IEEE ICBC'24 1570981790

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

***Abstract*—In Internet-scale distributed consensus systems (DCSs), a single peer exposes functionality to other peers to agree on a single shared state within a computational problem; this is the de facto model that finds realizations in cryptocurrencies and distributed file systems. We call it the *peer-centric model*. The distributed nature of the consensus in these DCSs permits a peer to partake by exposing functionality in exchange for fees and rewards. Indeed, if one considers a peer exposing functionality as providing a service, in that case, those benefits motivate a peer to instantiate a service multiple times. We refer to this trend as the *service-centric model* and outline its raison d'etre and growth based on empirical observations for the Ethereum system. We quantify the observations of this trend to report its impact; while doing so, we highlight an increased availability and concentration of peers. In particular, we notice a high percentage of non-reachable service instances running in highly available peers. Specifically, we uncover a security context mismatch, part of the unreachability issues, rooted in the peer-centric design.**

I. Introduction

DCSs aim to achieve consensus over a shared state in a continuous computation. Indeed, DCSs seek to maintain a distributed, coherent state among the entire system or its majority [26] to permit the progress of a computation.

Among others, Bitcoin [25], Ethereum [30], Stellar [21], Chia [6], and Algorand [9] explored a permissionless or open membership character of distributed consensus achievement and participation. Peers openly sign up for these permissionless DCSs. These peers then participate by maintaining the consensus with less stringent requirements.

Participation in the system is motivated by benefits from fees and rewards. Fee collection and system rewards are the main streams of benefit. Peers collect fees for committing and prioritizing changes to the shared state [8]. Besides fees, the DCS rewards peers for maintaining a coherent state [25].

In Ethereum, for instance, peers collect fees for committing financial transactions with a given priority. Here, a transaction updates the distributed shared state, i.e., the distributed ledger. In addition, peers earn rewards for maintaining this distributed ledger, i.e., for actively participating in the DCS [25].

Peers download and launch applications to partake in a DCS. Those applications realize the functionality required to receive and diffuse a shared state.

An Ethereum node, implemented by the software geth [10], [11], provides functionalities through an application, as illustrated in Fig. 1-a. This application actively probes for peers to build and replenish a collection of successful connections, as outlined in [14]. 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.

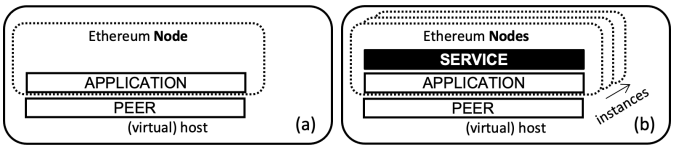

Fig. 1. (a) Peer Centric Model and (b) its Evolution

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 and partake in the DCS with several running service instances, as in Fig. 1-b. This is because additional instances increase the likelihood of earning rewards and seizing more transaction requests [8]. Indeed, this tendency, in Fig. 1-b, denotes the evolution of the peer-centric. This paper explores this shift of DCSs from a peer to a service-centric model and how this affects the systems we run today.

We summarize our main contributions as follows: (i) observing and quantifying the service-centric trend and (ii) assessing its impact. While our work outlines traffic and infrastructure concentration, we avoid the ties made towards the centralization of DCSs, in contrast to [2]. Instead, we highlight the high availability, also boosted by cloud providers, of most of the peers in the system to discuss our observations. Indeed, we provide evidence that *despite highly available peers in the DCS, reachability still is an issue*.

More specifically, in Section II, we show the empirical observations for this trend in the now-deployed Ethereum system. This trend's impact on the current deployment follows in Section III. Then, we draw recommendations in Section IV over the observations before concluding in Section V.

II. Empirical 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 running 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 [25], [30].

Peers realize particular **mechanisms** to expand their participation in the DCS. These mechanisms include configuring aggressive fanouts, short-timing pool replenishments [31], 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 [14], [31].

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 [13], [24] for Bitcoin, in [12] and [19] as node degree for Ethereum, and in [29] 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.

Peers setting shorter timeouts for re-establishing connections to endpoints realize the pool replenishment. For example, the default timeout for establishing a new connection in Ethereum is fifteen seconds [10].

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

In Ethereum, application replication on a per-peer basis realizes this service instance exposure. This mode for consensus participation and state diffusion is realized following 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**.

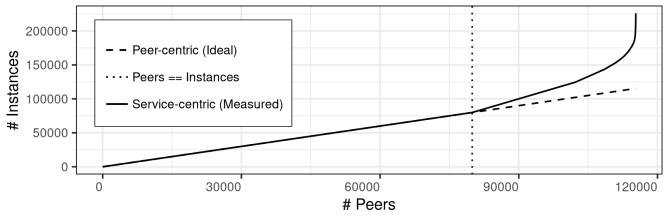

Fig. 2. The Service-centric Trend

The empirical evidence in Fig. 2 highlights the degree of the current divergence between service instances and peers in Ethereum. This divergence from a total of $\approx 115k$ different peers identified by IP addresses.

The dataset shared by the authors in [14], [28] logs discovery and connect communication towards peers in campaigns from 01-2022 to 06-2022, and from 02-2023 to 06-2023, as detailed in [14]. The campaigns performed passive measurements with a probe crawling the Ethereum network and synchronizing its entire blockchain with a default configuration of *geth* [10].

In these campaigns, 86% peers actively provided a total of $\approx 185.5k$ service instances. This discrepancy allows us to elaborate on the starting gap at $79.9k$ peers, meaning that 69% of the total peers find a one-to-one correspondence between an IP address and a service instance ID. This gap grows non-linearly to an aggressive number of service instances in single points of reachability with a maximum degree of a peer exposing $\approx 1.7k, 1\%$ of the service instances.

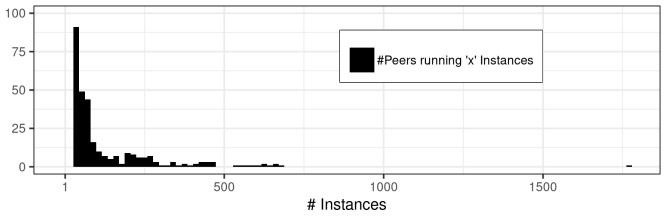

Fig. 3. The Growth in the Service-centric Trend

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.

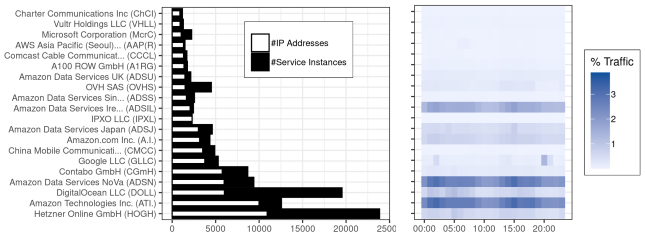

Fig. 4. (a) Infrastructures hosting Peers and Instances, and (b) its Traffic Concentration on a Day Time relative to CET Time sharing the same y-axis

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 [29] for XRP Ledger and in [13] for Bitcoin. Fig. 4-a presents this degree of accumulation based on an IP address to autonomous system (AS) translation [5]. 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. 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%. Third, service instances generate traffic throughout the day, as shown in Fig. 4-b along the x-axis.

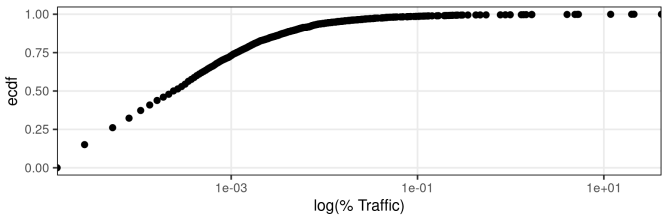

Fig. 5. ECDF for the Percentage of Traffic Generated by an Infrastructure

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.

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% [16] and 99.99% [1] respectively. But, *even though highly available infrastructures support DCS instances, there are still issues with service reachability.*

III. Impact on the Ethereum Network

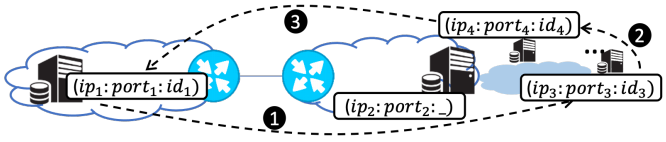

Fig. 6. Incoming and Outgoing Service Relation Establishment

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

Discovery and pool establishment are part of a service relation building, as described in [14, Section 2.2] and [19, 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 [22]. 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 [14], [19]. These IN relations are the majority and shape the impact of reachability in the DCS [19].

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 compromise 86% of the system.

Failures in infrastructures such as HOGH and ATI seem unlikely since their average network availability is 99.9% [16] and 99, 99% [1], respectively. However, we outline how the service unreachability, accounting for 87% of communications, impacts these highly available infrastructures.

87% of the service establishment trials in the system show unreachability issues, also reported as 90% in [14], and 93.98% in [19]. Out of the 87% of reachability issues, ≈ 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 [19]). The security handshake errors relate to the service security context establishment (disconnects in [19]).

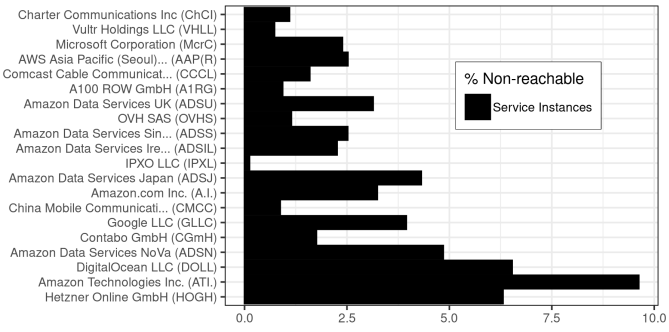

Fig. 7. Reachability Issues Spread along the DCS

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. Indeed, a service instance encounters disconnects proportionally to the degree of infrastructure hosting.

This proportion holds approximately for larger hosting degree infrastructures, as shown in Fig. 8, meaning that the more services are concentrated in an infrastructure, the more issues a peer finds. In Fig. 8, this correlation is biased due to the tuning mechanisms, e.g., short-timed replenishment, outlined before and detailed in Appendix A.

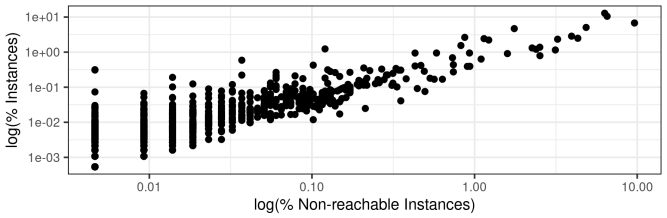

Fig. 8. Correlation for % Non-reachable against % Instances per Infrastructure

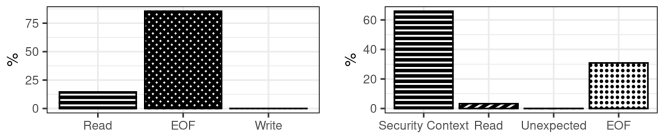

Fig. 9. (a) Outgoing and (b) Incoming Service Reachability Issues

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, as in ❶ in Fig. 6. But, if the relation is established through, e.g., point of reachability $(ip_2 : port_2 : \_)$ in Fig. 6, the response to this request is performed by the latest used (virtual) instance. The latter results in unsolicited responses leading to an unexpected size of message that is reported as EOF in Fig. 9-a. Moreover, TCP resets are reported as read and write errors.

IN relations handshake issues include 3% read due to TCP resets, 30% EOF errors due to unavailable packets to read [10], and 65.9% a **security context mismatch**, as in Fig. 9-b.

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) was 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 than id_1, hence the tuple for the remote endpoint is (ip_4, id_3) causing a message decryption problem that results in the high percentage of the service security context mismatch.

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 thread for network resilience [23]. Moreover, the network's resilience is affected by the concentration of service instances and by an inherent well-spread high percentage of unreachability, 87% of service relations. This unreachability is rooted in the initial randomized communication design, e.g., oversubscribed pools, and the peer-centric model, e.g., security context mismatch. The latter is due to the continuous evolution of how peers expose functionality to the network, the service-centric trend.

IV. Recommendations

The observations on the evolution of the peer-centric model in Section II, where we described a non-linear divergence between the service instances and peers, let us introduce specific reachability issues. These service reachability issues, mainly composed of pool oversubscription and handshake errors, guide us onto recommendations over the root **mechanisms** behind these issues, namely methods to boost peer participation in the DCS.

An aggressive pool fanout may be reduced and restricted, and the interval time for pool replenishment may be increased to decrease the amount of disconnections caused by oversubscribed peers that deny further communication intents. This approach represents a tradeoff between the will to reduce shared state diffusion and consensus convergence latency towards the degree of the fanout and the replenishment timing. The diffusion and convergence latency, however, can be considered as one for a small concentrated network, as shown by our insights, and not for an Internet-scale network. The concentration consideration may further improve the consensus convergence latency definition, e.g., defining a reduced time slotting for staking [30] or minimized time slotting for mining blocks [25].

Moreover, to tackle the handshake issues rooted in the peer-centric design, *service dispatchers* may be deployed in the concentrated points of reachability to ensure a proper security context establishment. This approach, again shown by our insights, will require the involvement of as few as ten stakeholders to handle and solve reachability issues in at least 86% of the entire running system. Indeed, avoiding further tussles and issues [2] introduced by approaches such as peer proxies [24], concentrators [20], peer validators [9], and application level routers [27].

As pointed out in Section II, peers running on well-known infrastructures show a DCS traffic and infrastructure concentration produced by less than 1% of the infrastructures. This concentration behavior shapes the concensus convergence and diffusion delay.

For the DCS as a whole, this evolution represents a tradeoff: due to the traffic and infrastructure concentration around infrastructure providers, the DCS achieves higher availability, lower state diffusion, and consensus convergence latency; however, it gives up **decentralization**, i.e., its initial design premise when considering the control of the underlay infrastructure, service reachability, and therefore also network **resilience**. Considering the high concentration we observe, failures for individual infrastructure providers can result in significant outages of services instances, ranging from a maximum of 21% to 4% of the system that heavily impacts the state diffusion and consensus convergence latency, and most interestingly the time of *continuous work on inconsistent state* [15], e.g., unsolved forks in chains of blocks [18].

Lastly, also due to infrastructure concentration we question the original design of the randomized diffusion approach. This since the initial considerations for designing it [22] were the

high churn and lower reliability in old systems, assertion that our observations contradict. Hence, we recommend exploring further approaches such as the ones in [4], [15], [18], since in current systems, there is negligible churn and high availability of participants of the distributed consensus provided by the ease of cloud infrastructures.

More specifically, given that the binary distance between service instance IDs is topologically independent, consolidation drive geographical bias towards those strong *hosting infrastructures* [14]. Hence, we recommend a location guided selection of services to prevent collusion and further enhance the state diffusion delay.

V. Conclusion

The evolution and impact of the peer-centric model is inevitable. Hence, we described the growth, quantification, and orthogonal effects of the service-centric model for the currently deployed Ethereum system. Indeed, we shed light on the impact and issues of the service-centric model based on extensive empirical observations. Moreover, we quantified the degree of spread of the high amount of reachability issues and described the reasons for those issues. In particular, we detail the amount of errors while establishing a mandatory security context handshake.

These analyses allowed us to consider the traffic and infrastructure consolidation degree to understand the current system resilience. Also, we notice a few points regarding decentralization premises and ease of deployment for unreachability solutions. The latter is part of the recommended enhancements to tackle the design of specific mechanisms, such as aggressive fanouts and service instance exposure, to improve diffusion and consensus convergence latency.

In future work, we envision quantifying the practical costs, implications, and benefits of implementing specific recommendations at scale on fanout size, pool replenishment, service instantiation, and service selection guided by location and fairness.

References

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

[2] L. Balduf, M. Korczyński, O. Ascigil, N. V. Keizer, G. Pavlou, B. Scheuermann, and M. Król, "The cloud strikes back: Investigating the decentralization of ipfs," in *Proceedings of the 2023 ACM on Internet Measurement Conference*, ser. IMC '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 391405. [Online]. Available: https://doi.org/10.1145/3618257.3624797

[3] 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

[4] 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. 19.

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

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

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

[12] 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.

[13] 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.

[14] 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.

[15] D. Guzman, D. Trossen, and J. Ott, "If iterative diffusion is the answer, what was the question?" 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. 2934. [Online]. Available: https://doi.org/10.1145/3607504.3609288

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

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

[18] M. Jin, X. Chen, and S.-J. Lin, "Reducing the bandwidth of block propagation in bitcoin network with erasure coding," *IEEE Access*, vol. 7, pp. 175 606–175 613, 2019.

[19] 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. 91104.

[20] X. T. Lee, K. Arijit, S. G. Sourav, H. O. Yu, and L. Xuan, "Measurements , Analyses , and Insights on the Entire Ethereum Blockchain Network," in *WWW '20: Proceedings of The Web Conference 2020*, 2020.

[21] 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. 8096. [Online]. Available: https://doi.org/10.1145/3341301.3359636

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

[23] 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/

[24] 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. 176185. [Online]. Available: https://doi.org/10.1145/3427796.3427840

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

[26] J. V. Newman, "Probabilistic Logics and the Synthesis of Reliable Organism from Unreliable Components," *Automata Studies*, vol. C. Shannon, 1956.

[27] D. Trautwein, A. Raman, G. Tyson, I. Castro, W. Scott, M. Schubotz, B. Gipp, and Y. Psaras, "Design and evaluation of ipfs: A storage layer for the decentralizedweb," *SIGCOMM 2022 - Proceedings of the ACM SIGCOMM 2022 Conference*, pp. 739–752, 2022.

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

[29] 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. 12771284. [Online]. Available: https://doi.org/10.1145/3555776.3577611

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

[31] 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

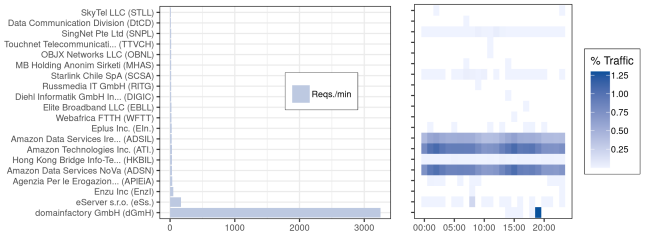

Fig. 10. (a) Density of Traffic arranged by Top 20 Infrastructures generating Traffic while Discovering the Network (b) % of Traffic along the Daytime relative to CET Time

In comparison to Fig.4 in Section II, we notice in the y-axis in Fig. 10 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 the different configuration schemes of pool replenishment and a banning event reported in [7], [17]. As outlined before, short-timing the execution of discovery for a pool replenishment is the cause of this behavior.

B. Pool Establishment

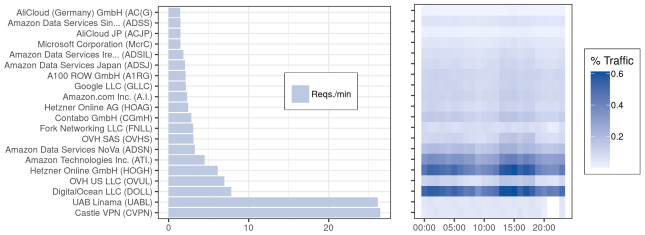

Fig. 11. a) Density of Traffic arranged by Top 20 Infrastructures generating Traffic while Establishing Connections (b) % of Traffic along the Daytime relative to CET Time

We perform the same comparison, Fig. 4 in Section II against Fig. 11 for connection establishments. For example, HOGH is the top hosting infrastructure. Still, CVPN is the one establishing the most connections, as shown in Fig. 11, achieved by faster pool replenishment. This behavior for this infrastructure is also reported in [3] for Bitcoin.