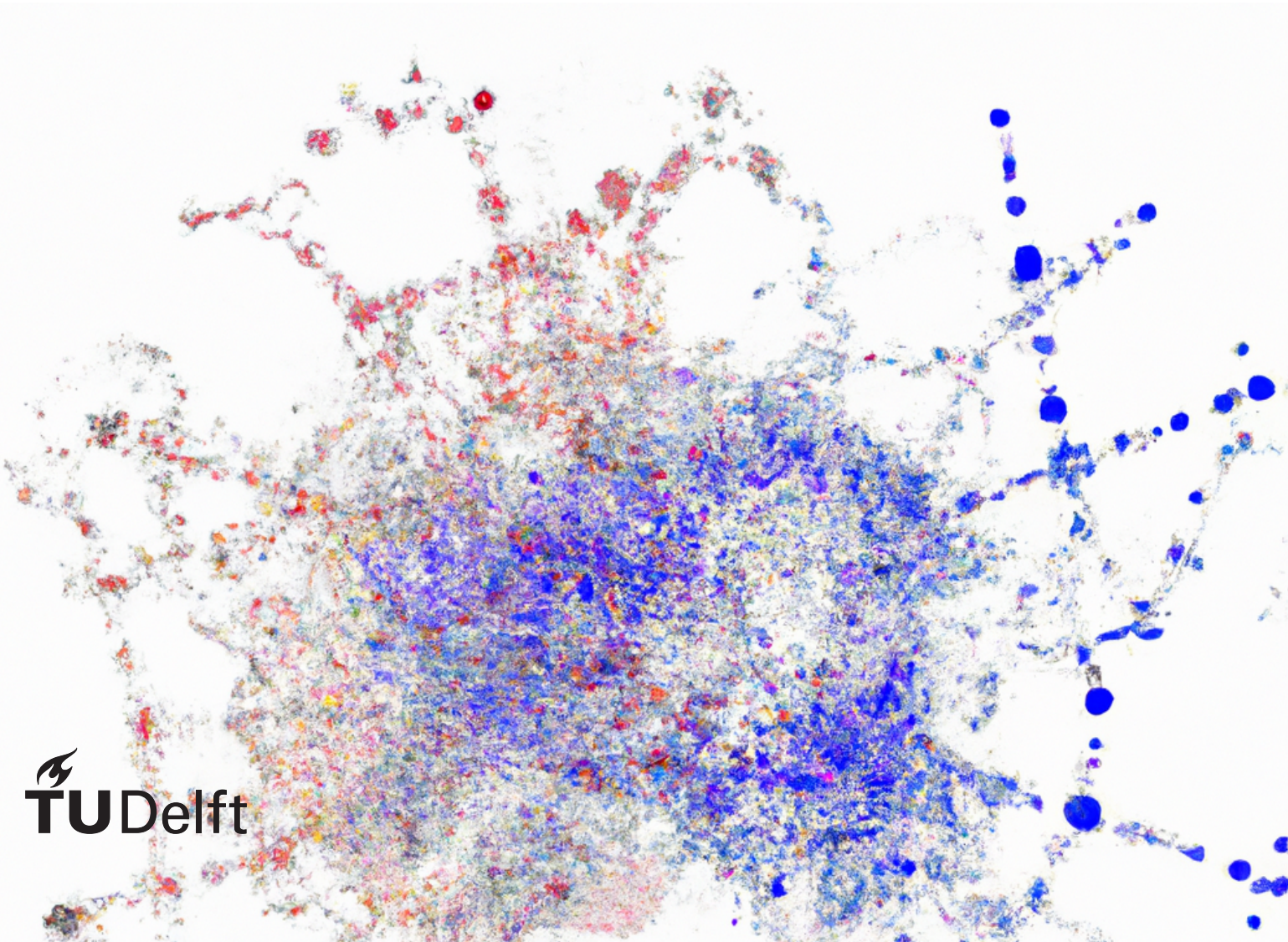
Towards Sovereign Domains P2P Overlays in Blockchain Systems

Naqib Zarin



TU Delft

Towards Sovereign Domains P2P Overlays in Blockchain Systems

by

Naqib Zarin

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday March 17, 2023 at 10:30 PM (CET).

Student number:	4384474	
Thesis committee:	Dr. ir. Stefanie Roos,	TU Delft, daily supervisor
	Dr. ir. Lydia Y. Chen,	TU Delft, supervisor
	Dr. ir. Stjepan Picek,	TU Delft

This thesis is confidential and cannot be made public until March 17, 2023.

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Abstract

Blockchain technology has proven to be a promising solution for decentralized systems in various industries. At the core of a blockchain system is the peer-to-peer (P2P) overlay, which facilitates communication between parties in the blockchain system. Recently, there is increasing evidence that this P2P overlay plays a major role in limited transaction throughput, limited scalability, and security threats in a blockchain system. In this work, it is explained that this bottleneck is caused by the fact that currently deployed P2P overlays cannot fully capture the complexity of a blockchain system. We propose a novel approach to address these issues by introducing a novel overlay architecture. Our approach, Sovereign Domains P2P Overlays, allows for nodes in a single network to belong to multiple heterogeneous groups (called domains), each with their own set of protocols tailored to the characteristics and needs of the nodes in that domain. To demonstrate the effectiveness of the proposed overlay architecture, two novel node discovery protocols, FedKad and SovKad, were designed and implemented. The simulations conducted in a custom-built simulator show that SovKad outperforms node discovery in a Structured P2P Overlay (Kademlia) and node discovery in a Federated P2P Overlay (FedKad), providing evidence of the potential of the Sovereign Domains P2P Overlay in blockchain systems.

Preface

With this master thesis, I conclude my time at Delft University of Technology for now. It has been an incredible journey, and I am grateful to have had the opportunity to learn from some of the best teachers in the world. I want to express my sincere gratitude to all the teachers who have taught me during my bachelor's and master's studies. Your dedication and commitment to education have been a great source of inspiration to me.

I would also like to thank Lydia and Stjepan for being a part of the thesis committee. In particular, I want to extend my heartfelt to my supervisor Stefanie Roos for her invaluable supervision and guidance. She has always challenged me to push my limits and sparked my curiosity, and it is because of her that I fell in love with academic research. I am eternally grateful to her for her support and encouragement.

I would like to thank my colleagues Isaac Sheff, Christopher Goes, and Fatemeh Shirazi, for their invaluable contributions and insights from an industry perspective. I have learned so much from them and I am grateful for their support and encouragement to explore this topic.

Working on this thesis has not been easy, and it would not have been possible without the support of my friends. In particular, I want to thank Dirk van Bokkem, who was also my sparring partner throughout this thesis.

I also want to thank my family for their unconditional love and support throughout this journey. In particular, I want to thank my parents Habib "Dada" Zarin and Rabea "Dadai" Arefi for their constant encouragement and life lessons that helped me gain perspective when things got tough. To my brothers Nawid, Hamid, Monib, Hasib, and Emal, thank you for always being there for me and supporting me through thick and thin. It is a privilege knowing that I can always count on you. To my sisters Fahima and Marzia, thank you for your love and support. I also want to thank my in-laws, all members of the Hashemi family, for their enthusiasm and support along the way.

Lastly, I want to express my heartfelt gratitude to the love of my life, Morwarid. Your unwavering support and encouragement have been my strength, and it was a great comfort to know that I would come home to you. Thank you for your understanding and patience during this journey. You have been invaluable, for which I feel gratitude I cannot describe with words.

Once again, thank you to everyone who has contributed to this thesis in one way or another. This work is dedicated to all of you.

Naqib Zarin
Zanzibar, March 2023

Contents

1 Introduction

Nowadays, a wide range of services and platforms are built as a distributed system. A distributed system is a computer system that consists of multiple independent components that communicate and work together to achieve a common goal [59]. For instance, during the COVID-19 pandemic, independent public hospitals and clinical laboratories work together by sharing data about patients so that vaccination certificates can be issued and the spread of the virus can be traced [52].

However, many distributed systems of today are centralized, and relying on single trusted entities to perform certain tasks raises several concerns. Administrators of these central entities often have access to sensitive data, and hackers can target a single point of failure to break the system. For example, in the case of COVID-19, private data of more than 60,000 people who have taken a COVID-19 test has leaked in The Netherlands; this data was in possession of a centralized entity [19]. In addition to these security threats, obtaining services from a central entity often also involves additional operational costs, such as the costs of maintaining and scaling a central infrastructure. These issues do not only highlight the need for more secure distributed systems, but also systems that are more decentralized.

To solve these kinds of issues, there has been a growing interest in blockchain technology as a promising solution for distributed systems that require decentralization. Blockchain systems are not only popular in nation-specific industries such as healthcare [37, 52] and identity management [3, 20], their application can also be found on a global scale. For example, cryptocurrencies such as Bitcoin [41] and Ethereum [12] function as world wide decentralized digital payment systems with a combined market capitalization of more than 600 billion US dollars [45].

Blockchains are based on a distributed ledger technology, which allows multiple mistrusting parties, also known as *nodes*, to reach consensus on the state of the ledger without relying on a central authority for coordination [66]. This state is typically an append-only and immutable history of *transactions*, which are stored in a chain of *blocks*. Each block is virtually linked to the previous block by adding the *cryptographic hash value* of that previous block to the header of the current block. This ensures that previously added blocks are immutable, because modifying such a block results in a new hash value that is conflicting with the hash value in the header of the next block. This allows nodes to detect tampering. Whenever a new block of transactions has been executed by nodes in the blockchain system, the next state is achieved (see Figure 1.1).

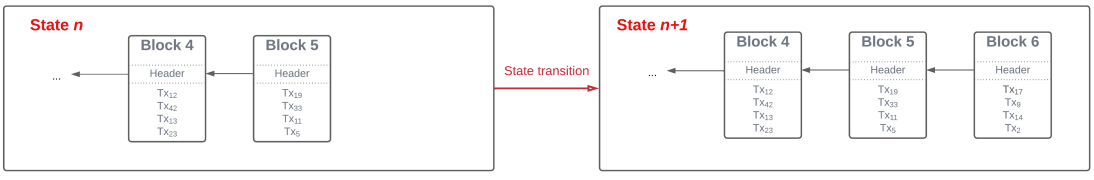
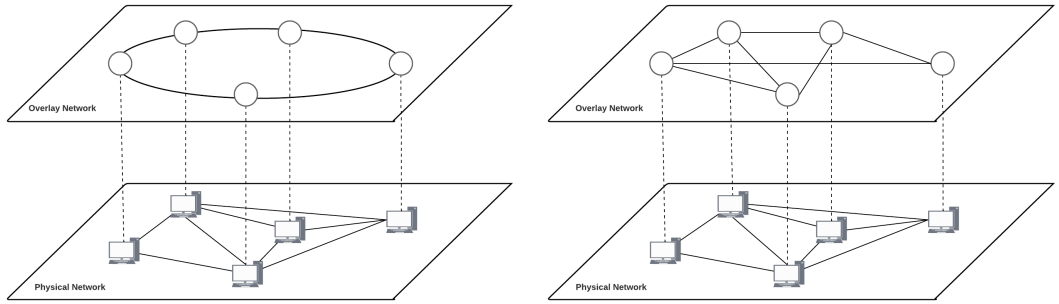


Figure 1.1: State transition from state n to $n+1$ because a new block (containing transactions Tx_{17}, Tx_9, Tx_{14} and Tx_2) is added to the chain of blocks.

1.1. Importance of P2P Networks in Blockchain Systems

At the core of a blockchain system is the *peer-to-peer (P2P) network*, which facilitates communication and exchange of transactions and blocks between nodes. Blockchain systems use a P2P architecture as a base due to their decentralized and self-stabilizing nature [65, 17]. Because nodes in the network communicate with each other directly, there is no need for a central server or entity to mediate between nodes [53]. Not only are P2P architectures more robust than client-server models as they eliminate a single point of failure, they also scale naturally because new nodes typically make new computing, bandwidth and storage resources available [32].



(a) Structured P2P Overlay.

(b) Unstructured P2P Overlay.

Figure 1.2: Structured P2P overlays adhere to a pre-defined topology, such as a ring (on the left), whereas unstructured P2P overlays are characterized by nodes being connected to other nodes randomly (on the right).

A P2P network is composed of multiple nodes, which are connected to each other through a set of links. These links are often established by using a variety of protocols, such as *Transmission Control Protocol (TCP)*, which allow nodes to exchange information over the Internet. The P2P network in a blockchain system is commonly structured through the use of a *P2P overlay*. An overlay is the virtual network that is built on top of the underlying physical infrastructure (i.e., the Internet) and is responsible for managing the links between nodes and ensuring that information is propagated efficiently throughout the network [53]. P2P overlays in blockchain systems are typically either *structured* or *unstructured* (see Figure 1.2). A structured topology is tightly regulated, with specific rules regarding where nodes are positioned in the topology [34], whereas nodes in unstructured overlays are connected to each other in a random manner [27]. Different from the underlying network, links in a P2P overlay are directed, indicating that the node at the start of this link has the contact information (e.g., *Internet Protocol (IP)* address) of the node at the end of this link - the *neighbor*.

More recently, there is increasing evidence that the P2P overlay is the source of critical challenges in a blockchain system [4, 18, 15], which are currently a major factor holding back mass adoption of blockchain technology [18]. As we will see in chapter 2, one of the main symptoms of an inefficient P2P network design is slow transaction and block dissemination among nodes in the P2P overlay, which causes low transaction throughput[48], poor scalability [64], and security vulnerabilities [21] in a blockchain system.

1.2. Main Problem with Current P2P Overlays in Blockchain Systems

The main problem with currently deployed P2P overlays, both structured and unstructured, is that they are not fully capable of capturing the complexity of a blockchain system. The many interdependent components as well as the high degree of heterogeneity among the nodes in terms of motivations, resources and behaviors make blockchain systems complex. Structured P2P overlays, such as *Distributed Hash Tables (DHTs)* [39, 36], offer efficient routing protocols that ensure that data is propagated quickly within the network. However, DHTs are not flexible enough to handle the complex and dynamic nature of a blockchain system. For example, nodes in a blockchain system can join or leave the network at any time, make different resources available, and the network topology can change frequently. DHTs rely on a fixed routing protocol, which can result in reduced efficiency and slower dissemination of information if the network undergoes changes [40]. On the other hand, unstructured P2P overlays can better handle the dynamic nature of a blockchain network [5]. Nodes can join or leave the network at any time without disrupting the overall network structure. However, unstructured overlays often suffer from high communication overhead, which can lead to slower dissemination of information and reduced efficiency. For example, 44% of the overall traffic of the Bitcoin network is redundant [42]. Blockchain systems require a sophisticated P2P overlay to ensure efficient

and secure operations in presence of node heterogeneity. Therefore, the P2P overlay needs to offer flexibility to adapt protocols and topologies to the specific needs of various groups of nodes in a blockchain system, without sacrificing performance and security.

Recently, new P2P overlays have emerged that aim to improve the performance, while also offering this level of flexibility. Typically, this is achieved by grouping nodes into smaller sub-networks that require less communication with other sub-networks, where each sub-network has its own independent P2P overlay [57, 11, 62, 48]. For instance, Ethereum has one dedicated P2P overlay for consensus nodes and one for execution nodes [57], and heterogeneous parachains in Polkadot [11] each have their own separate P2P overlays. We refer to this type of architecture as *Federated P2P Overlays*.

While such a Federated P2P Overlays architecture can improve performance and provide more flexibility, it also brings about new challenges. One major issue is that, in current blockchain systems, security may be sacrificed to achieve flexibility. This is because independent P2P overlays that are smaller in size are generally more vulnerable to network layer attacks. An adversary can potentially compromise a significant portion of the network with fewer resources [55], which can undermine the overall security of the system.

1.3. Objectives and Research Questions

A well-designed P2P overlay architecture for blockchain systems should provide the benefits of a single, large overlay while still offering the advantages of smaller, optimized protocol groups. The main goal of this research is to propose a novel P2P overlay architecture for blockchain systems that meets the requirements of high performance, strong security, and flexibility to accommodate node heterogeneity. More specifically, the proposed architecture should segment the network into smaller groups of nodes, each with its own optimized protocols, while maintaining the security of a single, large P2P overlay.

To achieve the main objective of the present study, the following main research question is answered:

> How can we design and evaluate a scalable P2P overlay architecture for blockchain systems that has the performance of a small overlay, the security of a large overlay, and the flexibility to accommodate node heterogeneity?

Towards answering this research question, the following sub-questions need to be explored first:

> **(1)** What are the strengths and limitations of existing P2P overlay architectures utilized in blockchain systems with respect to performance, security, and flexibility to accommodate node heterogeneity?
>
> **(2)** What design considerations can be taken into account to overcome the limitations of existing P2P overlay architectures in blockchain systems and to develop an architecture that delivers high performance, strong security, and flexibility to accommodate node heterogeneity?
>
> **(3)** How can we evaluate the effectiveness of our proposed P2P overlay architecture, and how does it compare to state-of-the-art architectures in blockchain systems?

1.4. Main Contributions

We propose a novel P2P overlay architecture for blockchain systems called *Sovereign Domains P2P Overlay* that has the benefits of a small and large P2P overlay, while also offering flexibility to accommodate node heterogeneity. In our architecture, the overlay is segmented into *sovereign domains*, with each domain operating independently and having its own internal structure and protocols such as routing and node discovery. These protocols can be tailored to the characteristics and needs of the domain. Furthermore, nodes can be part of multiple domains simultaneously.

The performance of Sovereign Domains P2P Overlay is enhanced because domains are smaller in size than the entire overlay. Therefore, disseminating information in a domain has a similar effect as disseminating information in a smaller P2P overlay, which is more efficient than disseminating information in a larger P2P overlay [60]. Furthermore, Sovereign Domains P2P Overlay have the security of a larger P2P overlay, because nodes maintain also a small number of links to nodes from other domains, thus increasing the network connectivity and allowing for building statistical models about other domains to monitor a predefined state of quality. This makes it easier to detect malicious behavior and repair domains if necessary. Finally, Sovereign Domains P2P Overlay accommodate node heterogeneity by allowing nodes to participate in multiple domains simultaneously, each with its own internal structure and protocols tailored to the characteristics and needs of its nodes.

To demonstrate the effectiveness of Sovereign Domains P2P Overlay, we have designed and implemented a concrete node discovery protocol called *SovKad* using a custom built simulator. We have compared this protocol to state-of-the-art node discovery protocols used in blockchain systems, specifically Kademlia, which is a widely adopted single-overlay node discovery protocol. To further evaluate the performance of SovKad, we have also designed and implemented a node discovery protocol called *FedKad* for Federated P2P Overlays based on available research in the field of blockchain systems. We have compared the performance of SovKad to FedKad and to Kademlia.

Our results show that SovKad outperforms node discovery in a Structured P2P Overlay (Kademlia) and node discovery in a Federated P2P Overlay (FedKad), providing evidence of the potential of the Sovereign Domains P2P Overlay in blockchain systems.

1.5. Thesis Organization

The present study consists of two parts. In the first part, the main goal is to introduce our novel architecture. We start by providing background information on P2P overlays and P2P networking in blockchain systems, and explore related work in chapter 2. Then, we outline the model assumptions, and define the goals of our architecture in chapter 3. We conclude the first part by describing the Sovereign Domains P2P Overlay in chapter 4. We formally define its key components and also provide a list of notations that are relevant for the second part of this thesis.

In the second part, our main goal is to evaluate the effectiveness of our architecture. We do this by designing and implementing two novel node discovery protocols - FedKad and SovKad - for both Federated P2P Overlays and Sovereign Domains P2P Overlay. Because FedKad and SovKad are inspired by Kademlia, we describe Kademlia first in chapter 5. In chapter 6, we describe FedKad and SovKad in more detail. To achieve the main goal of the second part of this thesis, we compare FedKad and SovKad to each other, as well as to Kademlia in chapter 7. Finally, we present a conclusion that provides answers to the main research question and summarizes the key findings of this study (chapter 8). Additionally, we provide suggestions for future research directions in this field.

1.6. Publications

The research conducted during this Master Thesis has lead to the submission of the following publication:

- N. Zarin, S. Roos, and I. Sheff. "Towards Sovereign Domains P2P Overlays in Blockchain Systems". Submitted to the *fifth ACM Conference on Advances in Financial Technologies (AFT'23)*. March, 2023.

I

Introducing Sovereign Domains P2P Overlay

2

Background and Related Work

2.1. P2P Overlay Fundamentals

A P2P overlay network is computer network built on top of an existing network, usually the Internet, and enables direct communication and resource sharing among nodes. It abstracts out the physical network switches and routers and defines virtual links between nodes. P2P overlays have been around for decades, with early examples like Napster and Gnutella [25] pioneering the use of decentralized systems for file sharing and content distribution. Today, P2P networks continue to be widely used in a variety of applications, from file sharing [9] and data-streaming [63] to blockchain technology [41].

Historically, there are two classes of P2P overlays: centralized and decentralized. Centralized P2P overlays (e.g., Napster [51]) rely on a central party in the overlay for coordination. For example, if the P2P system is a file system, and a node wants to retrieve a certain file, it could contact this central party to get the location of the node that is responsible for storing this file. Decentralized P2P overlays, on the other hand, rely on distributed algorithms to manage the communication among nodes and do not make use of a central coordinator. The absence of a central coordinator has the advantage of scalability and security. A decentralized P2P overlay is more scalable, because there is not a certain infrastructure that needs to be maintained and scaled with an increase in the number of nodes in the network. It also eliminates a single point of failure.

However, decentralized P2P overlays are more difficult to design and maintain than centralized P2P overlays as they consist of various components that influence each other. In this section, we briefly discuss those components.

2.1.1. Overlay Topologies

The overlay topology refers to a virtual network that is formed by nodes in the P2P overlay [53]. It is an abstract representation of how nodes are connected and therefore, of which nodes can directly communicate with each other.

Two factors that have an influence on the performance and security of an overlay are the number of links between nodes and the characteristics of the selected neighbor [1]. The more links there are between nodes, the more efficient the network can be in terms of disseminating transactions and blocks. However, too many links can also lead to increased network congestion and latency, which can slow down the network and impact its performance. Furthermore, too many links can also increase the attack surface of the network, providing more opportunities for attackers to exploit vulnerabilities and compromise the system. On the other hand, too few links between nodes can create for vulnerability in the network, making it easier for attackers to disrupt or compromise the system. Therefore, a balance must be struck between the number of links and the overall performance of the system. In addition to the number of links, the characteristics of the selected neighbor also plays a critical role in determining the performance and security of an overlay topology. For example, selecting neighbors that are geographically closer to a node can result in faster communication and improved network performance [47]. On the other hand, selecting neighbors that have less resources or are untrustworthy can pose a security risk to the network, as they may attempt to disrupt activities in the system or engage in malicious behavior. Selecting neighbors that are known to be trustworthy and have a good reputation can help to reduce the risk of attacks and improve the overall security of the network [46]. Conversely, selecting neighbors that are untrustworthy or have a low reputation can increase the risk of attacks

and compromise the security of the system.

2.1.2. Node Discovery Protocols

The node discovery protocol refers to how a node can establish and maintain a link to reachable neighbors (i.e., nodes that have not left the P2P overlay) in the overlay [43]. Node discovery is one of the most important aspect of P2P overlays, because it determines the potential connections between nodes and the speed at which those connections can be established.

The performance of a P2P overlay network is directly influenced by the effectiveness of the node discovery protocol [43]. A slow or inefficient node discovery protocol can cause delays in transaction processing, leading to a decrease in the network's overall throughput.

The security of a P2P overlay network is also influenced by the node discovery protocol [43]. The protocol determines which nodes a particular node is connected to, and these connections can be exploited by malicious actors to launch attacks on the network. If the node discovery protocol is not secure, it can be manipulated by attackers to launch overlay attacks such as eclipse attack [43]. Therefore, the node discovery protocol is essential to protect the network from attacks and maintain the integrity of the blockchain system.

2.1.3. Routing Mechanisms

The routing mechanism refers to the selection of nodes for relaying communication through the network [54]. This has a direct impact on the speed and reliability of information dissemination in a blockchain network.

A well-designed routing mechanism can significantly improve the speed and reliability of information dissemination in a blockchain network. For instance, some routing mechanisms use algorithms (e.g., [26]) select nodes based on their proximity to the destination node. By selecting nodes that are closer to the destination, these algorithms can reduce the number of hops required to transmit the information, thereby reducing latency and improving the speed of communication. Other routing mechanisms use algorithms (e.g., [46]) that prioritize nodes based on their reliability or availability. Nodes with high trust scores are more likely to be reliable and available, and are therefore more likely to be selected for relaying communication. By prioritizing reliable and available nodes, these algorithms can improve the reliability of information dissemination, reducing the risk of data loss or corruption. Finally, some routing mechanisms use algorithms (e.g., [8]) that incorporate redundancy to improve the fault tolerance of the network. For example, an algorithm may use multiple paths for transmitting information, so that if one path fails, the information can be transmitted through an alternate path. By incorporating redundancy, these algorithms can improve the resilience of the network, reducing the risk of data loss or disruption.

The routing mechanism also influences the security of a blockchain system [1]. By controlling the flow of information through the network, the routing mechanism can help prevent attacks that aim to disrupt the network or compromise its integrity. For instance, a routing mechanism that is able to detect and isolate malicious nodes can help limit the influence of malicious nodes. In other words, the routing mechanism can help prevent malicious nodes from disrupting the network or compromising its security.

2.1.4. Attacks on P2P Overlays

Overlay attacks are a class of attacks that exploit the decentralized, distributed nature of P2P networks to compromise their security and availability [13]. In the context of blockchain systems, overlay attacks can have severe consequences, including the theft of funds, the manipulation of transaction history, and the disruption of the consensus process.

One type of overlay attack is the *Sybil attack*, in which an attacker creates multiple fake identities or nodes to control a significant portion of the network. This allows the attacker to manipulate the consensus process, disrupt transaction flow, and potentially steal funds [44]. To mitigate Sybil attacks, blockchain systems can implement mechanisms such as Proof-of-Work, Proof-of-Stake, or other consensus algorithms that require a certain level of computational or stake-based investment.

Another type of overlay attack is the *eclipse attack*, in which an attacker takes control of a node's incoming and outgoing connections to isolate it from the rest of the network [8]. This allows the attacker to prevent the node from receiving valid transactions or blocks and can lead to double-spending attacks. To mitigate eclipse attacks, blockchain systems can implement techniques such as random peer selection and connection diversity (as explained in [43]) to reduce the likelihood of an attacker successfully eclipsing a node.

Routing attacks are another class of overlay attacks that involve manipulating the routing information of nodes to redirect traffic and isolate or manipulate nodes [8]. These attacks can disrupt transaction flow,

prevent the propagation of new blocks, and can even allow attackers to steal funds. To mitigate routing attacks, blockchain systems can implement techniques such as random routing to make it harder for attackers to manipulate the routing information.

Denial-of-service (DoS) attacks are another type of overlay attacks that involves overwhelming a node or network with a high volume of traffic or malicious messages. This can lead to network congestion, slowdowns, and even denial of access to the network [13]. To mitigate DoS attacks, blockchain systems can implement techniques such as rate limiting, throttling, and anti-spam mechanisms to limit the volume of incoming traffic and prevent nodes from being overwhelmed [33].

2.2. Blockchain Networking

Blockchains are based on a distributed ledger technology, which allows multiple mistrusting parties to reach consensus on the state of the ledger without relying on a central authority for coordination [66]. This state is typically an append-only and immutable history of *transactions* which are stored in a chain of *blocks*. Each block is virtually linked to the previous block by adding the *cryptographic hash value* of that previous block to the header of the current block. This ensures that previously added blocks are immutable, because modifying such a block results in a new hash value that is conflicting with the hash value in the header of the next block. This allows nodes to detect tampering. Whenever a new block of transactions has been executed by parties in the blockchain system, the next state is achieved (see Figure 1.1).

2.2.1. Node Heterogeneity

Node heterogeneity is a common phenomenon in blockchain systems that arises due to the diversity of nodes that participate in the network [58]. Different types of nodes have different motivations, resources, and behaviors, which can affect the overall performance, security, and stability of the system. A typical blockchain system has the following type of nodes: *wallet nodes*, *mempool nodes*, *miner nodes*, *validator nodes*, and *full nodes*.

Wallet nodes are nodes that do not store the full blockchain history, but rely on other nodes to provide them with the necessary information to verify transactions. They are motivated by the desire for low storage and bandwidth requirements. Mempool nodes store a subset of unconfirmed transactions that are waiting to be added to the next block by miners. These nodes are motivated by the desire to earn transaction fees by prioritizing transactions in the mempool, but they require high processing power to handle the large volume of transactions that are constantly being added to the mempool. Miner nodes are responsible for adding new blocks to the blockchain and are motivated by the reward of newly created coins and transaction fees. They require significant computational resources to perform Proof-of-Work or Proof-of-Stake consensus algorithms to validate transactions and add new blocks to the chain. Validator nodes participate in consensus algorithms and are responsible for ensuring that the blockchain is secure and valid. They are motivated by the desire to maintain the integrity of the blockchain and earn rewards for their work. Validator nodes typically require high processing power and storage capacity to perform complex consensus algorithms and store the full blockchain history. Finally, full nodes store the entire blockchain history and participate in the network as peers. They are motivated by the desire to contribute to the security and decentralization of the network and may provide services such as transaction broadcasting, block propagation, and storage. Full nodes require high storage capacity and processing power to handle the large volume of transactions and blocks that are added to the blockchain.

In conclusion, node heterogeneity in blockchain systems is a complex phenomenon that can have significant implications for the performance, security, and stability of the system. Understanding the motivations, resources, and behaviors of different types of nodes can help in the design and optimization of P2P overlays that can accommodate for this heterogeneity and ensure the long-term viability of a blockchain system.

2.2.2. P2P Networking From a Transaction Life Cycle Perspective

Before any state transition can take place in a blockchain system, transactions and blocks must be disseminated and validated, which requires communication between different type of nodes in the P2P overlay. This can be portrayed by the typical life cycle of a transaction (see Figure 2.1).

It starts with a user having the intention to change the current state into a preferred state. For example, a user may want to send 1 Ethereum token to another user. A *wallet* can be used to express this intent in a standardized manner, so that other parties in the blockchain can understand the preferred state change. More concretely, a wallet transforms an intent into a transaction. This transaction is then sent to a group of

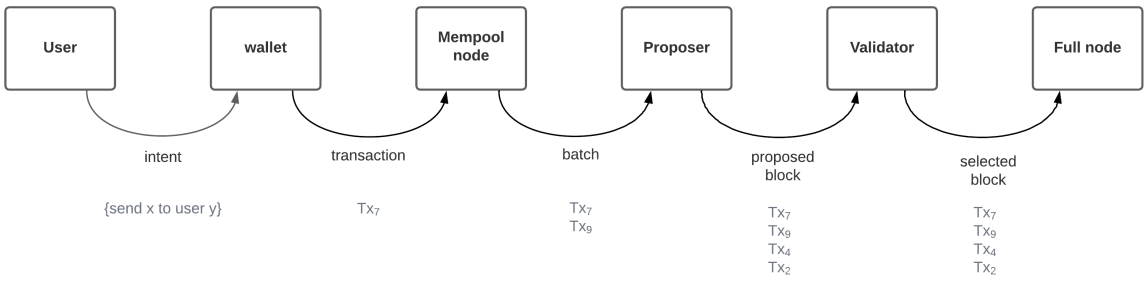


Figure 2.1: Typical life cycle of a transaction in a blockchain system. A transaction is created, disseminated, added in a proposal block, verified and finally executed.

parties called *mempool nodes*. Mempool nodes are responsible for storing and disseminating a transaction to other nodes, so that the network becomes aware of this desired state change. As many users create various transactions, the blockchain must reach a consensus on the order in which these transactions are added to the blockchain. Note that this order is important as some transactions may render other transactions invalid. For example, if a user has 1 Ethereum token and it creates two transactions sending that Ethereum token to two other users, the transaction that will be executed first renders the other transaction invalid as the same Ethereum token cannot be spent twice. Various consensus algorithms exist to decide on which block to add next. In a typical blockchain, *proposer* nodes (sometimes referred to as *miners*) batch several transactions into a single block and propose this block, which then needs to be validated and accepted by a significant portion of *validators* in the network through a voting mechanism. Once validators reach consensus on which block to add next to the chain of blocks, *full nodes* can execute the transactions in that block, resulting in a new state. Proposers, validators and full nodes are often rewarded for their contribution.

2.2.3. Consequences of Slow Information Dissemination

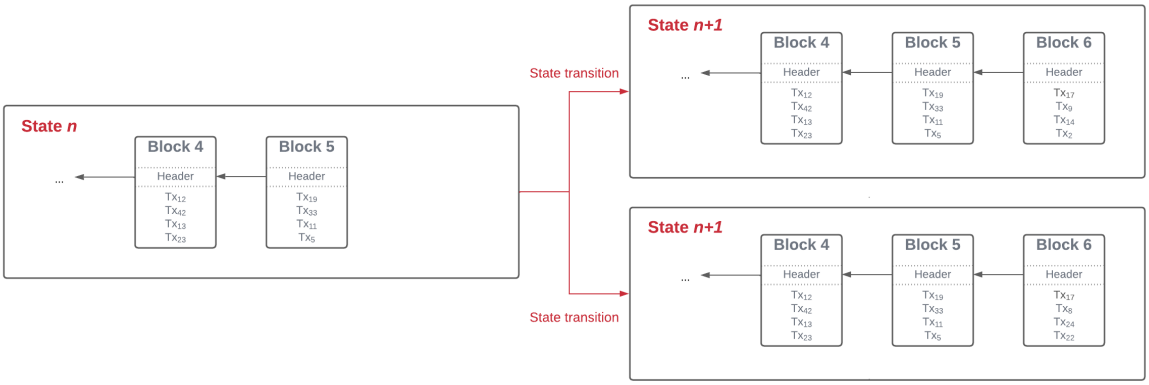


Figure 2.2: The blockchain has forked due to non-identical transactions in block 6 after the two nodes transition to *n+1*.

Slow information dissemination decreases a blockchain system's transaction throughput[1], limits its scalability, and also imposes security threats.

The system's throughput is decreased because transactions and blocks that are not received can also not be processed. This increases the latency of reaching consensus on the next state of the blockchain. Additionally, slow information propagation can limit the scalability of a blockchain system. Although a growing blockchain system also implies an increase of available resources, an inefficient P2P network design can limit the blockchain system's ability to scale the throughput linearly with the number of nodes [64].

Slow information dissemination makes a blockchain system less resilient against malicious nodes as it provides an opportunity to perform attacks such as *forking* and *double-spending* [22]. If not all parties have quick access to the same information because of inefficient P2P overlay, groups of nodes may have conflicting

[1] The number of processed transactions per time unit

perceptions about the new state of the blockchain, resulting in a *chain fork* (see Figure 2.2). By the time honest nodes become aware that they are building on the wrong chain (e.g., the shorter chain), they may have already wasted computation, bandwidth and storage resources on *stale blocks* [22]. When they learn about the correct chain, they revert these stale blocks and continue building on the correct chain. However, during this time window, an attacker may create conflicting transactions, potentially leading to a double-spending attack. For example, an attacker can purchase an item for 1 Ethereum token on the wrong chain and send the same Ethereum token to another wallet node that it controls on the correct chain. By the time the blockchain re-configures and rejects the first transaction, the attacker may have already received the purchased item and still have the coin it spent in its possession, thus successfully performing a double-spending attack.

2.2.4. Current P2P Overlays in Blockchain Systems

P2P overlays are essential for ensuring secure and efficient communication between nodes in a blockchain network. As we saw in this chapter, a blockchain system has a great level of node heterogeneity. One important factor that can contribute to slow information dissemination in a P2P overlay architecture is to have protocols and topologies that do not account for this node heterogeneity [56].

In recent years, new P2P overlays have been developed to improve performance and provide greater flexibility to accommodate node heterogeneity. These overlays typically group nodes into smaller sub-networks that require less communication with other sub-networks, with each sub-network having its own independent P2P overlay [57, 11, 62, 48]. For example, Ethereum has dedicated P2P overlays for consensus nodes and execution nodes [57], while Polkadot uses separate P2P overlays for its heterogeneous parachains [11].

Although having this so-called *Federated P2P Overlay* architecture improves performance and gives more flexibility, it introduces new issues. First, each overlay must be secured independently, and smaller P2P overlays are known to be more vulnerable against network layer attacks because an adversary can more easily compromise a significant portion of the network with fewer resources [55]. To ensure the security of the entire blockchain system, there is a need for a unified security system that can provide end-to-end protection across different overlays. This is difficult to design, because nodes in different overlays do not have access to the same information [29]. Second, having multiple also disconnected overlays implies that nodes from different overlays cannot communicate with each other, unless those independent overlays are interoperable. Making multiple P2P networks interoperable requires complicated interfaces, which are difficult to develop and costly to maintain [14]. Finally, in an interconnected blockchain ecosystem, it may not be practical to have a varying number of P2P overlays for each individual blockchain system, as this can lead to a linear increase in the number of interfaces that must be managed with every overlay integration. Consequently, the costs of maintenance and operation also grow linearly, which is undesirable.

3

System Model and Goals

3.1. Network Model

This work models a P2P overlay as a connected directed graph, denoted by $G = (V, L, D)$. Here, V is the set of participants, and any $v_i \in V$ represents a node in the network. There exists a link $(i, j) \in L$ if and only if node v_i has node v_j in its routing table. Furthermore, we assume that there exists a mechanism that partitions nodes in the P2P overlay into one or multiple domains $d_i \in D$. The specifics of this mechanism is out of scope of this work. Nodes are not restricted to a single domain and can co-exist in multiple domains simultaneously. However, since participation in multiple domains can be resource-intensive, we upper bound the fraction of nodes that have these capabilities to p and the number of domains they can be part of to m. Furthermore, nodes can join and leave the P2P overlay as they please, and the overall churn rate is c. It is also assumed that nodes in the network establish identities through a Sybil-resistant identity generation mechanism. This means that each node has a unique identifier that enables two peers to locate one another. Finally, the present work assumes a partially synchronous communication model that guarantees that messages will be delivered to their targets within an unknown time. This is in line with existing models in blockchain systems, such as Ethereum [12] and Polkadot [2].

3.2. Threat Model

We assume that the P2P overlay consists of honest and malicious nodes. The fractions of malicious users in the entire network is f, meaning that the fraction of honest nodes is $h = 1 - f$. Honest nodes follow the node discovery protocol correctly, whereas malicious nodes adopt various strategies to disrupt the activities in the network.

We assume that the adversary positions itself within the system (*internal*) and that each colluder node observes part of the system (*local*), but that altogether the adversary knows the topology. We also assume an *active* adversary, that is, colluding malicious nodes try to decrease the performance by deviating from the lookup protocol. What it exactly does depends on the node discovery protocol and the state of the system. This implies that the adversary is *adaptive* as it changes its behavior from time to time based on its observations. However, the adversary's computational power is polynomially bounded in the security parameter of the encryption scheme. Similar to most blockchain systems [57, 41, 48, 62], we assume that $f < \frac{1}{3}$.

The main goal of the adversary is to disrupt the activities in the network. Because the P2P overlay is naturally split in multiple smaller sub-networks, we assume that the attacker's goal is to decrease the performance of the node discovery protocol in any domain. We can model the influence and power of the adversary as the fraction of colluding nodes in a domain. The quality of a domain is determined by the performance of the protocols in that domain, which depends on the fraction of malicious nodes the domain. The exact strategies depend on the state of the network as we assume an adaptive adversary. For example, the adversary might send a fraction (or all) colluder nodes to a single domain and try to corrupt the entire domain. Finally, we assume that the adversary can also have colluders that co-exists in multiple domains simultaneously as long as it happens within the boundaries of p and m.

3.3. System Goals

P2P overlays operate similarly to conventional networks, requiring an initial creation, provisioning of services and customers, maintenance of quality, and repair when needed. Similar to standard requirements for current P2P overlay architectures [1], the architecture of sovereign domain overlays must at least meet requirements such as performance, scalability, security, and robustness. Additionally, we require that the architecture has the flexibility to accommodate node heterogeneity. Therefore, Sovereign Domains P2P Overlays must meet at least the following requirements:

1. *Performance*: Routing should be efficient with a minimal number of overlay hops, and the bandwidth for constructing and maintaining the overlay should be kept to a minimum.

2. *Scalability*: The architecture must support (numerical) scalability, allowing for a large number of participating nodes without significant performance degradation.

3. *Security*: The overlay must offer protection against the most severe overlay attacks performed by adversaries.

4. *Robustness*: The architecture should be resilient to node and network link failures, with all resources remaining accessible to all peers.

5. *Flexibility*: The architecture must be flexible enough to accommodate node heterogeneity, thus allowing nodes to operate in multiple sovereign domains with their own internal structure and protocols tailored to their specific needs.

4

Design of Sovereign Domains P2P Overlay

4.1. Overview

We propose a novel P2P overlay architecture for blockchain systems called *Sovereign Domains P2P Overlay* that can achieve the performance of a small P2P overlay, the security of a large P2P overlay, and flexibility to accommodate for node heterogeneity. Our approach segments the overlay into several sovereign sub-networks called *domains*, with each domain operating independently and having its own internal structure and protocols such as routing and node discovery, tailored to the characteristics and needs of the nodes in that domain. Unlike Federated P2P Overlays, nodes also maintain links to other sub-networks, which allows nodes to detect attacked domains and help recover it.

4.2. Key Components and Their Definitions

It is important to establish clear definitions for the components that comprise a Sovereign Domains P2P Overlay. After all, without a clear understanding of the terminology used, it can be challenging to comprehend and compare future research in this field and to facilitate cooperation among researchers.

As such, below goes an overview with a description of the main components of a Sovereign Domains P2P Overlay:

- **Nodes**: These are the participants of the network that form the overlay. They can be part of (a) single or multiple *domains* and are responsible for maintaining a list of *links* to reachable neighbors in the virtual overlay according to a certain network topology.
- **Links**: These are the connections between nodes in the overlay, which links are formed based on the node routing table entries.
- **Protocols**: These are a set of rules that govern how nodes interact with each other, including node discovery, link establishment, and message routing.
- **Domains**: These are the logical partitions of the network that nodes are grouped into. Each domain can have its own independent set of protocols and behavior that are optimized for their specific requirements, responsibilities and security assumptions.

More formally, we define nodes, links, protocols, and domains as follows:

Definition 1 (Node)**.** *Let V be a set of nodes, D be a set of domains, L be a set of links between two nodes and u be a unique identifier for a node. A node $v \in V$ can be defined as the triplet (u, D', L') with $D' \subset D$ and $L' \subset L$.*

Definition 2 (Link)**.** *Let V be a set of nodes. A link $l \in L$ can be defined by tuple $((u', D', L'), (u'', D'', L''))$, where node $u' \in V$ has node $u'' \in V$ in its routing table.*

Definition 3 (Protocol)**.** *Let P be a set of protocols, where a protocol $p \in P$ is a set of rules that defines how nodes interact with each other in a Sovereign Domains P2P Overlay. These rules may include node discovery, link establishment, message routing, and other aspects of network operation.*

Definition 4 (Domain). *Let V be a set of nodes, D be a set of domains and P be a set of protocols for maintaining the overlay. A domain $d \in D$ is a subset of nodes in V that uses a set of protocols $p \in P$, and is defined by a domain function $h(d)$ such that $h(d) = \{ v \in V | f(v,d) = 1, p_v = p_d \}$, where $f(v,d)$ is a domain membership function indicating whether a node v is part of a domain d.*

It is important to note that the definition of a domain is ultimately determined by the system designer, and should be based on the specific goals and requirements of the P2P overlay being created. In general, a domain should be created when there is a need to cluster nodes that frequently communicate with each other or when certain nodes have specialized roles or requirements that are different from other nodes in the network. For example, a domain could be a parachain in Polkadot, a set of consensus nodes in Ethereum, a pool of solvers collaborating to transform intents into transactions in Anoma, or a shard in RapidChain.

Based on these definitions, we can define Sovereign Domains P2P Overlay as follows:

Definition 5 (Sovereign Domains P2P Overlay). *Let V be a set of nodes, L be a set of links between those nodes, and D be a set of domains. Then, a Sovereign Domains P2P Overlay is the triplet (V, L, D).*

4.3. Notations

An overview of the notations with respect to Sovereign Domains P2P Overlays can be found in Table 4.1.

Notation	Description
V	The set of all nodes $\{v_1, v_2, \ldots, v_{\|V\|}\}$ in the overlay
L	The set of all links $\{l_1, l_2, \ldots, l_{\|L\|}\}$ in the overlay
D	The set of all domains $\{d_1, d_2, \ldots, d_{\|D\|}\}$ in the overlay
h	The fraction of honest nodes in the overlay
f	The fraction of malicious nodes in the overlay
h_{d_i}	The fraction of honest nodes in domain d_i
f_{d_i}	The fraction of malicious nodes in domain d_i
se_{d_i}	The success rate for inter-domain lookups for target nodes in domain d_i
si_{d_i}	The success rate for intra-domain lookups in domain d_i
t	The quality threshold such that domain nodes are sent to domain d_i if s_{d_i} falls below t
x_{d_i}	The fraction of inter-domain lookups initiated by domain nodes from domain d_i
p	The fraction of nodes that can co-exist in multiple domains simultaneously
m	The fraction of domains in which nodes can participate simultaneously

Table 4.1: The most important notations for Sovereign Domains P2P Overlay.

4.4. Discussion

Sovereign Domains P2P Overlay allows for achieving (i) the performance of a small P2P overlay, (ii) the level of security as provided in a large P2P overlay and (iii) optimized topologies and protocols for hetereogeneous groups of nodes. This is further explained below.

Sovereign Domains P2P Overlay architecture allows to achieve the performance of a small P2P overlay, because the domains are smaller in size. Therefore, the dissemination of information within a domain takes place in a manner similar to the dissemination of information in a smaller P2P overlay, which is more efficient than the dissemination of information in a larger P2P overlay [60].

Furthermore, Sovereign Domains P2P Overlay allows to have the security of a large P2P overlay, because nodes can maintain links to nodes from other domains. This has two main benefits. First, having links to other domains increases the connectivity of the network, making it more resilient against overlay attacks, such as DoS attacks. Second, having links to other domains allows to build statistical models about other domains, which can be used to monitor a predefined state of quality. This data can be used to, for example, detect malicious behavior from an adversary that has compromised a significant portion of the domain, and then repair that domain if necessary. In a large P2P overlay, it becomes challenging for nodes to build accurate models of individual nodes. Due to the large number of nodes, it becomes more difficult to collect enough data from a single node to create a reliable statistical model.

Sovereign Domains P2P Overlay also allows to optimize topologies and protocols for heterogeneous groups of nodes. For example, a domain with a high node churn may employ an unstructured internal topology,

whereas a domain with less churn may employ a structured internal topology. Because different domains may provide different services and nodes may be interested in multiple services, our architecture allows nodes to participate in multiple domains simultaneously. That is, the domains are not disjoint in their set of participants.

II

Case Study: Node Discovery in Sovereign Domains P2P Overlays

5

Preliminaries: Kademlia

This chapter provides a brief overview of the Kademlia structure and describes the main algorithms as background information for our novel node discovery protocols for Federated P2P Overlays and Sovereign Domains P2P Overlay.

5.1. Overview

Kademlia [36] is a well-known DHT protocol that has found extensive use in P2P networks, particularly in file sharing [9] and content delivery systems [28]. In recent years, Kademlia has also been increasingly utilized in blockchain systems for designing node discovery protocols (e.g., [57, 62, 11]).

In addition to establishing and maintaining a list of links to neighbors, Kademlia can also store data in a distributed fashion. It includes mechanisms for determining where to store data and how to look it up. However, since the focus of this study is on node discovery, we do not provide details on data storage or retrieval using Kademlia.

In Kademlia, each node is assigned a unique b-bit identifier (usually $b = 128$ or 160). One of the key features of Kademlia is its use of an XOR-based metric to determine the distance between nodes in the network. The XOR metric has several advantages over other distance metrics, such as Euclidean distance, including its ability to preserve the symmetry and triangle inequality properties [28].

5.2. Routing Tables

Nodes maintain a routing table that contains the nodeIDs and IP addresses of neighbors in the overlay topology. The routing tables are constructed such that they have more entries for nodes that are closer in identifier space distance, based on the XOR metric. Specifically, each node maintains a routing table consisting of up to b buckets. Each bucket, known as a *k-bucket* in the original paper [36], contains up to k entries with relevant information about nodes. The relevance of information depends on the application; in blockchain systems, relevant information could be the address information <nodeID, IP address, port number, timestamp> of a node. The parameter k is a redundancy factor that makes the routing more robust by spanning several disjoint paths between overlay nodes [8]. Buckets are arranged as a binary tree, and neighbors are assigned to buckets according to the shortest unique prefix of their nodeIDs.

5.3. Key Algorithms

5.3.1. Joining the Overlay

Once a new node joins a Kademlia network, it must discover other nodes in the network to enable efficient routing of messages. To do this, the node starts by establishing a connection to a *bootstrap node*, which is a known node that can help the new node learn about the rest of the network [36]. Then, the new node initiates a node lookup operation, choosing itself as the target node to discover nodes that are close to itself based on the XOR metric.

5.3.2. Looking Up a Node

Algorithm 1 Initiate Node Lookup in Kademlia

1: $v_s \leftarrow$ source (domain node)
2: $v_t \leftarrow$ target (domain node)
3: $R \leftarrow \emptyset$ (result list)
4: **function** *FindNode(v_s, v_t)*
5: $R \leftarrow$ k-bucket corresponding to the first k bits of v_t in v_s's routing table
6: **if** R has less than k nodes
7: $R' \leftarrow$ k-bucket corresponding to next k bits of v_t
8: Add nodes from R' to R until $|R| = k$
9: **end if**
10: Sort R in ascending order of XOR-distance to v_t
11: **for** node v_i in set $\{v_0, ..., v_{\alpha-1}\}$ in R
12: Send a REQUEST message to v_i with v_t
13: Mark node v_i as visited
14: **end for**
15: **end function**

Algorithm 2 Handle REQUEST Message in Kademlia

1: $v_r \leftarrow$ receiver of message (domain node)
2: $C \leftarrow \emptyset$ (candidates list)
3: **function** *handleRequestMessage(v_s, v_t)*
4: Add sender v_s to corresponding k-bucket in routing table
5: $C \leftarrow$ k-bucket corresponding to the first k bits of v_t in iDHT
6: **if** C has less than β nodes
7: $C' \leftarrow$ k-bucket corresponding to next k bits of v_t
8: Add nodes from C' to C
9: **end if**
10: Sort C in ascending order of XOR-distance to v_t and trim list such that $|C| = \beta$
11: Send a RESPONSE message to v_s with candidates C
12: **end function**

Algorithm 3 Handle RESPONSE Message in Kademlia

1: $v_s \leftarrow$ source (domain node)
2: $v_t \leftarrow$ target (domain node)
3: $R \leftarrow$ closest nodes to target so far (result list)
4: **function** *handleResponseMessage(v_r, C)*
5: Add sender v_r to corresponding k-bucket in routing table
6: Add β nodes from C to R
7: Sort R in ascending order of XOR-distance to v_t and remove β last nodes from R
8: **if** R contains v_t
9: Return (search operation terminated)
10: **end if**
11: **if** there is a node v_i in R that has not been visited
12: Send a REQUEST message to v_i with target v_t
13: Mark node v_i as visited
14: **else**
15: Return (search operation terminated)
16: **end else**
17: **end function**

A source node initiates a lookup operation to find a target node by sending a request message to α nodes (usually $\alpha = 3$) that are closest to the target node according to the local view of the source node (i.e., according to the entries in the routing table of the source node). It contacts nodes to speed up the process of discovering nodes. As we can see in 15, the XOR metric is used to determine which nodes are the closest to the target node.

Upon receiving a request, the receiver node first attempts to add the sender of the request message to its routing table. It does this by calculating the XOR distance to its own nodeID to determine in which k-bucket this node belongs. However, if the k-bucket is full and the sender is not present, then the node sends a ping message to the least recently seen node in this k-bucket. If this least recently node sends a pong message back, then it's timestamp is updated and it becomes the most recently seen node, and the newly discovered node is disregarded. Otherwise, the least recently node is replaced for the newly discovered node. After attempting to add the sender to its routing table, the node replies withβ nodes that are closest to the target node in terms of XOR-distance, based on the entries in the routing table (see 12).

Upon receiving the response message, the source node updates its routing table and sends new request messages to the β nodes it just received. The node asks each of these nodes for any nodes that are closer to the target node, based on their own routing tables (see 17).

This process continues until the source node has discovered the k closest nodes to the target node or when it has received the contact information of the target node. As a side-effect of this lookup operation, nodes that have received a request message now also become aware of this node and can update their routing tables, enabling it to reach more nodes in the network. In this way, the routing tables of nodes in the network gradually become more complete, allowing efficient and effective routing of messages.

5.3.3. Leaving the Overlay

A node in a blockchain system (e.g., Ethereum[1]) that uses Kademlia can leave the P2P overlay in two ways. A node can leave the P2P overlay unannounced or it can announce that it is leaving by sending its neighbors a "goodbye" message. In the first case, nodes will learn that the node has left because the node does not respond in a timely manner. This means that there is a gap between the moment a node has actually left the network and the moment a neighbor has learned about this. This is problematic, since in this time window, a neighbor may try to send the node a message and rely on this stale node to forward information in the blockchain system. When a node sends a goodbye message, this time window is minimal and this allows other nodes to keep their routing table fresh, which makes the network remain stable and efficient as the probability that a stale node is contacted to forward information is minimal.

5.4. Discussion

One of the key advantages of Kademlia is its efficiency. Because the routing table is organized into buckets based on XOR distance, the number of nodes in each bucket grows exponentially with the distance from the local node. This means that the number of nodes that need to be queried to find a particular node grows logarithmically with the size of the network [50]. This is why Kademlia is often referred to as a "logarithmic" DHT.

However, as with many structured P2P overlays, Kademlia is vulnerable for overlay attacks (see subsection 2.1.4). Two of the most severe overlay attacks on Kademlia are eclipse and routing attacks [8]. Eclipse attacks take place when a network node is effectively cut off from the rest of the network by an attacker who has taken control of a sizable part of that node's connections. This can be done in Kademlia by controlling a sizable portion of the nodes that are, in terms of their XOR distance, closest to the targeted node. This can be accomplished by altering the targeted node's routing table or by forging network addresses for other nodes in the network [8]. Once the attacker has eclipsed the node, it can prevent it from receiving or sending messages to other nodes in the network, effectively isolating it. This can be used to disrupt the network or to launch other attacks, such as Sybil attacks or routing attacks. Furthermore, routing attacks occur when an attacker manipulates the routing table of a node. The attacker causes it to route traffic to malicious nodes instead of legitimate nodes. In Kademlia, this can be accomplished by sending false or manipulated requests or response messages to a node, causing it to update its routing table with false information. Once the routing table has been manipulated, the attacker can redirect traffic to nodes under their control, effectively intercepting messages and compromising the security of the network.

[1] https://github.com/ethereum/consensus-specs/blob/dev/specs/phase0/p2p-interface.md

6

Design of Node Discovery Protocols

To explore the effectiveness of Sovereign Domains P2P Overlay, we focus on node discovery. By focusing on one aspect of the overlay architecture, we can provide a more in-depth analysis and comparison with existing solutions. This approach can also help to demonstrate the strengths and weaknesses of our novel architecture and show its potential impact in specific scenarios. Furthermore, we focus on node discovery because this is an aspect of P2P overlays in blockchain systems that can actually be modified in client software [43]. Although the communication aspect is also highly relevant for fast information dissemination in blockchains, it assumes that the overlay is already formed and there is already a working node discovery protocol in place.

6.1. Scope

Sovereign Domains P2P Overlay encompasses a range of overlay types, with each instance based on the specifics of the domains. A blockchain system, for instance, may implement a Sovereign Domains P2P Overlay with multiple domains, where each domain utilizes either a structured or unstructured topology and protocol. Alternatively, certain domains may be structured while others are unstructured. In some cases, even two structured domains may employ different node discovery protocols, depending on the characteristics of the nodes. Consequently, we expect that different blockchain systems will implement different instances of Sovereign Domains P2P Overlay, owing to the inherent variability of blockchain systems.

However, for the purpose of evaluating the effectiveness of our architecture, we limit the domains to either structured or unstructured in this research. Doing so enables for a clearer statistical and sharper analysis of the architecture's effectiveness. If, for example, a node in a structured domain wishes to find a node in an unstructured domain, determining the likelihood of a successful lookup can be challenging, given the influence of different domain structures. Similarly, if two structured domains use different node discovery protocols, calculating the likelihood of a successful lookup becomes similarly complex. Therefore, given the scope of this research, we exclude the evaluation of Sovereign Domains P2P Overlay that feature domains with different node discovery protocols.

In this study, we select a widely used node discovery protocol for single-overlay blockchain systems and adapt it for use in a Federated P2P Overlays architecture and a Sovereign Domains P2P Overlay architecture. It is important to note that we could not find any existing literature on node discovery protocols for blockchain systems using Federated P2P Overlays. Thus, we have designed a Federated P2P Overlay node discovery protocol based on existing research on how cross-chain messaging is intended to function in Polkadot.

6.2. Assumptions

We have made several assumptions in our node discovery protocols for Sovereign Domains P2P Overlay and Federated P2P Overlays to enable for a better comparison.

First, we assume that Kademlia is the base protocol for node discovery in both Sovereign Domains P2P Overlay and Federated P2P Overlays, as it is widely adopted and used in blockchain systems such as Ethereum, Polkadot, and RapidChain. Therefore, we assume that all domains in Sovereign Domains P2P Overlay and independent overlays in Federated P2P Overlays use Kademlia internally.

Second, we assume that the heterogeneous overlays in Federated P2P Overlays architecture are the same as domains in the Sovereign Domains P2P Overlay architecture. However, in practice, these two architectures

differ, as all nodes in all domains refer to nodes in the same underlying network in a Sovereign Domains P2P Overlay architecture, whereas in Federated P2P Overlays, each overlay is mapped to a different underlying network. The drawbacks of this architecture are discussed in chapter 2.

We also assume that the values p and m are the same for both architectures. Although Polkadot does not explicitly state that a single node can be part of multiple parachains, the design does not fundamentally disallow it, as the parachains offer different services. Therefore, we can assume that nodes in Federated P2P Overlays can be part of multiple domains as long as they have the resources to do so.

Finally, we assume that all overlays (i.e., domains) in Federated P2P Overlays have no mechanism to recover from domain failure. This means that even if nodes detect that a certain domain is under attack, they cannot change their behavior based on this information. This is in line with a recent study by Kai Mast [35], which shows that blockchain systems with a Federated P2P Overlays architecture have no mechanism to recover from subnetwork (or zone) failures. We emphasize that domain recovery is possible in Sovereign Domains P2P Overlay architecture.

6.3. Requirements

In order to achieve the goals outlined in section 3.3, our node discovery protocol must meet the following requirements:

1. *Connectivity*: The overlay G must be connected so that there is a path between any pair of honest nodes in V with domains, but also across domains. An overlay that is not connected implies that some parts of the overlay cannot send or receive information. This also means that nodes that join the network must be able to quickly build a local view and announce themselves to neighbors in order to be reachable.

2. *High accuracy*: Nodes must have an accurate local view and unavailable nodes (i.e., nodes that left the overlay) must be removed to prevent them from being selected to forward information. We measure the accuracy of a graph as the average accuracy of all correct nodes, which is the fraction of reachable nodes in a routing table.

3. *Low average routing path length*: The path length between any two random nodes in V must be low. This property has a major impact on the performance of the overlay as it affects the time it takes to propagate information in the blockchain. We measure the average path length as the number of hops (i.e., nodes) that an honest source node must contact in order to reach the target node.

4. *Low bandwidth cost:* It is important that the bandwidth cost of using the system is low. If the protocol requires a lot of bandwidth, nodes may not join the network because the cost of participation is too high [43]. As our node discovery protocols also involve nodes maintaining links to nodes in other domains, we aim to ensure that the total overlay's bandwidth cost for a node is no more than twice that of participating in a single domain.

5. *High resilience:* The overlay must be highly resilient to network churn and ensure that it continues to function even in the face of compromised nodes. This requires robustness in the protocol's design.

6. *Protection against compromised domains*: Nodes in the overlay must be protected against nodes from compromised domains. A domain is considered compromised when a majority of its nodes are controlled by an adversary. Once an incident occurs in one domain, it may not immediately affect performance and security of other domains.

6.4. FedKad: Node Discovery Protocol for Federated P2P Overlays

6.4.1. Overview

The core idea of our Kademlia-inspired node discovery protocol for Federated P2P Overlays is to have one specific domain that enables cross-domain communication. This is inspired by the architecture of Polkadot, where messages between parachains are forwarded through the relay chain [11]. Available research on Federated P2P Overlays [30, 23, 10] have a similar architecture. Intra-domain communication takes place within domains, whereas inter-domain communication is routed through these so-called *gateway domains*[1]. We refer to nodes in the gateway domains as gateway nodes (see Figure 6.1).

[1] Referenced papers refer to this as *super overlay* or *gateway overlay*

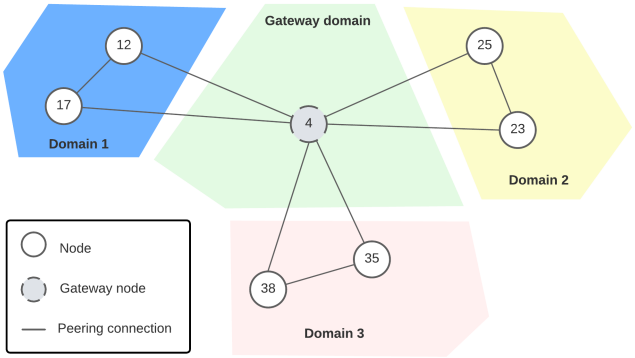


Figure 6.1: An example of FedKad overlay with one gateway node and six other nodes. In this example, a peering connection means that two nodes have each other in their routing tables. All inter-domain communication goes through the gateway node.

6.4.2. Routing Tables

In FedKad, non-gateway nodes maintain two separate routing tables. The first routing table, *iDHT*, is used to establish and maintain links to nodes that are in the same domain. Nodes maintain their iDHT according to the Kademlia protocol described in chapter 5. The second routing table *xDHT* is used to establish and maintain links to gateway nodes. Gateway nodes, on the other hand, maintain a routing table for each domain as they should be able to forward information from one domain to any other domain. Because gateway nodes are only tasked with forwarding information, i.e, they do not perform node lookups, they can have many routing tables.

6.4.3. Key Algorithms

Joining the Overlay

When a node in FedKad joins the Federated P2P Overlays, it joins a domain. As mentioned before, the specifics of the mechanism which domain to join is beyond the scope of this work. In most protocols, new nodes are expected to know at least one participant in that domain, a bootstrap node. This node can be programmed in the client application or a similar effect can be achieved via social-based primitives. The exact bootstrapping strategy depends on the internal rules of that domain.

Once the node in FedKad has established a link to the bootstrap node, it will follow the Kademlia procedure in order to establish a list of reachable neighbors in its domain. However, the bootstrap node in FedKad also sends a new node a list of gateway nodes, thus enabling the node to discover nodes from other domains.

Looking Up a Node

In FedKad, there are two types of node lookups. The first type, *intra-domain node lookups*, are standard Kademlia lookups for looking up a node that is in the same domain.

In the second type, *inter-domain node lookup*, a source node tries to discover the address information of a target node from a different domain. The source node initiates the lookup operation by sending the lookup request to a randomly selected gateway node (see 7). We do not map gateway nodes to specific domains as this will make it easier for an attacker to perform, for example, DoS attacks to make nodes in certain domains unavailable for nodes in the rest of the overlay.

When a gateway node receives a lookup request, the gateway node tries to update its routing table according to the Kademlia protocol. The gateway node then checks whether it has the target node in its routing table. If it does, then the gateway node sends a response message to the source node with the contact information of the target node. However, if it does not have the target node in its routing table, it forwards the request message to a node in its routing table that is closest to the target node. This algorithm is described in 12.

When a node receives a request message from a gateway node, it initiates a normal Kademlia intra-domain node lookup. When the lookup operation terminates, the final result is sent back to the gateway

Algorithm 4 Initiate an Inter-Domain Node Lookup in FedKad

```
1: v_s ← source node
2: v_t ← target node
3: B ← gateway nodes
4: function FindNode(v_s, v_t)
5:    b_i ← randomly selected gateway node from B
6:    Send a REQUEST message to b_i with target t
7: end function
```

Algorithm 5 Gateway Node Handles REQUEST Message in FedKad

```
1: v_s ← source node
2: v_t ← target node
3: B ← gateway nodes
4: function HandleFindNodeMessage(v_s, v_t)
5:    Add sender v_s to corresponding k-bucket in xDHT for domain of v_s
6:    if v_t in xDHT of target domain
7:       Send a RESPONSE message to v_s with target v_t
8:    else
9:       v_r ← closest node to target v_t
10:      Send a REQUEST message to v_r with target v_t
11:   end else
12: end function
```

node it received the request from, after which the gateway node simply forwards the final result back to the source node.

Leaving the Overlay
Nodes leave the P2P overlay in a similar way as Kademlia. However, in FedKad, when a node leaves a domain but still remains in one or more other domains, it sends a goodbye message to nodes in the domain it leaves.

6.4.4. Discussion

FedKad has both advantages and disadvantages from a performance and security perspective, which we will explore now.

From a performance perspective, we notice that for intra-domain lookups, the average routing path length is most likely decreased compared to a lookup in single P2P overlay of size $|V|$. Because nodes use Kademlia for intra-domain node lookups, a single large overlay would require in a worst case scenario $O(log(|V|))$ hops to find a node. In FedKad, an intra-domain lookup requires $O(log\left(\frac{|V|-|B|}{|D|}\right))$ hops, which is less for any $|D| > 1$ and for any $|B| > 1$. For inter-domain lookups, FedKad requires that we contact the gateway node first. This leads to an overhead C to our worst case complexity, resulting in $O(log\left(\frac{|V|-|B|}{|D|}\right) + C)$. If $|V|$ is small, this overhead could have a negative impact on the performance. However, for large size networks, this will most likely not be a problem.

On a more critical note, FedKad has an important (potential) performance limitation. The protocol heavily relies on the availability of gateway nodes. High churn among gateway nodes can be costly as nodes in each domain must become aware of the new gateway nodes. However, a small portion of nodes in Bitcoin [24] and Ethereum [57] remain available for over a month, which would make these nodes good candidates to become gateway nodes. In Polkadot, nodes in the relay chain coordinate cross-chain transactions [11], which also makes them suitable for forwarding node lookup requests. However, the nodes in the relay chain are occupied with intensive operations, and additional node discovery burden may not be desirable. Thus, a better approach would be to directly forward lookup requests to nodes from the target domain.

From a security perspective, the main benefit of FedKad is that honest domain nodes in one domain are protected against malicious domain nodes from another domain. That is, the probability that an intra-domain lookup succeeds in one domain is independent of whether an intra-domain lookup succeeds in another domain. Since domain nodes have a clear separation of routing tables for gateway nodes and other nodes, pollution of gateway nodes does not influence the pollution of the routing table for nodes within the

domain.

However, next to an important (potential) performance limitation, Fedkad has also a major security vulnerability. Attacked domains could remain undetected if malicious nodes decide to perform incoming inter-domain lookups normally. This is a likely scenario, since we assumed an adaptive adversary. For example, the adversary could position all colluder nodes in a single domain (i.e., $f_{d_i} = f$) and only fail intra-domain lookups. As a result, the honest fraction in domain d_i remains

$$h_{d_i} = \frac{h * |d_i|}{h * |d_i| + f * |V|} \tag{6.1}$$

$$|d_i| = \frac{|V| - f * |V|}{|D|} + f * |V| \tag{6.2}$$

This is problematic as increase of $|D|$ in Equation 6.2 decreases $|d_i|$, which translates to a decrease of honest fraction h_{d_i} in Equation 6.1. This means that the fraction of honest nodes renders to $O\left(\frac{1}{|D|}\right)$, which approaches 0 for a large |D|.

6.5. SovKad: Node Discovery Protocol for Sovereign Domains P2P Overlay

6.5.1. Overview

SovKad is a Kademlia-inspired node discovery protocol for Sovereign Domains P2P Overlays. The core idea of SovKad is to allow nodes to maintain links directly to a small group of nodes from other domains, in addition to links to nodes from their own domain (see Figure 6.2). This is not only beneficial from a performance perspective as a lookup request does not have to be routed through gateway nodes, it also allows nodes to build statistical models about other domains and help repair a domain if it is under attack. In SovKad, honest nodes join an attacked domain to help recover the fraction of honest nodes in that domain and thereby also help restore the performance in the domain.

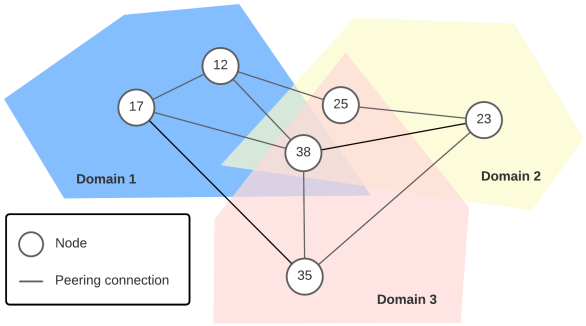


Figure 6.2: An example of SovKad overlay with six nodes. In this example, a peering connection means that two nodes have each other in their routing tables. Nodes can directly communicate with nodes from other domains without routing information through a gateway domain first.

6.5.2. Routing Tables

In SovKad, nodes maintain a total of $|D|$ routing tables. More specifically, SovKad nodes maintain two types of routing tables. The main routing table, known as iDHT, is used to establish and maintain links to nodes within the same domain, similar to FedKad and Kademlia. In addition, nodes also maintain smaller routing tables, referred to as xDHTs, to connect with nodes from other domains. In SovKad's current design, each xDHT has a maximum of 40 entries, equivalent to 2 k-buckets in iDHTs. The reason for this design choice is that maintaining routing tables requires communication, which incurs bandwidth costs for nodes. We have previously established that nodes' bandwidth costs can be at most twice that of participating in a single overlay (as described in section 6.3). Therefore, we assume that the total number of k-buckets per node can be at most twice of that of an iDHT. Since nodes in FedKad and SovKad have 128-bit nodeIDs, their iDHTs

have at most 128 k-buckets. Consequently, nodes are allowed to have an additional 128 k-buckets for all other domains. By assigning two k-buckets to each domain, nodes can handle up to 68 domains, which is close to the number of shards in Ethereum, which is 64 [6].

6.5.3. Key Algorithms

Joining the Overlay

SovKads algorithm for joining a domain is similar to that of FedKad, with one key difference. Because in SovKad there is not a special gateway domain for routing messages across domains, the bootstrap node provides a new node with a list of nodes from other domains, instead of a list of gateway nodes.

Looking Up a Node

Similar to FedKad, intra-domain node lookups in SovKad are standard Kademlia node lookups.

Inter-domain lookups, on the other hand, are somewhat different in SovKad. Although nodes have direct links to nodes from other domains, they route the message to a node from the target domain instead of performing the lookup themselves. There are two main reasons for this approach. First, because nodes have more entries in their iDHT than in their inter-domain hash table xDHT, performing a Kademlia intra-domain node lookup is generally more efficient than performing an inter-domain lookup in a domain where a source node has limited view of. Second, this design decision enables monitoring the domain quality and therefore also helps to repair domains that are under attack. When a domain is under attack, meaning the majority of nodes in that domain are malicious, an adversary may try to deceive nodes that perform inter-domain lookups in that domain. For example, if a node were to perform a lookup in a different domain, the adversary could behave honestly for receiving inter-domain lookups, whereas it would behave dishonestly during intra-domain lookups. This would mean that the quality of the domain is low, but appears to be high. This makes it difficult for honest nodes to detect an attacked domain and help repair it.

When a source node wants to lookup a target from a different domain, it forwards the lookup request to α nodes in the target domain (see 9). By sending it to different nodes, we increase the likelihood that an honest node receives the request and fails the lookup if the domain is truly under attack, because malicious nodes cannot distinguish an inter-domain lookup from an intra-domain lookup.

Algorithm 6 Initiate an Inter-Domain Node Lookup in SovKad

1: $v_s \leftarrow$ source (domain node)
2: $v_t \leftarrow$ target (domain node)
3: $xDHT \leftarrow$ list of domain nodes from target domain
4: **function** *FindNode*(v_s, v_t)
5: **for** node v_i in set $v_0, ..., v_{\alpha-1}$ in R
6: Send a FINDNODE message to v_i with target v_t
7: Mark node v_i as visited
8: **end for**
9: **end function**

Upon receiving an inter-domain lookup request, a SovKad node starts performing an intra-domain (Kademlia) node lookup. When the lookup terminates, it sends the results back to the source node. When the source receives all the results from α nodes, it updates the success rate for the target domain. We select the success rate as a measure to detect the quality of a domain, because it is a universal metric with a binary outcome. Other metrics, such as latency, are not suitable, because they are influenced too much by the internal node discovery and topology of a domain.

If the updated success rate falls below quality threshold t, the source node leaves its current domain and joins the attacked domain. But before leaving its current domain, the node creates an xDHT for its current domain and adds 40 randomly selected nodes from its iDHT. Additionally, it sends a "joined attacked domain" message to its neighbors, letting them know to remove source node from their iDHT and add it to their xDHT of the attacked domain.

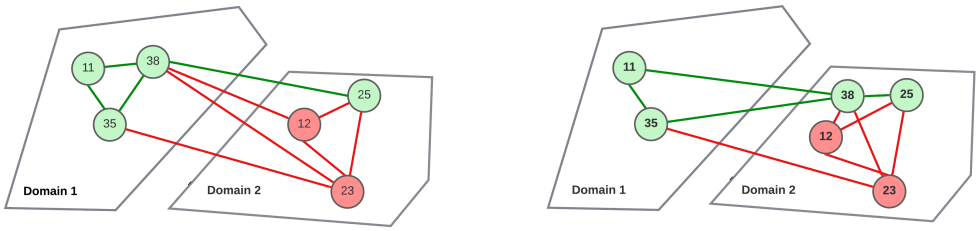
Leaving the Overlay

SovKad nodes leave their domains or the overlay in a similar fashion to FedKad nodes. However, unlike FedKad nodes, SovKad nodes can also leave a domain and send a "joined attacked domain" message. When a SovKad node receives a "goodbye" message, it removes the sender of this message from its iDHT. However, if

a SovKad node receives a "joined attacked domain" message, it removes the node from its iDHT and adds it to the xDHT that contains nodes from the attacked domain.

6.5.4. Discussion

SovKad has two major advantages compared to FedKad. First, SovKad resolves the single point of failure component of FedKad by allowing nodes to establish links to nodes from other domains. Second, it allows to build statistical models for groups of nodes. These models can be used to detect whether a domain is under attack, and help repair it. Nodes in SovKad help repair an attacked domain by joining that domain and letting its current neighbors know about this decision. By allowing them to update their routing tables, the fraction of honest links (i.e., links between two honest nodes) is increased (see Figure 6.3). This makes the attacked domain more available for honest nodes in that attacked domain, as well as for nodes that are not part of that domain.



(a) Before node 38 moved to attacked domain 2.

(b) After node 38 moved to attacked domain 2.

Figure 6.3: In this an example, we see how the proportion of honest and malicious peering connections can change when an honest node moves to an attacked domain. In this figure, green nodes represent honest nodes, while red nodes represent malicious nodes. A peering connection is classified as malicious (red) if it involves at least one malicious node. Otherwise, it is considered honest (green). When an honest node moves to an attacked domain, it can contribute to an increase in the proportion of honest intra-domain peering connections in that domain, while also increasing the proportion of inter-domain peering connections that are honest.

7

Evaluation

The main objective of our experiments is to assess the effectiveness of the Sovereign Domains P2P Overlay architecture in comparison to the structured single overlay architecture and Federated P2P Overlays architecture. We do this by comparing the performance of SovKad, FedKad, and Kademlia for each of the three respective architectures. To simulate the three node discovery protocols, we use an open-source custom-built simulation tool designed specifically for P2P overlay architectures that split the network into multiple heterogeneous sub-networks.

In this chapter, we describe the methodology used in our experiments. We explain the simulation scenarios we have tested, which are relevant in the context of blockchain systems, and the relevant metrics we have used to evaluate them. Then, we present and discuss the results in detail.

7.1. Methodology

We choose to run simulations instead of conducting experiments on a real network, because it is less expensive to run experiments on a single computer where nodes communicate using shared memory. Furthermore, simulations allow us to more easily monitor, analyze, and address unexpected behavior. Additionally, we argue that we can achieve results that closely resemble real-world network behavior by adjusting the simulator settings, such as the reliability and latency of message transport, to match real-world measurements. For instance, based on publicly available data[1], we assume a reliable network where messages are not dropped and have a round-trip message latency of 200 milliseconds.

In the simulation, we use a custom-built tool specifically designed for P2P overlay architectures, which splits the network into multiple heterogeneous sub-networks. We measure various metrics, such as routing efficiency and network stability, to evaluate the performance of each architecture. We also vary the network size, the number of nodes per domain, and network churn to assess the scalability and robustness of each architecture.

By conducting our experiments in this way, we aim to provide a comprehensive evaluation of the three P2P overlay architectures and their node discovery protocols in a controlled environment, allowing us to draw meaningful conclusions about their effectiveness.

7.1.1. Evaluation Scenarios

Happy Path

First, we evaluate Kademlia, FedKad and SovKad in an ideal setting where all participants are online and follow the described algorithms. While this scenario may not fully reflect real-world conditions, it does allow us to reason about each protocol's ability to fulfills its primary requirements, and also identify unexpected outcomes or bugs. Additionally, the results of this initial evaluation can serve as a baseline when evaluating the fault tolerance of the protocols in later experiments.

Network Churn

Churn is a major factor that significantly impacts the stability and performance of a blockchain system and must therefore be examined. A large number of nodes can join or leave a P2P network at any time, and studies

[1] https://www.verizon.com/business/terms/latency/

have shown that churn can have a significant effect on the blockchain's overall performance. For instance, high churn rates in Bitcoin have been associated with a 135% increase in block propagation time[24], which can make the blockchain more vulnerable to forking or double-spending attacks. It is therefore essential that any node discovery protocol has a mechanism that detects and removes stale nodes from its routing table to prevent offline nodes from being selected to forward information.

Byzantine Behavior

Malicious nodes have a significant negative impact on the performance of any P2P overlay, as they deliberately try to disrupt the network. The extent of the damage caused by malicious nodes varies depending on their proportion of the network and their position in the overlay [48]. The larger the fraction of malicious nodes, the more control they have over the network and the more damage they can do. In blockchain systems, malicious nodes attempt to slow down the propagation of transactions or blocks, although in most cases their impact is limited by the assumption that the fraction of malicious nodes is less than $\frac{1}{3}$ of the total network size (see 3.2).

However, it has been shown that even with limited malicious node power, adversaries can still cause damage to the network by strategically positioning themselves in the overlay. For example, in FedKad and SovKad, the adversary can gather in a single domain, reducing the number of honest nodes there. If they dominate the domain, they can decrease the performance of that domain.

This raises important questions about how different fractions and distributions of malicious nodes affect the performance of intra- and inter-domain node lookups, which is not well understood yet. Therefore, this work conducts experiments to gain a better understanding of how different fractions and distributions of malicious nodes affect the performance of FedKad and SovKad, and how effective SovKad's detection and reaction mechanism is.

7.1.2. Evaluation Metrics

In order to evaluate our node discovery protocols, we need to have a set of metrics that is relevant for our problem. That is, we need to have metrics that allow us to compare whether FedKad and SovKad have met their requirements. We use the following metrics:

- **Success rate:** The success rate is the fraction of successful lookups in all lookups and it indirectly measures the *connectivity* of the overlay [31]. We consider a lookup successful if the source node is provided with contact information of the target node before the lookup terminates. Low success rate typically means that source nodes cannot find the target nodes, because at least one node along the path towards the target node is unreliable. A high failure rate may indicate that a node is surrounded by unreliable nodes and that this victim node cannot receive or send information to nodes outside its local view. Therefore, a low success ratio indicates that transactions and blocks will not be disseminated effectively. High node lookup success ratio is positive.

- **Fraction of reachable nodes in routing tables:** We measure the *accuracy* of the local view by periodically computing the fraction of reachable nodes in the in a routing table, and then average over all routing tables. High reachability fraction is positive.

- **Average routing path length:** We measure the *average length of the path* taken from the source node to the target node by calculating the number of nodes that were contacted along the path. For example, if source node v_1 wants to find target node v_3 and contacts relayer node v_2 which has node v_3 in its routing table, we say that the required number of hops for this node lookup is 1. If the source node already has the target node in its routing table, we say that this node lookup requires 0 hops. To ensure the efficiency of the information dissemination, it is important to keep the average path length low, as this value is directly related to the time it will take for information to reach all nodes. Therefore, low hop count is positive.

- **Message count:** We measure the *bandwidth costs* to find a node by calculating the total number of messages that were sent as part of a single lookup procedure. The message count gives us an estimation of the bandwidth cost of our protocol. High bandwidth cost makes it more expensive to participate in the overlay and can also lead to higher latencies or, even worse, network congestion. Low message count is positive.

7.1.3. Experiment Approach

A few steps were taken to conduct the experiments. Initially, we set up and configured the simulation tool by installing the necessary software and configuring the parameters in the configuration file. Then, we defined various simulation scenarios and executed them on our simulator. This included varying the network size, the number of domains and the fraction and position of malicious nodes in the network. Finally, we analyzed the results of the simulations and created visualizations to aid in understanding the findings.

Step 1: Set Up Simulator

The simulator was set up as follows. Our simulator was installed on the computer that runs the simulations by cloning the project from our Github repository[2]. For network sizes below 5k nodes, a laptop with 8 cores that runs on Linux was used. For larger simulations (more than 5k nodes), we used a High Performance Computer (HPC) called DAS-5 [7]. This is a cluster of 68 machines with dual 8-core CPUs connected with InfiniBand FDR links. Programs are started using the SLURM batch queuing system [61]. After the simulator was installed on the computer, we set the simulation conditions in the configuration file.

Step 2: Set Experiment Settings

We conducted experiments with various network sizes, number of domains and fractions of malicious nodes. It is important to evaluate FedKad and SovKad for different networks, because the performance of any node discovery protocol can vary significantly depending on the size of the network. Therefore, we simulated our protocols for $|V| = \{0.1k, 1k, 2k, 4k, 8k, 16k\}$. We also varied the number of domains and the fractions of malicious nodes to learn more about their influence on the performance. For our final evaluations, we tested our protocols for $|D| = \{1, 2, 4, 6, 8\}$ and for $f = \{0.1, 0.2, 0.3\}$. Note that inter-domain lookups require the existence of at least 2 domains and that intra-domain lookups for a network with one domain is equivalent to the standard Kademlia protocol.

We set the simulated time to 24h, in line with existing literature on peer-to-peer protocols in blockchains [49] to allow for cross-study comparison. The actual CPU time was more affected by the number of find operations than the simulated time. The number of find operations was adjusted to scale with the size of the network and simulated time. This is to ensure that all nodes perform an equal number of node lookups. On average, every node initiated one node lookup per minute.

The probability that a search operation is an intra-domain lookup was adjusted to the number of domains in the overlay. We base the fraction of inter-domain node lookups on a recent study [16] that measured the fraction of transactions that are across different shards in Ethereum. Given our previous statements that a shard could be a domain in our architecture, we use a similar fraction of node lookup as [16] use for cross-shard transactions. More concretely, the fraction of inter-domain lookups in our experiments are 5%, 7,5%, 10% and 15% for 2, 4, 6, 8 domains respectively. The final results will also be weighted with these numbers when, for example, visualizing the average path length in SovKad.

The churn rate, or the probability that a node will leave or join the network, was set to 0.5. Additionally, the probability that a node leaves the network was also set to 0.5, meaning that at the end of the experiment, 25% of the nodes left and 25% of new nodes joined the network. It is hard to provide a specific churn rate that is representative for (federated) peer-to-peer overlays in blockchain systems as there are various factors that influence how long nodes remain online. On the one hand, nodes can remain online for a long period when there is an incentive mechanism in place for not leaving the network. For example, nodes are more likely to remain online if they can earn monetary rewards for participating in the peer-to-peer overlay. Another factor is the role of a node and the responsibilities that come with it. For example, some blockchain systems rely on validators (or full nodes) to decide on the next block. In Proof-of-Stake consensus protocols, nodes need to stake their assets and can even get slashed when they leave before an agreed-upon time. On the other hand, a blockchain system can also have nodes that are more likely to join and leave unexpectedly. For instance, some blockchain systems allow nodes to retrieve only necessary information from other nodes instead of maintaining a full copy of the blockchain (light clients). In Sovereign Domains P2P Overlay, we expect that homogeneous nodes will congregate in the same domain. This means that some domains will have a very low churn rate and other domains will have higher churn rates. In our experiments, we set the churn rate in all domains to 0.5, which is in line with recent literature on churn in Bitcoin's P2P network [24].

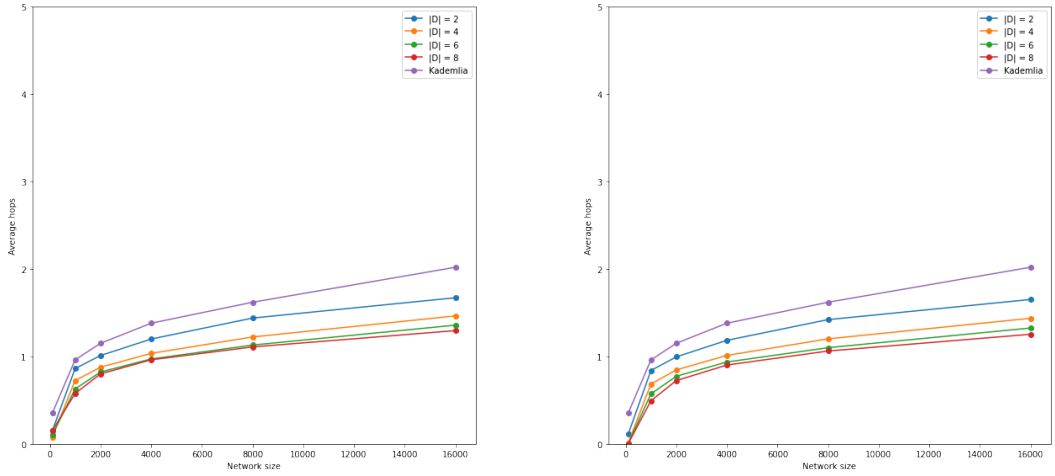
We assumed that 10% of the total network would consist of gateway nodes. This is equivalent to saying that $p = 0.1$ and $m = |D|$. Given that running a node in a blockchain system is resource-intensive, we expect that only a small portion of the network can handle the level of workload of participating in $|D|$ domains.

[2] https://github.com/nztud/thesistud

7.2. Results and Discussions

Our simulation results have revealed several interesting properties of FedKad and SovKad. In this section, we will examine the three scenarios we previously described, one by one.

7.2.1. Happy Path



(a) FedKad (happy path).

(b) SovKad (happy path).

Figure 7.1: The average path lengths in FedKad and in Sovkad are shorter than in Kademlia. In the happy path, FedKad and SovKad achieve somewhat similar performance.

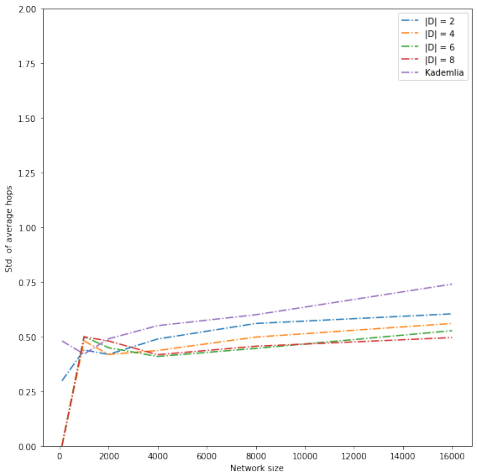
The success rate of node lookups in our happy path is 1.0 for all scenarios. In other words, node lookups in Kademlia, FedKad and SovKad are equally successful, and the source node is always able to find any random target node for various network sizes and number of domains when there is no churn and all nodes are honest.

Furthermore, FedKad and SovKad outperform Kademlia in terms of average routing path length (see Figure 7.1). The results for FedKad and Sovkad were very much alike, which is not very surprising since in the happy path the algorithms show much similarity. For example, in both cases, an inter-domain lookup becomes an intra-domain lookup and in both cases the source node does not perform such an lookup itself. One thing that we notice from Figure 7.1 is that for a very small network size (i.e., 100 nodes), the average routing path length in SovKad is lower than in FedKad. This is because inter-domain lookups in SovKad require at least one hop. In Figure 7.2, we see that variability in routing path length follows a similar pattern as the average routing path length. Also, we see that the variability of hops in FedKad and SovKad is lower than in Kademlia, indicating that the paths in Federated P2P Overlays and in Sovereign Domains P2P Overlay are more consistent in length, and that such topologies can provide more optimal paths for all nodes rather than for some.

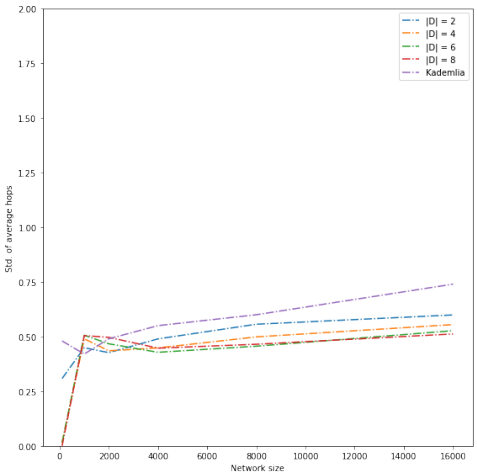
Finally, we can see in Figure 7.3 and Figure 7.4 the average and variable bandwidth cost of Kademlia, FedKad, and SovKad under different network sizes and different $|D|$. We can observe that an increase in the number of nodes and domains leads to a higher average bandwidth cost for all three overlay networks. This increase in cost is due to longer paths that require more nodes to be contacted in order to complete the lookup operation. Specifically, longer paths are associated with an increased hop count, which in turn requires more messages to be exchanged between nodes. Thus, longer paths result in a higher number of messages exchanged and a greater average bandwidth cost. Likewise, the variability in bandwidth costs exhibits a similar pattern to that of the routing path length variability.

7.2.2. Network Churn

Our simulations indicate that Kademlia, FedKad and SovKad perform very well during network churn. In contrast to the findings in [38], our results suggest that nodes in Kademlia-based protocols are always able to find the target node for churn rate of 0.5. Factors that contribute to this mismatch in results are as follows. First, we set the bucket-size k to 20, whereas the referenced study set k to 8. Larger routing tables allow nodes to have a more global view of their domain and are therefore more likely to select a candidate that is close

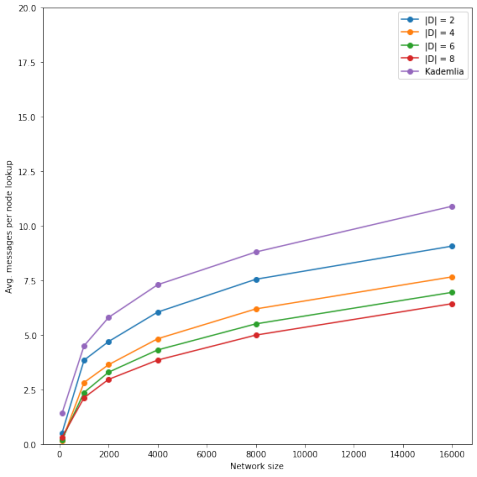


(a) FedKad (happy path).

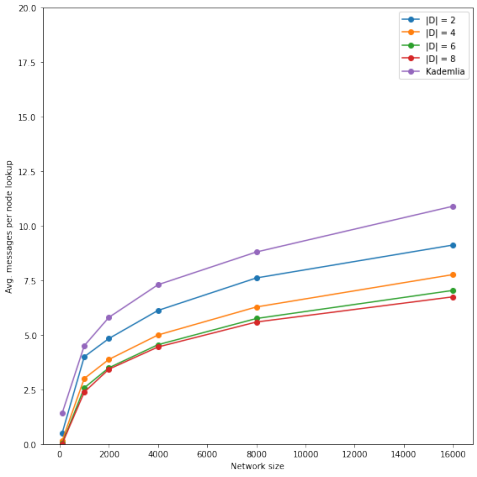


(b) SovKad (happy path).

Figure 7.2: The variability in average routing path lengths in FedKad and in Sovkad in terms of standard deviation. The variability in FedKad and SovKad is smaller than in Kademlia. The variability of the two novel node discovery protocols, however, are similar.

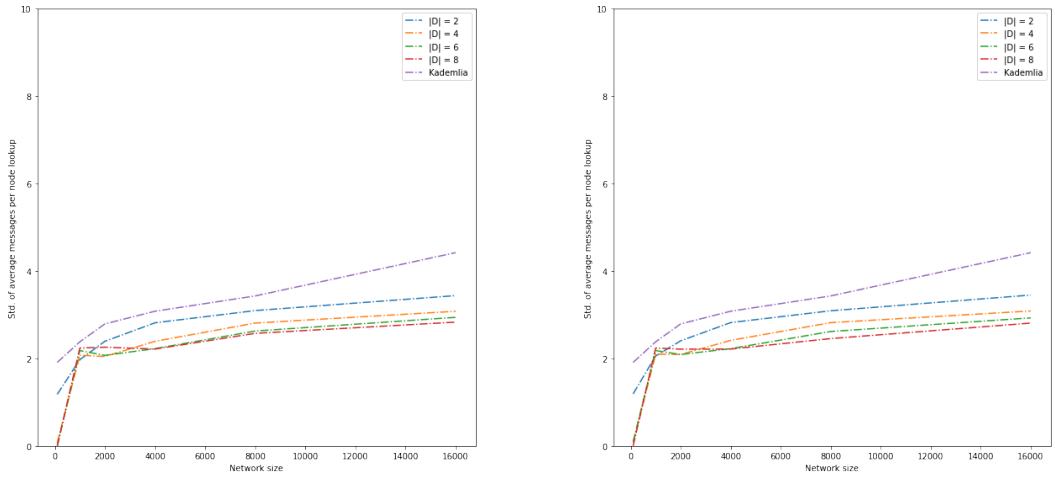


(a) FedKad (happy path).



(b) SovKad (happy path).

Figure 7.3: The average bandwidth of node lookups in terms of total messages sent per lookup operation in FedKad and SovKad. Both protocols show a similar pattern as the average routing paths, because the longer the path, the more nodes are contacted during the lookup operation.



(a) FedKad (happy path).

(b) SovKad (happy path).

Figure 7.4: Variability of average bandwidth cost of node lookups in FedKad and SovKad.

		Avg. reachability rate when there is network churn	
		Lookups/node/minute = 1	Lookups/node/minute = 10
Kademlia	DHT	0.90	0.93
FedKad	iDHT	0.90	0.94
	xDHT	0.90	0.93
SovKad	iDHT	0.90	0.94
	xDHT	0.89	0.93

Table 7.1: The fraction of nodes per routing tables in Kademlia, FedKad and SovKad that are not stale. Because FedKad and SovKad have the same routing table maintenance mechanism as Kademlia, they inherit its performance.

to the target node instantly. A second explanation could be that the authors of [38] select the target node differently. For example, if we allow the target node to go offline before or during a node lookup, the lookup will always fail, resulting in a lower success rate. Finally, it is unclear whether the referenced study retries another node after a timeout event occurs, because the receiver of a "request" message is offline.

Furthermore, FedKad and SovKad have a similar average reachability rate as Kademlia does. This is not surprising, as both protocols use Kademlia for intra-domain lookups and have a similar protocol for updating routing tables during inter-domain lookups. Because nodes in Kademlia update their routing table as a side-effect of a lookup operation, we can see that increasing the number of lookups per node per minute also increases the reachability rate (see Table 7.1).

Finally, our results show that the average routing paths and the average bandwidth costs in the happy case and churn case are almost identical. This is desirable, as it demonstrates SovKad's ability to adapt to changes in the network topology, ensuring efficient communication between nodes. This is particularly crucial for our node discovery protocols, since its aim is to facilitate effective communication, not only between nodes in a single domain, but also between nodes from different domains. By maintaining similar routing paths and bandwidth costs in both scenarios, the protocol can effectively support the communication needs of the nodes, despite changes in the conditions of the network.

7.2.3. Byzantine Behavior

FedKad and SovKad are interesting in that malicious nodes can congregate their power in a single domain and disrupt activities in that domain. In this section, we analyse this type of attack on FedKad and SovKad.

In Figure 7.5, we see how the performance of FedKad and SovKad are affected when the attacker is uniformly distributed over all the domains. We learn that when the fraction of byzantine nodes in every domain is the same ($\frac{1}{3}$ in our set-up), nodes in FedKad and SovKad are capable of successfully finding the target node. In this case, an increase in $|D|$ improves the success rate of node lookups, despite the fact that a third of that domain is controlled by the adversary. However, when the attacker decides to congregate all of its power in a single domain, the succes rate of FedKad drops drastically (see Figure 7.6a). One interesting observation is

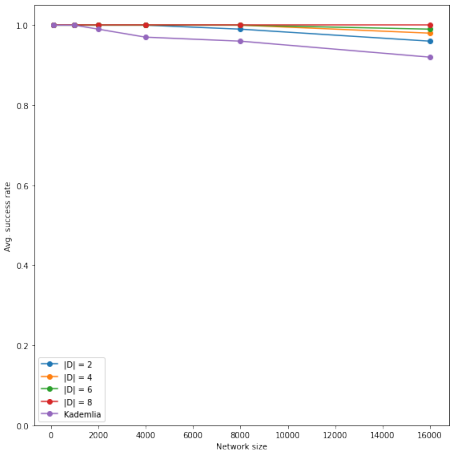
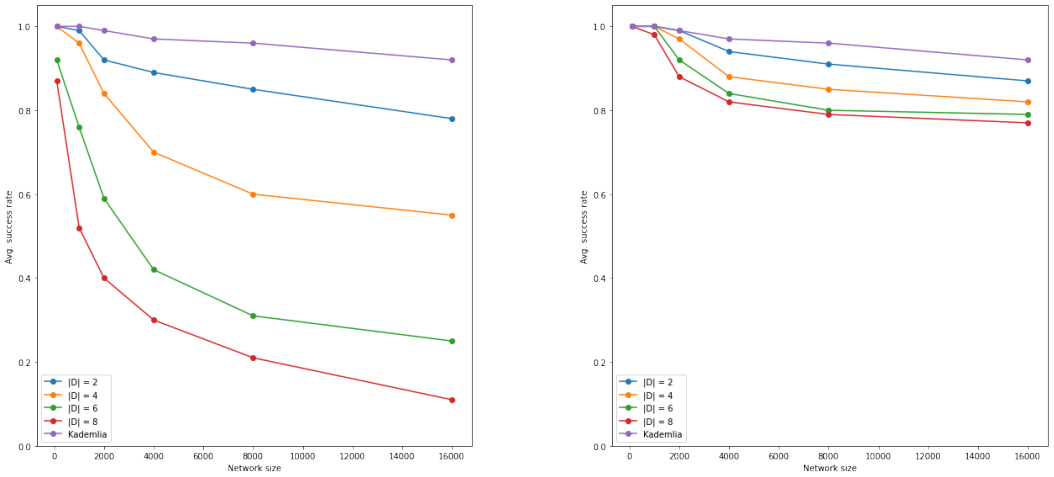


Figure 7.5: Avg. success rate if malicious nodes are spread evenly across domains in FedKad and SovKad. Because the results for FedKad and SovKad did not differ significantly, we averaged them out in this figure.

that the success rate goes up as $|D|$ increases in Figure 7.5, whereas an increase in $|D|$ decreases the success rate in Figure 7.6a. This can be explained as follows. When the adversary congregates all of its power in a single domain d_i, the smaller the size of d_i the higher f_{d_i}. This leads to a decrease in success rate of intra- and inter-domain node lookups.



(a) Avg. success rate if malicious nodes are congregated in a single domain in FedKad.

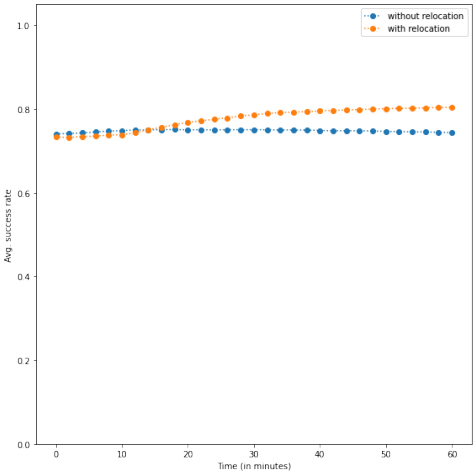
(b) Avg. success rate if malicious nodes are congregated in a single domain in SovKad.

Figure 7.6: hello

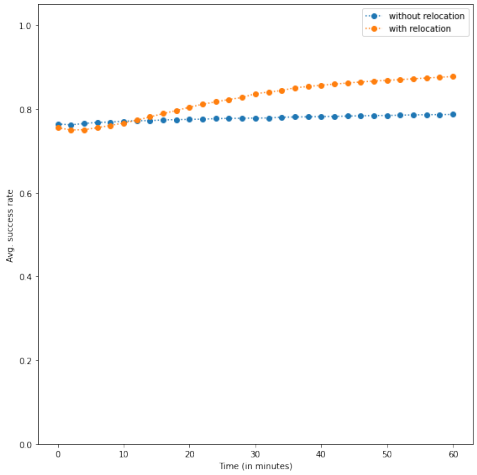
In Figure 7.6b, we observe that SovKad is more resilient against an adversary that concentrates all its power in a single domain. While the pattern of a decrease in success rate with an increase in $|D|$ is still evident, the success rate is considerably higher than that of FedKad. This shows that our security mechanism is effective in detecting and repairing attacked domains, allowing honest nodes in the same domain to establish new links with newly arrived nodes, thereby improving the success rate of intra-domain lookup. Additionally, it enables nodes from other domains to establish links with honest nodes in the attacked domain, thereby improving the success rate of incoming inter-domain lookup.

In Figure 7.7, we see an example of how the success rate of an attacked domain evolves over time. We see that for our simulation set-up, nodes detect the attacked domain after approximately 10 minutes. When honest nodes start joining the attacked domain, we see that the success rate increases fast in the beginning, but starts to settle around 0.8. This is because we set the domain quality threshold t to 0.8.

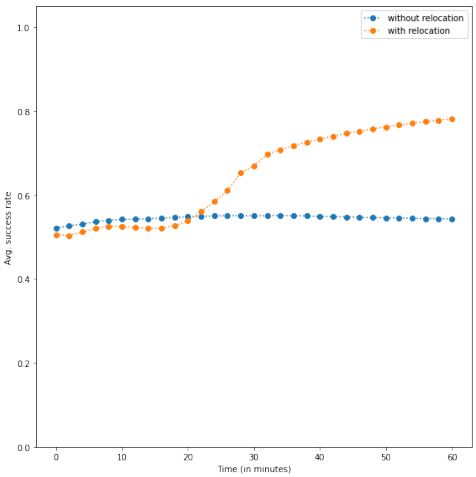
Finally, it is also interesting to learn how the adversary influences the average routing path length when it congregates all of its power in a single domain. In Figure 7.8a, we see that the average routing path in FedKad is increased significantly. The performance for $|D| = 8$ is even worse than in Kademlia. On the contrary, we



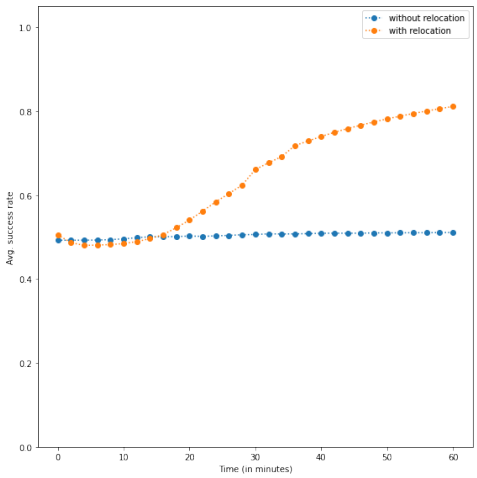
(a) Intra-domain node lookups for $|D| = 2$.



(b) Inter-domain node lookups for $|D| = 2$.

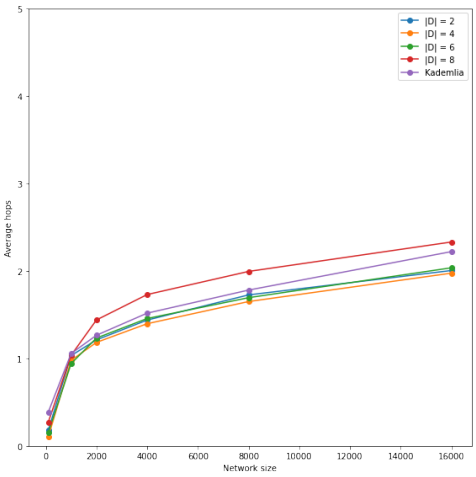


(c) Intra-domain node lookups for $|D| = 4$.

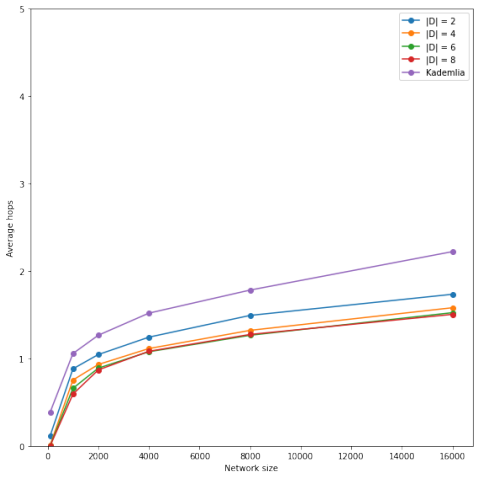


(d) Inter-domain node lookups for $|D| = 4$.

Figure 7.7: Average success rate of node lookups in an attacked domain over a period of 60 minutes in SovKad. These results are for an overlay with 16k nodes. The success rate increases more rapidly as $|D|$ increases. This is due to the fact that the relative increase in the proportion of honest individuals also becomes more pronounced for smaller domain sizes.



(a) Avg. success rate if malicious nodes are congregated in a single domain in FedKad.



(b) Avg. success rate if malicious nodes are congregated in a single domain in SovKad.

Figure 7.8: Avg. success rate if malicious nodes are congregated in a single domain. This figure shows that SovKad is more resilient against this type of attack than FedKad and Kademlia.

see that SovKad still outperforms Kademlia in terms of the average routing path. Our results for bandwidth costs are similar.

8

Conclusion and Future Work

In this work, we proposed a new approach to designing and evaluating a scalable P2P overlay architecture for blockchain systems. To our knowledge, our architecture is the first that meets the requirements of performance, security, and flexibility. Our approach, called Sovereign Domains P2P Overlay, segments the overlay into several sovereign sub-networks or domains, which operate independently and have their own internal structure and protocols, tailored to the characteristics and needs of heterogeneous nodes.

The Sovereign Domains P2P Overlay is a networking architecture that enables efficient dissemination of information within a domain, similar to a smaller P2P overlay. This is because the domains in this architecture are smaller in size, which makes them more efficient than larger P2P overlays. Additionally, the architecture has the security benefits of a larger P2P overlay as nodes maintain links to nodes from other domains. This increases network connectivity and enables the building of statistical models about other domains. These models can be used to monitor a predefined state of quality and detect malicious behavior from an adversary. The architecture also optimizes topologies and protocols for heterogeneous groups of nodes and allows nodes to participate in multiple domains simultaneously.

To demonstrate the effectiveness of the proposed overlay architecture, two novel node discovery protocols, FedKad and SovKad, were designed and implemented. The simulations conducted show that SovKad outperforms node discovery in a Structured P2P Overlay (Kademlia) and node discovery in a Federated P2P Overlay (FedKad), providing evidence of the potential of the Sovereign Domains P2P Overlay in blockchain systems.

8.1. Future Work

Interesting future directions in the area of Sovereign Domains P2P Overlay architecture could include:

- *Incentivizing domain nodes and migrations*: There is a need to design incentive mechanisms for nodes to join a particular domain and stay there. This can be done by offering rewards or other incentives for participating nodes. Additionally, it is important to consider how to incentivize nodes to migrate from one domain to another.
- *Combining different node discovery protocols*: Future work could evaluate the performance of a combination of different node discovery protocols, including both structured and unstructured protocols, to further improve the efficiency of node discovery.
- *Node behavior analysis*: A more in-depth analysis of node behavior in blockchain systems could be performed to group nodes based on their behavior and resources. This could improve the efficiency of the internal topology and protocols used within each domain. Additionally, we currently assume that nodes are already segmented to different domains. Future work could investigate mechanisms for segmenting nodes to domains and how this impacts the performance and security of the overlay.
- *New communication schemes*: Because communication strategies were out of scope of this work, it would be interesting to investigate how communication can be optimized in Sovereign Domains P2P Overlay. For example, future work could explore new multicasting schemes that can be used within and between domains to improve communication efficiency.

- *Influence on consensus mechanisms*: Future work could investigate how current or new consensus mechanisms can be used in conjunction with the Sovereign Domains P2P Overlay architecture. *Privacy and anonymity*: Some blockchains have strict privacy and anonymity requirements. Future work could explore how to incorporate these requirements into the Sovereign Domains P2P Overlay architecture.

Bibliography

[1] Karl Aberer et al. “The essence of P2P: A reference architecture for overlay networks”. In: *Fifth IEEE International Conference on Peer-to-Peer Computing (P2P'05)*. IEEE. 2005, pp. 11–20.

[2] Keith Alfaro. *Polkadot and Cosmos*. 2022. URL: https://guide.kusama.network/docs/learn-comparisons-cosmos/. (accessed: 01.11.2022).

[3] Gergely Alpár et al. “IRMA: practical, decentralized and privacy-friendly identity management using smartphones”. In: *10th Workshop on Hot Topics in Privacy Enhancing Technologies (HotPETs 2017)*. 2017, pp. 1–2.

[4] Salem Alqahtani and Murat Demirbas. “Bottlenecks in Blockchain Consensus Protocols”. In: *2021 IEEE International Conference on Omni-Layer Intelligent Systems (COINS)*. 2021, pp. 1–8. DOI: 10.1109/COINS51742.2021.9524210.

[5] Robert Antwi et al. “A survey on network optimization techniques for blockchain systems”. In: *Algorithms* 15.6 (2022), p. 193.

[6] Aditya Asgaonkar. “Scaling Blockchains and the Case for Ethereum”. In: *Handbook on Blockchain*. Springer, 2022, pp. 197–213.

[7] H. Bal et al. “A Medium-Scale Distributed System for Computer Science Research: Infrastructure for the Long Term”. In: *Computer* 49.05 (May 2016), pp. 54–63. ISSN: 1558-0814. DOI: 10.1109/MC.2016.127.

[8] Ingmar Baumgart and Sebastian Mies. “S/kademlia: A practicable approach towards secure key-based routing”. In: *2007 International conference on parallel and distributed systems*. IEEE. 2007, pp. 1–8.

[9] Juan Benet. “IPFS - Content Addressed, Versioned, P2P File System”. In: (July 2014).

[10] David Braun et al. “UP2P: a peer-to-peer overlay architecture for ubiquitous communications and networking”. In: *IEEE Communications Magazine* 46.12 (2008), pp. 32–39. DOI: 10.1109/MCOM.2008.4689205.

[11] Jeff Burdges et al. “Overview of polkadot and its design considerations”. In: *arXiv preprint arXiv:2005.13456* (2020).

[12] Vitalik Buterin et al. “A next-generation smart contract and decentralized application platform”. In: *white paper* 3.37 (2014), pp. 2–1.

[13] Miguel Castro et al. “Secure routing for structured peer-to-peer overlay networks”. In: *ACM SIGOPS Operating Systems Review* 36.SI (2002), pp. 299–314.

[14] Vincenzo Ciancaglini, Luigi Liquori, and Giang Ngo Hoang. “Towards a common architecture to interconnect heterogeneous overlay networks”. In: *2011 IEEE 17th International Conference on Parallel and Distributed Systems*. IEEE. 2011, pp. 817–822.

[15] George Danezis et al. “Narwhal and Tusk: A DAG-Based Mempool and Efficient BFT Consensus”. In: *Proceedings of the Seventeenth European Conference on Computer Systems*. EuroSys '22. Rennes, France: Association for Computing Machinery, 2022, pp. 34–50. ISBN: 9781450391627. DOI: 10.1145/3492321.3519594. URL: https://doi.org/10.1145/3492321.3519594.

[16] Sourav Das, Vinith Krishnan, and Ling Ren. “Efficient cross-shard transaction execution in sharded blockchains”. In: *arXiv preprint arXiv:2007.14521* (2020).

[17] Maya Dotan et al. “Survey on Blockchain Networking: Context, State-of-the-Art, Challenges”. In: *ACM Comput. Surv.* 54.5 (May 2021). ISSN: 0360-0300. DOI: 10.1145/3453161. URL: https://doi.org/10.1145/3453161.

[18] Maya Dotan et al. “Survey on blockchain networking: Context, state-of-the-art, challenges”. In: *ACM Computing Surveys (CSUR)* 54.5 (2021), pp. 1–34.

[19] Ronan Fahy et al. *Major data leak affects Dutch Covid-19 testing system and app*. 2021. URL: https://algorithmwatch.org/en/tracers/major-data-leak-affects-dutch-covid-19-testing-system-and-app/. (accessed: 01.01.2023).

[20] Md Sadek Ferdous, Farida Chowdhury, and Madini O. Alassafi. "In Search of Self-Sovereign Identity Leveraging Blockchain Technology". In: *IEEE Access* 7 (2019), pp. 103059–103079. DOI: 10.1109/ACCESS.2019.2931173.

[21] Arthur Gervais et al. "On the Security and Performance of Proof of Work Blockchains". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. CCS '16. Vienna, Austria: Association for Computing Machinery, 2016, pp. 3–16. ISBN: 9781450341394. DOI: 10.1145/2976749.2978341. URL: https://doi.org/10.1145/2976749.2978341.

[22] Arthur Gervais et al. "On the Security and Performance of Proof of Work Blockchains". In: CCS '16. Vienna, Austria: Association for Computing Machinery, 2016, pp. 3–16. ISBN: 9781450341394. DOI: 10.1145/2976749.2978341. URL: https://doi.org/10.1145/2976749.2978341.

[23] Giang Ngo Hoang et al. "A Backward-Compatible Protocol for Inter-Routing over Heterogeneous Overlay Networks". In: *Proceedings of the 28th Annual ACM Symposium on Applied Computing*. SAC '13. Coimbra, Portugal: Association for Computing Machinery, 2013, pp. 649–651. ISBN: 9781450316569. DOI: 10.1145/2480362.2480485. URL: https://doi.org/10.1145/2480362.2480485.

[24] Muhammad Anas Imtiaz et al. "Churn in the Bitcoin Network: Characterization and Impact". In: *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. 2019, pp. 431–439. DOI: 10.1109/BLOC.2019.8751297.

[25] Gene Kan. "Gnutella". In: *Peer-to-Peer: Ökonomische, technologische und juristische Perspektiven* (2002), pp. 189–199.

[26] Hidehiro Kanemitsu and Hidenori Nakazato. "KadRTT: Routing with network proximity and uniform ID arrangement in Kademlia". In: *2021 IFIP Networking Conference (IFIP Networking)*. IEEE. 2021, pp. 1–6.

[27] Harshit Kapoor et al. "Survey of various search mechanisms in unstructured peer-to-peer networks". In: *International Journal of Computer Applications* 68.6 (2013).

[28] Erick Lavoie, Miguel Correia, and Laurie Hendren. "Xor-overlay Topology Management Beyond Kademlia". In: *2017 IEEE 11th International Conference on Self-Adaptive and Self-Organizing Systems (SASO)*. IEEE. 2017, pp. 51–60.

[29] Shen Lin, François Taıani, and Gordon Blair. "Exploiting synergies between coexisting overlays". In: *IFIP International Conference on Distributed Applications and Interoperable Systems*. Springer. 2009, pp. 1–15.

[30] Luigi Liquori et al. "Synapse: A scalable protocol for interconnecting heterogeneous overlay networks". In: *International Conference on Research in Networking*. Springer. 2010, pp. 67–82.

[31] Zhiyu Liu et al. "Survive under high churn in structured P2P systems: evaluation and strategy". In: *Computational Science–ICCS 2006: 6th International Conference, Reading, UK, May 28-31, 2006, Proceedings, Part IV 6*. Springer. 2006, pp. 404–411.

[32] Eng Keong Lua et al. "A survey and comparison of peer-to-peer overlay network schemes". In: *IEEE Communications Surveys & Tutorials* 7.2 (2005), pp. 72–93.

[33] Tasnuva Mahjabin et al. "A survey of distributed denial-of-service attack, prevention, and mitigation techniques". In: *International Journal of Distributed Sensor Networks* 13.12 (2017), p. 1550147717741463.

[34] Apostolos Malatras. "State-of-the-art survey on P2P overlay networks in pervasive computing environments". In: *Journal of Network and Computer Applications* 55 (2015), pp. 1–23.

[35] Kai Mast. "Building Protocols for Scalable Decentralized Applications". In: *Handbook on Blockchain*. Springer, 2022, pp. 215–255.

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

[37] Thomas McGhin et al. "Blockchain in healthcare applications: Research challenges and opportunities". In: *Journal of Network and Computer Applications* 135 (2019), pp. 62–75. ISSN: 1084-8045. DOI: https://doi.org/10.1016/j.jnca.2019.02.027. URL: https://www.sciencedirect.com/science/article/pii/S1084804519300864.

[38] Adán G Medrano-Chávez, Elizabeth Pérez-Cortés, and Miguel Lopez-Guerrero. “A performance comparison of Chord and Kademlia DHTs in high churn scenarios”. In: *Peer-to-Peer Networking and Applications* 8.5 (2015), pp. 807–821.

[39] Robert Morris et al. “Chord: A scalable peer-to-peer look-up protocol for internet applications”. In: *IEEE/ACM Transactions On Networking* 11.1 (2003), pp. 17–32.

[40] Ashika R Naik and Bettahally N Keshavamurthy. “Next level peer-to-peer overlay networks under high churns: a survey”. In: *Peer-to-Peer Networking and Applications* 13.3 (2020), pp. 905–931.

[41] Satoshi Nakamoto. “Bitcoin: A peer-to-peer electronic cash system”. In: *Decentralized business review* (2008), p. 21260.

[42] Gleb Naumenko et al. “Erlay: Efficient transaction relay for bitcoin”. In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2019, pp. 817–831.

[43] Till Neudecker and Hannes Hartenstein. “Network Layer Aspects of Permissionless Blockchains”. In: *IEEE Communications Surveys Tutorials* 21.1 (2019), pp. 838–857. DOI: 10.1109/COMST.2018.2852480.

[44] Pim Otte, Martijn de Vos, and Johan Pouwelse. “TrustChain: A Sybil-resistant scalable blockchain”. In: *Future Generation Computer Systems* 107 (2020), pp. 770–780.

[45] Sangjin Park, Kwahngsoo Jang, and Jae-Suk Yang. “Information flow between bitcoin and other financial assets”. In: *Physica A: Statistical Mechanics and its Applications* 566 (2021), p. 125604.

[46] Riccardo Pecori. “S-Kademlia: A trust and reputation method to mitigate a Sybil attack in Kademlia”. In: *Computer Networks* 94 (2016), pp. 205–218.

[47] Asfandyar Qureshi. “Exploring proximity based peer selection in bittorrent-like protocol”. In: *MIT* 6 (2004), p. 824.

[48] Ranvir Rana et al. “Free2Shard: Adversary-Resistant Distributed Resource Allocation for Blockchains”. In: 6.1 (Feb. 2022). DOI: 10.1145/3508031. URL: https://doi.org/10.1145/3508031.

[49] Elias Rohrer and Florian Tschorsch. “Kadcast: A Structured Approach to Broadcast in Blockchain Networks”. In: *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*. AFT ’19. Zurich, Switzerland: Association for Computing Machinery, 2019, pp. 199–213. ISBN: 9781450367325. DOI: 10.1145/3318041.3355469. URL: https://doi.org/10.1145/3318041.3355469.

[50] Stefanie Roos, Hani Salah, and Thorsten Strufe. “Determining the hop count in kademlia-type systems”. In: (2015).

[51] Stefan Saroiu, P Krishna Gummadi, and Steven D Gribble. “Measurement study of peer-to-peer file sharing systems”. In: *Multimedia computing and networking 2002*. Vol. 4673. SPIE. 2001, pp. 156–170.

[52] Abhishek Sharma et al. “Blockchain technology and its applications to combat COVID-19 pandemic”. In: *Research on Biomedical Engineering* (2020), pp. 1–8.

[53] Xuemin Shen et al. *Handbook of peer-to-peer networking*. Vol. 34. Springer Science & Business Media, 2010.

[54] Fatemeh Shirazi et al. “A survey on routing in anonymous communication protocols”. In: *ACM Computing Surveys (CSUR)* 51.3 (2018), pp. 1–39.

[55] Emil Sit and Robert Morris. “Security considerations for peer-to-peer distributed hash tables”. In: *International Workshop on Peer-to-Peer Systems*. Springer. 2002, pp. 261–269.

[56] Mudhakar Srivatsa, Bugra Gedik, and Ling Liu. “Large scaling unstructured peer-to-peer networks with heterogeneity-aware topology and routing”. In: *IEEE Transactions on Parallel and Distributed Systems* 17.11 (2006), pp. 1277–1293.

[57] Ethereum Team. *Networking layer*. 2022. URL: https://ethereum.org/en/developers/docs/networking-layer/#devp2p. (accessed: 01.09.2022).

[58] Dennis Trautwein, Moritz Schubotz, and Bela Gipp. “Leveraging node heterogeneity to improve content discovery and content retrieval in peer-to-peer networks”. In: *Proceedings of the CoNEXT Student Workshop*. 2021, pp. 17–18.

[59] Maarten Van Steen and Andrew S Tanenbaum. “A brief introduction to distributed systems”. In: *Computing* 98 (2016), pp. 967–1009.

[60] Gang Wang et al. “Sok: Sharding on blockchain”. In: *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*. 2019, pp. 41–61.

[61] Andy B Yoo, Morris A Jette, and Mark Grondona. “Slurm: Simple linux utility for resource management”. In: *Workshop on job scheduling strategies for parallel processing*. Springer. 2003, pp. 44–60.

[62] Mahdi Zamani, Mahnush Movahedi, and Mariana Raykova. “RapidChain: Scaling Blockchain via Full Sharding”. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. CCS ’18. Toronto, Canada: Association for Computing Machinery, 2018, pp. 931–948. ISBN: 9781450356930. DOI: 10.1145/3243734.3243853. URL: https://doi.org/10.1145/3243734.3243853.

[63] Dongyan Zhang et al. “Identification and analysis of skype peer-to-peer traffic”. In: *2010 Fifth International Conference on Internet and Web Applications and Services*. IEEE. 2010, pp. 200–206.

[64] Lihao Zhang, Taotao Wang, and Soung Chang Liew. “Speeding up block propagation in Bitcoin network: Uncoded and coded designs”. In: *Computer Networks* 206 (2022), p. 108791.

[65] Zibin Zheng et al. “An overview of blockchain technology: Architecture, consensus, and future trends”. In: *2017 IEEE international congress on big data (BigData congress)*. Ieee. 2017, pp. 557–564.

[66] Zibin Zheng et al. “Blockchain challenges and opportunities: a survey”. In: *International Journal of Web and Grid Services* 14.4 (2018), pp. 352–375. DOI: 10.1504/IJWGS.2018.095647. eprint: https://www.inderscienceonline.com/doi/pdf/10.1504/IJWGS.2018.095647. URL: https://www.inderscienceonline.com/doi/abs/10.1504/IJWGS.2018.095647.