Decentralised DNS Resolution Based on Distributed Ledger Technology

George Giamouridis, BooJoong Kang and Leonardo Aniello

School of Electronics and Computer Science, University of Southampton, UK

Abstract

The Domain Name System (DNS) is the backbone of the Internet, but remains highly centralised, creating risks related to censorship, surveillance, and single points of failure. While much of the focus has been on securing and decentralising DNS as a whole, recursive resolvers continue to rely on central infrastructure. This paper proposes a decentralised DNS resolver architecture that replaces traditional recursive resolvers with a decentralised alternative. The system uses the Chord protocol and distributed ledger technology (DLT) to distribute DNS resolution across a network of cooperating nodes, without altering existing zone authority structures. We present the key components of the architecture, including peer discovery, query routing, caching, updating and trust handling, and we analyse how this architecture overcomes the previously mentioned limitations.

Keywords

Blockchain, Domain Name System, Decentralisation

1. Introduction

The DNS is a distributed, hierarchical system responsible for mapping domain names to IP addresses on the Internet. Despite its distributed design, much of its infrastructure relies on centralised components, especially recursive resolvers. These resolvers handle most DNS traffic by acting as intermediaries between clients and authoritative servers. NS1's 2023 Global DNS Traffic Report [1] shows public resolvers handle nearly 60% of recursive DNS traffic, with Google at over 30% and AWS at 16%. ISP resolvers, such as those run by Xfinity [2] and T-Mobile [3], account for 9%, but they're regionally concentrated and subject to similar central controls. The rest includes private, enterprise, and smaller caching resolvers. Despite DNS's design, query resolution remains largely centralised.

This centralisation introduces critical vulnerabilities, including single points of failure, exposure of user queries to potential surveillance, and censorship through manipulated resolver responses [4] [5]. Notably, the 2016 Dyn DDoS attack [6] disrupted major websites, like Twitter, Spotify, and Reddit, showing how resolver-level failures can impact large parts of the Internet. Centralised resolvers also enable surveillance, as seen in the NSA's PRISM program [7], which intercepted DNS queries without user consent. Additionally, censorship practices like China's Great Firewall [8] systematically alters DNS responses at the resolver level to block access to

g.giamouridis@soton.ac.uk (G. Giamouridis); b.kang@soton.ac.uk (B. Kang); l.aniello@soton.ac.uk (L. Aniello)

DLT2025: 7th Distributed Ledger Technology Workshop, June, 12-14 2025 - Pizzo, Italy

specific websites. These cases highlight the security, privacy, and censorship risks of centralised DNS, underscoring the need for an alternative resolution system.

To address these issues, this paper proposes a decentralised DNS resolver architecture. Unlike traditional DNS resolvers that rely on centralised authorities, the proposed architecture uses a Chord DHT and distributed ledger technology (DLT) to decentralise both DNS query resolution and record updates across a network of nodes. The DHT assigns DNS records to nodes, and it handles lookup, routing, and replication, while a permissioned blockchain-based DLT ensures consistency during updates, even if some nodes are malicious. The primary objectives of this architecture are to improve availability, reduce reliance on trusted third parties, strengthen resistance to censorship and tampering and address security threats related to the DNS resolver itself. Threats originating from or targeting the DNS Root, TLD, and Authoritative Name Servers are considered out of scope. This paper outlines the system's architecture and the mechanisms enabling secure cooperation between nodes.

The rest of the paper is structured in four sections. The background section provides the foundational context, followed by related work that highlights existing solutions. The architecture section provides a high-level overview of the proposed system, and describes the design principles and processes. Finally, the discussion section reflects on the implications of the traditional DNS architecture and explains how the proposed design overcomes those limitations.

2. Background

2.1. The Domain Name System (DNS)

DNS operates on a client-server model with a hierarchical and distributed protocol [9]. When a stub resolver (integrated into the end user's device) requests a domain name, it first contacts a recursive DNS resolver as shown in Figure 1. If the resolver has the requested information cached, it returns the IP address directly.

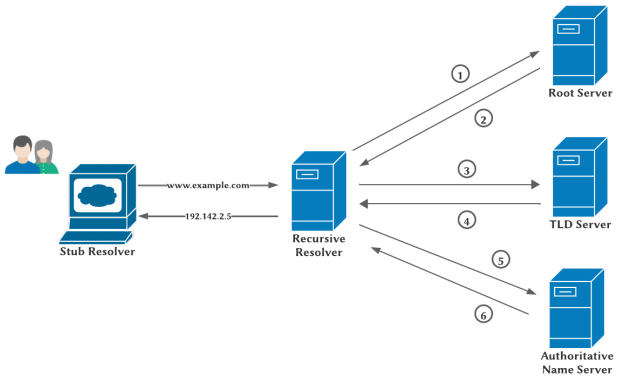

Figure 1: The components of the DNS protocol and the lifecycle of a request.

If the information is not in the cache or has expired, the resolver queries root DNS servers [10] to identify the authoritative DNS server for the top-level domain (TLD) (e.g., ".com", ".org").

It then queries the TLD server [11] for the authoritative name server (ANS) [12] of the domain. Once the resolver obtains the IP address of the ANS, it queries the authoritative server for the domain's IP address. After receiving the IP address, the resolver caches it for future use [13], and the result is returned to the user's device, allowing it to connect to the Web server.

2.2. The Chord Protocol

Chord is a structured peer-to-peer (P2P) protocol for scalable, decentralized key-based lookup in distributed systems [14]. It uses consistent hashing to assign data keys and node identifiers to a circular space of size 2^b, where b is the number of bits in the hash function (e.g., SHA-1). Each key is stored at its successor node, the first node whose ID is equal to or greater than the key. For efficient routing, each node maintains a finger table [15] with up to b entries. The i-th entry points to the node responsible for key $(x + 2^{i-1}) \bmod 2^b$, where x is the node's ID. This design enables $O(\log P)$ lookup time, with P being the number of nodes (peers). Chord includes stabilisation mechanisms to handle node joins and leaves, ensuring consistent key assignment and correct routing.

A DHT is a decentralized storage system that maps keys to values and distributes this mapping across a network of nodes without requiring central coordination. In Chord, the DHT is implemented by storing key-value pairs at the node responsible for each key, as determined by the consistent hashing scheme. This enables efficient insertion, lookup, and replication of data in a fully decentralized manner.

3. Related Work

Several studies have explored decentralised DNS approaches and their potential benefits and limitations. Blockchain-based DNS (BDNS) is one of the most popular ones, with Namecoin [16] being the first attempt to decentralise DNS and provide a blockchain alternative to traditional systems. Giamouridis et al. [17] provide a comprehensive review of existing BDNS systems and evaluates their potential to transform the domain name landscape.

Liu et al. [18] propose a transitional architecture that integrates legacy DNS with a peer-to-peer network, improving data availability and query efficiency while retaining DNS elements. Danielis and Altmann [19] introduce P-DONAS, a DHT-based P2P DNS for ISP access nodes that reduces DNS traffic load and operational cost while maintaining compatibility with traditional DNS. Sancho and Pereira [20] present an open, censorship-resistant DNS using a P2P network and distributed, user-verifiable zone files, enabling users to bypass censorship with selective trust. Cox and Muthitacharoen [21] explore using DHash over Chord for DNS record storage, aiming to decouple domain ownership from hosting responsibilities and simplify administration. Song and Koyanagi [22] propose a hybrid DNS model combining DHT robustness with the efficiency of traditional DNS to mitigate latency issues in purely P2P-based systems. The work by Berners-Lee et al [23] discusses OpenNIC in the context of alternative Internet governance models, discusses OpenNIC in the context of alternative Internet governance models, focusing on its role as a decentralised DNS solution.

Contrary to existing decentralised DNS solutions, that focus on modifying the traditional DNS infrastructure, this paper proposes a distinct approach by introducing an alternative architecture at the recursive resolver level. While prior systems aim to replace the root and the authoritative name servers, this architecture shifts the focus to replacing the recursive resolver with a decentralised network. By distributing resolver functionality through this network in combination with consensus-based updates, the system improves availability, censorship resistance, and trust without interfering with existing DNS hierarchies.

4. Constructing the Resolver: A High-Level Overview

Our decentralised DNS resolver is designed to address the limitations of traditional centralised DNS resolution systems. The goal is to eliminate reliance on centralised recursive resolvers, which introduce privacy risks, single points of failure, and potential censorship, by distributing the resolution process across independent nodes.

As shown in Figure 2, in our model each node $p \in P$ is placed on a Chord ring and stores a subset $C_i \subset F$ called a *chunk*, where F is the initial global dataset of DNS records. The peers responsible for handling the same chunk C_i form a permissioned blockchain [24], which is used solely for updating the chunk. The service is only accessible to nodes that have joined the network as peers. When a DNS request is initiated, the node uses the Chord DHT to locate the node responsible for the relevant chunk and extracts the full domain record. This resolution process operates in a decentralised manner, with the DHT providing an up-to-date list of peers with their corresponding chunks.

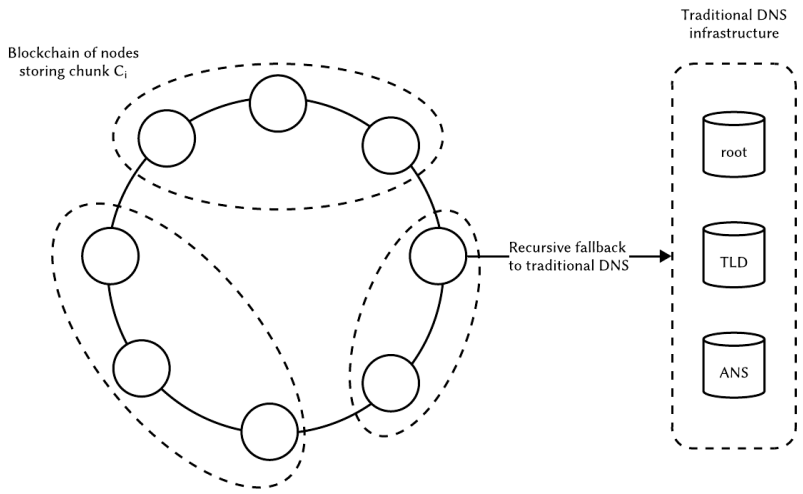

Figure 2: The structure and components of the decentralised resolver.

DNS records are time sensitive, meaning that entries expire based on time-to-live (TTL) values, and new records are frequently added or updated. As a result, many queries may fail if the necessary data is unavailable locally or in the relevant peer set. In this situation, when a node cannot resolve a query from local or peer stored data, it initiates the resolution process from the root servers, traversing the DNS hierarchy as needed (root $\rightarrow$ TLD $\rightarrow$ authoritative servers). This approach removes the dependency on any external recursive resolver, retains decentralisation, and guarantees completeness in query resolution.

4.1. Storage Model and Node Initialisation

The global dataset of DNS records is partitioned into N chunks and distributed using the Chord-based DHT. Each chunk C_i contains a subset of domain name to IP mappings (DNS records), corresponds to the i-th partition where $i \in \{1, 2, \ldots, N\}$, and is stored by a set of peers. A unique key $k_i = H(C_i)$ is assigned to each chunk by applying a cryptographic hash function H (e.g., SHA-1) to its content. These keys are mapped onto the Chord identifier ring. Each chunk is assigned to the first node in the ring whose identifier is equal to or greater than k_i, known as the successor node. To improve consistency (as explained in Section 4.2), replicas of each chunk are also stored at the next s successors in the ring, where $1 \leq s \ll P$. This assignment ensures uniform distribution of chunks across the network, supports efficient query routing with $O(\log P)$ hops, and maintains resilience to node joins and leaves. The DHT manages both the placement and retrieval of chunks in C, enabling dynamic membership and reliable access to DNS data.

To join the network, a node connects to one or more known bootstrap peers [25], which serve as initial contact points. After verifying the connection and protocol compatibility, the node uses the DHT's routing mechanism to discover existing peers. The new node checks for which key it should take responsibility based on its position in the identifier space and requests the relevant chunk from the current responsible nodes. Replication is updated to reflect the new topology. The new node and its successors now share responsibility for storing the affected chunks, ensuring availability and redundancy.

4.2. Processing DNS Queries

To resolve a domain D, the node applies the function $\mathcal{M}$, which hashes D using a cryptographic hash function H (e.g., SHA-1), producing the key k_i for the corresponding chunk C_i, so $\mathcal{M}(D) = H(D) = k_i$. If the node does not store the relevant C_i, it uses the Chord DHT to locate the set of peers responsible for that chunk (lookup). The DHT enables efficient routing to the correct nodes based on the chunk's key. The node queries these peers and collects their responses. Each response is checked for structural validity, ensuring it contains required DNS fields such as name, type, TTL, and value. Invalid or abnormal responses are discarded. From the valid responses, the node evaluates majority agreement. If a majority return matching records, the result is accepted; otherwise is rejected.

$$\text{Let } R = \{r_1, r_2, \ldots, r_m\} \text{ be the set of replies.}$$

$$\text{If } \exists v \text{ such that } |\{r_k \in R \mid r_k = v\}| > \frac{m}{2}, \text{ then accept } v.$$

Otherwise, reject the response.

If the requested DNS record D is not found within any chunk C_i ($D \notin C_i$) stored in the network, the node initiates a fallback procedure, performing recursive resolution via the standard DNS hierarchy as explained in Section 2.1. Once the record is retrieved, the node completes the DNS request and shares the new record to be added to the network, as explained in Section 4.3.

If the record is not returned by the traditional DNS, the protocol responds with an *NXDOMAIN* message, indicating the non existence of the requested domain. This process ensures that any domain resolution can be traced back through the authoritative DNS chain, providing a fallback mechanism while maintaining the decentralised nature of the system.

4.3. Updating DNS Records

Updating the network involves two main scenarios: updating an existing DNS record due to expiration, and adding a new DNS record that comes from the fallback request to the traditional DNS. Records in the network are updated based on their TTL values, as is the case with traditional DNS resolvers. When a record expires, it is automatically updated by querying the traditional DNS and going through the consensus process as will be explained further below. In both cases, the network must ensure that the records remain consistent, accurate, and available across more nodes (expired or modified records need to be replaced, while new records must be introduced into the network). To achieve consistency and tolerate malicious nodes, a permissioned blockchain is used to manage DNS record updates. A permissioned blockchain is chosen over a permissionless one to avoid transaction fees and reduce overhead (e.g., computational and storage). This blockchain operates at the chunk level, meaning that peers managing the same chunk of DNS records act as a permissioned blockchain to validate and record updates. When serving DNS requests, however, these peers operate as part of the Chord DHT for efficient lookups.

DNS Record Update When a record expires (based on its TTL value), the nodes storing it are responsible for fetching an updated copy from the traditional DNS and updating their local version. When the record R_{old} is updated, the responsible nodes submit a transaction $T_{\text{update}}(R_{\text{old}}, R_{\text{new}})$ to their blockchain. The consensus mechanism of the blockchain ensures that only valid updates are accepted based on rules such as record format and TTL validity. All peers storing the chunk C_i containing R_{old} participate in the consensus process $\mathcal{C}^{\text{cons}}$, which validates the update:

$$\mathcal{C}^{\text{cons}}(T_{\text{update}}) = \begin{cases} \text{valid}, & \text{if consensus is reached} \\ \text{invalid}, & \text{if consensus fails} \end{cases}$$

Once the update is validated, the updated record R_{new} is stored locally by all nodes responsible for chunk C_i.

$$\text{Update}(\text{IP}, R_{\text{new}}(D)) \quad \forall\, \text{IP} \in C_i$$

This process guarantees that all nodes storing the chunk agree on the update, ensuring the integrity and consistency of the record across the network.

Adding a New Record When a requested record is not stored in a chunk, the nodes managing that chunk are responsible for retrieving and storing the record. The nodes perform a fallback to the traditional DNS to retrieve it and insert it into their chunk. The requesting node uses the DHT to determine which chunk the retrieved record belongs to as explained in Section 4.2. If the node stores the relevant chunk, it notifies the other nodes on its blockchain about the new record insertion. All nodes are responsible for falling back to the traditional DNS to retrieve the record. They then submit a transaction $T_{\text{new}}(R_{\text{new}}(D))$ to the blockchain, following the DNS record update process described above. The same consensus process $\mathcal{C}^{\text{cons}}$ is used for validation.

$$\mathcal{C}^{\text{cons}}(T_{\text{new}}) = \begin{cases} \text{valid}, & \text{if consensus is reached} \\ \text{invalid}, & \text{if consensus fails} \end{cases}$$

Once validated, the new record $R_{\text{new}}(D)$ is added to the corresponding chunk and stored by the responsible peers.

$$\text{Add}(\text{IP}, R_{\text{new}}(D)) \quad \forall\, \text{IP} \in C_i$$

If the requested record is in a chunk managed by other nodes, the node uses the DHT to identify them and forwards the request. Those nodes then retrieve the record via fallback to traditional DNS, submit the transaction to the blockchain, and follow the record insertion process.

5. Discussion

In this section, we analyse the strengths of the proposed architecture compared to traditional DNS, highlighting its advantages in terms of decentralisation, security, and transparency. By addressing key weaknesses in the traditional DNS architecture, the system ensures greater resilience against failures, tampering, and censorship, and shows how the proposed design overcomes those limitations. Below, we explore these improvements in detail.

Centralisation and Single Points of Failure As discussed in Section 2, traditional DNS resolvers depend on centralised infrastructure, making them vulnerable to outages, misconfigurations, and targeted attacks. The proposed model distributes DNS records across a P2P network with each chunk replicated across multiple nodes for availability and balance. If a node fails or a partition occurs, the DHT-based routing layer redirects queries to other replicas. This decentralised design removes single points of failure and improves resilience against denial-of-service (DoS) attacks. By replacing static trust in centralised resolvers with verifiable, distributed replication, the system ensures reliable resolution even under adverse conditions.

Censorship and Tampering Traditional DNS queries are vulnerable to censorship and tampering, as shown in real-world cases in Section 1. The proposed resolver mitigates these threats

by decentralising resolution and validating responses from multiple independent peers. A query is only accepted if more than $\frac{m}{2}$ out of r responses agree, making isolated manipulation ineffective and coordinated tampering detectable unless most peers in a chunk are compromised. Since nodes are independently operated without central oversight, large-scale manipulation is significantly harder. Inconsistencies trigger retries or conflict resolution, preserving response integrity.

Security Considerations Traditional DNS is vulnerable to threats like cache poisoning and forged responses due to its reliance on individual resolvers and limited validation. The proposed system counters this by decentralising trust and requiring consensus for every DNS record update. Whether from record changes or fallback to traditional DNS, each update must pass a consensus protocol $\mathcal{C}_i^{\text{cons}}$ among the responsible blockchain peers. An update is only accepted if enough peers validate it; otherwise, it is discarded. This prevents a single compromised node from corrupting data and protects against injection or replay attacks. By requiring agreement before any change, the system secures DNS integrity and reduces attack surfaces.

Lack of Transparency and Control. Traditional DNS lacks transparency, leaving clients unaware of the resolution path or the DNS servers involved, which can lead to vulnerabilities like incorrect or outdated responses due to misconfigurations or compromised servers. Clients also cannot verify server liveness, making it difficult to detect unresponsive or compromised resolvers. In contrast, our model ensures full transparency. Each node independently handles resolution, selects peers, and verifies the data they return using consensus. This reduces the risk of outdated or incorrect responses, as nodes do not rely on potentially compromised third-party resolvers.

6. Conclusion

In this paper, we proposed a decentralised DNS resolution system architecture that addresses the vulnerabilities and limitations of traditional DNS, including centralisation, single points of failure, and lack of transparency. By distributing query resolution and using consensus for updates, the system improves availability, security, and resilience. It defends against cache poisoning and forged responses by validating updates and responses through multiple peers. Nodes maintain full control over resolution, making tampering detectable. The replication mechanism improves fault tolerance and resists censorship. This approach offers a secure, censorship-resistant alternative to traditional resolvers. Future work includes evaluating the system's performance and security through real-world simulations.

7. Acknowledgments

This work was partially supported by the UK Research and Innovation (DTP Scholarship under grant EP/T517859/1); and the Academic Centre of Excellence in Cyber Security Research - University of Southampton (EP/R007268/1).

8. Declaration on Generative AI

During the preparation of this work, the author(s) used ChatGPT and LanguageTool in order to: Grammar and spelling check, Paraphrase and reword. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] NS1, Analysis of 7.5 trillion dns queries reveals public resolvers dominate the internet, 2023. URL: https://circleid.com/posts/20230316-analysis-of-7.5-trillion-dns-queries-reveals-public-resolvers-dominate-the-internet#:~:text=It%20reveals%20that%20public%20resolvers,Amazon%20Web%20Services%20at%2016%25., accessed: 2025-04-17.

[2] Xfinity, Xfinity by comcast, 2023. URL: https://www.xfinity.com, accessed: 2025-04-17.

[3] T-Mobile, T-mobile usa, 2023. URL: https://www.t-mobile.com, accessed: 2025-04-17.

[4] L. Jin, S. Hao, H. Wang, C. Cotton, Understanding the impact of encrypted dns on internet censorship, in: Proceedings of the Web Conference 2021, 2021, pp. 484–495.

[5] S. Wang, K. MacMillan, B. Schaffner, N. Feamster, M. Chetty, Measuring the consolidation of dns and web hosting providers, arXiv preprint arXiv:2110.15345 (2021).

[6] M. Mimoso, Dyn ddos attack: What we know so far, Dark Reading (2016). URL: https://www.darkreading.com/news/dyn-ddos-attack-what-we-know-so-far/d/d-id/1326357.

[7] G. Greenwald, E. MacAskill, Nsa prism program taps in to user data of apple, google and others, The Guardian (2013). URL: https://www.theguardian.com/world/2013/jun/06/us-tech-giants-nsa-data.

[8] S. Zeng, X. Wu, The great firewall of china: A study of dns filtering, International Journal of Computer Applications 96 (2014) 28–35. doi:10.5120/16913-3154.

[9] H. Gao, V. Yegneswaran, Y. Chen, P. Porras, S. Ghosh, J. Jiang, H. Duan, An empirical reexamination of global dns behavior, in: Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM, 2013, pp. 267–278.

[10] D. Conrad, Brief overview of the root server system (2020).

[11] M. Paul, K. J. Dunlap, Development of the domain name system, ACM 18 (1988).

[12] S. Adiwal, B. Rajendran, et al., A quantitative method for measuring health of authoritative name servers, International Journal of Information Security and Privacy (IJISP) 16 (2022) 1–19.

[13] J. Jung, E. Sit, H. Balakrishnan, R. Morris, Dns performance and the effectiveness of caching, in: Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement, 2001, pp. 153–167.

[14] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, H. Balakrishnan, Chord: a scalable peer-to-peer lookup protocol for internet applications, IEEE/ACM Transactions on networking 11 (2003) 17–32.

[15] G. Chiola, G. Cordasco, L. Gargano, A. Negro, V. Scarano, Optimizing the finger tables

in chord-like dhts, Concurrency and Computation: Practice and experience 20 (2008) 643–657.
[16] H. A. Kalodner, M. Carlsten, P. M. Ellenbogen, J. Bonneau, A. Narayanan, An empirical study of namecoin and lessons for decentralized namespace design., in: WEIS, volume 1, 2015, pp. 1–23.
[17] G. Giamouridis, B. Kang, L. Aniello, Blockchain-based dns: Current solutions and challenges to adoption (2024).
[18] Y. Liu, D. Wang, Y. Hu, A scheme of integrating dns into p2p service network, in: 2011 4th IEEE International Conference on Broadband Network and Multimedia Technology, IEEE, 2011, pp. 8–11.
[19] P. Danielis, V. Altmann, J. Skodzik, T. Wegner, A. Koerner, D. Timmermann, P-donas: a p2p-based domain name system in access networks, ACM Transactions on Internet Technology (TOIT) 15 (2015) 1–21.
[20] R. Sancho, R. L. Pereira, Hybrid peer-to-peer dns, in: 2014 International Conference on Computing, Networking and Communications (ICNC), IEEE, 2014, pp. 977–981.
[21] R. Cox, A. Muthitacharoen, R. T. Morris, Serving dns using a peer-to-peer lookup service, in: Peer-to-Peer Systems: First InternationalWorkshop, IPTPS 2002 Cambridge, MA, USA, March 7–8, 2002 Revised Papers 1, Springer, 2002, pp. 155–165.
[22] Y. Song, K. Koyanagi, Study on a hybrid p2p based dns, in: 2011 IEEE International Conference on Computer Science and Automation Engineering, volume 4, IEEE, 2011, pp. 152–155.
[23] K. Kohler, One, Two, or Two Hundred Internets?: The Politics of Future Internet Architectures, Technical Report, ETH Zurich, 2022.
[24] C. V. Helliar, L. Crawford, L. Rocca, C. Teodori, M. Veneziani, Permissionless and permissioned blockchain diffusion, International Journal of Information Management 54 (2020) 102136.
[25] C. G. Dickey, C. Grothoff, Bootstrapping of peer-to-peer networks, in: 2008 International Symposium on Applications and the Internet, IEEE, 2008, pp. 205–208.