NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

DIOGO FILIPE MARCOS BARRETO

BSc in Computer Science

GENERIC DECENTRALIZED MEMBERSHIP AND COMMUNICATION ABSTRACTIONS FOR EDGE SYSTEMS

Dissertation Plan
MASTER IN COMPUTER SCIENCE

NOVA University Lisbon
February, 2023

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTMENT OF
COMPUTER SCIENCE

GENERIC DECENTRALIZED MEMBERSHIP AND COMMUNICATION ABSTRACTIONS FOR EDGE SYSTEMS

DIOGO FILIPE MARCOS BARRETO

BSc in Computer Science

Adviser: João Carlos Antunes Leitão
Assistant Professor, NOVA University Lisbon

Co-adviser: Nuno Manuel Ribeiro Preguiça
Associate Professor, NOVA University Lisbon

Dissertation Plan
MASTER IN COMPUTER SCIENCE

NOVA University Lisbon
February, 2023

Abstract

In today's world many systems rely on decentralized architectures as a way to provide services to many users that require high availability, fault tolerance, and scalability. Avoiding a centralized component allows avoiding both bottlenecks and a single point of failure making these systems more autonomous and robust. One very common application for these architectures are systems that require some form of coordination between different actors in order to perform a particular task. Examples of these systems can be swarms of satellites, IoT appliances, industrial machines, clusters of computers processing large amounts of data, among others.

Decentralized architectures need to provide two basic functionalities: ways to allow the management of peers participating in the system, commonly called membership management, and ways to allow peers to communicate between themselves in order to send and receive information. Many solutions for these requirements were already developed in the context of peer-to-peer systems specially when considering the development of overlay networks, but these solutions often fall short by relying on mechanisms that are specific for a given scenario thus requiring extensive modifications when used in order to develop different applications or cannot be used at all when there is need to be applied to systems with different necessities or even when the initial conditions of the system suffer modifications.

In this work existing solutions will be evaluated focusing on their design choices, main advantages and disadvantages, application scenarios, and performance results. Then, leveraging the lessons learned with the evaluation of these existing solutions, we will propose a generic solution for supporting multiple decentralized systems with different requirements by taking into consideration the configurations and/or information obtained by nodes during the system operation. In order to study systems with different requirements that can benefit from this novel approach, use cases from the TaRDIS European project will be used in the context of the work being developed.

Keywords: Distributed systems, Decentralized systems, Overlay networks, Edge computing, Peer-To-Peer systems

Resumo

No mundo atual vários sistemas baseiam-se na utilização de arquiteturas descentralizadas para fornecer serviços com garantias de alta fiabilidade, disponibilidade e escalabilidade. Evitar a utilização de um componente centralizado permite evitar pontos de estrangulamento bem como a existência de um único ponto de falha, tornando estes sistemas mais autónomos e robustos. Uma das aplicações mais comuns para este tipo de arquiteturas são sistemas que necessitam de alguma forma de coordenação entre os diferentes atores que os compõem para realizar uma determinada tarefa. Estas tarefas podem tratar-se de coordenação entre grupos de satélites, dispositivos IoT, máquinas industriais ou computadores responsáveis pelo processamento de grandes quantidades de dados, entre outros.

De uma forma geral, sistemas baseados em arquiteturas descentralizadas necessitam de fornecer duas funcionalidades fundamentais: funcionalidades que permitem gerir os nós que participam no sistema, habitualmente chamadas gestão de filiação (*membership management*), e funcionalidades que permitem a comunicação entre esses mesmos nós de forma a possibilitar a troca de informação. Apesar de diversas soluções já existirem, em particular quando se leva em consideração os sistemas entre-pares (*peer-to-peer*) estas soluções são habitualmente concebidas considerando as necessidades específicas de um sistema, não sendo, desta forma, capazes de se adaptar a sistemas com diferentes necessidades ou até mesmo quando as condições iniciais de um sistema mudam necessitando, nestes casos, de modificações profundas ou podendo mesmo a sua utilização ficar impossibilitada.

No trabalho que pretendemos desenvolver, diversas soluções já existentes serão avaliadas considerando o seu desenho, os principais pontos positivos e negativos, cenários de aplicação e desempenho. De seguida, considerando o estudo realizado das soluções existentes, uma nova solução genérica será apresentada capaz de suportar múltiplos sistemas descentralizados com diferentes necessidades, considerando as configurações realizadas para cada sistema e/ou dados obtidos durante a sua operação. De forma a considerar os diferentes requisitos, casos de uso relacionados com o projeto europeu TaRDIS serão utilizados como motivadores para este trabalho.

Palavras-chave: Sistemas distribuídos, Sistemas descentralizados, Sistemas entre-pares

Contents

List of Figures

List of Tables

1 Introduction

Distributed architectures are nowadays widely used as a way to build robust, scalable, and fault-tolerant systems. We can define a network, the Internet for example, as a set of interconnected machines (or processes) able to communicate between themselves [26].

Systems that operate on top of a network can take advantage of a more centralized or decentralized approach. Whereas on a centralized system all information needs to be sent to a central point, like a data center, in order to be processed making this central component responsible for a significant part of the operation, in a decentralized architecture the nodes present on the network can work together and cooperate, by sending and receiving messages directly among themselves, in order to perform the required operation. Steen and Tanenbaum discuss in [26] the difference between a *distributed system* and a *decentralized system* stating that both may rely on multiple machines performing a certain operation but in the first case "processes and resources are sufficiently spread across multiple computers", as an example an email service can rely on multiple servers as a way to distribute load and improve fault-tolerance, yet on the second case "processes and resources are necessarily spread across multiple computers" making the distribution of processes a core aspect of the system.

A decentralized approach offers many advantages when compared to a single process or a group of autonomous processes with no communication between themselves sending and receiving information to/from the same central location. These advantages include improved availability by avoiding single points of failure, because a node can substitute another one in the execution of a given task, better scalability by opening possibilities for clients and data to be distributed across nodes [20, 26], improved reliability with the possibility of replicating information in different nodes minimizing the risk of data loss or even privacy and malicious attack tolerance due to the lack of a single target and the possibility of using the network of nodes to hide the identity of users or the information exchanged in the system [10]. On the other hand centralized systems can be considered easier to maintain as information only needs to be sent to a central point and, as such, they do not have to deal neither with the heterogeneity questions related with the characteristics of each node nor the membership management and communication issues regarding the

distributed nature of the infrastructure.

Distributed systems can vary from large sets of globally interconnected machines and processes relying on proprietary network infrastructures, and in many cases running in the same data centers, when considering large companies like Google, Microsoft, or Amazon [2]; to smaller scale networks of connected devices based on already existing networks like the Internet.

A particular type of network architectures are called Peer-To-Peer networks or P2P. The idea behind P2P consists on directly connecting a set of computers among themselves in order to allow direct information exchange and coordination which can be done by leveraging an underlying network on top of which connections, like TCP channels, are open between machines. Although a *global view* of the system could be maintained this would require every node to know about every other peer in the system leading to problems in dynamic network environments where a high number of nodes entering and leaving the system is expected as this information would need to be constantly updated. In order to deal with this challenge a common solution is to allow nodes to only maintain connections to a small set of other peers, i.e., relying on a *partial view* of the system. If these connections are defined at the application level they define structures known as *Overlay Networks* which can act as the membership protocol for a P2P system [20].

In overlay networks, a process should maintain a set of other nodes to whom connections are established thus being considered as its neighbors. The connections between nodes then form a graph that allows messages from one node to reach, eventually with the help of other ones, any process on the network. Generally it is possible for new nodes to join a pre-existent overlay network by contacting current participants [20]. These networks can vary from numerous connected users cooperating, for example, to process large amounts of data or to share information in a decentralized way (as in the BitTorrent [33] protocol) to smaller number of nodes cooperating in the execution of a given task like Internet Of Things (IoT) sensors in a house sending information between themselves, the industrial machines on a factory exchanging performance data or swarms of satellites in space communicating and autonomously adjusting their positions to avoid collisions.

In 2020, according to Forbes [34], 59 zettabytes of data were "created, captured, copied, and consumed in the world", which compares to the 33 zettabytes two years early. These large amounts of data that are produced and need to be processed, alongside with the need for a faster and more efficient processing of such data, paved the way for the usage of what is called *edge computing* allowing for computations to be executed closer to clients and outside big data center facilities [23, 27]. This approach allows for faster response times for clients and, at the same time, reduces the load induced in each component of the network from machines in data centers, to the amount of traffic in networks [9]. As we will describe ahead, the usage of peer-to-peer architectures that take advantage of the interconnected devices can play a significant role in the development of new approaches to edge computing as these devices can be themselves considered more "on the edge" of the network and closer to end-users.

1.1 Motivation

Independently of what type of distributed system is considered two main capabilities should be guaranteed to enable its operation: i) mechanisms for membership management (or membership protocols [20]) that are responsible for managing the nodes that are part of the system, the ones entering and the ones leaving either by failure or explicit request and ii) mechanisms for allowing nodes to communicate between themselves in order to coordinate and exchange information.

Many protocols have been proposed, in particular in the context of peer-to-peer architectures, to deal with both the membership management and support for different communication primitives between peers, but most of them fall short by addressing very specific challenges or by requiring particular operational circumstances. This means that usually specific abstractions that are used to provide membership management or communication mechanisms can be hard to be reused to work in a performant way when other requirements are needed thus requiring in many cases the development of a totally different solution. Possibly even more problematic is the incapacity of these abstractions to adapt if the conditions of the network suffer changes during the operation of the system they support thus impacting the membership management and/or the communication performance under the new circumstances.

In particular, when considering peer-to-peer systems, overlay networks can be used, as described previously in this chapter, in order to address the challenges related with the system membership. In fact, various examples of overlays networks have been proposed in the context of P2P architectures such as Chord [42], Kademlia [30], Freenet [10], Overnesia [24], among others with two main types arising based on the way peers are organized and connected between themselves forming the topology of the network: *structured* and *unstructured* overlay networks. The first ones are defined by enforcing specific patterns on the connections between peers thus leading the overlay network to evolve until a specific topology, usually more beneficial to the task that needs to be performed by the network, is reached (nodes can form a ring, a tree, among others). The second ones do not enforce any type of network topology, allowing nodes to organize freely in a flexible way [20, 27].

Situations can also arise where a distributed system is constituted by a heterogeneous set of nodes with specific characteristics [23] in which some subsets of nodes might benefit from using different strategies for membership management and/or communication in order to work more efficiently. Some examples will be presented in Section 2.2 of this document. These situations can become increasingly common in the future, as more nodes are expected to be connected in even more complex networks covering multiple types of devices with different characteristics like sensors to measure vehicles and houses' energy consumption in an energy management network. These cases, where nodes with significantly different characteristics, guarantees, and needs are expected in the same network, can pose a challenge because the usage of the same specific methods

for membership management and communication on all peers can lead to performance issues on some of them when the method is not adequate to its particular properties or operational conditions. On the other hand, the usage of different approaches in different subsets of network peers, taking into consideration the nodes' heterogeneity, can pose problems related with the ability to interact and cooperate across those segments of the system.

As discussed previously, the motivation for the development of this work comes from the need for the development of new generic abstractions that allow performing the basic operations required by a decentralized system on the edge of the network, namely the ones related with membership management and communication, using strategies more suitable for each set of nodes with particular characteristics yet allowing all nodes to communicate using the same generic abstractions with all other nodes independently of the concrete strategies employed underneath.

1.2 Expected contributions

The main contributions expected from the elaboration of this work are now discussed. First a study will be performed of existing solutions for membership management and communication strategies, in the context of peer-to-peer networks and edge computing, to identify their positive and negative aspects including performance benchmarks across multiple execution scenarios. An evaluation about the suitability of each one of the studied solutions, or parts of them, for the final objective of developing generic abstractions for membership management and communication will also be performed in order to serve as base for the work being developed.

Then, new generic decentralized membership and communication abstractions that can operate effectively on different settings and under different conditions will be developed. The presented abstractions should be suitable to be used in different scenarios based on the configurations defined and/or the evolution of the network during the operation of the system. The abstractions whose objective of this dissertation is to develop can be considered as a generic *Application Programming Interface* (API) for membership management and communication operations, common to all nodes taking part in the network, that could then be implemented by leveraging multiple protocols and strategies available and configured in a per-node manner. For this development it is expected that the study described before of existing approaches contributes as a basis for integrating some aspects of existing solutions into the developed solution.

Finally, an evaluation of the solution(s) developed will be available, considering the operation of a system leveraging the developed abstractions, taking into consideration various use cases namely in terms of correctness of operation, limitations, performance, and adaptation capabilities to multiple situations.

1.3 Research Context

The work presented here has been partially motivated by the *Trustworthy and Resilient Decentralised Intelligence for Edge Systems* (TaRDIS) [44] European project aimed at "supporting the correct and efficient development of applications for swarms and decentralised distributed systems, by combining a novel programming paradigm with a toolbox for supporting the development and executing of applications".

1.4 Document Structure

Besides this introductory chapter this document is composed by:

- Chapter 2 provides insight on the necessary fundamental concepts to understand the presented work and the related work on which this dissertation is based. These concepts include a more extensive explanation of peer-to-peer systems, overlay networks, and edge computing; their capabilities and characteristics as well as a study of existing approaches with an evaluation of the strong and weak aspects of each one. The use cases in which this dissertation will be mainly based, as a way to develop and test more generic membership and communication abstractions, will also be presented in this chapter. Also, the description and study of different types of services relying on a peer-to-peer approach like broadcast and publish-subscribe services is depicted in this chapter. The chapter concludes with a general discussion that provides a more focused perspective between the solutions discussed before and the specific goals of this work.
- Chapter 3 of this document presents the future work that will be developed in order to successfully complete the objectives presented in this document as well as the expected schedule and description of tasks that will be performed.

2

Related Work

In this chapter the work that serves as basis for the research that we plan to do will be presented and discussed. First the aspects related with peer-to-peer architectures (P2P) and edge computing are introduced followed by the presentation of some use cases on which the work and subsequent evaluation will be based. Then we provide an overview of what are overlay networks, existing architectures, characteristics, and properties identifying the main positive and negative aspects of each one, while also taking into consideration the previously described use cases. We further discuss several services which can be provided by relying on peer-to-peer architectures and overlay networks like publish-subscribe or broadcast services. A study of existent peer-to-peer systems is also presented. At the end of this chapter, a discussion regarding the presented related work is provided.

2.1 Edge Computing and Peer-To-Peer architectures

In this section both the edge computing paradigm and peer-to-peer architectures will be presented more in-depth, as well as a study on how systems can leverage the combination of both approaches.

2.1.1 Edge Computing

As described earlier in this document, the development of systems that rely on edge computing paradigms should be considered in the world of today as a way to tackle i) the significant increase in the quantity of data that needs to be processed [34] and ii) the needs, from end users, of a fast and efficient processing of information and from systems that need to quickly detect changes in order to trigger response actions [9]. This scenario raises questions related to the scalability capacity of centralized architectures and shows the necessity of removing computations from centralized environments, like data centers, and transfer them, partially or totally, towards the edge of the network near the client devices or even, if possible, to client's devices [23, 9].

A particular example of using the network edge to accelerate computational tasks, especially when considering IoT devices, is called *Fog Computing* [27, 23, 9]. This approach

considers the usage of nodes (*fog nodes*) available in the network between the client and large scale data centers, based on properties like location, processing capacity, among others, to distribute the execution of a given computation thus allowing for a more expedite and efficient data processing and, at the same time, reducing the amount of data that needs to be transmitted throughout the network. In this case various types of fog nodes can be used and, as an example, data can be analyzed first in nodes near the client as a way to provide a quick result, which might be important if we consider a situation where a sensor triggers some automation based on the analysis of the information retrieved, and then sent to a data center for gathering and statistic treatment [9]. Mechanisms considering fog computing can also play a role in situations where privacy concerns exist with critical data, for example, medical information, being analyzed in the nearest nodes with only more generic and anonymized information sent to data centers which can even help to deal with privacy regulations at a regional level [27, 23].

An evolution of fog computing consists in doing the computations on the actual devices that retrieved the information by taking advantage of their embedded processing capabilities. This evolution is called *Mist Computing* [29] and by itself does not imply that some data cannot be sent to fog nodes and/or central processing data centers for further analysis and aggregation.

The distribution of computation between machines on the edge of the network relying on a decentralized architecture is not something new as it has been employed for decades in projects like the SETI@Home [1] to allow computers to cooperate in processing large amounts of scientific data with the objective of studying the existence of extraterrestrial life by processing radio signals retrieved from space. This project is an interesting case study as it effectively uses end-user devices, like computers, to help with the processing of large amounts of data without the need of central processing. In this case a central server was effectively used for distributing the computation between client devices as the data is retrieved by radio telescopes and, as so, a central point of operation needs to exist in order to provide the information, yet, this is an important example of how devices on the edge of the network can be used, relying on decentralized mechanisms, to process large amounts of information. When situations are considered where data sources are in itself decentralized (like in the case of IoT devices) this approach can even be more interesting. Notably, the computation model employed in SETI@Home (and similar projects) is composed of small and fully independent computational tasks (named *embarrassingly parallel*). More complex computations require more complex approaches in order to be executed on the edge.

One of the challenges that appear when considering edge computing paradigms is related with the heterogeneity that may exist due to significant differences between the characteristics of nodes in terms of processing power, storage capacity, network capabilities, reliability, among others [23, 29]. As an example, if ground based devices and satellites are put together in the same network it is possible that the first ones have more processing power and reliability guarantees than the second ones, considering that the satellites

communicate via radio waves. This heterogeneity makes difficult the management of the devices as some protocols may leverage one type of devices in detriment of the others. Some studies [23] even consider the separation of the devices in a set of categories based on their characteristics, forming levels of devices where the first level is constituted by machines "in the center" of the network with high processing capacity and availability, like data centers, and last level by devices on the edge with limited availability and varied performance, like IoT sensors. To allow devices to be divided into levels, as presented before, authors suggest the study of a set of characteristics such as processing and storage capabilities, availability guarantees, domain (if the device is in the application or user domain) and computational capabilities (generic, data aggregation or filtering). Some of these aspects will also be widely used in the context of this work to characterize the different use cases. The expected increase in technical limitations of devices more close to clients emphasizes even more the need to develop methods that can cope and be performant even in presence of significant heterogeneity as the computational resources can be limited.

2.1.2 Peer-To-Peer architectures

Peer-to-peer architectures allow for devices to be interconnected without the need for a centralized point of coordination. This is achieved by having nodes directly connected to a set of known neighbors with which they cooperate in order to perform a given operation, thereby removing the need to contact a central point like a server. This connection between peers generally happens on top of an existing network, like the Internet, effectively creating what is called an *overlay network*. As will be discussed later in this chapter, there are situations in which a peer does not even need to know all the other participants in the network and relies on a partial view where, although it is only connected to a subset of all participants, it should be able to communicate throughout the entire network by, for example, having neighbor nodes to relay messages to their respective neighbors and so on until reaching the intended destination. One of the most well known protocols leveraging on a P2P architecture is the BitTorrent protocol in which peers communicate between themselves to perform file transfers in a distributed manner, with files being divided into chunks that can then be traded between peers and combined to reconstruct the complete file [33, 27].

Peer-to-peer networks have many advantages when compared with centralized solutions, namely when considering that they can overcome challenges related with the existence of a single point of failure/attack thus possibly improving security and reliability. Privacy questions can also be tackled as their decentralized nature allows for data to be dispersed and processed along the network instead of doing so using machines in central locations like data centers. Some P2P networks, like the one proposed in the Freenet [10] system, even strive for anonymization of the data traveling between peers making it harder to know which peers have a specific segment of data or which ones are

looking for it. On the other hand not relying on a central server means that for creating, maintaining, and operating a peer-to-peer system, specialized mechanisms for allowing peer management and communication are required to exist as they need to be able to manage a possibly large number of participants and allow for communication between them. Overlay networks can be effectively considered as a peer management mechanism to be used in P2P architectures [20].

Napster is considered the pioneer in peer-to-peer file sharing architectures. Created in 1999, the system brought the idea of a set of decentralized users sharing content stored on their own machines, cooperating with other network peers which could then directly download the content from the host machines without the need for a central storage server. Although this approach effectively allowed for a decentralized file-sharing system it had a centralized component, responsible for providing an index of contents available in the network and their location, effectively materializing a centralized *resource location* service. The Napster approach relied on the usage of a centralized directory server that should be contacted with the objective of finding which peers in the network had a specific resource, in order to coordinate with them and obtain the files [6]. Napster was closed due to legal reasons related to copyright and, although the legal matters of this closure are outside the scope of this document, the technical aspects that allowed this outcome should be considered, as the usage of a centralized server was effectively the technical reason that enabled the closure, as it represented a central point of failure and if offline the system was not capable of operating properly because users were unable to discover other peers with the desired content [6, 5].

It is when these mechanisms for peer management and communication are considered, that a relation can be established between edge computing and peer-to-peer systems. On one hand, as seen before, edge computing allows for computations to be transferred from central points, like data centers, to machines near the clients or even into the client device, on the other hand, peer-to-peer networks allow for multiple devices to be connected without the need of a centralized architecture. It is now easy to understand that peer-to-peer architectures can contribute to the edge computing paradigm by taking advantage of the solutions for both membership management and communication as a way to interconnect devices "on the edge" of the network. In fact, the distributed nature of devices on the edge matches with the principles behind P2P and overlay networks as they are "highly decentralized, robust and can be easily adapted to promote hierarchical topologies" [23] like the ones considered in fog [9] or mist computing [29] platforms. The usage of a P2P paradigm for edge computing does not remove the role of data centers for gathering, processing, and analyzing big quantities of data as they could effectively be present in the network as high capacity nodes that can be contacted by other peers as any other network participant. It is important to note however that, as it will be described later, these solutions are mainly focused on working in a specific scenario which might be a problem when considering the already discussed heterogeneity among edge devices and nodes in the network. This is what motivates the importance of developing generic

abstractions for allowing the use of different solutions, considering the ones already employed in P2P architectures, by different edge devices based on their characteristics or network conditions specially when taking into consideration that a high number of very different devices are expected to be participating in the network.

2.2 Use Cases

In order to develop generic solutions for providing membership management and communication abstractions, a number of use cases will be studied for both development and evaluation proposes. Each of the presented use cases is expected to rely on edge computing based on a P2P approach. The objective is that the generic solutions devised can deal with the heterogeneity introduced by the conditions expected in the networks for each use case, as well as being applicable to multiple use cases. The set of use cases considered, partially adapted from the TaRDIS European project [44], includes the following: Energy management network for Smart Grids, Satellites self-coordination in space, Industrial controllers for Smart Factories, and IoT devices for interacting with personal assistants.

Each one of the presented use cases will be described and studied more in-depth in the next section. For each, we provide a description of the objective, expected operational environment, and network participants behavior as well as the characteristics of the devices connected, such as processing power, storage, network capabilities, availability guarantees, and expected network conditions, e.g., number of nodes or latency. In each section works related to the use case being presented are also discussed as well as the main challenges that we expect to encounter.

2.2.1 Energy Management

One of the possible use cases for edge computing relying on a P2P paradigm is related with the management of electric energy delivery networks. Today the development of technologies for energy production at home, which take advantage of devices like photovoltaic panels, opens new possibilities for green energy sources especially in a world increasingly more dependent on electric energy, however, this translates itself into more challenges for managing and overseeing the energy grid. If until now the energy sources were located into specific production infrastructures, under direct control of an organization, now energy is also produced in lower production environments scattered between hundreds or thousands of locations like homes, office buildings, industrial facilities, among others.

The individual energy producers should be able to choose how they want to manage the energy they produce, for example, by choosing if it should be sold and injected into the grid, stored in batteries for later usage or used for running home appliances or charging electric vehicles. Producers can even choose to sell energy stored in their vehicles or batteries to make profit, for example, by selling energy at a time when the cost is higher

and buy it when the cost is lower or adapting their power usage to the evolving energy costs.

It is important to consider that not only on the production side the management of energy is a challenge nowadays but also in the consumption with the demand of electric energy increasing due, for example, to the adoption of electric mobility and electric home appliances. These challenges paved the way for the appearance of what are called *Smart Grids*.

Smart Grids can be described as applying information processing and communication capabilities to the power grid by leveraging on an information technologies infrastructure. This allows for devices withing the grid to gather information about the behaviors of suppliers and consumers that will then be sent through an information network and processed for making grid management decisions in an automated way, ensuring gains in reliability, efficiency and economy by lowering costs [4]. However, these grids need an information infrastructure capable of dealing with a high quantity of data that needs to be quickly processed and events, generated as a result of that data analysis, that should be swiftly propagated to the end devices. As seen before this is exactly the type of situation in which edge computing could have a positive impact.

The usage of the P2P paradigm leveraging overlay networks as the base for building a decentralized information infrastructure that supports Smart Grids is not new and is already addressed in multiple published works [19, 31, 4]. In these works the usage of P2P emerges as a solution for building a reliable, secure, and scalable infrastructure that can quickly exchange information and respond to events like a drop in energy production by solar panels due to a rapid weather change [19]. Nevertheless, one of the problems already described when dealing with P2P networks, and which is the main focus of this work, already exists here: the significant heterogeneity between devices participating in the network [31]. This issue, explained in more detail in the following paragraphs, highlights the need for generic membership management and communication primitives that can be applied to edge computing leveraging a decentralized P2P approach.

In order to address the heterogeneity in a P2P network for managing power grids two types of nodes will be considered: i) devices at home connected to the network which are responsible for managing the power usage and ii) electrical vehicles that consume energy. Both types of devices are expected to act as network nodes sending and receiving information between themselves and with the other type via the P2P network. The objective of this network is to allow clients to manage the power usage within their home, for example, by deciding if the electricity stored in a vehicle's battery should be sold at a given hour on which the energy has a higher cost, but the vehicle is generally unused, and then bought back at a time of the day when the energy cost is lower. Other scenarios can be devised, for example, the transfer of real time information from the home based devices to the vehicle in order to notify about the current status of electric production at home allowing the user to choose if the vehicle should be charged at home or in a public charging facility. Data could also be transferred from these devices, at the edge of the

network, to centralized servers for gathering and statistic treatment. In fact, in the example of the energy market this data transfer needs to happen in order to process payments, inform energy suppliers about the buy/sell operations executed as well as gather and aggregate statistical information.

When considering the characteristics of the nodes that take part in the P2P network on the previously presented situation, the heterogeneity becomes even more evident. Using the same aspects considered in [23] and already described in Section 2.1, i.e., processing capacity, availability guarantees, domain, storage capacity, and computation capabilities, it is possible to note that, besides having similar characteristics regarding domain (user domain) and computation as it is expected that both vehicles and home devices to allow energy management can have generic computation capabilities, the processing capacity, storage, and availability can differ significantly.

On one hand, for the home nodes, a medium processing capacity can be considered as computers with higher specs can be placed at home with higher availability guarantees due to the expected constant connection to an underlying network, such as the Internet, via technologies like optical-fiber or coaxial cables. On the other hand, it is expected that vehicles have less processing and storage capabilities due to size, cost and energy usage limitations and, maybe even more important, lower availability guarantees as the connection to the underlying network needs to be done through mobile infrastructure. Although considering technologies like 4G and 5G that allow for greater connection speeds and availability guarantees, the constant movement of vehicles means that dead zones are common, for example, in tunnels or in remote areas of the territory where latency may be high or even network connection could be lost completely which might lead, in conjunction with the expected high number of connected devices, to significant levels of *churn*, a term related with the number of nodes entering and leaving a P2P network at a given time [20].

2.2.2 Satellites Self-Coordination

The use of edge computing paradigms and P2P architectures as way to manage swarms of satellites in space is also one of the use cases considered in this work.

Satellites can be a good option in scenarios where other forms of communication are not possible or are severely damaged, like sensors placed in remote areas that need communication capabilities and mission critical operations providing, for example, communication capability in natural disasters where it must be quickly restored in order to facilitate the work of emergency services [37]. In this last case the reliability of the network is of the upmost importance as emergency operations may be dependent on it. In fact, Routray et al. [37] even consider the use of satellite networks in order to provide smart healthcare and telemedicine systems which, as explained before, could also leverage on the employment of edge computing due to privacy reasons which suggests an even profound connection between edge computing and satellite networks.

The main issues regarding the usage of satellite based networks are related with latency and reliability due to the fact that communications can be highly influenced by the movement of both satellites and earth, as well as weather conditions. One other question, which will be extensively presented in this dissertation is related with the expected heterogeneity in the network due to the different characteristics, protocols, and operating conditions between space and ground based devices [8, 12].

Two types of significantly distinct nodes will be considered in this example as taking part in the network: satellites and ground based stations. On one hand satellites should be able to communicate with ground based stations in a bidirectional way, for example, in the case of meteorological satellites sending weather data to be analyzed and receiving information about positioning correction if needed. On the other hand both satellites and ground stations are also expected to communicate with nodes of the same type as satellites could communicate with other satellites in order to avoid collisions and ground stations might need to communicate in order to transfer data between themselves.

It is important to note that, even when considering that a node needs to communicate with another one of the same type, it is possible that this communication transverses multiples types of nodes as in the case of a satellite (A) that needs to send information to another satellite (B). This could be done through ground stations with a station receiving the information and sending it to another station that then retransmits the data to B to start the avoidance maneuver. The situation where satellites communicate directly, i.e., with data not needing to pass through ground stations, should also not be discarded as if a network with many satellites around the globe is considered the expected proximity between them might enable for this capability. In fact, this could even allow for only some satellites in a swarm to be connected to ground nodes in order to communicate by receiving or passing information to other ones.

The usage of the edge paradigm to tackle this use case considering a decentralized network is in line with the expected distributed architecture of such network, as relying on central points of coordination not only could bring latency issues related with the time needed for information to be sent to a centralized infrastructure, processed and then the results or response action sent back to the appropriate peer as it could also bring reliability issues because relying on centralized architectures can lead to more possibilities of failure which can have catastrophic consequences, for example, if collision avoidance systems between satellites are dependent on it.

It is true that an adequate number of servers could act as a solution to this problem but the highly distributed environment that this use case poses, as satellites are expected to be all around the world and so ground stations to, could imply that a significant number of data centers would be needed to quickly receive and process that information, with the associated costs, or the latency will be higher due to the distance between these central points. However, even when considering edge computation, the presence of data centers as nodes within the network to process large quantities of data could be considered when working together with the other types of peers. Another question is related with the

scale of the network as, if one could expect that the number of peers would be low, it is also true that nowadays networks of numerous small satellites are already a reality with systems like Starlink [41], aimed to provide satellite based Internet access around the world, expected to have thousands of satellites in orbit [40].

Again considering the characteristics presented in [23], satellites can be expected to have low processing and storage capabilities due to energy efficiency requirements and size restrictions and are not expected to be able to perform heavier computations that the ones needed for data collection, handling communication and navigation equipments, and possibly some filtering of data. The availability guarantees can also be considered to be low with a high possibility of entering dead-zones due to the harsh conditions of the operating environment and communication mechanisms reliant on radio waves and not in technologies like optical fiber. In comparison, the latencies expected for *Medium Earth Orbit* (MEO) satellites are around 100ms, while for optical fiber are around 5ms [8].

On the other side, stations on the ground are expected to have high speed connectivity to an underlying network effectively ensuring high availability. Medium processing power and medium/high storage capability are also expected as the energy consumption and spacial limitations should not pose a problem as these infrastructures can even do general computation operations. The usage of techniques that allow partial replication of the application inside these structures can also be considered. Information received by stations can be quickly processed and, if needed, relayed for triggering actions with messages being then sent through the network again. If large amounts of data are received they can also undergo some form of treatment and then sent to more capable data centers for further processing.

When taking into account the characteristics of edge devices described before namely domain, processing and storage capability, network reliability, and computational capacities the heterogeneity present in this network becomes even more evident. In fact, the usage of different protocols for network management and communication when considering space-space, space-ground, and ground-ground interactions could even be considered as a way to deal with the inherent heterogeneity existent between these situations. Again this brings back the central question of this work, as generic abstractions are needed in order to allow different approaches to be put together, easily and efficiently, in terms of network management and communication protocols in the context of edge computing and P2P networks where significant differences between nodes are expected.

Although in the evaluation presented before only satellites and ground based stations with significant specifications were considered, the existence of lower specification nodes in the network should not be overlooked. These devices can either be directly connected to satellites, the case of IoT sensors in remote locations as an example, or connected through terrestrial network infrastructures to the ground stations responsible for handling the communication with the satellites [8]. In fact, Apple has announced that iPhones will start to have satellite network connectivity for emergency situations [13], one of the situations already described before as an expected usage, and Starlink offers a service, called

Swarm [43], exactly developed to provide smaller IoT devices with satellite based connectivity. In any case these nodes are effectively part of the network and their characteristics should also be considered as contributing even more for the expected heterogeneity.

2.2.3 Industrial Controllers

Another possible application for edge computing leveraging on a P2P paradigm, that will be considered in this section, is related with controllers for managing and monitoring industrial facilities.

One of the aspects currently in discussion is related with the usage of the IoT paradigm applied to industrial controllers as a way to help in the operation, management, and monitoring of industrial facilities. Sensors placed around the factories alongside with industrial controllers with information collection and network communication capabilities allow an interoperability between different industrial machines [35] which can lead, for example, to autonomous machine operation or enable systems that are able to predict when a machine is about to fail improving productivity and "ensuring a safe and efficient operation of the manufacturing process system" [32]. These advances lead to the appearance of industrial facilities known as *Smart Factories* [35, 25, 7].

The industrial sensors and controllers are expected to be connected by a fast network able to exchange information with high reliability and fault tolerance guarantees as they need to react quickly to events happening in the facility. These needs may not be fully satisfied when using a centralized client/server approach as all nodes need to contact a central coordination point which not only poses scalability problems, if a significant number of devices is considered, but also reduces fault tolerance by relying on a single point of failure. Due to these limitations a solution based on edge computing principles, leveraging a P2P architecture and overlay networks, can be considered where all nodes are able to contact each other without the need for a centralized server. In fact various authors [32, 35, 25] consider the usage of decentralized architectures for smart factories networks with Poonpakdee et al. [32] even proposing a solution to convert current centralized systems to decentralized architectures, taking advantage of P2P paradigm and overlay networks.

When evaluating the peers expected to be present in these types of networks, using the same criteria as before, and considering devices like industrial machines controllers, the issues related with nodes heterogeneity do not appear to pose a significant problem as controllers can be considered as fairly similar devices. Controllers operate on the user domain and are expected to have low processing and storage capacity as they should only be able to control machines operation and extract information, eventually with aggregation capabilities in order to be able of locally aggregate information before sending it throughout the network. It is important to note, however, that in this use case a low level of churn is expected, due to devices being constantly connected in a controlled environment. This is important, as the capability to transfer information between nodes

is expected to be much more important than the local processing and storage capabilities as controllers may need to coordinate and respond to events triggered by other nodes.

Although not requiring special care when challenges like heterogeneity and churn are taken into consideration, an important aspect that should be addressed in this use case is the need for reconfiguring the controllers in the network in order to allow the factory reconfiguration [35, 7]. The reconfiguration of nodes can be related, as an example, to the ability of deploying new versions of the code executing on a given controller with the objective of changing the way an industrial machine is operated or even changing the machine controlled by a given node. These reconfigurations are expected to be frequent with different nodes possibly having different behaviors at a given time and the same node possibly changing its behavior multiple times during operation.

This reconfiguration capability may have impact on the network due, for example, to changes in communication patterns. Consider three different nodes: A, B and C. Node A that before the reconfiguration regularly exchanged information with node B might now, after the reconfiguration, decrease significantly the amount of information exchanged with B and increase data exchanged with a node C. The network should be able to adapt to this type of situations in order to allow nodes to continuously send data and respond to events using a reliable, low latency, and efficient network.

Even though, as described before, this use case does not pose significant issues regarding the heterogeneity of network participants, the generic abstractions for both communication and membership management that will be developed during this dissertation should also take into consideration and be evaluated against the conditions expected in this network, particularly considering the questions related with reconfiguration, with the objective of providing an efficient and reliable operation under all scenarios.

2.2.4 IoT Devices

Finally, we consider a use case related with IoT devices placed at homes with the objective of working as personal assistants. In this use case a set of intelligent devices able to control home appliances like thermostats, lights, among others are considered. The objective is to allow these devices to be coupled together in the same network taking advantage of the coordination possibilities between them in order to provide helpful services by devising common usage patterns in some geographic location. As an example, one could expect that if sensors in the same geographical area are able to get and exchange information about the hours at which users turn on the thermostats, as well as the defined temperatures, this could lead to the automatic execution of this task without the need for human intervention.

A first - cloud based approach - to this problem would be to send all information retrieved by sensors in a house directly to data centers with high processing capabilities in order to gather the data, correlate it, and trigger the necessary actions, but, in addition to the expected high flow of information with which the data center would have to deal

with, this could lead to significant privacy issues [11] related, for example, with different privacy regulations between the house and data center location especially considering that sensible information, such as location and user behavior, needs to be sent.

The usage of edge computing can help in situations like these as it allows data to be processed and anonymized locally in a decentralized manner instead of being sent to data centers. In fact, different layers of devices could be devised in a fog computing style [9] allowing nodes near the data source to gather data and trigger actions by correlating the received information about their area of operation. Data can even be processed locally by having a home gateway with enough processing capabilities responsible for gathering information from sensors, process and anonymize it before sending it through the network using a mist computing [29] approach. These solutions are in line with the expected behavior of devices acting as home assistants as they are expected to be more interested in data from the closest geographical devices than the ones far away and, as so, the network architecture should consider this situation.

A step forward would be allowing these devices to communicate directly between themselves using a P2P strategy. If adequately anonymized this would allow a flow of information to exist between home gateways without requiring any type of central processing and devices could set off some action just by processing information obtained from other gateways in the same area. By relying on this approach it would be expected that all gateways in the same area came to the same conclusion because they would roughly receive the same data and thus present the same behavior. However, this approach does not mean that data cannot be sent to central processing infrastructures, both global or per area, that are part of the network and would only receive pre-processed and anonymized information.

Regarding the comparison between the kinds of peers expected to be present in the network described above three main types should be considered: IoT sensors, home gateways and data centers. By evaluating these nodes, based on the characteristics presented in [23], it is possible to conclude that data centers are expected to be nodes with high specifications in terms of processing, storage, and availability guarantees, working within the application domain and capable of generic computations. It is important to note that local processing nodes responsible for handling the information of a specific area, as described above in this section, also fall into this category although having lower computation power. Home gateways responsible for gathering, treating, and possible displaying of data originated on IoT sensors and other devices within a home operate on the user domain and are expected to have medium processing capabilities and availability guarantees, with a medium/low storage capability and should be able to at least aggregate data possibly even performing generic computations. The IoT sensors placed at homes also operate on the user domain and are considered lower capacity devices with low processing capabilities, small or even none storage capacity, and medium availability guarantees. These IoT devices are designed to only be able to obtain data and transmit it, although in some cases filtering capabilities could be considered for locally obtained data.

Both home gateways and IoT sensors are expected to be responsible for some churn in the network, as they do not have high availability guarantees.

The nodes' evaluation presented before allows to identify a significant heterogeneity among various types of nodes in the network, from small sensors with low processing capabilities at home, to large data centers with high computational power. Also, the network conditions may vary a lot when comparing the network conditions at home with the ones that connect data centers with high speed and reliability guarantees. Even when considering home locations it is expected that not all home networks have the same guarantees in terms of speed or reliability, as differences exist related with their location and Internet connection speed.

Finally, it is also important to consider that, in order to allow the home assistant devices to trigger actions based on learned patterns, not only mechanisms for information exchange are needed but also for storage and search of information which also represents a challenge in terms of privacy. A possible approach presented in [11] takes advantage of the blockchain technology as a way to store information on the end devices without requiring centralized storage but falls short on two aspects: i) the expected limitations in storage capacity when considering home gateways and ii) the scalability problems regarding blockchain technology also mentioned in the referenced work. An option for dealing with these privacy issues could be to store already anonymized information in the central processing locations. This way data stored, for example, in the processing nodes responsible for an area may not need to keep information regarding the exact origin location because the information received and stored in the node is from devices in the vicinity and, as so, can be correlated. On the other hand if central processing locations exist they only need to know about the area from which the information is originated rather than the exact location.

Although the solution presented before allows for tackling the privacy concerns it also poses questions related with the heterogeneity of communication mechanisms necessary in the network as different requirements arise for different types of nodes, i.e., on one hand home gateways need communication protocols based on geographical distance, for efficiently finding and communicating with other gateways and edge processing nodes in the area, on the other hand, both edge and central processing locations need to communicate between themselves independently of the location.

Along this section, by describing and studying the specific details, expected nodes, system behavior, and operating conditions of each use case, it has possible to further understand how each one of them can leverage on edge computing and a peer-to-peer approach, as well as the main strengths and challenges of relying on this approach with particular focus on the questions related with the heterogeneity of the network, a problem to which the results of this dissertation are expected to contribute positively. To end this study Table 2.1 presents a summary of the expected properties for each type of node present in the considered use cases, that were already depicted before based on the evaluation suggested in [23].

Use Case	Devices	Processing capability	Availability	Storage capability	Computation
Energy Management	Home Controllers	Medium	Medium	Medium	Generic
	Vehicles	Medium/ Low	Low	Low	Generic/ Aggregation
Satellites Management	Satellites	Low	Low	Low	None/ Filtering
	Ground stations	Medium	High	Medium/ High	Generic
Industrial Controllers	Controllers	Low	High	Low	Aggregation
IoT Devices	IoT Sensors	Low	Medium	None/ Very Low	None/ Filtering
	Home gateways	Medium	Medium	Low	Generic/ Aggregation
	Data centers	High	High	High	Generic

Table 2.1: Comparison between characteristics of considered nodes for each use case

2.3 Overlay Networks

In this section a more in depth study of overlay networks will be conducted regarding their main characteristics and properties in particular related with network view and structure. First an overview of overlay networks will be carried out regarding its basic concepts and main properties, then a comparison between global and partial views is provided and, after that, both structured and unstructured overlay networks will be presented as well as a comparison between the positive and negative aspects of each one. An overview of decentralized communication protocols will also be provided in this section. The different overlay networks solutions presented here will serve as a starting point for the study of the different approaches that should be considered during the development of this dissertation.

As described before in this document, an *Overlay Network* can be defined as a set of nodes connected between themselves on top of an underlying network (hence the name *overlay*) with each node representing a process and considering a set of other nodes as its neighbors. In fact, in many cases overlay networks rely on the Internet as the underlying network and the processes open TCP connections between themselves creating a link [27, 21, 24]. In an overlay network nodes can be connected to other ones independently of their physical location or the physical links that exist between them, i.e., in a network operating on top of the Internet, a node located in Portugal can have its neighbors located in Japan because, although a direct connection may not exist, links like TCP connections are logical and are created on top of the underlying network. Many P2P systems rely on overlays as membership management mechanism because this type of networks enables processes to be effectively interconnected with the other nodes on the network acting as peers in the

system [20]. In the literature, overlay networks are often described as graphs $G = (V, E)$ where the vertices of the graph correspond to the nodes and the edges correspond to the connections established between them [22, 21, 18, 6].

2.3.1 Global view vs Partial view

Before presenting a more in-depth study of overlay networks it is important to characterize peer-to-peer systems based on the view that each node has of the system membership, therefore, P2P architectures can be characterized based on the connections established by each node participating in the system with the other ones in *Global View* or *Partial View* architectures, with each one of these approaches having a profound impact not only on the mechanisms that need to be used for membership management and communication but also on the performance of the network depending on the operational scenario.

(a) Global View (b) Partial View

Figure 2.1: Comparison between global view and partial view on a P2P system network

Global View Systems relying on a global view are characterized by having each node know all other nodes in the network, therefore, having "access to the full membership information" [20], i.e., if we consider Π as the set of all network participants, the set of neighbors of a node A is $\Pi \setminus \{A\}$. A global network view allows for every node to directly communicate with any other node present in the network thus ensuring a very efficient network topology for both direct communication or broadcast of messages, a situation where a node just needs to send the message to all neighbors in order to accomplish broadcast. Simple membership management protocols can also be used because when a new peer (A) joins the network, for example by connecting to an already participating process (B), A can receive all the neighbors of B and establish connections with them relying on the underlying network, using TCP as an example [27], gaining a global view of the system and letting the other nodes know about the new one joining. On the other side, when a node leaves the network, the neighbors only need to detect the fault - or be informed by the leaving node if possible - and no longer consider it as neighbor. The main disadvantage of a global view is related with the high load placed on the network participants due to the number of neighbors that each node will need to manage, especially in situations where the network has more than a few peers and/or a significant churn is expected [20]. This problem leads to architectures relying on a global view only being

useful in situations where both the number of peers and the churn effect are expected to be low, like in the use case of industrial controllers considered in Section 2.2.3.

Partial View Systems relying on a partial view [20, 22] are characterized by having each node to know only a small portion of the peers in the network, meaning that a node A may not be able to communicate directly, i.e., using a direct link such as a TCP channel [27, 21, 24], with a node B also participating in the network and instead needs to rely on other participants that should relay the message - possibly multiple times - until it reaches the intended destination. When considering overlay networks, relying on partial views is the most common architecture as it solves the scalability and capability issues related with keeping track of the entire membership that arise when considering global views. On the other hand, this approach means that more sophisticated and complex membership management mechanisms are needed in order to define which (and how many) nodes should be chosen as neighbors, how changes in the overlay membership are handled when a new node needs to join or a participant becomes disconnected, and how nodes can communicate efficiently with others throughout the network. Many approaches for dealing with these aspects will be considered when studying the different overlay network solutions [21, 24, 30, 42, 6, 10].

Figure 2.1 shows the comparison between a global view and a partial view P2P systems, both with the same set of nodes $V = \{A, B, C, D\}$ but with a different set of edges. Note that the global view architecture is, in fact, represented as a complete graph.

2.3.2 Overlays Properties

Below we present some important properties of overlay networks, namely *connectivity* and *accuracy*, that have a direct impact on the correctness of operation, and *network diameter*, *clustering* and *node degree*, that should also be considered in order to improve performance.

Connectivity and Accuracy In order to enable the correct operation of an overlay network both *connectivity* and *accuracy* should be guaranteed [20, 21]. On one hand connectivity is related with the ability of any node to exchange information with any other, by leveraging on the overlay links established between nodes, which should guarantee that exists "at least one path from each node to all other nodes" [20]. If a node, or a set of nodes, loses connectivity it means that it is no longer possible to contact these peers using the network. On another hand, peers should also eventually drop existent connections to failed nodes in the network in order to guarantee higher accuracy, i.e., that a node does not have a significant number of neighbors that are no longer available. The accuracy value can be calculated, per node, as the division between the number of available neighbors and the total number and, at a network scale, as the average of accuracies. In fact a lower accuracy value can have impact on mechanisms like random walks, described in Section 2.4.1, causing messages to be sent more often to neighbors that are not available [20].

Network diameter, Clustering, and Node's Degree Some aspects should be considered when studying unstructured overlay networks with the objective of improving efficiency namely *network diameter*, *clustering*, and *node's degree* [20, 21]. The diameter is related with the lengths of the paths between two peers and networks should try to maintain a small diameter, which can be considered as the average of path lengths between peers, as this make it easier for messages to reach the entire network. Node's degree is related with the number of neighbors that a node have and the network should ensure that a uniform degree is maintained in order to not have a significant discrepancy between the number of each node's neighbors thus originating load imbalance. Clustering is related with the heterogeneity of neighbors between processes as overlay networks should strive to maintain a low clustering by maintaining significant differences in neighbor sets between peers, i.e., participants should not have all the same set of neighbors and heterogeneity should exist between them. A *clustering coefficient* can be defined, per node, as "the number of edges between that node's neighbors divided by the maximum possible number of edges across those neighbors" [21]. The average of clustering coefficients allows to evaluate the clustering level of an overlay network with higher values (between 0 and 1) meaning higher levels of message redundancy and high probability of isolation of network segments, therefore having a negative impact on fault-tolerance [21].

2.3.3 Overlays Topology

When studying the characteristics of overlay networks two main types of network topologies arise: structured and unstructured overlay networks. This characterization is related with how the peers participating are linked together between themselves which has significant impacts on which operations can leverage on the network and how can be implemented. Both of these categories will be described in the next section. It is important to note at this point that in this section only networks based on a partial view will be considered because when a global view approach is used the questions related with the topology of the network no longer matter as, in fact, the only possible topology is one with all nodes connected between themselves in a complete graph and always able to communicate over direct links. However, it is possible that nodes on an overlay network constructed based on a partial view approach can have, under certain scenarios, a global view of the network, for example, if the number of peers participating in the network is low (as in the special case of a single node).

2.3.3.1 Structured Overlay Networks

Structured Overlay Networks can be described by relying on peer management protocols that strive to maintain a specific network topology with the objective of making a certain operation, generally search within the network, more efficient. In structured overlays the set of links maintained between processes on the network are not random, and these links are created (or discarded) with the objective of maintaining a certain topology in

the structure of the network, for example, a ring, a tree, among others [27]. This specific topology should be chosen taking into consideration the expected operation of the overlay, network conditions and/or peer characteristics with the objective of ensuring a reliable and efficient operation.

One of the most common situations for the usage of structured overlay networks, in the context of P2P, is resource location by relying on *Distributed Hash Tables* (DHT's) [20, 27]. A DHT works, on a high level view, in the same way as a common hash table, i.e., data is organized using a key-value interface where the key is composed by a hash of some information correlated to the data (or the data itself) and the value represents the information to store, with the difference relying on how key-value pairs are actually stored in the nodes using a distributed approach. Generally the set of values S, that can be issued by the hash function h used, is distributed between all nodes present in the network becoming each node responsible for a subset of S (which could be chosen, for example, taking into consideration the identifiers of the nodes), then when data related with a key k needs to be found a node can perform a hash of the key $h(k)$ and contact the peer which is expected to be responsible for storing the information [27]. When a new node joins the network the subset of values that will become his responsibility should be computed and, possibly, information can be transferred from other nodes to it [42].

As discussed before, when relying on structured overlay networks for implementing DHT's the topology of the network should primarily ensure the efficiency of the search operation, i.e., that a node searching for a key could as efficiently as possible contact the node responsible for that key. Two well known P2P systems relying on structured overlays for providing a DHT based key-value pair storage are Chord [42] and Kademlia [30].

2.3.3.2 Unstructured Overlay Networks

Contrary to structured overlays, *Unstructured Overlay Networks* do not strive to impose a specific topology to the set of peers participating in the network [27, 20]. In this case the links maintained by nodes with his neighbors depend on many factors namely the moment when nodes joined the network, the peer to which a node connects when joining or the properties of nodes such as location, processing capacity, network connection, reliability, among other aspects.

Generally when considering unstructured overlays a new process joins the network membership by contacting an already participant one who is then responsible for enabling the joining node to obtain a set of neighbors, as an example, by forwarding the information about the join throughout the network in order to ask participants to establish connections with the new node [21, 24]. The membership management mechanisms in unstructured overlays are expected to be more simple and less computationally expensive when compared to the structured counterparts as a specific topology does not need to be enforced [20] and, as so, each node is responsible for choosing the set of neighbors based on previously defined heuristics and possibly also taking into consideration constraints

regarding the minimum and maximum number of neighbors. In fact, it is expected that when a node needs to add a new neighbor, due to not having a sufficient amount of neighbors or a connection failure, it should try to connect with other processes in the network, for example obtained through its neighbors, choose the best one according to the heuristics and attempt to establish a connection. In some cases nodes can even drop currently available neighbors in order to substitute them for newer ones [21, 6].

Generally we can consider all nodes participating in an overlay network as contributing equally, nevertheless, there are architectures where a few nodes contribute, in some form, more that others to the network operation.

Super-Peers A relevant architecture commonly used in unstructured overlays is related with the definition of special nodes called *Super-Peers*. This approach is leveraged in many overlay designs, like the ones proposed in Overnesia [24] as well as (newer versions of) Gnutella [3, 17], and can be defined by the existence of a specific type of peers in the network that have some set of characteristics that allow them to be "promoted" to this category based on aspects defined by the network implementation [20]. These characteristics are often related with processing capability, storage and/or reliability. The idea behind super-peers relies generally on biasing participants in order to prefer the connection with a certain set of super-peers to help build a more stable and efficient network. It is important to note that the usage of this mechanism does not impose a topology on the network, thus not forming a structured overlay, but instead the mechanisms for choosing a super-peer as neighbor are inserted into the heuristics used by processes to choose a neighbor over others [6]. In terms of search we can think of super-peers as neighbors expected to be able to store more information than others which explains the preference for more connections to this type of peers. Although relying on a super-peer based approach can help by improving efficiency, in fact, this strategy puts in question the uniformity of nodes degree described in Section 2.3.2 and can have negative consequences as a failure in a super-peer will produce a major effect on the network when compared with a common node [24]. The definition of the candidates to super-peer may also constitute a challenge as the characteristics may not be trivial to define [20].

2.3.4 Decentralized Communication Protocols

If a system relies on a centralized approach processes are only required to contact a central node, e.g., a server, in order to obtain the desired resource or execute a given operation. On the other hand, if a decentralized P2P system is in place, specially considering architectures leveraging on overlay networks, more complex communication protocols are required.

In the case of P2P systems relying on structured overlay networks the communication is often performed by contacting directly the node responsible for the identifier of a given resource [20], as described in Section 2.3.3.1. However, this does not mean that a (generally) small set of neighbors will not need to be contacted in order to reach the node

responsible for the resource, as a partial view of the system is maintained, hence requiring mechanisms, like the ones used in Kademlia or Chord, for a node to find the responsible, e.g., the node with the nearest identifier to the key [20, 42, 30].

The communication protocols applied to P2P systems leveraging on unstructured overlays are much different from the ones used in structured overlays, as the absence of a specific topology makes it harder for one peer to contact another one specifically responsible from providing a given resource. In many situations systems rely on *Gossip-based Dissemination Protocols* [20] where peers collaborate in exchanging messages throughout the network by sending it consecutively to n neighbors, which should then relay the message again if they have not received it yet. The parameter n in called *fanout* and in many systems is configurable with higher fanout values leading to faster message dissemination in exchange for more redundancy and load in the network [20].

2.3.5 Structured versus Unstructured Overlay Networks

As presented in Section 2.3.3.1, structured overlay networks have the main advantage of allowing an efficient discovery of a node or resource if the information about the full identifier is known. Chawathe et al. [6] present some challenges of using structured overlay networks starting with the effort that is needed for maintaining the expected topology and perform the management of the information stored, especially when considering large and/or dynamic networks with a significant churn effect due to a high number of nodes continuously joining and leaving the network. Another problem of the DHT approach is related with searching for a resource based on a partial key instead of the complete one (known as *exact-match queries*) as, for example, the hash of "NOVA School of Science" can - and will most likely - be completely different from "NOVA School of Science and Technology" and if the first expression is used in order to search for a resource whose key is, in fact, the second one the resource would not be found.

Unstructured overlay networks, on the other hand, are more suitable for situations where a high number of exact-match queries are not expected, and the searches are mainly done using partial query expressions because, due to the usage of gossip-based dissemination protocols, the search for a given resource can be made locally at each node when it receives the query and all matching results can be sent to the source node [6]. Another advantage is related with the membership management mechanisms that are expected to require less computational resources [20], due to not needing to impose a specific topology, thus being able to work more efficiently in highly dynamic networks with significant levels of churn and/or high number of peers.

The main disadvantages of an approach based on unstructured overlays are related with the high level of communication in the network that is expected due to the possibility of a significant number of messages being retransmitted throughout the network, and even though the usage of adequately configured mechanisms like time-to-live (TTL), discussed in detail in Section 2.4.1, and fanout parameters can mitigate this problem, if

incorrectly configured or implemented the network can become useless due to a high load. Relying on a super-peer mechanism, described in Section 2.3.3.2, can also contribute to the mitigation of this problem by reducing the traffic in the network [20]. Although more resilient to unstable networks conditions and churn effect than structured overlays, the mechanisms for neighbor selection should also be considered in order to not create an excess of dynamism in the network which can undermine its operation. One other problem arises when using unstructured overlay networks: it is not guaranteed that a resource is located even though it is present in the network as situations may occur when a combination of the defined parameters like TTL, network diameter, topology, and/or randomly chosen walks can lead a query returning no result, although a matching resource exists in the network. This problem has more importance if it is expected that a high level of *needles* [6], i.e., lower replicated resources, are expected to be present. In fact, the replication of data is not only important in overlays due to better efficiency in search but also in order to guarantee that information is not lost even if nodes become disconnected.

2.4 Services provided by Peer-To-Peer systems

Multiple services can be provided by relying on peer-to-peer systems taking advantage of overlay networks as a membership protocol. In this section some of the most common services that leverage on the usage of P2P architectures will be described such as *Resource Location*, *Broadcast*, *Publish-Subscribe*, *Distributed File Download*, and *Distributed Computation*.

2.4.1 Resource Location

This is the most commonly described service when considering peer-to-peer architectures and this could be explained by the fact that the first widely used P2P systems, like the already presented Napster and Gnutella networks, were designed exactly for this purpose as explained before in this document. Generally speaking *Resource Location* consists on using a network to find a given resource which can be, as an example, a file, a machine with specific resources or the node with a given identifier.

As stated in Section 2.3.3 a resource location service can be provided by P2P systems relying on both a structured overlay network, generally leveraging on consistent hashing to find a resource to which the full identifier is known, or an unstructured overlay network, by using mechanisms based on gossip-based dissemination, like *flooding* or *random walks* [27, 20], to allow finding a resource even without knowing its full identifier.

Flooding Flooding consists on sending a query throughout the entire network by forwarding it to all neighbors that should then forward the query again. This approach can put a high level of load into the network so, in order to deal with this situation, a mechanism called *Flooding with Limited Horizon* [20] can be put in place by using a time-to-live (TTL) value that, when reached, leads the query to be discarded. If the matching resource

is found the node that contains the resource can reply directly to the origin node or send the result back through the other peers [27]. It is important to note that even when a matching resource is found the queries may continue crossing the network giving the possibility of obtaining multiple results.

Random walks Random walks, on the other hand, consist on a node sending a query not to all neighbors but just to a small set (or only one) chosen at random. The nodes receiving the query then verify if the requested resource is present and, if successful, reply to the origin node as described before in the flooding method. Although originating much less stress on the network when compared to flooding, random walks also need to use a TTL like mechanism in order to stop queries from continue indefinitely and end up in a situation where too much effort is put on the network [27].

2.4.2 Broadcast

Broadcast consists, as the name suggests, in disseminating data throughout an entire network in order to reach every participating peer. Although questions related with the effort put on the network should be considered, broadcasting services can leverage on the usage of underlying P2P architectures. A simple implementation of broadcast can be done by relying on the flooding mechanisms already described in Section 2.4.1 [28]. If all messages that are flooded through the network contain a unique identifier, like a UUID, the participating nodes are able to verify if a message was already received by them and, if not, deliver it locally and retransmit throughout all neighbors [20].

When studying broadcast services provided on top of peer-to-peer systems the topology of the network can have a significant impact on the operation. As an example, a tree based topology would allow a reduction in the load put on the network as message redundancy is avoided due to the absence of cycles, however, a failure on a link could lead to a network partition [28]. An interesting solution for building a tree for broadcast and, at the same time, increase fault tolerance is relying on systems such as PlumTree [22] which are able to maintain a tree, even in presence of node failures, on top of a system relying on an unstructured overlay acting as peer sampling service, like HyParView described in Section 2.5.

In some situations services may also need to use *Multicast* in order to deliver information only to a subset of nodes. A multicast service can also be provided on top of overlay networks, with Steen and Tanenbaum [28] proposing a solution where nodes maintain multiple sets of neighbors, each one from a different multicast group. This way messages can be flooded only in the context of each group that acts as a separate overlay.

2.4.3 Publish-Subscribe

Publish-Subscribe services are nowadays widely used in a variety of situations with companies like Google [47] or Amazon [46] offering this type of service through their respective

cloud computing platforms. Publish-Subscribe allows for an asynchronous form of event-driven communication where information consumers subscribe to a set of topics related with the data they wish to receive and, on the other hand, producers send messages tagged with topics that should only be delivered to participants in the network that subscribed the topic [27, 14].

In some publish-subscribe systems it is even possible for a consumer to receive information sent while it was disconnected from the network when it becomes online again [14]. This can be important in cases where it is expected that a node does not always have a stable network connection, like when we consider vehicles on the energy management infrastructure described in Section 2.2.1, thus leading to possible reliability issues that can be tackled with this approach. In fact, some authors such as Kim et al. [19] consider the usage of publish-subscribe services working on top of P2P architectures for energy management networks.

A simple approach to publish-subscribe services based on P2P networks can rely on broadcast operations. If a message is broadcasted throughout the entire network with information about the topic, or the set of topics, related to it, each node can then decide if they should deliver the information received by comparing the message topics with a local subscriptions set. Although this approach could lead to a significant load on the network if a high number of messages was sent, which is expected when considering publish-subscribe services, and possibly generate a high quantity of unuseful traffic related with the retransmission of messages that would not be delivered by nodes because the topic was not subscribed, it has the advantage of allowing for the subscription and unsubscription operations to be done locally without the need for any exchange of information between network participants. Nevertheless, other approaches exist for implementing publish-subscribe services on top of P2P networks like Scribe, "a large-scale event notification infrastructure for topic-based publish-subscribe applications" [39] which relies on an overlay network called Pastry [38].

2.4.4 Distributed File Download

Peer-to-peer networks are also widely used to build distributed file download services like BitTorrent. The idea behind BitTorrent is to allow decentralized downloads of files by providing mechanisms to clients that enables them to transfer chunks from other network participants that are then merged together in order to obtain the complete file. [33, 27]

Services related with distributed file download provided over P2P architectures should also ensure that users are required to collaborate in the sharing of information and discourage a practice called *free riding* [27, 17] where a user takes advantage of the network to download files but does not contribute. This practice can create a significant imbalance on the system by creating a situation where, although having a high number of data requests, the number of nodes effectively sharing files is low. The free riding practice is a problem which was already studied in the context of the Gnutella network in [17].

BitTorrent protocol limits this problem by imposing a mechanism that requires users to effectively exchange chunks of information who own at approximately the same rate at which the data is being received [27].

2.4.5 Distributed Computation

The distribution of computation between machines with the objective of parallelizing data processing, in an approach commonly called *Grid Computing*, is one more example of a service where the usage of P2P architectures can be leveraged. In fact, the SETI@Home [1] infrastructure already described in Section 2.1 is an example of how nodes connected on a peer-to-peer network can contribute to the parallel processing of data. However, this service relied only partially on a decentralized architecture as a central server was needed for both data distribution and post-processed data retrieval.

Foster and Iamnitchi [15] present a comparison between the Grid Computing and P2P paradigms and, according to the authors, although the main objective of both approaches is related with "resource sharing within virtual communities" and both rely on the same approach, in some aspects differences exist between the two such as the ones related with the computational power of nodes, which is expected to be higher in the case of Grid Computing, or the mechanisms in place to deal with a high number of nodes, lower guarantees of availability or enforcing a fair usage of the system by clients which are more developed in the context of the P2P paradigm. However, specially considering, on one hand, the increase in the size of computer grids and, on the other hand, the increase in computational performance of day-to-day devices, both approaches seem to converge into each other and the usage of P2P based mechanisms to handle grids of machines executing distributed computations is something that should be considered. An example of this overlapping between both approaches is the work by Verbeke et al. [45] where a framework for enabling Grid Computing on top of a P2P architecture operating on a heterogeneous environment is presented.

2.5 Examples of Peer-To-Peer systems

In this section some examples of Peer-To-Peer systems, reliant on overlay networks as membership management protocol, will be described. It is expected that the approaches leveraged by these systems can serve as base for the work being developed in the context of this dissertation.

Kademlia Kademlia [30] is one of the most well known P2P systems reliant on structured overlay networks. The system relies on a network of peers, each one with an assigned identifier, that store neighbor nodes on buckets, called *k-buckets* and seen as leaves in a binary tree, based on a XOR distance metric between the binary representation of the identifiers. When a node wants to contact other node with a given identifier, it needs to

contact α peers (where α is a configurable parameter) from the nearest k-bucket, based on a XOR operation, and should receive, from the contacted nodes, the closest k known nodes on which the operation is repeated until no other node with the nearest identifier is received. This solution allows for peers in Kademlia to approach, step by step, a node with a given identifier or the closest one which could, for example, be responsible for storing a value related with that key. An interesting use scenario for Kademlia are situations where the interaction of a node with the closest ones, regarding the XOR metric applied to the identifiers, is more common as peers are expected to have more information about the network topology near its identifier due to the split of known peers in buckets.

Gnutella After the already described Napster approach, Gnutella was devised as a fully decentralized file sharing peer-to-peer protocol taking advantage of a flooding technique with limited scope [6] as a way to eliminate the need for a centralized search model. In the first versions of Gnutella a simple approach was used where each network node formed links, using TCP connections, to a set of other nodes and resource location was performed by flooding queries throughout the network and expecting that the nodes with matching resources reply directly to origin nodes with the requested data [36]. This approach showed scalability problems as the network usage increased, leading the network to became flooded with queries and, although a solution based on time-to-live (TTL) values was put in place this lead to queries failing to obtain the expected results as only a few nodes were visited [17, 6].

Gia Gia [6] is a file-sharing P2P system reliant on an unstructured overlay network developed with the objective of improving the mechanisms already proposed by Gnutella. Gia relies on biased random walks (an approach also described in [20]) to find resources, making nodes choose higher capacity peers when there's a need to forward a query, although also relying on a token based flow control mechanism where a node that handles more queries also earns the right, from its neighbors, to send a higher number of queries. In the approach followed by Gia, in addition to a TTL mechanism, a limitation on the max responses that a query should originate is also put in place due to the possible existence of multiple results. Another solutions are also used by this overlay in particular a topology adaptation mechanism that strives to put nodes with low capacity near higher capacity ones with the objective of increasing connectivity on high capacity nodes, effectively following a super-peer approach.

HyParView HyParView [21] is an unstructured overlay network devised with the objective of supporting a reliable broadcast of messages in a gossip-based approach. Unlike many overlay networks which rely on a single partial view composed by the known neighbors, the HyParView overlay relies on two distinct partial views one used to disseminate messages, called *active view*, and another large one, called *passive view*, used to substitute nodes of the first one when needed. The active view relies on a reactive strategy where

nodes are replaced when failures occur, however, the passive view relies on a cyclic strategy where a node periodically exchanges information about its known set of peers, from both views, with a member of the active view. The main advantages of HyParView are related with its high reliability even in the presence of a high level of node failures.

2.6 Summary

In this chapter the fundamental concepts, related work, and considered use cases were presented. The chapter started with an overview of edge computing and peer-to-peer architectures, followed by a presentation of four use cases that will be considered in the development of this work and which show the heterogeneity questions that we intend to tackle, next a study of overlay networks was provided by presenting the base aspects and work developed in this area that will be taken into consideration, finally, a description of a set of services that can leverage on the use of a P2P paradigm as well as a presentation of existent P2P systems, relying on overlay networks, was provided.

When looking back it is possible to understand that, although a significant quantity of work exist related with the subjects presented in this chapter, there is yet space for research and new solutions. One of the challenges is concerned with the heterogeneity of nodes, changes in operating conditions, and multiple intended usages that exist when considering matters like edge computing, peer-to-peer approaches, and decentralized environments. In fact, we believe that this heterogeneity is what explains, at least in part, the significant amount of different solutions, lines of work, and research existent in this area, each one focusing on a specific aspect, use case, or situation.

The requirements of the described use cases, as well as the multiple available solutions in the context of edge computing and peer-to-peer systems, both for communication and membership management, show that an integrating approach based on generic abstractions is needed in order to allow these solutions to be easily adapted and used under different conditions, either by configuration or in reaction to changes in the network conditions. Another line of work is related with the interoperability between different solutions, on the same network, if major differences arise between segments.

3

Future Work

In this chapter the challenge to which this work aims to provide response will be revisited, followed by a description of both the proposed solution and the evaluation methods. The chapter finalizes with the definition of the expected work schedule.

3.1 Proposed Solution

As described before in this document, we aim at define a unified solution that empowers developers to build a decentralized system, leveraging on a peer-to-peer based approach, to enable edge computing in a way that allows them to keep the benefits related with scalability, reliability, and even the possibility of addressing privacy concerns [20, 26, 10]. In fact, the expected topology of the network defined by edge devices matches the one considered in many P2P systems that implement some form of membership management and communication primitives on which the edge computing approaches can be built on top of. On the other hand, although presenting many advantages, relying on a P2P approach for edge computing raises questions related with the membership management mechanisms and communication primitives devised by those P2P systems as they are, in many cases, only suitable to be used in specific scenarios and unable to adapt to dynamic conditions, high heterogeneity, or when taking part on a different system [23, 29].

In more detail, the work being developed addresses the necessity of devising a generic approach to enable the usage of different membership management solutions and communication primitives under the same generic *Application Programming Interface* (API) which should provide access to all necessary operations, such as, join and leave operations, message broadcast, among others. This could allow for a simple development of edge computing systems that can rely on a common API and only need to change the mechanisms that implement it, for example, by configuration or even dynamically in reaction to changes in the network operating environment, relying on different approaches, interoperable between them, to materialize these abstractions. Leveraging a common interface would also allow for the deployment of different strategies in a per-node manner in order to tackle situations where a significant heterogeneity is expected between different

network nodes as described in Chapter 2.

In order to reach the expected results two main lines of research will be covered considering both the communication and membership management strategies.

3.1.1 Generic Communication Abstractions

When considering the communication strategies different solutions are being put in place by the already existent P2P systems, from using mechanisms like reliable broadcast to publish-subscribe based approaches. In this work it is expected that a generic approach to communication abstractions will be devised with the objective of allowing multiple communication primitives to be used by implementing a common generic API.

It is also expected that in a later development stage the proposed solution provides support for allowing different segments of the network to rely on different communication primitives, more adequate to the segment conditions and expected node behavior, maintaining interoperability between components and the correct system operation. Considering a simple example, one could expect that in the context of message broadcast a segment of a network relies on broadcast on top of a tree topology, as an example taking advantage of the PlumTree [22] solution described in Section 2.4.2, while another segment (possibly where a high level of node failures is expected) relies on a completely unstructured approach using the HyParView [21] system presented in Section 2.5. In fact, the example described before allows for an interoperability, through the same API, of distinct segments of a network reliant on different communication approaches.

3.1.2 Generic Membership Management Strategies

Regarding the membership management strategies it is expected that by studying various already proposed solutions, in particular considering the different approaches reliant on overlay networks followed by multiple P2P systems, a generic solution will be devised to support a (possibly extensive) set of membership management protocols under the same API that allows to choose which solution will be used underneath on a particular system or under specific circumstances. As described before, it is also expected that the devised solution supports the individual configuration by each node of the mechanisms in use, without impacting the correct system operation, with the objective of dealing with nodes heterogeneity and improving performance.

In a more advanced stage of the work, it is also expected to allow multiple membership management strategies to be used by different segments of the same network in order to take advantage of more adequate approaches to each segment operational conditions and/or nodes characteristics, therefore improving the system performance under high heterogeneity conditions, without compromising the interoperability between participants and the correctness of operation of the system as a whole. This is one of the challenges that we will encounter in the development of this dissertation as multiple approaches, based on the ones described before in Section 2.5, will be expected to work on the same

network, thus, requiring a solution on how a node will deal with multiple membership management strategies operating together.

3.2 Evaluation

The evaluation of the proposed solution is expected to be performed by emulating the network conditions for each use case presented in Section 2.2, as this approach allows running real code in an environment with the expected conditions [16]. The solution correction as well as values of metrics, like latency and reliability, will be studied and compared with other already proposed solutions for peer-to-peer systems, such as Gia [6], Freenet [10], HyParView [21] or Kademlia [30], on the conditions defined by the use cases with the objective of evaluating the work developed.

In the context of this work, considering latency and reliability, it is important to note that these metrics will be considered based on the following approach: *Latency* will be defined as maximum time for a message to be delivered, i.e., the time elapsed between the moment when information was sent by a node in the system and the moment when it was delivered by the target node, possibly by the last one in situations where multiple target nodes exist [22]. *Reliability* will be considered as the percentage of successful operations performed for a given definition of success, i.e., if we consider that a message was published on a topic in a publish-subscribe system, reliability can be defined as the percentage of nodes that received the message when considering all peers who subscribed the topic.

Regarding the development of generic membership management solutions, in particular when considering overlay networks, metrics like connectivity, accuracy, network diameter, clustering, and node degree [20], already described in Section 2.3.2, should also be taken into consideration when evaluating the proposed solution, as there is a known relationship between the graph properties and the impact on the performance of processes operating on top of these overlays.

3.3 Work Schedule

In this section an overview of the different work phases will be presented, as well as a brief explanation of what is expected to be developed in each one, and the predicted schedule of each phase. Figure 3.1 presents the expected work schedule for the development of the dissertation described in this document.

Preliminary Solution Over the period scheduled for the development of the preliminary solution it is expected that the design and implementation of the generic abstractions for both the membership management and communication mechanisms will be performed as described in Section 3.1. In particular in this phase it is expected that the generic API interface is defined as well as implemented with some membership management and

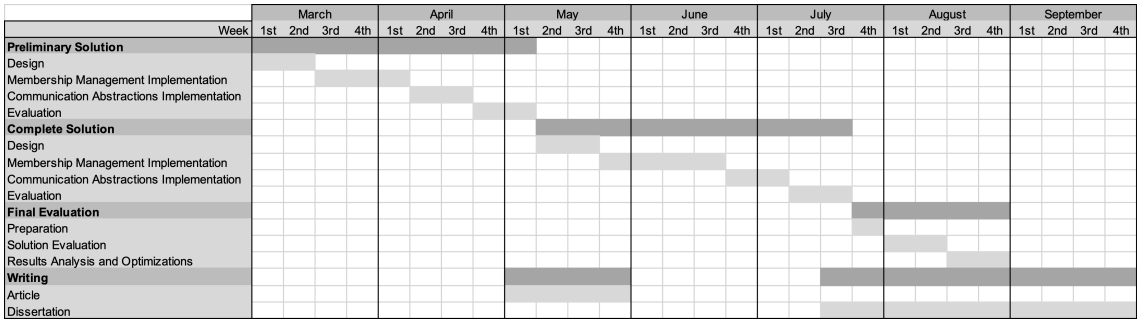

Figure 3.1: Gantt Chart representing the expected work schedule

communication approaches that are able to materialize it. This solution should also be subject to a preliminary evaluation both in terms of correctness and performance.

Complete Solution After the development and validation of a preliminary solution for providing generic membership management and communication abstractions it is expected that this solution will be further evolved and extended, as well as undergo optimizations based on the preliminary evaluation results. During this period it is also expected that the mechanisms for interoperability between different segments of the network relying on distinct mechanisms, available through the common API, will be designed and implemented as described in Section 3.1. As in the preliminary solution, this complete solution should also undergo validation and a first evaluation, before the final evaluation described below, in order to assess both the correct operation and solution performance.

Final Evaluation This phase is related with the complete proposed solution evaluation and consists on a preparation of the evaluation environment, in particular taking into account the conditions expected in each one of the use cases described in Section 2.2, the execution of the evaluation as described in Section 3.2, and the analysis of results. Possibly during this period some optimizations can also be implemented based on the obtained results.

Writing The writing section represents the work related with the development of both an article to be submitted to a conference and the final dissertation document.

Bibliography

[1] D. Anderson et al. "SETI@home: An Experiment in Public-Resource Computing". In: *Commun. ACM* 45 (2002-11), pp. 56–61. DOI: 10.1145/581571.581573 (cit. on pp. 7, 29).

[2] T. Arnold et al. "(How Much) Does a Private WAN Improve Cloud Performance?" In: *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*. 2020, pp. 79–88. DOI: 10.1109/INFOCOM41043.2020.9155428 (cit. on p. 2).

[3] B. Beverly Yang and H. Garcia-Molina. "Designing a super-peer network". In: *Proceedings 19th International Conference on Data Engineering (Cat. No.03CH37405)*. 2003, pp. 49–60. DOI: 10.1109/ICDE.2003.1260781 (cit. on p. 24).

[4] J. Cao and M. Yang. "Energy Internet – Towards Smart Grid 2.0". In: *2013 Fourth International Conference on Networking and Distributed Computing*. 2013, pp. 105–110. DOI: 10.1109/ICNDC.2013.10 (cit. on p. 11).

[5] B. Carlsson and R. Gustavsson. "The Rise and Fall of Napster - An Evolutionary Approach". In: *Active Media Technology*. Ed. by J. Liu et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 347–354. ISBN: 978-3-540-45336-9 (cit. on p. 9).

[6] Y. Chawathe et al. "Making Gnutella-like P2P Systems Scalable". In: *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. SIGCOMM '03. Karlsruhe, Germany: Association for Computing Machinery, 2003, pp. 407–418. ISBN: 1581137354. DOI: 10.1145/863955.864000. URL: https://doi.org/10.1145/863955.864000 (cit. on pp. 9, 20, 21, 24–26, 30, 34).

[7] B. Chen et al. "Smart Factory of Industry 4.0: Key Technologies, Application Case, and Challenges". In: *IEEE Access* 6 (2018), pp. 6505–6519. DOI: 10.1109/ACCESS.2017.2783682 (cit. on pp. 15, 16).

[8] W.-C. Chien et al. "Heterogeneous Space and Terrestrial Integrated Networks for IoT: Architecture and Challenges". In: *IEEE Network* 33.1 (2019), pp. 15–21. DOI: 10.1109/MNET.2018.1800182 (cit. on pp. 13, 14).

[9] Cisco. *Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are*. Tech. rep. 2015 (cit. on pp. 2, 6, 7, 9, 17).

[10] I. Clarke et al. "Freenet: A Distributed Anonymous Information Storage and Retrieval System". In: *Lecture Notes in Computer Science* 2009 (2001-03). DOI: 10.1007/3-540-44702-4_4 (cit. on pp. 1, 3, 8, 21, 32, 34).

[11] M. Conoscenti, A. Vetrò, and J. C. De Martin. "Peer to Peer for Privacy and Decentralization in the Internet of Things". In: *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*. 2017, pp. 288–290. DOI: 10.1109/ICSE-C.2017.60 (cit. on pp. 17, 18).

[12] G. Cui et al. "Latency and Energy Optimization for MEC Enhanced SAT-IoT Networks". In: *IEEE Access* 8 (2020), pp. 55915–55926. DOI: 10.1109/ACCESS.2020.2982356 (cit. on p. 13).

[13] *Emergency SOS via satellite available today on iPhone 14 lineup*. 2022-12. URL: https://www.apple.com/newsroom/2022/11/emergency-sos-via-satellite-available-today-on-iphone-14-lineup/ (cit. on p. 14).

[14] P. T. Eugster et al. "The Many Faces of Publish/Subscribe". In: *ACM Comput. Surv.* 35.2 (2003-06), pp. 114–131. ISSN: 0360-0300. DOI: 10.1145/857076.857078. URL: https://doi.org/10.1145/857076.857078 (cit. on p. 28).

[15] I. Foster and A. Iamnitchi. "On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing". In: *Peer-to-Peer Systems II*. Ed. by M. F. Kaashoek and I. Stoica. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 118–128. ISBN: 978-3-540-45172-3 (cit. on p. 29).

[16] P. Gouveia et al. "Kollaps: Decentralized and Dynamic Topology Emulation". In: *Proceedings of the Fifteenth European Conference on Computer Systems*. EuroSys '20. Heraklion, Greece: Association for Computing Machinery, 2020. ISBN: 9781450368827. DOI: 10.1145/3342195.3387540. URL: https://doi.org/10.1145/3342195.3387540 (cit. on p. 34).

[17] D. Hughes, G. Coulson, and J. Walkerdine. "Free riding on Gnutella revisited: the bell tolls?" In: *IEEE Distributed Systems Online* 6.6 (2005). DOI: 10.1109/MDSO.2005.31 (cit. on pp. 24, 28, 30).

[18] M. Jelasity, A. Montresor, and O. Babaoglu. "T-Man: Gossip-based fast overlay topology construction". In: *Computer Networks* 53.13 (2009). Gossiping in Distributed Systems, pp. 2321–2339. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2009.03.013. URL: https://www.sciencedirect.com/science/article/pii/S1389128609001224 (cit. on p. 20).

[19] Y.-J. Kim et al. "A secure decentralized data-centric information infrastructure for smart grid". In: *IEEE Communications Magazine* 48.11 (2010), pp. 58–65. DOI: 10.1109/MCOM.2010.5621968 (cit. on pp. 11, 28).

[20] J. Leitão. "Topology Management for Unstructured Overlay Networks". PhD thesis. Technical University of Lisbon, 2012-09 (cit. on pp. 1–3, 9, 12, 20–27, 30, 32, 34).

[21] J. Leitão, J. Pereira, and L. Rodrigues. "HyParView: a membership protocol for reliable gossip-based broadcast". In: *Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. Edinburgh, UK, 2007-06, pp. 419–429 (cit. on pp. 19–24, 30, 33, 34).

[22] J. Leitão et al. "On Adding Structure to Unstructured Overlay Networks". In: *Handbook of Peer-to-Peer Networking*. Ed. by X. Shen et al. Boston, MA: Springer US, 2010, pp. 327–365. ISBN: 978-0-387-09751-0. DOI: 10.1007/978-0-387-09751-0_13. URL: https://doi.org/10.1007/978-0-387-09751-0_13 (cit. on pp. 20, 21, 27, 33, 34).

[23] J. Leitão et al. *Towards Enabling Novel Edge-Enabled Applications*. 2018. DOI: 10.48550/ARXIV.1805.06989. URL: https://arxiv.org/abs/1805.06989 (cit. on pp. 2, 3, 6–9, 12, 14, 17, 18, 32).

[24] J. C. A. Leitão and L. E. T. Rodrigues. "Overnesia: A Resilient Overlay Network for Virtual Super-Peers". In: *2014 IEEE 33rd International Symposium on Reliable Distributed Systems*. 2014, pp. 281–290. DOI: 10.1109/SRDS.2014.40 (cit. on pp. 3, 19, 21, 23, 24).

[25] D. Lucke, C. Constantinescu, and E. Westkämper. "Smart Factory - A Step towards the Next Generation of Manufacturing". In: 2008-01, pp. 115–118. ISBN: 978-1-84800-266-1. DOI: 10.1007/978-1-84800-267-8_23 (cit. on p. 15).

[26] A. S. T. Maarten van Steen. *Distributed Systems*. 4th ed. distributed-systems.net, 2023. Chap. 1 (cit. on pp. 1, 32).

[27] A. S. T. Maarten van Steen. *Distributed Systems*. 4th ed. distributed-systems.net, 2023. Chap. 2 (cit. on pp. 2, 3, 6–8, 19–21, 23, 26–29).

[28] A. S. T. Maarten van Steen. *Distributed Systems*. 4th ed. distributed-systems.net, 2023. Chap. 4 (cit. on p. 27).

[29] R. Mahesa. *How cloud, fog, and mist computing can work together*. 2018-03. URL: https://developer.ibm.com/articles/how-cloud-fog-and-mist-computing-can-work-together/ (cit. on pp. 7, 9, 17, 32).

[30] 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 pp. 3, 21, 23, 25, 29, 34).

[31] E. Patti et al. "Distributed Software Infrastructure for General Purpose Services in Smart Grid". In: *IEEE Transactions on Smart Grid* 7.2 (2016), pp. 1156–1163. DOI: 10.1109/TSG.2014.2375197 (cit. on p. 11).

[32] P. Poonpakdee, J. Koiwanit, and C. Yuangyai. "Decentralized Network Building Change in Large Manufacturing Companies towards Industry 4.0". In: *Procedia Computer Science* 110 (2017). 14th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2017) / 12th International Conference on Future Networks and Communications (FNC 2017) / Affiliated Workshops, pp. 46–53. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2017.06.113. URL: https://www.sciencedirect.com/science/article/pii/S1877050917312929 (cit. on p. 15).

[33] J. Pouwelse et al. "The Bittorrent P2P File-Sharing System: Measurements and Analysis". In: vol. 3640. 2005-02, pp. 205–216. ISBN: 978-3-540-29068-1. DOI: 10.1007/11558989_19 (cit. on pp. 2, 8, 28).

[34] G. Press. *The State Of Data, December 2020*. 2020. URL: https://www.forbes.com/sites/gilpress/2021/12/27/the-state-of-data-december-2020/?sh=6d2fbc333462 (cit. on pp. 2, 6).

[35] J. Qin, Y. Liu, and R. Grosvenor. "A Categorical Framework of Manufacturing for Industry 4.0 and Beyond". In: *Procedia CIRP* 52 (2016). The Sixth International Conference on Changeable, Agile, Reconfigurable and Virtual Production (CARV2016), pp. 173–178. ISSN: 2212-8271. DOI: https://doi.org/10.1016/j.procir.2016.08.005. URL: https://www.sciencedirect.com/science/article/pii/S221282711630854X (cit. on pp. 15, 16).

[36] M. Ripeanu. "Peer-to-peer architecture case study: Gnutella network". In: *Proceedings First International Conference on Peer-to-Peer Computing*. 2001, pp. 99–100. DOI: 10.1109/P2P.2001.990433 (cit. on p. 30).

[37] S. K. Routray et al. "Satellite Based IoT for Mission Critical Applications". In: *2019 International Conference on Data Science and Communication (IconDSC)*. 2019, pp. 1–6. DOI: 10.1109/IconDSC.2019.8817030 (cit. on p. 12).

[38] 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. 28).

[39] A. Rowstron et al. "Scribe: The Design of a Large-Scale Event Notification Infrastructure". In: *Networked Group Communication*. Ed. by J. Crowcroft and M. Hofmann. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 30–43. ISBN: 978-3-540-45546-2 (cit. on p. 28).

[40] M. Sheetz. "FCC authorizes spacex to begin deploying up to 7,500 next-generation starlink satellites". In: *CNBC* (2022-12). URL: https://www.cnbc.com/2022/12/01/fcc-authorizes-spacex-gen2-starlink-up-to-7500-satellites.html (cit. on p. 14).

[41] *Starlink*. URL: https://www.starlink.com/ (cit. on p. 14).

[42] 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 pp. 3, 21, 23, 25).

[43] *Swarm - Low cost, global satellite connectivity for IoT*. URL: https://swarm.space/ (cit. on p. 15).

[44] *TaRDIS: Trustworthy and Resilient Decentralised Intelligence for Edge Systems*. URL: https://cordis.europa.eu/project/id/101093006 (visited on 2023-02-05) (cit. on pp. 5, 10).

[45] J. Verbeke et al. "Framework for Peer-to-Peer Distributed Computing in a Heterogeneous, Decentralized Environment". In: *Grid Computing — GRID 2002*. Ed. by M. Parashar. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 1–12. ISBN: 978-3-540-36133-6 (cit. on p. 29).

[46] *What is Pub/Sub Messaging?* URL: https://aws.amazon.com/pt/pub-sub-messaging/ (cit. on p. 27).

[47] *What is pub/sub?* URL: https://cloud.google.com/pubsub/docs/overview (cit. on p. 27).