DEPARTMENT OF
COMPUTER SCIENCE

TIAGO SEQUEIRA TELES
BSc in Computer Science

TOWARDS A SECURE AND PRIVACY-AWARE OPEN ENERGY MARKET

Dissertation Plan
MASTER IN COMPUTER SCIENCE

NOVA University Lisbon
February, 2023

DEPARTMENT OF
COMPUTER SCIENCE

TOWARDS A SECURE AND PRIVACY-AWARE OPEN ENERGY MARKET

TIAGO SEQUEIRA TELES
BSc in Computer Science

Advisers: João Leitão
Assistant Professor, NOVA University Lisbon
Carla Ferreira
Associate Professor, NOVA University Lisbon
Rafael Rodrigues
Researcher, EDP New R&D

Dissertation Plan
MASTER IN COMPUTER SCIENCE
NOVA University Lisbon
February, 2023

Abstract

The increasing concerns about privacy and trust in centralized entities has resulted in a growing interest in decentralized solutions, one such case is decentralized resource trading. With the proliferation of smart devices and IoT systems the opportunity to realize such systems has become a reality. The work planned in this document is motivated by the desire to address this need and explore the potential of decentralized peer-to-peer systems in creating open markets that prioritize the privacy and security of its users.

We will explore the potential of decentralized peer-to-peer systems in creating secure and privacy-aware open markets. The research will be conducted as part of the European TaRDIS project which aims to build a robust and extensive toolkit for decentralized swarm systems programming. The main use case for the research will be a smart decentralized energy market for communities, where electric vehicles will play a significant role in the energy grid, namely by enabling the possibility of using them as batteries for energy generated by home users solar panels. In particular we will focus on developing decentralized peer-to-peer communication and management protocols that facilitate the exchange of resources while taking into account the aspects of privacy and security. The proposed solution will be evaluated through simulations, and a comparison of the decentralized solution with a baseline centralized approach will also be conducted, focused on the use case described above.

Keywords: Distributed Systems, Peer-to-peer markets, Security, Energy

Resumo

As preocupações emergentes com privacidade e confiança em entidades centralizadas resultaram num interesse crescente em soluções descentralizadas, um destes casos é o de comércio descentralizado de bens. Com a proliferação de dispositivos inteligentes e sistemas IoT, a oportunidade de realizar tais sistemas tornou-se uma realidade. O trabalho traçado neste documento é motivado pelo desejo de responder a essa necessidade e também de explorar o potencial de sistemas *peer-to-peer* descentralizados na criação de mercados abertos que priorizam a privacidade e a segurança de seus utilizadores.

Exploraremos o potencial destes sistemas *peer-to-peer* descentralizados na criação de mercados abertos, seguros e amigos da privacidade. A pesquisa irá ser efetuada como parte do projeto europeu TaRDIS, que visa construir um conjunto de ferramentas robustas e extensas para programação descentralizada de sistemas em *swarm*. O principal caso de utilização para esta pesquisa será um mercado de energia descentralizado inteligente para comunidades energéticas, onde os veículos elétricos terão um papel significante na rede de energia, nomeadamente ao permitir a possibilidade de os usar como baterias para energia gerada por painéis solares de utilizadores domésticos.

Em particular, concentrar-nos-emos no desenvolvimento de protocolos descentralizados de comunicação e gestão *peer-to-peer* que auxiliem a troca de bens, tendo em consideração os aspectos de privacidade e segurança. A solução proposta será avaliada por meio de simulações, e também será realizada uma comparação da solução descentralizada com uma abordagem centralizada servindo de base, com foco na utilização descrita acima.

Palavras-chave: Sistemas distribuídos, Mercados descentralizados, Segurança, Energia

Contents

List of Figures

1 Introduction

The proliferation of smart devices and IoT systems [4] has opened the door for decentralized resource trading via decentralized markets. However, these systems must be designed with security and privacy in mind to ensure the integrity and privacy of its users. The work proposed in this document will explore the potential of distributed peer-to-peer systems in creating secure and privacy aware open markets and their practical applications.

Decentralized systems are increasingly more popular, as privacy, and trust in centralized entities become more widespread concerns [6]. However, these systems are inherently more complex to work with [2], as there are more possible points of failure, the physical conditions have less quality control than in a data center, the networks are heterogeneous and outages may happen. We will design and implement a protocol that works correctly in these conditions.

We seek to propose a peer-to-peer communication protocol for the management of resource supply and demand within the industry. The protocol will be designed to operate in a decentralized fashion, with a partially hierarchical network structure, ensuring the privacy of its users while also providing robust defense against malicious actors.

The proposed solution will require the development of a decentralized membership and data dissemination mechanism for automatically matching of new resource offers and requests, along with a filtering mechanism to exclude unwanted messages and minimize network congestion caused by the transmission of irrelevant or malicious offers.

1.1 Objective

This dissertation aims to investigate the potential of decentralized peer-to-peer (P2P) local resource trading markets, with a fallback mechanism to more centralized systems in the event that resources cannot be obtained locally. The work intends to develop decentralized peer-to-peer communication and management protocols that facilitate the exchange of resources, taking into consideration the aspects of privacy and security.

The proposed solution will enable the effective dissemination and acceptance of resource offers, and provide the means for subscribing to offers and requests of interest, such as notifications of resource availability at specific times and prices. The work will document the design and implementation of the protocol and evaluate its performance through simulations, compared to a baseline centralized approach.

1.2 Expected contributions

1. The design and implementation of a centralized control mechanism, and its benchmarks, in order to set a baseline performance to compare our solution against.

2. The design, implementation and evaluation of the proposed decentralized solution.

3. A comparison of the two systems.

1.3 Research Context

This dissertation is part of a larger project, the European TaRDIS (Trustworthy and Resilient Decentralised Intelligence for Edge Systems), inserted in the Horizon Europe project, an effort to build a robust and extensive toolkit for distributed systems programming, for use in the industry, from satellites, to decentralized machine learning, to smart factory control, to our partner's usecase, which is a decentralized dynamic market [31].

The research will be conducted in partnership with EDP New R&D, a research and development branch of the Portuguese energy company EDP, one of the Portuguese members of the TaRDIS project.

1.4 Use case

Since the research to be conducted is part of the TaRDIS project, our main use case will be a smart decentralized energy market for communities, which is one of the main use cases of the TaRDIS project.

Electric vehicles are a large concern for electric grids across the world as they represent an unforeseen load on these systems [3]. Their immense power consumption, as well as their quick rise in numbers, means grid operators have to satisfy their client's needs without having the luxury of time to upgrade their power generation and other infrastructure. They are also a new opportunity for optimization, being able to store energy at low load times on the network, releasing it when there is higher demand, this way distributing the load while taking advantage of nearby energy resources to reduce energy loss due to Joule's effect [32].

Our solution will exploit this opportunity by assisting the coordenation of the electric vehicle's charging and discharging cycles, the local energy production and consumption, and the general electric grid.

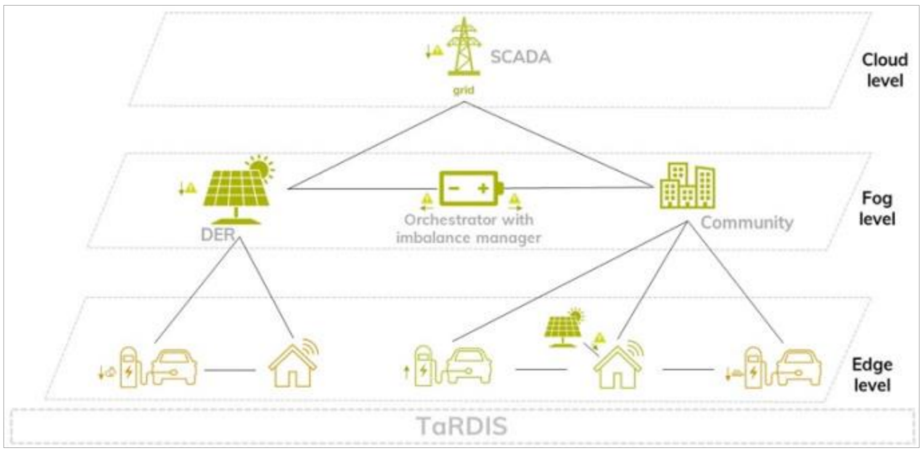

Figure 1.1: Diagram of the hierarchical structure proposed by the TaRDIS project [5].

1.5 Dissertation Structure

The rest of the document is structured as follows:

Chapter 2 will present distributed systems in more detail. More specifically, we will elaborate on the concepts of peer-to-peer systens, overlay networks, publish/subscribe and dissemination solutions that operate in a decentralised fashion. We will then take an overview of the current state of the art in the field, explaining existing solutions and discerning their advantages and disadvantages, and how they can be applied to our work. Although there is some writing on the topic of decentralized local markets, nearly all use some sort of blockchain technology, which is not what we intend to do.

Chapter 3 will present ideas on how to design a suitable protocol for our usecase, how to validate it, and finally, a rough scheduling of future work.

2 Related Work

This chapter serves as the foundation for the current work, provides an overview of the key concepts and an examination of relevant literature. It aims to provide an understanding of the base on which this dissertation will be developed.

We will further detail the issues in existing solutions and explain why we believe we can improve on them, justifying the search for decentralized solutions. We will also characterize the execution environment of our network.

In Section 2.1, we will introduce the concept of peer-to-peer; the overview of different peer-to-peer services can be found in Section 2.2, while the modes of peer-to-peer communication are discussed in Section 2.3; the topic of peer-to-peer overlays is explored in Section 2.4; finally, in Section 2.6, we will provide some insight into the field of decentralized markets, and in Section 2.7, we will discuss privacy issues; in Section 2.5, we will consider the possible attack vectors in peer-to-peer systems.

2.1 General Peer-to-Peer

It is relevant to specify the meanings with which we will use some terms in this work.

To define what peer-to-peer systems are, we have to explain the context that they are inserted in. In terms of communication, there are two main types of networks: centralized and decentralized, peer-to-peer falls into the latter.

Centralized networks feature a designated logical point, often referred to as a "server," through which all communication is funneled. This contrasts with "client" nodes, which do not receive communications from each other. Because of the centralized nature of the communication, the server knows everything happening in the system exactly when it happens, meaning there is a central source of truth, making complex synchronization mechanisms unnecessary. This, combined with the fact that these servers are generally controlled by whoever is deploying the system, meaning better hardware conditions, means that centralized systems have a speed advantage. On the other hand, centralized systems are harder to scale, since we cannot infinitely scale the speed and bandwidth of the central server, they also suffer from privacy, reliability and security issues, as having all traffic routed through a central point means it can spy on everything at once, and if there is a security issue, this single point can compromise the whole network.

Unlike centralized networks, peer-to-peer systems have no central server. In these network architectures, nodes communicate directly amongst themselves in order to achieve their shared goal. This conveys them a few advantages as well as disadvantages. On the positive side, they have no single point of failure, therefore as a total network they will be reliable, even if each node is unstable, these network types can also theoretically scale infinitely, as we can always get more computers, even if we cannot get faster ones. On the negative side, these networks are very hard to synch globally (all of the nodes having the same information with no conflicts) and they might also incur redundancy costs that would not be necessary if we could grant that the main server would be perfectly available nearly all of the time.

In our solution, we do not intend to synchronize the whole network, we only intend to have nodes get notified of data with an origin near them, as this information is the one that is likely to be useful for each node. This automatically lessens the main disadvantage of decentralized systems. The mitigation of this disadvantage, combined with our need for privacy, scalability, and security, makes distributed systems a likely good fit for our problem.

2.1.1 Structure of peer-to-peer networks

In the context of peer-to-peer systems, the structure of the network can either be flat (single-tier) or hierarchical (multi-tier) [34]. A flat organization is one where all nodes have equivalent tasks and load, where a hierarchical organization would imply that different nodes have different types of tasks or load.
The Gnutella distributed file sharing protocol, from version 0.6 onwards [13], is one such example of a hierarchical network, as it makes use of so called "ultrapeers", peers that support a larger load than other nodes in the system. These are special nodes that have a much higher fanout than other "leaf nodes", and serve as a proxy for these nodes' queries to the network. This hierarchical peer-to-peer system presents performance benefits by effectively lowering the network size with these ultrapeers, diminishing the number of hops needed to transverse the network in the look for our search terms. The obvious downside of such mechanism is that these ultrapeers handle the load of all of their leaf nodes, meaning they have to have higher processing capacity and bandwidth.
An example of a fully flat protocol would be that of HyParView [14], a membership protocol for reliable message broadcast. This protocol achieves its performance and reliability not by the use of super peers, but by the agressive declustering caused by the use of random walks in the network any time new peers are requested, a random walk being a process by which a peer discovery request is forwarded to a random peer in a node's active view for a predetermined hop count. Random walks allow us to randomly sample the network, establishing connections with nodes that are far away from us.

Our solution intends to have a semi-hierarchical structure, with nodes communicating with eachother on the same level, and using nodes on a higher level as backup, or as assistance, when the low tier node is too low powered to perform its requests efficiently, thus super peers are a feature of our solution. We will not be using HyParView's random walks, as clustering is a positive effect in our use case, as this effect mirrors the real world's distances.

The use of these layers also allows us to better fit the control requirements of the intended use on industrial applications. This is demonstrated in the use case referred to in Chapter 1, where the usage of super peers under control of the energy company would facilitate the addressing of misbehaving users. We also intend to build a resilient system that can work reliably even if not constantly connected to the upper layers, this is why we are opting for a hybrid approach. The vast majority of the usage in the protocol will be end nodes communicating with end nodes, however there is still the option to terminate a node's access in a centralized fashion.

Overlays on peer-to-peer networks will be further detailed in Section 2.4

2.1.2 Peer-to-peer systems composition

Peer-to-peer systems are commonly divided into a few layers, these layers communicate with eachother to form the full peer-to-peer application stack, where each layer supports the one above it, supplying it with the functionality necessary to achieve its purposes. Each layer communicates with the equivalent layer on other nodes from the network and with the adjacent layers on the same node. This model allows us to divide the complexity of a full peer-to-peer system into manageable self contained segments. The large majority of peer-to-peer systems are built as overlays to the IP stack, or as an overlay to an overlay of the IP stack. These layers are often the ones described below [16].

Overlay Network This layer manages the membership of nodes to the network. This is generally achieved by using specifically designed methodologies that assign and maintain, for each peer in the system, a set of neighbors which that peer can be sure exist, and that it can attempt to communicate with. The layer should also notify the upper layers of changes in the membership, such as nodes leaving and entering the neighborhood. In our solution, this will be the layer that keeps the nodes connected to eachother, the neighbors we want to provide should be physically close to the requesting node, as most requests will want to be fulfilled in close proximity. The topic of overlay networks will be further detailed in Section 2.4

Service The service layer builds on top of the forementioned overlay layer to provide useful primitives, like message routing, subscription to a certain topic, or the publishing on that same topic. We intend to make use of all of these features in order to allow negotiation between nodes, and notification of new resource offers and requests. We will elaborate on this layer in Section 2.2

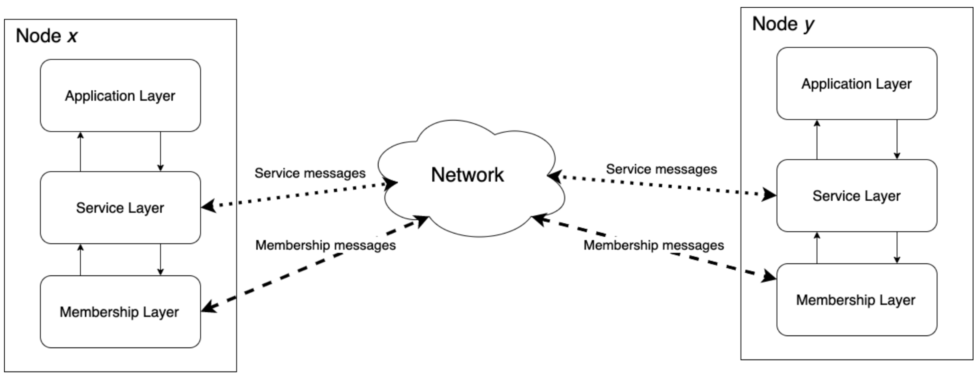

Figure 2.1: Peer-to-peer layered architecture [22]

Application Finally, this last layer implements the application logic and is also responsible for presenting some sort of interface to the user of the application. This might be in the form of an API, or an actual GUI application. This will be the layer where the heuristics for the decisions on what resources to offer and request, and also when to do so will be done. The solution for this layer presented in this work will be basic and will make no attempt at optimization, as resource availability prediction is a complex separate field of research, we will simply attempt to demonstrate that we have successfully built the previous two layers.

2.2 Peer-to-peer services

This section delves into the relevant services that are integral to our work and evaluates various existing solutions in relation to our objectives. These mechanisms, which are implemented through the service layer outlined previously, encompass functions such as message routing, which assists in transmitting a message from one specific node to another designated node through updated and efficient paths, shared storage, which facilitates network users in the storing and retrieving of data blobs, publishing/subscribing to information, where a node may subscribe to a particular topic and receive notifications when information is published on that topic, a simple broadcast, a way to effectively transmit a message to every node in the system.

A potential application that incorporates these services could be a rudimentary decentralized social network, enabling users to publish content on a specific topic and allowing other users, or the same users, to subscribe to it. The network could also facilitate long-term media hosting through decentralized data storage and assist in direct communication between users via point-to-point message routing.

2.3 Message Delivery

There are different kinds of message dessimination types, the ones that we will explore in the work are point-to-point message delivery, publish/subscribe systems and message broadcasting. Within these types, you can divide them into probabilistic or deterministic dissemination schemes. Probabilistic schemes will deliver the messages with a high degree of certainity, while deterministic schemes guarentee message delivery, both with the obvious requirement of the network graph having to be connected, meaning the network may not be separated into more than a single partition. We will also delve into performance metrics used for distributed systems.

2.3.1 Point-to-point message delivery

The concept of point-to-point message delivery refers to the process of transmitting a message from one specific node to another designated node. In order for this to occur, there must be a means of addressing the recipient node, such as an identifier. The most immediate approach to message dissemination would be a basic broadcast method, in which the sending node first sends the message it wants to transmit to its neighbors, then each neighbor transmits received messages to all of its adjacent nodes if it is the first time they have received the message and this continues until every node has received the message at least once, including the node we intended to target. This method completely floods the network and is not efficient in terms of bandwidth if our objective is to send a message to a specific other node.

A more efficient solution may be achieved in the use of DHTs, as we may slightly bend their intended use case to address each node individually without requiring a broadcast.

2.3.2 Broadcast

Message broadcasting refers to the mechanism by which a single node disseminates a message to all nodes within the network. The straightforward approach discussed earlier is more appropriate for this task than the for point-to-point message delivery.

A frequently utilized approach to address this issue is through the utilization of gossip protocols, which belong to a category of mechanisms that aim to disseminate information in an unstructured overlay network through forwarding received messages to some or all of the neighboring nodes. The strategy described previously operates as a gossip strategy, which is typically probabilistic in nature. However, in this basic strategy, this characteristic is not present as every contactable node will be made aware of the incoming communication if every node interacts with all possible nodes. Reducing the number of nodes to which each node forwards messages has the dual effect of decreasing reliability and reducing the amount of overhead incurred, this parameter in gossip protocols (how many nodes we forward the message to) has the name of fanout. Within gossiping protocols, there is a significant challenge in deciding on which is the best fanout to use, in order to keep very high reliability, but not overload the network.

2.3.3 Publish and subscribe

Publish/subscribe is a paradigm for large-scale, distributed systems. In general, subscribers register their interest in a topic or a pattern of events and then asynchronously receive events matching their interest, regardless of the events' publisher. There are two major types of publish/subscribe systems, topic, and content based, with a hybrid of the two also being possible. Topic-based publish/subscribe is very similar to group- based communication; subscribing is equivalent to becoming a member of a group [20]. In the case of content based publish/subscribe each individual subscriber sets their specific filters for which content they intend to receive.

This is generally achieved through a message broker that acts as an intermediary between publishers and subscribers. The broker is responsible for receiving published events, categorizing them based on the topics or patterns specified by the subscribers, and then delivering them back to the subscribers.

When referring to topic based publish/subcribe a prime case is the large scale notification infrastructure Scribe [20]. Scribe functions by using the Pastry DHT [28] to assign topics to brokers, each topic corresponds to a hash on the Pastry DHT, and the node with their id closest to that hash is deemed a rendez-vous point for that topic. When nodes intend to subscribe to a topic, they will send a message to the broker, and each node that forwards this message, as well as the broker node, will take note of who was attemping to subscribe. When a new event is published, this path will simply be traced back and the message will be delivered to every node that subscribed. This mechanism works in conjunction with the underlying DHT to efficiently create event dessimination paths to the interested parties.

A modified version of this classic publish/subscribe model would be very close to some of the properties we intend to achieve. In the context of this work, we could subscribe to a certain price from which we intend to be notified of offers of our resource, since this is a range and not a fixed value modifications to Scribe would be necessary, such as broker nodes for higher prices automatically subscribing to nodes with lower prices, since this way, if we subscribe to being notified of offers at a high price, we will also get offers at a lower price, which we are likely interested in. This leads us back to the other major type of publish/subscribe systems: content based publish/subscribe.

Since the interests of each node may be different, it is impossible to implement content based publish/subscribe in a purely peer-to-peer network [7]. One of the strategies suggested by *A.Datta et al.* is the use of super peers. These peers may prefilter the content that is effectively transmitted to the end nodes, grouping nodes with similar interests into a specific super node's *cluster*, with *cluster* meaning the group of nodes connected to that super node. This allows us to reduce the overhead occurred in transmitting messages to super nodes that will not be sent to end nodes, as the more nodes are interested in a topic from that super node, the more nodes will likely have a subcription policy that accepts the received content.

2.3.4 Performance Metrics

Performance metrics for distributed systems are the ways we are able to quantify the effectiveness of the network. In this work, we will analyze four metrics that are deemed relevant and explain their usefulness in specific cases.

Last delivery hop Last delivery hop, or LDH, measures the maximum number of nodes that a message passes through before it reaches its final destination. This metric is highly relevant in nearly all peer-to-peer systems. In the case of broadcasting solutions, we will assess the size of the broadcast with the most hops that delivered a message. For point-to-point solutions, the metric is simply the length of the chain of nodes that lead from the origin node to the destination. LDH serves as an estimation of the network's diameter, which is defined as the maximum length of the shortest path between two nodes in the network. It is important to understand the network's diameter as it provides insights into the level of clustering and the effectiveness of adjustments made to the protocol.

Latency Latency is a similar performance metric to LDH, and is applicable to all peer-to-peer systems. However, instead of measuring the number of hops, Latency measures the elapsed time from the initiation of the message transmission until its receipt at the destination. This metric is crucial in determining the desired low latency in the system and can provide insight into the performance of the network in terms of timely message delivery.

Reliability Reliability refers to the ratio of the number of intended recipients of messages to the number of messages that were actually received. For point-to-point message delivery, each individual message sent can usually have either 100% reliability if it reaches its intended destination or 0% reliability if it does not. However, by repeating this process multiple times, valuable metrics can be obtained. In the case of broadcasting, reliability would be calculated as the number of nodes that received the message divided by the size of the network minus one, to account for the node that initially transmitted the message. This metric is not very useful to deterministic delivery models, as it should always be at 100%.

Relative Message Redundancy Relative Message Redundancy, or RMR, is a metric that calculates the ratio of the number of messages exchanged during a message transmission to the number of nodes through which the message was forwarded or delivered. This metric is particularly useful in evaluating the overhead of the peer-to-peer network, as high values suggest that many unnecessary messages were exchanged, and improvements can be made. This metric is most effective in broadcast scenarios, as point-to-point communication typically does not forward messages through nodes that are not ultimately utilized. In the case of gossip protocols, RMR directly helps to adjust parameters, such as fanout, to strike a balance between reliability and overhead.

2.4 Overlay networks

An overlay network refers to a logical layer that is constructed on top of another network. It can be represented as a graph where each node is connected to its neighboring nodes through an edge, which signifies the capability for direct communication. The links between nodes in the overlay network can be either directed or undirected, depending on whether the communication between nodes is symmetric or asymmetric.

Another property of overlays is that of topology, where there is a distinction between structured and unstructured protocols. This property simply refers to the structure of the underlying data on the network. If a node has specific data delegated to it, through an ID or some other mechanism, it is structured. This is the case with distributed hash tables often referred to as DHTs, such as Chord [30] and Kademlia [21], the second of which will be explored in more detail in Section 2.4.1. In contrast, unstructured networks have their nodes be responsible for their data. We will explore a few unstructured overlays in Sections 2.4.2, 2.4.3 and 2.4.4.

Most overlays are built either on top of IP, or on top of other overlays. Why should we use overlays and neighbor management at all? Why not simply have every peer in the network be a neighbor of every other peer, and this way, any communication is possible with a single hop? There are a couple of issues that make this unfeasible on large systems. If any time a node wants to join a network it has to establish a connection with all other nodes, it will quickly become impossible to maintain these connections due to hardware constraints, both memory, for storing all the peers and their properties, and bandwidth. Peer-to-peer systems are dynamic, as new nodes often join and leave at will, this would mean also keeping track of leaving nodes, once again making every other node react. It would not be sustainable to do so.

We will detail a few relevant examples of overlay networks for our use case below.

2.4.1 Kademlia

Kademlia [21] is a protocol that creates and maintains a structured overlay. Despite having a similar number of expected hops per search query as Chord, Kademlia features some advantages over it. One of these advantages is that configuration information automatically spreads through search queries. Additionally, Kademlia uses parallel search queries to remain fault-tolerant in the case of node crashes. This protocol has a few optimizations compared to simpler protocols such as Chord, the most important being the use of an XOR metric to measure distances between nodes. This allows for symmetric links between nodes, resulting in greater flexibility in search query redirection. Unlike Chord, Kademlia

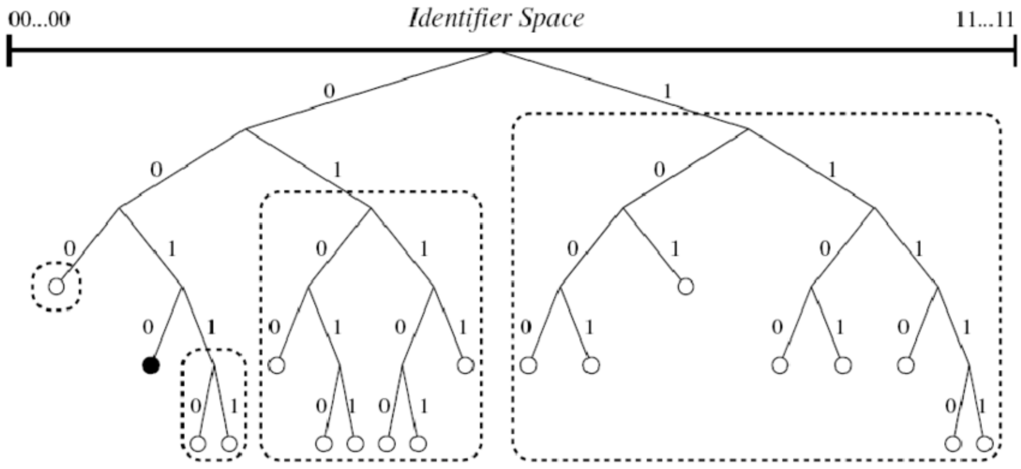

Figure 2.2: A visualisation of Kademlia's buckets and topology, each dotted line denotes a bucket, while the black node denotes the origin [11]

can redirect a search query to any node that is close to the desired object's hash, or even to multiple nodes in parallel.

Kademlia utilizes a virtual tree structure to organize its network, with nodes identified through 160-bit numerical IDs. This structure allows the network to be divided into fixed size buckets, depending on the prefix similarity between the nodes' IDs and the original node's ID, illustrated in Figure 2.2. The node aims to fill these buckets, leading to close nodes in terms of ID being well-mapped, while distant nodes being sparsely mapped. This enables Kademlia to nearly always guarantee a hop to a node that reduces the distance to the destination by half, leading to its O(log(n)) performance.

Kademlia, like other DHTs may be trivially used to implement a topic based publish/subscribe overlay. By simply hashing the topic's name, we can get a key to query Kademlia with, where the group of nodes close to that key may be used as a rendezvous point for that topic, handling subcriptions and new events, registering the ids of subscribing nodes and using Kademlia once again to route the events back.

2.4.2 X-Bot

This overlay is built to **B**ias the **O**verlay **T**opology with a certain criteria **X** [17, 15]. The purpose of this protocol is to minimize a link cost metric, which is set by an encapsulated local oracle within each node, while preserving the overall number of links. The X-Bot framework provides the flexibility to use any desired efficiency criterion for the link cost, including latency, distance, or in the context of our decentralized energy market, the efficiency of electricity transmission between two nodes. This decentralized protocol relies in a 4-node optimization technique, where these nodes participating in an optimization round swap links between themselves in order to achieve the lowest total cost. In this process, each node *O* starts optimization rounds where it attempts to swap a node *A* in its active view with a better node *B* in its passive view, and that node contacts a third node *C*.

The origin node will swap its connection with node A with a connection to node B, while node B will swap its connection to node C with a connection to node A, with A and C going into eachother's active views. To summarize, O-A and B-C turns into O-B and A-C. We should also mention X-Bot keeps its passive view unbiased as to avoid clustering, this is a very relevant feature for our use case, as in the real world, nodes on a similar grid are likely to go down at the same time, meaning we could become disconnected if there was no backup when all our neighbors dropped. X-Bot is promising as a flexible unstructured overlay for our work.

2.4.3 Legion

Legion is a framework that allows web applications to securely replicate data from a server and directly synchronize these replicas among the peers in the system [19]. It leverages WebRTC to allow these connections in the environment of a web browser. It uses its own unstructured overlay inspired by HyParView, but instead there is work put into minimizing latency, this is done by biasing the network with basis on each node's position in a Cartesian space of W dimensions, based on the node's latency to different servers. This is not simply based on RTTs between clients as WebRTC has a high startup cost. The most relevant parts of this paper for our work is the dessimination of new alterations to the shared data in a secure, causal consistency respecting fashion. The essential aspect of this paper that is pertinent to our work is the dissemination of new modifications to the shared data in a secure and causally consistent manner. To accomplish this, Legion employs a two-pronged approach. With regards to security, the system uses a central server to manage a set of symmetric keys and only grants access to authorized clients. This ensures that unauthorized clients cannot forge messages or intercept the communications of legitimate clients. With regards to data replication, the mechanism employed is straightforward. Each node maintains a list of the received deltas in a causally consistent manner, and a client propagates this list to every client it connects to. The receiving client will then either ignore the information if it already possesses it or add the new data to the end of its delta list and modify the stored data accordingly, forwarding its new list of deltas to neighbors. As our work requires every client in the system to have access to all of the events, we cannot solve the issue of security by simply encrypting all of the communication with a shared symmetric key.

2.4.4 Practical Client-side Replication

"Practical Client-side Replication: Weak Consistency Semantics for Insecure Settings" [18], like Legion, uses a centralized server for managing access to the network and achieves causal consistency. However, it focuses specifically on addressing client misbehavior and defining a secure form of causal consistency. It also lists a few properties and justifies how attacks against them are evaded. We will delve in detail into them, as these are particularly important to our work.

Immutable history Immutable history consists on an event's history not being modifiable after it is published to the system. This is achieved by each node signing their events as well as their dependencies, creating a linked tree of dependencies which bad actors cannot freely modify.

No Future Dependencies This property consists on a bad actor not being able to create an event that depends on events that it has not seen yet. This is solved similarly to the previous issue, since the event chain is signed, an adversary cannot sign a chain it has not seen yet.

Causal Execution Causal execution is the property that defines that all correct replicas execute their operations respecting the dependencies defined in the operation. This is assured through the verification that previous operations have already been executed, and using the dissemination strategy of Legion we can be confident replicas will only receive events once they have that event's dependencies.

Limited Omissions This property assures a malicious actor cannot omit events in the history of an event. This falls under the same protections as Immutable History

Eventual Sibling Detection As the name suggests, this property assures two events with the same id are eventually detected to be faulty, this will happen once both execution paths with sibling events reach any well behaving node, assisting with this, the server periodically broadcasts a summary of its state, if a client detects it has desynched with the server, it will contact it and verify each event to understand where the malicious event occurred, once again detecting the adversary's misbehaviour.

It is relevant to mention that whenever malicious activity is detected, a *proof-of-misbehavior* is produced and sent to the server, which in turn broadcasts the malicious node's id to the network, leading connections with it to be dropped, and the node to be isolated from the system. The concept of proof-of-misbehavior will be useful for our work, as well as the mechanisms through which the work of *A.Linde et al.* functions, as it allows our system to be protected against bad actors with little overhead. It has also been tested with larger amounts of data than what is expected to be handled by our system.

2.5 Attack vectors

There are a few attacks that can be leveraged against distributed markets. Some of these attacks may allow malicious actors to gain an advantage in the system, allowing them to acquire a certain resource without paying, paying less than they would in a non tampered scenario, or denying competitors their resources. These are the types of attacks that bad actors are interested in pursuing, so they are the ones we are most interested in defending against. The defenses against these attacks will be detailed in Chapter 3

Impersonation Impersonation is an attack where a malicious node sends a message pretending to be another node. In our scenario it may be used to send a buy order in the name of another node, charging them for something not ordered. The general solution for this attack is having the messages sent by the nodes be in some way authenticated.

Replay attacks Even with messages being authenticated we may still be subject to replay attacks. These attacks consist in simply re-sending an already authenticated message, possibly causing the system to act as if a certain node sent two offers instead of one, marking it for bad behaviour when it does not fulfil those two orders. It can be used against a competitor to assist in expelling it from the market.

Man-in-the-Middle Man in the Middle, or MitM attacks consist of a node intercepting communication between two other nodes, this might be used for eavesdropping, cutting communication, or altering messages. We could per example drop packets that were intended to bid on something we are interested in, or modify what is being offered by another node that is competing with us. We can significantly decrease the power of such an attack by not revealing the message contents to the nodes routing it, making modification and eavesdropping impossible if implemented correctly, but not defending against packet dropping.

Routing attack Routing attacks could be used to facilitate the previously referred MitM attacks. By making more routes pass through the attacking node, it can also correlate information to gain insights on what other nodes are doing, possibly gaining an advantage in the following bidding sessions. This attack can be countered by using routing metrics that are not falsifiable by the attacker, such as ping or an authenticated location.

Sybil attack This is an attack where an adversary creates multiple fake identities in a network in order to manipulate the network in it's favor. It may be used to facilitate Routing attacks and by consequence MitM. This could be used to isolate a node from the network, creating a ficticious network around it fully controlled by the attacker. This attack can be prevented by authenticating each member of the network against some proof of humanity, this proof being done outside of the system, and inserted in it on a new node join.

Message Repudiation A Message Repudiation attack is one where a malicious node sends a certain message and later denies that was the case. This could be used to flood the network with buy orders at a high price, and then deny responsability of doing so, causing biddings to be useless. This is another attack defeated by authenticating each message, so other nodes could conclude that no one else but the malicious node could have generated these messages.

2.5.1 Cryptography

Understanding the basics of cryptography is necessary for developing a secure distributed system. This is not meant to be a thorough explanation of the mathematical constructions behind cryptography as that would be outside the scope of this work, we will only describe these mechanisms through the lens of a user of such mechanisms.

There are two main types of cryptographic mechanisms, symmetric and asymmetric.

Symmetric cryptography, also known as shared secret cryptography, uses a single key for both encryption and decryption of data. The same key is used to encrypt and decrypt data, making it necessary for both parties involved in the communication to have access to it.

Asymmetric cryptography, also known as public key cryptography, uses a pair of keys – a public key and a private key – to encrypt and decrypt data. The public key is generally used to encrypt data and can be shared freely, while the private key is kept secret and used to decrypt data. The security of the encryption is ensured through the use of mathematical algorithms that make it computationally infeasible for anyone to determine the private key from the public key. The first public example of this type of cryptography was Diffie-Hellman key exchange [8].

The roles of these keys get reversed when talking about digital signature schemes. A digital signature is commonly created by encrypting a hash of a message with the private key of the sender, which can then be verified by the recipient using the public key. This ensures that the message has not been altered in transit and proves that the message came from the sender who claims to have sent it. In this way, digital signatures provide a secure method of authentication in a decentralized market. The first public example of a digital signature scheme was RSA, detailed in the [27] article.

Another relevant concept is that of key derivation. This is a mechanism through which we can generate new keys from a pre existing key in a way which allows us to later verify that the second key came from the original, a popular mechanism used for this is HMAC [1]. This is useful to generate ephemeral keys from existing private keys. Ephemeral keys are keys which are uniquely generated for each connection.

As non experts in cryptography we should use established cryptographic constructions and not attempt to roll our own, as if they are not fully understood they will lead to security issues. A common issue is the vulnerability to length extension attacks [9].

The authentication process in a decentralized system can be achieved through the use of private keys embedded in each node. These keys serve as a guarantee of the origin of a message by signing it, providing a secure method of authentication. This is achieved through the application of asymmetric cryptography, which differs from symmetric encryption in that it utilizes two keys – a private key and a public key – instead of a single key for encryption and decryption [8]. Through this method, messages can be signed with the private key and the public key can be published, allowing other nodes to verify the source of the message [26].

2.6 Decentralized Markets

Decentralized markets refer to systems of free trade that operate in a decentralized manner, where peers may trade directly with eachother without needing to trade through a central authority. Research on this topic is mostly related to the game theory aspect [24, 33, 36], however, we do not intend to explore this side of the topic extensively, we will instead focus on the communcation aspect of decentralized markets. [10] provides a comprehensive overview of the state of the art in the specific case of decentralized energy markets.

In our proposed solution, we will consider two stakeholders - the users and the resource operator. The aim of the users is to obtain their required resources, such as electricity, at the lowest possible economic cost, while the resource operator serves as the market regulator. The operator is responsible for certifying new users and acting as a backup in situations where resources cannot be traded locally among users. Furthermore, the role of the operator involves processing payment transactions and collecting user information in the process.

The market operators are considered a trusted party, as it is in their interest that their market continues operating in a trustworthy way. The operators want clients to join their market and the lack of trust in that same market would be a deteriment to their success.

Clients are not trusted as they may perform malicious actions such as offering enormous quantities of resources without intending to fulfil the offer. However, we do not have to guarentee that these actions are impossible, only that they are detectable, as misbehaving clients will be expelled from the network, and any single malicious action will not have long lasting consequences for the network.

In terms of a decentralized market, the approach described in 2.5.1 enables the network operator to onboard new nodes by signing their keys. Additionally, a certificate from the network operator should be obtained to verify the validity of the key. This eliminates the possibility of bad actors creating multiple self-signed keys in order to evade expulsion following misbehavior. The use of private and public keys in a decentralized market guarantees the authenticity of the sender, ensuring the security of the network.

2.7 Privacy

In the previous section we have already approached issues inside of the network, now we will approach issues which might affect users outside of it. Privacy is an important matter, as divulging our need of a certain resource might benefit competitors by indirectly informing them of our own plans, e.g. in some sort of store we could order a lot of a certain product as we predict it to be in large demand soon, pushing our competitors to do the same, it is also a good proxy for if people are in a building, meaning thieves could monitor our energy usage and rob our homes when they noticed there was little energy being consumed, it could also signal issues with our services, e.g. abnormally large volumes of requested energy at a data center might mean our servers are overloaded, and it could be an opportunity for another similar service to advertise to our customers. Our solution will have to deal with the privacy of its users, finally, an abnormal request of energy at some office could mean people are working overtime on something large, such as the story told in [35].

In a peer-to-peer network, messages are expected to be routed through intermediaries to reach their final destination, creating a potential security vulnerability. These intermediaries may eavesdrop on the communication or execute a man-in-the-middle attack. To prevent this we should consider some form of encryption scheme, where only the intended receiving node would have the necessary information to decrypt the incoming message.

The usage of a topic based publish/subscribe message dissemination scheme brings even more challenges, where to efficiently route messages, a single message must be decodable by multiple receiving it. This could trivially be solved by sending a single large message containing multiple individual messages, each one encrypted with the public key of each receiving node, but this would incurr in massive overhead, as the size of our large message would increase linearly with each new node that subcribed to that particular topic.

An alternative solution to the issue of third-party sniffing in a peer-to-peer system can be seen in GnuPG [25] implementing the OpenPGP [12] message format. This method involves encrypting the message with a symmetric key and then encrypting the symmetric key with each recipient's public key and adding the result to the message. This ensures that only the intended recipients can access the message by decrypting the symmetric key with their private key, followed by decryption of the message. This method still has the overhead of including one version of the encrypted symmetric key per receipient, which is not ideal if we intend to disseminate the message to a very large quantity of nodes.

Yet another possible solution is a modification of the version above, where when a new node subcribes to a topic it is given that shared symmetric key through a message directed only at them, encrypted with their public key, and when a node unsubscribes, a new key is distributed through the topic, encrypted with every one of the remaining subcribed nodes' public key. This method only incurs in significant overhead for node unsubcribing, as we have to invalidate that node's access to the broadcasts.

3 Future Work

This chapter outlines the future steps for the work described in this document. In Section 3.1, the problem outlined in Chapter 1 will be given a more detailed explanation. In Section 3.2, a high-level overview of the proposed solution is presented. Finally, in Section 3.3, a method for evaluating the efficacy of the proposed solution is described.

3.1 Problem Description

As outlined in Chapter 1, the majority of the research conducted on decentralized markets does not take into account the communication aspect of such systems. Our focus, however, lies in exploring this aspect. While some studies have addressed the inherent challenges in this type of system, such as preventing fraudulent activity while simultaneously being scalable, most solutions have primarily relied on blockchain approaches [29], which are not suitable for our intended uses. Therefore, it is imperative to develop a new solution that meets the specific requirements of decentralized markets.

These requirements are:

1. Running in a layered decentralized fashion, where there is a separation between a bottom client layer and an upper network operator controlled layer;

2. Minimizing the communication with upper layers as to promote stability in the event of network outages;

3. Having the possibility of falling back to a single network operator present in upper layers;

4. Being able to effectively, securely, and privately offer and acquire resources;

5. The solution should optimize for some arbitrary link cost metric, as this will mirror real life costs of the transportation of these goods.

3.2 Proposed Solution

We now outline the main aspects of a proposed solution which involves a modified version of the protocols originally proposed in [18], work which has been detailed in Section 2.4.4. The mentioned work has several of the properties we are looking for, but it does not deal with the massive scale our work intends to deal with. There is also an issue with the fact that the prototype developed in the overlay put forward by Legion [19] does not feature configurable cost metrics for the links that are established among participants.

We trust that a modified version of the forementioned work as our service layer, with federated servers controlled by the network operator substituting the central authority mentioned in the article, using X-Bot [17, 15] with an authenticated link metric mechanism, in order to prevent routing manipulation attacks, as the overlay layer of the system will be a fitting solution for the proposed problem.

The proposed system will be structured into three tiers: the first tier comprises untrusted client nodes, the second tier is managed by the network operator and the third tier is a logically centralized layer, also controlled by the network operator, that provides support to the second tier.

The lower layer of our solution will be built with a secure causal consistency model, leveraging A. Linde et al's work. Do remember that all client-generated messages will have their dependencies and digital signature attached. Since nodes will generally only have the central public key and not other node's they will be able to verify that the signature is valid, but not which node signed it. An extra step to take in order to prevent correlation attacks involves deriving an ephemeral key from the node's key, and using that to sign the messages instead.

3.2.1 Implementation Details

We will now provide additional details on the operation of our solution.

On initial setup signed individual keys will be transmitted from the central server for each registering node, a key A for use in its own authentication, as well as a signed verified property, that will be used as a biasing metric for our modified X-Bot. For ease of reference and intuition, we will refer to this signed property as the node's location, L.

Our work will differ from original X-Bot with the use of authenticated optimization rounds. X-Bot's link cost will be given as a function of the property L of both nodes. By transmitting the signed L along with the optimization round request, the receiving node can verify the distance metric. This is necessary to prevent eclipse attacks and other topoligy manipulation. Without this mechanism, by maliciously requesting optimization rounds with a large amount of nodes an attacker could significantly increase the amount of traffic routed through itself at the deteriment of the real total overlay cost.

The issue of scalability is addressed via the use of region locking and the use of multiple containers, as detailed in [19], this means nodes will not accept information that originated further than distance *D*, denoting a distance as calculated by the X-Bot metric, allowing each container to stay inside a certain zone, avoiding the tremendous cost of synchronizing the full network.

Separation of groups that require causal consistency will be achieved with the use of containers as detailed in Legion. These are a way to group relevant events in the same causal chain. A new container will be used for each offer's communication. This intuitively prevents possible deadlocks caused by our distance limiting, as there will be no case where an event relevant to a node wants to receive depends on another event outside of it's range, since each causal group has a single region limit.

The trading process within the system will be made explicit through the following mechanisms.

1. For creating a trading offer, a node creates a new container starting with a message including the offer's terms, the node's *L* and a generated public key *PK*.
2. To accept this offer a node will add another event to the container with the offer's id encrypted with *PK*.
3. When the confirmation mentioned above is received by the offering node the node may choose to accept it. If this is the case the node will produce a final message accepting the offer, which will serve as confirmation to the purchasing node. This step should have a fixed event identifier to leverage Linde et al's work in sibling event detection in order to prevent a malicious node from accepting two confirmations at once.

Notice how this same mechanism may be used for both offering and requesting resources. Just like a node that wants to sell its resources may produce an offer on our solution, a node which needs a resource, it may create offer to buy such resource.

Our solution ensures that the integrity of the system is maintained, as all events are recorded and signed, making it impossible for a node to deny having sent a particular message.

3.3 Evaluation

The evaluation of our solution will be carried out utilizing the OPAL grid simulation environment [23] provided by our partner: the company EDP New R&D.

As previously explained, a baseline centralized version, with all operations passing through a central node will be implemented and benchmarked. We will also benchmark our proposed solution, in similar conditions. The particular details about numbers of nodes, their latencies, and workloads will be provided in the future by EDP New R&D.

A basic policy for the generation of offers and their acceptance of them implemented. That policy will involve generating a "fair price" for each node, consisting in a predefined value modified by a small random factor. When deciding on accepting an offer, that node's

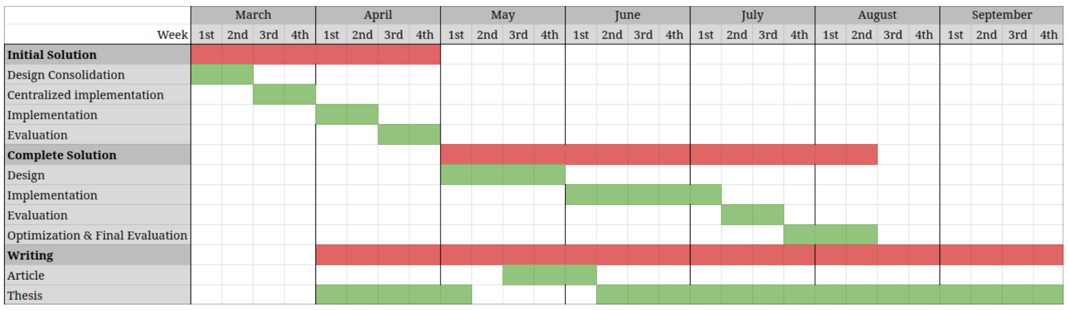

Figure 3.1: Gantt of expected task durations

"fair price" will be slightly randomly altered, and if the offer is more than a configurable fraction positive for that node it will be accepted. This fraction parameter is necessary so that offers are accepted less than half of the time and useful values will be obtained experimentally.

We will measure these solutions on a few different metrics.

1. Latency will be measured in our simulator by disseminating offers with a timestamp, replying to them, and measuring the time delta. This is doable as the simulated environment features a global clock.

2. Last delivery hop will be measured similarly to the latency, counting the hops taken by the message with the most hops that ended up being useful.

3. Reliability will be measured by producing increasingly larger node disconnections in the system and measuring the decrease in welfare generated. Welfare is defined as the sum of the difference between the least beneficial price that would be accepted, and the actual price that was accepted.

4. Scalability will be measured by increasing the number of nodes and measuring latency and the rate of satisfied requests.

5. Overlay efficiency will count how many messages were sent and how many proved useful.

6. General performance with different usage patterns will be measured by exploring other offer generation and acceptance policies and measuring their generated welfare.

3.4 Schedule

The scheduled work is split into the following main tasks:

Initial Solution This involves refining the suggested solution put forward in this work, as well as implementing a baseline centralized version of the protocol for us to use in performance comparisons. We will then implement the solution suggested in this work and benchmark it against the base approach.

Complete Solution With the initial benchmarks and newfound knowledge derived from our experimentation with the initial solution, we will make improvements on it, implementing them, and once again benchmarking this new solution against the base approach.

Writing We will produce an article to be published at an international conference and write the dissertation document describing the work developed.

Bibliography

[1] E. Barker and Q. Dang. *The Keyed-Hash Message Authentication Code (HMAC)*. en. 2002-03 (cit. on p. 17).

[2] G. Blair. “Complex Distributed Systems: The Need for Fresh Perspectives”. In: *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. 2018-07, pp. 1410–1421. DOI: 10.1109/ICDCS.2018.00142 (cit. on p. 1).

[3] A. Brown. 2020. URL: https://www.pewtrusts.org/en/research-and-analysis/blogs/stateline/2020/01/09/electric-cars-will-challenge-state-power-grids (visited on 2023-02-09) (cit. on p. 2).

[4] Cisco. *Cisco Annual Internet Report (2018–2023) White Paper*. 2020-03. URL: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html (visited on 2023-02-02) (cit. on p. 1).

[5] T. Consortium. *TaRDIS Internal Planning Document*. 2023 (cit. on p. 3).

[6] P. A. Council. *2021 Public Affairs Pulse Survey Report*. 2021. URL: https://pac.org/wp-content/uploads/Pulse_2021_Report.pdf (visited on 2023-02-02) (cit. on p. 1).

[7] A. Datta et al. “Anonymous publish/subscribe in P2P networks”. In: *Proceedings International Parallel and Distributed Processing Symposium*. 2003-04, p. 8. DOI: 10.1109/IPDPS.2003.1213174 (cit. on p. 10).

[8] W. Diffie and M. Hellman. “New directions in cryptography”. In: *IEEE Transactions on Information Theory* 22.6 (1976), pp. 644–654. DOI: 10.1109/TIT.1976.1055638 (cit. on p. 17).

[9] T. Duong and J. Rizzo. “Flickr’s API Signature Forgery Vulnerability”. In: 2009 (cit. on p. 17).

[10] K. Erdayandi, L. Cordeiro, and M. Mustafa. “Towards Privacy Preserving Local Energy Markets”. English. In: *Competitive Advantage in the Digital Economy (CADE 2022): Resilience, Sustainability, Responsibility, and Identity*. 2022-03 (cit. on p. 18).

[11] F. Falchi, C. Gennaro, and P. Zezula. "A Content–Addressable Network for Similarity Search in Metric Spaces". In: *Databases, Information Systems, and Peer-to-Peer Computing*. Ed. by G. Moro et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 98–110. ISBN: 978-3-540-71661-7 (cit. on p. 13).

[12] H. Finney et al. *OpenPGP Message Format*. RFC 4880. 2007-11. DOI: 10.17487/RFC4880. URL: https://www.rfc-editor.org/info/rfc4880 (cit. on p. 19).

[13] *Gnutella 0.6 RFC*. 2002. (Visited on 2023-02-02) (cit. on p. 6).

[14] J. C. A. Leitao, J. Pereira, and L. E. T. Rodrigues. "HyParView: A Membership Protocol for Reliable Gossip-Based Broadcast". In: *37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07)* (2007), pp. 419–429 (cit. on p. 6).

[15] J. Leitão et al. "X-BOT: A Protocol for Resilient Optimization of Unstructured Overlays". In: *Proceedings of the 28th IEEE International Symposium on Reliable Distributed Systems*. Niagara Falls, New York, U.S.A., 2009-09, pp. 236–245 (cit. on pp. 13, 22).

[16] J. Leitão. "Topology Management for Unstructured Overlay Networks". PhD thesis. Technical University of Lisbon, 2009 (cit. on p. 7).

[17] J. Leitão et al. "X-BOT: A Protocol for Resilient Optimization of Unstructured Overlay Networks". In: *IEEE Transactions on Parallel and Distributed Systems* 23.11 (2012-11), pp. 2175–2188. ISSN: 1558-2183. DOI: 10.1109/TPDS.2012.29 (cit. on pp. 13, 22).

[18] A. van der Linde, J. C. A. Leitao, and N. M. Preguiça. "Practical Client-side Replication: Weak Consistency Semantics for Insecure Settings". In: *Proc. VLDB Endow.* 13 (2020), pp. 2590–2605 (cit. on pp. 14, 22).

[19] A. van der Linde et al. "Legion: Enriching Internet Services with Peer-to-Peer Interactions". In: *Proceedings of the 26th International Conference on World Wide Web*. WWW '17. Perth, Australia: International World Wide Web Conferences Steering Committee, 2017, pp. 283–292. ISBN: 9781450349130. DOI: 10.1145/3038912.3052673. URL: https://doi.org/10.1145/3038912.3052673 (cit. on pp. 14, 22).

[20] P. Maymounkov and D. Mazières. "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric". In: *Peer-to-Peer Systems*. Ed. by P. Druschel, F. Kaashoek, and A. Rowstron. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 53–65. ISBN: 978-3-540-45748-0 (cit. on p. 10).

[21] P. Maymounkov and D. Mazières. "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric". In: *Peer-to-Peer Systems*. Ed. by P. Druschel, F. Kaashoek, and A. Rowstron. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 53–65. ISBN: 978-3-540-45748-0 (cit. on p. 12).

[22] V. H. Menino. "A novel approach to load balancing in p2p overlay networks for edge systems". MA thesis. NOVA School of Science & Technology, 2021-11 (cit. on p. 8).

[23] OPAL-RT. *Innovation-Ready Real-Time Digital Simulator*. 2023. URL: https://www.opal-rt.com/power-systems-overview/ (visited on 2023-02-09) (cit. on p. 23).

[24] A. Paudel et al. "Peer-to-Peer Energy Trading in a Prosumer-Based Community Microgrid: A Game-Theoretic Model". In: *IEEE Transactions on Industrial Electronics* 66.8 (2019-08), pp. 6087–6097. ISSN: 1557-9948. DOI: 10.1109/TIE.2018.2874578 (cit. on p. 18).

[25] G. Project. *GNU Privacy Guard*. 1999. URL: https://gnupg.org/ (cit. on p. 19).

[26] R. L. Rivest, A. Shamir, and L. Adleman. "A method for obtaining digital signatures and public-key cryptosystems". en. In: *Commun. ACM* 21.2 (1978-02), pp. 120–126 (cit. on p. 17).

[27] R. L. Rivest, A. Shamir, and L. Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". In: *Commun. ACM* 21.2 (1978-02), pp. 120–126. ISSN: 0001-0782. DOI: 10.1145/359340.359342. URL: https://doi.org/10.1145/359340.359342 (cit. on p. 17).

[28] A. Rowstron and P. Druschel. "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems". In: *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*. Vol. 2218. Springer Berlin Heidelberg, 2001-11. Chap. Middleware 2001, pp. 329–350. ISBN: 978-3-540-45518-9. URL: https://www.microsoft.com/en-us/research/publication/pastry-scalable-distributed-object-location-and-routing-for-large-scale-peer-to-peer-systems/ (cit. on p. 10).

[29] A. Samy et al. "SPETS: Secure and Privacy-Preserving Energy Trading System in Microgrid". In: *Sensors* 21.23 (2021). ISSN: 1424-8220. DOI: 10.3390/s21238121. URL: https://www.mdpi.com/1424-8220/21/23/8121 (cit. on p. 21).

[30] I. Stoica et al. "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications". In: *SIGCOMM Comput. Commun. Rev.* 31.4 (2001-08), pp. 149–160. ISSN: 0146-4833. DOI: 10.1145/964723.383071. URL: https://doi.org/10.1145/964723.383071 (cit. on p. 12).

[31] *TaRDIS: Trustworthy and Resilient Decentralised Intelligence for Edge Systems*. URL: https://cordis.europa.eu/project/id/101093006 (visited on 2023-01-26) (cit. on p. 2).

[32] S. Thangaraju and K. Munisamy. "Electrical and Joule heating relationship investigation using Finite Element Method". In: *IOP Conference Series: Materials Science and Engineering* 88 (2015-09), p. 012036. DOI: 10.1088/1757-899X/88/1/012036 (cit. on p. 2).

[33] W. Tushar et al. “Grid Influenced Peer-to-Peer Energy Trading”. In: *IEEE Transactions on Smart Grid* 11.2 (2020), pp. 1407–1418. DOI: 10.1109/TSG.2019.2937981 (cit. on p. 18).

[34] Q. H. Vu, M. Lupu, and B. C. Ooi. *Peer-to-peer computing*. en. 2010th ed. Berlin, Germany: Springer, 2009-11, pp. 12–16 (cit. on p. 6).

[35] warinner. *The Pentagon Pizza meter*. 2000. URL: http://home.xnet.com/~warinner/pizzacites.html#time (cit. on p. 19).

[36] W. Wei, F. Liu, and S. Mei. “Energy Pricing and Dispatch for Smart Grid Retailers Under Demand Response and Market Price Uncertainty”. In: *IEEE Transactions on Smart Grid* 6.3 (2015), pp. 1364–1374. DOI: 10.1109/TSG.2014.2376522 (cit. on p. 18).