Data Availability Sampling in Ethereum: Analysis of P2P Networking Requirements

Michał Król[‡], Onur Ascigil[†], Sergi Rene[*], Etienne Rivière[§], Matthieu Pigaglio[§]
Kaleem Peeroo[‡], Vladimir Stankovic[‡], Ramin Sadre[§], Felix Lange[¶]

[*] University College London, [†] Lancaster University
[‡] City, University of London, [§] ICTEAM, UCLouvain, Belgium
[¶] Ethereum Foundation

Abstract—**Despite their increasing popularity, blockchains still suffer from severe scalability limitations. Recently, Ethereum proposed a novel approach to block validation based on Data Availability Sampling (DAS), that has the potential to improve its transaction per second rate by more than two orders of magnitude. DAS should also significantly reduce per-transaction validation costs. At the same time, DAS introduces new communication patterns in the Ethereum peer-to-peer (P2P) network. These drastically increase the amount of exchanged data and impose stringent latency objectives. In this paper, we review the new requirements for P2P networking associated with DAS, discuss open challenges, and identify new research directions.**

Index Terms—**Ethereum, Scalability, Data Availability Sampling, Peer-to-peer, Networking.**

I. INTRODUCTION

Ethereum is the second-largest blockchain currently in operation. Thanks to its support of smart contracts, it enables a wide range of applications from financial services and sharing-economy systems to Internet-of-Things, supply chains, and digital health [15].

While the popularity of the Ethereum platform increases rapidly, its processing throughput remains very low, oscillating around 14 transactions per second (TPS). This low throughput is mainly a consequence of the consensus protocol [10] that validators (*i.e.*, platform maintainers) must run to agree on a common state. In each round of the consensus protocol, a chosen validator extends the blockchain by: *(i)* creating a block containing the most profitable transactions; *(ii)* linking the block to the previous one using a hash function, and *(iii)* disseminating the block to other validators using a peer-to-peer (P2P) gossip protocol [13]. Every block must be delivered and *verified* (*i.e.*, checked for correctness) by all the validators in a timely manner. Having more honest validators in the system increases security but also causes additional communication and lowers performance, leading to a dilemma between throughput and decentralization.

An emerging solution to increase the scalability and throughput of Ethereum is the use of rollups [11]. Rollups process their transactions off-chain and only periodically post small *commitments* of their current state on-chain, where consensus is reached. This hybrid approach enables smaller, rollup-specific communities to process transactions at high speed while being eventually secured by the validators of the blockchain.

In contrast with transactions and smart contract executions registered on chain, commitment data posted by the rollups *does not have to be verified* by Ethereum validators; it is sufficient to simply *include* this data in new blocks. Rollup participants can then verify commitment data using application-specific logic. If this data is incorrect, rollup participants simply withdraw their funds and quit the application [11].

In a rollup implementation over the current, unmodified Ethereum deployment, commitment data is included in an opaque data blob that is an integral part of a block and competes for on-chain space with regular Ethereum transactions. To limit the propagation time between the validator generating a new block and validators receiving and validating it, the size of a block is limited, and so is the size of this opaque data blob. Thus, while rollups can significantly improve Ethereum throughput, this improvement is capped at a few thousand TPS due to block size limitations.

The inclusion of commitment data from rollups in blocks disseminated for validation is, in fact, a hindrance to reaching order-of-magnitude higher throughputs (e.g., 100,000 TPS). Observing that commitment data must be available but is not checked for correctness by Ethereum validators, a new approach to block generation and validation emerged based on *Data Availability Sampling* (DAS). Unlike the previous approach, with DAS a validator only downloads in full the part of the block containing regular transactions, but not the opaque blob. Instead, each validator collects sufficiently strong evidence that the opaque blob was indeed produced and made available by its producer. The collection of this evidence is based on *random sampling*, a process that becomes an integral part of the consensus: if the sampling process succeeds, validators consider the block correct.

DAS has the potential to maintain the high decentralization and security of Ethereum while massively increasing throughput. However, its implementation presents significant challenges for the P2P network supporting the blockchain. Currently, Ethereum uses simple gossiping techniques which are unsuitable for DAS data flows. Indeed, the total amount of data generated per block increases from the current average of 90 KB up to 140 MB. Even a resourceful block producer might not be able to *directly* deliver the required samples to all network participants. At the same time, relying on *indirect* data dissemination is difficult. First, each validator requests different samples, making gossip inefficient. Second, relying on anonymous network participants opens a new range
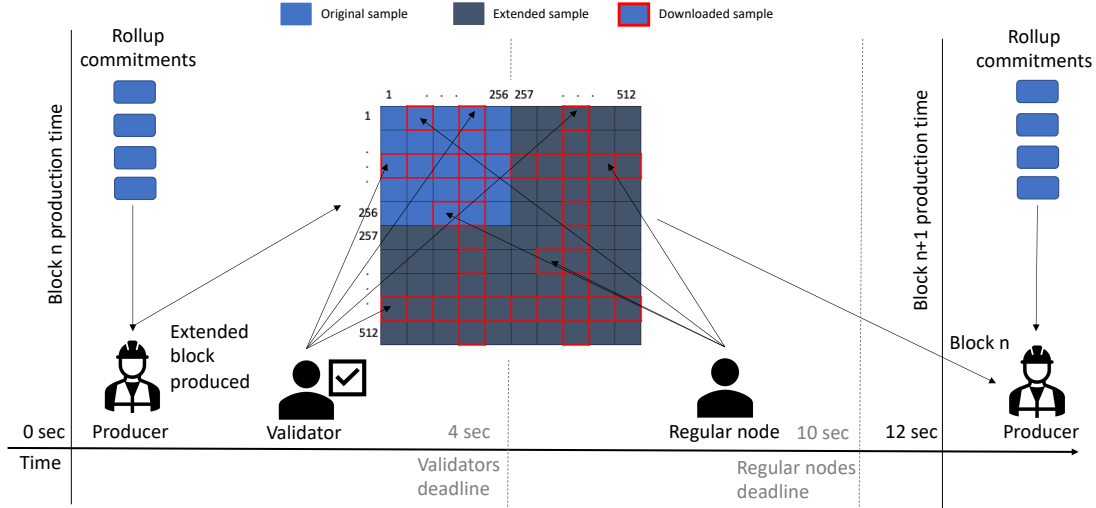
Fig. 1. Block data sampling. The producer gathers rollup commitments from the clients and creates an extended block. The block must be sampled by validators within 4 s from the block production by requesting 2 random rows and 2 random columns. Non-validator nodes sample the block within 10 s by requesting 75 random cells. After 12 s from the initial block production, the next block is created by a new block producer.

of network-level attacks, threatening the correctness of the consensus protocol. Finally, the sampling process must be completed quickly (within 4 s for validators) as it determines whether the block is considered correct and can be referenced by future blocks in the chain; larger latencies significantly hinder progress.

In this paper, we discuss design options for next-generation P2P networks that will be able to support Ethereum's rollup-centric DAS architecture at scale. We note that at the moment of writing, many details of the future DAS network are still under discussion. We base our analysis on emerging choices in the Ethereum development community, as they already give us a good sense of scale for this problem [8]. In Section II, we provide background on the new DAS approach. In Section III, we determine concrete P2P network properties required to support the new model. Section IV critically analyzes classical networking solutions and points out their shortcomings, Section V lists new open challenges and research directions for the ecosystem, and Section VI concludes the paper.

II. DATA AVAILABILITY SAMPLING

Ethereum joins users that hold accounts and submit transactions, and validators that include transactions in blocks and maintain a full copy of the blockchain. Ethereum recently migrated to a Proof-of-Stake (PoS) consensus protocol [1]. Any node can now become a validator by staking at least 32 ETH, the Ethereum cryptocurrency. Time is divided into slots of 12 seconds and epochs of 32 slots. One validator is selected pseudo-randomly and based on stake to be a block producer in every slot[1]. This validator is responsible for creating a new block and sending it out to other nodes. Also in every slot, a committee of validators is chosen randomly, whose votes are used to determine the validity of the block being proposed. The Ethereum design assumes that honest

[1]The Ethereum community recently proposed a *Proposer-Builder Separation* mechanism (PBS) that delegates block production to dedicated *builder* nodes. However, it does not change the requirements for the P2P network. Our discussion applies to both the current approach and this proposal.

validators control a majority of the stake in the system and the majority of the voting power in each committee.

The Data Availability Sampling (DAS) principle assumes that to every block is attached an opaque, binary blob (of up to 32 MB) holding rollup data. Such a blob can be represented as a 256×256 matrix of 512 B cells. Both validators and regular Ethereum nodes (*e.g.*, nodes that did not stake at least 32 ETH) do not verify the correctness of the blob (an operation that is specific to the different rollups) but only ensure that the blob was indeed released in full by the block producer. To avoid downloading the entire blob for this purpose, the network relies on DAS (*i.e.*, requesting a random subset of the data) [8]. Theoretically, a malicious block producer could release all but a small part of the blob hoping that this will not be noticed by DAS. Such behavior, called a *data withholding attack*, threatens the security of rollups. To prevent it, and as depicted in Figure 1, the block is extended using a two-dimensional Reed-Solomon erasure code [14]. Each row and column doubles in size but can be reconstructed from any 50% of its cells. The extended block takes, therefore, the form of a 512×512 matrix, where each cell contains 512 B of data. Furthermore, each cell includes a 48 B cryptographic Kate-Zaverucha-Goldberg commitment (KZGC) [5], enabling the downloading node to make sure that the cell is a valid part of the extended blob. In total, the extended blob uses a total of 140 MB of data including 12 MB of KZGCs. The original blob can now be reconstructed by acquiring at least 25% of the extended blob cells, making a withholding attack much easier to detect. Furthermore, KZGCs prevent a block producer from returning random data to sampling requests.

Depending on their role, nodes perform two types of data sampling:

- *Validators* randomly choose and attempt to download 2 rows and 2 columns of the extended blob (2044 cells in total). This *validator sampling* is considered successful when at least 50% (256 cells) of each column/row could be downloaded. The remaining parts of the selected row/columns can be reconstructed using the erasure code.

Each validator downloads at least $2046 \times 560\text{B} = 1.1$ MB of data per slot.

- *Regular nodes* randomly select and attempt to download 75 cells. This *regular sampling* is considered successful when all the selected cells can be downloaded. Each regular node downloads at least $75 \times 560\text{B} = 42$ KB of data per slot. Note that validators perform regular sampling in addition to validator sampling.

If the sampling process fails, nodes consider the sampled block (and all the blocks that would later build upon it) invalid. A malicious block producer may attempt to split the network by correctly responding to sampling requests from certain nodes while ignoring requests from others. Such a behavior, called a *network split attack*, results in honest nodes having a different view on the most recent valid block in the blockchain. As future blocks are built on top of the most recent block, the attack creates long-lasting forks, reducing both the efficiency and the security of the entire blockchain. The attack can be prevented by hiding the origin of sampling requests from block producers.

III. What does DAS mean for the network?

Currently, the Ethereum network is formed of 500,000 validators and 2,000 regular nodes. The target is, however, to support up to one million regular nodes participating in rollups. All nodes first join a Distributed Hash Table (DHT) to discover other blockchain participants with whom they form an unstructured, mesh peer-to-peer (P2P) network for block dissemination [6], [13]. In the unmodified version of Ethereum, once a new block is created, its producer sends it to its direct peers that propagate the block further using epidemic dissemination. Eventually, the block is delivered to all network participants.

However, such a solution cannot be adopted in the rollup-centric approach. Extended blobs are simply too big to be disseminated in full to the entire network during every 12 s slot. Furthermore, DAS assumes that each node downloads a small randomized subset of samples. Peers that are directly connected in the unstructured mesh are unlikely to request the same random samples, making the current epidemic dissemination intrinsically unsuitable.

A. Objectives

A P2P architecture supporting DAS should have the following functional and non-functional characteristics.

Functional requirements are threefold:

- **Support two retrieval modes:** For the long-term security of the ecosystem, the solution should support both validator (rows and columns) and regular (random cells) sampling.
- **Openness:** Ethereum is an permissionless blockchain. The P2P network should remain open for anyone to join.
- **Request unlinkability:** To protect against network split attacks, for any two sampling requests, the block producer should not be able to tell whether they originate from the same sampling node.

In terms of non-functional requirements, we identify:

- **Efficiency:** The network must be efficient in terms of the total amount of exchanged data by all its nodes. Higher overheads incur a higher monetary cost for the participants, threatening economic viability. They increase hardware requirements, hindering the decentralization of the network. An optimal solution transfers each requested sample exactly once. For the current network size, it means exchanging $500,000$ validators $\times 1.1$ MB $+ 1,200$ nodes $\times 42$ KB $= 489$ GB of data per slot.
- **Low dissemination latency:** The validators must have enough time to reach a consensus on the sampled block and start constructing the next block. Within a 12 s slot, the current consensus protocol requires that all requested samples be delivered within 4 s to the validators. Furthermore, to enable blockchain replication and reliable fork resolution the Ethereum community also suggests delivering samples to the regular nodes within 10 s [8].
- **Low cost for the producer:** The block producer is incentivized to keep creating blocks only if the process generates a profit. The sampling procedure, being part of the block creation, should incur low monetary costs for the block producer and ensure that they do not exceed rewards from block creation.
- **Sybil attack resistance:** DAS is an internal part of the consensus protocol and must be secure. As the network is open for anyone to join, the solution should be resistant to malicious actors using Sybil identities (*i.e.*, a large number of pseudonymous identities controlled by a single actor).

IV. Classic Networking Approaches

We now review classical approaches to data distribution and distributed architectures, in light of the DAS network requirements.

A. Centralized Approach

The first approach would be for all nodes to retrieve samples directly from the block producer. Joining the sampling process requires simply connecting to one of the producer's servers. As each block might have a different producer, the IP address/the DNS domain must first reliably be advertised to the network.

This approach would require significant resources from block producers to deliver the large amount of sampling data requested by validators and regular nodes. Using a well-provisioned cloud infrastructure, the cost for serving sampling requests for a single block would amount to roughly 25 USD[2], or a monthly cost of around 5.75M USD shared among all block producers. This should account for the fact, however, that the block producers are chosen on a per-slot basis. To be able to respond to such bandwidth demands in such a limited time, all potential block producers would have to set up permanent infrastructures that would need to be vastly over-provisioned, making cost figures significantly higher and severely impacting decentralization. An alternative would be to

---

[2]Taking as a reference the outbound data transfer cost for the Amazon AWS cloud https://aws.amazon.com/ec2/pricing/on-demand/, using a US-based datacenter

rely on cloud-based Content Delivery Networks, but this would unacceptably put the progress of the chain in the hands of a few providers. Another challenge would finally be to prevent Denial-of-Service (DoS) attacks targeting well-identified block producers.

Centralized sample delivery is conceptually simple and does not introduce intermediaries that may pose a security challenge. It makes, however, the provision of *request unlinkability* a challenge as requests are sent to the same location. While validator sampling queries could be issued from 4 IP addresses (*i.e.*, for 2 columns and 2 rows), regular sampling also requires that the 75 connections be indistinguishable from each other. Using an anonymization network (e.g., Tor) may answer this problem but lead to a number of additional challenges regarding efficiency, robustness, and reliability. While dedicated proxies may help, they would also become a critical part of the infrastructure and an easy target for attacks–simply magnifying the problem.

As a result, centralized approaches clearly go against the objectives of Ethereum in general, and DAS in particular, and should not further be considered.

B. Unstructured P2P Network

A peer-to-peer (P2P) network can assist in reducing the load on the block producer and increase decentralization. The block producer can push samples to a few network participants that will propagate them further. P2P networks establish an *overlay* over the regular network, where nodes connect to a few (relatively to the size of the complete network) direct neighbors, or peers.

Unstructured P2P networks do not impose a particular, rigid structure on the P2P overlay network. Nodes establish connections to each other either arbitrarily or following probabilistic choices leading to a globally random network [4], [13].

Data dissemination in unstructured P2P systems is typically achieved using gossip. Gossip is efficient and robust for broadcast (*i.e.*, sending the same message to everyone in the network) but requires additional measures for multicast (*i.e.*, sending a message only to a specific subset).

Multicast is typical of publish/subscribe systems where nodes register to receive content posted to specific multicast groups, or topics. Possible implementations of multicast include discovering the group during the dissemination, possibly leading to high overheads and the participation of many non-interested nodes in the dissemination. Another approach is to pre-establish sub-networks linking only nodes from the same group [2]. This approach is, however, only beneficial when groups are stable and re-used multiple times.

While blockchain systems and Ethereum benefitted from the simplicity and effectiveness of gossip-based broadcast, DAS introduces specific, novel needs for multicast. In a simple gossip scenario, if each node downloads and distributes only the samples it requires, the dissemination process never completes. On the other hand, making nodes participate in the distribution of additional samples (potentially up to a point where everyone downloads/distributes everything) is increasingly inefficient and requires additional resources when scaling up, reducing decentralization.

Pre-establishing dissemination structures is appealing but challenging due to the non-predictability of multicast groups. Indeed, while supporting the validator sampling mode requires establishing $512 + 512 = 1,024$ dissemination structures, one for each column or row, the regular node mode increases this number by an additional $512 \times 512 = 262,144$, one per sample. Additionally, each node randomly selects a different set of samples to download per slot, which means that the whole set of dissemination structures may need to be reconfigured every 12 s. With frequent reconfigurations, the signaling overhead can bypass the volume of data being exchanged and the network might never reach a stable state.

An alternative design could be for the producer to push samples to a *subset* of nodes only. Other participants would issue search queries to locate their content of interest. However, this solution also introduces significant signaling overhead in terms of search queries. Finding and downloading the relevant sample is unlikely to complete within the required 4 or 10 seconds.

On the positive side, unstructured P2P solutions significantly reduce operational costs for the producer. When each sample is delivered to a single node, the sampling process reduces the cost to 0,03$ per block or 6,000$ per month. However, delivering each sample to a single node threatens the security of the approach as those nodes may turn malicious and simply refuse to propagate the sample further. On the other hand, increasing the number of initial nodes also increases the monetary cost for the producer. This security risk can be reduced by a *peer reputation mechanism* [13]. It involves each node individually assessing the behavior of its direct neighbors and gradually replacing low-score peers. As a result, malicious nodes become isolated in the network and their impact is more limited. Unfortunately, such a technique offers limited protection against scenarios where attackers behave correctly, increase their scores, and suddenly act maliciously.

To join the network, newcomers have to either already know some participants or use dedicated bootstrap nodes. The latter, while used by almost all current P2P networks, is expensive for the bootstrap node operator and represents a single point of failure for the entire system. Unstructured P2P networks significantly reduce the possibility of data withholding attacks once the dissemination is underway. The producer does not have a direct connection with all the nodes and cannot decide to ignore requests on a per-sample basis. The producer can try to include its own Sybil nodes in the P2P network attempting to block certain nodes from accessing their samples. However, due to the lack of structure and pseudorandom nature of data propagation such actions are not efficient and require a significant amount of resources increasing with the network size.

C. Distributed Hash Table (DHT)

A DHT is a structured overlay network where nodes and data objects (stored in the network) are both identified with unique keys (*i.e.*, hash digests) drawn from a common hash space. DHTs allow efficient *lookup* operations to locate node(s) in charge of a given key, *i.e.*, nodes whose identifiers

are the "closest" to a key are responsible for storing the data object corresponding to that key. DHTs use a *distance metric* to measure the closeness of keys.

A DHT node s maintains a *routing table* to store information, *i.e.*, *node ID* and reachability (IP address and port number), on $O(\log(n))$ peers arranged by their distance to s, where n is the number of nodes in the DHT. Typically, a node's routing table provides a more detailed (*i.e.*, fine-grained) view of the subset of the network with closer node IDs and a less detailed view of nodes with distant IDs. This property leads to efficient lookup operations that take a logarithmic number of steps (*i.e.*, queried *hops*) in the number of nodes in the network. Building on these efficient lookup operations, DHTs support a *put* and *get* API to store and retrieve data objects. There exist several DHT implementations [7], [12] that mostly differ in terms of distance metric, lookup procedure, routing table structure, and so on. Kademlia [7] is nowadays the most commonly deployed DHT protocol, including by Ethereum.

DAS using a DHT: Because of its efficient data storage and retrieval operations, using a DHT as a *cache network*, where nodes collectively store and serve samples, is a natural choice for DAS. A DHT is a scalable network that can provide good load-balancing properties by assigning data objects uniformly to nodes. Ideally, a producer would only propagate a few copies of each cell of the block matrix to the DHT nodes responsible for storing them—*e.g.*, using the hash of the metadata of a cell (*e.g.*, row, column numbers) as its key. The efficiency of DHT lookup ensures that the producer can store (*i.e.*, put) samples in the DHT and that nodes retrieve (*i.e.*, get) samples efficiently.

An important threat to the operation of a DHT are Sybil identities. Because DHT nodes use self-generated identifiers (*i.e.*, the hash of a public key), malicious actors can spawn multiple identities (even within one physical machine) to operate many DHT nodes. Sybils can then attempt to disrupt lookups and even launch various attacks as we discuss below. The impact of Sybils on the lookup procedure is rather different for *iterative* and *recursive* lookups.

With iterative lookups (as used by Kademlia) the querying node contacts in turn each intermediate node on the route to the destination. Recursive lookups (as used by earlier DHTs [12]) let each initially contacted node continue the query on behalf of the querier. While iterative lookups increase the number of messages, they allow queries to better control the lookup and detect in particular if malicious nodes attempt to drop ongoing queries. At the same time, iterative lookups are more difficult to anonymize as the querier directly communicates with all the nodes contacted during the operation.

Another notable attack on DHTs is the *eclipse attack*, where Sybils trick other nodes to populate their routing tables with only Sybils, thereby isolating those nodes and partitioning the network. Initially, each node populates its routing table through trusted bootstrap nodes that are discovered out-of-band. Over time, nodes discover new peers (*e.g.*, ones who contact them) and store them in the routing tables subject to an *admission policy*: the Kademlia implementation used both by Ethereum and by prominent web3 protocols, such as the decentralized storage system IPFS, limits the number of nodes from the same /24 subnet in the same bucket or routing table, making eclipsing more difficult—*i.e.*, an adversary now has to deploy Sybils at multiple machines with diverse IP addresses to implement an attack.

S/Kademlia [9] proposes a Proof-of-Work (PoW)-based node ID generation to make Sybil generation more resource-intensive. The main idea is to require at least a configurable number of 0's at the end of the node IDs, which requires brute-force generation of potentially many public-private key pairs until an appropriate one is found. Another improvement by S/Kademlia is parallel iterative lookups that allow nodes to explore disjoint paths to obtain a higher probability of successful search results in the presence of Sybils.

DHT lookup overhead scales well with the number of participants but a single operation requires contacting a significant number of nodes (*e.g.*, $\sim$50 for a 20,000-node network). Combined with a large number of samples in DAS and frequent slots in Ethereum, this can put a significant load on the network. The high bandwidth consumption may be problematic, especially for constrained, non-validator devices that are essential for the decentralization of the platform. Fortunately, the random nature of the sampling process allows us to distribute the load across the network and avoid hotspots that handle a disproportionate number of queries.

DHT lookups can potentially take a long time to complete with each contacted node by the querier being possibly far away in terms of geographical distance. Slow lookups can be problematic for DAS, which has stringent timing requirements for the completion of sampling operations. Recursive lookups can improve the latency (*i.e.*, they require fewer round-trips), but they are also more difficult to secure, as discussed above.

V. RESEARCH DIRECTIONS

A straightforward application of existing P2P (*i.e.*, gossip and DHT) approaches fall short of satisfying the requirements of DAS in several aspects including security and delivery efficiency, while centralized approaches are unsuitable. In this section, we discuss possible directions to improve P2P approaches and make them suitable for DAS.

In terms of building a secure, Sybil-resistant P2P layer, one promising approach is to leverage the honest majority assumption of the consensus layer. In particular, the majority of the stake in the system is controlled by the honest parties as a security assumption of PoS. Therefore, an emerging approach is to use the honest majority assumption to build more robust P2P networks. Coretti *et al.* [3] propose a peer sampling approach to choose neighbors in which nodes prefer staking nodes to build Sybil-resistant gossip networks. The resulting network has a highly connected backbone that Sybil nodes cannot partition as long as the honest majority assumption is not violated.

Currently, regular sampling fails when a single cell cannot be downloaded. It imposes high robustness requirements on the P2P networks that, due to its open and decentralized nature, might be challenging to fulfill. Designing k *out of* n sampling schemes might reduce those requirements and make P2P network deployments more practical. In such a scheme,

regular nodes request $n > 75$ samples but still consider them successful when $k < n$ samples are downloaded within the time limit. Careful tuning k and n parameters can yield similar security guarantees to the current scheme while allowing the P2P network to tolerate sporadic node failures and malicious behavior. The scheme can be easily extended to improve the robustness of validator sampling.

The ultimate weakness points of DHTs is the inability to protect specific places in the hash space. An attacker can ultimately generate thousands of peer IDs to strategically place themselves close to a specific key and hijack all the key-specific traffic. Storing/retrieving data from specific regions rather than from nodes closer to a specific key has the potential to remove this weakness. During an attack, while the data will be stored on/retrieved from malicious nodes, honest nodes are also guaranteed to fall within the region due to the uniform distribution of pseudorandom, legitimate peer IDs. As samples contain KZGC integrity proofs, sampling nodes can also easily filter out incorrect data. However, such an approach requires developing new, region-specific routing procedures in the DHT and redesigning the structure of its routing table.

When traversing the DHT, nodes ask their peers for information on how to access a specific part of the network. An attacker can attempt to hijack this process by returning only malicious nodes as a way for the node to progress. Currently, there are no constraints on the information returned by each participant that can be verified by the querying node. Carefully setting additional rules on the peers each node can return might significantly increase the security of DHT routing while maintaining its high efficiency.

One of the main problems of multicast DAS data dissemination is the necessity of frequent reconfiguration of a large number of pre-established groups that cause high signaling overhead. Making group membership more stable (*e.g.*, across multiple slots) has the potential to make the multicast approach viable and much more efficient than broadcast-based dissemination. Alternatively, only a small number of nodes could migrate between fixed dissemination groups in every slot. However, such approaches require careful design to preserve the unpredictable nature of the samples requested by each node.

The majority of P2P networks currently in use were developed about 20 years ago using hardware specifications that are no longer relevant. Current hardware is much more powerful in terms of memory and computational power and can support much higher bandwidths. For instance, storing routing information about 20,000 peers in a in Kademlia requires only $\sim$300 MB of memory. Adjusting P2P network parameters (*e.g.*, DHT bucket sizes) can also significantly improve the performance of P2P networks to meet DAS requirements.

VI. Conclusion

We revised the networking requirements for the next generation, rollup-centric Ethereum blockchain. While the current, classical approaches fall short of providing scalable network support for the new architecture, solving open research challenges has the potential to realize this vision. Supporting 100,000 transactions per second may push blockchains into mainstream use and fully realize their potential.

References

[1] The merge: Ethereum switch to proof-of-stake. https://ethereum.org/en/upgrades/merge/. Accessed on February 14, 2023.

[2] Roberto Baldoni, Roberto Beraldi, Vivien Quema, Leonardo Querzoni, and Sara Tucci-Piergiovanni. TERA: topic-based event routing for peer-to-peer architectures. In *Inaugural international conference on Distributed event-based systems*, DEBS, 2007.

[3] Sandro Coretti, Aggelos Kiayias, Alexander Russell, and Cristopher Moore. The generals' scuttlebutt: Byzantine-resilient gossip protocols. In *29th ACM SIGSAC Conference on Computer and Communications Security*, CCS, page 595–608, 2022.

[4] Márk Jelasity, Spyros Voulgaris, Rachid Guerraoui, Anne-Marie Kermarrec, and Maarten Van Steen. Gossip-based peer sampling. *ACM Transactions on Computer Systems (TOCS)*, 25(3):8–es, 2007.

[5] Aniket Kate, Gregory M Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In *International conference on the theory and application of cryptology and information security*, ASIACRYPT, pages 177–194. Springer, 2010.

[6] Lucianna Kiffer, Asad Salman, Dave Levin, Alan Mislove, and Cristina Nita-Rotaru. Under the hood of the ethereum gossip protocol. In Nikita Borisov and Claudia Diaz, editors, *Financial Cryptography and Data Security*, pages 437–456, Berlin, Heidelberg, 2021. Springer Berlin Heidelberg.

[7] Petar Maymounkov and David Mazieres. Kademlia: A peer-to-peer information system based on the XOR metric. In *International Workshop on Peer-to-Peer Systems*, IPTPS, pages 53–65. Springer, 2002.

[8] Joachim Neu. Data availability sampling: From basics to open problems. Technical report, Paradigm, 2022. Accessed on February 17, 2023.

[9] Riccardo Pecori. S-Kademlia: A trust and reputation method to mitigate a Sybil attack in Kademlia. *Computer Networks*, 94:205–218, 2016.

[10] Sara Rouhani and Ralph Deters. Performance analysis of ethereum transactions in private blockchain. In *8th International Conference on Software Engineering and Service Science*, ICSESS, pages 70–74. IEEE, 2017.

[11] Cosimo Sguanci, Roberto Spatafora, and Andrea Mario Vergani. Layer 2 blockchain scaling: A survey. *arXiv preprint arXiv:2107.10881*, 2021.

[12] Ion Stoica, Robert Morris, David Liben-Nowell, David R Karger, M Frans Kaashoek, Frank Dabek, and Hari Balakrishnan. Chord: a scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Transactions on networking*, 11(1):17–32, 2003.

[13] Dimitris Vyzovitis, Yusef Napora, Dirk McCormick, David Dias, and Yiannis Psaras. Gossipsub: Attack-resilient message propagation in the filecoin and eth2. 0 networks. *arXiv preprint arXiv:2007.02754*, 2020.

[14] Stephen B Wicker and Vijay K Bhargava. *Reed-Solomon codes and their applications*. John Wiley & Sons, 1999.

[15] Kaspars Zīle and Renāte Strazdiņa. Blockchain use cases and their feasibility. *Applied Computer Systems*, 23(1):12–20, 2018.