# A Place for Everyone vs Everyone in its Place: Measuring and Attacking the Ethereum Global Network

Chenyu Li[1,2], Ren Zhang[3][0000−0003−2063−1769], and Xiaorui Gong[1,2]

[1] Institute of Information Engineering, Chinese Academy of Sciences
[2] School of Cyber Security, University of Chinese Academy of Sciences
[3] Cryptape and Nervos
ren@cryptape.com

**Abstract.** The Ethereum Global Network (EGN) is the peer-to-peer (P2P) network underlying Ethereum and thousands of subsequent blockchain services. Deviating from traditional single-service P2P networks, EGN's multi-service architecture has gained widespread acceptance for supposedly improving node discovery efficiency and security. This paper challenges this belief by critically examining EGN's design and its purported benefits. Our analysis reveals significant shortcomings in EGN's node discovery process. EGN nodes struggle to connect with peers offering the desired service: over three-quarters of connection attempts reach nodes of other services. In an extreme case, one node spent an average of 45 908 connection attempts to find each neighbor. Moreover, this blended architecture compromises EGN's security. The network demonstrates high susceptibility to DHT pollution and partition attacks. Even with only 300 malicious nodes in EGN, an attacker can isolate thousands of nodes, significantly hindering recovery. In contrast, such a small number of malicious nodes has minimal impact on every single-service P2P network. We propose solutions to improve EGN's node discovery efficiency and strengthen its resilience against attacks.

## 1 Introduction

Ethereum [4], a prominent blockchain system, operates on a peer-to-peer (P2P) network composed of individual servers called *nodes*. These nodes find and connect to each other through *node discovery protocols*. As of January 2025, the time of this writing, two versions of these protocols coexist—Discv4 and Discv5. Both protocols are based on Kademlia [23], a decade-old distributed hash table (DHT) protocol. In a DHT protocol, each node has a unique identifier (ID) and maintains a *routing table* that stores other nodes' IDs and routing information. The node discovery protocols periodically update the local routing table and provide candidate nodes for establishing application-layer connections. All nodes executing these protocols, i.e., the union of their routing tables, thus form an overlay P2P network called the *Ethereum Global Network (EGN)* [16].

In contrast to traditional overlay P2P networks, which typically cater to a single service, EGN embraces a vibrant ecosystem of applications reaching far beyond Ethereum, with a total market capitalization exceeding 500 billion USD [2]. Thousands of blockchain projects, from Ethereum testnets (Ropsten and Sepolia) and legacy Ethereum forks (Ethereum Classic) to other blockchains like BNB Smart Chain, Gnosis, and Polygon, find a home within EGN. This diverse spectrum even encompasses infrastructure providers like Bloxroute, Debank, and Quicknode, enriching the network with high-speed routing, user asset tracking, and blockchain data API services. Remarkably, despite coexisting on the same overlay, these services operate autonomously, collaborating only at the network layer without application-specific interactions.

This "living under the same roof" phenomenon is more of a design choice rather than the developers' negligence, as it is well-known and widely accepted. It is noticed and documented by academia as early as 2018 [15]; technical introductions of the Ethereum network, e.g., [6] and [3], also remind its audience of this unique feature/phenomenon. It is even introduced in Ethereum's official documentation that "there are many independent 'networks' that conform to the protocol without interacting with each other" [1].

The Ethereum community's acceptance, even enthusiasm, for this choice prompted us to explore its potential benefits. Through a combination of literature review and extensive discussions with seasoned developers, we identified two primary reasons supporting this choice:

- **Efficiency.** Each node's information spreads more widely, thus nodes may find same-service neighbors with fewer queries and shorter latency [24].
- **Security.** EGN's sheer size bolsters its resilience against Sybil attacks [16], where malicious actors swarm the network with their nodes, and their subsequent attacks. In other words, a larger pool dilutes the attacker's influence.

This paper argues that, without any implemented mechanism to prioritize same-service peers in node discovery, the blended nature of EGN is detrimental to services, especially smaller ones. Our technical work and contributions include:

**Measuring EGN.** While extensive research has explored the Ethereum network [5, 9, 10, 17, 18, 21, 28, 30], our understanding of EGN's multi-service architecture remains incomplete. The only study examining EGN's service diversity dates back to 2018 [15], predating nine out of the ten most prevalent services currently in use. Existing analyses of the network [8, 14, 19–22] overlook the composition of individual DHTs. Furthermore, the Discv5 network, deployed in late 2023, has yet to be analyzed.

Therefore, to facilitate subsequent analysis, we collect various measurements with a state-of-the-art crawler Nebula [26]. Our assessment encompasses EGN's scale, service distribution, node uptime, and critically, the composition of nodes' DHTs. We identify over 7000 services and reveal a concerning observation: the majority of Discv4 nodes maintain less than 5% of same-service peers within their DHTs. While the specific services of most Discv5 nodes remain unidentified,

we anticipate similar behavior due to the shared DHT maintenance rules and workflows between Discv4 and Discv5.

**Evaluating the Node Discovery Efficiency.** Node discovery efficiency directly impacts the overall communication cost for EGN nodes. Our measurement reveals that roughly 20% of our Ethereum node's traffic is dedicated to node discovery, with this percentage even higher for less busy services.

To assess the impact of the blended architecture on node discovery, we deploy five nodes in EGN representing diverse services: two exclusively support Discv4 (BSC Mainnet and KCC Mainnet), while the remaining three support both versions (ETH Mainnet, ETH Holesky and ETH Sepolia). These nodes measure communication costs and latency associated with establishing connections. The results reveal substantial inefficiencies. Within a twelve-hour timeout, our nodes rarely established the desired number of connections, and each successful connection cost at least thousands of attempts. The lowest connection success rate was one in 45 908, starkly contrasting with Bitcoin's rate of one in four. Error messages from 45.6% to 87.3% of connection attempts explicitly indicated attempts to reach nodes of other services. Analysis of incoming connection requests revealed that 76.2% to 96.9% of these requests were from other services.

Discv5 exhibits significantly poorer performance compared to Discv4. Within twelve hours, a node established at most three connections with nodes discovered via Discv5. A notable factor contributing to Discv5's inefficiency is the absence of the *topic discovery mechanism*. This mechanism, designed to enable nodes to advertise their services and prioritize same-service connection attempts, remains unimplemented in all client programs we examined.

**Comparing the DHT Pollution Attack Resistance in Blended and Separate Overlay Networks.** To assess EGN's security, we simulate a *DHT pollution attack*, where malicious nodes inject their information into other nodes' DHTs. Regarded as a weaker version of Sybil attacks, this attack also serves as a foundation for more sophisticated attacks. Even with only a few hundred malicious nodes (0.3% of the EGN), the attack reduced connection success rates to below 1% for all services except ETH Mainnet and Polygon. Additionally, after a 24-hour attack duration, the attacker successfully isolated thousands of nodes. Our analysis reveals that the vulnerability does not stem from implementation or protocol flaws but rather from the blended nature of the overlay network. The same attack proved largely ineffective in simulations of separate overlay networks.

**Fixing EGN.** We conclude by proposing solutions to increase the percentage of same-service nodes in nodes' DHTs while maintaining future interoperability and decentralization. Additionally, we suggest strategies to bolster EGN's resilience against DHT pollution and partitioning attacks.

This study challenges the commonly held assumption that a larger P2P network inherently translates to better data availability and security. A larger network does not guarantee improved data availability unless the data itself—in our case, nodes' routing information—becomes more accessible. Similarly, a larger network dilutes attacker resources only if it maintains robust internal connec-

tivity. When network size outpaces the ability of nodes to locate reliable and honest neighbors, it can become a vulnerability.

## 2 Ethereum Global Network: a Hodgepodge

We begin by outlining the technical details of the node discovery protocols. We then explore the formation and visionary goals of EGN itself.

### 2.1 Discv4: Messages and Key Data Structures

**ENR.** *Ethereum Node Records (ENRs)* are the basic data units propagated via Discv4. An ENR encapsulates the routing information of the node that generates it. Key ENR entries include: (1) secp256k1, the compressed representation of the node's secp256k1 public key; (2) signature, a cryptographic signature of the record content, verifiable with secp256k1; (3) seq, a sequence number that increases when the node updates and republishes its record; (4) ip, the node's IPv4 address; (5) tcp and (6) udp, the ports for incoming TCP and UDP connections, respectively. EGN identifies nodes using the hash of their public keys, known as the *node ID*. An ENR does not reveal the node's service. An ENR is sometimes stored or processed as an *enode*, which is an alternative format representing the same information. We use *node* to denote the server, and *peer* to denote the corresponding DHT entry.

**Message Types.** Discv4 relies on two key pairs of request-reply messages. (1) Ping/Pong: this pair serves basic *liveness check*—reachability checks and latency measurement. The initiator sends a Ping to check the recipient's liveness, who responds with a Pong. (2) FindNode(targetID)/Neighbors: this pair is used to discover nodes to construct the routing table. The initiator sends a FindNode(targetID) to request information about nodes whose IDs are closest to targetID; the recipient responds with a Neighbors message containing up to 16 closest nodes from its DHT buckets (see the next paragraph). In most requests, targetID is chosen randomly, as ID locality in EGN does not imply service or geographic proximity. Unsolicited replies (Pong and Neighbors) are discarded to prevent nodes from manipulating their liveness or injecting batches of ENRs into other nodes' routing tables.

**DHT Buckets.** The core of the DHT routing table is 17 *buckets*, each can accommodate up to 16 peers' enodes. Peers in bucket $i$ ($1 \leq i \leq 16$) share the same first $16 - i$ most significant bits with the local node in their IDs, with bucket 0 containing peers whose common prefix exceeds 15 bits. Within a bucket, enodes are ordered from the latest liveness-checked one to the earliest. When the context is clear, we use *DHT* instead of "DHT routing table" or "DHT buckets".

**ENR Insertion and Deletion.** When a new ENR is received, it is converted into an enode and inserted into its corresponding bucket as the latest one if space is available. Otherwise, it becomes a *replacement peer*, with a maximum of ten replacements stored per bucket. The oldest replacement peer is discarded

to make room for a new one when the limit is reached. Peers are removed from buckets only after they fail a liveness check. In such cases, the latest replacement peer (if any) is promoted to fill the vacancy.

**The discmix Queue.** This queue determines the order in which the node attempts to initiate application-layer connections. The pushing and popping strategy of discmix is discussed as "Outgoing Connection Maintenance" in Sec. 2.2.

### 2.2 Discv4: Key Workflows

**Bootstrap.** Upon first launch, a node adds pre-configured *bootnodes* into its buckets, sends Ping to confirm their liveness, and sends Findnode messages to acquire an initial set of peers from them.

**Active DHT Maintenance.** Throughout the node's lifetime, it continuously performs the following two tasks. (1) *Liveness check*: a random bucket is selected, and a Ping message is sent to its oldest peer, i.e., the one that resides at the end. If the peer replies timely, it is considered alive and moved to the bucket's front as the latest. Otherwise, it is deleted, and the newest replacement peer is promoted. This process is repeated after a random period, usually within ten seconds. (2) *Refresh*: a random node is picked, which has not been queried via Findnode by the current process, to receive a Ping message, and if it responds timely, a Findnode is sent to it. The replied ENRs are inserted as previously described. This process repeats immediately after receiving the reply.

**Passive DHT Maintenance.** Receiving a Ping message triggers the node to include the sender in its DHT. This, as we will explore, allows a malicious node to infiltrate a victim's buckets, either directly or as a replacement peer. In the former case, the malicious node remains within a bucket as long as it survives liveness checks; in the latter case, the malicious node has an opportunity to enter a bucket when a bucket peer is evicted. This chance increases with frequent Ping messages, ensuring it is often the latest replacement peer.

**Outgoing Connection Maintenance.** This process activates when outgoing connection slots remain available. Upon reaching the defined limit (16 by default), the process pauses. When discmix is empty, all enodes from the 17 buckets are sorted by distance to a random target and added to the queue. All these peers who have not been queried by a Findnode message by the current thread are then sent a Ping message and, if they respond timely, a Findnode message. The received ENRs update the local DHT.

A separate thread continuously processes the queue, attempting to establish outgoing connections—described next—with each popped enode until all connection slots are filled.

**Application-Layer Handshake.** This workflow is defined in Discv4's broader protocol stack Devp2p. The handshake commences with two nodes negotiating a session key derived from each other's secp256k1 public keys. Upon successful key negotiation, they share their supported upper-layer protocols' protocolStrings. For instance, "ETH68" signifies the version of the Wire protocol, used for block

and transaction synchronization. As different services often use the same protocol set for application-layer data exchange, service identification via the protocolString set is not possible. If the nodes find at least one common protocolString, they proceed to exchange a four-tuple of the protocolVersion, networkID, genesisHash, and forkID to check whether they operate on the same service and the same chain. The protocolVersion is the version corresponding to protocolString. The networkID distinguishes between different services and networks. For example, the networkIDs of the ETH Mainnet and the BNB Smart Chain (BSC) Mainnet are 1 and 56, respectively. The genesisHash is the hash of the very first block, i.e., *genesis block*, in the chain. When a service family possesses multiple chains with the same network ID, their genesis blocks must differ. Finally, the forkID pinpoints the *hard fork* the node considers valid. A hard fork occurs when some participants adopt some new consensus rules opposed by others. This splits the blockchain nodes into two groups with distinct forkIDs.

A successful handshake enables communication via upper-layer protocols, which maintain application-layer connections over TCP rather than the UDP-based node discovery protocol. Conversely, any mismatch in the four-tuple results in the handshake's termination with an error. Importantly, this error does not lead to the removal of the corresponding bucket entry.

### 2.3 Discv5: Key Differences

Discv5 is intended to provide increased confidentiality and improved node discovery efficiency. It offers a number of enhancements compared to its predecessor. First, it encrypts communications. Second, a new liveness check data format enables nodes to detect whether a peer resides behind a Network Address Translator (NAT). Third, Findnode now supports requests for peers at a specific distance from the receiver, facilitating targeted queries of specific buckets. Other messages, data structures, and the first four workflows remain largely unchanged from Discv4.

Determining a node's service in Discv5 presents a greater challenge. The aforementioned handshake workflow, which exchanges service information via the four-tuple, is embedded within the Devp2p protocol stack. However, Discv5 integrates with another protocol stack Libp2p. Consequently, each Discv5 service defines its own handshake protocol, typically requiring a successful handshake with a client of the same service to ascertain a node's service. This case-by-case approach lacks scalability given the multitude of services.

Although Discv5's design incorporates two mechanisms to facilitate efficient service identification, our investigation in Sec. 3.2 and experiments in Sec. 4.1 reveals that no EGN clients currently implement these mechanisms: First, the enhanced flexibility of the ENR format allows for the optional inclusion of service information. Second, a topic discovery mechanism enables nodes to broadcast an arbitrary topic string, conveying service information or other relevant data, across the network, thereby accelerating the discovery of same-service peers.

Consequently, the only straightforward approach to determine a node's service relies on an alternative mechanism within the Libp2p protocol stack. If

set by the client developer, two strings—agentVersion and protocols—are replied to each incoming connection attempt. These strings convey client information, enabling service identification. However, no current clients leverage this information to accelerate same-service node discovery.

### 2.4 EGN: Emergence

Standing apart in the realm of P2P networks, EGN accommodates a remarkable range of hundreds of diverse services. This unique phenomenon naturally piques our curiosity about the protocol designers' motivations and the process of service integration. While reconstructing the exact thought process behind EGN's emergence is impossible, we can examine some key design choices that likely facilitated its development.

**Discv4's Design Foreshadowed Expansion.** The Discv4 protocol's use of three identifiers (networkID, genesisHash, and forkID) for service-chain identification appears unnecessary for Ethereum's mainnet and testnets. This inclusion of networkID suggests the designers' foresight for a multi-service network from the outset.

**Leveraging Geth's Foundation.** Forking Geth, the official Ethereum client, is a common practice for new blockchain projects due to its comprehensive functionality and security advantages. Specifically, Geth offers a complete suite of modules, including consensus protocol, smart contract support, and P2P network organization. Forking allows developers to focus on their core innovation while inheriting established functionalities. Moreover, as a battle-tested client, Geth provides a solid security baseline; future security improvements can also be adopted by merging Geth updates.

**Early Projects' Choice and the Network Effect.** Early and influential projects like BSC and Gnosis chose to join EGN with separate networkIDs and genesisHashes, rather than creating their own P2P networks. This established EGN as the default choice for new Geth-based services, strengthening its dominance in the landscape.

### 2.5 EGN: Potential Benefits

EGN's open and inclusive nature has been widely embraced by the blockchain community. This is evident in the numerous academic studies [8, 9, 14, 15, 20–22] and technical articles [1, 3, 6] introducing and measuring the flourishing ecosystem. While most research accepts EGN's hospitality as a given, we examine its potential implications. We tentatively explore these potential benefits through a two-step approach. First, we scrutinize the existing literature and discuss with experienced developers to compile a comprehensive list of seemingly advantageous aspects. Second, we critically analyze these benefits and identify two key areas for further technical investigation: efficiency and security.

**Listing the Potential Benefits.** Complementing the existing literature, we discuss with experienced developers to compile a comprehensive list of seemingly advantageous aspects of the phenomenon.

1. *Maximized Code Reuse.* Most EGN services leverage Geth's P2P module as their foundation, reducing development complexity. Joining EGN further amplifies this benefit by allowing nearly identical network components with those of Geth, requiring only adjustments to service-chain identifiers and, optionally, bootnodes.
2. *Reduced Deployment Cost.* Smaller services can piggyback on the bootnodes of larger ones, saving resources on maintaining their own P2P infrastructure.
3. *Higher Node Discovery Efficiency.* Introduced as "Efficiency" in Sec. 1, a larger network may accelerate node discovery. Specifically, [24] indicates that gossiping ENRs across services, rather than partitioning the network by services, could decrease the latency of establishing connections.
4. *Better Attack Resilience.* Introduced as "Security" in Sec. 1, a larger network dilutes the attacker's damage. Specifically, Michał et al. mentioned in [16] that the blended architecture "offers good resilience to malicious behaviors".
5. *Potential for Interoperability.* Sharing a common overlay network lays the groundwork for future interoperability between services, potentially fostering collaboration and innovation within the ecosystem.

**Initial Analysis.** We critically analyze these benefits and identify two key areas worthy of further technical investigation. While code reuse (item 1 above) offers convenience for developers, it is primarily an internal benefit with no direct external impact. Piggybacking bootnodes (2) offers a clear economic advantage for smaller services, but comes with high security costs. The smaller services are susceptible to censorship attacks, and these bootnodes become more attractive targets for attackers. While the shared overlay network opens doors for future interoperability (5), this remains speculative without concrete plans or implementation efforts, rendering it infeasible to evaluate its potential impact. Therefore, we prioritize further technical investigation into EGN's efficiency (3) and security (4).

## 3 Measuring EGN

While prior measurement studies on EGN exist [8,9,14,15,20–22], we find it necessary to conduct a new one for two reasons. First, existing results are outdated. As of 2025, the most recent measurement is over three years old, and the only study involving EGN's service distribution (by Kim et al. in 2018 [15]) predates the launch of most current services. Second, these studies lack an analysis of the composition of nodes' DHTs, which is indispensable for our investigation. Our measured data are used to simulate our attacks in Sec. 5.

### 3.1 Crawler Setup and Deployment

This study uses Nebula [26], a state-of-the-art crawler previously employed in research such as [27]. The crawler was deployed on a US-based commercial VPS equipped with a 32-core CPU, 128 GB of RAM, and 4 TB of NVMe SSD storage.

Each crawling iteration was limited to a duration of thirty minutes. Extending this duration would not yield a more accurate snapshot, as the rate of newly discovered nodes diminishes to approximately one per second, which is indicative of typical network churn. The crawling process was conducted between December 12th and 27th, 2024, resulting in 720 snapshots.

In each iteration, the crawler simultaneously collected information from both Discv4 and Discv5 networks. Each discovered Discv4 node was queried with 15 Findnode messages, each targeting a randomly selected targetID as bucket-specific queries are not supported. Meanwhile, each discovered Discv5 node was queried with 17 Findnode messages, each targeting a specific bucket. The crawler recorded the DHTs of all discovered nodes for subsequent analysis. Newly discovered peers underwent a liveness check, and upon successful verification, were added to the processing queue. For each node, the crawler attempted to initiate a handshake and recorded the extracted four-tuple, agentVersion and protocols strings, or any error messages.

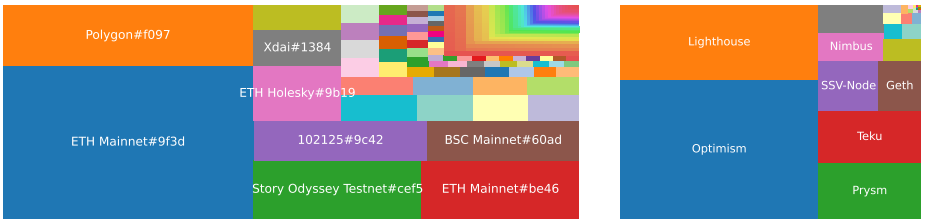
### 3.2 Network Scale, Service and Node Uptime Distribution

**Network Scale.** The Discv4 network size fluctuated between 94 180 and 113 524, averaging 103 432 nodes. This represents a threefold increase from 33 000 measured between February and March, 2022 [8]. The Discv5 network size fluctuated between 208 268 and 238 219, averaging 223 132 nodes. Approximately 8,000 nodes in each snapshot operated both Discv4 and Discv5 protocols using the same public key. We further analyze the distribution of unique identifiers. We observe 2 047 673 unique public keys and 197 449 unique IP addresses.

**Node Uptime Distribution.** We observe that 32 804 Discv4 nodes and 162 286 Discv5 nodes remained active throughout the measurement period, while 1 274 461 Discv4 nodes and 99 687 031 Discv5 nodes were only active for a single snapshot. Not surprisingly, the uptime patterns exhibit consistency across different services in Discv4, suggesting a general trend rather than service-specific behavior.

**Observation 1.** *Over two-thirds of active Discv5 nodes in any given snapshot remain online consistently throughout our measurement period.*

**Investigating the Source Code of Discv5 Clients.** Despite the inclusion of service information in Discv5's updated ENR format and topic discovery mechanism, we do not observe any ENRs containing this information. This leads us to investigate the implementation status of these claimed updates. We examine the source code of: (1) all five *Ethereum execution clients* (Geth, Nethermind, Besu, Erigon, and Reth), which propagate transactions and blocks; (2) all six *Ethereum consensus clients* (Prysm, Lighthouse, Teku, Nimbus, Lodestar, and Grandine), which participate in the consensus protocol; and (3) three other popular Discv5 clients providing the agentVersion or protocols string (Optimism, SSV-Base, and Base). Our analysis reveals that none of these clients incorporate service information in their ENRs or implement the topic discovery mechanism. To the best of our knowledge, most other clients are forked from those we inspected. Therefore,




(a) Service distribution of 39 795 Discv4 nodes. The fifth largest service "102125" lacks public information.

(b) Client distribution of 18 042 Discv5 nodes.

Fig. 1: EGN on December 17, 2024, at 8:12 UTC. In each figure, the service's area is proportional to its node count. Around 65% of Discv4 nodes and 91% of Discv5 nodes have no service/client information.

we believe these updates are not implemented in any Discv5 client. Furthermore, considering that over a year has passed since Discv5's initial deployment, we are skeptical about the imminent implementation of these updates.

**Observation 2.** *As of January 2025, no Discv5 client has implemented any mechanism to prioritize same-service node discovery and connection.*

**Service Distribution.** We analyzed the distribution of 7 134 services across nodes identified by unique public keys, which serve as persistent identifiers in EGN. In Discv4, two nodes provide the same service if they share identical protocolVersion, networkID, genesisHash, and forkID. As detailed in Sec. 2.3, Discv5 does not provide a universal mechanism for determining a node's service. We thus extract client information using the agentVersion and protocols strings.

The service distribution remains relatively stable across snapshots. We present the results from December 17, 2024, at 8:12 UTC, which has the highest number of Discv4 nodes. Of the 113 524 active Discv4 nodes, 73 279 refused our handshake requests, primarily due to no available connection slots. Figure 1a illustrates the distribution of the remaining 39 795 responsive nodes across different services. The mapping between public keys and services is consistent: no public key was associated with multiple services. The most prevalent services are displayed in Table 1. ETH Mainnet is the only service among the ten most prevalent services in 2018 [15] that persists in 2024, albeit with a declining dominance from 54% in 2018 to 31%. Of the 215 202 active Discv5 nodes, 197 160 provide no service information—their agentVersion or protocols strings are empty, and they refused our handshake. Figure 1b illustrates the client distribution of the remaining 18 042 nodes. The most popular clients (Table 2) include Ethereum consensus and execution clients, and clients of other services, such as Optimism, SSV-Node, and Base. Discv4 and Discv5 networks share 8 047 public keys, encompassing both Ethereum forks and some other services, such as LUKSO and Taiko.

Table 1: Top 10 Discv4 services.

| Service | Fork ID | Node Count |
|---|---|---|
| ETH Mainnet | 9f3d | 12404 |
| Polygon | f097 | 4947 |
| Story Odyssey Testnet | cef5 | 3179 |
| ETH Mainnet | be46 | 2970 |
| 102125 | 9c42 | 2219 |
| BSC Mainnet | 60ad | 1934 |
| ETH Holesky | 9b19 | 1554 |
| Xdai | 1384 | 1033 |
| Syscoin | 10d4 | 702 |
| ETH Sepolia | 88cf | 643 |

ETH Mainnet#be46 is an older fork of the Ethereum blockchain; "102125" corresponds to an unidentified service.

Table 2: Top 10 Discv5 clients.

| Client | Node Count |
|---|---|
| optimism | 8092 |
| Lighthouse | 4313 |
| Prysm | 1679 |
| teku | 1549 |
| SSV-Node | 869 |
| nimbus | 549 |
| base | 521 |
| lodestar | 236 |
| conduit | 85 |
| rust-libp2p | 53 |

### 3.3 Service Distribution in the Nodes' DHTs

We investigate the composition of Discv4 nodes' DHT buckets, focusing on the proportion of peers offering the same service. Using the December 17, 2024 snapshot, we remove the 73 279 nodes with unknown services from all nodes' routing tables and analyze the remaining 39 795 nodes' presence in each other's routing tables. Due to the lack of service information in Discv5, a similar analysis is not feasible. Nevertheless, Observation 2 suggests that Discv5 would not perform better than Discv4 in this regard.

**Public Nodes.** Table 3 presents the results for services of varying popularity. The percentage of same-service peers is low, even for the top two services (less than 20%). This proportion further decreases to below 5% for all other services.

**Bootnodes.** Bootnodes play a crucial role in a service by serving as entry points for all new nodes. Nodes within a bootnode's DHT are more likely to be connected to by newly-joined nodes. As shown in Table 3, bootnodes also exhibit a low percentage of same-service peers.

**Observation 3.** *The majority of Discv4 public nodes (52.9%) have less than 5% of same-service peers in their DHTs; the investigated bootnodes do not prioritize same-service peers in their DHTs.*

## 4 Efficiency Evaluation

We now investigate whether sharing a single overlay network improves EGN's node discovery efficiency. We evaluate this by measuring the communication

Table 3: Average proportion of same-service peers in the nodes' DHTs. The number preceding the service name is the service prevalence ranking in EGN.

| rank. service | % nodes | % same-service peers in public nodes | % same-service peers in bootnodes |
|---|---|---|---|
| 1. ETH Mainnet | 31.18% | 18.36% | 33.91% |
| 2. Polygon | 12.43% | 17.59% | 19.60% |
| 6. BSC Mainnet | 4.69% | 3.52% | 10.30% |
| 7. ETH Holesky | 3.91% | 2.94% | 2.43% |
| 10. ETH Sepolia | 1.61% | 1.73% | 0.59% |
| 43. KCC Mainnet | 0.04% | 4.19% | 1.70% |

overhead associated with establishing outgoing connections. We then compare these costs to the estimated overhead incurred if each service maintained its own separate overlay. We do not consider any adversarial behavior in this section.

### 4.1 Node Discovery Efficiency in EGN

**Experimental Setup.** We focus on establishing a predetermined number of outgoing connections. Intuitively, popular services find same-service neighbors more easily. To ensure a comprehensive evaluation, we select six services with varying prevalence rankings: ETH Mainnet (1st), Polygon (2nd), BSC Mainnet (6th), ETH Holesky (7th), ETH Sepolia (10th), and KCC Mainnet (43th).

For each service, we modify its official client to log all Findnode messages and handshake requests. Additionally, we log the type of error encountered in case a handshake fails. The number of sent messages directly translates to communication costs, reflecting both bandwidth consumption and processing time.

Each client is deployed as a public node with server configurations identical to those of our crawler. The execution terminates upon reaching either the predefined outgoing connection limit (66 for BSC Mainnet and Polygon, and 16 for other services) or a twelve-hour timeout. Data collection was conducted in January 2025.

**Results.** Each client was tested three times to mitigate potential diurnal variations, and we report the results of the execution with the median number of established connections in Table 4. The number of Findnode messages and handshake requests exhibited consistent order-of-magnitude across executions, with a maximum variation of 47%.

In Discv4, all involved services established outgoing connections slowly and costly. None of the clients achieved the target number of outgoing connections within 12 hours. The three most prevalent services' clients nearly reach the target number of outgoing connections, while other services fall far short. The number of Findnode messages per successful connection ranged from 30 535 (Polygon) to 243 694 (ETH Holesky), and the handshake success rate varied from 1/3842 (Polygon) to 1/45908 (KCC Mainnet). In contrast, we estimate the handshake

Table 4: Communication costs and effectiveness of node discovery in Discv4 (the first five lines) and Discv5 (the last three lines).

| rank. service | neighbors (out+in) | Findnode | handshake | TCP error | enc. error | mismatched protocolStr. | wrong service | no slot |
|---|---|---|---|---|---|---|---|---|
| 1. ETH Mainnet | 15 + 25 | 1367077 | 191571 | 1809 | 4881 | 449 | 82118 | 821 |
| 2. Polygon Mainnet | 64 + 129 | 1954232 | 245912 | 18382 | 44337 | 8148 | 151427 | 21921 |
| 6. BSC Mainnet | 62 + 74 | 2120671 | 332135 | 14156 | 16013 | 2 | 163477 | 16477 |
| 7. ETH Holesky | 6 + 17 | 1462161 | 189323 | 38953 | 79541 | 2376 | 83471 | 3511 |
| 10. ETH Sepolia | 11 + 19 | 1412537 | 202533 | 21916 | 39143 | 16 | 96113 | 5051 |
| 43. KCC Mainnet | 6 + 5 | 1319616 | 275446 | 24997 | 49669 | 4 | 146477 | 16477 |
| v5. ETH Mainnet | 0 + 3 | 1182935 | 191929 | 14174 | 323 | 525 | 71142 | 113 |
| v5. ETH Holesky | 0 + 0 | 1151383 | 192635 | 10551 | 31211 | 1908 | 93170 | 903 |
| v5. ETH Sepolia | 0 + 1 | 1102496 | 193127 | 19676 | 477 | 237 | 92115 | 2017 |

"Neighbors": outgoing and incoming connections upon termination after 12 hours; "Findnode" and "handshake": numbers of "Findnode" messages and handshake requests initiated by our node; the rightmost five columns are five types of handshake failures: TCP connection error, session key negotiation error, mismatched protocolString, wrong service, and no incoming connection slot.

success rate in the Bitcoin network to be 1/4. The reasoning behind this estimate is founded on [7] and is explained in Appendix A.

In Discv5, our ETH execution clients failed to establish any outgoing connections within 12 hours. This can be attributed to the presence of a substantial number of consistently online nodes (Observation 1). These nodes, likely consensus nodes of Ethereum and other proof-of-stake blockchains, occupy a significant portion of all Discv5 nodes' DHTs and persistently reject handshake requests.

**Observation 4.** *In Discv5, the two-thirds consistently online nodes occupy 81% of the nodes' DHTs, degrading the node discovery efficiency for other services.*

### 4.2 Node Discovery Efficiency in Dedicated Networks

We analyze potential efficiency gains from service-specific overlay networks using two approaches. First, we establish a lower bound by excluding three types of handshake failures clearly targeting wrong services and Findnode messages sent to other services. Appendix B provides a detailed explanation of these five error types. This results in 45.6% (ETH Mainnet) to 87.3% (ETH Holesky) efficiency gains in both bandwidth and execution time across all services. This approach underestimates the true benefits. Our next analysis suggests a large portion of *miscellaneous failures*—those not listed in the table—are likely due to EGN's blended architecture.

Our second approach, based on statistics of incoming handshake requests, bypasses this limitation and thus provides a more accurate estimate. As shown in Table 4, handshake failures due to "TCP connection error" and "no incoming

connection slot" range from 1.3% (ETH Mainchain) to 22.4% (ETH Holesky). This implies that all other handshake attempts, including the aforementioned miscellaneous failures, are *normal*: they reach their intended targets and, if the target is of the same service, find available slots. Our incoming handshake requests are a random sample of these normal attempts. During our ETH Mainnet client executions, before its incoming slots were full, only 1.83% of these requests resulted in successful connections. This data suggests that between $(1-22.4\%)\times 98.17\% = 76.2\%$ and $(1-1.3\%)\times 98.17\% = 96.9\%$ of all handshake attempts ultimately targeted a different service.

**Observation 5.** *In EGN, over three-quarters of handshake requests are directed towards nodes of other services.*

## 5 Security Evaluation

To assess the security benefits offered by EGN, we evaluate its resilience against a common P2P network threat: the DHT pollution attack. Since a robust DHT is the first line of defense against many network- and application-layer attacks, compromising EGN's resistance to this foundational attack would indicate a broader vulnerability. In other words, if EGN is more vulnerable to this weakest attack than dedicated overlay networks, we can conclude that EGN downgrades the attack resistance to all network-layer attacks.

Our approach differs from prior Ethereum network attacks in two key aspects. First, we focus on displacing honest peers in the nodes' DHT routing tables, contrasting with previous works that targeted application-layer connections. In fact, prior studies assume homogeneous routing tables populated exclusively by same-service peers before being attacked. Second, unlike prior studies that exploit protocol or implementation flaws, our attack leverages the inherent characteristic of EGN's shared overlay. The attack becomes ineffective when each service uses a dedicated overlay network.

### 5.1 EGN Pollution Attack

**Threat Model.** Our resource-constrained attacker controls a few hundred IP addresses, representing a small fraction (less than 0.3%) of the network. These resources are insufficient for a Sybil attack. A single server can manage their message traffic due to the low bandwidth and computational requirements of our attack. We still refer to these controlled entities as *attacker nodes* for clarity.

Our attacker operates independently of any service-specific logic and exploits no protocol or implementation flaws. All honest nodes function as intended, following the DHT maintenance workflows described in Sec. 2.2. Notably, our attack remains effective even though previously identified DHT vulnerabilities [8, 11–13] are already addressed.

**Attacker's Goal.** The attacker aims to pollute the DHT buckets of EGN nodes with the attacker nodes' ENRs. This manipulation further lowers the success rate

of application-layer handshakes. In the most extreme case, a group of nodes have no honest peer from the rest of the network in their DHTs, forming a partition.

Complete isolation is more challenging than removing all same-service peers. We consider a partition attempt unsuccessful if even one node within the group has an active and honest public peer outside the group in its DHT. This is because before the node initiates a handshake to the peer, it would, with high probability, send a Findnode message to the peer, which may reply to the node with some same-service peers, breaking the partition. Additionally, newly joining nodes with peers in both the isolated group and the broader network can bridge the partition. Therefore, a group of nodes is partitioned only if the union of their DHTs includes only inactive peers, attacker nodes, and other nodes of the group.

We highlight two capability limitations caused by our weak threat model. First, the attacker cannot remove long-term active honest peers from nodes' DHTs. These peers undergo liveness checks before eviction ("ENR Insertion and Deletion" in Sec. 2.1), ensuring their persistence within the DHT. As long as a node retains some long-term peers, it cannot be completely isolated unless all such peers are in the same partition. Second, partitioned nodes may continue to receive new blocks and transactions until their existing TCP connections terminate. A public node could be partitioned in our definition but still maintain synchronization through an application-layer connection traversing a NAT.

**Attack Strategy.** Our attack leverages two components: a crawler and a polluter. The crawler, as described in Sec. 3.1, continuously scans the network to maintain an up-to-date list of active nodes and their DHTs. The polluter, running on each attacker node, sends Ping message every second to all discovered active nodes. This interval is shorter than the liveness check's ten-second interval, granting attacker nodes an advantage over honest ones. Due to the "Passive DHT Maintenance" mechanism (Sec. 2.2), these messages insert the attacker node's ENR either in the target node's corresponding DHT bucket, or in the latest replacement slot. In the latter case, it would be promoted into the DHT bucket as long as a peer in the bucket fails the liveness check. To avoid being evicted, attacker nodes always respond promptly to Ping messages. Additionally, they reply to all Findnode requests with Neighbors messages containing only attacker node ENRs, passively propagating them throughout the network.

**Cost Estimation.** The primary cost lies in the infrastructure: each attacker node involves one IP address, no more than 1 GB of memory, and minimum bandwidth. These requirements are modest: the Standard DS1 v2 instance of Microsoft Azure can fulfill these needs, costing around 50 USD per node per month [29]. Therefore, for a deployment of up to 300 attacker nodes, the estimated monthly cost of continuously attacking EGN remains under 15 000 USD.

**Simulation Environment.** To avoid disrupting real-world services, we evaluate the attack in a simulated environment (detailed in Appendix C). We reuse Geth's source code where possible and faithfully implement its network communication to overcome our resource limitations. Network characteristics are derived from our measurements in Sec. 3.2 and 3.3. We experimentally verify that our simulated environment accurately models EGN's dynamics. We use the

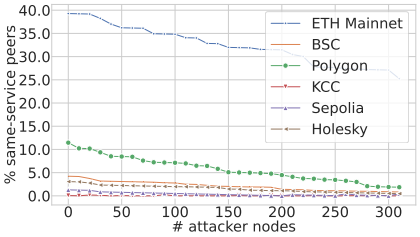



Fig. 2: Percentage of same-service peers in nodes' DHTs in EGN.

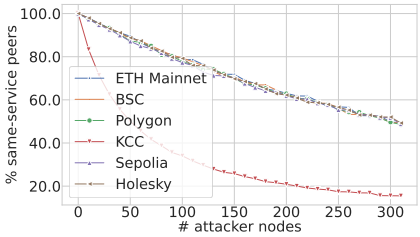



Fig. 3: Percentage of same-service peers in nodes' DHTs in dedicated networks

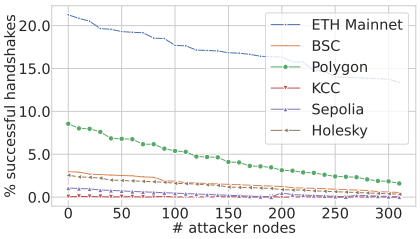



Fig. 4: Handshake success rate in EGN.

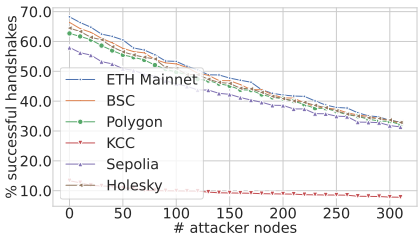



Fig. 5: Handshake success rate in dedicated networks.

service distribution of Discv4 as such information is not available in Discv5. For comparison, we also simulate a "dedicated networks" setting.

### 5.2 Same-Service Peers in DHTs and Handshake Success Rates

**Impact on DHTs.** We simulate one to 330 attacker nodes; each data point is averaged over ten runs. Figure 2 reveals a substantial decrease in the average percentage of same-service peers within nodes' DHTs across six services. Despite the attack's simplicity, the percentage plummets to near or below 1% for four services with 300 attacker nodes. ETH Mainnet and Polygon benefit from larger pools of long-term same-service nodes. EGN's vulnerability stems from its high percentage of different-service peers before the attack. Our attack exacerbates the issue by removing all temporarily inactive peers—same or different service.

Figure 3 demonstrates the contrasting behavior in dedicated networks. Here, the percentage of same-service peers remains consistently above 48% for five services. This resilience stems from a more favorable initial state—nodes only have same-service peers in their DHTs; they will not be washed out as long as they remain active. KCC Mainnet exhibits lower performance—15% with 300 attacker nodes—due to its smaller public node count.

**Impact on Handshake Success Rates.** For four out of six services, the success rates in EGN rapidly decline to below 1% (Fig. 4). These success rates are higher

Table 5: Network partitions with 300 attacker nodes.

| rank. service | EGN | | dedicated networks | |
|---|---|---|---|---|
| | # connected components | #, % partitioned nodes | # connected components | # , % partitioned nodes |
| 1. ETH Mainnet | 7.0 | 4124, 33.35% | 5.3 | 155, 1.26% |
| 2. Polygon | 7.2 | 1389, 28.15% | 5.3 | 49, 0.99% |
| 6. BSC Mainnet | 7.6 | 616, 33.05% | 5.0 | 18, 0.97% |
| 7. ETH Holesky | 7.2 | 576, 37.20% | 5.0 | 15, 0.96% |
| 10. ETH Sepolia | 7.1 | 147, 23.09% | 5.2 | 6, 0.93% |
| 43. KCC Mainnet | 5.3 | 9, 21.20% | 5.0 | 11, 25.50% |

than in Sec. 4.1, probably because real-world Discv4 DHTs also contain *private nodes*—those behind firewalls or NATs—that are inaccessible.

Dedicated networks exhibit higher success rates. A decreasing trend is expected as public nodes cannot distinguish attacker nodes before initiating handshakes. Naturally, more attacker nodes lead to lower success rates; yet still above 30% for five services. KCC Mainnet performs the worst—8% with 300 attacker nodes—due to its small node pool.

### 5.3 Network Partitioning

We now analyze the attack's most severe consequence: network partitioning. We run ten simulations with 300 attacker nodes in each. Table 5 displays the average number and size of the partitions. EGN experiences severe partitioning, with 21% to 33% of nodes partitioned for each service. In contrast, few nodes are partitioned in dedicated networks. Our analysis reveals two key reasons for their resilience. (1) *More application-layer connections:* nodes maintain more connections due to their high handshake success rates. These connections result in the corresponding nodes' ENRs entering the DHTs through "Passive DHT Maintenance". (2) *Bootnode coverage:* dedicated networks, being smaller, allow bootnode DHTs to cover a larger portion of honest nodes. When partitioned, newly joined nodes and active nodes reconnecting to the bootnodes have a higher chance of bridging the gaps. In contrast, EGN's bootnodes cover a smaller fraction, leaving newly joined nodes with no way to rejoin partitions if bootnode DHTs only contained peers from the same connected component.

**Observation 6.** *The scarcity of same-service peers in EGN nodes' DHTs compromises its robustness against DHT pollution attacks.*

## 6 Fixing EGN

**Service-Specific DHT.** EGN's low node discovery efficiency and vulnerability to pollution attacks stem from its blended architecture, where nodes store peers

from various services in their limited DHT buckets. This can be addressed by extending the ENR format to include all supported services of the node, in line with Discv5's original design. We believe the service information should not be optional. This approach offers two advantages. (1) Efficiency: other nodes can discover a node's services without direct interaction. (2) Potential interoperability: nodes can still receive ENRs of different services, preserving the potential for interaction. Each service can then decide how to manage its DHTs—storing only same-service peers or accommodating specific other services—potentially with a separate routing table as suggested in [16]. This allows EGN to enjoy higher node discovery efficiency and stronger security without sacrificing decentralization.

**Reliable Bootnodes.** Our simulations also highlight the crucial role of bootnodes as a last defense against network partitioning. However, as shown in Table 3, most services have bootnodes covering only a small fraction of their public nodes. If a partition includes no nodes represented in the bootnode DHTs, network recovery becomes highly improbable. Therefore, we recommend that most services: (1) increase the number and reliability of their bootnodes, and (2) configure bootnodes to prioritize storing same-service nodes and increase the size of their routing tables.

## 7 Related Works

**Design.** In a parallel study, DISC-NG was proposed to address the inefficiencies of Discv5 [16]. While the authors recognized EGN's low node discovery efficiency as a problem, they did not identify the underlying cause. Our research reveals that EGN's blended architecture is the root of this issue. Furthermore, we demonstrate that this architecture also compromises EGN's security, and quantitatively analyze its negative impact on both security and efficiency through comparisons with dedicated networks. These analyses are not present in [16].

**Security.** Prior security research primarily focuses on individual services within the network. These studies measure a service's resilience against Sybil attacks [8], network partitioning [13], eclipse attacks [11,12], and Denial-of-Service attacks [25]. Notably, all these studies assume a specific scenario where honest nodes' DHTs (if used) contain only same-service nodes and attacker nodes. None of them has examined the security of the EGN ecosystem as we do.

**Measurement.** Existing measurements focus on EGN's node statistics [8, 15, 20], protocol messages [14], and topology [18, 21]. None of these studies explore the individual DHT composition as we do.

*Remark.* We contacted the Ethereum Foundation via their official support email address several months ago, but have not yet received a response. We have recently included the Discv5 measurement data and plan to contact them again.

## 8 Conclusion

Since its inception in 2015, EGN's blended architecture has aimed to provide "a place for everyone," where all services, encompassing over 500 billion USD in

market capitalization, share a single overlay network. This architecture has been largely unchallenged, and some consider it beneficial for node discovery and attack resistance. However, our research reveals no measurable advantages to this approach. Compared to dedicated overlay networks, services within EGN experience: (1) a 50% to 99% increase in bandwidth waste during node discovery due to irrelevant service entries, and (2) increased susceptibility to DHT pollution attacks and network partitioning, which leaves them vulnerable to further exploitation. These drawbacks are particularly detrimental to small services, whose nodes nowadays only connect to their services' official bootnodes and servers. Consequently, these bootnodes and servers become single points of failure, easily compromised by inexpensive attacks. In essence, the node discovery protocol wastes significant bandwidth without achieving its intended purpose. We urge the community to adopt a preference for organizing as "everyone in its place," that is, to prioritize same-service peers in node DHTs, especially those of bootnodes, which contribute more to network security than previously recognized.

## References


1. Networks | ethereum.org (2023), https://ethereum.org/en/developers/docs/networks/
2. Cryptocurrency Market Capitalizations (2025), https://coinmarketcap.com/
3. Bouguera, A.: Ethereum Underlying P2P network (2020), https://p2p.paris/gen/attf4zFDPfhauUHyj-The_p2p_network_behind_Ethereum.pdf
4. Buterin, V.: Ethereum: A next-generation smart contract and decentralized application platform (2014), https://github.com/ethereum/wiki/wiki/White-Paper
5. Chen, T., Li, Z., Zhu, Y., Chen, J., Luo, X., Lui, J.C.S., Lin, X., Zhang, X.: Understanding Ethereum via graph analysis. ACM Trans. Internet Technol. **20**(2) (April 2020)
6. Dean: From Kademlia to Discv5 (2020), https://vac.dev/rlog/kademlia-to-discv5/
7. Donet, J.A.D., Pérez-Solà, C., Herrera-Joancomartí, J.: The Bitcoin P2P network. In: Financial Cryptography and Data Security - FC 2014 Workshops, BITCOIN. Lecture Notes in Computer Science, vol. 8438, pp. 87–102. Springer (2014)
8. Eisenbarth, J.P., Cholez, T., Perrin, O.: Ethereums peer-to-peer Network Monitoring and Sybil Attack Prevention. In: Journal of Network and Systems Management (2022)
9. Gao, Y., Shi, J., Wang, X., Tan, Q., Zhao, C., Yin, Z.: Topology Measurement and Analysis on Ethereum P2P Network. In: 2019 IEEE Symposium on Computers and Communications (ISCC) (2019)
10. Guo, D., Dong, J., Wang, K.: Graph structure and statistical properties of Ethereum transaction relationships. Information Sciences **492**, 58–71 (2019)
11. Heilman, E., Kendler, A., Zohar, A., Goldberg, S.: Eclipse Attacks on Bitcoin's Peer-to-Peer Network. In: SEC'15: Proceedings of the 24th USENIX Conference on Security Symposium (2015)
12. Henningsen, S., Teunis, D., Florian, M., Scheuermann, B.: Eclipsing Ethereum Peers with False Friends. In: 2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW) (2019)
13. Heo, H., Woo, S., Yoon, T., Kang, M.S., Shin, S.: Partitioning Ethereum without Eclipsing it. In: Network and Distributed System Security (NDSS) Symposium (2023)

14. Kiffer, L., Salman, A., Levin, D., Mislove, A., Nita-Rotaru, C.: Under the Hood of the Ethereum Gossip Protocol. In: Financial Cryptography and Data Security: 25th International Conference, FC 2021 (2021)
15. Kim, S.K., Ma, Z., Murali, S., Mason, J., Miller, A., Bailey, M.D.: Measuring Ethereum Network Peers. In: IMC '18: Proceedings of the Internet Measurement Conference 2018 (2018)
16. Król, M., Ascigil, O., Rene, S., Sonnino, A., Pigaglio, M., Sadre, R., Lange, F., Rivière, E.: DISC-NG: robust service discovery in the Ethereum global network. In: 9th IEEE European Symposium on Security and Privacy, EuroS&P 2024. pp. 193–215. IEEE (2024)
17. Lee, X.T., Khan, A., Sen Gupta, S., Ong, Y.H., Liu, X.: Measurements, analyses, and insights on the entire Ethereum blockchain network. In: Proceedings of The Web Conference 2020. WWW '20, ACM (2020)
18. Li, K., Tang, Y., Chen, J., Wang, Y., Liu, X.: Toposhot: Uncovering Ethereums Network Topology Leveraging Replacement Transactions. In: IMC '21: Proceedings of the 21st ACM Internet Measurement Conference (2021)
19. Li, Z., Xia, W., Cui, M., Fu, P., Gou, G., Xiong, G.: Mining the Characteristics of the Ethereum P2P Network. In: Proceedings of the 2nd ACM International Symposium on Blockchain and Secure Critical Infrastructure (2020)
20. Maeng, S.H., Essaid, M., Ju, H.T.: Analysis of Ethereum Network Properties and Behavior of Influential Nodes. In: 2020 21st Asia-Pacific Network Operations and Management Symposium (APNOMS) (2020)
21. Maeng, S.H., Essaid, M., Lee, C., Park, S., Ju, H.t.: Visualization of Ethereum P2P network topology and peer properties. In: International Journal of Network Management 2021 (2021)
22. Maeng, S.H., Essaid, M., Park, S., Ju, H.T.: Discovery of Ethereum Topology Through Active Probing approach. In: 2021 22nd Asia-Pacific Network Operations and Management Symposium (APNOMS) (2021)
23. Maymounkov, P., Mazieres, D.: Kademlia: A peer-to-peer information system based on the XOR metric. In: International Workshop on Peer-to-Peer Systems. pp. 53–65. Springer (2002)
24. nkeywal: Gossipping ENR protocol (2019), https://github.com/ConsenSys/wittgenstein/issues/35
25. Taverna, M., G. Paterson, K.: Practical Attacks on Go Ethereum Synchronising Nodes. In: SEC'23: Proceedings of the 32th USENIX Conference on Security Symposium (2023)
26. Trautwein, D.: Nebula: A network agnostic DHT crawler (2025), https://github.com/dennis-tra/nebula
27. Trautwein, D., Raman, A., Tyson, G., Castro, I., Scott, W., Schubotz, M., Gipp, B., Psaras, Y.: Design and evaluation of IPFS: a storage layer for the decentralized web. In: Kuipers, F., Orda, A. (eds.) ACM SIGCOMM '22. pp. 739–752. ACM (2022)
28. Wang, T., Zhao, C., Yang, Q., Zhang, S., Liew, S.C.: Ethna: Analyzing the Underlying peer-to-peer Network of Ethereum Blockchain. In: IEEE Transactions on Network Science and Engineering (2021)
29. Willocx, M., Bohé, I., Naessens, V.: Value for money: An experimental comparison of cloud pricing and performance. In: CLOSER. pp. 140–148 (2024)
30. Zhao, L., Sen Gupta, S., Khan, A., Luo, R.: Temporal analysis of the entire Ethereum blockchain network. In: Proceedings of the Web Conference 2021. WWW '21, ACM (2021)

## A Handshake Success Rate in the Bitcoin Network

Due to the lack of directly comparable studies, we derived this estimate through an approximation based on data from Joan et al. [7]. They measured the Bitcoin P2P network and found routing information for 16 000 nodes in the routing tables of all nodes at a given time. Among these, roughly 6,000 nodes were constantly active, and 70% of those can accept connection requests. Assuming a conservative estimate that a Bitcoin node's routing table contains only 37.5% (6,000 out of 16,000) of these stable public nodes—this is conservative because Bitcoin clients prioritize stable nodes, then $37.5\% \times 70\% = 26.25\%$ of the routing table entries represent active and available nodes, resulting in a handshake success rate of 1/4.

## B Explaining Five Types of Errors

*Session key negotiation error* (denoted as "enc. error" in Table 4) occurs when two nodes cannot agree on a common key negotiation algorithm, which only happens between nodes from different services. *Mismatched* protocolString indicates that two nodes could not find a compatible protocolString. This scenario only arises when the target node belongs to a different service, as our client supports all protocols within its own service. *Wrong service* signifies that at least one of the three service identifiers (networkID, genesisHash, and forkID) does not match. We explain the remaining two types for completeness. *TCP connection error* ("TCP error" in Table 4) signifies our client's inability to establish a TCP connection with the target node. *No incoming connection slot* ("no slot" in Table 4) indicates that the target node has reached its connection capacity, which occurs between nodes of the same service.

## C Simulation Environment

Ideally, emulating EGN would involve running unmodified clients in an isolated network. However, this approach demands computational resources exceeding our capacity. Existing studies using this approach, such as [16], can only maintain thousands of nodes, which falls short of EGN's scale by two orders of magnitude.

Therefore, we adopt an alternative approach: we reuse Geth's source code where possible, while simulating time-consuming and resource-intensive operations through a faithful implementation of their logic. To ensure our simulated system's fidelity, we emulate a smaller-scale EGN of 900 nodes using actual clients within a Kubernetes environment[4]. We then compare the experimental results obtained from our simulated system and the Kubernetes system under identical network conditions and initial configurations. Next, we first detail our simulated environment, and then present our verification results.

[4] https://kubernetes.io/

In our simulated environment, we represent each node as a data structure on disk rather than as a continuously running thread, minimizing memory and CPU consumption. Consequently, all node operations are modeled as events—either periodically triggered (e.g., liveness checks) or initiated by other events (e.g., receiving a Ping message). Furthermore, application-layer data exchange (e.g., transactions) is excluded as it is outside the scope of this study.

Despite these simplifications, our simulation accurately reflects EGN's dynamics. Simulated nodes replicate the routing table maintenance and handshake initiation logic of the actual client. We reimplement Geth's network communication line-by-line in Python, replicating the behavior. The initial network size, service distribution, and distribution of services and active peers within bootnodes' and public nodes' DHT buckets are directly derived from our measurements in Sec. 3.2 and 3.3. Node churn is simulated based on our measured node uptime distribution, with new nodes joining through their services' bootnodes.

For comparison, we also simulate a "dedicated networks" setting. Here, a service has the same number of nodes as in EGN, but their initial DHTs only contain same-service peers. An attacker node joins all networks and disseminates its ENRs using Ping and Neighbors as in EGN.

A simulation runs for a simulated 24-hour period, with zero to 400 attacker nodes. We log the results of each handshake request and the final state of nodes' DHT buckets at the simulation's conclusion. The simulation and data processing modules comprise roughly 2.2 thousand lines of code.

**Verification Results.** Both the Kubernetes system and our simulated environment consist of 600 honest nodes operating the same service and 300 attacker nodes (the Kubernetes system has a capacity limit of 900 nodes). Network churn is not considered, and all nodes are assumed to remain active throughout the execution. Other network conditions and initial configurations adhere to the previously described setting.

Following a 24-hour attack, honest nodes in both systems fragment into four partitions. The largest partition comprises 590 nodes in the Kubernetes system and 584 nodes in the simulated environment. The average percentage of same-service peers within DHTs declines to 10.9% in the Kubernetes system and 8.9% in the simulated environment. These comparable results provide strong evidence that our simulated environment accurately reflects the network's dynamics.