Universitatea POLITEHNICA din București

Facultatea de Automatică și Calculatoare,
Departamentul de Calculatoare

TEZĂ DE DOCTORAT

Structuri auto-adaptive de încredere pentru rețele Peer-to-Peer

Conducător Științific:
Prof. Dr. Ing. Valentin CRISTEA

Autor:
Ing. Bogdan-Costel MOCANU

București, 2019

University POLITEHNICA of Bucharest

Faculty of Automatic Control and Computers,
Computer Science and Engineering Department

PHD THESIS

Trusted Auto-Adaptive Peer-to-Peer Overlays Networks

Scientific Adviser:
Prof. Dr. Ing. Valentin CRISTEA

Author:
Ing. Bogdan-Costel MOCANU

Bucharest, 2019

First and foremost, I would like to thank Professor Valentin Cristea and Professor Florin Pop, for all the valuable guidance he has offered during these three years and beyond. They have been extremely supportive in the outcome of this thesis, as they have helped me with advice, ideas, and encouragements. I have enjoyed working with them, and hope to be able to continue to do so in the future.

Secondly, I would also like to thank Professor Ciprian Dobre, whom I have worked and exchanged valuable ideas with at the faculty and as well as Professor Mariana Mocanu whom I have worked with.

Furthermore, I have had a successful and enjoyable collaboration with Aniello Castiglione from the Department of Computer Science, University of Salerno, and want to extend my thanks to him as well.

Finally, I would like to express all my love and gratitude to my family, who supported me from the start in this endeavor, especially to my wife Alec and my son Matei who was born during these years.

Abstract

The total Internet traffic in 2018 was around 1.2 ZettaBytes with a cost of 0.82 GWh. In the same year more than 1451 billions of videos on Youtube were viewed and more than 16 billions of pictures were uploaded on Instagram, while the total number of Facebook active users was more than 2.2 billion. These numbers show the fact that global Internet traffic is expanding at a high rate. Therefore, the global Internet traffic in 2018 is approximately 20000GB/second in comparison with 0.001GB/second in the early 90's. Therese significant numbers set the plot for an interconnected environment that shares media content, sensors data or computing data in an ubiquitous manner. Internet of Things and Smart*, as well as ubiquitous computing concepts, have become very well spared in the last years especially with the growth of mobile data bandwidth from 3G to 4G networks and the highly advances in the development of the 5G networks. This evolution has led to a new type of ubiquitous connectivity through devices in an unconventional manner. The deprecated, client-server approach is not anymore feasible in this new ecosystem, therefore the Peer-to-Peer concept is more suitable.

The shift of paradigm from server-based computation to user-centric paradigm is mandatory because increasing the centralized cloud data centers causes a significant decrease in the Quality of Service and Quality of Experience of the network systems. Therefore, the main objectives of this thesis are the research and implementation of auto-adaptive Peer-to-Peer overlay systems that can be used for large scale heterogeneous networks in the Internet of Things, Smart-* or ubiquitous computing ecosystems. The heterogeneity and the lack of reliability of the interconnected devices in large scale systems represent one of the most challenging research topics in the Peer-to-Peer systems. Therefore, the main problem addressed in this thesis is the robustness auto-adaptive Peer-to-Peer overlays.

The dimension of Peer-to-Peer systems is about thousands or million peers depending on the purpose of the system. Therefore, this decentralized approach has many applicabilities. An important assumption is the communication cost, which is essential in local area networks. The data types managed in Peer-to-Peer systems usually include multimedia content, sensor data, or computational queries.

Cyber-Physical Infrastructures represent engineering systems with high impact in the decision-making process of complex real-life systems such as surveillance or management of dynamic processes systems. The most important requirements of such systems are reliability, availability, fault tolerance, trust, load balancing, and auto-adaptability.

The main contributions of this thesis include the research, design, implementation and evaluation of an auto-adaptive Peer-to-Peer overlay that scales over heterogeneous and unsafe data sources in a fault-tolerant way and the usage of the proposed overlay is real life scenarios included in several projects developed in Politehnica University of Bucharest.

Rezumat

În 2018 traficul total de Internet a fost aproximat la 1,2 ZettaBytes, cu un cost energetic de 0,82 GWh. În același an, au fost vizionate aproximativ 1451 de miliarde de videoclipuri pe Youtube și au fost încărcate peste 16 miliarde de imagini pe Instagram, în timp ce numărul total de utilizatori activi pe Facebook a fost mai mare de 2,2 miliarde. Aceste cifre arată faptul că traficul global de Internet se află într-o continuă expansiune. Prin urmare, viteza globală a tot ceea ce înseamnă traficul pe Internet în 2018 a fost de aproximativ 20000 GB/secundă, comparativ cu 0,001 GB/secundă la începutul anilor '90. Aceste valori semnificative, stabilesc premisele pentru un mediu interconectat care partajează într-o manieră omniprezentă conținutul media, datele provenite de la senzori sau datele computaționale. Internetul obiectelor, Smart-*, precum și conceptele omniprezente de calcul au devenit subiecte dificile de cercetare în ultimii ani, mai ales datorită creșterii lărgimii de bandă pentru rețelele mobile de la rețelele 3G la rețelele 4G și progreselor foarte importante în dezvoltarea rețelelor 5G. Această evoluție a dus la un nou tip de conectivitate omniprezentă a dispozitivelor într-o manieră neconvențională. Abordarea învechită, client-server, nu mai este de actualitate în acest nou ecosistem, prin urmare conceptul Peer-to-Peer este mai potrivit.

Schimbarea opticii de la modul de calcul bazat pe servere, la o paradigmă bazată pe utilizator, este absolut necesară, deoarece creșterea dimensiunii centrelor de calcul de tip Cloud determină scăderea calității serviciilor și calitatea experienței sistemelor de calcul bazate pe rețele de calculatoare. Astfel, tematica principală a acestei teze de doctorat este structurile Peer-to-Peer care pot fi utilizate pentru rețele eterogene de mari dimensiuni în ecosisteme precum Internetul obiectelor, Smart-* sau sisteme de calcul omniprezente. Eterogenitatea și lipsa de fiabilitate a dispozitivelor interconectate în sistemele de mari dimensiuni reprezintă una dintre cele mai provocatoare teme de cercetare din domeniul sistemelor Peer-to-Peer. Prin urmare, principalul obiectiv de cercetare abordat în această teză este structurile Peer-to-Peer auto-adaptive sigure.

Dimensiunea sistemelor Peer-to-Peer este de ordinul miilor sau milioanelor de noduri, în funcție de scopul sistemului. Astfel, această abordare descentralizată are o aplicabilitate foarte variată. Costul comunicării reprezintă o ipoteză importantă în cadrul rețelelor de calculatoare și datorită acestui aspect, datele

gestionate în sistemele Peer-to-Peer includ de obicei conținutul media, date provenite de la senzori sau seturi de date utilizate pentru procesare.

Infrastructurile fizice cibernetice reprezintă sisteme inginerești cu impact ridicat în procesul de luare a deciziilor în cadrul sistemelor reale complexe cum ar fi sistemele de supraveghere sau sistemele de monitorizare și coordonare a unor procese dinamice majore. Cele mai importante cerințe a unor astfel de sisteme includ încrederea, disponibilitatea, toleranța la defecte, balansarea încărcării și auto-adaptabilitatea.

Contribuțiile principale ale acestei teze includ cercetarea, proiectarea, implementarea și testarea unei structuri Peer-to-Peer auto-adaptive scalabilă pentru surse de date eterogene și nesigure cu toleranță la defect, și utilizarea acestor structuri în scenarii reale incluse în multiple proiecte de cercetare dezvoltate în cadrul Universitatea Politehnica din București.

Table of Contents

Index of Figures

Chapter 1

Context and Goals

The highly increased volume of multimedia content from unreliable sources shared through Internet determines growth in the development of multimedia applications for handling such data from gathering to manipulation and data analysis. The efficient management of this kind of data must rely on uniform overlay networks. One suitable network overlay for managing a high volume of data from unreliable sources is the Peer-to-Peer overlay presented in many valuable research papers like [55], [102] or [101], because peers join or leave the system any time. For example, the project SIPTVMON [137] presents a secure application – layer multi-cast overlay network for IPTV. The proposed solution for IPTV (Internet Protocol Television) data assurance is based on the SIP (Session Initiation Protocol) protocol and uses cryptographic algorithms such as elliptic curves Diffie Hellman for key exchange and AES (Advanced Encryption Standard) encryption for data confidentiality. Another project that focuses on security assurance of distributed architecture overlays is SecP2PSIP [71]. Even though, the Peer-to-Peer concept is considered insecure, it might be possible for researchers to develop new security protocols for such network overlay. One challenging research topic in this field is the creation of an automated secure bootstrapping for P2P nodes. Taking into consideration the high increase in the number of intelligent devices interconnected such sensors and mobile devices, highly debated topics in the scientific community are trust and security assurance for such type of large-scale networks. Concerning different applications constraints, these networks are organized in different overlays.

According to Gartner forecast,[2] it is estimated that by 2020 over 20.5 billion

[2]https://www.gartner.com/newsroom/id/3165317, Accessed: 07-03-2019

devices will be connected to the Internet and will generate more than 43 trillion gigabytes of data. Also, CISCO[1] estimates that the mobile data traffic alone will reach 49.00 exabytes per month in 2021, with a growth of 47% from 2016, when the overall mobile data traffic was 7.00 exabytes. These numbers pose high network loads and create major computing challenges that alter the Quality-of-Service (QoS) and Experience (QoE). Increasing the number of centralized cloud data centers will not be a proper solution for solving these challenges. We must take into consideration a different approach by shifting the existing computing ecosystems beyond data centers towards the users. This new paradigm is reflected in emerging trends (see Figure 1.1) based on Peer-to-Peer systems and include: volunteer computing ([38]), fog and mobile edge computing ([112]), server-less computing ([93]) and software-defined computing ([134]).

Starting from the users of social networks ([23], [24]) to Gamisfication ([120]), volunteer computing may have different forms ([89], [35], [33]).

The leveraging of the computing resources to edge nodes, such as mobile devices, network equipment, sensors, or smart devices, represent the premise of the mobile edge computing ([139], [59]) and fog computing ([84]). The main constraint, in this case, is the fact that these nodes are resource limited and are prone to malfunction causing unexpected failures in the system.

The concept of server-less computing doses does not imply an architecture without servers ([65], [123]) but rather the fact that the server is not rented as a conventional cloud server and developers do not think of the server and the residency of applications on a cloud virtual machine.

In order to facilitate the data transfer from one location to another, the support applications must rely on a dynamic architecture ([75], [97]). An approach for this challenge is software-defined networking, where the underlay of the network is isolated from the logical communication overlay.

The main objective of this Ph.D. thesis is the research, development and evaluation of a dependable framework for the management of different large-scale heterogeneous distributed systems in generic network overlays that assures security and trust of processed data. Although the resources are not reliable, through the algorithms developed, the proposed solution creates a trusted computation and communication overlay concerning fault-tolerance,

[1]https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html, Accessed: 07-03-2019

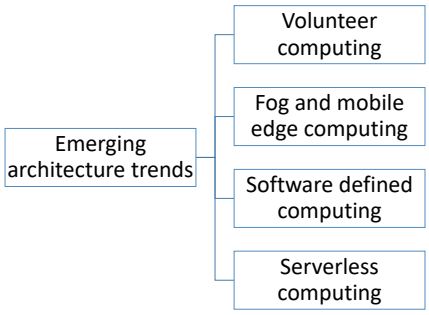


Figure 1.1: Emerging architecture trends and directions.

security and reliability.

We identify four scenarios that can benefit from our research and solutions:

- Cyber-physical Infrastructures;
- Smart-cities;
- Internet of Things;
- E-commerce systems.

The research perspectives of this thesis are designed as a 3 level iteration process as follows:

- Level 1: SPIDER - an adaptive Peer-to-Peer overlay that scales over heterogeneous and unsafe data sources in a fault-tolerant way considering as main challenges the reliability/uptime of the applications, performance, and data integrity, and having as results storage/processing mechanisms and consistency;
- Level 2: SPIDER - an auto-adaptive trusted Peer-to-Peer overlay that uses reliable and unreliable physical resources with the main challenges of ensuring deadlines, budget, QoS, and results in generic principles and algorithms for self-* and dependable high-level organization;
- Level 3: real-life use cases of SPIDER, an adaptive Peer-to-Peer overlay for large-scale heterogeneous and unsafe data sources: networked systems (Cyber-physical Infrastructure systems, Smart-cities), sensor networks (Internet of Things) and trust management systems (E-commerce).

This thesis is one step towards solving via technological advances in "data-centered" networking and cloud computing one of the significant

challenges concerning heterogeneous large-scale distributed systems. Another essential technological advancement would be to offer the possibility of connecting directly multiple heterogeneous data sources via flying clouds [19]. Our proposal can be a viable alternative to the Content Distribution Networks (CDN) which, to the best of our knowledge, do not offer a solution for unreliable data sources.

1.1 Problem Statement and Thesis Objectives

The research presented in this thesis addresses the questions regarding the design and implementation of auto-adaptive Peer-to-Peer overlays networks that assure robustness and trust of processed data. These questions investigate the architecture of an auto-adaptive Peer-to-Peer overlay, the response of the Peer-to-Peer overlay to failure, the integration of data in Peer-to-peer overlays and what type of applications are suitable in auto adaptive Peer-to-Peer overlay networks.

RQ1.How to design a Peer-to-Peer overlay that is auto-adaptive and scales over heterogeneous data sources? We have designed a bio-inspired Peer-to-Peer overlay based on the natural shape of the spider web, called SPIDER. The advantage of this structure is the processing manner as a two dimension matrix formed by chains and rings. Another aspect relevant in this architecture is the fact that each node has maximum four neighbors. A formal description of the SPIDER Peer-to-Peer overlay is presented in Chapter SPIDER: Auto-Adaptive Peer-to-Peer Overlay.

RQ2.How to create a fault-tolerant Peer-to-Peer overlay and what types of failure is it tolerant to? In order to achieve auto-recovery with a minimal cost, the design of SPIDER Peer-to-Peer overlay is taken into consideration especially because of its structure based on rings and chains. A comprehensive analysis of the fault tolerance properties of the SPIDER overlay are presented in Chapter Robustness of SPIDER Peer-to-Peer Overlay.

RQ3.How to aggregate data from unreliable heterogeneous sources for management decision support systems? We have proposed novel methods for data fusion, in large scale distributed systems organized in the SPIDER Peer-to-Peer overlay, based on its architecture These methods are outlined in Chapter Data Fusion in SPIDER Peer-to-Peer Overlay.

RQ4. What are the requirements and the architectures for state of the art applications that gather data from many heterogeneous sources and bind them together for better decision management? In order to determine the type of applications that are suitable for the SPIDER Peer-to-Peer overlay architecture, we have to define the requirements of those applications. Therefore, we have proposed a set for real-life distributed applications in Chapter SPIDER Peer-to-Peer Overlay Applications. .

Furthermore, taking into consideration the research questions presented above, we define the specific objectives of this thesis:

- design an auto-adaptive Peer-to-Peer overlay that scales over heterogeneous and unsafe data sources. Present the structure, the most common operations that can be realized and evaluate the overlay - SPIDER Peer-to-Peer overlay;
- offer a fault tolerance framework that is suitable for the SPIDER Peer-to-Peer overlay evaluate it;
- provide a data fusion framework for the SPIDER overlay and evaluate its algorithms;
- identify a set of requirements for applications of Peer-to-Peer overly networks;
- present a pack of solutions that can benefit from the usage of the SPIDER Peer-to-Peer overlay and evaluate their performances:
  - live video-streaming for Cyber-physical Infrastructures(CPIs);
  - Peer-to-Peer applications for heterogeneous and unsafe data sources in Smart cities;
  - trust management in E-commerce systems.

1.2 Research Methodology

In this section, we elaborate on the research methodology applied in this thesis. The methods used represent a set of generic operations performed in such a manner in order to answer the research questions identified in the previous section.

Therefore, we offer in Table 1.1 a brief overview of the research methods used in this thesis.

Table 1.1: Overview of the research methods used in this thesis.

Challenge	Research question	Thesis chapter	Research method
Design of SPIDER	RQ1	3	Fundamental,
Robustness of SPIDER	RQ2	4	Fundamental, qualitative, quantitative and empirical
Data fusion in SPIDER	RQ3	5	Fundamental, qualitative, quantitative and empirical
Applications of SPIDER	RQ4	6	Fundamental, qualitative, quantitative and empirical

We aim to find a generic solution for the design of an innovative trusted auto-adaptive Peer-to-Peer overlay based on the spider web construction natural phenomenon through fundamental research. Furthermore, we present the properties of the SPIDER overlay and we prove the advantages and this Peer-to-Peer structure in comparison with other communication overlays through qualitative and quantitative and empirical research methods. We asses the performance of the SPIDER Peer-to-Peer overlay through a realistic implementation in several projects developed in UPB such as ClueFarm, DataWay and CyberWater with real datasets.

Although each chapter has its own flavor, throughout this thesis we follow the research methodology based on specific methods in order to identify general solutions for problems such as auto-adaptability, robustness, data dissemination, and trust management in the SPIDER based Peer-to-Peer networks.

Therefore, we enclose the qualitative, quantitative and empirical research methods in a realistic software implementation, and evaluate the properties of the SPIDER Peer-to-Peer overlay with real datasets in real use cases.

We develop a realistic implementation of the SPIDER Peer-to-Peer overlay over TCP ([104]). The architecture of the implementation is presented in Figure 1.2.

The main components of the implementation are:

- the SpiderOverlay package;
- the Communication package;
- the Logger package;
- the Deploy package;
- the Message package.

In the Deploy package of the implementation, we initialize the number of peers of the network and set its threads and workers. This package does not contain any

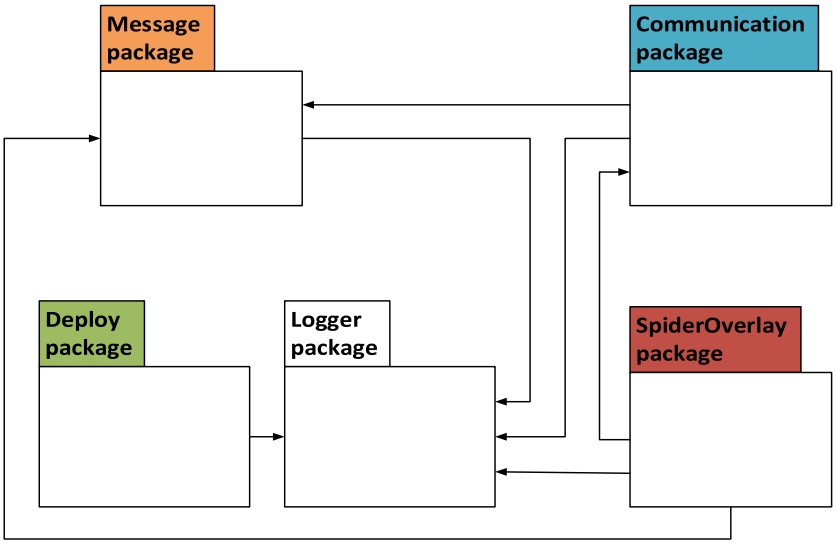


Figure 1.2: The implementation of the SPIDER Peer-to-Peer overlay architecture.

Peer-to-Peer computation logic. Its main purpose is only to create and initialize the environment for the real network.

The Logger package offers the log-trace of the implementation in a text file format. The pieces of information that are logged are:

- the node ID;
- the node chain;
- the node ring;
- the message;
- the timestamps of the transactions.

The Communication package provides the ability of the solution to use the TCP protocol by implementing the .NET System.Net.Sockets[1] in an asynchronous mode.

The Message package contains the classes used to pass the information between the peers of the system. This implementation is realized in a serialized manner based on Json.NET framework[2]

[1]https://docs.microsoft.com/en-us/dotnet/api/system.net.sockets.socket?view=netframework-4.7.2, Accessed: 15-01-2019

[2]https://www.newtonsoft.com/json, Accessed: 15-01-2019

The SpiderOverlay package contains the main logic of the SPIDER Peer-to-Peer overlay. The classes in this package defines the architecture of the overlay and the manner in which the nodes interact with each other. In Figure 1.3, we present the architecture of the Peer-to-Peer overlay package.

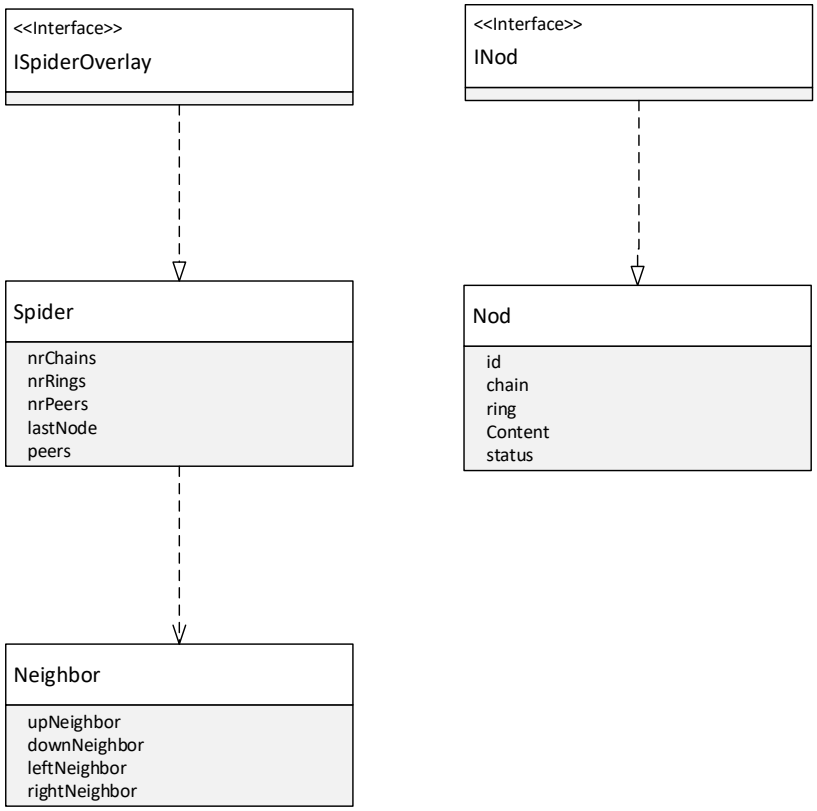


Figure 1.3: SpiderOverlay package.

The SPIDER overlay is defined by the number of chains and rings as listed in 1.1.

```
1
2 public SpiderOverlay(int nrChains, int nrRings)
3 {
4 this.NrChains = nrChains;
5 this.NrRings = nrRings;
6
7 this.NrPeers = nrChains * nrRings;
8 Peers = new List<Nod>();
9 LastNode = new Nod();
```

```
10 }
```

Listing 1.1: SpiderOverlay class contructor.

In the constructor, we define the number of chains and rings, the total amount of peers, a list of nodes and the last node joined the overlay. The overlay structure is stored in a linked list of nodes. Each Node is defined as Listing 1.2.

```
1  public Nod(int chain, int ring, bool flag)
2  {
3  Chain = chain;
4  Ring = ring;
5  Id = chain.ToString() + "-" + ring.ToString() + "␣";
6
7  Content = CreateContnt();
8
9  if (flag == true)
10 {
11 IsSuperPeer = true;
12 }
13 else
14 {
15 IsSuperPeer = false;
16 }
17
18 Status = new Neighbour();
19 }
```

Listing 1.2: Nod class constructor.

Each node stores its position in the SPIDER overlay through the ring and chain. Also, each node has a flag that indicates whether it's super-peer or not. In the SPIDER overlay, every node can become super-peer some time. Also, each node has a status variable the indicates the availability of its neighbors: left, right, top, bottom.

1.3 Thesis Outline

After attaining the necessary background of Peer-to-Peer networks (2) (Peer-to-Peer Overlays: Comparative Analysis), we move to the core of this thesis. Chapter 3 (SPIDER: Auto-Adaptive Peer-to-Peer Overlay) propose and describe the SPIDER bio-inspired and auto-adaptive Peer-to-Peer overlay that scales over heterogeneous and unsafe data sources. Chapter 4 (Robustness of SPIDER Peer-to-Peer Overlay) and Chapter 5 (Data Fusion in SPIDER Peer-to-Peer Overlay) addresses the robustness of the SPIDER Peer-to-Peer overlay and the data fusion methods with the respect to the SPIDER Peer-to-Peer overlay limitations. Chapter 6 (SPIDER Peer-to-Peer Overlay Applications) investigate the requirements of applications based on the Peer-to-Peer overlay networks and present se pack of applications based on the SPIDER Peer-to-Peer overlay.

Furthermore, we answer the research questions through the following scientific contributions.

Propose the SPIDER Peer-to-Peer overlay - a bio-inspired, auto-adaptive Peer-to-Peer overlay that scales over heterogeneous and unsafe data sources. In Chapter 3 (SPIDER: Auto-Adaptive Peer-to-Peer Overlay) we answer the research question RQ1, by designing an auto-adaptive Peer-to-Peer overlay based on the spider web architecture. Also, in this chapter, we present and evaluate the most common operations available in the SPIDER overlay. This chapter is based on the following publication as part of the PROSCIENCE Posdru project[1]:

Mocanu, B. et al., 2015. SPIDER: A Bio-inspired Structured Peer-to-Peer Overlay for Data Dissemination. 2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC). doi:10.1109/3pgcic.2015.121. [Conference Paper, ISI proceedings].

Offer a fault tolerance generic framework for the SPIDER Peer-to-Peer overlay.In Chapter 4 (Robustness of SPIDER Peer-to-Peer Overlay) we answer the research question RQ2, by proposing a set of algorithms for flaw recovery in the SPIDER overlay and evaluate them. The presented set

[1]Sectorial Operational Programme Human Resources Development 2007-2013 of the Ministry of European Funds through the Financial Agreement PROSCIENCE: Science and Research Quality Promotion through Ph.D. Scholarships POSDRU/187/1.5/S/155420 (2015-2016, http://www.proscience.upb.ro/index.html, Accessed: 07-03-2019)

of algorithms is one step toward solving the issues presented in DataWay[1] research project developed in UPB. This chapter is based on the following publication:

Mocanu, B. et al., 2017. Flaw Recovery in Cloud Based Bio-inspired Peer-to-Peer Systems for Smart Cities. Lecture Notes in Computer Science, pp.338–352. doi: 10.1007/978-3-319-57186-7_ 26. [Conference Paper, ISI proceedings].

Design a generic data fusion framework for the SPIDER overlay. In Chapter 5 (Data Fusion in SPIDER Peer-to-Peer Overlay) we answer the research question RQ3 by dealing with data fusion methods with the respect of the SPIDER Peer-to-Peer overlay limitations. Also, in this section, we extend the fault tolerance methods presented in section 4 with the features of the data fusion. Finally, in this chapter, we evaluate the proposed algorithms. This chapter is based on the following publication:

Mocanu, B. et al., 2019. Data fusion technique in SPIDER Peer-to-Peer networks in smart cities for security enhancements. Information Sciences, 479, pp.607–621. doi: http://dx.doi.org/10.1016/j.ins.2018.06.070. [ISI Journal, IF = 4.378, Q1].

Present a pack of solutions that can benefit from the usage of the SPIDER Peer-to-Peer overlay and evaluate their performances.In Chapter 6 (SPIDER Peer-to-Peer Overlay Applications) we answer the research question RQ4 by investigating the requirements of applications based on the Peer-to-Peer overlay networks. Also, we mapped the application type with the fault type for the SPIDER Peer-to-Peer structure. Furthermore, we proposed and evaluated two applications of the SPIDER overlay for live video streaming in CPIs. The presented applications are one step toward solving the issues presented in CLUeFARM[2] research project and in CyberWater[3] project

[1]DataWay: Real-time Data Processing Platform for Smart Cities: Making Sense of Big Data Project - PN-II-RUTE- 2014-4-2731 (2015-2017, Project director: prof. Florin Pop, https://dataway.hpc.pub.ro/, Accessed: 07-03-2019)

[2]CLUeFARM: Information System Based on Cloud Services, Accessible through Mobile Devices, for Quality Improvement of Products and Business Development in Farms - PNII no. 29/2014 grant of the Romanian National Authority for Scientific Research and Innovation (2014-2017, Project director: prof. Valentin Cristea, https://cluefarm.hpc.pub.ro/, Accessed: 07-03-2019)

[3]CyberWater: Prototype Cyber Infrastructure-based System for Decision-Making Support in Water Resource Management - PNII no. 47/2012 grant of the Romanian National Authority for Scientific Research and Innovation (2012-2016, Project director: prof. Mariana Mocanu, https://cw.hpc.pub.ro/, Accessed: 07-03-2019)

developed in UPB. Moreover, in this chapter, we propose and evaluate three applications for the Smart-city concept: Smart-streets, a new approach for path optimization in crowded cities, IoT support for smart-homes and IoT support for waste recycling. Furthermore, in this section, we present and evaluate a trusted e-Commerce system based on the SPIDER Peer-to-Peer overlay. This chapter is based on the following publication:

Pop, F. et al., 2015. Trust models for efficient communication in Mobile Cloud Computing and their applications to e-Commerce. Enterprise Information Systems, 10(9), pp.982–1000. doi: 10.1080/17517575.2015.1100756. [ISI Journal, IF = 1.683, Q3].

Finally, in Chapter 7 (Conclusions) of this thesis we present an overview of the way we achieve the objective of this research, our main contributions, draw our conclusions and present the remaining open questions.

Chapter 2

Peer-to-Peer Overlays: Comparative Analysis

Internet-connected devices and sensors are transforming the way businesses, and governance of business is interacting with the world. This aspect is enforced by the fact that worldwide companies are willing to spend over 5 trillion dollars in the next five years according to [8].

This chapter presents a critical analysis of unstructured Peer-to-Peer overlays (Section 2.2) (Critical Analysis of Unstructured Peer-to-Peer Overlays), structured Peer-to-Peer overlays (Section 2.3) (Critical Analysis of Structured Peer-to-Peer Overlays) and bio-inspired Peer-to-Peer overlays (Section 2.4) (Critical Analysis of Bio-inspired Peer-to-Peer Overlays), a brief description of the most important terms used in this thesis (Section 2.1) (Background), as well as a brief open issues for Peer-to-Peer overlays (Section 2.5) (Challenges in SPIDER Peer-to-Peer Overlay). Also, in this chapter we present our discussions and conclusions (Section 2.6) (Discussions and Conclusions).

Based on these premises, classical Client-Server approaches are not feasible anymore because the number of interconnected devices is massive. Thus, we take into consideration a distributed non-centralized approach. One of the most suitable paradigms for such networks is the Peer-to-Peer overlay approach. There are several significant advantages of Peer-to-Peer networks in comparison with the traditional Client-Server systems. The most important benefits of the Peer-to-Peer networks are higher complexity, heterogeneity, mobility, and dynamicity.

High degrees of heterogeneity and mobility, lead to very challenging topics in the research community entitled Ubiquitous computing [76] and Pervasive computing [37]. One of the most critical aspects of highly adaptable networks is the fact that the nodes join and leave the system very frequently unexpectedly. This phenomenon happens because of the short battery lifetime, weak Internet Service Provider (ISP) signal, or even on purpose. To deal with such a great dynamic environment researchers have designed a natural phenomenon inspired Peer-to-Peer concept called bio-inspired overlays.

In this chapter we analyze multiple Peer-to-Peer overlays and identify their advantages and disadvantages and introduce an auto-adaptive Peer-to-Peer overlay based on the spider web.

We split the critical analysis of the existing solutions for Peer-to-Peer overlays into three categories based on their taxonomy: unstructured, structured and bio-inspired overlays. Also, we synthesize the conclusions of these in 3 tables that are listed below.

To validate the feasibility of the SPIDER Peer-to-Peer overlay we take into consideration several criteria such as redundancy, dynamicity, load-balancing features, security, lookup methodology, query types and auto-adaptiveness.

2.1 Background

When talking about Peer-to-Peer networks, it is essential to emphasize the most important terms in an easy to understand manner.

According to [115] a distributed network architecture may be called a ***Peer-to-Peer*** (P-to-P, P2P) network if the participants share a part of their hardware resources (processing power, storage capacity, network link capacity, printers).

The authors of [7] present a wide range of definitions of the most important terms used in Peer-to-Peer networks. The structure of the network Peer-to-peer systems can be ***unstructured***, which means that the location of the content is unrelated to the overlay ***structure***. Another type of Peer-to-Peer structure is structured Peer-to-Peer, which means that the overlay design is bound to the network.

Bio-inspired Peer-to-Peer networks are that type of structured Peer-to-Peer systems that take the advantages of natural phenomenon structures such as:

Swarm intelligence and social insects ([40], [85], [45], [79], [41]), spider web ([94]), epidemic spreading ([136], [22], [156]).

The most important metrics and properties that we take into consideration when talking about Peer-to-Peer systems are ***redundancy, dynamicity, security, lookup complexity load-balancing, auto-adaptability and fault tolerance***. Auto-adaptability is the feature of a Peer-to-Peer overlay to change its structure to maintain its functionality. Redundancy of a Peer-to-Peer overlay represents the property to store the same content in various locations to be able to retrieve it in case of failure. A dynamic Peer-to-Peer overlay is a structure that can change the node positions for a specific task. The look-up complexity is a fundamental metric of Peer-to-Peer overlays because it measures the number of hops needed by a node to fetch another node in the system. Concerning security, for Peer-to-Peer systems, the most important feature is the trust. This feature assures the fact that the peers are trusted.

On the other hand, Peer-to-peer networks can also be used for ***data dissemination***, which adds a modified form of publish/subscribe methodology.

2.2 Critical Analysis of Unstructured Peer-to-Peer Overlays

In fully decentralized networks, every node is free to participate. This aspect leads to high degree of robustness and resilience. On the other hand, this type of networks is suffering from frequent changes in architecture because peers join and leave the system without any warning anytime, thus keeping the robustness of the network is considered to be the most challenging aspect.

The performance of lookups in unstructured Peer-to-Peer overlays is not so excellent because it is realized through flooding the network, which creates a considerable overhead.

Furthermore, we chronologically analyze several unstructured Peer-to-Peer overlays.

Freenet (2001)

The Freenet unstructured Peer-to-Peer overlay was proposed in paper [32]. This free approach has allowed peers to exchange information anonymously. The primary goals of Freenet are reliability and security. Even though Freenet is entirely decentralized, and there are no rules concerning node joining and leaving, it has a proper mechanism to facilitate information retrieval without flooding.

Joining the Freenet overlay is realized by merely finding an existing peer in the network and connecting to it. When the new nod has become part of the overlay, it holds its data-store along with a routing table to other peers and also an index of their stored resources. Each piece of a resource is uniquely identified through an indexing system, that is computed by all nodes with a hash function. Therefore, lookups in Freenet are realized through this unique keys. When a request for a key can be satisfied by a distant node, the data will return to the node that initially queried it. along the reverse path

The protocol of the Freenet Peer-to-Peer overlay is depicted in Figure 2.1.

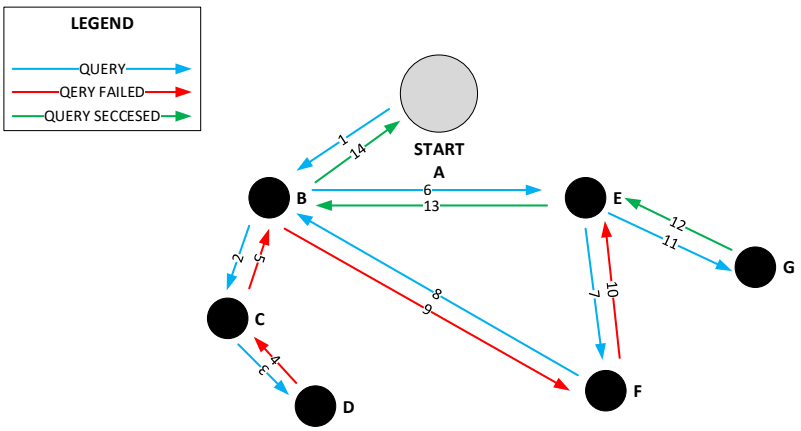


Figure 2.1: The Freenet Peer-to-Peer overlay.

Leaving the Freenet overlay is realized asynchronously, without any announcement in the network. The resources stored by the node how has just left diaper until the peer joins the overlay again if there is no other peer in the system with that resources.

Gnutella (2002)

The most widely known unstructured Pee-to-Peer overlay is Gnutella [73]. The purpose of this overlay is to allow peers to share resources in a fully decentralised, scalable, reliable and anonymous way.

Similar to Freenet overlay, joining in Gnutella is realized by searching an existing node and contacting it. When a node joins Gnutella can execute 3 operations:

- Locate and connect to other nodes;
- Query and retrieve resources;
- Push data resources.

Furthermore, to execute the operations mentioned above, Gnutella protocol has 5 type of messages and can be seen in Figure 2.2:

- PING – check if a node is available;
- PONG – reply of the PING message if the node is available;
- QUERY – search for a particular resource;
- QUERYHIT – reply from a node that has the queried resource;
- PUSH – file downloads from nodes that are behind firewalls.

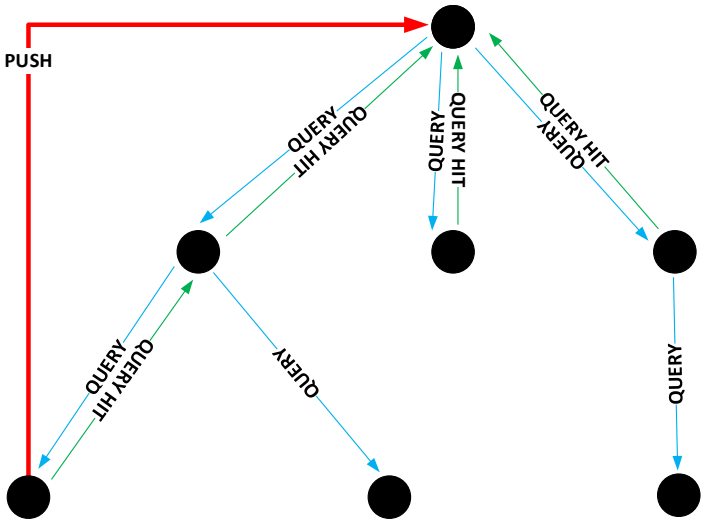


Figure 2.2: The Gnutella P2P overlay.

FastTrack (2006)

FastTrack [81] is a fully decentralized unstructured Peer-to-Peer overlay with a two-tier hierarchical topology. In this overlay, there are two kinds of peers: super-nodes and ordinary nodes.

Concerning bootstrapping, every node stores locally a super-node list that carries specific workload information. New nodes select the most appropriate super-node to connect to, based on the information about workload and its location. In terms of lookup, each ordinary node stores an index of resources, that is shared with its associated super-node, thus guaranteeing that super-nodes can correctly reply about resources of all their underlying regular nodes.

The FastTrack protocol is depicted in Figure 2.3.

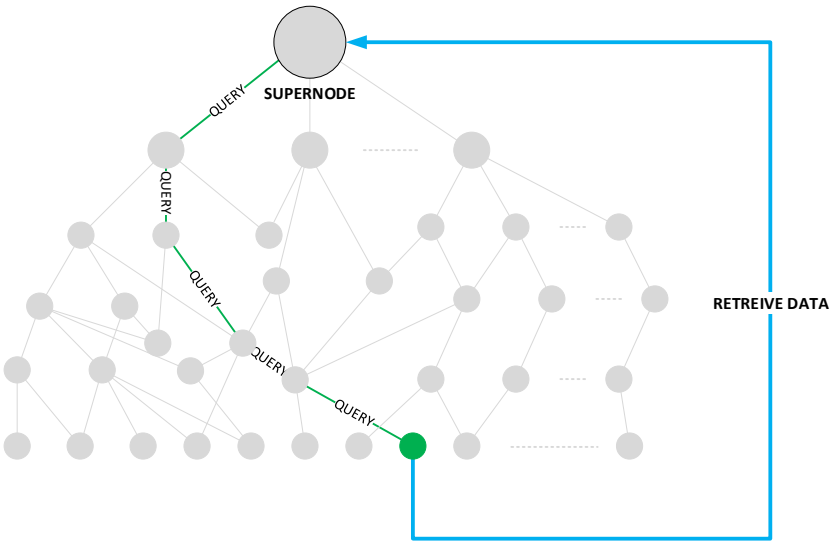


Figure 2.3: The FastTrack Peer-to-Peer overlay.

Analysis

In Table 2.1 we present a multi-property analysis of several unstructured Peer-to-Peer overlays. For this appraisal we take into consideration the following properties and metrics:

- auto-adaptiveness - the property of a Peer-to-Peer system to auto-adapt their structure in different scenarios;

- dynamicity - the property of a Peer-to-Peer system to have mobile nodes;
- redundancy - the property of a Peer-to-Peer system to ensure a redundant manner to store data;
- Security - all features of a Peer-to-Peer system that ensure that the system is reliable;
- queries - this property refers to the type of queries available in a Peer-to-Peer system.

In terms of auto-adaptability, the SPIDER Peer-to-Peer overlay is the only Peer-to-Peer overlay that is auto-adaptive in comparison with Freenet, Gnutella and FastTrack overlays. The auto-adaptability of the SPIDER overlay is guaranteed by its architecture developed on rings and chains.

The redundancy and the fault-tolerance properties of the SPIDER overlay are structured oriented (chain and ring based). In comparison with the other analyzed overlays, SPIDER is the only one that offers fault-tolerance.

In terms of security, SPIDER Peer-to-Peer overlay offers very good trust. In comparison with the other overlays, SPIDER is the only overlay that has build-in trust features.

Table 2.1: Taxonomy of unstructured Peer-to-Peer overlays.

Evaluated metric	**Freenet**	**Gnutella**	**FastTrack**
Auto adaptability	No	No	No
Redundancy	Path redundancy	No	Resource indices replication
Dynamicity	Distributed design	Periodic initiation of node discovery	No
Security	Anonymity and privacy	Anonymity	No
Queries	Any type of queries	Any type of queries	Any type of queries

2.3 Critical Analysis of Structured Peer-to-Peer Overlays

Structured Peer-to-Peer overlays represent a type of network organization type where data is distributed in a deterministic manner, not randomly. The access to specific information is realized in a finite number of steps based on an

algorithm. The main advantage of structured Peer-to-Peer overlays consists of high performance concerning resource discovery and resource access.

Furthermore, we analyze several structured Peer-to-Peer overlays in chronological order.

CAN (2001)

The most mature structured Peer-to-Peer overlays is CAN (Content Addressable Network). Ratnasamy proposed CAN overlay in article [109]. Based on DHT (Distributed Hash Tables), CAN overlays present the following features: scalability, self-organization along with fault tolerance. The organization of nodes in a CAN overlays are in a multidimensional coordinate space.

In CAN overlays, every peer has an assigned a key/value pair. Based on this fact every node has the responsibility of a certain area in the CAN space, by storing additional information regarding its neighboring such IP addresses or other coordinates. The critical value stored by each node is generated based on a hash function because this cryptographic operation has no proven collisions 2.4.

Furthermore, we analyze the fundamental operations in Peer-to-Peer overlays: joining, leaving and lookup.

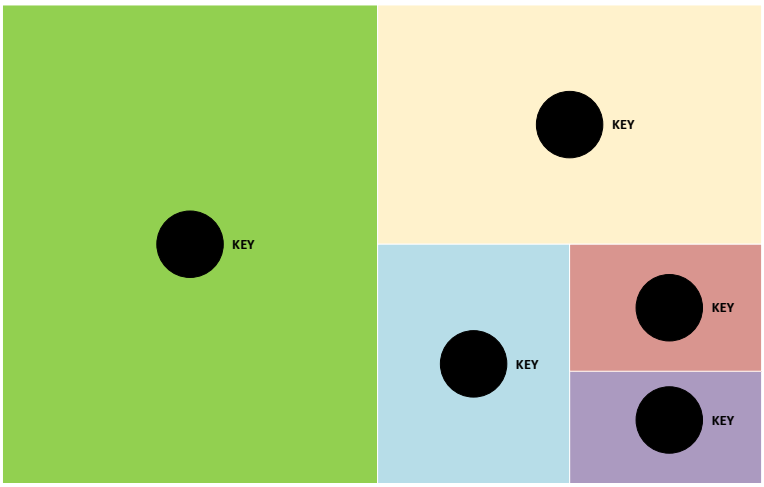


Figure 2.4: The CAN Peer-to-Peer overlay.

Joining the CAN overlay must follow three fundamental steps: finding a node that already is in the overlay, determine the zone that will be allocated to that node and update its neighbors' state tables. Therefore, when a node joins the overlay, a point in the CAN space is picked and after this operation it is sent a JOIN message through a known node in the network. When the targeted node receives a JOIN message, it will begin the process of zone allocation for the new node, by splitting its zone in half and assigning it to the newly arrived node. After the zone allocation, the targeted node transfers the key/value pairs to the entered node for which it has become responsible. Finally, allocation tables neighbors' are updated to be aware of the new peer through UPDATE messages.

Leaving the CAN overlay deliberately by a node occurs when the node notifies the neighbor with the smallest zone and transfers all the content to keep de resources available.

Moreover, according to [110], the average routing path length ($L_{routing-path}$) in a d dimensional space divided in n equal areas is given by Equation 2.1. This means that, for a d dimension space, the number of nodes cad be increased while the per node state remains the same.

$$L_{routing-path} = (d/4)n^{1/d} \tag{2.1}$$

Therefore, we consider the CAN overlay reliable in respect of its construction.

An exciting application based on the CAN overlay is presented in [110]. The authors proposed a solution for an application-level multi-cast service. Their proposal has two significant advantages: simplicity of the schema because of the architecture of CAN and scalability regarding large groups without modifying the service architecture.

Another application based on CAN overlay is presented in paper [44]. This paper presents a prototype for document sharing and text classification over CAN. The authors proposed an algorithm for classification that produces a key value for each document found in the system.

Kademlia (2002)

Maymounkov and Mazieres proposed Kademlia overlay in paper [92]. This Peer-to-Peer overlay is entirely decentralized and structured. This overlay uses

the XOR operation to measures the distance between 2 nodes a and b (see Equation 2.2). The Kademlia overlay structure is presented in Figure 2.5

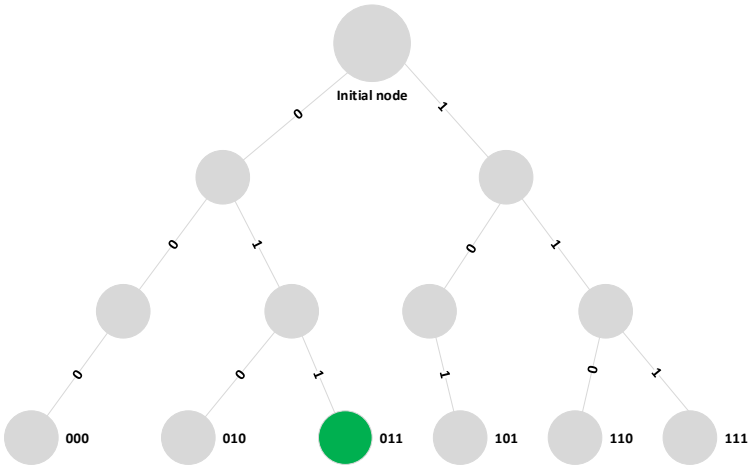


Figure 2.5: The Kademlia Peer-to-Peer overlay.

Each node in the overlay stores a table of nodes that has a distance from itself between $2i$ and $2i + 1$, where $i \in [0, 160)$. This table is called a κ bucket, with a dynamic size given by parameter k.

$$d(a, b) = a \oplus b \tag{2.2}$$

Joining the Kademlia overlay means that the new node will appear in a known nodes κ bucket and after that, through lookup operations, the new node populates its list of node distances. When a new node that has not yet participated yet, wants to join the overlay, it computes a random number that has not been used by other peers and assign this as it's ID. The peer keeps this value until it leaves the system.

By using κ buckets, the Kademlia overlay is considered to be resistant to DoS (Denial of Service) attacks, because by its construction the κ bucket is not populated with the new node until an old node leaves the system.

Kademlia overlay is one of the most popular overlays implemented in open-source applications like: BitTorrent [105] and eMule [78].

Chord (2003)

Even though Chord [125] is not the most mature structured Peer-to-Peer overlay is considered the most popular one. The Chord protocol uses consistent hashing for identifying each node or resource in the system.

Moreover, based on its construction Chord is entirely decentralized and distributed. Therefore, the structure of this overlay consists of a virtual ring, as shown in Figure 2.6, where every node has an identifier more significant than it's higher than its predecessor's one and has knowledge only on its successor neighbor.

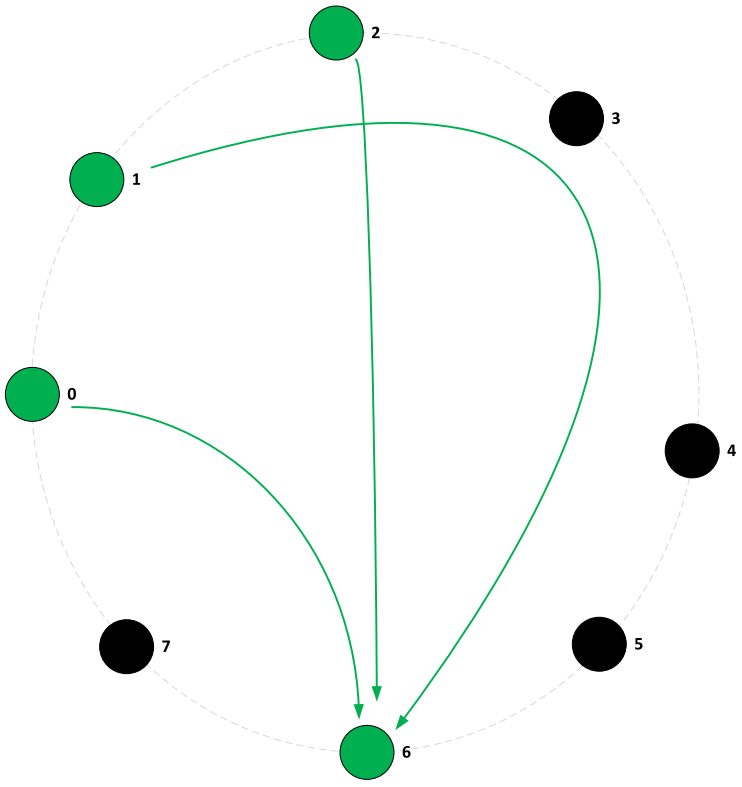


Figure 2.6: The Chord Peer-to-Peer overlay.

When a new node joins the system, it must satisfy three conditions:

- Every node predecessor points correctly to its first successor;
- Each key is stored in successor k;

- For each node the finger table is correct.

When a node leaves the overlay, the robustness of the system is not affected because each node keeps a list of successor nodes of a fixed size. If the immediately next successor fails to respond the node, it can modify the entry point of the second successor and become the first one.

Considering a stable N-node Chord overlay the number of messages needed to locate a specific resource is given by $O(\log N)$.

Based on Chord overlay researchers have found an exciting application for Wireless Mesh Chord [20]. In this paper, the authors present a new approach called MeshChord, that explores the features of wireless mesh networks.

Analysis

In Table 2.2 we present the taxonomy of structured Peer-to-Peer overlays. For this critical analysis, we take into consideration the same properties and metrics as the unstructured Peer-to-Peer overlays:

- auto-adaptiveness;
- dynamicity;
- redundancy;
- Security;
- lookup complexity;
- load-balancing;
- fault-tolerance;
- queries.

In terms of auto-adaptability, the SPIDER and Chord Peer-to-Peer overlays are the only Peer-to-Peer overlays that are auto-adaptive in comparison with CAN and Kademlia overlays. The auto-adaptability of the SPIDER overlay is guaranteed by its architecture developed on rings and chains.

In comparison with the other analyzed overlays, SPIDER and Chord offer fault-tolerance properties. In terms of redundancy, SPIDER overlay benefits from its structure thus, it offers chain and ring based redundancy.

Table 2.2: Taxonomy of structured Peer-to-Peer overlays.

Evaluated metric	**SPIDER**	**CAN**	**Kademlia**	**Chord**
Auto adaptability	Yes	No	No	Yes
Redundancy	Chain based Ring based	Routing redundancy	Resource replication and caching	Yes No
Dynamicity	Yes	No	Node behavior for routes longevity	No
Security	Trust	No	Old nodes more trusted	No
Lookup complexity	$O(\sqrt{N})$	$O((d/4)n^{1/d})$	$O(logN)$	$O(logN)$
Load balancing	Yes	No	No	Yes
Fault tolerance	Chain based Ring based	No	No	Yes
Queries	Any type of queries	Standard queries	Standard queries	Standard queries

Another very important metric taken into consideration in this analysis is the lookup algorithm complexity. As it can be seen SPIDER, has fixed square root time complexity given by $O(\sqrt{N})$, where N is the total number of nodes in the overlay. In comparison with the other overlays, Chord overlay offers a logarithmic lookup complexity given by $O(logN)$, where N is the total number of nodes in the overlay.

In Figure 2.7 we present a numerical analysis of the number of maximum hops in SPIDER and Chord overlays. The obtained values show the fact that in terms of number of hops the Chord overlay is more efficient than SPIDER Peer-to-Peer overlay. The maximum number of hops is higher for the SPIDER overlay in comparison with Chord because the architecture of the SPIDER overlay is based on the fact that each node can have maximum 4 neighbors.

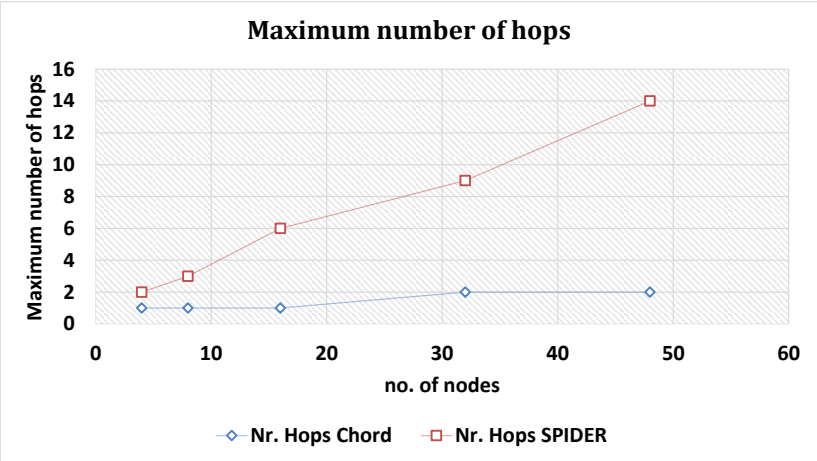


Figure 2.7: Maximum number of hops Chord Vs SPIDER.

Taking into consideration the types of queries that can be executed, in SPIDER overlay can be used any type of queries, in comparison with the other overlays, where can be used only standard queries.

In terms of security, SPIDER Peer-to-Peer overlay offers good trust properties as presented in Section 6.4 (Trust Management System for E-commerce in Peer-to-Peer Networks). In comparison with other overlays, Kademlia offers a trust mechanism based on the old nodes witch are more trusted.

2.4 Critical Analysis of Bio-inspired Peer-to-Peer Overlays

In the last years, a great interest has been shown by researchers for the nature-inspired design of self-adaptive Peer-to-Peer overlays. Modeling a natural phenomenon and implement a distributed computing algorithm on its model is a non-trivial task. Furthermore, we present several bio-inspired protocols for Peer-to-Peer systems. These models were translated from a mathematical model of a natural phenomenon into a Peer-to-Peer protocol with predictable behavior

Bio-inspired solutions such as ant colonies or bees have shown a better efficiency for complex systems in the computer networks field in terms of scalability because of the reduced communication cost.

Furthermore, we present a sequential analysis of several bio-inspired.

SelfChord (2009)

The Self-Chord overlay was proposed in paper [49] as a solution for grid and cloud computing infrastructures. The construction of this overlay is based on ant colonies and swarm intelligence, using multiple independent mobile agents. For a better understanding of the Self-Chord a comprehensive example is presented in Figure 2.8. The values of the peers' indexes are depicted inside the ring, while the keys and values of the centroid are depicted outside of the ring in a sorted manner.

The architecture of the overlay is based on the Chord structure. Thus, the nodes in this Peer-to-Peer system are organized in a ring form. Each node is ordered

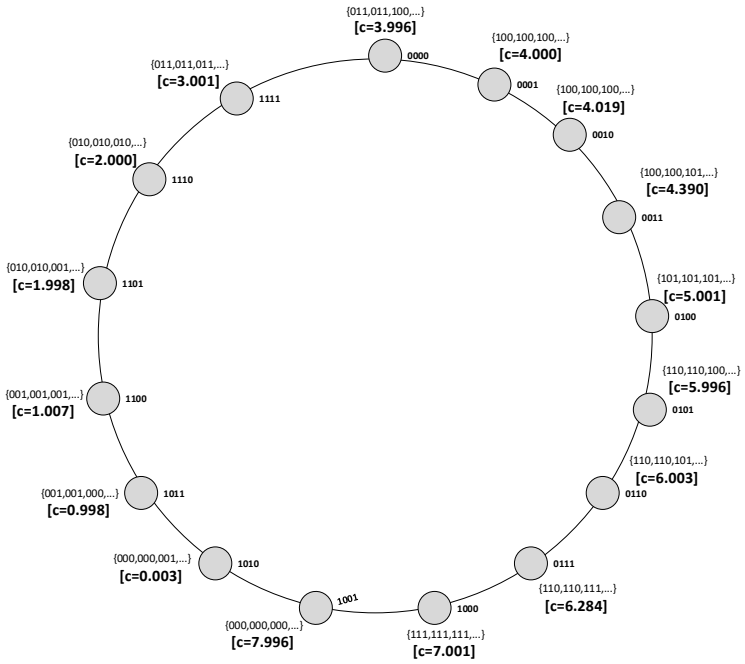


Figure 2.8: The Self-Chord Peer-to-Peer overlay.

according to its unique key, computed by using a hash function. The difference lies in the placement of resource identifiers. The main difference between Chord lies in the fact that resources obtain their ids from a different name-spaces.

Honeycomb (2014)

The Honeycomb overlay [102] is a Peer-to-Peer overlay that is based on the hexagons of the honeycomb. This model was developed in the research group from Distributed System in UPB. The honeycomb overlay (Figure 2.9) is a 3D system because each node in this overlay is uniquely identified a three coordinate point. The main advantage of this overlay is the fact that each node has a maximum of 3 direct neighbors.

The construction method of the Honeycomb Peer-to-Peer overlay uses the chains of the honeycomb, the indexes within each chain, which are set in a clockwise direction, and the coordinates from the three-axis system. The authors constructed the connection algorithm by using the lemma given in [126]: nodes of a honeycomb of size k can be coded by integer triples (x, y, z)

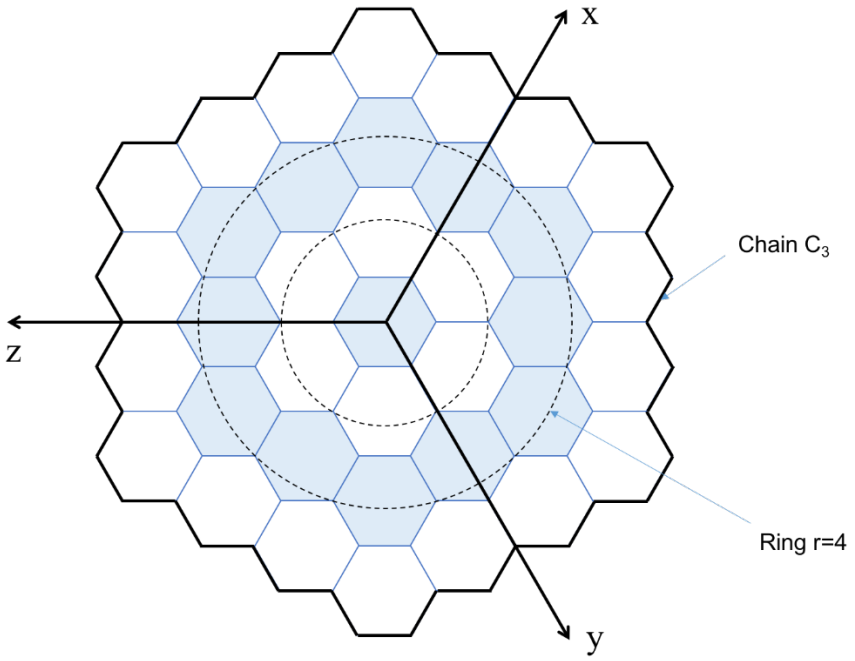


Figure 2.9: The Honeycomb Peer-to-Peer overlay.

using Equation 2.3 and 2.4.

$$-k+1 \leq x, y, z \leq k \quad and \quad 1 \leq x+y+z \leq 2 \tag{2.3}$$

Two nodes (x_1, y_1, z_1) and (x_2, y_2, z_2) are connected by an edge if:

$$|x_2 - x_1| + |y_2 - y_1| + |z_2 - z_1| = 1 \tag{2.4}$$

The coordinates of the neighbors of any node are obtained by traveling across edges: $(\pm 1, 0, 0)$, $(0, \pm 1, 0)$ and $(0, 0, \pm 1)$. Considering a node (x, y, z), we can compute a list of possible coordinates for its neighbors: $(x+1, y, z)$, $(x, y+1, z)$, $(x, v, z+1)$, $(x-1, y, z)$, $(x, y-1, z)$, and $(x, y, z-1)$. Out of these six triplets, only three define real neighbors. The correct coordinates are the ones that respect the relation $1 \leq x+y+z \leq 2$. The overview of the honeycomb structure is presented in Figure 2.9.

AFT (2018)

In [101], the researchers from the Computer Science and Engineering department of Politehnica University from Bucharest propose a Peer-to-Peer overlays structured a torus as shown in Figure 2.10. The AFT overlay is composed by rings, which represent the circles that are perpendicular on the plane of the inner circle and chain which are the circles parallel to the parallel plane to the inner circle.

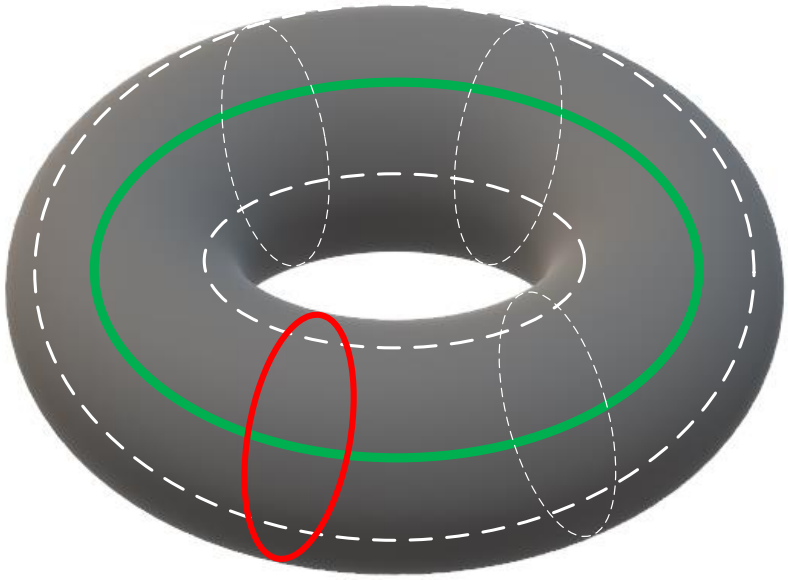


Figure 2.10: The AFT Peer-to-Peer overlay.

In the AFT overlay, each node stores the addresses of all the nodes on its ring and chain. This procedure is necessary for obtaining fault-tolerance.

In the AFT Peer-to-Peer overlay the maximum number of neighbors for a node is four. There are two neighbors on the same ring and two neighbors on the same chain. Therefore, taking into consideration this property, the SPIDER Peer-to-Peer overlay has a similar structure to the AFT Peer-to-Peer overlay.

The joining process in the AFT Peer-to-Peer overlay starts with a request to another node that has already joined the overlay before. The newcomer node is pushed along the overlay's links to the last node that has entered the overlay.

First, is checked if there is still room in the previous ring. In that case, the new node will join that position. Otherwise, a new ring is created. The joining process is terminated when all the nodes on its ring and chain acknowledge its existence.

Analysis

In Table 2.3 we present a critical analysis of several bio-inspired Peer-to-Peer overlays in comparison with the SPIDER Peer-to-Peer overlay. For this evaluation we take into consideration the following properties and metrics based on [36]:

- auto-adaptiveness;
- dynamicity;
- redundancy;
- Security;
- lookup complexity;
- load-balancing;
- fault-tolerance;
- queries.

Table 2.3: Taxonomy of bio-inspired Peer-to-Peer overlays.

Evaluated metric	**SPIDER**	**SelfChord**	**HoneyComb**	**AFT**
Auto adaptability	Yes	No	Yes	Yes
Redundancy	Chain based Ring based	No	Yes	No
Dynamicity	Yes	No	Yes	Yes
Security	Trust	No	Trust	Trust
Lookup complexity	$O(\sqrt{N})$	$O(logN)$	$O(\sqrt{N})$	$O(\sqrt{N})$
Load balancing	Yes	No	No	Yes
Fault tolerance	Chain based Ring based	No	Yes	Yes
Queries	Any type of queries	Standard queries	Any type of queries	Any type of queries

In terms of auto-adaptability and dynamicity, only SelfChord does not satisfy this property. The auto-adaptability properties of the other analyzed overlays are unique and innovative and are based on the architecture of each overlay.

The redundancy property can be found on the SPIDER Peer-to-Peer overlay and is based on the structure of the overlay but also on the Honeycomb overlay.

In terms of security, trust properties ca be found at the SPIDER overlay and also on Honeycomb and AFT overlay. SelfChord does not pride itself with any security properties.

Another very important metric is the lookup complexity. Shown before, SPIDER has fixed square root time complexity given by $O(\sqrt{N})$, where N is the total number of nodes in the overlay. In comparison with the other overlays, SPIDER, Honeycomb and AFT lookup algorithms have a square root complexity while SelfChod lookup algorithm has a logarithmic complexity of $O(logN)$, where N is the total number of nodes in the overlay.

Taking into consideration the types of queries that can be executed, in SPIDER can be used any type of queries, as well as in Honeycomb and AFT.

In terms of security, SPIDER Peer-to-Peer overlay offers very good trust features in comparison with the Kademlia overlay that claims a trust feature based on the old nodes witch are more trusted.

Moreover, for the fault-tolerance property analyzed, SPIDER overlay is obeying to this property through its original structure, therefore fault-tolerance is chain and ring based in case of the SPIDER overlay.

Even though, SPIDER Peer-to-Peer overlay is the third Peer-to-Peer overlay proposed by the research group from Distributed System in UPB, besides HoneyComb ([102]) and AFT ([101]) is the most versatile. Each node in the overlay has a maximum of 4 neighbors, which is optimal in terms of lookups, in comparison with the HoneyComb overlay. Despite the fact that SPIDER and AFT nodes have the same maximum number of neighbors, the AFT overlay is limited to the number of nodes. In the SPIDER overlay, the maximum number of nodes is theoretically unlimited because the number of rings can grow without limitation. The SPIDER Peer-to-Peer overlay is optimal in terms of auto-adaptability and fault tolerance in comparison with the AFT and HoneyComb overlays.

2.5 Challenges in SPIDER Peer-to-Peer Overlay

This section aims to present a brief description of the open research issues treated in this thesis. The main challenges of SPIDER Peer-to-Peer overlay systems management are:

- Auto-adaptability;
- Data dissemination;
- Fault tolerance;
- Data fusion for security enhancement;
- Trust as security feature.

2.5.1 Auto-adaptability

Auto-adaptability of a Peer-to-Peer overlay is the central most important research topic of this thesis. The SPIDER Peer-to-Peer overlay is an auto-adaptive Peer-to-Peer structure. This feature is allowing the overlay to modify its structure concerning the number of nodes to maintain its functionality in a particular situation.

Analyzing the literature ([61] [21]), the construction of bio-inspired Peer-to-Peer overlays is inferior to human-centric models ([53], [98], [122]).

In [3], the authors proposed a way of organizing devices such manner that they adapt and change to enable fast data sharing.

The authors of the Forward Feedback Protocol [51] proposed a novel method for solving errors. In their vision, in case of error occurrence at the same time as the routing message, signals are sent through the routing paths in. When deciding the next hop, is taken into consideration the signals that are processed by peers. Concerning performance, the protocol offers delivery success rate with 30% better in comparison with other solutions.

2.5.2 Data dissemination

A good overview of the video streaming evolution in the last two decades is described in paper [80]. This paper talks about three types of video streaming:

client-server video streaming, Peer-to-Peer video streaming and HTTP video streaming over the Cloud.

In [39], the authors present a Peer-to-Peer based application designed for live video streaming encoded at different bit rates. The proposed solution was evaluated regarding performance by a prototype implementation in JAVA an ICN (Information Centric Networking) architecture.

The authors of [141] presented an optimized solution for high-quality adaptive video streaming with low costs concerning replication and computational resources.

The UrbanCPS described in [155] is an integration cyber-physical system based on a sizable multiple sources infrastructure in Shenzhen, China, that has over 42 thousand vehicles, 10 million residents, and 16 million smart-cards. The authors of this project proposed Speedometer, a real-life application for inference of traffic speeds in the geographical area. The experimental evaluation reveals the fact that Speedometer increases the inference accuracy by 21%.

In paper [87] the authors proposed an ICN architecture based system for live video streaming called NetInf. The proposed model was tested on multiple days during major sports events. The most significant advantage of this system is the fact that it streams video content with low communication overhead.

An exciting article regarding agriculture monitoring is [26]. The authors of this paper proposed a geo-spatial web service interoperability of heterogeneous sensors in precision agriculture (PA) monitoring. The system was evaluated with a set of 2 real experiments conducted in Baxie field, Wuhan City. The investigations reveal the fact that the proposed service was able to acquire data from the sensors and also to process and disseminate it via the World Wide Web. Also, the authors presented a critical analysis of their solution in comparison with other PA monitoring systems.

The authors of [54], proposed a framework for QoE fairness in a shared network environment. The solution was evaluated in a home networking scenario and showed an optimization of the video streaming QoE across heterogeneous devices located in the network.

Despite the scientific contributions of the authors of the papers mentioned above, we propose an original method of data dissemination in the SPIDER overlay that shared data only between the neighbors of each node (maximum 4 nodes).

This method reduces throughput and increases the performance of the SPIDER Peer-to-Peer overlay based systems.

2.5.3 Fault tolerance

Smart cities and Cyber-physical infrastructures have become beautiful subjects of research in the past years. Trust management in Bio-inspired Peer-to-Peer systems has many challenges especially coming from the fact that there is no central authority to trust.

The authors of [99] have presented an emergency situation management system based on a hybrid cloud architecture that manages storage and computing resources for command and control activities. The efficiency of the proposed approach is based on the fact that the system is aggregating information from motion sensors correlated with the signal strength from landmark nodes.

In paper [30], the authors present a model for effective fault reduction of devices in an IoT environment. One particular example of this modes take advantage of the door sensor and the heat sensor to replace the functionality of the video camera.

Paper [143] presents TRANSIT an exciting approach for the transition between different types of mechanisms in live video streaming with respect to performance in a highly dynamic environment from fluctuation point of view. The proposed approach was evaluated through trace-based workloads. The experimental results showed the fact that the proposed solution offers good resilience.

Another exciting approach is presented in paper [86]. The authors of this paper evaluated the performances of SPS (Swarm-based Peer-to-Peer Streaming) for video streaming and proposed OLIVES; an ISP-friendly Peer-to-Peer live video streaming solution. The resulted values for the evaluation of the OLIVES mechanism were obtained through simulation, and they demonstrate the fact that the proposed solution can deliver high-quality video streams over different real scenarios. The limitations taken into consideration in this paper were the excess connection bandwidth and the limited connection bandwidth.

The authors of paper [151] present an excellent survey for smart greed communications. In this paper are described several requirements such as QoS expressed regarding latency, bandwidth, interoperability, scalability, security

and standardization

The authors of paper [4], presents a scheduling algorithm for video broadcasting with maximum stream rate. The hypothesis in their paper is based on the fact that each peer in the system interacts only with a few peers in their neighborhood.

In paper [18], the authors tackles the energy consumption problem of in Peer-to-Peer networks designed for file sharing, content streaming and epidemics. This survey classifies Peer-to-Peer networks in 6 categories: proxy-based, sleep/wakeup, resource allocation, peer selection, message reduction and structure optimization.

In the smart home field has emerged in the latter years driven both by the technological advances in the sensor field and by the new Peer-to-Peer networks approaches according to [148]. Software-defined software for smart-homes is now an emerging field in which the implementation aspects of the intelligent house are no longer the only issue. The users' habits customization are also taken into consideration of the control of the network.

Even though, the contributions of the authors of the papers mentioned above, has a high scientific value we propose an original framework for the auto-recovery of the SPIDER based system from local and global faults. We also take into consideration the case of temporary faults caused by minor defects or by power supply or signal loss hazards.

2.5.4 Data fusion for security enhancement

Data fusion in Peer-to-Peer sensor networks represents a significant research challenge for many computer science researchers worldwide as mentioned in [15].

The continuously increasing amount of data coming from heterogeneous sources in the field of Smart Home has created several challenges for researchers especially for the management of such data. The authors of [128] have investigated ontology-based on data semantic management application and proposed a fusion model data semantic. Also, they have developed a new method for data decomposition based on a relational database. The efficiency of the proposed model was evaluated through experiments for user behavioral reasoning.

The authors of [5] has debunked some myths regarding the usage of the Peer-to-

Peer networks for sensor networks. Therefore, they have proposed a protocol for integration over the Chord Overlay called Tiered Chord (TChord).

The primary focus on paper [124] is upon the Smart City concept, where billions of interconnected heterogeneous sources provide data that must be processed to be used. One of the most significant problems of this concept is the fact that due to the heterogeneity of the devices, the data have several imperfections such as imprecision, ambiguity or uncertainty. The authors of this paper have handled imperfection in the process of information retrieval and created an evidence-based theory database for increasing the efficiency of Smart Cities.

The authors of paper [31] have proposed the iSapiens platform, a Smart City IoT-based application platform for Cyber-physical systems [91, 60, 69]. The main features of the iSapines platform are the implementation of the edge computing paradigm concerning a distributed network of computational nodes placed in an urban environment. The proposed platform is addressed to the development of dynamic computational nodes and software agents to receive data regarding their geographical positions. In this paper, it is also presented a brief methodology for implementation of real Smart City applications. The experimental results conducted in this research were realized based on actual data from the city of Cosenza (Italy).

The authors of paper [149] presented a generalized view of data fusion in multi-sources systems. A voting-based data fusion method for conflict adjudication was proposed in this paper.

In paper [17] is presented a very comprehensive state of the art survey of both of the Internet of Things and Cloud Computing paradigms and their interconnection called CloutIoT. The authors have shown their vision regarding the integration of the Internet of Things with Cloud Computing in the ages of massively interconnected devices. They have analyzed the complementarity of Cloud Computing and Internet of Things to provide an accurate and updated image of CloudIoT applications in the specialized literature. Another exciting aspect covered by this survey is the availability of both open-source or proprietary platforms for CloudIoT concept.

In paper [43], the authors have described several data fusion architectures such: hierarchical data fusion, distributed data fusion and decentralized data fusion.

The authors of paper [147] proposed a Transparent Data as a Service framework

that combines Transparent Computing and Representational State Transfer (REST) Web Services based on Linked Data. The feasibility of the proposed solution was evaluated by implementing a prototype.

Services that provide location information, also known as location-based services (LBS), have become very popular and have gained considerable interest from researchers. The authors of paper [100] have focused their attention on the preservation of privacy in continuous LBSs. Thus, they have proposed a privacy preservation method based on a collaborative trajectory for constant queries. The main idea used was the obfuscation paradigm, which is used to mask the actual user trajectory by issuing fake queries. Also, the proposed scheme was evaluated regarding communication throughput and processing time.

In paper [58] is proposed a hierarchical Peer-to-Peer approach for achieving data fusion in distributed systems for increased security awareness.

A well-written survey of wireless sensor networks (WSN), is paper [138]. Also in this article, the authors have presented the advantages and disadvantages of cluster-based schema and chain based schema for data fusion.

In paper [66] an indexing scheme for wild-card search queries in Peer-to-Peer systems based on characters. The proposed approach supports fast updates and continuously changing information for the lowest increase in signaling connected peer device. Also, the authors of this paper have proven that a chord based hash table is more scalable regarding size while supporting multidimensional queries.

The authors of paper [114] have presented a state of the art survey for data fusion for context awareness in Intelligent Traffic Systems (ITS).

The authors of paper [13] have proposed a state of the art algorithm called ASBP for multi-phase adaptive sensing that provides better data quality with fewer nodes in the system. The experimental results show the fact that while keeping the data quality to a satisfactory level the ASBP algorithm can preserve 80% more energy than without using this algorithm.

Due to the increased number of live multimedia streaming applications, the authors of paper [62] has presented a novel Peer-to-Peer architecture based on a hierarchical ring tree called Hierarchical Ring Tree (HRT) for live video streaming. The proposed method uses both three-based and ring-based structures to provide a scalable and robust architecture for Application-layer Multi-cast (ALM) applications. The validation of the model was realized

through experiments, which showed the fact that this model is suitable for high-churn Peer-to-Peer networks with a small delay. Also, the proposed model was compared against the ZIGZAG approach ([132]), and the authors concluded that their plan has a lower overhead.

An exciting approach in improving Smart Cities security is presented in paper [52]. The novelty of the proposed method consists in using Computer Vision by taking the advantages of IP cameras.

In paper [72] the authors have proposed a secure end-to-end service concerning privacy concerns in smart cities. Also, the authors have presented a state of the art use case of their privacy-aware framework called SSServProv. The offered services were evaluated by using the Scyther ([34]), an automated security verification tool.

The authors of paper [133] have presented an exciting design for Smart City communication based on the nervous system model for resilient and ubiquitous systems called Smart Gateways. The proposed approach was evaluated through simulation for different technologies.

Smart-home IoT applications have become a very challenging research topic in the last years. Therefore, in paper [29], the authors proposed a novel lightweight authorization service based on mobile cloud authorization

Another aspect of security in the Smart-City concept is the use of cryptography. The authors of paper [113] have presented the Midgar framework, a platform that evaluates several combinations of photographically algorithms for the Internet of Things such as RSA, AES and SHA3.

A very well written definition of multi-sensor data fusion, is given in the book [82]. Also in this book is presented a brief history of multi-sensor applications.

In comparison with the probabilistic data fusion, where data uncertainty is based on the probability of distribution Bayes function ([56]), the SPIDER based fusion methods are not based on probabilities. Our approach, is based on the structure of the SPIDER overlay, therefore, we propose two algorithms for fusion one based on chains and other based on the rings of the overlay. Because the fusion methods respect the SPIDER overlay structure, this approach is fault tolerant.

2.5.5 Trust as a security feature

A good overview regarding reputation on Peer-to-Peer systems, has been created in [90]m where is presented a useful taxonomy of Peer-to-Peer reputation models composed by gathering of information, ranking the peers and rewarding the peers. This paper motivates the adaptive behavior of our proposed trust model. Having a right trust metric based on Social-Networks is a significant challenge because of the reputation ranking depends on the peers' social position. Therefore, the most dangerous peers are the front ones.

In [130], the authors proposed a better approach entitled Poisonwater for Social-Network based trust metrics that is resistant to front peers attack than Eigentrust and Powertrust. In comparison, the Poisonwater can mitigate front peers attack by 20%. Another interesting approach regarding building good trust relationships in decentralized Peer-to-Peer systems is presented in [140], where the authors proposed a new model of trust based on reputation and risk evaluation. This model is suitable for defending against simple malicious attacks, collusive and strategic attacks. The applicability of reputation in the case of web services was studied in [118] and [1] where a reputation based selection mechanism is proposed

Reputation management in Peer-to-Peer networks based on Distributed Hash Tables (DHT) is the main idea presented by [47]. The authors proposed an algorithm that permits the peers in the system to get individual ranking values for each node based on knowledge exchange.

Due to the high increase of mobile devices in the market, an essential category of decentralized systems can be considered the mobile Peer-to-Peer networks. Creating trust and manage reputation in this kind of networks is a significant challenge. Therefore in [106] is realized a brief overview of four trust management schemes for Mobile Peer-to-Peer systems and further proposed a new reputation trust management schema entitled M-trust. Concerning the hybrid Peer-to-Peer networks, in [131] is suggested a Super-peer trust model called SuperTrust that determines the peers to cooperate based on the common interests. Therefore, peers that share fewer things in common with others are likely to interact with others more frequently. This model can handle with success several attacks like pure malicious, denigrating peers, collusive peers and strategic peers attacks.

Trust is a very complex measuring value because it can be computed local or global based on the past interactions of the node or based on a single

transaction. Also, it is dynamic and asymmetric because the trusted value for a node is computed based on the previous interaction of that node with other peers. The dynamic behavior of the trust value consists in the fact that the trust value for every node can rapidly decrease for instance if the node starts to share the malicious date in the system. In this case, a solution to establish a proper bound is to used entropy computed with interaction probabilities [129]. Another property for trust is transitivity. If we consider three nodes in the system X, Y and Z, if X trusts Y and Y trusts Z it means that also X trusts Z. The trust value for this relations is computed as presented in [42]. This computational model involves the interactions of a node with another node in a specific context. Reputation is defined as perception formed through past actions about the intentions and norms of a peer. And trust as a subjective expectation that a peer has towards another, based on the previous behavior and history. The flow of the system is the following: an exchange takes place based on the trust value. After it is completed, the reputation is updated, and the trust value recomputed and so on. The computation process for reputation may be affected by false recommendations. As a solution to this, in [64] is presented a new trust model that extends the Dempster-Shafer theory [119], that improves the filtering of false recommendations and the dynamic adaptive to strategic behaviors.

We identify a few limitations of the Peer-to-Peer reputation systems mentioned above. Therefore, EigenTrust algorithm must take into account the entire system history for each peer in the network. In comparison with SPIDER Peer-to-Peer overlay that take into the consideration only the history of the neighbor nodes. In terms of fault tolerance, SPIDER Peer-to-Peer overlay offers better features in comparison with the PowerTrust algorithm because the PowerTrust method keeps a centralized authority witch is a single point of failure.

Even though, the authors of the papers mentioned above brought high value to the scientific communities, we propose an innovative fully decentralized and adaptive trust management method based on the structure of the SPIDER Peer-to-Peer overlay.

2.6 Discussions and Conclusions

In this chapter, we made a comparative analysis of various Peer-to-Peer overlays. We described the most important aspects of the discussed Peer-to-Peer overlays and reviewed the essential features and metrics in comparison with the SPIDER Peer-to-Peer overlay.

We have also presented in this chapter a brief description of the most important terms used when talking about Peer-to-Peer systems.

The most important contribution of this chapter is the identification of the open issues of Peer-to-Peer overlays, issues analyses and researched deeply in this thesis: auto-adaptability, fault-tolerance, data-fusion and trust.

Chapter 3

SPIDER: Auto-Adaptive Peer-to-Peer Overlay

In this chapter we aim to answerer RQ1 research question announced in Section 1.1 (Problem Statement and Thesis Objectives) by proposing an auto-adaptive Peer-to-Peer overlay based on the spider web architecture. We describe in a formal manner the design of the overlay, its structure, and its fundamental operations. Also, in this chapter, we present and evaluate the most common operations available in the SPIDER overlay and compare this solution with other similar overlays.

Despite the valuable findings described in Section 2.5.2 (Data dissemination), we propose a decentralized Peer-to-Peer overlay that is one step forward improving data dissemination in heterogeneous systems by sharing data only between the neighbors of each node (maximum 4 nodes).

Therefore, in this chapter we offer the structure of the SPIDER Peer-to-Peer, a bio-inspired, self-adaptive overlay, as well as the most important operations available in SPIDER: naming, joining, leaving and routing (Section 3.1) (Concept of the SPIDER Peer-to-Peer Overlay). Also, in this chapter we present the experimental result of the overlay's evaluation (Section 3.2) (Performance Analysis of the SPIDER Peer-to-Peer Overlay) as well as our discussions and conclusions (Section 3.3) (Discussions and Conclusions).

3.1 Concept of the SPIDER Peer-to-Peer Overlay

According to CISCO white paper[1], the amount of Internet traffic composed by Video content, VoIP data and Web data will grow significantly by the year 2021 (Figure 3.1).

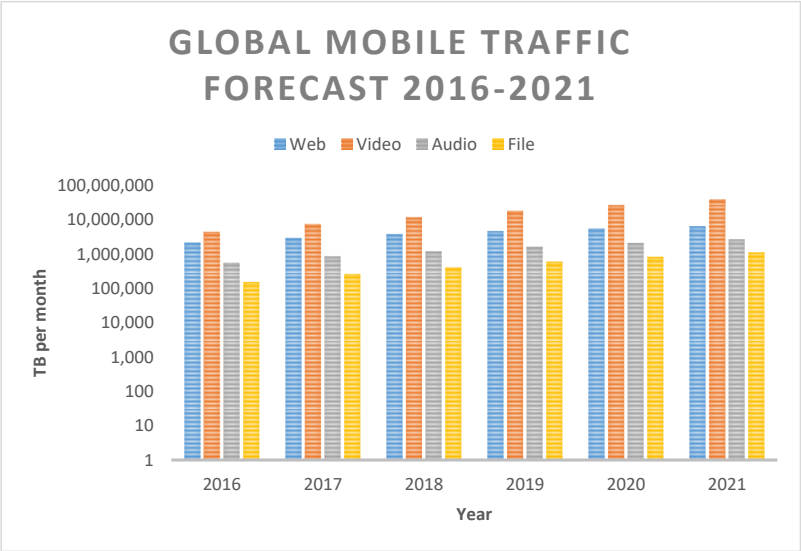


Figure 3.1: Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016–2021

Peer-to-peer systems have proven their significant contribution to data distribution over the Internet. Bio-inspired Peer-to-Peer overlays have become a challenging research topic in the computer science field since the beginning of this decade. From ant colonies to bee swarms or fish banks all these natural phenomenons have become an inspiring source for researchers. Data dissemination for crowd management is also a very actual research topic in the computer science community. Furthermore, we present SPIDER, a new Peer-to-Peer structured overlay based on a natural structure of a spider web and some possible uses cases for crowd management using data dissemination.

The construction of this Peer-to-Peer overlay is inspired from the natural phenomenon of a spider web creation. Therefore, the SPIDER overlay consists of a fixed number of chains and a dynamic number of rings. The chain

[1]https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html, Accessed: 07-03-2019

represents the line between the Super Peer and all the nodes above in a hierarchical manner. Rings are the circular line between nodes on the same hierarchical level. According to this approach, the management of the SPIDER overlay is a two dimension matrix read on columns, where each chain represent a row and each circular line represents a ring (Figure 3.2)

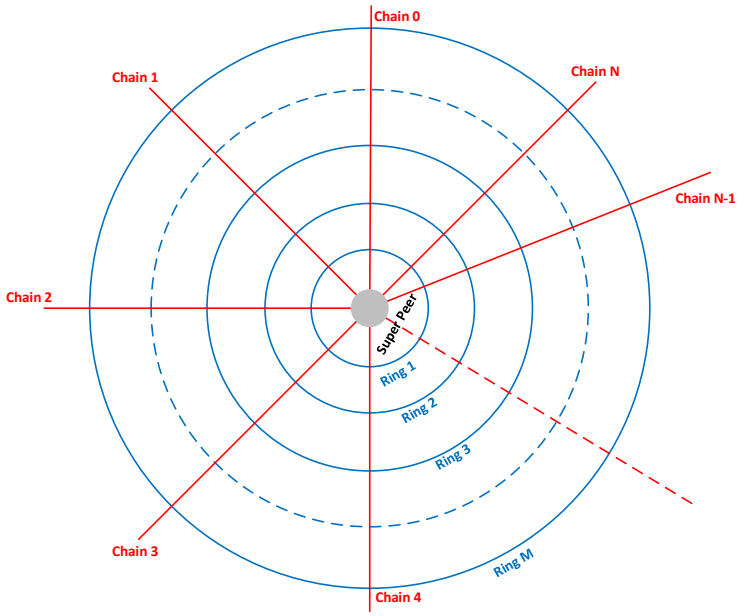


Figure 3.2: SPIDER Peer-to-Peer overlay overview.

The SPIDER overlay is formed around a central node called Super Peer. This node is situated in the center of the overlay. Despite its position, the Super Peer is not considered a single point of failure because its primary role is to facilitate the construction of the overlay and after the first node joins the system is not anymore needed. If its presence is necessary in case of failure it can be any time easily replaced. The architecture of the SPIDER overlay consists of several nodes placed in a specific order based on their time of joining the system.

Each node in the SPIDER overlay is uniquely identified by its coordinates, the chain and the ring. Also, in this overlay, each node can have a maximum of four neighbors as sown in Figure 3.3. Furthermore, we present the equations of the neighbors of a fixed node [c,r], where c represents the chain, r represents the ring and n is the number of nodes on the chain (Equation 3.1).

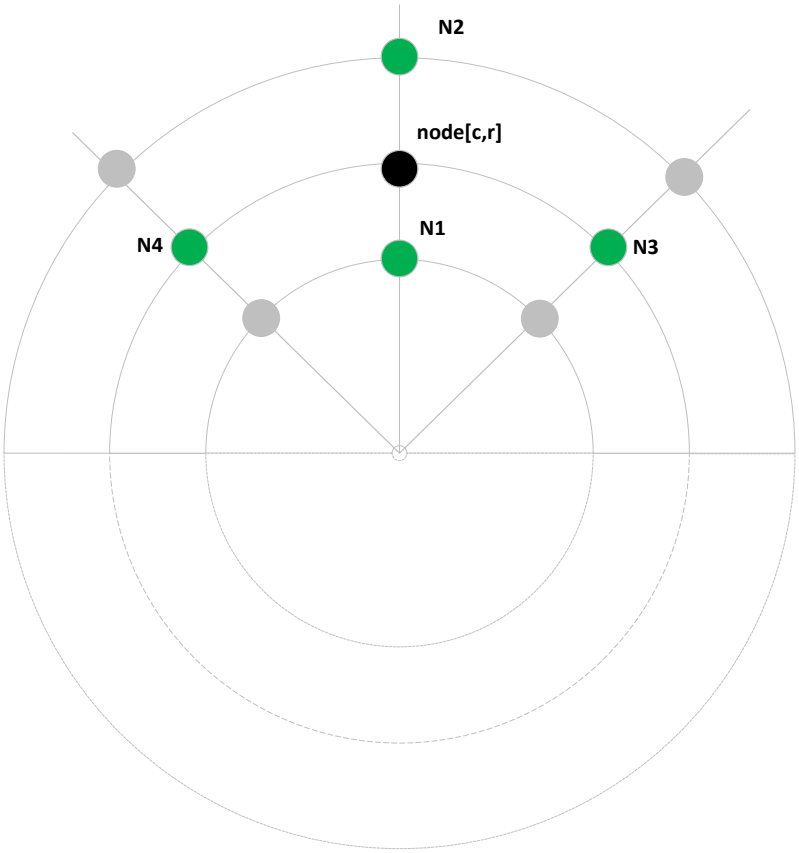


Figure 3.3: SPIDER Peer-to-Peer overlay node position.

$$\begin{aligned} N1 : node[(c \bmod n), r-1] - below\ neighbour \\ N2 : node[c, r+1] - upper\ neighbour \\ N3 : node[(c+1) \bmod n, r] - right\ neighbour \\ N3 : node[(c-1) \bmod n, r] - left\ neighbour \end{aligned} \tag{3.1}$$

Another important property of the SPIDER Peer-to-Peer overlay is the fact that each node in the overlay does not carry the same task, thus the SPIDER Peer-to-Peer overlay is asymmetric. The only tasks, that each node can perform are the basic operations available in the overlay: joining, naming, leaving and routing.

Furthermore, we present the essential operations available in the SPIDER overlay to create a whole picture of the SPIDER overlay. In the following subsection we will describe the joining, naming, routing and leaving operations.

3.1.1 Naming in SPIDER Peer-to-Peer Overlay

The Super Peer unique identifier is 0, and it belongs to all chains in the network, and there is no ring containing it. Then, for each node we consider the chain c, with $1 \leq c \leq n$, where n is the number of nodes on the chain, and the ring r, with $r \geq 1$. The node id is computed as the product of the number of the nodes on the chain and the below ring number added with the chain number shown in Equation 3.2.

$$n_i = node[c, r] = n * (r - 1) + c \tag{3.2}$$

The total number of nodes per chain is r, and the total number of nodes in a complete network is $N = r * n$. If the last ring is incomplete, then the total number of nodes in SPIDER Peer-to-Peer Overlay is given by Equation 3.3.

$$(r - 1) * n + 1 \leq N \leq r * n \tag{3.3}$$

The total number of links for a ring is equal with n. Then we have maxim n links between the current ring and the upper ring. So, the total number of links in the network follows Inequality 3.4, where r is the number of rings.

$$n * r + 1 \leq L \leq n * (r + 1) \tag{3.4}$$

For a node n_i, we can compute its location as shown in Equation 3.5.

$$\begin{aligned} chain(n_i) &= (n_i - 1)\%n + 1 \\ ring(n_i) &= \frac{n_i - chain(n_i)}{n} + 1 \end{aligned} \tag{3.5}$$

In order to achieve node monitoring and data dissemination, it is vital to calculate the minimum number of hops between two nodes. Fist, we introduce a

function that computes the number of hops between two nodes from the same ring (Algorithm 1). The number of hops on the same ring is given by the absolute value of the difference between the chains of the two nodes. In case $\frac{NumberOfRings}{2}$ is smaller than the absolute difference between the two nodes, first it is tested if the absolute difference is higher than the $\frac{NumberOfChains}{2}$ and the number of hops is computed as the difference between $NumberOfChains$ and the absolute difference of the chains of the two nodes.

Algorithm 1 SPIDER - Number of hops between two nodes on the same ring.

1: **if** $(|ring(n_i) - ring(n_j)| \leq \left[\frac{NumberOfRings}{2}\right])$ **then**
2: $\quad ChainDIF(n_i, n_j) = |chain(n_i) - chain(n_j)|;$
3: **else**
4: $\quad$ **if** $(|chain(n_i) - chain(n_j)| > \left[\frac{NumberOfChains}{2}\right])$ **then**
5: $\quad\quad ChainDIF(n_i, n_j) = NumberOfChains - |chain(n_i) - chain(n_j)|;$
6: $\quad$ **end if**
7: **end if**

Second, we compute the difference between rings containing n_i and n_j as the difference between the maximum between the rings of the two nodes and the minimum between the two nodes as shown in Equation 3.6.

$$RingDIF(n_i, n_j) = \max\{ring(n_i), ring(n_j)\} - \min\{ring(n_i), ring(n_j)\} \quad (3.6)$$

And, after some calculations we obtain the ring difference $RingDIF(n_i, n_j)$ as the absolute value o the difference between the rings of the nodes as shown in Equation 3.7.

$$RingDIF(n_i, n_j) = |ring(n_i) - ring(n_j)| \quad (3.7)$$

Now, we obtain the total number of hops between node n_i and node n_j as the sum of the chain difference ($ChainDIF$) and ring difference ($RingDIF$) as given in Equation 3.8, where $ChainDIF$ is obtained through Algorithm 1 and $RingDIF$

is computed through Equation 3.7.

$$hops(n_i, n_j) = ChainDIF(n_i, n_j) + RingDIF(n_i, n_j) \tag{3.8}$$

3.1.2 Joining in SPIDER Peer-to-Peer Overlay

Joining (Figure 3.4) represents the process of becoming a member of the overlay for a new node. Thus, joining in the SPIDER Peer-to-Peer overlay when knowing an existing node in the system. When a current node receives a joying request from an outsider node, it cheeks its neighbor list to determine if it has any free positions. If there is available at least one free position, the new node will join the overlay in one of those locations, and the existing nodes in the overlay will update their neighbors list towards the node on the last ring of the overlay. If the last node on the rings has no available positions, a new join request is sent by the known node to its upper neighbor and a new ring is constructed, and the new node joins on that new ring at the same chain as the known node in the overlay as presented in Algorithm 2. Also, the neighbors lists are updated.

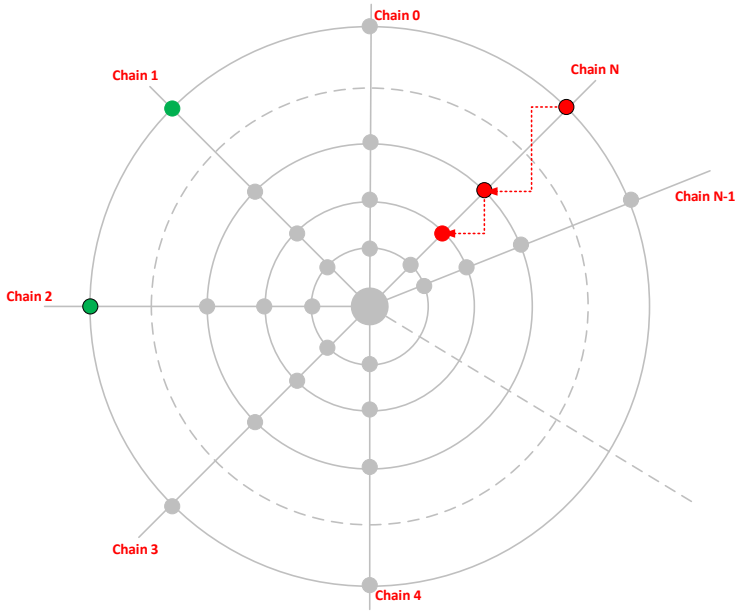


Figure 3.4: SPIDER Peer-to-Peer overlay node joining.

Algorithm 2 SPIDER Joining request algorithm

1: $S[nc, nr]$ //SPIDER overlay with nc chains and nr rings
2: $n_{known} \leftarrow node[c, r]$ // known node in the overlay located on the c chain and r ring
3: $SUCCES \leftarrow false$ //boolean flag for successful identification of a free position in the overlay;
4: $JOINREQUEST(n_{known})$
5: $node \leftarrow n_{known}$ // variable for node iteration through SPIDER overlay
6: **while** SUCCES $\neq$ TRUE $||$ $r = nr$ **do**
7: $RES \leftarrow CHECK(node)$ // check the neighbors of the *node* for free position
8: **if** $RES = SUCCES$ **then**
9: JOIN() //New node joins the SPIDER overlay on c chain and the ring number of *node* ring;
10: $SUCCES \leftarrow true$
11: **else**
12: $node \leftarrow node[c, r++]$ //the *node* variable becomes the upper node
13: **end if**
14: **end while**
15: **if** $r = nr$ **then**
16: BUILD_RING() //Build a new ring in the overlay
17: JOIN() //New node joins the SPIDER overlay on the c chain and $nr + 1$ ring
18: **end if**

The joining algorithm in the SPIDER overlay is similar to the one in the AFT Peer-to-Peer overlay because booth overlays share a structure with chains and rings and maximum 4 neighbors for each node. Therefore, the number of messages for joining in booth cases is the same.

3.1.3 Leaving in SPIDER Peer-to-Peer Overlay

Another important operation in the SPIDER overlay is leaving. Taking into consideration that in Peer-to-Peer systems peers might join and go any time without any announcement, it is crucial that when a peer leaves the system, the overlay not to fall apart and to auto-adapt to its original state with a cost of time

minimal. Therefore, the peer that is located on the same chain and on the next upper ring replaces the exit node.

We consider this approach to be the most opportune method because it is unlikely to break an existing ring. Taking into consideration the mechanism presented above, the Super Peer can be easily replaced without affecting the structure of the overlay.

3.1.4 Routing in SPIDER Peer-to-Peer Overlay

The SPIDER overlay can be used for two scenarios of data dissemination: regular and hierarchical dissemination. Concerning the proper spread, a node queries its neighbors for an specific resource. If the requested data is available, it is sent immediately to the requester, if not, each node creates a resource query for its neighbors without the one that started the process. Regarding the hierarchical data dissemination approach, we take into consideration the following assumption. The nodes on a chain share common interests or are part of the same department. Resources are being delivered only on the same chain to the upper neighbor or the lower neighbor. Also if a resource is required from another chain, this can be realized only on the same level.

The complexity of the routing algorithm in the SPIDER Peer-to-Peer overlay is square root witch means that the algorithm requires $\sqrt{N}$ evaluations, where N is the product of the number oc chains c and the number of rings r as shown in Equation 3.9.

$$Complexity = O(\sqrt{c \times r}) \tag{3.9}$$

3.2 Performance Analysis of the SPIDER Peer-to-Peer Overlay

The performance analysis of the SPIDER Peer-to-Peer overlay was realized using the real implementation presented in Section 1.2. In order to evaluate the SPIDER overlay with high number of peers (10000 nodes) we conduct several experimental evaluations through simulation. This method of evaluation was

chosen because the high number of peers in the SPIDER Peer-to-Peer overlay cannot be achieved in a real-life experiment. Even though the assessment of the SPIDER overlay was conducted through simulation for high number of nodes, we also performed a real evaluation of the SPIDER Peer-to-Peer overlay based on real-life datasets.

First, we focus on the evolution of the number of hops for the lookup operation in the SPIDER overlay. Second, we focus on the number of links between nodes in the SPIDER Peer-to-Peer overlay. Third, we asses the number of the links depending on the number of rings in the SPIDER overlay.

In Figure 3.5, we represent the evolution of the number of hops with the number of peers in the system, in case of the lookup algorithm. As depicted, the evolution trend is linear. The number of hops for 6000 peers in the system is 1000, which is significantly low than the number of hops in a classical system. As it can be seen, the SPIDER Peer-to-Peer overlay scales perfectly for structures of thousands of nodes.

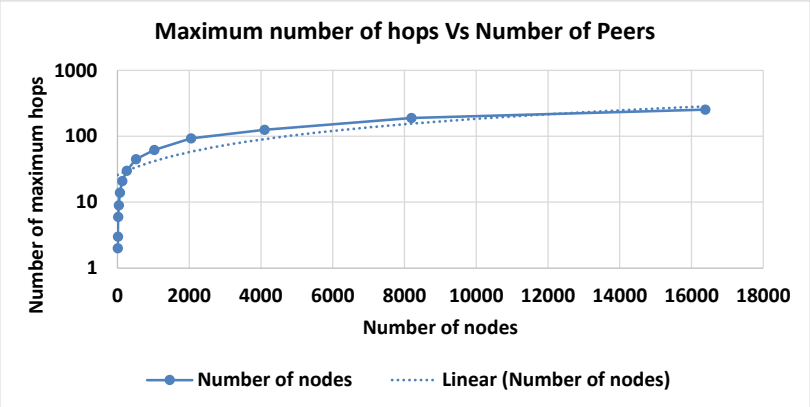
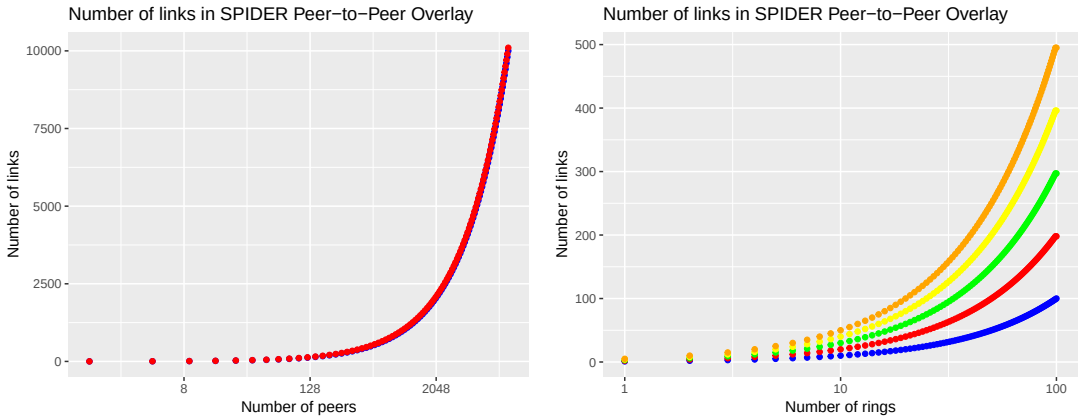


Figure 3.5: SPIDER Peer-to-Peer overlay number of max hops Vs number of peers.

In Figure 3.6, we present the evolution of the minimum number of links and the maximum number of connections available in the SPIDER Peer-to-Peer overlay. We can see the evolution of the number of links and the maximum number of connections in a SPIDER Peer-to-Peer overlay is linear.

In Figure 3.6b, we show the experimental results for the construction of the SPIDER overlay. This graph indicates the evolution of the maximum number of peers in the SPIDER overlay with the number of chains and rings. As it can be



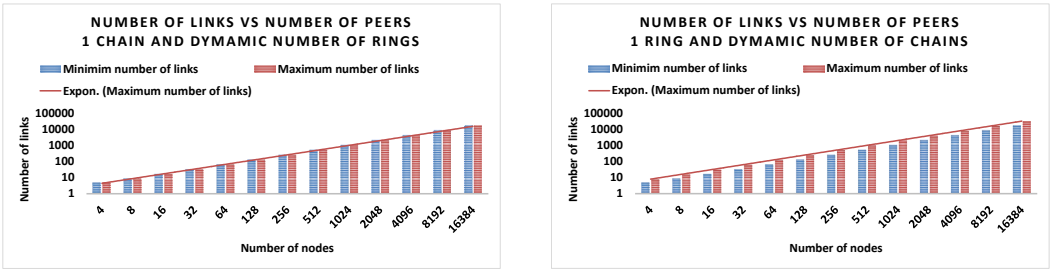
(a) (b)

Figure 3.6: SPIDER Peer-to-Peer overlay evolution of the minimum and maximum number of links (a) SPIDER Peer-to-Peer overlay number of max hops Vs number of peers (b).

seen, an overlay with 100 rings can accumulate 100, 200, 300, 400 or 500 peers depending on the number of the chains, therefore, the evolution is linear.

In order to evaluate the performance of the SPIDER Peer-to-Peer overlay, we also conducted several stress tests. Therefore, we analyze the performance of the SPIDER overlay with a minimum number of chains and rings. The obtained numbers are presented in Figure 3.7 and Figure 3.8. In both cases, the minimum number of links in the SPIDER overlay is the same, but the maximum number of links is 50% higher in case of a SPIDER overlay with 1 ring and a variable number of chains (Figure 3.7b) than a SPIDER overlay with 1 chain and a variable number of rings (Figure 3.7a). Also, in both cases the evolution of the maximum number of links is linear. In terms of the number of hops, the structure of the overlay does not affect significantly the values obtained. The obtained results show the fact that is better to build real life SPIDER-based Peer-to-Peer applications organized on many rings rather that on many chains.

Furthermore, we performed an evaluation of the number of messages needed to construct the SPIDER overlay in comparison with the construction process of the AFT ([101]) and HoneyComb ([102]) overlays. As mentioned in section 3.1.2 in SPIDER overlay the joining process takes the same number of messages as AFT overlay. Figure 3.9 presents the number of messages used for an overlay construction with hundreds of nodes. As shown for an overlay with small number of nodes (less than 150) the construction of the SPIDER and AFT overlays is the



(a) (b)

Figure 3.7: SPIDER Peer-to-Peer overlay stress test. Evolution of the number of links with the number of peers structured as 1 chain and variable number of rings (a) and as 1 ring and variable number of chains (b).

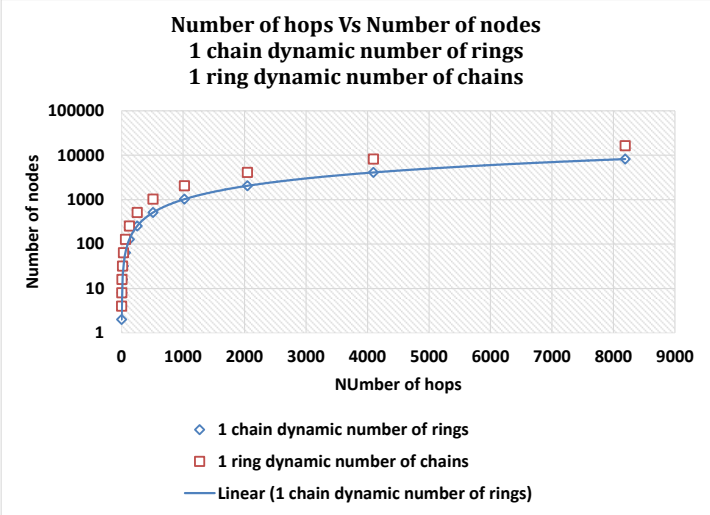


Figure 3.8: SPIDER Peer-to-Peer number of hops Vs number of nodes in case of 1 chain many rings and 1 ring many chains SPIDER structures.

most efficient in terms of number of messages.

3.3 Discussions and Conclusions

To validate the properties of this new Peer-to-Peer overlay in the field of data dissemination, we make a critical comparison with other Peer-to-Peer overlays such as: HoneyComb, Chord and SelfChord as shown in Table 3.1.

The comparison we make focuses on several properties and metrics wildly used in

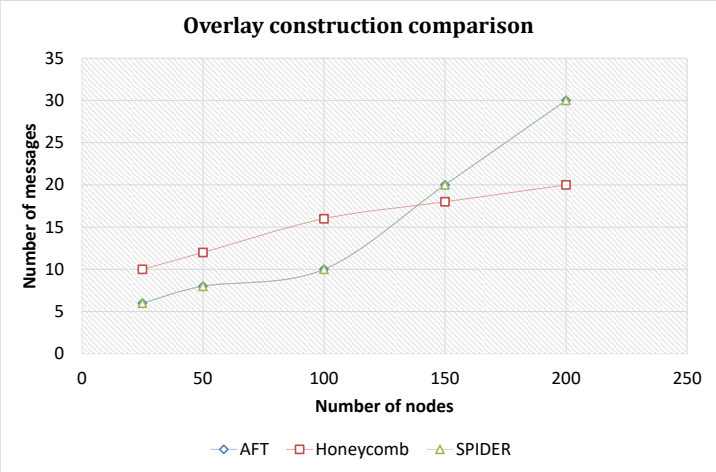


Figure 3.9: Overlay construction comparison of SPIDER overlay with AFT and HoneyComb overlays.

this thesis a key characteristics of Peer-to-Peer overlays: redundancy, dynamicity, security, lookup algorithm complexity an query types.

Table 3.1: Peer-to-Peer overlays critical analysis.

Overlay name	Redundancy	Dynamicity	Security	Lookup	Queries
SPIDER	No	Yes	Yes	$O(\sqrt{N})$	Any type of queries
HoneyComb	No	Yes	Yes	$O(\sqrt{N})$	Any type of queries
Chord	Yes	No	No	$O(\log N)$	Any type of queries
Self-Chord	No	No	No	$O(\log N)$	Standard queries

Therefore, it was taken into consideration two auto-adaptive overlays and one structured overlay. For this critical analysis, we used one of the most mature and overlay Chord, a bio-inspired view of the Chord ([125]) called Self-Chord ([49]). Another overlay taken into consideration in this comparison is a model developed proposed by the research group of UPB and presented in paper [102]. The honeycomb overlay uses the properties of hexagonal structures of a honeycomb.

There were taken into consideration five metrics: redundancy, dynamicity, Security, Look-up and type of queries. The redundancy property rearrests the ability of an overlay store duplicates of the same piece of information to assure the availability of data 24/7. The dynamicity property of a system in the capability to self-adapt and reconstruct itself in case the system becomes incomplete. This property deals with the fact that in Peer-to-Peer systems nodes can leave the system anytime without any announcement. The ability of

a system to use secure communication channels between peers is called security and is also considered in this study. The lookup property represents the number of steps needed for a piece of resource to be discovered in the system. The queries type property represents the available type of messages in a SPIDER Peer-to-Peer system.

Concerning redundancy, none of the overlays taken into consideration satisfies this property. Even though Chord is the most mature overlay presented in this analysis, it meets the dynamicity property neither the security property. The lookups in Chord and Self-chord have a logarithmic time complexity while for SPIDER and Honeycomb the lookup algorithms have a square root time complexity given by $O(\sqrt{N})$ steps, where N represents the total number of nodes in the system. Finally, Self-Chord, SPIDER and Honeycomb can handle any queries while Chord supports only standard queries.

In order to validate the performance of the construction of the SPIDER Peer-to-Peer overlay, we demonstrate that for small overlays (less than 150 nodes), the number of messages needed for the overlay construction is significantly lower than the number of messages needed for the construction of the HoneyComb overlay.

Chapter 4

Robustness of SPIDER Peer-to-Peer Overlay

In this chapter we aim to answerer RQ2 research question, announced in Section 1.1 (Problem Statement and Thesis Objectives) by proposing a novel set of algorithms for flaw recovery in the SPIDER overlay with respect to the structure of the SPIDER overlay. In contrast with the valuable existing solutions described in Section 2.5.3 (Fault tolerance), we present a set of algorithms for fault recovery in case of local failures (minor flaws that does not affect the structure of the overlay) and global failures (major flaws that affect the structure of the overlay). Also, in this chapter, we evaluate these methods by using the experimental methodology described in Section 1.2 (Research Methodology).

Therefore, this chapter presents a fault-tolerance framework for the SPIDER Peer-to-Peer (Section 4.1) (Fault Tolerance in SPIDER Peer-to-Peer Overlay) that includes three algorithms for local flaws and three algorithms for global flaws. Also, in this chapter we present the experimental result of the evaluation of the fault-tolerance algorithms (Section 4.2) (Experimental Evaluation of the Fault Tolerance algorithms) as well as our discussions and conclusions (Section 4.3) (Discussions and Conclusions).

4.1 Fault Tolerance in SPIDER Peer-to-Peer Overlay

In this section we analyze several types of flaws in the SPIDER Overlay and the impact of the reorganization of the logical structure of the system to regain functionality.

For a better understanding of the proposed algorithms for fault tolerance we use the following notations:

- SPIDER overlay with N nodes organized in nr rings and nc chains : $S[nr, nc]$;
- Number of peers in the SPIDER Overlay : n;
- A node in the SPIDER Overlay named by its coordinates in the overlay (chain and ring) : $*[c, r]$;
- Node N neighbor table : $NTable(N)$;
- Upper neighbor of N : $NTable(N).UP$;
- Lower neighbor of N : $NTable(N).DOWN$;
- Left neighbor of N : $NTable(N).LEFT$;
- Right neighbor of N : $NTable(N).RIGHT$;
- Free position in the SPIDER Overlay : $free$;
- Node in the SPIDER overlay fall : $*[c, r].FALL$;
- Create node with empty neighbor table : $NTable(NewNod[c, r]).EMPTY$;
- Delete node form SPIDER Overlay : $*[c, r].DELETE$;
- Entire chain from SPIDER Overlay fall : $*[*, r].FALL$;
- Entire ring from SPIDER Overlay fall : $*[c, *].FALL$.

4.1.1 Local Flaws in SPIDER Peer-to-Peer Overlay

These type of flaws might appear in the SPIDER overlay more frequently because nodes are electronic devices such as sensors, cameras, smart-home devices etc. that are prone to malfunction. In case of mobile devices, another

fact that determines flaws is the loss of GSM/3G/4G signal or battery life failure.

To explain the local flaws in the SPIDER overlay, we create a N node full SPIDER overlay with nr rings and nc chains. The maximum number of nodes in a full SPIDER overlay is $N = nr * nc$, and the maximum number of nodes on a chain is $NoNodesChain = nr$, and the maximum number of nodes on a rings is $NoNodesRing = nc$.

One Node Randomly Fall

Joining and leaving from Peer-to-Peer systems are the most common operations. As mentioned above, when a node leaves the overlay can be considered a flaw when nodes fail due to battery live failure or loss of radio connectivity. In this case, nodes disappear without any announcement. Thus, when a random node from the overlay leaves the overlay, we consider that the overlay is facing the one random node flaw. In this case, the overlay structure is not affected, neither the functionality of the system. Even though, the overlay is still functional the neighbor nodes have to update their neighbors table to a vacant position.

We describe the behavior of the SPIDER overlay, in this case, in Algorithm 3. Each node X located un the top ($X[c, r+1]$), bottom ($X[c, r-1]$), left ($X[c-1, r]$) and right ($X[c+1, r]$) of the failed nodes updates their neighbor table to a free position.

The algorithm for one node randomly fall is given in Algorithm 3:

Algorithm 3 SPIDER - One node randomly fall.

1: Randomly Node $X[c, r].FALL$
2: $NTable(X[c, r+1]).DOWN = free$
3: $NTable(X[c, r-1]).UP = free$
4: $NTable(X[(c-1)\%nc, r]).UP = free$
5: $NTable(X[(c+1)\%nc, r).UP = free$

In Algorithm 3, we show that when a random node fall from the SPIDER overlay, the neighbors tables of the neighbor nodes of the fallen one (top, bottom, left and right neighbors) updated their positions related to the fallen node to free.

When a node in the SPIDER overlay fail, the overlay has to recover as fast as possible to maintain the functionality of the system. Therefore, the impact of the overlay in case a random node fall is low. The actions that must be done refer only to the neighbor nodes. These nodes have to update their neighbor tables in order to ignore the fallen one and to reestablish the connections with the remaining ones.

Because the structure of the overlay is not affected in terms of the number of chains and rings, the impact of this type of failure is minimal. The process takes only four updates of the neighbors tables of the four neighbors of the fallen node.

The impact, in this case, is minor because the structure of the SPIDER overlay is not affected. Only the neighbors of the fallen node update their tables. Therefore, we compute only one operation/node for four nodes in the overlay resulting in a total of 4 operations.

In case of one randomly node fall, the overlay does not need to change to reestablish its structure. Thus, there is no recovery method necessary for this type of failure.

Two Nodes Fall from the Same Chain

Based on the same hypothesis mentioned above, regarding the structure of the SPIDER overlay, another flaw that might affect the overlay is when two nodes from the same chain fall at the same time. Even though, the number of dropped nodes of the overlay is higher this type of flaw is still considered to be a local one because the number of chains and rings remains the same.

The mechanism for two nodes fall from the same chain is given in Algorithm 4.

Algorithm 4 SPIDER - Two nodes fall from same chain.

$X[c,r].FALL$ and $Y[c,r+1].FALL$
2: $NTable(Y[c,r+2]).DOWN = free$
$NTable(X[c,r-1]).UP = free$
4: $NTable(X[(c-1)\%nc,r]).RIGHT = free$
$NTable(Y[(c-1)\%nc,r+1]).RIGHT = free$
6: $NTable(X[(c+1)\%nc,r]).RIGHT = free$
$NTable(Y[(c+1)\%nc,r+1]).RIGHT = free$

In Algorithm 4, we show the fact that when 2 nodes fall from the same chain, the neighbor tables of the neighbor nodes of the fallen ones X updated their positions related to the fallen nodes to free.

The impact in this case of flaw is a bit higher, but does not affect the entire structure of the overlay. Thus, in this case, the number of computations that must be executed by the neighbors of the fallen node is 6, because there are 2 mo operations than the one node randomly fall caused by the fact that for the second fallen node exist another 2 more neighbors.

Although two nodes from the same chain fall at the same time, this flaw is considered to be a local one. The structure of the overlay is not compromised, and the operations we make, affect only the neighbor tables neighbor nodes. Therefore, the number of transactions, in this case, increases to 2 in the case of one randomly node failure.

In case of auto-recovery, in the two nodes fall from the same chain defect, we propose a method that reduces, in fact, this fault to a random node failure. Thus, the structure of the overlay is not affected, and there is no need for the overlay to auto-configure itself.

A particular case of this flaw is when two neighbor nodes from the same ring fall. In this case, we solve the fault in the same manner as two random nodes failure except for the number of operations. Therefore, we compute the number of operations as Equation 4.1, where we multiply the number of computations at algorithm one node randomly fail by 2 because here are 2 failures in this case and reduce it by 2 because 2 of the neighbors of the fallen nodes refers already the failed node.

$$noOfOperations = oneRandomNodeFailureOperations \times 2 - 2 \tag{4.1}$$

Two Nodes Randomly Fall

This type of flaw appears when two nodes in the SPIDER overlay fall at the same time for various reasons: power supply failure, physical malfunction etc. In this case, the flaw is considered to be local because it does not affect the structure of the overlay. This issue can be easily divided into two random node flaws.

When two randomly nodes fall in SPIDER overlay can be used Algorithm 3 for booth nodes resulting an eight operation process for the overlay.

Taking into consideration the fact that this issue we can split it in a more straightforward local flaw, the SPIDER overlay will not have to auto-recover.

A particular case of two randomly node failure is when two closed nodes from the same ring fail. In this case, we transform the flaw in the same manner mentioned above with a particular exception as shown in Equation, where we multiply the number of computations at algorithm one node randomly fail by 2 because here are 2 failures in this case and reduce it by 2 4.2.

$$noOfOperations = oneRandomNodeFailureOperations \times 2 - 2 \tag{4.2}$$

4.1.2 Global Flaws in SPIDER Peer-to-Peer Overlay

In this section, we present the global flaws that might appear in the SPIDER Peer-to-Peer overlay. These type of flaws affect the entire structure of the SPIDER overlay. The number of chains or rings is modified and the overlay must auto-rebuild to be functional again. We divide the global flaws into the following categories:

- entire chain fall;
- entire ring fall.

We take into consideration a network formed by$N$ nodes organized in nr rings and nc chains as shown in Figure 4.1. This section aims to present how the SPIDER overlay reorganizes itself when the nodes from one chain fall all at the same time or when all the nodes from a ring all fall at the same time. In both cases, the SPIDER Peer-to-Peer network reorganizes itself to be functional again.

Entire Chain Lost Scenario 1

Starting from the scenario overview presented with a n nodes network, the first critical disaster scenario implies the fault of one chain at the same time. For this scenario, we consider that all nodes $N[2, *]$, from Chain 2 disappear at once along

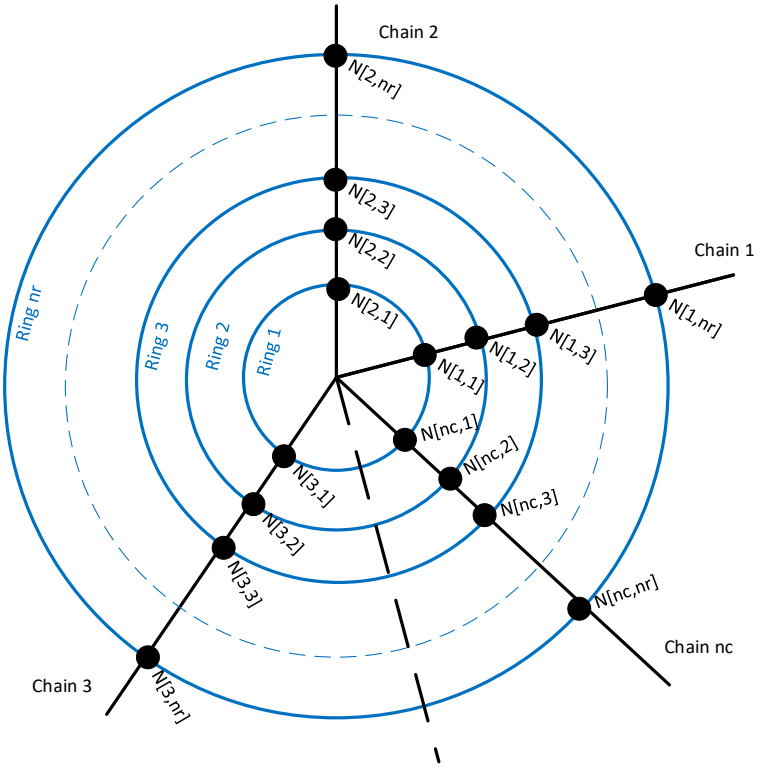


Figure 4.1: SPIDER Peer-to-Peer Overlay network organized in nc chains and nr rings.

with the chain. Therefore, the overlays have to reconfigure itself from a structure with N nodes and nc chains in one of $N - nr$ nodes and $nc - 1$ chains Figure 4.2.

In this scenario, the reorganization consists of updating the neighbor table of each node on the chain, on the left and the right of the lost chain. Thus, every node on Chain 1 updates its left neighbor to the node on the same Ring and Chain 3, and every node on Chain 3 updates its right neighbor to the node on the same Ring and Chain 1.

The impact in this type of flaw is high because about 66% of the nodes of the overlay are affected. Equation 4.3 gives the number of the operations for the reconfiguration of the SPIDER overlay.

$$noOfOperations = 2 \times noOfRings \tag{4.3}$$

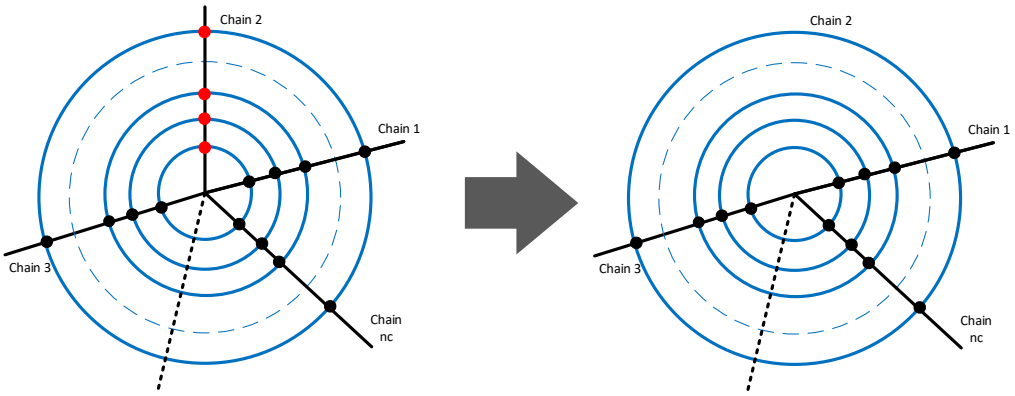


Figure 4.2: SPIDER Overlay self-reorganization in case of chain fall scenario 1.

Entire Chain Lost Scenario 2

Starting from the scenario overview presented with a n nodes network, the first critical disaster scenario implies the failure of all the nodes located on the same chain at the same time. For this scenario, we consider that nodes $N[2,*]$, from Chain 2 disappear at once, but the chain remains. Therefore, the overlays have to reconfigure itself from a structure with N nodes and nc chains in one if $N-nr$ nodes and nc chains.

In this scenario, the self-adaption technique that the overlay adopts is different from the ones mentioned in the previous section. The main difference consists of creating two new nodes from the remaining ones in the lost chain as seen in Figure 4.3. After the creation of the new nodes, we must update the neighbors' tables.

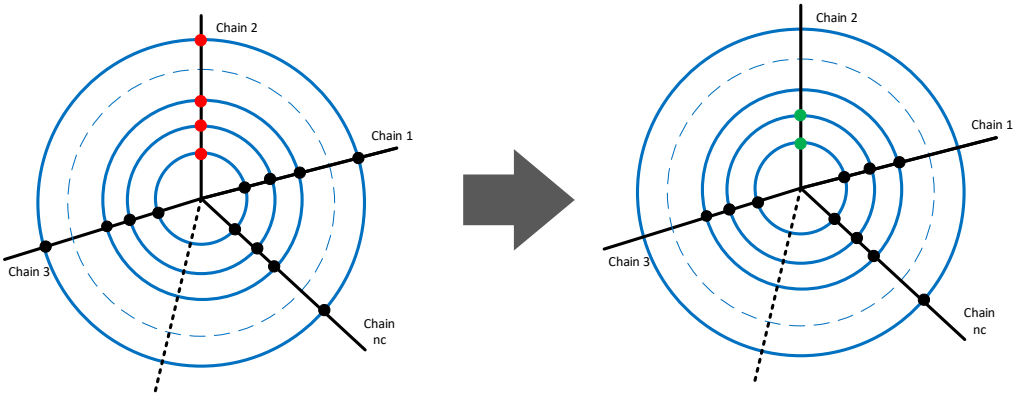


Figure 4.3: SPIDER Overlay self-reorganization in case of chain fall scenario 2.

For this scenario, we have taken into consideration that the structure of the overlay remains the same concerning the number of chains. Thus, when a flaw where all nodes from a chain fall at the same time the SPIDER overlay is reconstructed by keeping as much as possible of the initial structure of the overlay. In Algorithm 5 we present the self-recovery algorithm used in this case.

Algorithm 5 SPIDER - Entire chain fall keeping the chain structure of the overlay.

$X[*, r].FALL$
$NewNode[c, 1] = X[(c-1)\%nc, r]$
$NTable(NewNode(c, 1)).EMPTY$
$NewNode[c, 2] = X[(c+1)\%nc, r]$
$NTable(NewNode(c, 1)).EMPTY$
$X[(c-1)\%nc, r].DELETE$
$X[(c+1)\%nc, r].DELETE$
$NTable(X[(c-2)\%nc, r-1]).RIGHT = free$
$NTable(X[(c+2)\%nc, r-1]).LEFT = free$
$NTable(X[(c-1)\%nc, r-1]).UP = free$
$NTable(X[(c+1)\%nc, r-1]).UP = free$
for $i = 1 : nr - 2$ **do**
  if $i <= 2$ **then**
    $NTable(X[(c-1)\%nc, i]).LEFT = X[c, i]$
    $NTable(X[(c+1)\%nc, i]).RIGHT = X[c, i]$
    $NTable(X[c, i]).LEFT = X[(c+1)\%nc, i]$
    $NTable(X[c, i]).LEFT = X[(c-1)\%nc, i]$
  else
    $NTable(X[(c-1)\%nc, i]).LEFT = free$
    $NTable(X[(c+1)\%nc, i]).RIGHT = free$
  end if
end for
$NTable(X[c, 1]).UP = X[c, 2]$
$NTable(X[c, 2]).UP = free$
$NTable(X[c, 2]).DOWN = X[c, 1]$

The impact in this scenario is the highest. Not only, the number of updates in this scenario is very high, but in this case, we must create two new nodes on the lost chain. After this construction, a problematic neighbor table update schedule

must be effectuated. Equation 4.4 gives the number of operations in this scenario.

$$\begin{aligned} noOfOperations = newNodeCreation(1) \times 2 + deleteExistingNodes(1) \times 2 + \\ 4 + 2 \times newNodesUpdates(4) + \\ (nr - 3) \times nodeUpdateOfTheNeighborChains(2) + 3 \end{aligned} \tag{4.4}$$

Entire Ring Fall

Another common flaw in the SPIDER overlay is when all the nodes from a ring fail at the same time as shown in Figure 4.4. This flaw happens when all the nodes from a specific ring fall at the same time. This type of fault is considered to be a global one because the structure of the overlay is affected and the overlay has to auto-recover in a manner that changes its structure.

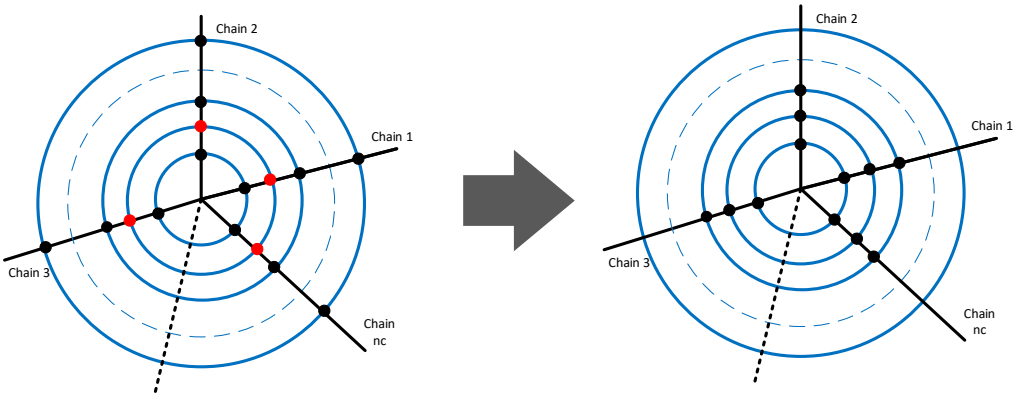


Figure 4.4: SPIDER Overlay self-reorganization in case of ring fall.

For this scenario, we take into consideration a SPIDER overlay network with N nodes formed by nc chains and nr rings. If a ring from the overlay falls, the overlay auto-adapts itself by making the upper and bottom connections of the collapsed nodes. Therefore, we present the Algorithm 6 for this scenario.

Algorithm 6 SPIDER - Entire ring fall.

$S[nr, nc]$ with n nodes
$X[c, *].FALL$
for $i = 0 : nc - 1$ **do**
  $NTable(X[c, r + i + 1]).DOWN = X[c, r - i - 1]$
  $NTable(X[c, r - i - 1]).UP = X[c, r + i + 1]$
end for
$nr = nr - 1$

This type of flaw is considered a global flaw because the overlay has to change its structure. The number of operations needed to be executed to auto-recover the SPIDER overlay in this case of flaw is given by $c * 2$ because for each chain there must be realized two connections between the upper neighbors to the lower ones and all the way around.

4.1.3 Permanent Flaws in SPIDER Peer-to-Peer Overlay

Taking into consideration the time metric, we assume that flaws can be permanent or temporary, depending on the failure timing frame. In case of permanent defects, nodes fall from the SPIDER overlay irrecoverably. In this case, the position of the node can be replaced by a new node. It is likely impossible that the same node to appear again in the overlay.

In the case of a SPIDER overlay with nodes that generates data such as sensors, cameras, microphones, etc., the flaws are considered to be permanent. The overlay has to auto-adapt to work correctly again. In this case, the reappearance of the same node in the overlay is possible only in the case when the device is repaired and not replaced by a new one.

4.1.4 Temporary Flaws in SPIDER Peer-to-Peer Overlay

A temporary flaw is considered when a node from the SPIDER overlay vanishes for a specified time frame. For instance, a mobile phone can disappear from the system when it runs out of battery or mobile signal.

In case of temporary flaws, we assume that the nodes are not replaced with other nodes, because when they reaper, they will have coordinates that they had before.

Temporary flaws are likely to occur in SPIDER overlays where the nodes are considered computational nodes.

4.2 Experimental Evaluation of the Fault Tolerance algorithms

The experimental evaluations of the SPIDER Peer-to-Peer flaw recovery algorithms was conducted through the implementation described in Section 1.2. We chose to evaluate the fault tolerance algorithms presented in this Chapter with different instances of SPIDER Peer-to-Peer overlay that varies from 4 to 12000 nodes orgasmed on maximum 6 chains and 2000 rings . We split the experimental assessment into two parts, because of the nature of the problems they resolve: local faults and global faults.

First, we analyze the evolution of the number of operations necessary for overlay recovery in case of failure for the 4 local flaws described in Section 4.1.1. Second, we evaluate the number of operations that must be executed in case of failure for the 3 global flaws recovery algorithms presented in Section 4.1.2.

In Figure 4.5, we analyze the number of operations that must be executed for failure recovery for each local flaw: one random node flaw, two nodes from the same chain flaw, two random nodes flaw and two nodes from the same ring randomly flaw in case of different instantiations of SPIDER Peer-to-Peer overlay from 4 nodes to 12000 nodes.

The obtained results show the fact that the number of operations for recovery in case of local flaws does not vary with the size of the SPIDER overlay, because the structure of the overlay is not affected and the coordinates of fallen nodes (chain and ring) are set to free. The most efficient algorithms in case of local flaws recovery is the One random node flaw. In this case, the number of operations for recovery is only 4. The most inefficient algorithms in case of local flaws recovery in terms of the number of operations for recovery is the two random nodes flaw, with a total amount of 8 operations per flaw.

In Figure 4.6, we present the experimental results obtained after the evaluation of the global flaw recovery algorithms: entire ring fall (Figure 4.6a), entire chain fall scenario 1 (Figure 4.6b) and entire chain fall scenario 2 (Figure 4.6c). We conduct the experimental evaluation based on the SPIDER overlay implementation with

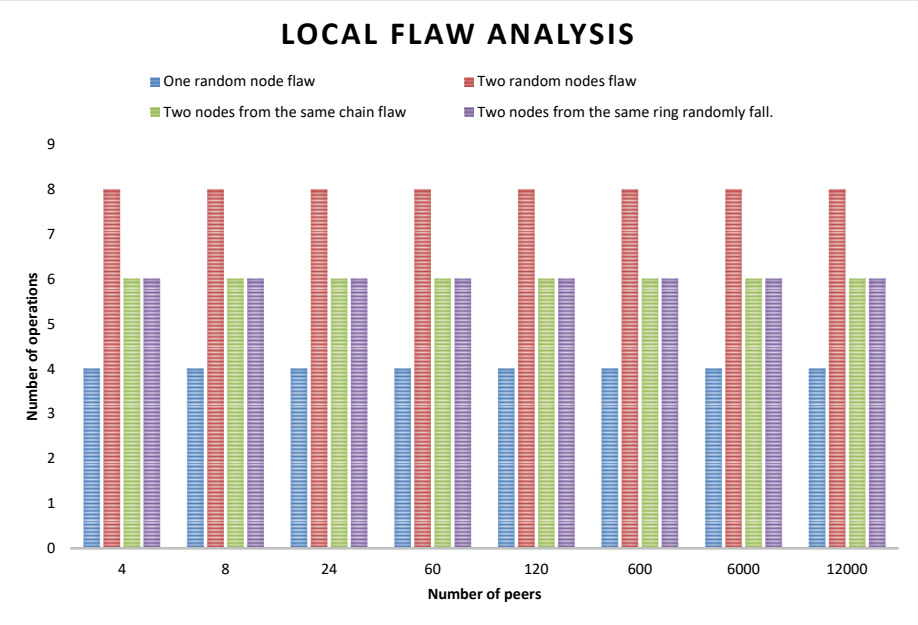
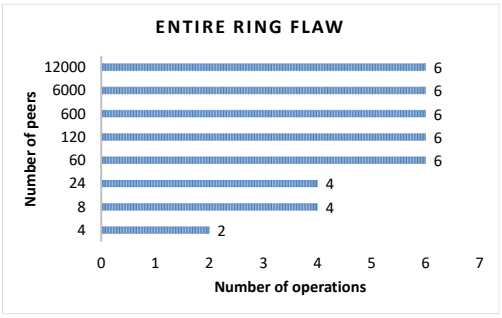


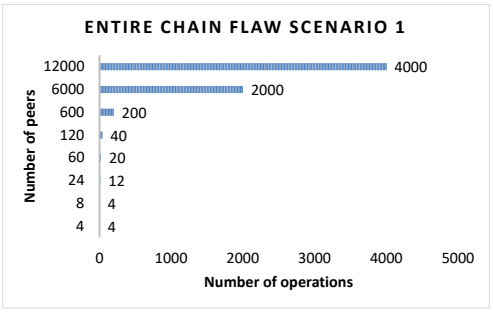
Figure 4.5: Number of operations needed for SPIDER Peer-to-Peer self-adaption in case of local flaws.

4 to 12000 nodes. For the entire ring flaw, the number of operations for failure recovery depends on the size of the SPIDER overlay, but with a small difference. The maximum number of operations remains the same for large size overlays, more than 60 nodes and the total amount of operations is only 6. This value is obtained because the structure of the SPIDER overlay evaluates on vertical (small number of chains with big number of rings). In comparison with the local flaws algorithms, the entire rig flaw algorithm has similar performances to the local flaws recovery algorithms in terms of number of operations that must be executed.

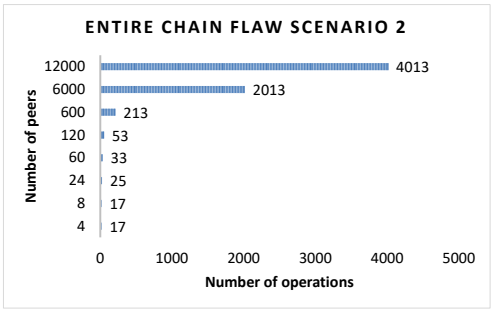
The performance evaluation of the entire chain flaw scenarios 1 and 2, show the fact that the recovery from these global type of flaws is highly coupled with the size of the SPIDER Peer-to-Peer overlay. Thus, for a SPIDER overlay with 12000 peers constructed on in a vertical manner , the maximum numbers of operations of recovery in case of the entire chain flaw scenarios 1 and 2 are 4000 and 4013 operations. These values are obtained because the number of rings is very high and the operations affect the nodes on the chain. The reason why the entire chain fall scenarios 2 is less efficient than the entire chain fall scenario 1 is the fact that in case of scenario 2 the structure of the overlay changes and are created two new



(a)



(b)



(c)

Figure 4.6: Number of operations needed for SPIDER Peer-to-Peer self-adaption in case of global flaws. Entire ring flaw algorithm (a), Entire chain flaw scenario 1 algorithm (b), Entire chain flaw scenario 2 algorithm (c).

nodes from the remaining ones in the lost chain.

We demonstrate the fact that SPIDER overlays implementations organized on rings (developed on vertical) are fault safer than SPIDER overlays organized on chains.

4.3 Discussions and Conclusions

In this section we presented several possible flaws of the SPIDER Peer-to-Peer overlay and the way the system auto-recovers from those states. Also were presented four algorithms for fault self-recovery. We have stated the fact that local faults do not affect the structure of the SPIDER overlay and the only actions that must be executed are the update of the neighbor tables. In the case of global flaws, the structure of the overlay is affected and it must

self-reconstruct. In the case of entire chain fall, we reconstruct the overlay with a decremented number of chains by 1. The same behavior happens when an entire ring falls, but the recovery process is more efficient.

Another essential conclusion of this research is the fact that SPIDER overlay can react faster to major flaws when the system is built on vertical. This aspect leads for the conclusion that IoT support applications are more suitable as use cases of SPIDER overlay in terms of robustness.

Chapter 5

Data Fusion in SPIDER Peer-to-Peer Overlay

In this chapter we aim to answerer RQ3 research question, announced in Section 1.1 (Problem Statement and Thesis Objectives) by proposing a novel set of methods for data fusion with the respect of the SPIDER Peer-to-Peer overlay limitations.

The proposed data fusion algorithms in this chapter does not refer the Bayesian Clustering methods presented in ([12]). In contrast, our approach, is strictly connected to the structure of the SPIDER overlay. Despite the existing solutions for data aggregation from heterogeneous sensor networks described in Section 2.5.4 (Data fusion for security enhancement), we present a set of algorithms fault tolerant data fusion methods and evaluate them through the experimental methodology described in Section 1.2 (Research Methodology).

Therefore, in this chapter we present a set of data fusion methods for the SPIDER Peer-to-Peer (Section 5.1) (Data Fusion Methods) that includes 2 algorithms for chain-based and ring-based data fusion. Also, in this chapter we present the experimental result of the evaluation of the data fusion algorithms (Section 5.2) (Experimental Evaluation of the Data Fusion Algorithms) as well as our discussions and conclusions (Section 5.3) (Discussions and Conclusions).

5.1 Data Fusion Methods

In this chapter we present two methods of data fusion realized on a SPIDER Peer-to-Peer overlay. In order to be more accurate, the formal description of the methods uses the following set of annotations:

- $S[nc, nr]$ - SPIDER overlay with nc chains and nr rings;
- n_i - node in the SPIDER overlay;
- $chain(n_i)$ - chain of node n_i;
- $ring(n_i)$ - ring of node n_i;
- $H(n_i, n_j)$ - number of hops between node n_i and node n_j;
- $DR(n_i, n_j)$ - difference of rings between node n_i and node n_j;
- $DC(n_i, n_j)$ - difference of chains between node n_i and node n_j.

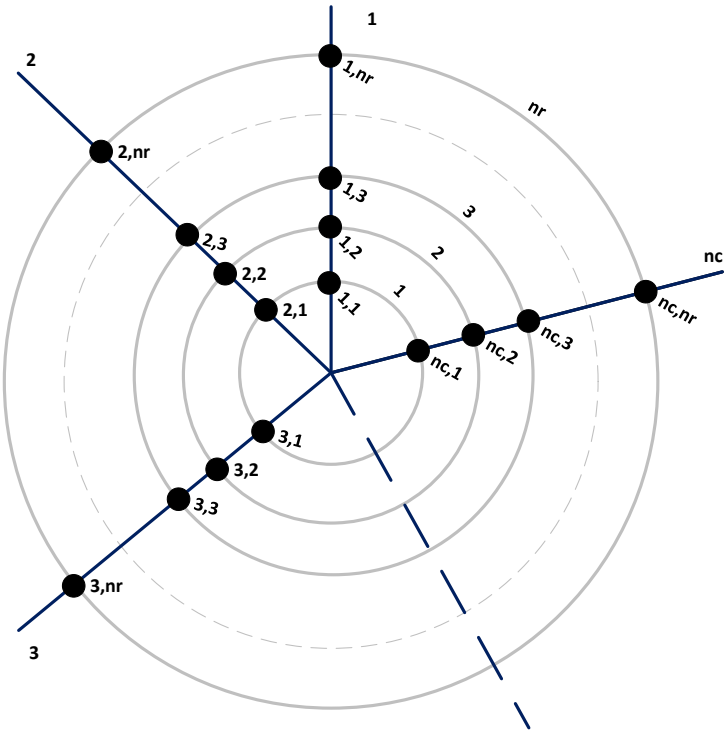


Figure 5.1: A SPIDER overlay with nr rings, nc chains and $totalNumberOfNodes = nr * nc$.

5.1.1 Chain-based Data Fusion

A particular characteristic of the SPIDER Peer-to-Peer overlay is the fact that the majority of operations can be realized in two manners: chain-based or ring-based. Therefore, data fusion on the SPIDER overlay can be done in the same manner.

Data fusion in the SPIDER overlay can be realized in a hierarchical manner. Thus the fusion is executed on chains since each ring represents a level in the SPIDER overlay architecture. In the chain-based fusion, also known as hierarchical fusion, data is transmitted from a node $node[c, r]$ to it's upper neighbor node $node[c, r+1]$ or to i's lower neighbor node $node[c, r-1]$ as shown in Figure 5.2.

The maximum number of hops in the chain-based fusion model is equal to the number of rings of the SPIDER overlay nr as shown in Equation 5.1.

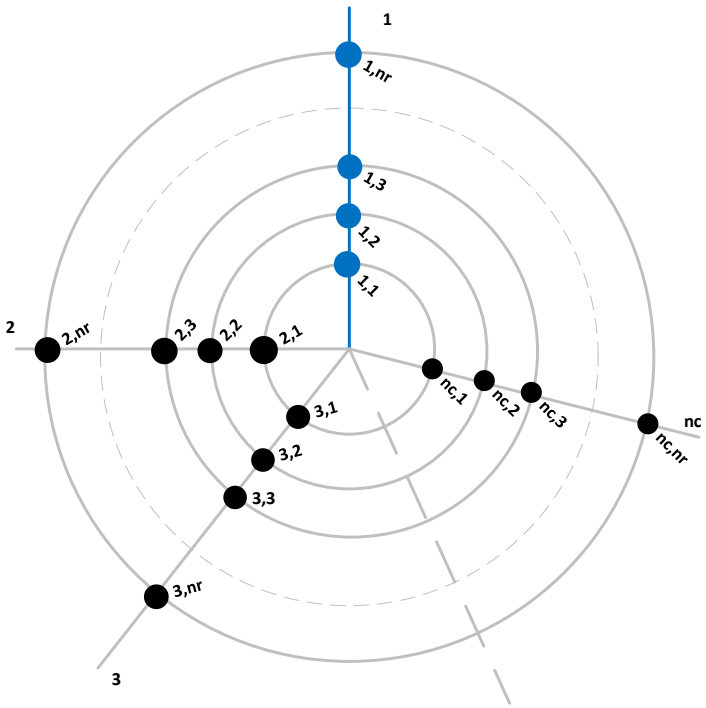


Figure 5.2: Data fusion on chains for the SPIDER Peer-to-Peer overlay.

$$ChainBasedMaximimHops = nr \tag{5.1}$$

If the overlay maps a system where data is aggregated only in a hierarchical

manner from one node to another, the overhead of the network is minimum, thus resulting only one hop.

In a SPIDER Peer-to-Peer overlay defined as $S[nc, nr]$, with nr number of rings and nc number of chains, data fusion on a chain can be realized by flowing a few steps. We must define the first two different nodes in the SPIDER overlay regarding their coordinates like:

- node n_i is the node on chain c and ring r_i;
- node n_j is the node on chain c and ring r_j

Each node in the SPIDER overlay also can defined as a simple function $n * (r - 1) + c$, where n is the number of nodes, c is the chain and r is the ring.

Furthermore, the chains and rings of the two nodes are computed as:

- node $chain(n_i) is (n_i - 1)\%nc + 1$;
- node $chain(n_j) is (n_j - 1)\%nc + 1$;
- node $ring(n_i) is \frac{n_i - chain(n_i)}{nc} + 1$;
- node $ring(n_j) is \frac{n_j - chain(n_j)}{nc} + 1$.

Furthermore, we compute the chain difference between the two nodes DR. This distance can have two values depending on the condition of $(|ring(n_i) - ring(n_j)| \leq \left[\frac{nr}{2}\right])$. Than is computed the chain distance as the absolute values of the subtraction of $chain(n_i) and chain(n_j)$.

Finally, we compute the chain based data-fusion value as the sum of chain distance and ring distance as $H(n_i, n_j) = DR(n_i, n_j) + DC(n_i, n_j)$.

The method proposed for data fusion on chains in a SPIDER overlay implementation is described in Algorithm 7.

Algorithm 7 SPIDER - Data fusion on chain.

1: $i \neq j$;
2: $n_i = node[c, r_i] = n * (r_i - 1) + c$;
3: $n_j = node[c, r_j] = n * (r_j - 1) + c$;
4: $chain(n_i) = (n_i - 1)\%nc + 1$;
5: $ring(n_i) = \frac{n_i - chain(n_i)}{nc} + 1$;
6: $chain(n_j) = (n_j - 1)\%nc + 1$;
7: $ring(n_j) = \frac{n_j - chain(n_j)}{nc} + 1$;
8: **if** $(|ring(n_i) - ring(n_j)| \leq \lceil \frac{nr}{2} \rceil)$ **then**
9: $DR(n_i, n_j) = |ring(n_i) - ring(n_j)|$;
10: **else**
11: $DR(n_i, n_j) = nr - |ring(n_i) - ring(n_j)|$;
12: **end if**
13: $DC(n_i, n_j) = |chain(n_i) - chain(n_j)|$;
14: $H(n_i, n_j) = DR(n_i, n_j) + DC(n_i, n_j)$.

A sensitive aspect regarding this type of data-fusion is the presence of node failure. Therefore, when talking about data fusion on chains we must take into consideration the two kinds of global flaws that are feasible in this scenario: Entire chain fall scenario 1 and scenario 2 described in Section 4.1.2.

5.1.2 Ring-based Data Fusion

Furthermore, as mentioned in the previous subsection, data fusion in the SPIDER overlay can also be realized on rings. In comparison with the chain-based data fusion method, the ring-based data fusion method is not a hierarchical one, because, the nodes are on the same level. A better understanding of this concept is presented in Figure 5.3. Thus, considering a SPIDER Peer-to-Peer overlay as $S[nc, nr]$, with nc chains and nr rings, the data aggregation is realized between the nodes from the same ring as the left neighbor $node[c-1, r]$ or the right neighbor $node[c+1, r]$.

Concerning the number of hops, in the ring-based data fusion algorithm the fusion values is a located between 1 and the total number of chains of the overlay as

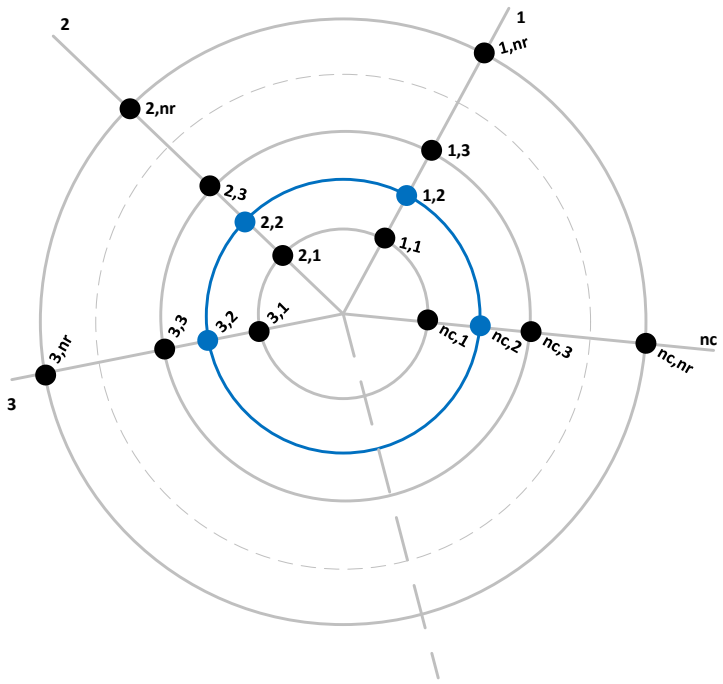


Figure 5.3: Data fusion on rings for the SPIDER Peer-to-Peer overlay.

shown in Equation 5.2.

$$NrHops \in [1, NrOfChains] \tag{5.2}$$

The maximum number of hops in ring fusion is equal to the number of chains of the SPIDER overlay. If the data is aggregated only with the nearest neighbors (left neighbor and right neighbor), the overhead of the Peer-to-Peer system network is minimal.

Stating with the same hypothesis of the chain-based data fusion method, we take into consideration two different nodes from the same ring, defined as $node[c_i, r]$ and $node[c_j, r]$.

For both of the chosen nodes, using the same technique described in the previous subsection, we compute the chain and ring position, and after these operations we compute the chain distance DC and the ring distance DR. We obtain the he ring-based data fusion value as the sum between the DC value and DR value.

In Algorithm 8 we propose the algorithm for the data fusion on rings.

Algorithm 8 SPIDER - Data fusion on ring.

1: $i \neq j$;
2: $n_i = node[c_i, r] = n * (r_i - 1) + c$;
3: $n_j = node[c_j, r] = n * (r_j - 1) + c$;
4: $chain(n_i) = (n_i - 1)\%nc + 1$;
5: $ring(n_i) = \frac{n_i - chain(n_i)}{nc} + 1$;
6: $chain(n_j) = (n_j - 1)\%nc + 1$;
7: $ring(n_j) = \frac{n_j - chain(n_j)}{nc} + 1$;
8: **if** $(|chain(n_i) - chain(n_j)| \leq \left[\frac{nc}{2}\right])$ **then**
9: $DC(n_i, n_j) = |chain(n_i) - chain(n_j)|$;
10: **else**
11: $DC(n_i, n_j) = nc - |chain(n_i) - chain(n_j)|$;
12: **end if**
13: $DR(n_i, n_j) = |ring(n_i) - ring(n_j)|$;
14: $H(n_i, n_j) = DC(n_i, n_j) + DR(n_i, n_j)$.

5.1.3 Fault Tolerance Aspects for SPIDER Data Fusion

In connection with the two methods for data fusion presented in this section we bridge the fault tolerance algorithms described in 4.1.

The proposed data fusion methods, are constructed based on the architecture of the SPIDER Peer-to-Peer overlay, thus they are build on chains and rings. Therefore, we bridge the chain based data fusion method with both local and global flaws methods because it is likely to in the data fusion process one of the following faults:

- One node randomly fall;
- Two nodes fall from same chain;
- Two nodes randomly fall;
- Entire chain flaw scenario 1;
- Entire chain flaw scenario 2.

Furthermore, concerning the data fusion method based on the rings, we take into consideration the following failures:

- One node randomly fall;

- Two nodes randomly fall from the same ring
- Two nodes randomly fall;
- Entire ring fall.

Considering these aspects, booth chain-based and ring-based data fusion methods presented in this chapter are prone to malfunction due to several failures. Because, the SPIDER overlay have good fault tolerance properties as demonstrated in Section 4.2 failures in the data fusion process are resolved via the fault tolerance algorithms proposed in Section 4.1. Taking into consideration the fact that local flaws are the most common type of failures in the SPIDER overlay, and the recovery from this type of states is speedy concerning computational operations, booth data fusion methods safe to be used. On the other hand, if we take into consideration the eventuality of a global flaw, the most suitable data fusion method is the one based on the rings because the number of operations is lower in this case in comparison with the number of operations for the chain-based data fusion.

5.2 Experimental Evaluation of the Data Fusion Algorithms

In order to evaluate the data fusion algorithms presented in this Chapter, first, we present an innovative concept for data aggregation in crowded cities named Smart streets. We evaluate the data fusion algorithms based on the results obtained from the smart-streets method for SPIDER Peer-to-Peer overlay mapping to the streets in a geographical area.

The smart-streets concept [2] is based on the street intersections paradigm, which is the most common construction in city. In order to have autonomous cars, one of the most important characteristic of Smart-streets concept is the aggregation of data from traffic infrastructure, sensors and other devices and deliver it to decision making factors in order to improve the quality of autonomous driving.

The main advantage of this approach is the fact that data is aggregated through streets and intersection not only on intersections, because we take into consideration the street sectors. Even though, the complexity of the SPIDER overlay is increased with this approach, the fusion techniques aggregate data

from more nodes, thus resulting more accurate information for the decision making factors.

To emphasis this concept, we take into consideration a generic neighborhood with perpendicular streets as shown in Figure 5.4. For a better understanding of this concept, we offer a schematic representation of a neighborhood, where each street is given a generic name like A, B, C, D for the horizontal streets and $1, 2, 3, 4$ for the vertical ones. Each street is divided into multiple street sectors depending on the number of intersections. For instance, for a neighborhood with four parallel streets and four vertical streets we generate in the flowing street sectors:

- $A : A12, A23, A34$;
- $B : B12, B23, B34$;
- $C : C12, C23, C34$;
- $D : D12, D23, D34$;
- $1 : 1AB, 1BC, 1CD$;
- $2 : 2AB, 2BC, 2CD$;
- $3 : 3AB, 3BC, 3CD$;
- $4 : 4AB, 4BC, 4CD$.

To validate our data fusion methods and the transformation algorithm from a perpendicular streets neighborhood to a SPIDER Peer-to-Peer overlay, we first settle the premises of the evaluation. Therefore, for the smart-streets use case we use the following test configurations:

- neighborhood formed by 2^2 streets : 2 horizontal - 2 vertical;
- neighborhood formed by 2^3 streets : 4 horizontal - 4 vertical;
- neighborhood formed by 2^4 streets : 4 horizontal - 12 vertical;
- neighborhood formed by 2^5 streets : 12 horizontal - 20 vertical;
- neighborhood formed by 2^6 streets : 20 horizontal - 44 vertical;
- neighborhood formed by 2^7 streets : 44 horizontal - 84 vertical.

We decide to chose these exponential growing configurations for the validation of the mapping method from perpendicular streets to the circular architecture of the SPIDER overlay because they demonstrate the scalability of the proposed

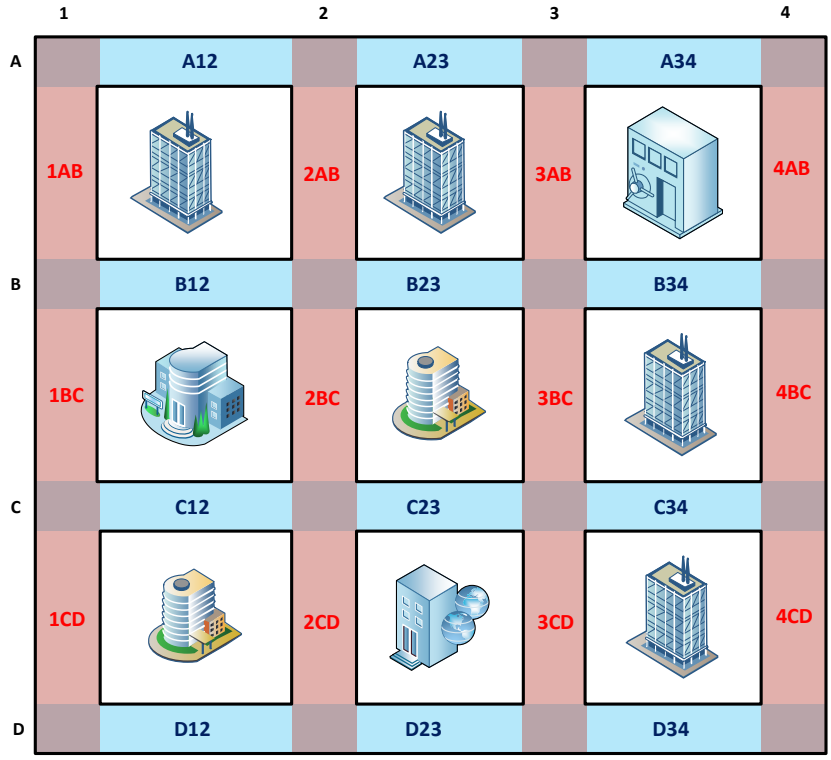


Figure 5.4: A section of streets in a neighborhood.

algorithm in case of an exponential growth of the number of streets associated with large cities.

Furthermore, we evaluate the values of the two data fusion methods proposed in this chapter with respect for the same exponential growing test scenario. The obtained results are presented in Figure 5.5. The obtained values show the fact that the values for the chain-based and ring-based data fusion methods are on an ascending trend. An interesting aspect shown is the fact that the ring-based data fusion method has a value of approximately 0.5 of the number of streets. In the case of a small number of streets (4 and 8), the values of the chain and ring data fusion are the same.

In addition, we analyze the values of the two data fusion methods concerning the same scenarios. The obtained numbers are presented in Figure 5.6. Examining the mean values of the data fusion proposed in this section, it is observable that both values are the same for a small SPIDER Peer-to-Peer overlay. The difference between the two values is growing with the size of the SPIDER overlay. Another aspect shown by the mean values of the data fusion methods is the fact that

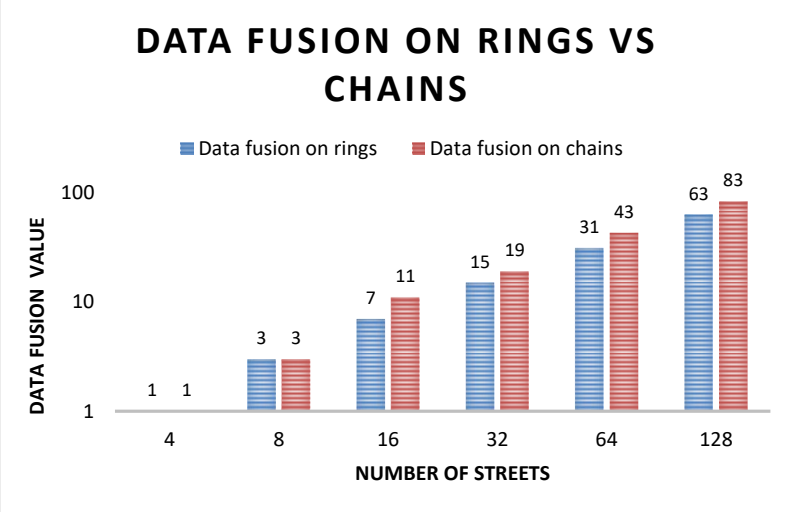


Figure 5.5: Data fusion on chains Vs Data fusion on rings.

ring-based data fusion is more efficient that chain-based data fusion.

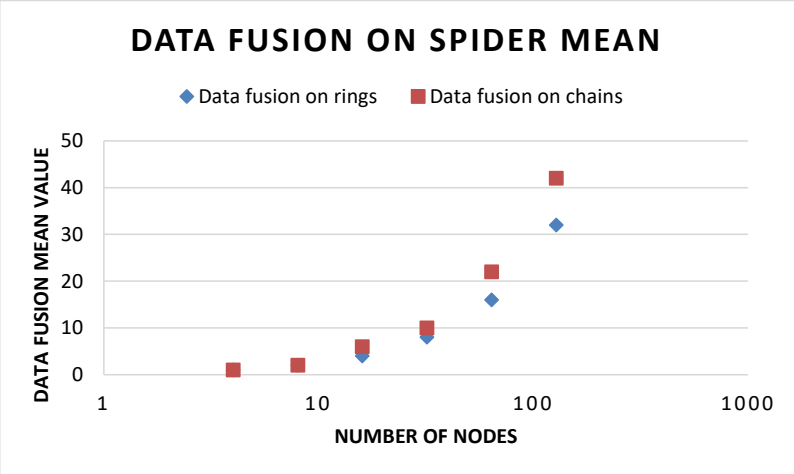


Figure 5.6: Mean data fusion values in SPIDER Peer-to-Peer.

5.3 Discussions and Conclusions

In this chapter, we presented two methods for data fusion in the SPIDER Peer-to-Peer overlay. In Section 6.3 we give two test scenarios for data fusion in smart

cities: IoT support for smart-streets and IoT support for waste recycling. For the intelligent streets scenario, we have proposed a new approach for creating the SPIDER overlay by considering the street sectors nodes in the overlay.

The experimental results obtained proved two essential aspects. The first one is considering local flaws in the SPIDER Peer-to-Peer overlay. In this case, booth data fusion methods can be applied without any concern because the recovery from a local defect is straightforward. The second one is taking into consideration the possibility of a global flaw in the overlay. Thus, the most efficient data fusion method, in this case, is the one based on rings.

Another aspect also demonstrated in this section is the fact that the SPIDER Peer-to-Peer overlay is more efficient when the nodes are organized on rings.

Chapter 6

SPIDER Peer-to-Peer Overlay Applications

In this chapter we aim to answerer RQ4 research question, announced in Section 1.1 (Problem Statement and Thesis Objectives) by investigating the requirements of modern applications designed to facilitate the decision-making process in large systems based on the Peer-to-Peer overlay networks.

We present two real use cases for live video streaming applications based on the SPIDER Peer-to-Peer overlay that has a major benefit from using the SPIDER approach in comparison with the classical approach where data is pushed through all nodes of the system.

Also, in this chapter we tackle the domain of Internet of Things by presenting the novel concept of Smart-streets that is used for data fusion, and 2 SPIDER-based approaches for IoT for smart-homes and recycling facilities.

In addition, we propose a set of a trust management algorithm for node trust and link trust that benefit from the advantages of the structure of the SPIDER overlay in terms of the history of the nodes as described in Section 2.5.5 (Trust as a security feature).

We evaluate these use cases in terms of performance using the experimental methodology described in Section 1.2 (Research Methodology) and real life use cases.

Therefore, in this chapter we offer a set of applications that benefit from the SPIDER Peer-to-Peer overlay: live video streaming applications (SCA) for

Cyber-physical infrastructures (Section 6.2) (Live Video Streaming in Cyber Physical Infrastructures), Smart-city applications (Section 6.3) (SPIDER Peer-to-Peer Overlay Approach in Smart-cities) and trust management system for eCommerce in Peer-to-Peer networks (Section 6.4) (Trust Management System for E-commerce in Peer-to-Peer Networks) as well the experimental result of the evaluation of the proposed applications. Also in this chapter we provide a set of requirements for Peer-to-Peer applications (Section 6.1) (Requirements for Peer-to-Peer Applications) and our discussions and conclusions (Section 6.5) (Discussions and Conclusions).

6.1 Requirements for Peer-to-Peer Applications

The development of novel applications based on Peer-to-Peer theologies has set an increasing trend in computer science research communities in the past years. On a daily basis, researchers are developing new innovative methods for data management in Smart-cities or Cyber-physical infrastructures. Therefore, a specific set of requirements for these applications must be satisfied.

In this section, we identify and elaborate on several general requirements for Peer-to-Peer applications.

We analyze as well, two major types on Peer-to-Peer applications: live video streaming applications and IoT support applications. Both types of applications are complex ecosystems that benefit from the advantages of Peer-to-Peer technologies. In [57], the authors propose a fundamental set of requirements for Peer-to-Peer systems that includes decentralization, scalability efficiency and fault tolerance.

In terms of decentralization, the solution must not rely on the client-server paradigm. It rather be fully decentralized, where only local data is processed locally. The scalability of the solution is an important requirement because it must scale in systems with a large amount of nodes. Concerning the efficiency of the Peer-to-Peer application, it must be able to process queries from many peers in parallel without deadlocks. Finally, the last fundamental requirement tackles the capability of the system to function in case of failures.

In addition to these generic demands, the authors of [142] and [145] identified a series of network requirements for Peer-to-Peer systems such as bandwidth, delay,

and loss. The bandwidth of the system is the size of the communication channel through the data is passed. The delay represents the accepted amount of time, usually in milliseconds, that it takes the data to pass from one peer to another. The loss property represents the number of network patches that are dropped without affecting the quality of the service.

Taking into consideration the requirements mentioned above, we propose a set of requirements for Peer-to-Peer applications designed over the SPIDER overlay:

- physical infrastructure ([96], [70], [152], [9]);
- transport protocol ([68], [121], [150]);
- bandwidth and throughput ([95]);
- fault-tolerance ([25], [30]);
- trust ([127], [67]);
- auto-adaptability ([51], [6]).

Peer-to-Peer applications are complex systems that run over a physical underlay. The physical connection between nodes is a significant requirement for these applications because of the functionality of the applications rely on it. Thus, for critical Cyber-Physical Infrastructure applications like live video streaming the network between nodes must be stable and trusted. In case of systems that are located in large geographical areas, such as live video streaming for river pollution monitoring, the network between nodes must be cable-based, or GSM-based. In the case of IoT applications, the network between nodes must be WiFi.

In terms of the transport protocol, for live video streaming applications the only protocol supported, is UDP because if some packets are dropped or lost the system is not affected. On the other hand in IoT support applications booth UDP and TCP protocols can be used.

Two critical requirements for Peer-to-Peer applications is bandwidth and throughput. According to [28] in the case of IoT support applications, the bandwidth should not be higher than regular 2G networks (250KB/s). On the other hand, when talking about live video streaming applications, the bandwidth must high (more than 2 MB/s).

Fault-tolerance of Peer-to-Peer systems is also an important requirement because these network systems are prone to malfunction. Therefore, booth live video

streaming Peer-to-Peer systems and IoT support systems must be fault-tolerant. We consider 2 major types of failures: local flaws and global flaws. We consider a local flaw in the system that affects only a small part of the system and the impact, in this case, is minor, meaning that the structure of the overlay is not altered. While on the other hand global failures affect multiple nodes and the impact of the system's functionality is seriously affected. In this case, the structure of the overlay changes and the nodes in the system must reconfigure in order to maintain the functionality of the network.

In terms of security, we propose an important requirement for the trust of nodes and links between nodes. Taking into consideration, the fact that the Peer-to-Peer applications are fully decentralized, in their architecture there is no central authority of trust, each node in the system must have a trust value and keep a reference to the trust values of its neighbors. Another important aspect in terms of security of the system is the trust of the links between nodes. Therefore, both type of applications must implement trust mechanisms.

To summarize the requirements described above, in Table 6.1 we present a short overview of the requirements of the Peer-to-Peer applications. As it can be seen, the requirement of bandwidth for Live video streaming applications is higher than the necessary bandwidth of IoT applications.

Table 6.1: Peer-to-Peer overlay application requirements.

Application type requirements	**Live video streaming**	**IoT Support**
Physical infrastructure	Cable/GSM (2G-4G)	WiFi
Transport protocol	UDP	UDP/TCP
Bandwidth and throughput	>2 MB/s	>250KB/s
Self-recovery from local flaws	Yes	Yes
Self-recovery from global flaws	Yes	Yes
Security	Trust	Trust

Based on [108, 154] in Table 6.2 we present a brief description of the specifications of several WiFi standards that are feasible for the SPIDER Peer-to-Peer IoT applications.

Concerning the fault-tolerance requirements of Peer-to-Peer systems we present a more in-depth analysis of the methods for fault recovery in Chapter 4 (Robustness of SPIDER Peer-to-Peer Overlay) of this thesis.

We defined 2 general types of faults for the SPIDER Peer-to-Peer overlay: local flaws and global flaws.

Table 6.2: WiFi specifications comparison.

	Zigbee	**UWB**	**Bluetooth**	**BLE**	**ZWave**	**Wavevis**
Frequency	868/915 MHz 2.4 GHz	3.1-10.6 GHz	2.4 GHz	2.4 GHz	<1 GHz	868/915/ 433 MHz
Data rate Mbps	0.25	110	3	1	0.04	0.1
Range km	0.1	0.01	0.1	0.2	0.03	4
Battery life	days-years	years		weeks	years	years
Network topology	P2P STAR MESH		P2P	P2P	MESH	P2P
Power consumption	low	low	low	ultra-low	low	ultra-low
Open	yes	yes	yes	yes	yes	yes
IPv6	yes	no	mo	yes	yes	yes
Target	smart devices	real-time	smart devices	health fitness devices	smart home	M2M telemetry smart home

Local flaws appear in the SPIDER overlay very often because nodes are electronic devices such as sensors, cameras, smart-home devices, etc. that are prone to malfunction. In the case of mobile devices another fact that determines flaws is the loss of GSM/3G/4G signal or battery life failure. These failures are:

- one random node fault;
- two nodes fall from the same chain;
- two random nodes fault;
- two nodes fall from the same ring.

Global flaws appear in the SPIDER Peer-to-Peer overlay. These type of flaws affect the entire structure of the SPIDER overlay. The number of chains or rings is modified and the overlay must auto-rebuild to be functional again. We divide the global flaws into the following categories:

- entire chain fault - scenario 1;
- entire chain fault - scenario 2;

- entire ring fault.

We assume that for live video streaming Peer-to-Peer applications, applications with a high impact in the process of decision management of large scale phenomenon fault-tolerance for local flaws should be present as well as for the entire chain fault - scenario 1. In this type of systems, local flaws must be resolved very fast and the impact, in this case, should be small.

Concerning IoT support Peer-to-Peer applications all the fault-tolerance methods should be available, except the entire chain fault - scenario 2 because the nodes of the Peer-to-Peer are sensors. In case of these type of applications, fault recovery is mandatory.

6.2 Live Video Streaming in Cyber Physical Infrastructures

Cisco [63] estimates that in 2019, video streaming over the Internet will reach 80% of the World Wide Web traffic.

The authors of [83] distinguish 2 types of video streaming services: video streaming on demand and live video streaming. In live video streaming systems, the video footage is captured and transmitted through a communication channel. In this case, the most important characteristic of the system is to assure that the videos are delivered with a low packet loss rate ([46], [146]).

In this section, we present two real-life use cases for live video streaming systems for decision making in Cyber-Physical Infrastructures: ClueFarm and CyberWater. In booth use-cases, the video stream is delivered to a management system where the decision-making process is performed. We explore the physical underlays of the systems, the logical overlays based on the SPIDER Peer-to-Peer overlay and demonstrate the performances of these systems in comparison with systems that do not benefit from the advantages of the SPIDER overlay.

Boot real live use cases were part of 2 research projects developed and managed in UPB.

6.2.1 SPIDER Peer-to-Peer Overlay Approach in ClueFarm Project

A state of the art concept for the farming industry based on the vertical farming paradigm is presented in [14]. This concept uses a new model of farming that is located inside of shipping containers. This type of farming is cleaner and greener in comparison with classical farming.

First, there are 3 types of containers depending on their size as shown in Table 6.3:

Table 6.3: Container specifications for the ClueFarm Peer-to-Peer.

	3 m container	**6 m container**	**12 m container**
Length (m)	3	6	12
Width (m)	2.4	2.4	2.4
Hight (m)	2.6	2.6	2.9

The main idea of the ClueFarm monitoring applications is that each greenhouse is equipped with a cabled or WiFi IP surveillance camera with HD resolution and low light sensitivity.

Taking into consideration that these applications focus on live video streaming the requirements in terms of communication infrastructure are listed below:

- physical infrastructure - UTP cable Cat5/Cat6 or WiFi IEEE 802.11 a/g/n/ac;
- transport protocol - TCP/UDP;
- bandwidth - more that 2MB/s;
- fault-tolerance in case of small hazards - yes;
- fault-tolerance in case of global hazards - yes;
- throughput - high.

Furthermore, we take into consideration a farm for fresh vegetables with $NoGH$ vertical greenhouses, organized in NoZ zones as it can be seen in Figure 6.1. The organization of the farm is given by the type of containers used.

Each greenhouse has an IP surveillance camera installed in the corner of the container to visualize the entire greenhouse. The main idea of this approach

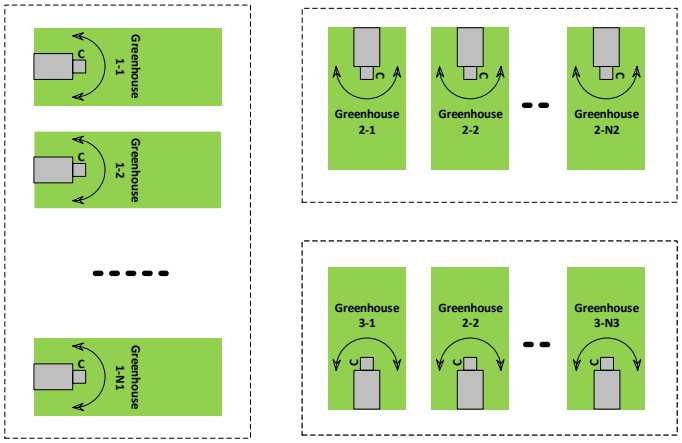


Figure 6.1: Farm physical organization overview. Variable number of greenhouses organized in NoZ zones.

is that by using smart technologies, the farmer can envision in real time the activity of the greenhouse on the farm. Therefore, all the video captured by the surveillance cameras is automatically transmitted to a Cloud Service (CS) to be visualized in a grid form on the farmers mobile phone as shown in Figure 6.2.

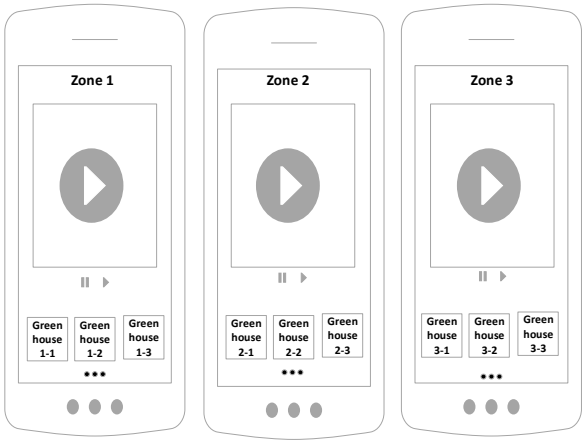


Figure 6.2: Centralized view of the surveillance cameras in the farm.

The physical underlay of the farm scenario is described in Figure 6.3. There is a centralized Cloud service for centralization and dissemination of the video content. Each zone has a fixed number of cameras equal to the number of greenhouses in each zone.

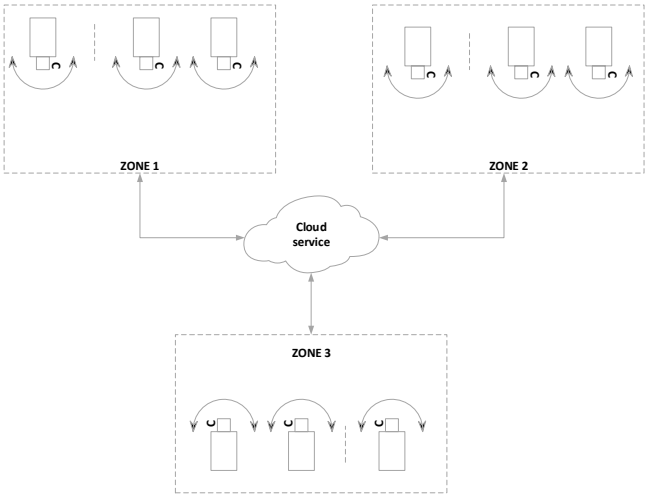


Figure 6.3: Farm underlay. Farming zone having a fixed number of cameras and interconnected to a Cloud-based service.

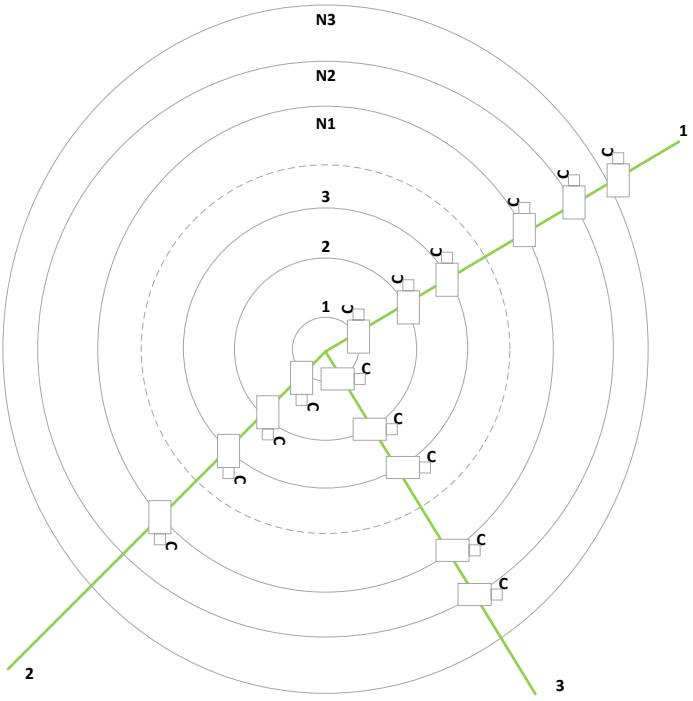


Figure 6.4: Farm SPIDER overlay. 3 chains representing the farming zones and 3 rings representing the video cameras in each zone.

As mentioned above, all video content is synchronized in real time with a Cloud service. We organize the physical underlay presented in Figure 6.3 in a structured based on the SPIDER Peer-to-Peer overlay. Therefore, each chain in the overlay

is represented by a farming zone, and a ring represents each greenhouse. Thus the number of greenhouses is variable. Figure 6.4 presents the SPIDER overlay logical organization of the farming service.

In this subsection, we presented a real use-case for vertical farming from the physical architecture to the logical structure based on the SPIDER overlay. We mapped each zone of the farm with a chain in the overlay because these values are fixed. The number of containers in each zone is variable, thus, we mapped the number of the rings fo the overlay with the maximum number of containers in each zone. This approach is based on the best use case of SPIDER construction as we demonstrate in Section 3.2. 6.2.3.

6.2.2 SPIDER Peer-to-Peer Overlay Approach in CyberWater Project

In this section, we present a live video streaming Peer-to-Peer based solution for river pollution surveillance in wide geographical areas.

Therefore, we consider a vast geographical territory like a country. For instance, in Romania, there are 10 hydrographic basins including the Danube. Each basin is composed of the main river and its tributaries. In Table 6.4 we present the largest hydrographic basins located in Romania and their number of hydrographic stations.

Table 6.4: Hydrographic basins in Romania.

Hydrographic basin	Length of rivers (Km)	Number of hydrographic stations
Somes-Tisa	2689	6
Crisuri-Barcau	1073	5
Mures	3050	6
Nera-Timis-Barzava	1913	10
Cerna-Jiu	1830	12
Olt	5029	5
Arges-Dambovita	1328	13
Ialomita	321	16
Siret-Prut	6400	16
Danube	1133	9

For a better understanding of the scale of the hydrograpfic basins in Romania we offer in Figure 6.5 a more comprehensive visual representation.

We asseume that on each river are situated several observation points. The number of observation points is variable and depends on the length of the river or the number of hydrographic stations, which determines the size of the hydrographic basin.

Figure 6.5: Hydrographic basins in Romania.

Furthermore, we offer a Peer-to-Peer system based on the SPIDER overlay for river pollution surveillance.

The basic architecture for the CyberWater live video streaming system using the SPIDER Peer-to-Peer overlay is presented in Figure 6.6. It consists of a cloud service that receives the video signal captured by a video camera via a GSM-GPRS module.

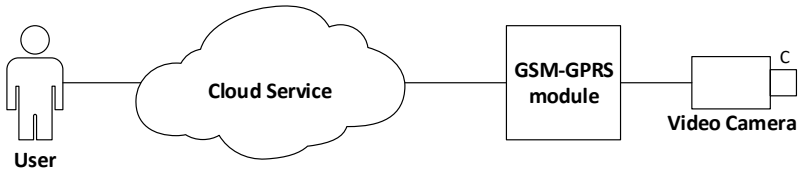


Figure 6.6: Basic arhitecture of the SPIDER Peer-to-Peer system for CyberWater live video streaming.

Therefore, we identify the main requirements in terms of communication infrastructure as:

- physical infrastructure - GSM-GPRS;
- transport protocol - TCP/UDP;

- bandwidth - more that 2MB/s;
- fault-tolerance in case of small hazards - yes;
- fault-tolerance in case of global hazards - yes;
- throughput - high.

In Figure 6.7, we show the physical architecture of the live video streaming system. This thesis aims to focus on the organization and management of the observation points. The Hydro-graphic headquarters and the command center of each hydrographic basin are not considered in this research. As mention, each hydrographic basin can have multiple observation points, each node having several surveillance cameras.

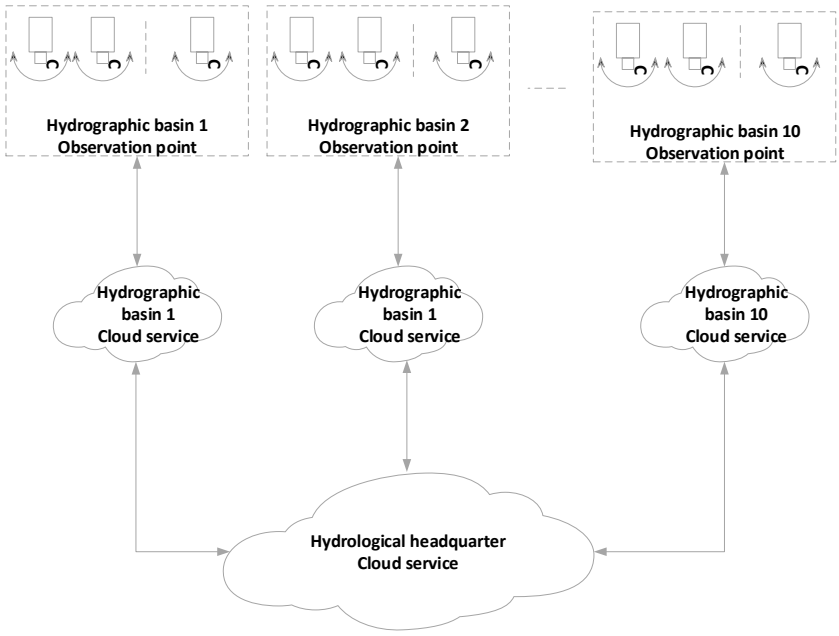


Figure 6.7: Hydro-graphic physical underlay. Several observation points on each hydrographic basin having multiple video cameras.

The video content captured by each observation point is sent to a Cloud-based service which is synchronized in real time with the river command center and the hydro-graphic headquarter. In Figure 6.8, we present the logical architecture of this deployment. The Peer-to-Peer overlay used for these scenarios is SPIDER. Because, there is a fixed number of hydrographic basins,

from 1 to 10, as mentioned in Table 6.4. Each hydrographic basin is considered to be a chain. And the observation points are represented by the rings. Thus, the number of observation points is variable, based on the number of observation points on each hydrographic basin.

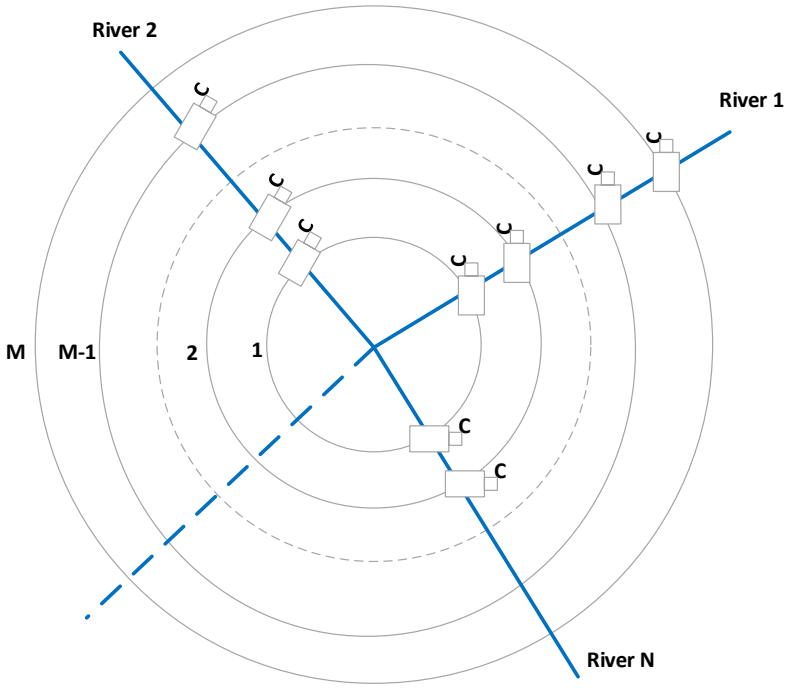


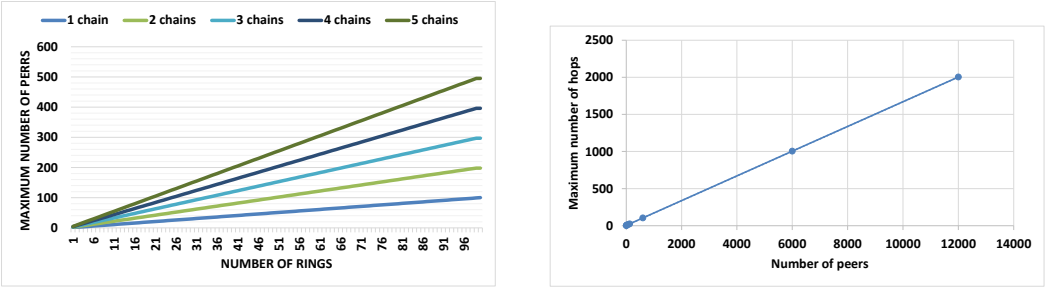
Figure 6.8: Hydrographic SPIDER overlay for the observation points.

In this subsection, we presented a real use-case for river pollution surveillance in wide geographical areas. The ClueFarm use-case takes into consideration the hydrographic basins of the rivers in Romania. Based on the results obtained in 3.2 we mapped the rivers as the chains in the overlay because their number is fixed. The maximum number of hydrographic basins of each river is mapped as the number of rings in the overlay because these values are variable.

In Section 6.2.3 we demonstrate the performances of these systems in comparison with a system that does not benefit from the SPIDER overlay advantages.

6.2.3 Experimental evaluation of the Live Video Streaming Applications

In this section, we present the experimental results obtained based on the data presented in the previous section. We evaluated the performance of the SPIDER Peer-to-Peer overlay for a Cyber-Physical Infrastructure system for water pollution monitoring.



(a) (b)

Figure 6.9: SPIDER Peer-to-Peer overlay. Number of peers Vs Number of Rings (a), Number of peers Vs Number of maximum hops (b).

Therefore, in Figure 6.9a, we show the evolution of the number of the maximum permitted peers in the overlay depending on the number of chains and rings. Therefore, when the number of rings is high, 100 rings and the number of chains is big, five chains, the total number of peers in the overlay is 500 peers. The evolution of the number of nodes in the system is linear.

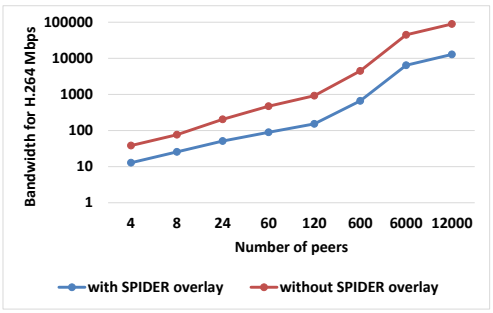
In Figure 6.9b, we present a graph of the maximum number of hops in the overlay depending on the number of the peers in the overlay. As can be seen, for 12000 peers in the SPIDER overlay, there are only 200 maximum hops.

The following experiments, evaluate the performance of the SPIDER overlay in the context of live video streaming in a Cloud-based service. Thus, all the video data captured by the cameras in booth scenarios presented in the previous section are transmitted to a Cloud service via a GSM-GPRS communication module. The cloud service implements the spider web overlay properties, and therefore the performance of the overlay is measured in this context. For this evaluation we have considered 3 different outdoor video cameras with the following specification:

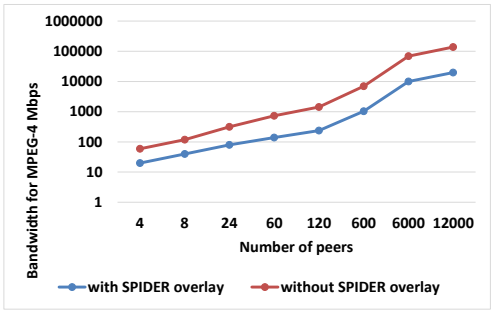
- network IP camera with the flowing specifications: Camera Stream: H.264 [144], Camera Resolution: 1280x1014, Video Quality: High, Frame Rate:

30fps (Figure 6.10a);

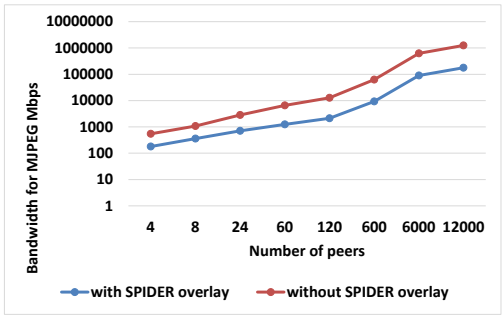
- network IP camera with the flowing specifications: Camera Stream: MPEG-4 [107], Camera Resolution: 1280x1014, Video Quality: High, Frame Rate: 30fps (Figure 6.10b);
- network IP camera with the flowing specifications: Camera Stream: MJPEG [48], Camera Resolution: 1280x1014, Video Quality: High, Frame Rate: 30fps (Figure 6.10c).



(a)



(b)



(c)

Figure 6.10: SPIDER Peer-to-Peer overlay bandwidth analysis for the ClueFarm use case. Bandwidth for H.264 (a). Bandwidth for MPEG-4 (b). Bandwidth for MJPEG (c).

For all 3 cases we evaluate the bandwidth needed depending of the number of peers in the SPIDER overlay in comparison with bandwidth needed if there is no overlay organization in the Cloud. Thus, the obtained values are presented in Table 6.5.

As it can be seen, the values of the bandwidth for the implementation of the SPIDER overlays in all the cases (H.264, MPEG-4, and MPEG) are almost 0.5

Table 6.5: Experimental results for 3 case studies: H.264, MPEG-4 and MJPEG with SPIDER overlay and without overlay.

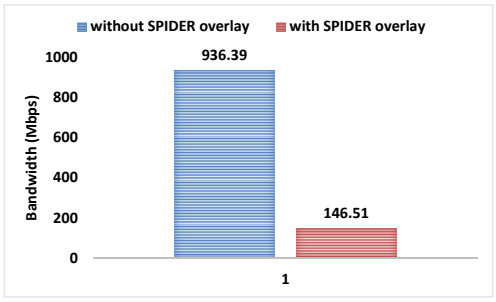
No. peers	H264 without overlay (Mbps)	H264 with overlay (Mbps)	MPEG-4 without overlay (Mbps)	MPEG-4 with overlay (Mbps)	MJPEG without overlay (Mbps)	MJPEG with overlay (Mbps)
4	12.74	25.48	19.8	39.6	178.24	365.48
8	25.48	50.96	39.6	79.2	365.48	712.96
24	50.96	152.88	79.2	237.6	712.96	2138.88
60	89.18	382.2	138.6	594	1247.68	5347.2
120	152.88	764.4	237.6	1188	2188.88	10694.4
600	662.48	3822	1029.6	5940	9268.48	53472
6000	6395.48	38220	9939.6	59400	89476.5	534720
12000	12765.5	76440	19839.6	118800	178596	1069440

better than the values of the bandwidth without the implementation of the SPIDER Peer-to-Peer overlay, in case of small sized networks. Another interesting aspect is that the values of the bandwidth in case of a SPIDER implementation with a large number of nodes, more than 10000 nodes, are even 4 times smaller than the values of the bandwidth without the SPIDER implementation. These numbers show that in the case of video content delivery is better to use SPIDER Peer-to-Peer overlay in comparison with classical network systems.

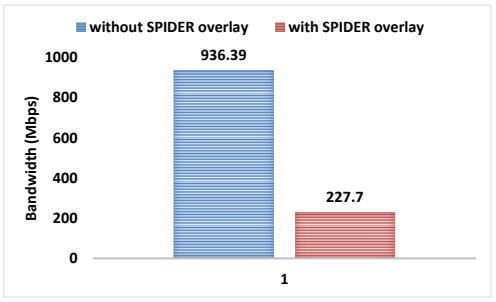
In Figure 6.11, we present the experimental results for the CyberWater case study presented in subsection 6.2.2. Based on the hydrographic basins located in Romania we defined a SPIDER Peer-to-Peer overlay with 10 chains and 16 rings. The number of chains is given by the number of major hydrographic basins and the number of rings is defined by the maximum number of hydrographic stations on each basin. As it can be seen, the implementation of the SPIDER Peer-to-Peer overlay reduces massively the bandwidth needed for video streaming. The blue series represent the bandwidth without the implementation of the SPIDER overlay. For all three cases, the bandwidth when implementing the SPIDER Peer-to-Peer overlay is 80% lower than the values obtained without the SPIDER implementation.

In this section, we described two realistic applications of SPIDER Peer-to-Peer overlay for the ClueFarm and CyberWater case scenarios. We evaluated the performances of the SPIDER Peer-to-Peer overlay on real datasets. The results assure that live video streaming applications developed based on the SPIDER overlay use significantly lower bandwidth than classical approach applications.

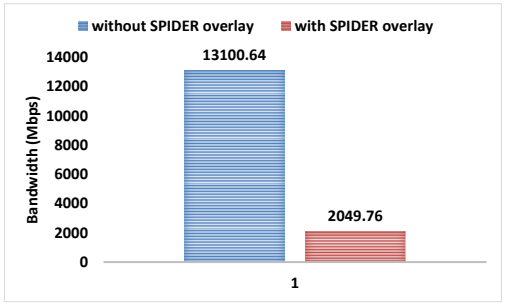
The obtained values explained in this section show that the implementation of the SPIDER Peer-to-Peer overlay increases the performances of the systems. The



(a)



(b)



(c)

Figure 6.11: SPIDER Peer-to-Peer overlay bandwidth analysis for the CyberWater use case. Bandwidth for H.264 (a). Bandwidth for MPEG-4 (b). Bandwidth for MJPEG (c).

main reason for this gain is the fact that in the SPIDER overlay each node has a maximum 4 nodes resulting in better performances because the path between nodes is significantly lower, in this case.

6.3 SPIDER Peer-to-Peer Overlay Approach in Smart-cities

According to Fraunhofer FOKUS [1], by 2050, around 70% of the overall population will live in the urban areas called Smart Cities. It is expected that buildings, streets, vehicles, waste recycling facilities, institutions, and many others will embed a communication channel, where data is transfered for faster

[1]https://www.fokus.fraunhofer.de/en/sqc/news/smart_city_botschaft, Accessed: 29-05-2019

and reliable decision making processes.

In such a complex technological environment, the most important challenge is the manner data is aggregated and delivered to decision-making factors. According to [11, 10, 77], the best approach for smart-home systems is the Peer-to-Peer one. Therefore, in this section, we present 3 real use cases implementations for smart-streets, smart-home and smart-recycling using the SPIDER Peer-to-Peer overlay. By the end of this section, we demonstrate using real datasets booth the fault tolerance and data fusion characteristics of the SPIDER Peer-to-Peer overlay implemented in these real use cases.

6.3.1 Smart Streets Concept

The Smart-city concept ([153]) has become one of the most offering research topic in the last years. An emerging ecosystem in the Smart-cities is the Smart-streets concept [2] that deals with data collection, aggregation, and dissemination from the infrastructure of the streets.

In this section, we refine the concept of Smart-streets and we propose a novel method for mapping the streets of real cities to the circular SPIDER overlay structure. Also, we evaluate the data fusion algorithms, presented in detail in Section 5.1 using real-life datasets.

For this use case scenario we take into consideration an area from a city that is composed with streets and avenues crossing each other into a perpendicular way (for instance New-York Manhattan as shown in 6.12). We define N as the number of streets and M as the number of avenues.

For a better understanding we reduce the complexity of problem given by the streets and avenues in New-Yorks to a schematic representation of N streets and M avenues as shown in Figure 6.13.

A generic formula for computing the total number of street sectors is given by the Equation 6.1., where $f(NVS)$ represents the number of vertical streets -1 and $f(NHS)$ represents the number of horizontal streets -1. The total number of street sectors is computed the sum of the products of the $f(NVS)$ and number

Figure 6.12: New-York Manhattan street overview.

of parallel streets and $f(NHS)$ and number of vertical streets.

$$\begin{aligned} f(NVS) &= NrVerticalStreets - 1 \\ f(NHS) &= NrHorizoltalStreets - 1 \\ TotalNrOfStreetSectots &= f(NHS) * NrHorizoltalStreets + \\ & \quad f(NVS) * NrVerticalStreet \end{aligned} \tag{6.1}$$

Because the structure of the streets is perpendicular, a SPIDER based approach, in this case, is not convenient, because of the circular structure of SPIDER overlay.

Furthermore, we propose a method of mapping perpendicular streets to the SPIDER overlay structure. This method consists of 2 stages:

1. intermediate SPIDER structure;

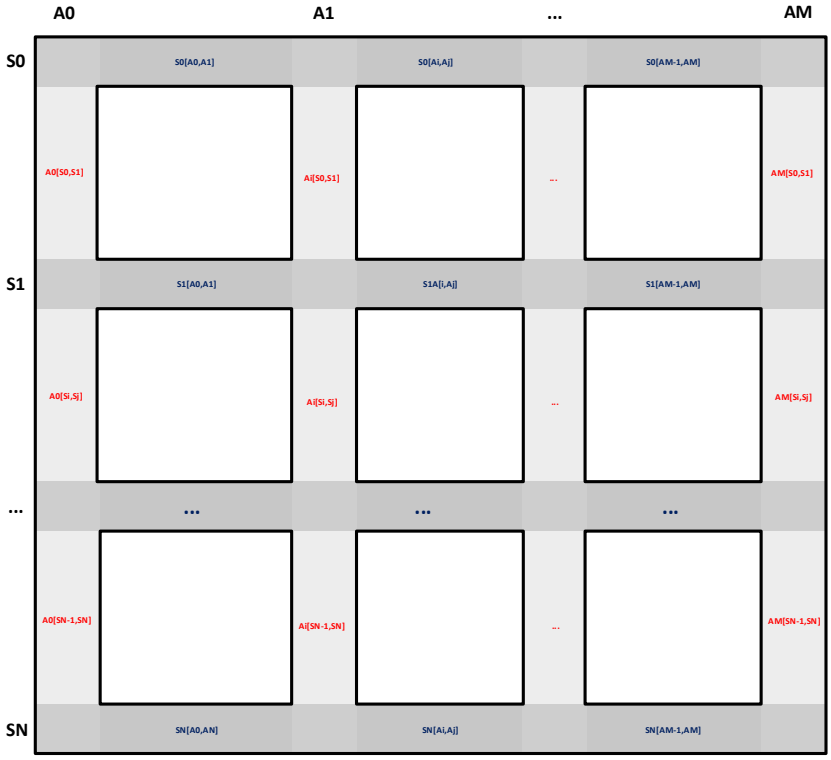


Figure 6.13: A neighborhood formed of N number of streets and M number of avenues.

2. final SPIDER structure.

The first phase of creating a proper SPIDER P2P overlay from across street neighborhood is to create a SPIDER overlay from the crossroads as shown in Figure 6.14. In the presented case scenario (N horizontal and M vertical streets), the intermediate SPIDER overlay is formed from N chains and M rings, the same as the number of streets and avenues. On this overlay are identified the street sectors obtained. In this construction, the intersections of the streets are considered the nodes.

Furthermore, in the second stage, each street sector identified on the intermediate overlay is translated to SPIDER Peer-to-Peer overlay nodes as shown in Figure 6.15. Taken into consideration the case scenario presented above, we give a generic method of computation of the parameters of the SPIDER overlay in Equation 6.2.

$$\begin{aligned} NrOfChains &= NrHorizoltalStreets - 1 + NrVerticalStreets \\ NrOfRings &= NrHorizoltalStreets \end{aligned} \tag{6.2}$$

$N - 1 + M$ chains and N rings form the resulted configuration.

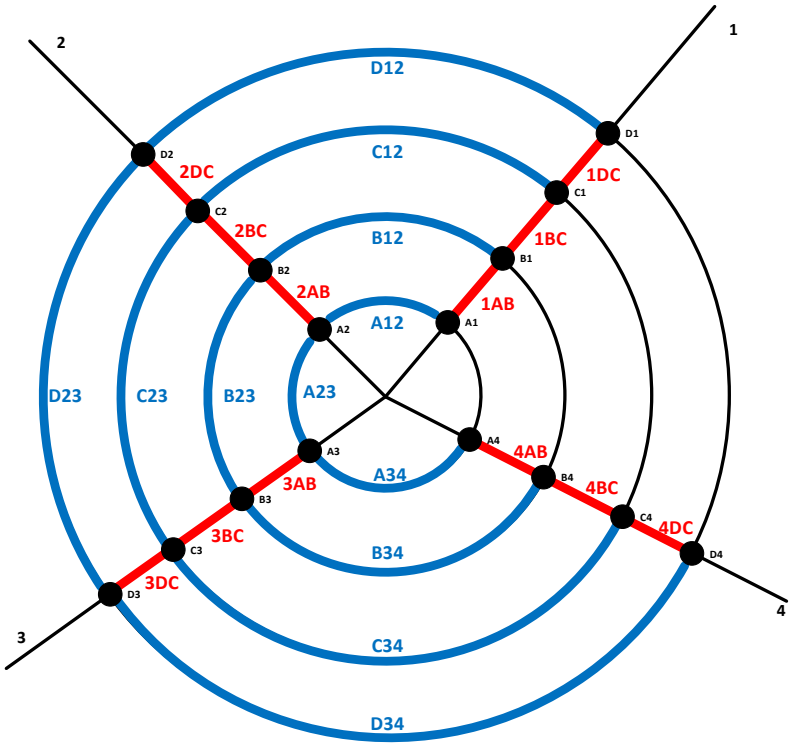


Figure 6.14: SPIDER Peer-to-Peer Overlay for the first stage for $N = 4$ streets and $M = 4$ avenues.

After the end of the first phase of the proposed method, we are taking into consideration the streets as nodes in the SPIDER Overlay.

Upon the finalization of the method's second phase, we obtain a pure SPIDER Peer-to-Peer overlay that maps the neighborhood that can be used for data fusion applications in smart-cities.

The proposed method takes into consideration the advantage of constructing the SPIDER overlay on rings as demonstrated in Section 3.2.

Furthermore, in the flowing section we evaluate the proposed method by mapping real cities in the USA and Romania to SPIDER overlays.

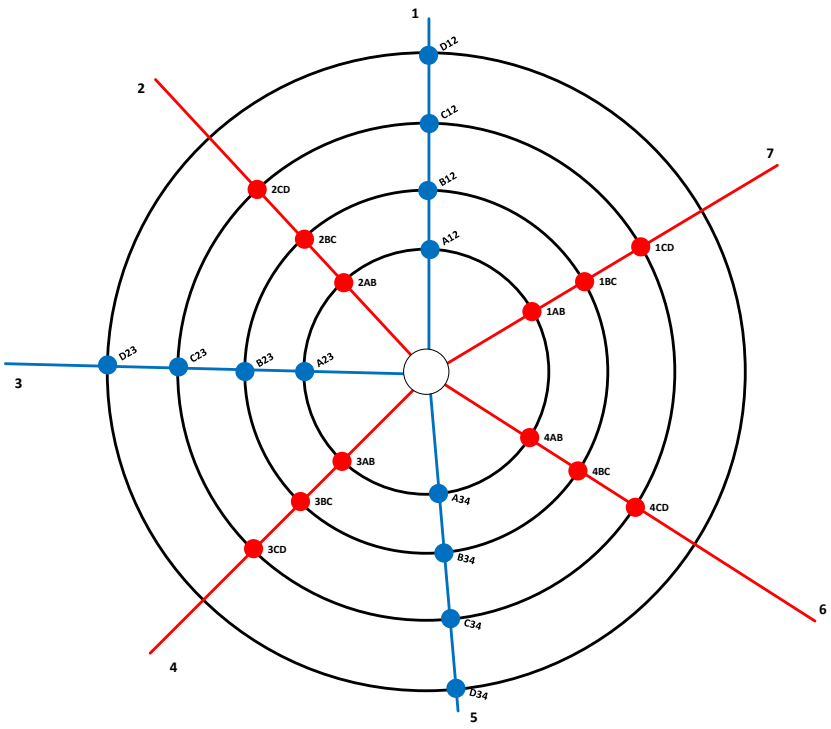


Figure 6.15: SPIDER Peer-to-Peer Overlay for second stage for for $N = 4$ streets and $M = 4$ avenues.

Experimental Evaluation of the Smart-streets Concept

In [16], the authors define the streets orientation entropy H_o as Equation 6.3.

$$H_o = \sum_{i=1}^{n} P(O_i) \log_e P(O_j) \tag{6.3}$$

where n represents the number of bins (the compass heading with an angle of 10°), $P(O_i)$ represents the proportion of the orientation of the fall in the i_{th} bin.

A city with maximum street orientation entropy is defined as a city with uniform streets distribution across all bins, while a city with minimum street orientation entropy is defined as a city where streets are distributed only on a single bin (north-south or east-west).

Taking into consideration the real world, the minimum street orientation is more plausible to be a grid city with all streets in four equal proportions (north-south and east-west).

To validate our data fusion methods and the transformation algorithm from cities with a low orientation-order indicator (a value obtained based on the street orientation entropy) to SPIDER Peer-to-Peer overlay, first we must settle the premises of the evaluation. Therefore, for the smart-streets use case, we are using the arrangement of the streets in 3 cities in USA, with the lowest street entropy and a city in Romania with also low street entropy:

- Chicago, USA downtown: 31 streets orientated from north to south and 50 streets orientated from east to west;
- Portland, USA downtown: 15 streets orientated from north to south and 16 streets orientated from east to west;
- New-York Manhattan, USA: 37 avenues orientated from north to south and 214 streets orientated from east to west;
- Calafat, Romania: 18 streets orientated from north to south and 6 streets orientated from east to west.

The SPIDER Peer-to-Peer overlay structures obtained from the experimental data sets are presented in Figure 6.16. As can be seen, the number of street sectors generated by the proposed method is around %50 higher than the number of chain and rings in the final SPIDER overlay. Also, we observe that the number of chains is higher than the number of rings. Furthermore, We observe that the number of street sectors varies with the number of vertical streets and it is presented in Table 6.6.

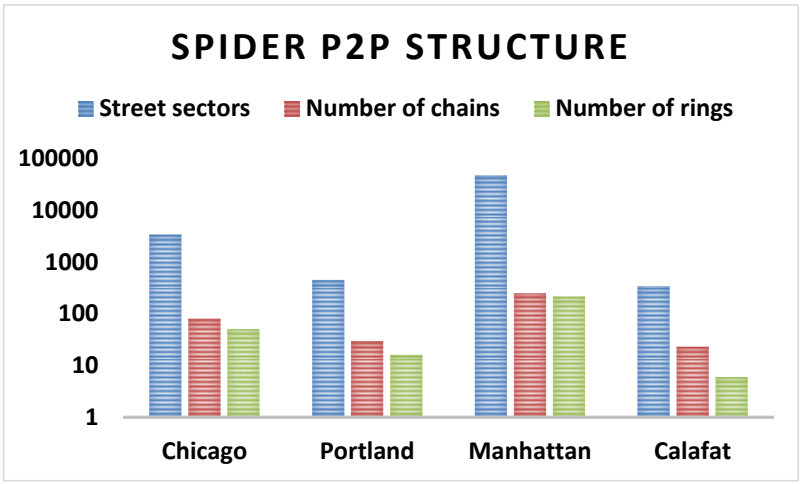


Figure 6.16: SPIDER overlay structure depending of the number of streets.

Table 6.6: Experimental values for the transformation algorithm from cities with low street orientation entropy.

	Number of street sectors	Number of chains	Number of rings
Chicago USA	3380	80	50
Portland USA	450	30	16
Manhattan USA	46914	250	214
Calafat RO	336	23	6

Also, the preliminary values show that the SPIDER overlay will have $NoOfStreets - 1$ chains. Another important conclusion drawn from the experimental results is that the number of rings in the SPIDER overlay is smaller than the number of chains.

We proposed the concept of smart-streets as part of smart-cities. We also developed an innovative algorithm for the smart-streets that manages a city with low street entropy as a SPIDER Peer-to-Peer overlay. We emphasized the adaptiveness properties of the SPIDER Peer-to-Peer algorithm along with its and fusion properties. We demonstrated that in the smart-streets use case the ring-based data fusion is more efficient that chain-based data fusion.

The proposed SPIDER mapping method can be used in every city with low street entropy. We chose the city of Calafat from Romania because to the best of our knowledge the streets entropy, in this case, is the smallest in Romania.

Furthermore, we evaluated the data fusion methods proposed in Section 5.1 with the values obtained from the SPIDER mapping method. The results are presented in Figure 6.17 and show that the data fusion algorithm on rings is more efficient than the data fusion algorithm on chains in case of large number of streets like New-York. In the case of small cities, like Calafat, the best data fusion algorithm is the one based on chains.

6.3.2 IoT Support for Smart-homes

The authors of [88] present a good overview of the main domains of IoT. Therefore, the most important Internet of Things domain is Smart-city witch encompasses the Smart-home, Smart-production, Smart-transportation and

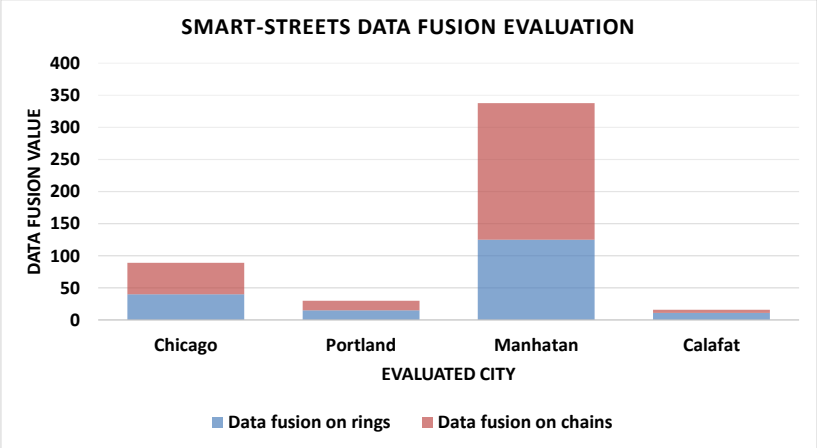


Figure 6.17: SPIDER overlay structure depending of the number of streets.

Smart-Energy approaches.

According to The Deloitte Consumer Review [1], in Germany, in 2018 more than 1 million households were considered smart.

Taking into consideration the high interest of the research communities in the Internet of Things, we present a use case for Smart-homes based on the SPIDER overlay.

In this use case, we take into consideration an autonomous house that has installed in its rooms several different sensors and actuators. Sensors record a physical parameter in an analogical manner and converts it to a digital value that is sent to other devices in a telegram form. Actuators are devices that receive the telegrams sent by the sensors and execute different operations based on the received parameters.

The main challenge of this use case is the aggregation of data generated by all sensors in order to offer valuable information for the actuators, in order to make an autonomous house. Another important aspect we take into consideration is the fault tolerance of such a system because sensors are prone to malfunction and the system must keep its accuracy in order to deliver correct information.

Furthermore, we show in Figure 6.18 the underlay of the Smart-house use case.

Therefore, the number of nodes in the overlay is given by the total number of

[1]https://www2.deloitte.com/content/dam/Deloitte/uk/Documents/consumer-business/deloitte-uk-consumer-review-16.pdf, Accesed: 03.06.2019

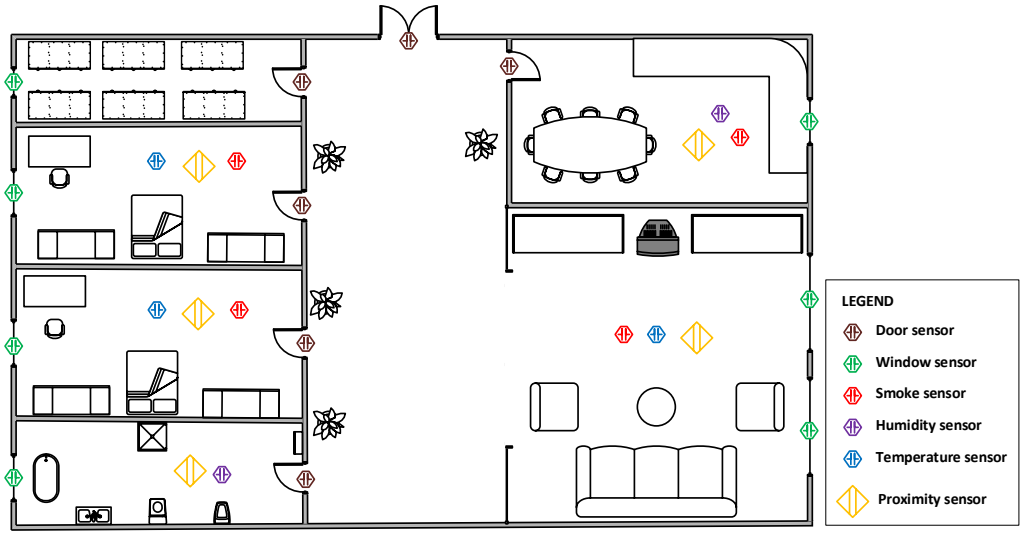


Figure 6.18: IoT support use case.

sensors in the house. In this particular case, the nodes are organized in a SPIDER overlay of nr rings and c chains. The number of chains is given by the number of chambers in the overlay, and the number of chains is provided by the maximum number of sensors in a room.

The formed SPIDER overlay is presented in Figure 6.19.

After the creation of the SPIDER overlay from the autonomous house described before, we obtain a logical structure that can be used for data aggregation from the same type of sensors, because each chain in the overlay represents a type of senor. Using this approach we use the data fusion algorithm on rings.

Furthermore, the obtained SPIDER overlay has good fault tolerance properties because the most common flaws that might happen in such systems are the local flaws because they appear as a result of the malfunction of one or several sensors. These flaws appear when the sensors have no energy left in their batteries. In this case, the SPIDER Peer-to-Peer overlay flaws, which matches this use case, are the one random node fall or two nodes fall. This type of defects can be frequent, and the recovery from this state is realized by replacing the broken sensors with new ones.

Another flaw that might appear in this use case is the entire ring fall. In this case, it might happen all the sensors in a room falls at the same time, due to when due to a power supply surge in a certain room.

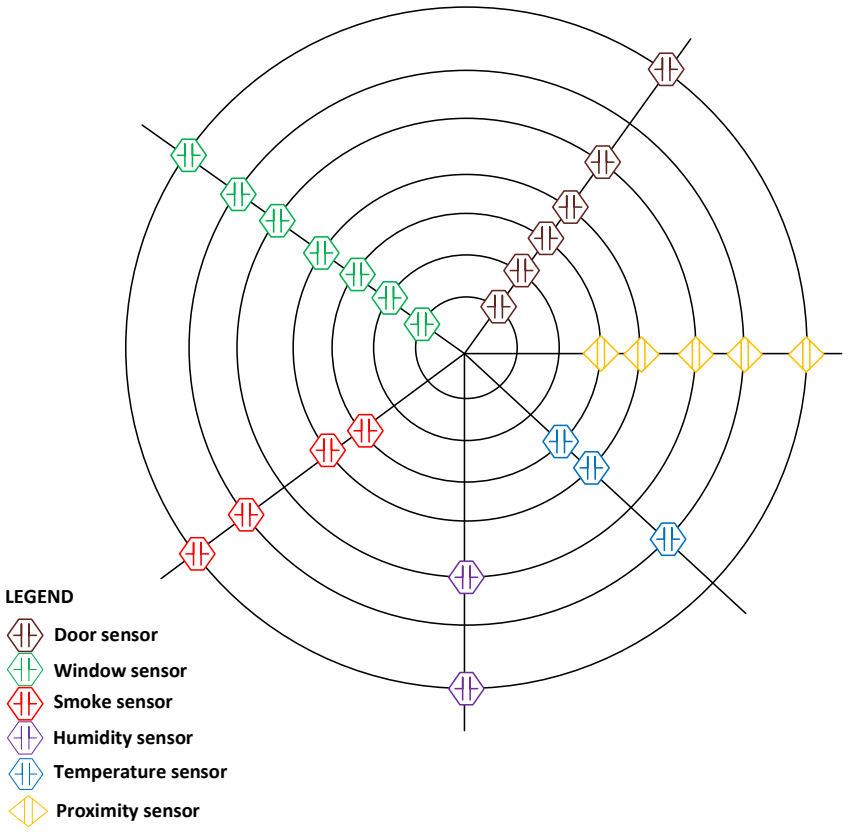


Figure 6.19: SPIDER Overlay for the IoT support use case formed by 6 chains and 7 rings.

Experimental Evaluation of Smart-homes

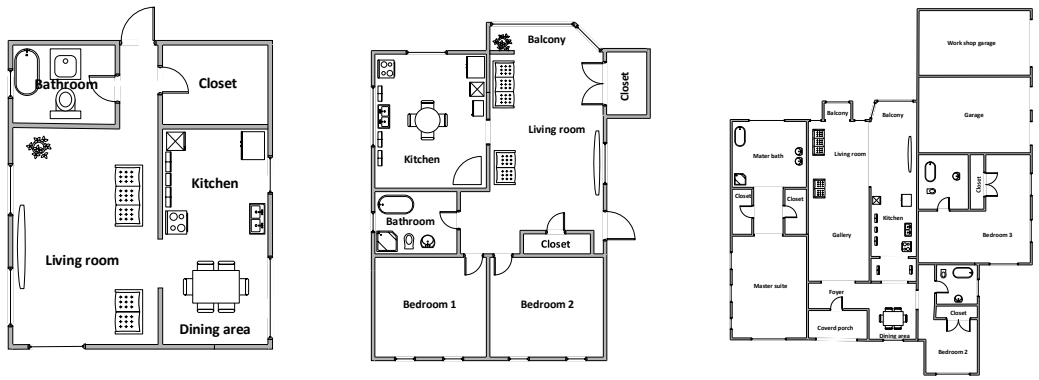
For the evaluation of the IoT support application for Smart-Home we take into consideration the realistic dataset Smart Home Device Scheduling [1] (SHDS) presented in [74]. Therefore, we consider three house dimensions (small (6.20a), medium (6.20b) and large (6.20c)) as shown in Figure 6.20.

In Table 6.7, we present the structural parameters of each house in the SHDS dataset. The height of the walls are standardized, thus the height of each house is 2.4 meters.

Furthermore, we present the smart devices used in the SHDS dataset. Thus, we consider:

- sensors: for testing purposes, we assume that all sensors are permanently

[1] https://github.com/nandofioretto/SHDS_dataset, Accesed: 03.05.2019



(a) (b) (c)

Figure 6.20: Small house floor plan (a), medium house floor plan (b) and large house floor plan (c).

Table 6.7: Structural parameters of each house type in the SHDS dataset.

Structural parameters	**Small house**	**Medium house**	**Large house**
House size (m^2)	6 x 8	8 x 12	12 x 15
Wall areas (m^2)	67.2	96	129.6
Window areas (m^2)	7.2	10	16

active. The sensors used for this evaluation are presented in Table 6.8;

- actuators: The actuators used for this evaluation are presented in Table 6.9.

In the SHDS dataset, each sensor is considered to be always active, meaning that it is not controlled by an agent. Every actuator in the dataset is strictly connected to one or multiple sensors. For the evaluation of the SPIDER Peer-to-Peer data fusion methods, we define for each sensor and actuator their location in the house. Based on their location, we create the overlay structure that is used in the evaluation.

Taking into consideration the sensors and actuators mentioned above, we map 3 houses to the SPIDER Peer-to-Peer overlay using the method described in Section 6.3.2. Therefore, we obtain 3 different SPIDER overlays with their specific parameters as shown in Table 6.10.

We evaluate the performance of the SPIDER overlay for different WiFi protocols: ZigBee, Bluetooth, ZWave, Wavenis and BLE. In Figure 6.21 we

Table 6.8: List of sensors used in SHDS dataset.

Sesor	Location
Thermostat-heat	Room
Thermostat-cool	Room
Water-heat	Water tank
Dust sensor	Room
iRobot 651 battery	Roomba 880
Tesla S battery	Tesla S
GEWSM2420D3WW washing machine	GE WSM2420D3WW
GEWSM2420D3WW drier	GE WSM2420D3WW
Kenmore 790 sensor	Kenmore 790.91312013
Kenmore 665 sensor	Kenmore 665.13242K900
Air humidity	Room
Luminosity	Room
Occupancy	Room
Movement	Room
Smoke detector	Room

present the experimental results for the presented case study of the SPIDER Peer-to-Peer implementation in the Smart-home scenario. It is obvious that the implementation of the SPIDER Peer-to-Peer overlay reduces the throughput for every WiFi protocol with more than 60% in comparison with a system that does not implement the SPIDER Peer-to-Peer overlay.

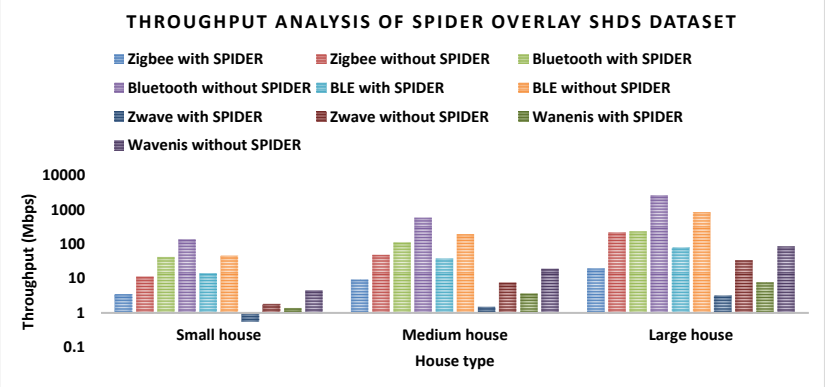


Figure 6.21: Throughput analysis of SPIDER Peer-to-Peer overlay impact in case of the IoT Smart-home case study based on the SHDS dataset.

Furthermore, we evaluated the data fusion methods proposed in Section 5.1 using the real scenario described in the smart-house use case. The results are presented in Figure 6.28. The obtained results show that the data fusion algorithm on rings is more efficient than the data fusion algorithm on chains.

Table 6.9: List of actuators used in SHDS dataset.

Actuator	Model	Actions	Location
Heater	Dyson AM09	off fan heat	Room
AC	BRYANT 697cn030b	off fan cool	Room
Watter-heater	Tempra 36	off on	Water tank
Vacuum Bot	iRobot Roomba 880	off vacuum charge	Room
Electric Vehicle	Tesla Modes S	off 48 A wall charger 72 A wall charger supercharger	Garage
Clothes Washer	GE WSM2420D3WW	off wash (Regular) spin (Regular) rinse (Regular) wash (Perm-Press) spin (Perm-Press) rinse (Perm-Press) wash (Delicates) spin (Delicates) rinse (Delicates)	Bathroom
Oven	Kenmore 790.91312013	off bake broil	Kitchen
Clothes Dryer	GE WSM2420D3WW	off on (Regular) on (Perm-Press) on (Timed)	Bathroom
Dishwasher	Kenmore 665.13242K900	off wash rinse dry	Kitchen

In terms of robustness, we evaluate the fault tolerance algorithms proposed in Section 4.1. The result for the local flaws fault tolerance algorithms are presented in Figure 6.23a. As it can be seen, the number of operations for fault recovery are constant. The experimental results obtained for the evaluation of the global

Table 6.10: SPIDER Peer-to-Peer overlay parameters for the case study based on the SHDS dataset.

House type	No. of nodes	No. of chains	No. of rings
Small house	21	12	4
Medium house	43	33	6
Large house	82	66	13

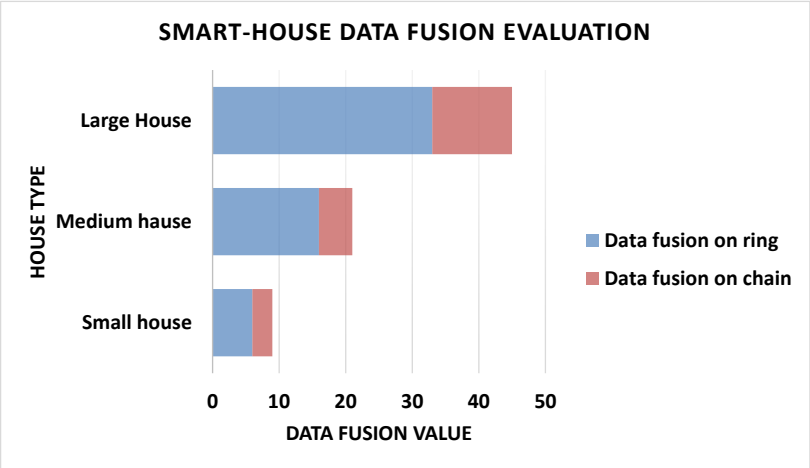
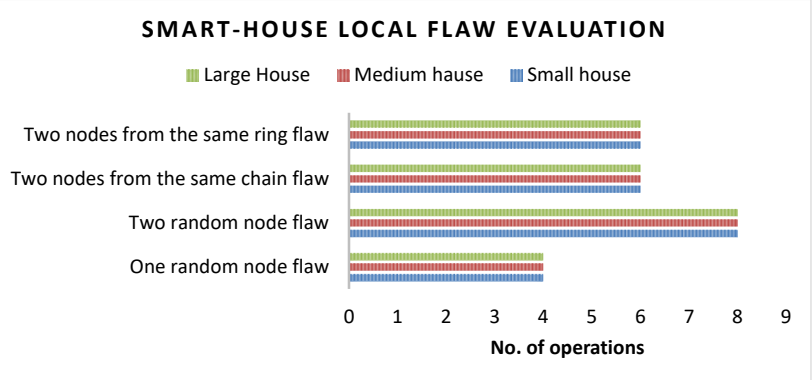


Figure 6.22: Data fusion evaluation in Smart-house use case using SHDS dataset.

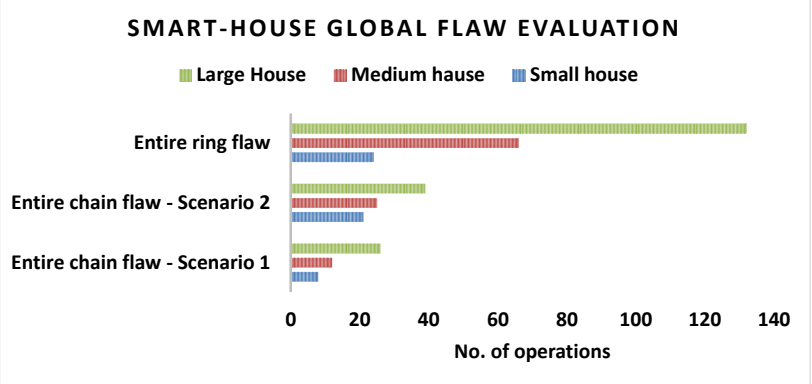
flaw recovery algorithms are presented in Figure 6.23b. As it can be observed, the recovery from the entire ring flaw is the most inefficient one, because the structure of the SPIDER overlay has many chains. In this case, the most efficient global flaw tolerance algorithm is the entire chain flaw - scenario 1.

In this section, we presented a realistic application for smart-homes based on the SPIDER Peer-to-Peer overlay and evaluate its robustness and auto-adaptiveness with the SDHS dataset for smart-homes. Also, we performed a performance evaluation in terms of throughput of these applications and the results show that the SPIDER Peer-to-Peer approach is better than standard IoT applications based on client-server approach.

Also, in this section, we evaluated the data fusion and fault tolerance properties if the SPIDER Peer-to-Peer overlay using the SHDS dataset.



(a)



(b)

Figure 6.23: Fault tolerance algorithms evaluation using the SDHS data set. Local flaws (a), Global flaws (b).

6.3.3 IoT Support for Smart-recycling

According to Eurostat, the total amount of waste generated by Romania in 2011-2012, was 266.776 tonnes. From this amount of waste, only 6-7% is recycled. According to the EU legislation, Romania will have to recycle half of the produced waste.

In order to improve the recycling process for citizens and to increase awareness regarding recycling, it is critical that the recycling process benefits from the advantages of the latest technologies ([50]). Therefore, interconnected sensors can be used to collect information from the recycling bins and deliver them in an aggregated form to end users via mobile applications.

In this scenario, we assume that there are four categories of waste, all placed in a recycling spot around a neighborhood:

- paper and cardboard;
- metal and plastic;
- glass;
- general waste.

To increase efficiency in the recycling process, people must first be aware of the position of the recycling spots relative to their house. Therefore, citizens can go to the nearest recycling spot.

Another requirement for such a system is waste level measuring in the bins. Thus, citizens are aware that a certain type of bin is full and can go to another recycling spot without wasting any time.

The architecture of the recycling system is presented in Figure 6.24. It includes sensors for measuring the level of waste in the recycling bins. The data collected from the sensors is transfered to decision making factors via a Cloud-service.

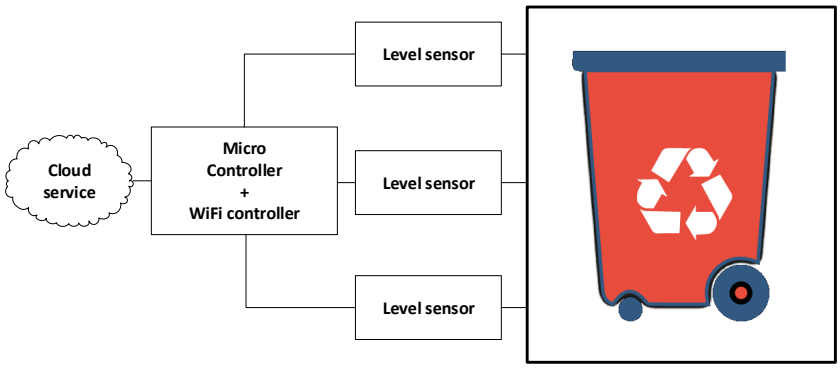


Figure 6.24: Recycling bin use case architecture.

The hardware configurations for a recycling spot is composed by the following components:

- Micro controller: ARM LPC2148 a 32 bit architecture CPU;
- IR sensor: TSOP 1738. This sensor indicates the level of garbage in the bin. In each bin are placed 3 IR sensors in order to indicate the actual garbage level ;

- Wifi module: 802.11b/g/n standard.

In Figure 6.25, we show the location of the recycling bins spots placed in a small neighborhood. We assume that there are placed $NoBins$ recycling bin spots. The bins are interconnected to provide aggregated information about the waste level in each bin to the end users.

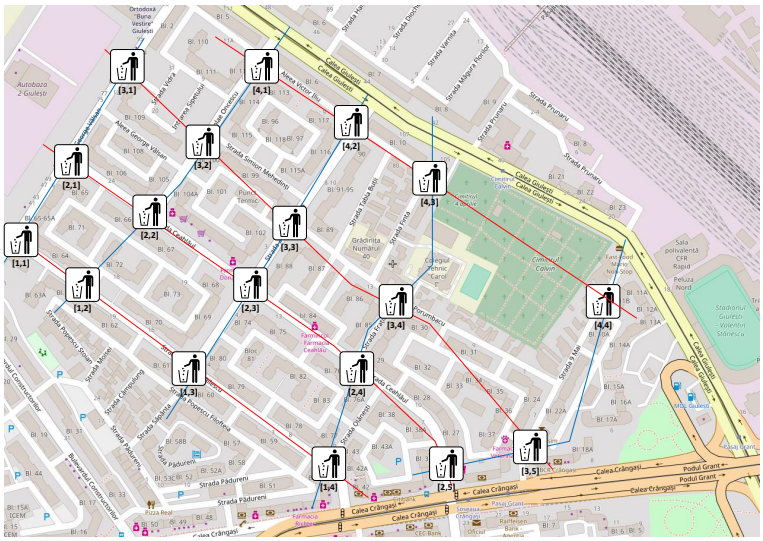


Figure 6.25: Recycling bin spots in a neighborhood underlay, where $No.Bins = 18$.

Based on the location of the recycling spots, the logical structure of the presented underlay is a SPIDER overlay formed by Nc chains given by the red streets in Figure 6.25 and Nr rings given by the number of blue streets in Figure 6.25. As expected the number of bin spots does not cover the entire overlay nodes. Therefore an extension of the overlay can be easily added. We present the overlay in Figure 6.26.

The most frequent flaw that might happen in the recycling IoT scenarios is the failure of a recycling spot. This failure is translated one random node failure. Even though, the most common failure is the one random node flaw other flaws presented in this thesis might appear.

Experimental Evaluation of Smart-recycling

For the evaluation of the IoT support application for recycling, we take into consideration the localization of the recycling bins in the Crângași neighborhood in Bucharest as shown in Figure 6.25.

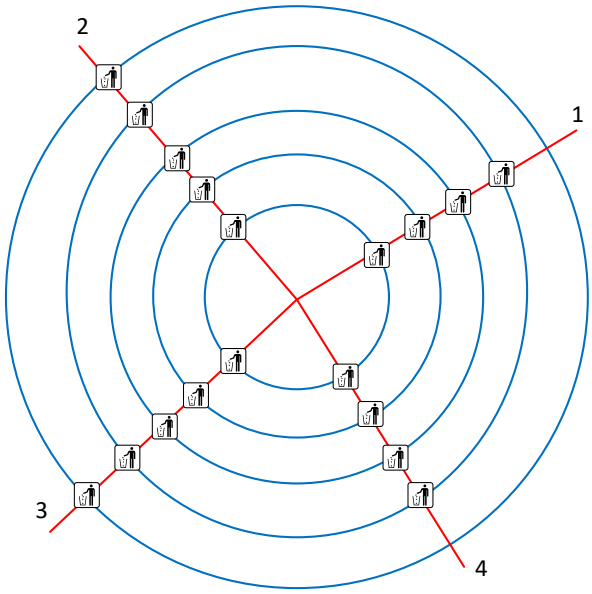


Figure 6.26: SPIDER overlay for the recycling bin scenario with $NoBins = 18$ nodes organized in $Nc = 4$ chains and $Nr = 5$ rings.

In Figure 6.27, we present the experimental results obtained. As it can be seen the throughput is with 30% lower in case of the SPIDER Peer-to-Peer overlay.

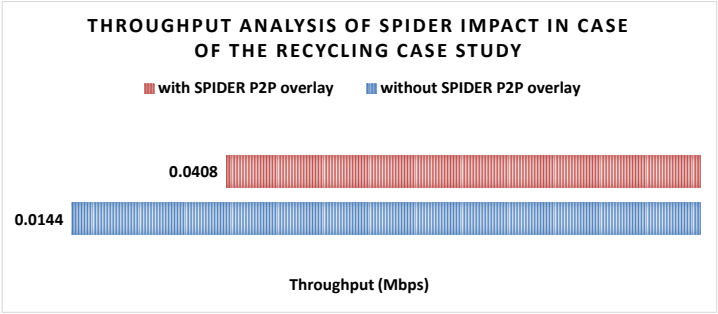


Figure 6.27: Experimental results for the IoT recycling use case.

Furthermore, we evaluated the data fusion methods proposed in Section 5.1 using the real scenario described in the smart-recycling use case. The results are presented in Figure 6.28 demonstrate that data fusion on ring algorithm is more efficient that data fusion on chain algorithm.

In terms of fault tolerance we evaluate the algorithms proposed in Section 4.1. The obtained results for the Smart-recycling use case are shown in Figure 6.29. We show that in terms of local flaw the most efficient algorithm for flaw recovery

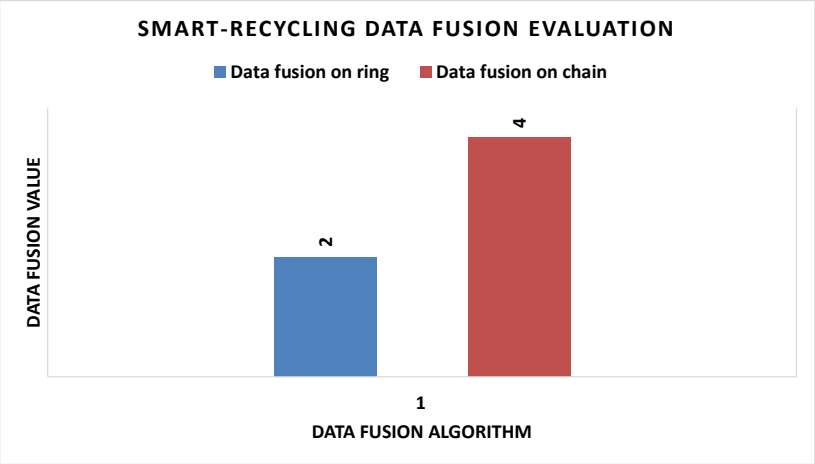


Figure 6.28: Data fusion evaluation in Smart-house use case.

is the one random node flaw, while the most efficient algorithm for global flaw recovery is the entire ring flaw. As demonstrated, when the structure of the SPIDER overlay has fewer chains than rings, the flaw recovery in case of entire ring failure is more efficient.

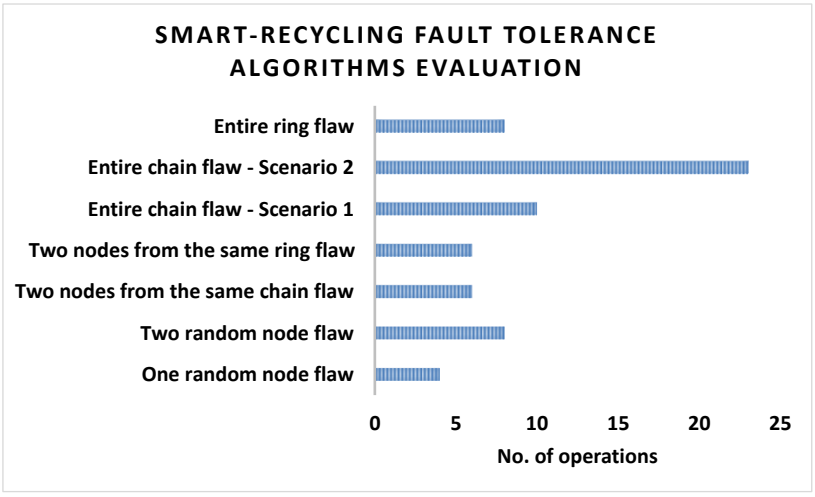


Figure 6.29: Fault tolerance evaluation in Smart-recycling use case.

The experimental evaluation of the local flaws fault tolerance algorithms is presented in Figure 6.23a. As can be seen the number of operations for fault recovery are constant. The experimental result obtained for the evaluation of the global flaw recovery algorithms are presented in Figure 6.23b. As it can be

observed, the recovery from the entire ring flaw is the most inefficient one, because the structure of the SPIDER overlay has many chains. In this case, the most efficient global flaw tolerance algorithms is the entire chain flaw - scenario 1.

In this section we presented a realistic application for recycling based on the SPIDER Peer-to-Peer overlay and evaluate its robustness with the real recycling infrastructure in the neighborhood of Crângași in Bucharest. The performance evaluation in terms of throughput of these applications showed that the SPIDER Peer-to-Peer approach is better that client-server applications based. Also, the recycling application based on SPIDER Peer-to-Peer overlay benefits from the auto-adaptiveness and robustness of the SPIDER overlay.

6.4 Trust Management System for E-commerce in Peer-to-Peer Networks

Besides the auto-adaptability ([61, 21, 3]), fault tolerance ([99, 30, 101]) and data fusion ([128, 124, 31]) the authors of [27] propose the trust management a fundamental characteristic of Peer-to-Peer systems.

Because the nodes inside a Peer-to-Peer system can join the network and leave at any moment without announcing their exit, the structure on which the system is built has to be able to respond quickly and efficiently to these changes [103].

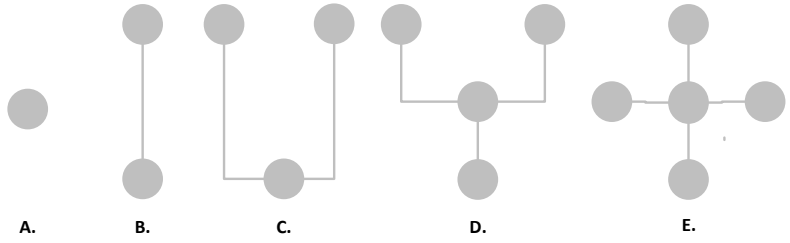


Figure 6.30: Nodes connection types in a SPIDER Peer-to-Peer overlay during the construction phase.

In the SPIDER Peer-to-Peer overlay the maximum number of neighbors of a node is four in comparison with HoneyComb overlay, where the maximum number of neighbors is three. Even though, a more significant number of neighbors would seem to be a better choice because a node has more direct contacts with other

nodes, it can reduce the efficiency of the system. This happens because of the dynamics of the network. If nodes change regularly their availability, the structure needs to be updated. The smaller the number of nodes that are influenced, the less time is required to rebuild the system. In a SPIDER Peer-to-Peer overlay construction phase, a node may have from 0 to 4 neighbors (see Figure 6.30 (E)).

The SPIDER structure has been chosen because each peer has a maximum of 4 direct neighbors. The nodes can be placed so that they form chains that are connected through links. This aspect means that a peer has two neighbors on its chain and two neighbors on an adjacent chain. Depending on its position, a peer can be linked to an exterior chain or an interior one.

6.4.1 Trust Model of Nodes in the SPIDER Peer-to-Peer Overlay

In this section, we present a trust management system based on the SPIDER Peer-to-Peer overlay (6.31). In order to be more accurate, we use the following notations for the formal description of the methods:

- $S[nc, nr]$ - SPIDER overlay definition;
- $n_{i,j} = node[c_i, r_j] = n * (r_j - 1) + c_i$ - node n_{ij} in the SPIDER overlay;
- $chain(n_{i,j})$ - chain of node n_{ij};
- $ring(n_{i,j})$ - ring of node n_{ij};
- $(TV_{n_{i,j}})^{n_{m,n}}$ - trust value of node $n_{i,j}$ stored in node $n_{m,n}$, where $n_{m,n}$ is a neighbor of $n_{i,j}$;
- $T(X, Y)$ - trust value of the link between two nodes X and Y.

Each node in the SPIDER Peer-to-Peer overlay store set of maximum five values for the trust. The first value is its own trust value, and the rest four values represent the trust values of its neighbors.

- $(TV_{n_{i,j}})^{n_{i,j}}$ - trust value of node $n_{i,j}$ stored in node $n_{i,j}$;
- $(TV_{n_{i+1,j}})^{n_{i,j}}$ - trust value of node $n_{i+1,j}$ stored in node $n_{i,j}$;
- $(TV_{n_{i-1,j}})^{n_{i,j}}$ - trust value of node $n_{i-1,j}$ stored in node $n_{i,j}$;
- $(TV_{n_{i,j+1}})^{n_{i,j}}$ - trust value of node $n_{i,j+1}$ stored in node $n_{i,j}$;

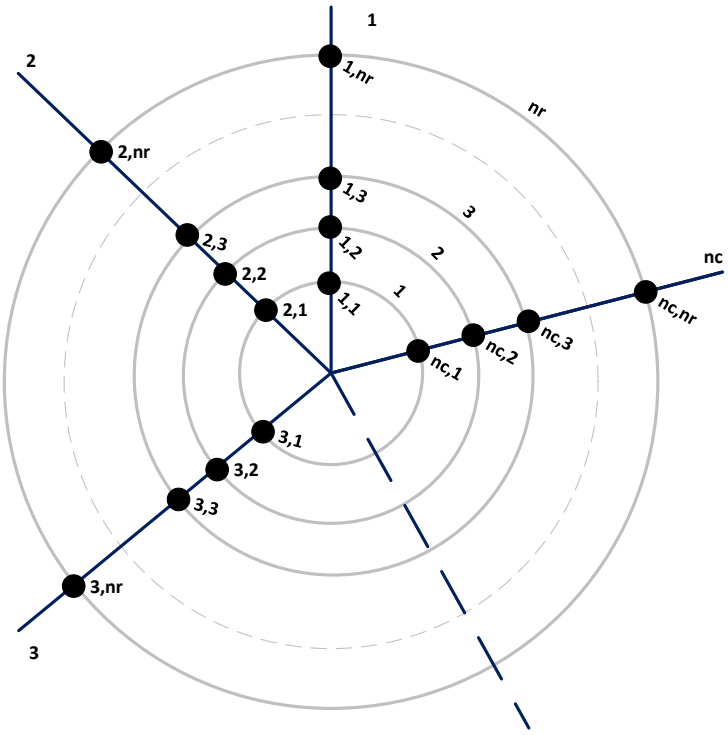


Figure 6.31: A SPIDER overlay with nr rings, nc chains and $totalNumberOfNodes = nr * nc$.

- $(TV_{n_{i,j-1}})^{n_{i,j}}$ - trust value of node $n_{i,j-1}$ stored in node $n_{i,j}$;

When a new node joins the system, it receives a set of coordinates (chain and ring). All messages that are sent through the system are routed based on these coordinates. This aspect increases the system's security because the only nodes knowing the identities of others are the direct neighbors However, it has a downside: nodes need to route messages that are not related to their requests. The total number of routed messages needs to be kept as low as possible so that nodes are not overloaded with messages that do not bring direct value. This issue requires employing a routing algorithm that offers the shortest path between 2 nodes when routing the messages.

The node that queries for a specific resource, rates each interaction in the process. We assume that the feedback is given based on validity of the received data correlated with the time interval in which the request is completed. If the message is corrupted in any way, the trust values of that transactions is the lowest possible

value. We compute the trust values by considering the response time and the number of hops between the nodes. If the distance is short, then the time interval in which the transaction is expected to take place is smaller. If the range is vast, then the time interval automatically increases in strict dependence with the number of hops. If the node sends the needed resource in shorter or equal time to the minimum computed time interval, then the trust value is the largest allowed. The trust values decreases with the amount of time passed over the calculated limit.

The trust value (TV) of a peer is given in [135], where the authors describe a solution for peer-to-peer trust management. The trust value is computed adaptively and is based on the feedback received for the last transaction, but it also takes into account all the feedback values previously received in Equation 6.4, where the $[TV]$ represents the number of instances of feedback received for the node. The feedback value is F, and $\{TV^{new}\}$ is the new trust value. The main downside of this algorithms is the fact that it does not handle very well traitor nodes. After behaving correctly in a specific time frame, they can make a large number of transactions before their trust value is reduced.

$$\begin{aligned} [TV]^{new} &= [TV]^{old} + 1 \\ \{TV^{new}\} &= \frac{[TV^{old}]\{TV\}^{old} + F}{[TV^{new}]} \end{aligned} \tag{6.4}$$

6.4.2 Trust Model of Links in the SPIDER Peer-to-Peer Overlay

We propose a new model to manage the trust between two nodes via a link that does not depend on the number of transactions that takes place for a specific node based on [135]. It also makes all the feedback values previously received, the current feedback value and the trust value of the neighbor node that rated

the transaction into account as shown in Equation 6.5.

$$
\begin{aligned}
TV_{mean} &= \frac{\sum_i TV_i}{NoOfNeighbors + 1} \\
T^{new}(X,Y) &= \alpha T^{old}(X,Y) + (1-\alpha)[TV_{mean}F_X + (1 - TV_{mean})T^{old}(X,Y)], \\
T^{new}(Y,X) &= \beta T^{old}(Y,X) + (1-\beta)[TV_{mean}F_Y + (1 - TV_{mean})T^{old}(Y,X)],
\end{aligned} \tag{6.5}
$$

where:

- TV_{mean} - mean trust value of a node and its neighbors trust values;
- $T(X,Y)$ is the trust value of the link (X,Y);
- T_X and T_Y are the trust values of the neighbor nodes that gave the feedback; for example X gave the feedback F_Y to Y and Y gave the feedback F_X to X, X and Y are direct connected;
- F_X and F_Y are the feedback values.

All trust values are fixed in the interval $[0,1]$. We compute the feedback value, using the same concept described in the previous section, by taking into consideration the validity of the data, and the time it took to handle the request. If a message is corrupt (the checksum is wrong), then the feedback value will be 0. Otherwise, the value will be computed based on the time it taken from the moment the initial request for the resource was made until the response that the node can share it was received. The general model is adaptive with $0 \leq \alpha, \beta \leq 1$. We consider that $\beta = \alpha$ (the trust values for links are computed with the same combination ratio). In this model, we consider only one old value, but an extended model may take into consideration the mathematical properties of unitary processes [117] and of non-commutative Markov processes [116].

6.4.3 Experimental evaluation of the Trust Management Algorithms

In order to evaluate the proposed methods for trust management using the SPIDER Peer-to-Peer overlay we used a real dataset presented in [111] of touristic reviews for 10 categories of destinations in East Asia. Each user rated

the destination as Excellent (4), Very good (3), Average (2), Poor (1) and Terrible (0). The dataset is populated by crawling the TripAdvisor[1] website. The ratting were given fo the following attributes:

- art galleries [01];
- dance clubs [02];
- juice bars [03];
- restaurants [04];
- museums [05];
- resorts [06];
- parks/picnic spots [07];
- beaches [08]
- theaters [09];
- religious institutions [10].

The first operation that we must execute in order to use this dataset is to convert the feedback values form the [0,4] interval to the [0,1] interval, because our algorithm works with feedback values in the [0,1] range.

The experimental results, are split in two categories: trust on nodes and adaptive trust on links.

Both methods for the trust management (for nodes and links) are shown below. We consider two main scenarios when a peer first connects to the system, and how a peer behaves after a period. We present trust values obtained for different transactions.

The first graph contains the trust values for the first 10 peers that are managed using the formula for node trust value computation. The best and worst case scenarios and the most frequent case are taken into consideration. In Figure 6.32 we present the new trust values of the peers after each new transaction takes place.

In Figure 6.32, the blue series shows the case of an ideal peer that received only the highest value for feedback, 1. After only four transactions, it can be considered

[1]http://tripadvisor.com/, Accessed: 24-01-2019

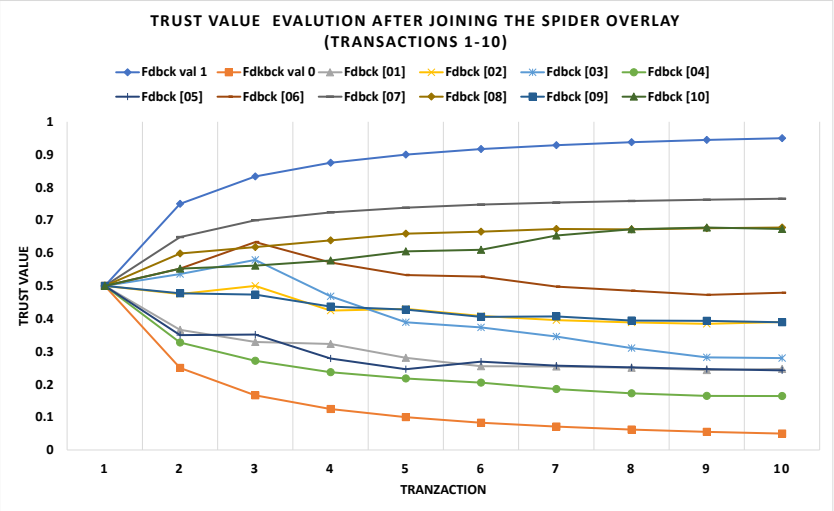


Figure 6.32: Node's trust evolution after joining the SPIDER overlay network.

a trusted peer, with his trust value reaching 0.875. The orange series shows a malicious peer whose feedback value is 0. After four transactions, no other peer will interact with him because his trust value will be 0.125. The remaining series contain the behavior of a correct peer. The proper feedback values for him range between $[0.83, 0.95]$.

The next four graphs (Figure 6.33, Figure 6.34, Figure 6.35 and Figure 6.36) show the behavior of a peer that has reached a trust value of 0.85. As in the previous case, the ideal peer is shown in blue and the malicious one in red. The first image contains the transactions from 20 to 29 and the one after it from 50 to 59. The third and forth images contains the transactions from 100-199 and 500-599. All images include the exact trust value. The transaction number gives the only difference.

It is noticeable that the transaction number has an enormous influence on the trust values. If a malicious peer behaves well for a specific time, it can then distribute malicious content for an extended period before his trust value is lowered enough to reflect its real behaviour. In the first case, after 10 transactions, its trust value will be 0.58, but in the last one it will reach 0.72, so he will still be seen as a trusted peer.

For this evaluation of the link trust, we consider multiple values for α. The first value considered is $\alpha = 7/8$. We chose this value because it is the default value for computing the RTT value in the TCP protocol [104].

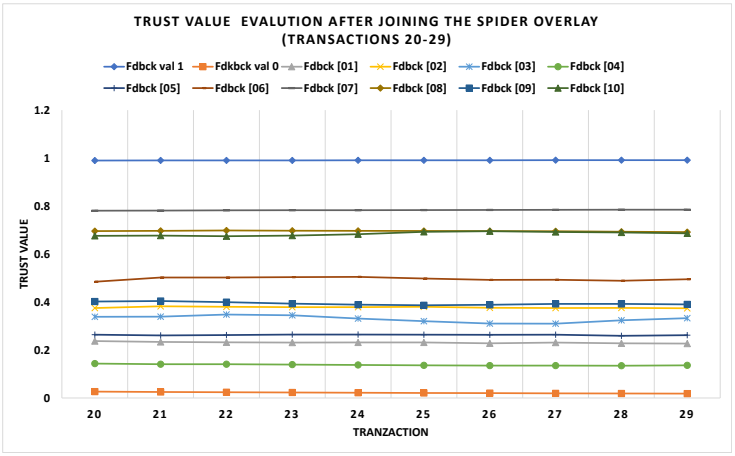


Figure 6.33: Node's trust: transactions 20-30.

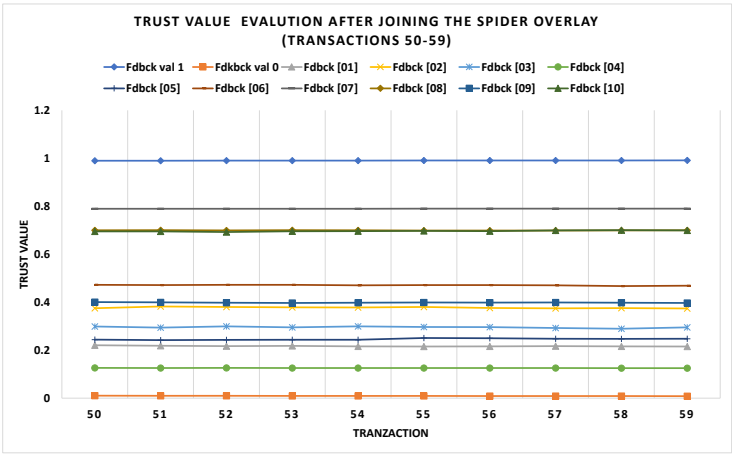


Figure 6.34: Node's trust: transactions 50-60.

Using the $\alpha = 7/8$ value, we influence the trust values the least. When a node first connects to the system, only after 7 transactions its value will be greater than 8. The $\alpha = 5/8$ value requires only 2 transactions for the trust value to reach 8. Finally, when we use $\alpha = 3/4$, the trust values modifies in 4 transactions.

The $\alpha = 3/4$ value seems to be the best choice for when a node first joins the system. The correct nodes can gain a good trust value in a relatively short time frame, and the malicious ones can be detected in an early stage. The $\alpha = 5/8$ value has not been chosen because, if a node receives false feedback, then its trust value is lowered too much. We consider the following case to gain a better perspective of how false feedback can influence a node. The trust values for the nodes that give the feedback are in the interval $[0.87, 0.95]$. The feedback values

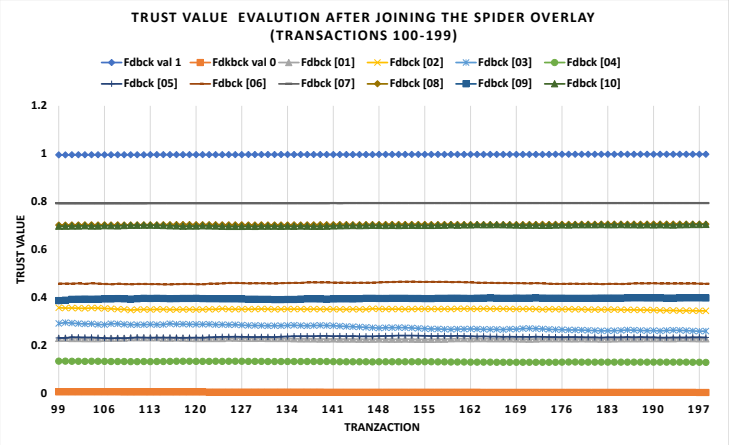


Figure 6.35: Node's trust: transactions 100-200.

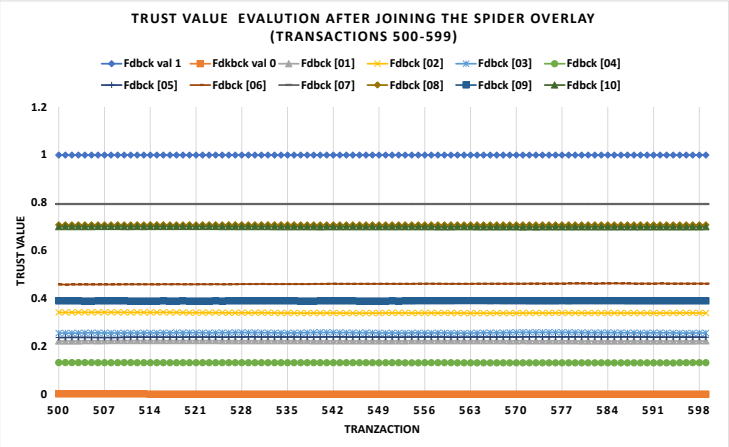


Figure 6.36: Node's trust: transactions 500-600.

range from $[0.83, 0.95]$. The initial trust of the node is 0.85. The false feedback is given for the 3rd transaction, and its value is 0.3 (see Figures 6.37, Figures 6.38 and Figures 6.39).

In this section we described and evaluate two algorithm for trust management in SPIDER based applications. The first algorithm is designed for the trust of individual nodes, while de second one deals with the trust value of the links between nodes.

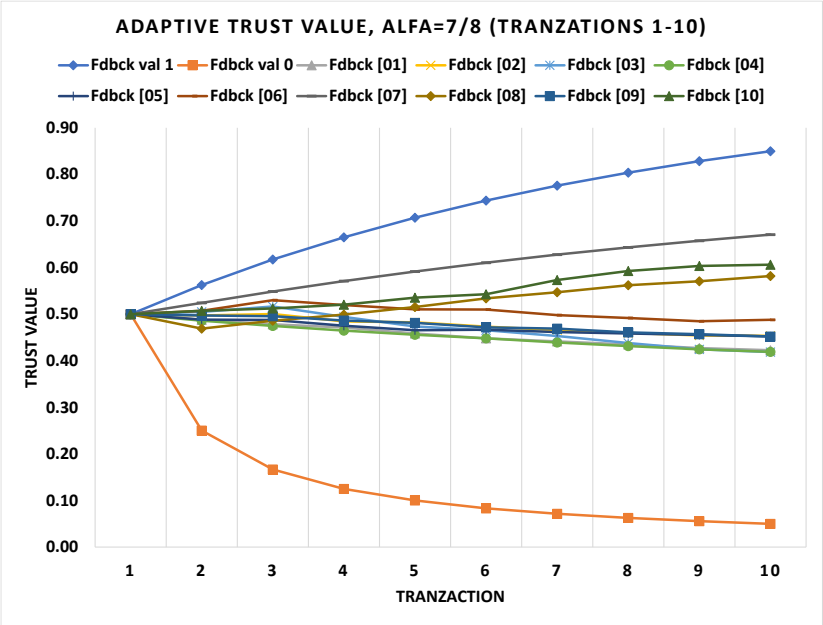


Figure 6.37: Adaptive trust: join network. $\alpha = 7/8$.

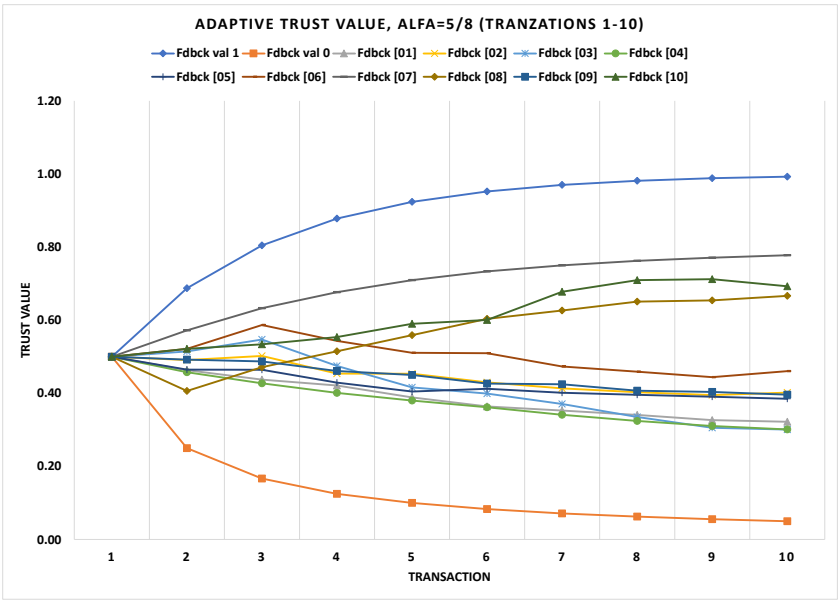


Figure 6.38: Adaptive trust: join network. $\alpha = 5/8$.

6.5 Discussions and Conclusions

In this chapter we offered a set of applications that benefit from the SPIDER Peer-to-Peer overlay: live video streaming applications for Cyber-physical

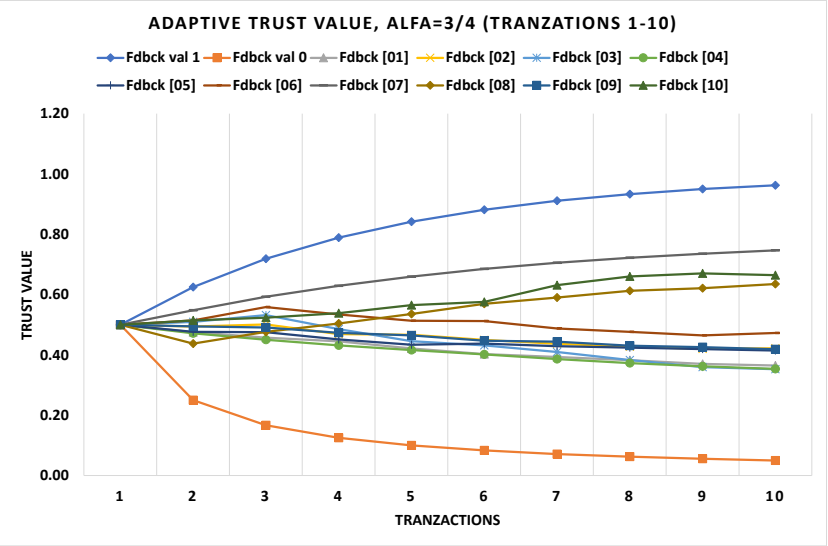


Figure 6.39: Adaptive trust: join network. $\alpha = 3/4$.

infrastructures, Smart-city applications and trust management system for E-Commerce in Peer-to-Peer networks.

The SPIDER Peer-to-Peer overlay applications for live video streaming in Cyber-physical infrastructures are based on real case scenarios that are part of the two projects developed in UPB:

- CLUeFARM: Information system based on cloud services, accessible through mobile devices, for quality improvement of products and business development in farms project developed in UPB;
- CyberWater: Prototype Cyber infrastructure-based System for Decision-Making Support in Water Resource Management project developed in UPB.

Furthermore, the SPIDER Peer-to-Peer overlay applications for Smart-cities are also part of an important project developed in UPB entitled DataWay: Real-time Data Processing Platform for Smart Cities: Making sense of Big Data.

Furthermore, in this chapter we provided a set of requirements for Peer-to-Peer applications: physical communication infrastructure, transport protocol, bandwidth, self-recovery from local faults, self-recovery from global faults and throughput.

Finally, we evaluated the proposed applications in terms of performance and presented the experimental result.

Chapter 7

Conclusions

In this thesis, we researched, developed and evaluated a dependable framework for the management of large-scale heterogeneous distributed systems as generic overlays that assures security and trust of processed data in Cyber-physical infrastructures, Smart-cities, Internet of Things and E-commerce systems.

We successfully developed an auto-adaptive overlay that scales over unsafe heterogeneous data sources in a fault-tolerant way with high reliability over unreliable physical resources. Also, we evaluated the proposed overlay with real datasets on various realistic uses cases such as live video streaming, smart-streets, IoT in Smart-homes and smart-recycling facilities, and trust management E-commerce applications.

To this end, we focused on different characteristics of the SPIDER Peer-to-Peer overlay from its design and mathematical properties to its operations. Also, we elaborated on the robustness of SPIDER, as the way it deals with minor and major failures. We also focused our research on data fusion, as the way that data is aggregated in a more efficient manner in order to be disseminated for better decision support systems. Finally, we handled trust management, as a way of evaluation of the trust of each node or link between nodes in the overlay.

To address each research question, we proposed algorithms and methods that respect the properties of the SPIDER Peer-to-Peer overlay. We demonstrated that our solutions lead to significant improvements in the performance of modern distributed systems based on the Peer-to-Peer approach.

In order to enforce the advantages of the SPIDER Peer-to-Peer overlay, we conducted a set of experimental investigations with realist datasets such as: the

vertical farming concept, the hydrological stations situated on the rivers in Romania, the streets in 3 major cities in the USA and one in Romania, the SDHS dataset for Smart-homes, the location of the recycling bins in a neighborhood in Bucharest city and a dataset from TripAdvisor for tourist attractions in Asia.

In this chapter, we present a summary of our original contributions that offer answers to the research questions outlined in the Section 1.1 (Problem Statement and Thesis Objectives), the lessons learned during the Ph.D. years, several future research directions and also the projects and publications that strengthen the value of this research.

7.1 Contributions

In this section, we outline the main conclusions that contribute to the answers of the research questions stated in Chapter 1 (Context and Goals) for the design and implementation of auto-adaptive Peer-to-Peer systems. These conclusions provide insights into the construction of trusted auto-adaptive Peer-to-Peer overlays that scales over heterogeneous data sources.

1. We designed an original auto-adaptive Peer-to-Peer overlay based on the spider web architecture. This contribution is presented in Chapter 3 (SPIDER: Auto-Adaptive Peer-to-Peer Overlay). We described the structure of the SPIDER overlay, it's construction and the most important operations (joining, naming, leaving and routing). The SPIDER overlay is a Peer-to-Peer structure that satisfies the following properties: reliability, availability, fault tolerance, trust, load balancing, and auto adaptability;

2. We provided a set of fault tolerance algorithms for local and global flaws suitable for the SPIDER Peer-to-Peer overlay and evaluated their features using real datasets. This contribution is presented in Section 4.1.1 (Local Flaws in SPIDER Peer-to-Peer Overlay) and Section 4.1.2 (Global Flaws in SPIDER Peer-to-Peer Overlay);

3. We presented a set of fault tolerance methods for permanent and temporary flaws suitable for the SPIDER Peer-to-Peer overlay and evaluated their features. This contribution is presented in Section 4.1.3 (Permanent Flaws in SPIDER Peer-to-Peer Overlay) and Section 4.1.4 (Temporary Flaws in

SPIDER Peer-to-Peer Overlay);

4. We offered a set of data fusion algorithms for the SPIDER overlay and evaluated them using real datasets. This contribution is described in Section 5.1 (Data Fusion Methods);

5. We proposed the concept of Smart-streets as a suitable implementation of the SPIDER peer-to-Peer overlay for data aggregation in large urban areas. We provided an original method that converts the streets of a city with low street entropy to a convenient SPIDER overlay network. This contribution is presented in Section 5.2 (Experimental Evaluation of the Data Fusion Algorithms) and Section 6.3.1 (Smart Streets Concept);

6. We bridged the requirements of Peer-to-Peer applications to the SPIDER overlay and presented a pack of solutions that can benefit from the usage of the SPIDER Peer-to-Peer overlay. This contribution is presented in Chapter 6 (SPIDER Peer-to-Peer Overlay Applications). Therefore, we assessed applications for live video streaming, for the Smart-homes and Smart-recycling. The proposed solutions were evaluated through real implementations in DataWay, CyberWater and ClueFarm projects developed in UPB with real datasets;

7. We proposed an effective set of methods for trust management in decision-making systems for E-Commerce. We evaluated booth node trust and link trust in SPIDER systems through realistic implementation with real datasets. This contribution is presented in Section 6.4 (Trust Management System for E-commerce in Peer-to-Peer Networks).

Throughout all the previously stated phases, we successfully accomplish the objectives of this thesis.

7.2 Lessons Learned

In the past 5 years I have been working on this thesis and I have learned a great amount and transitioned from a computer science student to an experienced researcher. In this section, I outline the most valuable lessons I have learned.

Lesson 1. Design a trusted auto-adaptive Peer-to-Peer overlay based on the natural phenomenon of a spider web. We took into consideration the advantage

of a spider web structure that can have a maximum of 4 neighbors and data is disseminated only between those nodes. This process reduces throughput and increases the performance.

Lesson 2. The fault recovery of SPIDER based systems in case of failures. We proposed several methods of fault recovery in case of real faults caused by local malfunction of certain nodes or major hazards. We also consider the temporary possibility of network loss such as 4G signal loss or cable unplug hazard.

Lesson 3. Data aggregation from several sources in order to offer accurate information regarding a specific problem to decision making factors in case of critical management systems such as river pollution or farming monitoring, traffic routing through crowded smart cities, smart-homes and smart recycling facilities.

Lesson 4. Implementation of SPIDER Peer-to-Peer overlay in smart-cities applications does not depend on the streets entropy. We proposed an algorithm and demonstrated that a city with low street entropy can be easily converted to a SPIDER overlay structure.

Lesson 5. Mapping the requirements of Peer-to-Peer applications with the properties of the SPIDER overlay and implementing them in real-life applications for live video streaming in Cyber-physical infrastructures, smart-cities and E-commerce systems.

7.3 Future Work

Even though, the achievements of this thesis leads one step towards solving critical issues regarding Peer-to-Peer systems via technological advances in "data-centered" networking and cloud computing, a significant amount remains on the interoperability of the SPIDER Peer-to-Peer overlay with other overlay structures.

In addition, a future research direction could be the design and implementation of other security features besides trust, such as, anonymity and a secure communication channel without the usage of a certification authority.

7.4 List of Publications

In this Section, we present the list of publications in various conferences, workshops, journals, and scientific books that strengthen the quality of this thesis.

- Mocanu, B. et al., 2015. SPIDER: A Bio-inspired Structured Peer-to-Peer Overlay for Data Dissemination. 2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC). doi:10.1109/3pgcic.2015.121. [Conference Paper, ISI proceedings];
- Mocanu, B.-C., F. Pop, C. Dobre, and V. Cristea. "Self-Adaptive Overlay Networks." Pervasive Computing (2016): 27–44. doi: 10.1016/b978-0-12-803663-1.00002-4. [Book Chapter, Elsevier];
- Pop, F. et al., 2015. Trust models for efficient communication in Mobile Cloud Computing and their applications to e-Commerce. Enterprise Information Systems, 10(9), pp.982–1000. doi: 10.1080/17517575.2015.1100756. [ISI Journal, IF = 1.683, Q3];
- Mocanu, B. et al., 2016. Cloud live streaming System based on Auto-adaptive Overlay for Cyber Physical Infrastructure. Proceedings of the Third International Workshop on Adaptive Resource Management and Scheduling for Cloud Computing - ARMS-CC'16. doi: 10.1145/2962564.2962571. [Conference Paper, ISI proceedings];
- Mocanu, B. et al., 2017. Flaw Recovery in Cloud Based Bio-inspired Peer-to-Peer Systems for Smart Cities. Lecture Notes in Computer Science, pp.338–352. doi: 10.1007/978-3-319-57186-7_26. [Conference Paper, ISI proceedings];
- Mocanu, B. et al., 2019. Data fusion technique in SPIDER Peer-to-Peer networks in smart cities for security enhancements. Information Sciences, 479, pp.607–621. doi: http://dx.doi.org/10.1016/j.ins.2018.06.070. [ISI Journal, IF = 4.378, Q1].

7.5 Projects

In this Section, we list the projects, that we brought our contributions to during the Ph.D. years.

- *Sectorial Operational Programme Human Resources Development 2007-2013 of the Ministry of European Funds through the Financial Agreement PROSCIENCE: Science and Research Quality Promotion through Ph.D. Scholarships POSDRU/187/1.5/S/155420* developed in UPB.
- *DataWay: Real-tim Data Processing Platform for Smart Cities: Making Sense of Big Data* research project developed in UPB and coordinated by prof. Florin Pop;
- *CLUeFARM: Information system Based on Cloud Services, Accessible through Mobile Devices, for Quality Improvement of Products and Business Development in Farms* research project developed in UPB and coordinated by prof. Valentin Cristea;
- *CyberWater: Prototype Cyber infrastructure-based System for Decision-Making Support in Water Resource Management* research project developed in UPB and managed by prof. Mariana Mocanu;

Bibliography

[1] Ovidiu-Marian Achim, Florin Pop, and Valentin Cristea. Reputation based selection for services in cloud environments. In *Network-Based Information Systems (NBiS), 2011 14th International Conference on*, pages 268–273, Sept 2011.

[2] Syed Hassan Ahmed and Shalli Rani. A hybrid approach, smart street use case and future aspects for internet of things in smart cities. *Future Generation Computer Systems*, 79:941–951, 2018.

[3] Raja Naeem Akram, Konstantinos Markantonakis, and Damien Sauveron. A novel consumer-centric card management architecture and potential security issues. *Information Sciences*, 321:150–161, 2015.

[4] Bahaa Aldeen Alghazawy and Satoshi Fujita. A scheme for maximal resource utilization in peer-to-peer live streaming. *arXiv preprint arXiv:1510.02138*, 2015.

[5] Muneeb Ali and Koen Langendoen. A case for peer-to-peer network overlays in sensor networks. In *International Workshop on Wireless Sensor Network Architecture (WWSNA'07)*, pages 56–61, 2007.

[6] Himaja Amidala, Karthik Shankar, and Tarik El Taeib. An efficient peer-to-peer platform for large scale data processing in network applications. In *2016 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*, pages 1–5. IEEE, 2016.

[7] Stephanos Androutsellis-Theotokis and Diomidis Spinellis. A survey of peer-to-peer content distribution technologies. *ACM computing surveys (CSUR)*, 36(4):335–371, 2004.

[8] Mohsen Attaran. The internet of things: Limitless opportunities for business and society. *Journal of Strategic Innovation and Sustainability Vol*, 12(1):11, 2017.

[9] Shikhar Bahl, Peeyush Chandra, Vandana Rathore, Alka Shukla, and Akash Garg. Wireless ethernet for iot: A case study. In *2016 10th International Conference on Intelligent Systems and Control (ISCO)*, pages 1–6. IEEE, 2016.

[10] Jordi Mongay Batalla, Piotr Krawiec, Constandinos X Mavromoustakis, George Mastorakis, Naveen Chilamkurti, Daniel Negru, Joachim Bruneau-Queyreix, and Eugen Borcoci. Efficient media streaming with collaborative terminals for the smart city environment. *IEEE Communications Magazine*, 55(1):98–104, 2017.

[11] Jordi Mongay Batalla, George Mastorakis, Constandinos X Mavromoustakis, and Jerzy Zurek. On cohabitating networking technologies with common wireless access for home automation system purposes. *IEEE Wireless Communications*, 23(5):76–83, 2016.

[12] Omer F Beyca, Prahalad K Rao, Zhenyu Kong, Satish TS Bukkapatnam, and Ranga Komanduri. Heterogeneous sensor data fusion approach for real-time monitoring in ultraprecision machining (upm) process using non-parametric bayesian clustering and evidence theory. *IEEE Transactions on Automation Science and Engineering*, 13(2):1033–1044, 2015.

[13] Farshid Hassani Bijarbooneh, Wei Du, Edith C-H Ngai, Xiaoming Fu, and Jiangchuan Liu. Cloud-assisted data fusion and sensor selection for internet of things. *IEEE Internet of Things Journal*, 3(3):257–268, 2016.

[14] Jeff Birkby. Vertical farming. *ATTRA Sustainable Agriculture. NCAT IP516*, 12, 2016.

[15] Jens Bleiholder and Felix Naumann. Data fusion. *ACM Computing Surveys (CSUR)*, 41(1):1, 2009.

[16] Geoff Boeing. Urban spatial order: Street network orientation, configuration, and entropy. 2018.

[17] Alessio Botta, Walter De Donato, Valerio Persico, and Antonio Pescapé. Integration of cloud computing and internet of things: a survey. *Future Generation Computer Systems*, 56:684–700, 2016.

[18] Simone Brienza, Sena Efsun Cebeci, Seyed Saeid Masoumzadeh, Helmut Hlavacs, Öznur Özkasap, and Giuseppe Anastasi. A survey on energy efficiency in p2p systems: File distribution, content streaming, and epidemics. *ACM Computing Surveys (CSUR)*, 48(3):36, 2016.

[19] Rajkumar Buyya, Rajiv Ranjan, and Rodrigo N Calheiros. Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services. In *International Conference on Algorithms and Architectures for Parallel Processing*, pages 13–31. Springer, 2010.

[20] Claudia Canali, M Elena Renda, Paolo Santi, and Simone Burresi. Enabling efficient peer-to-peer resource sharing in wireless mesh networks. *IEEE Transactions on Mobile Computing*, 9(3):333–347, 2010.

[21] Vincenza Carchiolo, Michele Malgeri, Giuseppe Mangioni, and Vincenzo Nicosia. An adaptive overlay network inspired by social behaviour. *Journal of Parallel and Distributed Computing*, 70(3):282–295, 2010.

[22] Iacopo Carreras, Daniele Miorandi, Geoffrey S Canright, and Kenth Engo-Monsen. Understanding the spread of epidemics in highly partitioned mobile networks. In *Proceedings of the 1st international conference on Bio inspired models of network, information and computing systems*, page 2. ACM, 2006.

[23] Kyle Chard, Kris Bubendorfer, Simon Caton, and Omer F Rana. Social cloud computing: A vision for socially motivated resource sharing. *IEEE Transactions on Services Computing*, 5(4):551–563, 2012.

[24] Kyle Chard, Simon Caton, Omer Rana, and Kris Bubendorfer. Social cloud: Cloud computing in social networks. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, pages 99–106. IEEE, 2010.

[25] Le Chen, Rongxing Lu, and Zhenfu Cao. Pdaft: A privacy-preserving data aggregation scheme with fault tolerance for smart grid communications. *Peer-to-peer networking and applications*, 8(6):1122–1132, 2015.

[26] Nengcheng Chen, Xiang Zhang, and Chao Wang. Integrated open geospatial web service enabled cyber-physical information infrastructure for precision agriculture monitoring. *Computers and Electronics in Agriculture*, 111:78–91, 2015.

[27] Ray Chen, Fenye Bao, and Jia Guo. Trust-based service management for social internet of things systems. *IEEE transactions on dependable and secure computing*, 13(6):684–696, 2015.

[28] Mung Chiang and Tao Zhang. Fog and iot: An overview of research opportunities. *IEEE Internet of Things Journal*, 3(6):854–864, 2016.

[29] Bogdan-Cosmin Chifor, Ion Bica, Victor-Valeriu Patriciu, and Florin Pop. A security authorization scheme for smart home internet of things devices. *Future Generation Computer Systems*, 2017.

[30] Cristian Chilipirea, Andrei Ursache, Dan Octavian Popa, and Florin Pop. Energy efficiency and robustness for iot: Building a smart home security system. In *Intelligent Computer Communication and Processing (ICCP), 2016 IEEE 12th International Conference on*, pages 43–48. IEEE, 2016.

[31] Franco Cicirelli, Antonio Guerrieri, Giandomenico Spezzano, and Andrea Vinci. An edge-based platform for dynamic smart city applications. *Future Generation Computer Systems*, 2017.

[32] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Designing privacy enhancing technologies*, pages 46–66. Springer, 2001.

[33] Fernando Costa, Luis Silva, and Michael Dahlin. Volunteer cloud computing: Mapreduce over the internet. In *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on*, pages 1855–1862. IEEE, 2011.

[34] Cas JF Cremers. The scyther tool: Verification, falsification, and analysis of security protocols. In *CAV*, volume 8, pages 414–418. Springer, 2008.

[35] Vincenzo D Cunsolo, Salvatore Distefano, Antonio Puliafito, and Marco Scarpa. Volunteer computing and desktop cloud: The cloud@ home paradigm. In *Network Computing and Applications, 2009. NCA 2009. Eighth IEEE International Symposium on*, pages 134–139. IEEE, 2009.

[36] Neil Daswani, Hector Garcia-Molina, and Beverly Yang. Open problems in data-sharing peer-to-peer systems. In *International conference on database theory*, pages 1–15. Springer, 2003.

[37] LNC De Silva, Jeevani S Goonetillake, Gihan N Wikramanayake, and Athula Ginige. Harnessing mobile pervasive computing to enhance

livelihood processes: Farmer response to a mobile agriculture information system. In *International Conference on Green, Pervasive, and Cloud Computing*, pages 641–655. Springer, 2017.

[38] Travis Desell. Large scale evolution of convolutional neural networks using volunteer computing. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 127–128. ACM, 2017.

[39] Andrea Detti, Bruno Ricci, and Nicola Blefari-Melazzi. Mobile peer-to-peer video streaming over information-centric networks. *Computer Networks*, 81:272–288, 2015.

[40] Sanjay K Dhurandher, Sudip Misra, Puneet Pruthi, Shubham Singhal, Saurabh Aggarwal, and Isaac Woungang. Using bee algorithm for peer-to-peer file searching in mobile ad hoc networks. *Journal of Network and Computer Applications*, 34(5):1498–1508, 2011.

[41] Gianni Di Caro and Marco Dorigo. Antnet: Distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research*, 9:317–365, 1998.

[42] Chen Ding, Chen Yueguo, and Cheng Weiwei. A survey study on trust management in p2p systems. *Department of Computer Science, School of Computing, National University of Singapore*, 2004.

[43] Hugh Durrant-Whyte and Thomas C Henderson. Multisensor data fusion. In *Springer Handbook of Robotics*, pages 867–896. Springer, 2016.

[44] Tayfun Elmas and Oznur Ozkasap. Distributed document sharing with text classification over content-addressable network. *Content Computing*, pages 70–81, 2004.

[45] Muddassar Farooq. *Bee-inspired protocol engineering: from nature to networks*. Springer Science & Business Media, 2008.

[46] Nick Feamster and Hari Balakrishnan. Packet loss recovery for streaming video. In *12th International Packet Video Workshop*, pages 9–16. PA: Pittsburgh, 2002.

[47] Natalya Fedotova and Luca Veltri. Reputation management algorithms for dht-based peer-to-peer environment. *Computer Communications*, 32(12):1400–1409, 2009.

[48] Sven Fleck, Florian Busch, Peter Biber, and Wolfgang Straber. 3d surveillance a distributed network of smart cameras for real-time tracking and its visualization in 3d. In *Computer Vision and Pattern Recognition Workshop, 2006. CVPRW'06. Conference on*, pages 118–118. IEEE, 2006.

[49] Agostino Forestiero, Carlo Mastroianni, and Michela Meo. Self-chord: A bio-inspired algorithm for structured p2p systems. In *Cluster Computing and the Grid, 2009. CCGRID'09. 9th IEEE/ACM International Symposium on*, pages 44–51. IEEE, 2009.

[50] Jianjie Fu, Haiyan Zhang, Aiqian Zhang, and Guibin Jiang. E-waste recycling in china: a challenging field, 2018.

[51] Wojciech Galuba, Karl Aberer, Zoran Despotovic, and Wolfgang Kellerer. Self-organized fault-tolerant routing in peer-to-peer overlays. In *Self-Adaptive and Self-Organizing Systems, 2009. SASO'09. Third IEEE International Conference on*, pages 30–39. IEEE, 2009.

[52] Cristian González García, Daniel Meana-Llorián, B Cristina Pelayo G-Bustelo, Juan Manuel Cueva Lovelle, and Nestor Garcia-Fernandez. Midgar: Detection of people through computer vision in the internet of things scenarios to improve the security in smart cities, smart towns, and smart homes. *Future Generation Computer Systems*, 2017.

[53] Pedro Garcia Lopez, Alberto Montresor, Dick Epema, Anwitaman Datta, Teruo Higashino, Adriana Iamnitchi, Marinho Barcellos, Pascal Felber, and Etienne Riviere. Edge-centric computing: Vision and challenges. *ACM SIGCOMM Computer Communication Review*, 45(5):37–42, 2015.

[54] Panagiotis Georgopoulos, Yehia Elkhatib, Matthew Broadbent, Mu Mu, and Nicholas Race. Towards network-wide qoe fairness using openflow-assisted adaptive video streaming. In *Proceedings of the 2013 ACM SIGCOMM workshop on Future human-centric multimedia networking*, pages 15–20. ACM, 2013.

[55] Bogdan Ghit, Florin Pop, and Valentin Cristea. Epidemic-style global load monitoring in large-scale overlay networks. In *P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2010 International Conference on*, pages 393–398. IEEE, 2010.

[56] Neil J Gordon, David J Salmond, and Adrian FM Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. In *IEE proceedings F (radar and signal processing)*, volume 140, pages 107–113. IET, 1993.

[57] Xiaohui Gu and Klara Nahrstedt. A scalable qos-aware service aggregation model for peer-to-peer computing grids. In *Proceedings 11th IEEE International Symposium on High Performance Distributed Computing*, pages 73–82. IEEE, 2002.

[58] Fangfang GUO, Yibing HU, Longting XIU, Guangsheng FENG, and Shuaishuai WANG. A hierarchical p2p model and a data fusion method for network security situation awareness system. *Wuhan University Journal of Natural Sciences*, 2:007, 2016.

[59] Karim Habak, Mostafa Ammar, Khaled A Harras, and Ellen Zegura. Femto clouds: Leveraging mobile devices to provide cloud service at the edge. In *2015 IEEE 8th International Conference on Cloud Computing (CLOUD)*, pages 9–16. IEEE, 2015.

[60] Vasos Hadjioannou, Constandinos X Mavromoustakis, George Mastorakis, Jordi Mongay Batalla, Ioannis Kopanakis, Emmanouil Perakakis, and Spiros Panagiotakis. Security in smart grids and smart spaces for smooth iot deployment in 5g. In *Internet of Things (IoT) in 5G Mobile Technologies*, pages 371–397. Springer, 2016.

[61] David Hales and Bruce Edmonds. Applying a socially inspired technique (tags) to improve cooperation in p2p networks. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 35(3):385–395, 2005.

[62] Nen-Fu Huang, Yih-Jou Tzang, Hong-Yi Chang, and Chia-Wen Ho. Enhancing p2p overlay network architecture for live multimedia streaming. *Information Sciences*, 180(17):3210–3231, 2010.

[63] Cisco Visual Networking Index. Forecast and methodology, 2014-2019 white paper. Technical report, Technical Report, Cisco, 2015.

[64] Liming Jiang, Jian Xu, Kun Zhang, and Hong Zhang. A new evidential trust model for open distributed systems. *Expert Systems with applications*, 39(3):3772–3782, 2012.

[65] Eric Jonas, Qifan Pu, Shivaram Venkataraman, Ion Stoica, and Benjamin Recht. Occupy the cloud: Distributed computing for the 99%. In *Proceedings of the 2017 Symposium on Cloud Computing*, pages 445–451. ACM, 2017.

[66] Yuh-Jzer Joung and Li-Wei Yang. On character-based index schemes for complex wildcard search in peer-to-peer networks. *Information Sciences*, 272:209–222, 2014.

[67] Janmejay Kale and VR Chirchi. Result and analysis: Data sharing between peer-to-peer using trust model. *International Journal of Computer Applications*, 157(8):30–33, 2017.

[68] Vasileios Karagiannis, Michael Borkowski, Alexandre Venito, Rodrigo Coelho, and Gerhard Fohler. Edge computing with peer to peer interactions: use cases and impact. In *Proceedings of the Workshop on Fog Computing and the IoT*, pages 46–50. ACM, 2019.

[69] Konrad Karolewicz, Andrzej Beben, Jordi Mongay Batalla, George Mastorakis, and Constandinos X Mavromoustakis. On efficient data storage service for iot. *International Journal of Network Management*, 27(3), 2017.

[70] John A Kay, David C Mazur, and Rob A Entzminger. Basics of communication networks for electrical engineers in the forest products industries. In *2018 IEEE IAS Pulp, Paper and Forest Industries Conference (PPFIC)*, pages 1–5. IEEE, 2018.

[71] Rasib Khan and Ragib Hasan. Secp2psip: A distributed overlay architecture for secure p2psip. In *Proc. of Cyber Security Conf*, 2014.

[72] Zaheer Khan, Zeeshan Pervez, and Abdul Ghafoor Abbasi. Towards a secure service provisioning framework in a smart city environment. *Future Generation Computer Systems*, 77:112–135, 2017.

[73] Tor Klingberg and Raphael Manfredi. The gnutella protocol specification v0. 6. *Technical specification of the Protocol*, 2002.

[74] William Kluegel, Muhammad A Iqbal, Ferdinando Fioretto, William Yeoh, and Enrico Pontelli. A realistic dataset for the smart home device scheduling problem for dcops. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 125–142. Springer, 2017.

[75] Diego Kreutz, Fernando MV Ramos, Paulo Esteves Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, 2015.

[76] John Krumm. *Ubiquitous computing fundamentals*. CRC Press, 2016.

[77] Yiannos Kryftis, George Mastorakis, Constandinos X Mavromoustakis, Jordi Mongay Batalla, Evangelos Pallis, and Georgios Kormentzas. Efficient entertainment services provision over a novel network architecture. *IEEE Wireless Communications*, 23(1):14–21, 2016.

[78] Yoram Kulbak, Danny Bickson, et al. The emule protocol specification. emule project, 2005.

[79] Uichin Lee, Eugenio Magistretti, Mario Gerla, Paolo Bellavista, Pietro Lió, and Kang-Won Lee. Bio-inspired multi-agent data harvesting in a proactive urban monitoring environment. *Ad Hoc Networks*, 7(4):725–741, 2009.

[80] Baochun Li, Zhi Wang, Jiangchuan Liu, and Wenwu Zhu. Two decades of internet video streaming: A retrospective view. *ACM transactions on multimedia computing, communications, and applications (TOMM)*, 9(1s):33, 2013.

[81] Jian Liang, Rakesh Kumar, and Keith W Ross. The fasttrack overlay: A measurement study. *Computer Networks*, 50(6):842–858, 2006.

[82] Martin Liggins II, David Hall, and James Llinas. *Handbook of multisensor data fusion: theory and practice*. CRC press, 2017.

[83] Yong Liu, Yang Guo, and Chao Liang. A survey on peer-to-peer video streaming systems. *Peer-to-peer Networking and Applications*, 1(1):18–28, 2008.

[84] Tom H Luan, Longxiang Gao, Zhi Li, Yang Xiang, Guiyi Wei, and Limin Sun. Fog computing: Focusing on mobile users at the edge. *arXiv preprint arXiv:1502.01815*, 2015.

[85] Zhanshan Sam Ma and Axel W Krings. Insect sensory systems inspired computing and communications. *Ad Hoc Networks*, 7(4):742–755, 2009.

[86] Nazanin Magharei, Reza Rejaie, Ivica Rimac, Volker Hilt, and Markus Hofmann. Isp-friendly live p2p streaming. *IEEE/ACM Transactions on Networking*, 22(1):244–256, 2014.

[87] Adeel Mohammad Malik, Bengt Ahlgren, and Börje Ohlman. Netinf live video streaming for events with large crowds. In *Proceedings of the 2nd International Conference on Information-Centric Networking*, pages 209–210. ACM, 2015.

[88] Stefan Marksteiner, Víctor Juan Exposito Jimenez, Heribert Valiant, and Herwig Zeiner. An overview of wireless iot protocol security in the smart home domain. In *2017 Internet of Things Business Models, Users, and Networks*, pages 1–8. IEEE, 2017.

[89] Attila Marosi, József Kovács, and Peter Kacsuk. Towards a volunteer cloud system. *Future Generation Computer Systems*, 29(6):1442–1451, 2013.

[90] Sergio Marti and Hector Garcia-Molina. Taxonomy of trust: Categorizing p2p reputation systems. *Computer Networks*, 50(4):472–484, 2006.

[91] Constandinos Mavromoustakis, George Mastorakis, and Jordi Mongay Batalla. *Internet of Things (IoT) in 5G mobile technologies*, volume 8. Springer, 2016.

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

[93] Garrett McGrath and Paul R Brenner. Serverless computing: Design, implementation, and performance. In *2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW)*, pages 405–410. IEEE, 2017.

[94] Bogdan Mocanu, Florin Pop, Alexandra Mihaita Mocanu, Ciprian Dobre, and Valentin Cristea. Spider: A bio-inspired structured peer-to-peer overlay for data dissemination. In *2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)*, pages 291–295. IEEE, 2015.

[95] Nima Jafari Navimipour and Farnaz Sharifi Milani. A comprehensive study of the resource discovery techniques in peer-to-peer networks. *Peer-to-Peer Networking and Applications*, 8(3):474–492, 2015.

[96] Rim Negra, Imen Jemili, and Abdelfettah Belghith. Wireless body area networks: Applications and technologies. *Procedia Computer Science*, 83:1274–1281, 2016.

[97] Bruno Astuto A Nunes, Marc Mendonca, Xuan-Nam Nguyen, Katia Obraczka, and Thierry Turletti. A survey of software-defined networking: Past, present, and future of programmable networks. *IEEE Communications Surveys & Tutorials*, 16(3):1617–1634, 2014.

[98] Sergio F Ochoa and Rodrigo Santos. Human-centric wireless sensor networks to improve information availability during urban search and rescue activities. *Information Fusion*, 22:71–84, 2015.

[99] Francesco Palmieri, Massimo Ficco, Silvio Pardi, and Aniello Castiglione. A cloud-based architecture for emergency management and first responders localization in smart city environments. *Computers & Electrical Engineering*, 56:810–830, 2016.

[100] Tao Peng, Qin Liu, Dacheng Meng, and Guojun Wang. Collaborative trajectory privacy preserving scheme in location-based services. *Information Sciences*, 387:165–179, 2017.

[101] Andrei Poenaru, Roxana Istrate, and Florin Pop. Aft: Adaptive and fault tolerant peer-to-peer overlay—a user-centric solution for data sharing. *Future Generation Computer Systems*, 80:583–595, 2018.

[102] Florin Pop, Oana-Maria Citoteanu, Ciprian Dobre, and Valentin Cristea. Resource trust management in auto-adaptive overlay network for mobile cloud computing. In *Parallel and Distributed Computing (ISPDC), 2014 IEEE 13th International Symposium on*, pages 162–169. IEEE, 2014.

[103] Florin Pop, Oana-Maria Citoteanu, Ciprian Dobre, and Valentin Cristea. Resource trust management in auto-adaptive overlay network for mobile cloud computing. In *Proceedings of the 2014 IEEE 13th International Symposium on Parallel and Distributed Computing*, ISPDC '14, pages 162–169, Washington, DC, USA, 2014. IEEE Computer Society.

[104] Jon Postel. Transmission control protocol. *RFC: 793*, 1981.

[105] Johan Pouwelse, Pawel Garbacki, Dick Epema, and Henk Sips. The bittorrent p2p file-sharing system: Measurements and analysis. In *IPTPS*, volume 5, pages 205–216. Springer, 2005.

[106] Basit Qureshi, Geyong Min, and Demetres Kouvatsos. A distributed reputation and trust management scheme for mobile peer-to-peer networks. *Computer Communications*, 35(5):608–618, 2012.

[107] Hayder M Radha, Mihaela Van der Schaar, and Yingwei Chen. The mpeg-4 fine-grained scalable video coding method for multimedia streaming over ip. *Multimedia, IEEE Transactions on*, 3(1):53–68, 2001.

[108] AJ Dinusha Rathnayaka, Vidyasagar M Potdar, and Samitha J Kuruppu. Evaluation of wireless home automation technologies. In *5th IEEE International Conference on Digital Ecosystems and Technologies (IEEE DEST 2011)*, pages 76–81. IEEE, 2011.

[109] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. *A scalable content-addressable network*, volume 31. ACM, 2001.

[110] Sylvia Ratnasamy, Mark Handley, Richard Karp, and Scott Shenker. Application-level multicast using content-addressable networks. *Networked Group Communication*, pages 14–29, 2001.

[111] Shini Renjith, Sreekumar A, and Jathavedan M. Evaluation of partitioning clustering algorithms for processing social media data in tourism domain. *Proceedings of the IEEE*, 12 2018.

[112] Rodrigo Roman, Javier Lopez, and Masahiro Mambo. Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges. *Future Generation Computer Systems*, 78:680–698, 2018.

[113] Gonzalo Sánchez-Arias, Cristian González García, and B Cristina Pelayo G-Bustelo. Midgar: Study of communications security among smart objects using a platform of heterogeneous devices for the internet of things. *Future Generation Computer Systems*, 74:444–466, 2017.

[114] Farook Sattar, Fakhri Karray, Mohamed Kamel, Lobna Nassar, and Keyvan Golestan. Recent advances on context-awareness and data/information fusion in its. *International Journal of Intelligent Transportation Systems Research*, 14(1):1–19, 2016.

[115] Rüdiger Schollmeier. A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. In *Peer-to-Peer Computing, 2001. Proceedings. First International Conference on*, pages 101–102. IEEE, 2001.

[116] Cristina Şerbanescu. Noncommutative markov processes as stochastic equations' solutions. *Bull. Math. Soc. Sc. Math. Roumanie Tome 41*, 89(3):219–228, 1998.

[117] Cristina Şerbanescu. Stochastic differential equations and unitary processes. *Bull. Math. Soc. Sc. Math. Roumanie Tome 41*, 89(3):311–322, 1998.

[118] Vlad-Nicolae Şerbanescu, Florin Pop, Valentin Cristea, and Ovidiu-Marian Achim. Web services allocation guided by reputation in distributed soa-based environments. In *Parallel and Distributed Computing (ISPDC), 2012 11th International Symposium on*, pages 127–134, June 2012.

[119] Glenn Shafer et al. *A mathematical theory of evidence*, volume 1. Princeton university press Princeton, 1976.

[120] Alimohammad Shahri, Mahmood Hosseini, Raian Ali, and Fabiano Dalpiaz. Gamification for volunteer cloud computing. In *Utility and Cloud Computing (UCC), 2014 IEEE/ACM 7th International Conference on*, pages 616–617. IEEE, 2014.

[121] Wentao Shang, Yingdi Yu, Ralph Droms, and Lixia Zhang. Challenges in iot networking via tcp/ip architecture. *NDN, Technical Report NDN-0038*, 2016.

[122] Haiying Shen, Jinwei Liu, Kang Chen, Jianwei Liu, and Stanley Moyer. Scps: A social-aware distributed cyber-physical human-centric search engine. *IEEE Transactions on Computers*, 64(2):518–532, 2015.

[123] Josef Spillner. Snafu: Function-as-a-service (faas) runtime design and implementation. *arXiv preprint arXiv:1703.07562*, 2017.

[124] Hatem Ben Sta. Quality and the efficiency of data in "smart-cities". *Future Generation Computer Systems*, 74:409–416, 2017.

[125] Ion Stoica, Robert Morris, David Liben-Nowell, David R Karger, M Frans Kaashoek, Frank Dabek, and Hari Balakrishnan. Chord: a scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Transactions on Networking (TON)*, 11(1):17–32, 2003.

[126] Ivan Stojmenovic. Honeycomb networks: Topological properties and communication algorithms. *Parallel and Distributed Systems, IEEE Transactions on*, 8(10):1036–1042, 1997.

[127] Ugur Eray Tahta, Sevil Sen, and Ahmet Burak Can. Gentrust: A genetic trust management model for peer-to-peer systems. *Applied Soft Computing*, 34:693–704, 2015.

[128] Ming Tao, Kaoru Ota, and Mianxiong Dong. Ontology-based data semantic management and application in iot-and cloud-enabled smart homes. *Future Generation Computer Systems*, 2016.

[129] Nicolae Ţăpuş and Pantelimon George Popescu. A new entropy upper bound. *Applied Mathematics Letters*, 25(11):1887–1890, 2012.

[130] Chunqi Tian and Baijian Yang. Trust, a reputation and risk based trust management framework for large-scale, fully decentralized overlay networks. *Future Generation Computer Systems*, 27(8):1135–1141, 2011.

[131] Chunqi Tian, Baijian Yang, Jidong Zhong, and Xiaojian Liu. Trust-based incentive mechanism to motivate cooperation in hybrid p2p networks. *Computer Networks*, 73:244–255, 2014.

[132] Duc A Tran, Kien A Hua, and Tai Do. Zigzag: An efficient peer-to-peer scheme for media streaming. In *INFOCOM 2003. Twenty-second annual joint conference of the IEEE computer and communications. IEEE Societies*, volume 2, pages 1283–1292. IEEE, 2003.

[133] Noelia Uribe-Pérez and Carles Pous. A novel communication system approach for a smart city based on the human nervous system. *Future Generation Computer Systems*, 76:314–328, 2017.

[134] Blesson Varghese and Rajkumar Buyya. Next generation cloud computing: New trends and research directions. *Future Generation Computer Systems*, 79:849–861, 2018.

[135] Andreea Vişan, Florin Pop, and Valentin Cristea. Decentralized trust management in peer-to-peer systems. In *Parallel and Distributed Computing (ISPDC), 2011 10th International Symposium on*, pages 232–239. IEEE, 2011.

[136] Werner Vogels, Robbert Van Renesse, and Ken Birman. The power of epidemics: robust communication for large-scale distributed systems. *ACM SIGCOMM Computer Communication Review*, 33(1):131–135, 2003.

[137] Chia-Hui Wang, Yu-Hsien Chu, and Tsa-Ta Wei. Siptvmon: a secure multicast overlay network for load-balancing and stable iptv service using

sip. In *Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on*, pages 97–102. IEEE, 2011.

[138] Man-tao Wang, Jiang-shu Wei, Yong-hao Pan, and Zhe Wei. Study on data fusion techniques in wireless sensor networks. In *Proceedings of the 6th International Asia Conference on Industrial Engineering and Management Innovation*, pages 67–74. Springer, 2016.

[139] Shiqiang Wang, Rahul Urgaonkar, Murtaza Zafer, Ting He, Kevin Chan, and Kin K Leung. Dynamic service migration in mobile edge-clouds. In *IFIP Networking Conference (IFIP Networking), 2015*, pages 1–9. IEEE, 2015.

[140] Yufeng Wang and Akihiro Nakao. Poisonedwater: An improved approach for accurate reputation ranking in p2p networks. *Future Generation Computer Systems*, 26(8):1317–1326, 2010.

[141] Zhi Wang, Lifeng Sun, Chuan Wu, Wenwu Zhu, and Shiqiang Yang. Joint online transcoding and geo-distributed delivery for dynamic adaptive streaming. In *INFOCOM, 2014 Proceedings IEEE*, pages 91–99. IEEE, 2014.

[142] Cedric Westphal. Challenges in networking to support augmented reality and virtual reality. *IEEE ICNC*, 2017.

[143] Matthias Wichtlhuber, Björn Richerzhagen, Julius Rückert, and David Hausheer. Transit: Supporting transitions in peer-to-peer live video streaming. In *Networking Conference, 2014 IFIP*, pages 1–9. IEEE, 2014.

[144] Thomas Wiegand, Gary J Sullivan, Gisle Bjøntegaard, and Ajay Luthra. Overview of the h. 264/avc video coding standard. *Circuits and Systems for Video Technology, IEEE Transactions on*, 13(7):560–576, 2003.

[145] Dapeng Wu, Yiwei Thoms Hou, and Ya-Qin Zhang. Transporting real-time video over the internet: Challenges and approaches. *Proceedings of the IEEE*, 88(12):1855–1877, 2000.

[146] Jiyan Wu, Chau Yuen, Bo Cheng, Ming Wang, and Junliang Chen. Streaming high-quality mobile video with multipath tcp in heterogeneous wireless networks. *IEEE Transactions on Mobile Computing*, 15(9):2345–2361, 2015.

[147] Zhipu Xie, Weifeng Lv, Linfang Qin, Bowen Du, and Runhe Huang. An evolvable and transparent data as a service framework for multisource data integration and fusion. *Peer-to-Peer Networking and Applications*, pages 1–14, 2017.

[148] Ke Xu, Xiaoliang Wang, Wei Wei, Houbing Song, and Bo Mao. Toward software defined smart home. *IEEE Communications Magazine*, 54(5):116–122, 2016.

[149] Ronald R Yager. A framework for multi-source data fusion. *Information Sciences*, 163(1):175–200, 2004.

[150] Qiben Yan, Yao Zheng, Tingting Jiang, Wenjing Lou, and Y Thomas Hou. Peerclean: Unveiling peer-to-peer botnets through dynamic group behavior analysis. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 316–324. IEEE, 2015.

[151] Ye Yan, Yi Qian, Hamid Sharif, and David Tipper. A survey on smart grid communication infrastructures: Motivations, requirements and challenges. *IEEE communications surveys & tutorials*, 15(1):5–20, 2013.

[152] Xiaohui Yang. Application of gpon in campus network. In *2018 3rd International Conference on Humanities Science, Management and Education Technology (HSMET 2018)*. Atlantis Press, 2018.

[153] Andrea Zanella, Nicola Bui, Angelo Castellani, Lorenzo Vangelista, and Michele Zorzi. Internet of things for smart cities. *IEEE Internet of Things journal*, 1(1):22–32, 2014.

[154] Mahdi Zareei, Azar Zarei, Rahmat Budiarto, and Mohd Adib Omar. A comparative study of short range wireless sensor network on high density networks. In *The 17th Asia Pacific conference on communications*, pages 247–252. IEEE, 2011.

[155] Desheng Zhang, Juanjuan Zhao, Fan Zhang, and Tian He. Urbancps: a cyber-physical system based on multi-source big infrastructure data for heterogeneous model integration. In *Proceedings of the ACM/IEEE Sixth International Conference on Cyber-Physical Systems*, pages 238–247. ACM, 2015.

[156] Cliff C Zou, Weibo Gong, Don Towsley, and Lixin Gao. The monitoring and early detection of internet worms. *IEEE/ACM Transactions on Networking (TON)*, 13(5):961–974, 2005.