Proliferation of the Service-centric Distributed Consensus Model and its Impact on Ethereum

***Abstract*—In distributed consensus systems (DCSs), a single peer exposes functionality to other peers to agree on a shared state for a computational problem, such as for cryptocurrencies and distributed file systems. We observe, however, that this original, peer-centric model has evolved towards deploying several peers at a single network location, thus exposing the same DCS services many times, driven by the fees and rewards that can be gained by doing so. We refer to this trend as the *service-centric model* and provide in our paper evidence for this trend, its growth, and its impact on DCS operations, using empirical observations in the Ethereum system. Specifically, we shed light on the opposing observations of increasing reliance on highly available cloud infrastructures and large numbers of non-reachability events in the DCS. We provide recommendations on how to tackle this impact through changes to the Ethereum platform and identifier generation, believing that those recommendations and our empirical observations provide useful insights for building resilient and bias-free DePIN platforms.**

I. Introduction

The vision of a distributed consensus system (DCS) foresees participating peers to achieve consensus over a shared space in a continuous computation, where the system does not exhibit bias towards the location or nature of the peer in the participating infrastructures in which the peers are deployed. More so, Bitcoin [28], Ethereum [33], Stellar [24], Chia [6], and Algorand [9], among others, explore a permissionless or open membership character for participating in such DCS, imposing thus less stringent requirements on the peers.

Participation in the DCS is motivated by benefits from fees and rewards, i.e., peers collect fees for committing and prioritizing changes to the shared state [8], while the DCS rewards peers for maintaining a coherent state [28].

This vision aligns well with a decentralized physical infrastructure (DePIN), where the participating physical resources equal peers in a DCS, with tokens used as rewards for provisioning and using the decentralized resources.

However, as our main contribution, we outline in this paper that the vision of complete decentralization in DCSs needs to be more effectively skewed by the realities of deploying peers on the Internet. More specifically, we show through empirical evidence that only very few major cloud-based infrastructures contribute to the consensus-making process, thus opening the potential for bias and impacting the resilience of the overall distributed system. The critical issue is that despite highly available peers in the DCS, achieved through deploying them in highly available cloud provider networks, reachability within the DCS remains an issue.

We then derive two concrete recommendations as our second contribution, which not only allow for improving existing DCS platforms through specific changes to their software basis but also consider them in the design for the platforms that form the control plane for the emerging DePIN.

For this, we start outlining in Section II the observed trend away from deploying single peers in a DCS towards deploying many service endpoints within highly available cloud infrastructures. We then show in Section III the empirical observations for this trend in the now-deployed Ethereum system, aligning with previously reported results from other DLT systems [17], [32]. Through this, we explain in Section IV the seemingly contradicting observation of unreachability of highly available peers in the DCS, leading us to concrete recommendations in Section V to counter the impact of the observed infrastructure consolidation on the DCS, before concluding in Section VI.

II. Move to a Service-centric DCS

In order to deploy a peer, users download and launch applications, where those applications realize the functionality required to receive and diffuse a shared state. As an example for Ethereum, software such as geth [12], [13], open Ethereum [11], erigon [10], nethermind [29], prysm [22], and others, provide functionalities, e.g., gossiping, through an application, as illustrated in Fig. 1-a, being geth the dominating application in the Ethereum system [14].

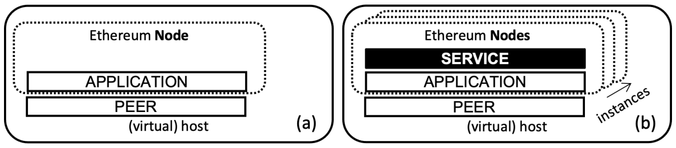

Fig. 1. (a) Peer Centric Model and (b) its Evolution

This application actively probes for peers to build and replenish a collection of successful connections, as outlined in [18]. The required lookups and connection functionalities run in a single IP-based endpoint, i.e., a peer exposes the overall functionality only once to the network. Thus, realizing a functionality per peer, we refer to this as the *peer-centric model* for the rest.

Contrasting this peer-centric view of a DCS, one can look at the functionality of a peer as providing a *service* to other peers. More so, aligned with known concepts of service provisioning, that service may be exposed by more than one *service instance*. In Ethereum, this is achieved through running multiple replicas in a single peer; we label this mode of DCS provisioning as following a *service-centric model*.

Motivated by boosting rewards and fees, peers join to partake in the DCS with several running service instances, as in Fig. 1-b in order to increase the likelihood of earning rewards and seizing more transaction requests [8]. This tendency, shown in Fig. 1-b, denotes the evolution of the peer-centric, original deployment to that of deploying several peer services in one network location.

III. Evidence for the Service-Centric Trend

Let us dive deeper into the evolution from the peer-centric model, a growing divergence between peers and services, a service-oriented trend mainly driven by profits.

Fees and rewards are **drivers** that motivate a service-oriented trend. Peers tune service instances to boost their participation in consensus building. Any peer joining the DCS may run several active instances to increase participation and, therefore, the probability of earning rewards; this is because the DCS aims to equally distribute dividends among the running services in the system [28], [33].

Peers realize particular **mechanisms** to expand their participation in the DCS. These mechanisms include configuring aggressive fanouts, short-timing pool replenishments [34], and exposing more than one service instance.

First, a pool configured with an aggressive fanout aims at diffusing a distributed shared state with low latency by reaching more peers than the default [18], [34]. Peers configuring a higher than the default number of endpoints to grow their pool degrees realize that aggressive fanout. In Ethereum, for example, peers set more than the default pool size of fifty endpoints. This high peer degree distribution is reported in [17], [27] for Bitcoin, in [16] and [21] as node degree for Ethereum, and in [32] for XRP Ledger.

Second, faster pool replenishment seeks to diversify the pool fanout to reach more peers in shorter periods while diffusing the distributed shared state. For example, the default timeout for establishing a new connection in Ethereum is fifteen seconds [12].

Lastly, exposing many service instances per peer increases partaking in the distributed state consensus.

In Ethereum, application replication er-peer realizes this service instance exposure, realizing consensus participation and state diffusion as a service-centric model.

We contrast the peer-centric against the service-centric model and observe the discrepancy between the number of peer IP addresses and service instances to introduce the **empirical evidence for the trend**.

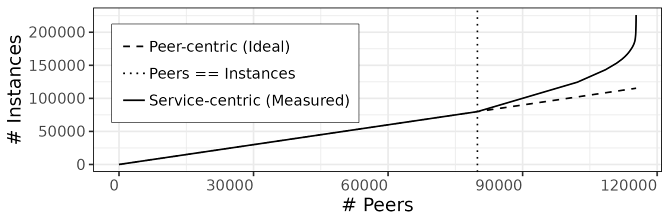

Fig. 2. The Service-centric Trend

Fig. 2 highlights the degree of the current divergence between service instances and peers, using empirical evidence from $\approx 115k$ different peers identified by IP addresses. These peers were discovered in campaigns between 2022 and 2023, as detailed in [18], [31]. In these campaigns, 86% peers actively provided $\approx 185.5k$ service instances. This discrepancy allows us to elaborate on the starting gap at $79.9k$ peers, meaning that 69% peers find a one-to-one correspondence between an IP address and a service instance ID. This gap grows non-linearly to points of reachability with a maximum degree of exposing $\approx 1.7k$ service instances in a peer.

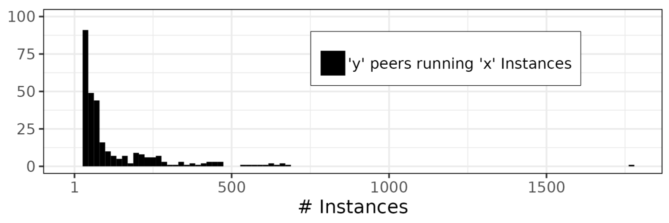

Fig. 3. The Growth in the Service-centric Trend

In Fig. 3, we show that this growth of the service-centric trend has followed a long tail distribution. This tailed growth means that few peers run many instances, and many run more than a single instance. For example, a single peer hosts 1771 instances, and around 175 peers provide between two and three service instances. Indeed, a few dedicated, well-known infrastructures aid this behavior, as explained in the following.

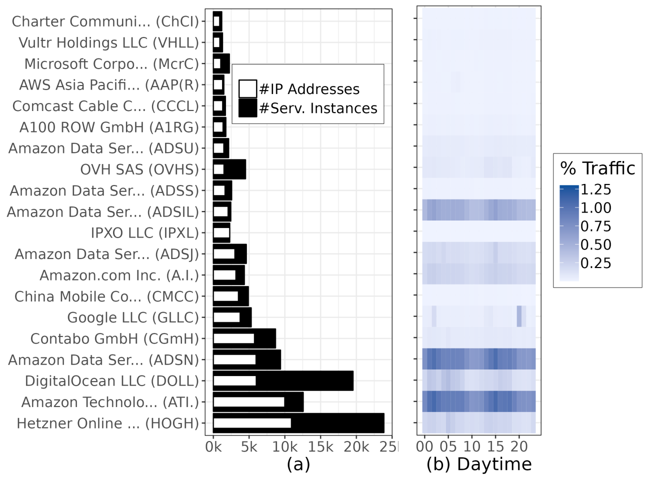

Fig. 4. (a) Infrastructures hosting Peers and Instances, and (b) its Traffic Concentration on a Day Time relative to CET Time sharing the same y-axis

The drivers and mechanisms previously outlined push the peers to seek highly available and reliable infrastructures. Hence, causing, as an orthogonal effect, a noticeable *infrastructure concentration*. Similar results for concentration have also been reported in [32] for XRP Ledger and in [17] for Bitcoin. Fig. 4-a presents this degree of accumulation based on an IP address to autonomous system (AS) translation [4]. Here, we consider an AS an infrastructure; it includes cloud providers, Internet service providers (ISPs), and hyperscalers identified by an AS number. With this in mind, we present the results for the top twenty most densely populated infrastructures. We found that 47.9% of the peers rely on ten out of 3795 infrastructures. But, when looking at the service instance distribution, we observe that 86.2% of the system instances run on the ten most influential infrastructures, with 21% of the instances running in the top infrastructure. Thus, *more than half of the system relies on less than ten infrastructures,* 0.26% *of all infrastructure providers.*

Intuitively, most instances in the main concentrated infrastructures generate *concentrated traffic*. Fig. 4-b characterizes the intensity of this traffic for each infrastructure as the percentage of requests a passive probe, i.e., a node, receives from the entire DCS. For example, during the entire measuring campaign of a passive probe in the DCS, the infrastructure ATI generated 21% of the whole traffic of the DCS, with approximately 1% of the requests of the entire DCS each hour, and HOGH generated 4.89% of the traffic in the Ethereum system. Notice, first, that more service instances do not necessarily lead to more traffic due to the tuned replenishment mechanisms detailed in Appendix A. Second, high intensity is concentrated in the top most populated infrastructures, with the second foremost provider generating the most traffic, 21.1% request out of the entire DCS.

Furthermore, service instances generate daily traffic, as shown in Fig. 4-b along the x-axis. Notice that service instance churning in these infrastructures is negligible to absent. This results as well from the economic driver outlined before. Moreover, we infer from the service traffic across the daytime and the hosting service that service instances are highly available: the traffic density across the day is profiled without gaps, and infrastructures such as HOGH and ATI in Fig. 4 run 32% of the instances in the system and offer the availability of 99.9% [19] and 99.99% [1] respectively. But, *even though highly available infrastructures support DCS instances, there are still issues with service reachability.*

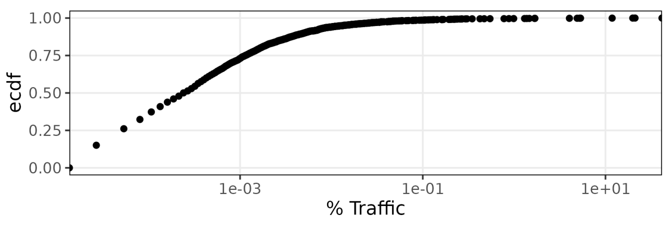

Fig. 5. ECDF for the Percentage of Traffic Generated by an Infrastructure

Moreover, when looking at the entire system as in Fig. 5, the top ten populated infrastructures generate approximately 77.2% of the total traffic in the system. The rest of the infrastructures produce less than 1% of the traffic in the DCS.

IV. Impact on the Ethereum Network

Before diving into the impact and issues, let us outline the communication pattern for establishing service relations.

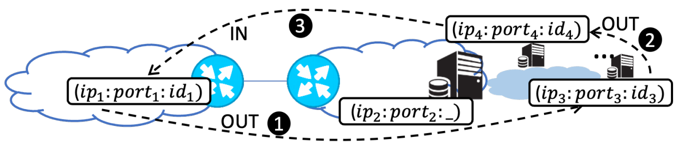

Fig. 6. Incoming and Outgoing Service Relation Establishment

Discovery and pool establishment are part of a service relation building, as described in [18, Section 2.2] and [21, Section 2]. These procedures are standard among instances in the DCS and categorized by the instance initiator into outgoing (OUT) and incoming (IN) relations. In Fig. 6, ❶ is an OUT relation, and ❸ an IN relation. The discovery populates a local list with tuples $(ip : id)$ by sending and receiving messages with neighboring service instances [25]. For example, $(ip_4 : port_4 : id_4)$ may discover $(ip_1 : id_1)$ through $(ip_3 : port_3 : id_3)$, as in ❷. The discovered list is tested for peer reachability and then randomized to establish an OUT relation, and since the remote peers are executing the same pattern, IN relations are expected. In Ethereum, an IN relation establishment includes a TCP connection establishment, security handshake, capability handshake, and checkpoint validation [18], [21]. These IN relations are the majority and shape the impact of reachability in the DCS [21].

Reachability affects the system's **resilience**, i.e., survivability, of the network. Under failures, infrastructure concentration represents a shortcoming in the system. A connectivity failure to an infrastructure such as HOGH will compromise 21% of the system. Indeed, a connectivity failure in the top ten infrastructures will undoubtedly affect 86.2% of the system. However, failures in infrastructures such as HOGH and ATI seem unlikely since their average network availability is 99.9% [19] and $99,99\%$ [1], respectively.

We outline how the service unreachability, accounting for 87% of issues out of all communication trials, impacts these infrastructures, also reported as 90% in [18], and 93.98% in [21]. Out of the 87% of reachability issues, $\approx 63\%$ are disconnections due to oversubscribed pools, 13% of the problems are security handshake errors, and the rest comprises capability and checkpoint validations. The oversubscribed disconnection is due to instances reaching its IN and OUT connection limits and denying further requests (too many peers in [21]). The security handshake errors relate to the service security context establishment (disconnects in [21]).

In Fig. 7, we illustrate the high percentage of **service reachability** issues impacting the entire system. The reachability issues are widely spread in the system, meaning that any passive probe at any geographical location will experience the same reachability issues from the entire DCS. Indeed, a service instance encounters disconnects proportionally to the degree of the infrastructure hosting.

This proportion of disconnections holds for more extensive hosting degree infrastructures, as shown in Fig. 8, meaning that the more services are concentrated in an infrastructure,

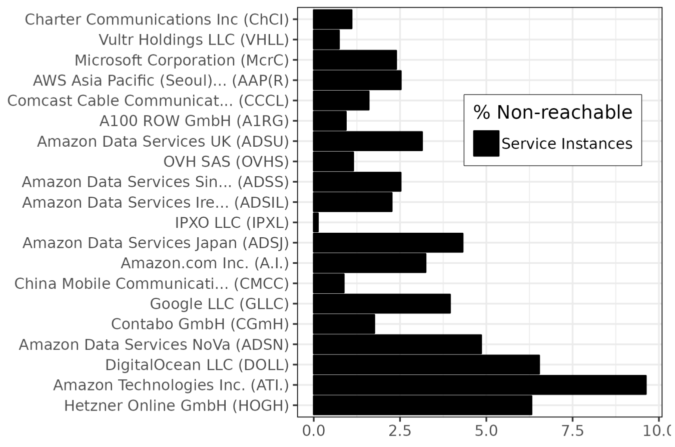

Fig. 7. Reachability Issues Spread along the DCS

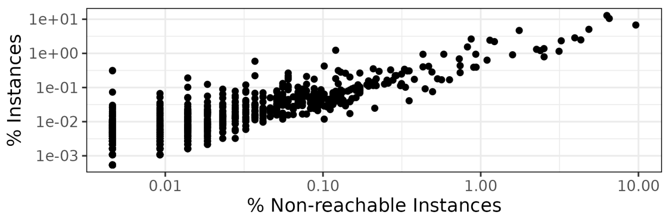

Fig. 8. Correlation for % Non-reachable against % Instances per Infrastructure

the more issues a peer finds. For example, ATI hosting 9988 peers with 12568 services exhibits $\approx$ 10% out of 87% of the reachability issues. In Fig. 8, this correlation is biased due to the tuning mechanisms, e.g., short-timed replenishment, outlined before and detailed in Appendix A.

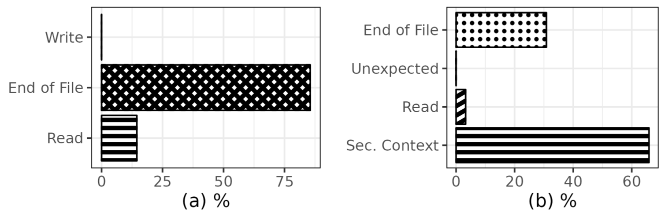

Fig. 9. (a) Outgoing and (b) Incoming Service Reachability Issues

As stated before, reachability issues are due to pool oversubscription and handshake issues. We detail these handshake issues for OUT and IN relations in Fig. 9.

The handshake for an OUT relation towards a remote instance requires a single remote point of reachability, i.e., a single IP address, as in ❶ in Fig. 6. However, if the response to this OUT request performs the latest used (virtual) instance through, e.g., point of reachability $(ip_2 : port_2 : \_)$ in Fig. 6 a host peer with many virtual service instances, it results in unsolicited responses. The unsolicited responses lead to an unexpected size of a message. The unexpected messages during handshakes cause exceptions where the application is not able to create a further response message, write exception, the application is not able to match a message in a protocol, read exception, or the application can not decode a response message, end of file (EOF) exception, in Fig. 9-a. Fig. 9-a and Fig. 9-b differ since they are logs for an outgoing service establishment and an incoming service relation, respectively.

Furthermore, IN relation issues in Fig. 9-b include 30% EOF errors due to decoding packets from unsolicited responses during handshakes [12], 3% read issues, and 65.9% from a **security context mismatch**.

More specifically, the service security context mismatch is due to the IP-centric lookup of a service relation. For example, if (ip_1, id_1) is discovered by a neighbor service instance sharing the same IP-based endpoint, as in ❷ in Fig.6. The local instance identifies the relation by the remote IP address ip_4 and loads a mismatched public key id_1; hence, the tuple for the remote endpoint is (ip_4, id_3) causing a message decryption exception that results in the high percentage of the service security context mismatch.

The high percentage of issues for IN relations matches the previous empirical observations in Section III of the service-centric model, where single IP endpoints host thousands of service instances. It is also worth noting that the reported reachability issues are given for a single probe crawling the DCS by binary distances through the system IDs. These issues are expected for the entire DCS, including the so-called execution and consensus layer [14]. Hence, *the observed issues are pervasive in the DCS and prone to continuously increase due to the service-centric trend.*

Finally, even though we did not conclude on the centralization of the system, we still describe how the system resilience depends on only 0.26% of all infrastructures. The service instance control may be decentralized, but its concentration represents a hazard for network resilience [26]. Moreover, the network's resilience is affected by the concentration of service instances and a high percentage of pervasive unreachability issues of service relations.

V. Recommendations

Our observations in the previous sections lead us to specific recommendations, discussed in the following, to adjust current DCS systems.

A. *Service Dispatcher*

To recall, the main issue on reachability is caused by the handling of all incoming connections at one of the many peers that are launched at a single IP address, thus causing the other peers at this address to become 'invisible' to the incoming connection, ultimately failing the handshake process.

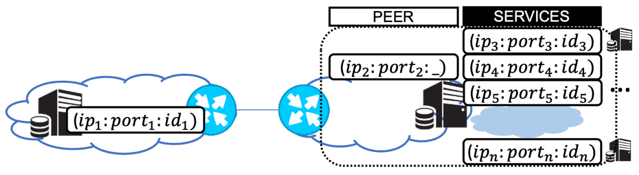

Fig. 10. Service Dispatcher

To tackle this issue, we recommend introducing a *service dispatcher* module to the current Ethereum stack at the P2P network level for handling incoming connection requests (in the distributed pool maintenance process) to forward the requests to the suitable local peer instance. Fig. 10 illustrates this new module, where the *peer context* is the tuple of IP address and port exposed to remote peers, while the *services contexts* represent locally deployed peer instances. Here, the *IP address* is likely situated behind a network address translator, while the *id* entry is being used to differentiate the various locally deployed peer instances and thus expose them correctly to the remote peers attempting to contact them. Based on our insights in Section III, we observe that up to 1800, service instances may run atop a single peer, thus limiting the required context information to less than 100kBytes.

With the service dispatcher being a logical module, it can be deployed solely in so-called *reachability points*, configured to deal with the expected load of incoming requests, while the peer instances may be deployed in a local sub-network (e.g., composed of several virtual machines in a cloud provider network). As shown in our insights of Section IV, adopting our recommendation in as few as ten infrastructures may reduce reachability issues in at least 86% of the services in the system.

Our recommendation here aligns with similar developments in other Internet-based systems, most notably web servers, where similar dispatcher modules handle site-specific network mechanisms like DNS, triangulation, HTTP redirection, and URL rewriting [5], dispatching the individual web requests to logical web servers with ≈ 4500 concurrent connections per second dispatching for ≈ 500 clients [23]. The latter shows that our above-expected load of around 1800 concurrent peer instances should be easily supported.

B. Location Guided Peer Selection

The concentration evidence in Section II sheds light on the *bias towards few ISPs* when executing a random-based service selection. Works in [3], [15], [30] recognized the same, albeit for different biases. Salting the service ID generation or incorporating hierarchies and vicinity path lookups are approaches to bypassing such infrastructure bias in these related works. Hence, our recommendation aligns with that of utilizing salt in the process of the (Ethereum) service ID generation. This salted approach would allow taking into account the infrastructure concentration of services, although we recognize that further work is required to understand the whole nature of the used salt and to avoid biases towards small ISPs (infrastructures with few service instances).

VI. Conclusion

We described the growth, quantification, and orthogonal effects of the trend towards a service-centric model in existing Ethereum. Specifically, we quantified, based on emperical observations, the degree of reachability issues caused through this trend and the reasons behind those issues, chiefly the traffic and infrastructure consolidation through utilizing the hosting in cloud provider infrastructures. Our insights led to two concrete recommendations to reduce the observe non-reachability of deployed peers, while also tackling the impact of centralization in few cloud infrastructures on the convergence process. In future work, we envision quantifying the practical costs, implications, and benefits of implementing our recommendations at scale.

References

[1] Amazon, "Amazon Compute Service Level Agreement," May 2022. [Online]. Available: https://aws.amazon.com/compute/sla/

[2] Binance, "Bitcoin User Data Stolen By Entity Using 812 Different IP Addresses," March 2023. [Online]. Available: https://www.binance.com/en-NG/feed/post/363882

[3] R. Bless, M. Zitterbart, Z. Despotovic, and A. Hecker, "Kira: Distributed scalable id-based routing with fast forwarding," in *21st IFIP Networking Conference, 13th-16th June 2022, Catania*. Institute of Electrical and Electronics Engineers (IEEE), 2022, p. 1–9.

[4] I. Brand Media, "Online Tools," March 2023. [Online]. Available: https://tools.iplocation.net/

[5] V. Cardellini, E. Casalicchio, M. Colajanni, and P. S. Yu, "The state of the art in locally distributed web-server systems," *ACM Comput. Surv.*, vol. 34, no. 2, p. 263–311, jun 2002. [Online]. Available: https://doi.org/10.1145/508352.508355

[6] B. Cohen and K. Pietrzak, "The Chia Network Blockchain," vol. 1, pp. 1–44, 2019.

[7] cointelegraph, "Hetzner anti-crypto policies: A wake-up call for Ethereum's future," August 2022. [Online]. Available: https://cointelegraph.com/news/hetzner-anti-crypto-policies-a-wake-up-call-for-ethereum-s-future

[8] M. Cortes-Goicoechea, T. Mohandas-Daryanani, J. L. Munoz-Tapia, and L. Bautista-Gomez, "Autopsy of Ethereum's Post-Merge Reward System," *2023 IEEE International Conference on Blockchain and Cryptocurrency, ICBC 2023*, pp. 1–9, 2023.

[9] N. Dimitri, "Proof-of-stake in algorand," *Distrib. Ledger Technol.*, vol. 1, no. 2, dec 2022.

[10] Erigon, "Erigon official client," 2024. [Online]. Available: https://github.com/ledgerwatch/erigon

[11] Ethereum, "Open ethereum official client," 2022. [Online]. Available: https://github.com/openethereum/openethereum

[12] ——, "Go ethereum official client," 2023. [Online]. Available: https://github.com/ethereum/

[13] ——, "Node architecture," 2023. [Online]. Available: https://ethereum.org/en/developers/docs/nodes-and-clients/node-architecture/

[14] ——, "Client diversity," 2024. [Online]. Available: https://ethereum.org/en/developers/docs/nodes-and-clients/client-diversity/

[15] P. Ganesan, K. Gummadi, and H. Garcia-Molina, "Canon in g major: designing dhts with hierarchical structure," in *24th International Conference on Distributed Computing Systems, 2004. Proceedings.*, 2004, pp. 263–272.

[16] Y. Gao, J. Shi, X. Wang, Q. Tan, C. Zhao, and Z. Yin, "Topology Measurement and Analysis on Ethereum P2P Network," *2019 IEEE Symposium on Computers and Communications (ISCC)*, pp. 1–7, 2019.

[17] M. Grundmann, M. Baumstark, and H. Hartenstein, "On the Peer Degree Distribution of the Bitcoin P2P Network," *IEEE International Conference on Blockchain and Cryptocurrency, ICBC 2022*, no. July 2021, 2022.

[18] D. Guzman, D. Trossen, M. McBride, and X. Fan, "Insights on impact of distributed ledgers on provider networks," in *Blockchain – ICBC 2022*, S. Chen, R. K. Shyamasundar, and L.-J. Zhang, Eds. Cham: Springer Nature Switzerland, 2022, pp. 3–17.

[19] Hetzner, "Terms and Conditions Version 2.0.0," October 2021. [Online]. Available: https://www.hetzner.com/assets/Uploads/downloads/AGB-en.pdf

[20] ——, "is it allowed to run a crypto block chain node?" August 2022. [Online]. Available: https://www.reddit.com/r/hetzner/comments/wucxs4/comment/ilfoj8u/

[21] S. K. Kim, Z. Ma, S. Murali, J. Mason, A. Miller, and M. Bailey, "Measuring ethereum network peers," in *Proceedings of the Internet Measurement Conference 2018*, ser. IMC '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 91–104.

[22] P. Labs, “Prysm,” 2022. [Online]. Available: https://prysmaticlabs.com/

[23] Q. Li and B. Moon, “Distributed cooperative apache web server,” in *Proceedings of the 10th International Conference on World Wide Web*, ser. WWW ’01. New York, NY, USA: Association for Computing Machinery, 2001, p. 555–564. [Online]. Available: https://doi.org/10.1145/371920.372158

[24] M. Lokhava, G. Losa, D. Mazières, G. Hoare, N. Barry, E. Gafni, J. Jove, R. Malinowsky, and J. McCaleb, “Fast and secure global payments with stellar,” in *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, ser. SOSP ’19. New York, NY, USA: Association for Computing Machinery, 2019, p. 80–96. [Online]. Available: https://doi.org/10.1145/3341301.3359636

[25] P. Maymounkov and D. Mazières, “Kademlia: A Peer-to-peer Information System Based on the XOR Metric,” in *Peer-to-Peer Systems*, 2002.

[26] M. McFadden, “A Taxonomy of Internet Consolidation,” Internet Engineering Task Force, Internet-Draft draft-mcfadden-consolidation-taxonomy-03, Oct. 2023, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-mcfadden-consolidation-taxonomy/03/

[27] A. Mühle, A. Grüner, and C. Meinel, “Characterising proxy usage in the bitcoin peer-to-peer network,” in *Proceedings of the 22nd International Conference on Distributed Computing and Networking*, ser. ICDCN ’21. New York, NY, USA: Association for Computing Machinery, 2021, p. 176–185. [Online]. Available: https://doi.org/10.1145/3427796.3427840

[28] S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” *Journal for General Philosophy of Science*, vol. 39, no. 1, pp. 53–67, 2008.

[29] Nethermind, “Nethermind,” 2024. [Online]. Available: https://www.nethermind.io/

[30] Y. Thomas, N. Fotiou, I. Pittaras, G. Xylomenos, S. Voulgaris, and G. C. Polyzos, “Peer clustering for the interplanetary file system,” in *Proceedings of the 2nd ACM SIGCOMM Workshop on Future of Internet Routing & Addressing*, ser. FIRA ’23. New York, NY, USA: Association for Computing Machinery, 2023, p. 8–14. [Online]. Available: https://doi.org/10.1145/3607504.3609289

[31] D. Trossen, D. Guzman, M. McBride, and X. Fan, “Impact of Distributed Ledgers on Provider Networks,” no. 935, 2021.

[32] V. Tumas, S. Rivera, D. Magoni, and R. State, “Topology analysis of the xrp ledger,” in *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing*, ser. SAC ’23. New York, NY, USA: Association for Computing Machinery, 2023, p. 1277–1284. [Online]. Available: https://doi.org/10.1145/3555776.3577611

[33] G. Wood, “Ethereum: a secure decentralised generalised transaction ledger,” *Ethereum Project Yellow Paper*, pp. 1–32, 2014.

[34] M. Zhou, L. Zeng, Y. Han, P. Li, F. Long, D. Zhou, I. Beschastnikh, and M. Wu, “Mercury: Fast Transaction Broadcast in High Performance Blockchain Systems,” *Proceedings - IEEE INFOCOM*, vol. 2023-May, pp. 1–10, 2023.

Appendix A

The tuning mechanisms such as short-timing the pool replenishment leads to high percentage of traffic in the consolidated infrastructures. We describe the resulting traffic for a tuned discovery and connection establishment procedure.

A. Pool Discovery

In comparison to Fig.4 in Section III, we notice in the y-axis in Fig. 11 that this is not in direct correspondence. The top infrastructure requesting to discover the network (dGmH) differs from the leading infrastructure hosting most peers (HOGH). This difference is due to different configurations for pool replenishment and a banning event reported in [7], [20]. As outlined before, short-timing the execution of discovery for a pool replenishment is the cause of this behavior.

B. Pool Establishment

We perform the identical comparison Fig. 4 in Section III against Fig. 12 for connection establishments. For example, HOGH is the top hosting infrastructure. Still, CVPN is the one establishing the most connections, as shown in Fig. 12, achieved by faster pool replenishment. This behavior for this infrastructure is also reported in [2] for Bitcoin.

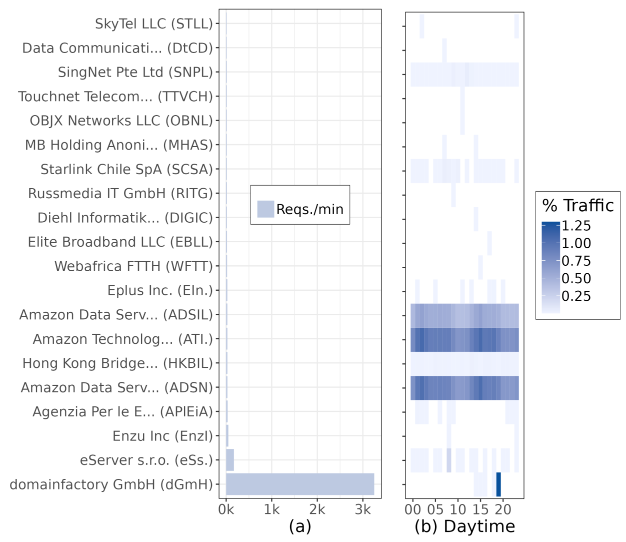

Fig. 11. (a) Density of Traffic arranged by Top 20 Infrastructures generating Traffic while Discovering the Network (b) % of Traffic along the Daytime relative to CET Time

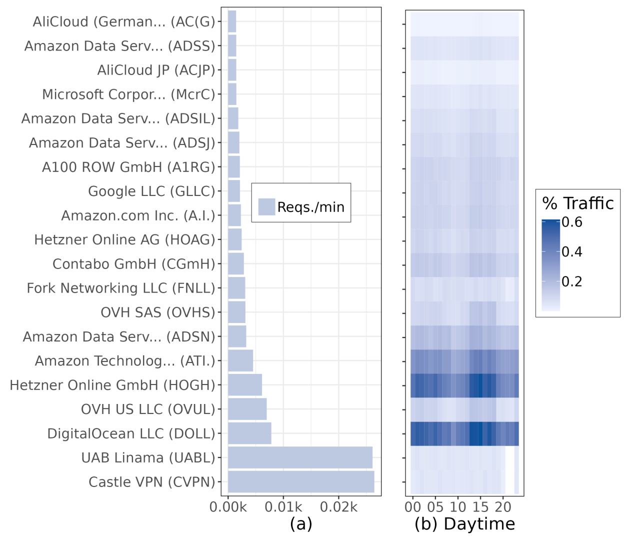

Fig. 12. a) Density of Traffic arranged by Top 20 Infrastructures generating Traffic while Establishing Connections (b) % of Traffic along the Daytime relative to CET Time