and use this to assess trustworthiness of peers who provide samples, as part of a trust-based peer selection mechanism. Specifically, a peer can assign trust scores to others based on their ability/inability to provide valid samples when queried.

In addition, we plan to incorporate diversity of peers in terms of their locations (i.e.,IP addresses) in the peer selection mechanism to make Sybil attacks more difficult: we assume that it is easier for an attacker to generate sybils with similar IP addresses (i.e. within a single subnet) than it is to generate ones with many diverse IP addresses. We have recently incorporated a similar diversity-based, Sybil-resistance mechanism to Discv5 topic-based service discovery [3, 9].

Another potentially useful approach (proposed by Cholez et al. in [2]) against targeted (i.e., localised) Sybil attacks (i.e., strategically placing Sybils in a target region of the keyspace to eclipse peers or samples) is to simply observe distribution of known peers' IDs in order to detect

occurrence of node density abnormalities in any region of the DHT. Although the work in [2] does not provide an exact recipe on how to avoid Sybils once an abnormality is detected, one possibility is to route requests around suspicious regions.

Potential Approaches

Approach 1: We plan to leverage Kademlia *Distributed Hash Table (DHT)* protocol for its efficient, fault-tolerant routing capability; *caching* to improve the performance and scalability of sample retrievals; and decentralised *peer selection mechanisms* for Sybil-resistance.

We plan to use the p2p DHT as a distributed cache. Once a block builder generates a new block, each verifier selects a set of samples to request. Each sample can be mapped to a path in the DHT by hashing the sample's unique ID (e.g., row and column identifiers in the block) and possibly other sample-specific meta-data, i.e., *Hash(sample ID, …, meta-data).* To request a sample, validators recursively traverse sample-specific paths. Only the last node in the path (i.e., **primary cache**) retrieves the corresponding samples directly from the block builder.

The paths are semi-deterministic so that: i) verifiers requesting the same sample will follow a similar path (and can re-use data already stored along the path and directly finish the query); ii) the paths chosen by different verifiers are not exactly the same making malicious placement of Sybil nodes difficult. This is in contrast to an approach where specific nodes in the DHT are responsible for storing samples exposing the system to Sybil attacks.

Random sampling combined with semi-deterministic paths should result with a balanced overhead across the DHT nodes given, reduced overhead for the builder (no need to send samples to everyone) and high security against malicious attacks.

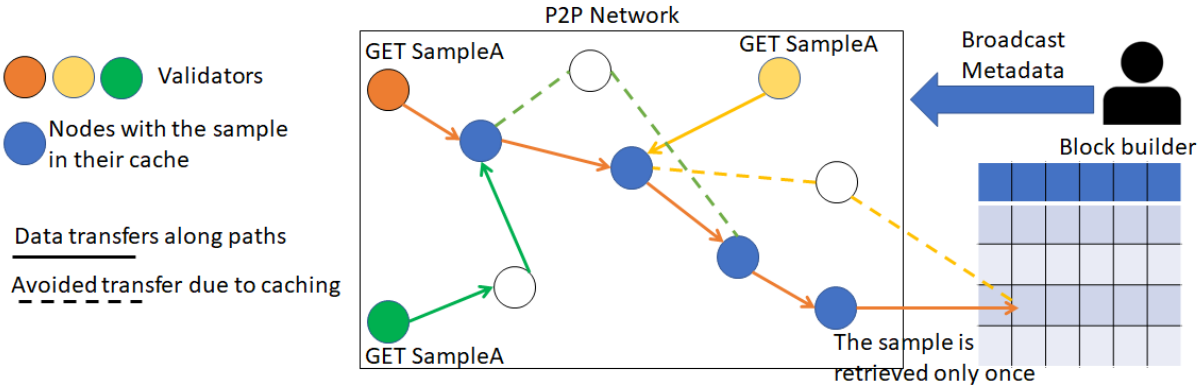

Fig. 1. Retrieving samples in a p2p network.

Approach 2: In the second approach, the DHT nodes first perform random peer sampling in order to determine which data samples to retrieve. Similar to the previous approach, the DHT nodes are designated as the primary cache for one or more data samples.

The objective of the peer sampling is to select peers uniformly at random in the presence of adversaries [5, 11]. Periodically, each node obtains a sample of peers, and uses the peers in the sample to retrieve their data samples for which the sampled peers are the designated primary cache. This approach has the potential advantage that the verifiers can immediately retrieve data samples from (already located) peers, as the peer sampling can take place in parallel and be completed before the data sampling.

Peers can perform *random walks* in the DHT to sample peers. However, adversaries can introduce bias in the distribution of peers obtained in the samples. We plan to investigate whether a random walk approach can sample peers uniformly at random (i.e., correct for the bias introduced by the adversaries) so that the peers can each obtain truly random data samples.

Project roadmap - 12 months

Stage 1: Network requirements definition and related work analysis (Month 0-2)

Tasks:
1.1) Pinning down the requirements: Identification and formalization of DAS requirements in terms of resource usage, networking and security.

1.2) Identify candidate approaches: Analyze existing approaches to DAS networking, identify their limitations, and assess their suitability to support Ethereum DAS networking based on the requirements identified in task 1.1.

Our analysis of existing approaches will include Celestia [7], Polkadot [8], GossipSub [4], libp2p and Discv5. Both Celestia [7] and Polkadot [8] blockchain do not provide documentation for their DAS networking solutions; therefore, we will investigate their source code to understand their approach.

Deliverable: Report including requirements and related work.

Stage 2: Network solution design and simulator implementation (Month 3 - Month 7)

We are already familiar with DHT networking simulation frameworks, and this will allow us to quickly prototype and evaluate various candidate approaches.

Tasks:
2.1) Build a DAS networking simulator: We plan to use PeerSim [10] simulator – a Java-based scalable p2p networking simulator.

2.2) Network design and initial specifications: We will design a network solution based on the approaches discussed in the Overview section.

2.3) Implement several candidate approaches: We plan to implement the approaches both examined in Task 1.2 and selected potential approaches discussed in the Overview section.

2.4) Performance evaluation: Analyse and compare the networking performance of the candidate approaches through simulations.

Deliverable: A github repository including network protocol specifications (for each candidate approach) and simulator implementation.

Stage 3: Security analysis (Month 8 - Month 12)

In this part of the project, we will analyze, implement in the simulator and evaluate any potential security threat, including the following:

- Denial-of-Service: ignore sample queries,
- Spamming: adversaries send bogus queries to overwhelm peers,
- Sybil attacks: strategically place a large number of Sybils to disrupt the random sampling process,
- Malicious DHT peer: launch DHT-level attacks to eclipse peers.

Tasks:

3.1) Identify possible threats and counter-measures: Explore potential attacks by adversaries for each candidate solution and identify potential counter-measures.

3.2) Implementation of adversaries and counter-measures: For each candidate approach, implement adversary behaviors and the corresponding counter-measures. Assess the performance of the counter-measures through simulations.

Deliverable: Threats model report and github repository including implemented countermeasures and final network protocol specifications.