Local peer-to-peer mobile access to linked data in resource-constrained networks

Mahamadou Toure

UNIVERSITÉ CÔTE D'AZUR

ÉCOLE DOCTORALE
SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

UNIVERSITE GASTON BERGER
L'excellence au service du développement

THÈSE DE DOCTORAT

Accès Mobile Local en Pair-à-Pair aux Données Liées sur des Réseaux à Ressources Limitées

Mahamadou TOURE

Laboratoire d'Informatique, Signaux et Systèmes de Sophia Antipolis (UMR7271)

Présentée en vue de l'obtention du grade de docteur en Informatique de l'Université Côte d'Azur et de l'Université Gaston Berger du Sénégal

Dirigée par :
Mr. Fabien GANDON, Directeur de Recherche, INRIA

Dirigée par :
Mr. Moussa LO, Professeur d'Etablissement Etranger, Université Gaston Berger (Sénégal)

Soutenue le : 04 Octobre 2021

Devant le jury, composé de :

Président du jury :
Mr. Johan MONTAGNAT, Directeur de Recherche, Université Côte d'Azur

Rapporteurs :
Mme. Hala SKAF, Maitre de Conférences - HDR, Université de Nantes
Mr. Ousmane SALL, Professeur d'Etablissement Etranger, Ecole Polytechnique de Thiès (Sénégal)

Invité :
Mr. Olivier CORBY, Chercheur INRIA, Université Côte d'Azur, I3S, SPARKS, CNRS

Inria inventeurs du monde numérique

CEA MITIC
Centre d'Excellence Africain en Mathématiques, Informatique et TIC

Local Peer-to-Peer Mobile Access to Linked Data in Resource-constrained Networks

Mahamadou Toure

A thesis in fulfilment of the requirements for the degree of

Doctor of Philosophy

Université Gaston Berger and Université Côte d'Azur

UGB/UCA

October 2021

Abstract

Access to the Web of Data is nowadays of real interest for research, mainly in the sense that the clients consuming or processing this data are more and more numerous and have various specificities (mobility, Internet, autonomy, storage, etc.). Tools such as Web applications, search engines, e-learning platforms, etc., exploit the Web of Data to offer end-users services that contribute to the improvement of daily activities.
In this context, we are working on Web of Data access, considering constraints such as customer mobility and intermittent availability of the Internet connection. We are interested in mobility as our work is oriented towards end-users with mobile devices such as smartphones, tablets, laptops, etc. The intermittency of Internet connection refers herein to scenarios of unavailability of national or international links that make remote data sources inaccessible.
We target a scenario where users form a peer-to-peer network such that anyone can generate information and make it available to everyone else on the network. Thus, we survey and compare several solutions (models, architectures,etc.) dedicated to Web of Data access by mobile contributors and discussed in relation to the underlying network architectures and data models considered. We present a conceptual study of peer-to-peer solutions based on gossip protocols dedicated to design the connected overlay networks and present a detailed analysis of data replication systems whose general objective is to ensure a system's local data availability.
On the basis of this work, we proposed an architecture adapted to constraining environments and allowing mobile contributors to share locally, via a browser network, an RDF dataset. The architecture consists of 3 levels: single peers, super peers and remote sources. Two main axes are considered for the implementation of this architecture: firstly the construction and maintenance of connectivity ensured by the gossip protocol, and secondly the high availability of data ensured by a replication mechanism.
Our approach has the particularity to consider the location of each participant's neighbours to increase the search perimeter and to integrate super-peers on which the data graph is replicated allowing data availability improvement.
We finally carried out an experimental evaluation of our architecture through extensive simulation configured to capture key aspects of our motivating scenario of supporting data exchange between the participants of a local event.

Keywords: Semantic Web, gossip protocol, peer-to-peer system, mobile access, Data

Web, graph replication, limited connectivity, semantic overlay, mobile contributor.

Résumé

L'accès au Web de données présente aujourd'hui un réel intérêt pour la recherche, notamment dans le sens où les clients consommant ou traitant ces données sont de plus en plus nombreux et présentent des spécificités diverses (mobilité, Internet, autonomie, stockage, etc.). Des outils tels que les applications Web, les moteurs de recherche, les plateformes de e-learning, etc., exploitent le Web de données pour offrir aux utilisateurs finaux des services qui contribuent à l'amélioration des activités quotidiennes.
Dans ce contexte, nous travaillons sur l'accès au Web de données en tenant compte de contraintes telles que la mobilité des clients et la disponibilité par intermittence de la connexion Internet. Nous nous intéressons à la mobilité du fait que nos travaux sont orientés vers les utilisateurs finaux disposant d'appareils mobiles tels que les smartphones, les tablettes, les ordinateurs portables, etc. L'intermittence de la connexion Internet fait ici référence aux scénarios d'indisponibilité des liaisons nationales ou internationales qui rendent inaccessibles les sources de données distantes.
Nous visons un scénario dans lequel les utilisateurs forment un réseau pair-à-pair tel que chacun peut générer des informations et les mettre à la disposition de tous les autres membres du réseau. Ainsi, nous analysons et comparons plusieurs solutions (modèles, architectures, etc.) dédiées à l'accès au Web de Données par des contributeurs mobiles et discutées par rapport aux architectures de réseau sous-jacentes et aux modèles de données considérés. Nous présentons une étude conceptuelle des solutions pair à pair basées sur les protocoles gossip dédiés à la conception des réseaux superposés connectés et présentons une analyse détaillée des systèmes de réplication de données dont l'objectif général est de garantir la disponibilité des données locales d'un système.
Sur la base de ces travaux, nous avons proposé une architecture adaptée aux environnements contraignants et permettant aux contributeurs mobiles de partager localement, via un réseau de navigateurs, un jeu de données RDF. L'architecture se compose de 3 niveaux : les pairs simples, les supers pairs et les sources distantes. Deux axes principaux sont considérés pour l'implémentation de cette architecture : d'une part la construction et la maintenance de la connectivité assurée par le protocole gossip, et d'autre part la haute disponibilité des données assurée par un mécanisme de réplication.
Notre approche a la particularité de prendre en compte la localisation des voisins de chaque participant pour augmenter le périmètre de recherche et d'intégrer des super-pairs sur lesquels le graphe de données est répliqué permettant d'améliorer la disponibilité des données.

Enfin, nous avons procédé à une évaluation expérimentale de notre architecture par le biais d'une simulation étendue configurée pour capturer les aspects clés de notre scénario de motivation consistant à soutenir l'échange de données entre les participants d'un événement local.

Mots-clés: Web sémantique, protocole de commérage, système pair-à-pair, accès mobile, Web de données, réplication de graphe, connectivité limitée, superposition sémantique, contributeur mobile.

Remerciements

Je tiens à remercier particulièrement mes directeurs de thèse, Pr. Moussa LO et Fabien GANDON, pour leur rigueur scientifique, leur encadrement et leurs conseils avisés, sans qui cette thèse n'aurait pas vu le jour.

Je souhaite exprimer ma gratitude aux membres du jury pour avoir accepté d'examiner et d'évaluer cette thèse. Merci pour vos commentaires perspicaces sur le manuscrit de la thèse qui m'ont permis de continuer à améliorer la qualité du rapport et je suis honoré de vous présenter mon travail.

Un grand merci au Dr. Kaladzavi GUIDEDI qui a apporté une dimension plus concrète à ce travail, et dont la vision critique et les conseils ont permis d'améliorer la qualité scientifique de cette thèse.

Merci à Christine pour toute l'aide qu'elle m'a apportée dans les démarches administratives.

Merci aux membres de l'équipe WIMMICS pour l'accueil chaleureux, pour les bonnes conditions de travail, et pour la qualité intellectuelle et humaine de tous. Merci pour les moments passés durant mes séjours à Sophia Antipolis.

A mon père, ma mère, mes frères et sœurs, à toute ma famille, je vous exprime ma gratitude pour votre soutient infaillible, pour vos prières, pour vos conseils, pour avoir été à mes côtés durant toutes ces années. Merci d'être là encore aujourd'hui pour moi.

Mes pensées vont particulièrement au Dr. Papa Fary Diallo, avec qui j'ai commencé cette aventure et qui nous a brutalement quitté au tout début du projet. Je lui dédie ce travail et je prie que le bon Dieu l'accueille dans son paradis céleste.

Contents

List of Figures

List of Tables

Abbreviations

API	Application Programming Interface
CCI	convergence, causality preservation and intention preservation
CFS	Cooperative File System
CRDT	Commutative Replicated Data Type
DHT	Distributed Hash Table
DPWS	Devices Profile for Web Services
ERS	Entity Registry System
GON	Geographic Overlay Network
GSM	Global System for Mobile Communications
GSMA	Global System for Mobile Communications Association
IoT	Internet of Things
IP	Internet Protocol
IT	Information Technology
JSON-LD	JavaScript Object Notation for Linked Data
KNN	K-Nearest-Neighbor
LDF	Linked Data Fragment
LinkedMDB	Linked Movies Database
LLD	Live Linked Data
MAAN	Multi-Attribute Addressable Network
MCC	Mobile Cloud Computing
MEC	Mobile Edge Computing

MMP	Membership Management Protocol
MORAI	Mobile Read Access in Intermittent
OR-Set	Observed-Removed Set
OT	Operational Transformation
P2P	Peer to Peer
P2P-TV	Peer to Peer Television
PaaS	Platform as a Service
PAGE	Put and Get Everywhere
QoS	Quality of Service
RAM	Random Access Memory
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
RDQL	Resource Description Framework Query Language
RPS	Random Peer Sampling
RQL	Resource Query Language
SECR	Semantic Edge Computing Runtime
SOC	Semantic Overlay Cluster
SON	Semantic Overlay Networks
SP	Super Peer
SP2P	Peer to peer Semantic Systems
TCP	Transmission Control Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VFC	Vehicle Fog Computing
VM	Virtual Machine
W3C	World Wide Web Consortium
WebRTC	Web Real-Time Communications
XML	Extensible Markup Language

Chapter 1

Introduction: the importance of making data accessible

1.1 Context: Linked Data with limited or unreliable I connectivity

The Web (or Web of Documents) is originally seen as a software architecture for making documents available, and for linking and sharing them over a network of connected machines [6]. This vision evolved very quickly. Indeed, during the W3C future directions presentation in 1994 [7], Tim Berners-Lee shows how the vision of hypertext, i.e. the linking of documents by hypertext links, must be overcomed to allow machines to automatically link Web data to real world-elements, which would at the same time allow intelligent agents to add and manipulate Web contents. In 2006, Tim Berners-Lee [8] clarified his semantic Web vision by emphasizing that its sole aim was not to publish information on the Web, but to link them in order to allow a machine or a human to browse the Web of data. He then presented a collection of guidelines for delivering and linking structured data on the Web: the Linked Data Principles. These principles offer directions on the application of semantic Web standards technologies (URI, RDF, SPARQL, etc.)

to link data from different sources. By analogy to traditional Web (of documents) that can be explored via hyperlinks, applications that adopt the principles of Linked Data can browse the Web of Data made up of different data sources by following the RDF (Resource Description Framework) links to provide more relevant responses to users needs [9].

In a number of the distributed systems such as Web applications, search engines, e-learning platforms, etc., some network nodes provide services, and others consume these services. Internet is the common communication channel that allows the nodes that constitute data sources and services (server nodes) and consumer nodes of these services (client nodes) to be connected and interacting. However, the availability and particularly the quality of the Internet is not always ensured in certain areas such as the African continent. Indeed, despite the significant progress made in recent years, Internet access remains a major constraint in developing countries in general and Africa in particular. According to Internet Words Stats data [10] of December 2020/June 2021, the penetration rate was estimated at 46.2% in Africa compared to 93.9% in North America, 87.1% in Europe and 63.8% in Asia. Furthermore, even when Internet access is provided, the distributed systems must cope with the dynamics of their environments. Distributed systems have to mainly consider the eventual nodes failure and the popular client/server model is then revealing its limitations. In such model, the server is the only node that provides the service and in case of unavailability, it becomes the single failure point for the entire distributed system. Moreover, this centralized architecture, apart from the significant costs that may be incurred to keep it operational, does not efficiently exploit the available resources (storage memory, processor, etc.) of the client nodes, which participate passively to the architecture and benefit from the provided service. Nevertheless, these client nodes have the particularity of being increasingly mobile, numerous and constitute a privileged means of access and consumption of Web's services. For example, GSM Association (GSMA) 2020 report on mobile economy in Sub-Saharan Africa [11] shows that smartphone adoption continues to rise rapidly in the region, reaching 50% of total connections in 2020, as cheaper devices have become available. Over the next five years, the number of smartphone connections in Sub-Saharan Africa will almost double to reach 678 million by the end of 2025, representing an adoption rate of 65% of the population. Even beyond the context of

limited or unreliable Internet connectivity, we can identify other scenarios where relying on more distributed and local connectivity is beneficial, for instance when pursuing digital frugality or ensuring even more resilient networking.

In recent years, peer-to-peer (P2P) model positioned itself to be an efficient solution to meet the dynamism and scalability requirements of distributed systems [12]. Nodes act both as clients and servers, contributing to the services in which they participate. Regardless of the specific type of application, P2P systems can be defined according to the topology connecting the nodes to each other to form the underlying network usually called "overlay" [12]. There are two main classes of the superposed networks [12]: structured overlay networks well represented by distributed hash tables (DHT) and unstructured overlay networks. The former link their peers according to their identifiers in order to allow efficient routing between them. The identifier of each node determines its position on the network. For the unstructured overlay networks, links between nodes are created randomly or based on proximity measurements. Networks based on gossip protocols are good illustrations [13–16]. These protocols aim to build and maintain an unstructured topology with random graph properties [17]. They also provide load balancing between nodes and are used as building blocks in network management applications, in particular for very dynamic environments [15].

1.2 Objectives: allow mobile users to access Linked Data locally

Mobile access to the Web of data has been the subject of several studies in recent years, but there is still a need to design and provide users with tools that allow them to access or contribute to data, in mobile situations, with limited resources (storage memory, processor, RAM, etc.) and also in cases where Internet access is limited or unreliable. Our general objective is to propose models, protocols and algorithms for mobile applications that allow users to access, share and publish data in an environment where hardware and software resources are limited, with the aim of locally and automatically co-constructing

distributed RDF data triple warehouses representing a shared socio-cultural knowledge base on semantic Web.

An important part of our work consists in setting up a distributed network coverage with restricted access to Internet that will allow users to exchange information. To build such a network, we focus particularly on gossip-based or epidemic protocols. In this type of protocol, when a node (a user) wants to send information via the network, it randomly selects t nodes in the system (t is a configuration parameter called fanout) and sends them the message. After receiving the message for the first time, each node repeats this procedure. These protocols have an interesting approach in that they are highly resilient (they have an intrinsic level of redundancy that allows them to mask failures at the network and node level) and scalable (the load is distributed across all nodes in the system). We are therefore interested in studying and designing a protocol in the continuity of this family but dedicated to sharing and maintaining RDF local knowledge bases.

Another important objective of this work is to propose an approach that allows mobile users to access data locally when access to remote sources is intermittent. This approach should be supported by an efficient replication scheme of locally relevant data across the architecture. We are particularly interested in RDF data whose structuring is assimilable to the one of a graph. In decentralised RDF data sharing systems, replication allows to share, collaboratively modify, or store all or part of the graph on the nodes participating in the architecture. Conceptually different approaches such as semantic overlays, cloud/Fog computing or structured peer-to-peer systems are presented in the literature as efficient supports to ensure local access to the graph. In this thesis, we want to analyse all these approaches and propose a solution adapted to our context and environment.

In this thesis, we are also interested in peer-to-peer systems that take into account the location of peers to manage views, exchanges, etc. Research in virtual environments has demonstrated the importance of partner location information to support collaborative processes such as task dispatching within the group [18]. Using geographic coordinates, nodes in a peer-to-peer architecture can form or improve the quality of their views. The relationships between the peers constituting the architecture are no longer based solely

on social ties or similar interests, but are also based on geographical proximity. Each node has its own coordinates and, through an underlying mechanism (broadcast message, central server, query, etc.), accesses the coordinates of other peers in the system. In peer-to-peer systems using gossip protocols, for example, these coordinates will be used for geographical distance calculation in order to determine which peers are closest (in terms of metric distance) to the node running the protocol and then to integrate them into its local view or to initiate exchanges. More precisely, distance (e.g. Euclidean) can be considered in this case at two levels: either when sampling to obtain peers constituting the view; or when selecting peers at the view level to trigger a process of profile exchange in order to improve the quality of the view or to simply broadcast a message in the system. Geographical information is also relevant for the optimisation of routing systems (sometimes using gossip protocols) in the context of sensor networks where the network nodes are in principle limited in terms of energy resources. In the literature, several works have combined geolocation and peer-to-peer systems: Dimakis and al. [19], Fiorese and al. [20], Geokad [21], Vicinity [22], GeoSpray [23].

1.3 Motivating scenario: data sharing at a local event

We target a scenario where users form a P2P network such that anyone can generate information and make it available to everyone else on the network. To illustrate this scenario, let us consider the real case of the International Jazz Festival of Saint-Louis in Senegal. It is an annual event during which thousands of people are meeting in different locations around the city to celebrate Jazz music and enjoy the various concerts and cultural activities held for this purpose. During this event, the constant need to access and share information related to the schedule and their contents is an important point for improving the quality of the festival.

Mobile access to the Web of Data: Khadim, Thierno and Guirane are friends who came to attend the festival. Khadim is interested in concerts at Abdoulaye Wade Square and festivities held mainly on the edge of the Senegal River that borders the city. Thierno and

Guirane came specially for different quintets' show on Faidherbe Square. Their smartphones are assumed to be equipped with the application dedicated to this event. When they arrive at Saint-Louis bus station, they detect the wifi network in the area dedicated to the festival and get connected to it. Thierno and Guirane are then automatically integrated into clusters located in the area in relation to their points of interest. As for Khadim, being the only one interested in African Jazz, he is integrated in an empty cluster for this point of interest and in another cluster related to the river. Everyone wants to have information about their interests, so they query their applications. Thierno and Guirane receive identical information since they belong to the same clusters. Khadim receives information about festivities held near the river. He also receives information on concert planning. This information originates from a cluster located at Abdoulaye Wade Square. Their friend Samuel who lives in Saint-Louis has also been integrated into this cluster.

Local access and collaborative modification: Samuel has set up his profile by specifying his interest for concerts at Abdoulaye Wade Square. By visualizing the schedule of these concerts, he notices that the name of one of the artists was incorrectly written on the system, he decides to modify this information. A local band decides to informally join the festival by playing at the entrance of the bridge (of the city) and they add their event to the shared data. Khadim decides a few minutes later to refresh the data presented to him and finds that the artist's name has indeed changed and that a new event was added.

1.4 Methodology

1.4.1 Keywords and terms selection for the survey

We indicate here the keywords and the expressions and synonyms used to search for scientific papers to be considered in preparing the state of art. These queries and variations were the seeds to create the corpus of related work we studied.

Keywords: Semantic Web, gossip protocol, peer-to-peer system, mobile access, Data Web, graph replication, limited connectivity, semantic overlay, mobile contributor.

Search terms: "Semantic profile" or "semantic clustering" or "Semantic data exchange" and "gossip protocol" or "random peer sampling" or "membership management" and "peer-to-peer system" or "peer-to-peer membership" and "mobile access" or "local access" or "local access" or "local data" or "local data replication" and "decentralized caching" or "client cache" or "local cache" or "distributed cache" and "geolocation" or "geographical location".

1.4.2 Online sources

We indicate here the main sources used to produce this document:

- `DBLP`: dblp.uni-tier.de
- `HAL`: hal.archives-ouvertes.fr
- `ACM Digital Library`: dl.acm.org
- `IEEExplore`: ieeexplore.ieee.org
- `Google Scholar`: scholar.google.com
- `Mendeley`: www.mendeley.com
- `CiteSeer`: citeseerx.ist.psu.edu

1.4.3 Document selection criteria

We now present the different criteria considered when searching for documents. Our choices are justified by the orientation of the papers and led to the following inclusion and exclusion criteria.

Main Inclusion Criteria:

- Papers dealing with the design of gossip protocol
- Papers dealing with the problem of random sampling in gossiping
- Papers proposing data exchange systems based on gossip protocols
- Papers dealing with semantic data exchange on peer-to-peer architectures
- Papers dealing with local access to data sources
- Papers written in French or English
- Papers dealing with graph replication approach
- Papers dealing with data sharing systems
- Papers published in conferences or journals, short papers, workshop papers and research reports that address the points mentioned above
- Papers less than 5 years old (priority)

Main Exclusion Criteria:

- Papers that do not deal with peer-to-peer communication or local access to data
- Slides of presentations, unpublished studies
- Papers dealing with semantic Web techniques but not applied to peer-to-peer architectures
- Papers that present gossip mechanisms but are not dedicated to data exchange in physical environments
- Papers that offer local access solutions on non-mobile devices
- Papers dealing with gossip protocols on centralized peer-to-peer architectures

1.4.4 Design and evaluation logic

For the design of our solution we build on the previous phase which consists in making a state of the art of approaches related to our objectives. From there, we make a set of hypotheses from which we identify different possible approaches (Architecture 1 and Architecture 2 in chapter 4). We then make a theoretical analysis of these approaches. This analysis allows us to distinguish the advantages and disadvantages of each method. This will guide us towards the best method that will be most adapted to our context. We then draw up an evaluation plan with the different metrics to be considered.

The next phase is dedicated to the technical aspects. This phase allows us to design our solution based on the technical aspects studied in the state of the art to which we bring new proposals adapted to our objectives. We then implement our solution for an experimental evaluation phase. We set up a simulation environment with real data sets.

The last phase is dedicated to the execution of the experiments and the analysis of the results obtained. Here we analyse our results in relation to the different metrics retained during the theoretical analysis. Then we compare them to the results of other known approaches in the literature. This finally allows us to judge the relevance and efficiency of our proposal.

1.5 Scientific Contributions and Publications

The results of this thesis have been published in different international venues:

- `Mahamadou Toure, Fabien Gandon, Kaladzavi Guidedi, Christophe Guéret, Moussa Lo, Pascal Molli.` Comparaison des Modèles et Architectures pour un Accès Mobile Restreint et Local au Web de Données : Un état de l'art des architectures et solutions envisageables. [Rapport de recherche] RR-9121, 2017, INRIA Sophia Antipolis. pp.81

- `Mahamadou Toure, Fabien Gandon, Kaladzavi Guidedi, Christophe Guéret, Moussa Lo, Pascal Molli.` Models and Architectures Comparison for a Restricted and Local Mobile Access to the Web of Data: A State-of-Art of Architectures and possible Solutions (Paper under Review)

- `Mahamadou Toure, Kaladzavi Guidedi, Fabien Gandon, Moussa Lo` Towards an Architecture and Model for Restricted and Local Mobile Access to Web of Data CNRIA 2019, Jeunes Chercheurs

- `Mahamadou Toure, Kaladzavi Guidedi, Fabien Gandon, Moussa Lo.` MoRAI: Geographic and Semantic Overlay Network for Linked Data Access with Intermittent Internet Connectivity. 2020 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)

1.6 Outline of this Thesis

This document is composed of the following chapters:

- `Chapter 2`: this chapter aim to survey the existing solutions (models, architectures,...) that can contribute to solving the problem of the intermittent access to the Web of Data by mobile contributors. These solutions are discussed here in relation to the network architectures models. We present a conceptual study of peer-to-peer solutions based on gossip protocol dedicated to the design and management of connected overlay networks. We also present a comparison of the functional approaches of gossip protocols based on the underlying adhesion mechanism. For each of these approaches we also identify a set of existing applications and the future research trends we consider relevant.

- `Chapter 3`: the chapter present a classification of existing approaches dedicated to designing data sharing systems adopting an RDF data model. We describe here these approaches taken from structured and unstructured architectures with the aim

of identifying their effectiveness in accessing the Web of Data despite connectivity and hardware resource constraints.

- `Chapter 4`: we present here a set of mechanisms that can lead us to mobile and restricted access to Linked and shared peer-to-peer data. The chapter provides a theoretical analysis of the approach we wish to implement with a global view on the technical aspects that could be adopted, in particular in relation to the network architecture based on gossip protocols and the data model oriented on RDF graph replication on a decentralised architecture.

- `Chapter 5`: this chapter presents MORAI (Mobile Read Access in Intermittent), a distributed peer-to-peer architecture organized in three levels dedicated to RDF data exchanges by mobile contributors. We describe MORAI's logical architecture dedicated to environments with limited Internet access and hardware resources and ensuring participant connectivity and data availability. The conception of a geographical overlay network based on peer locations to consider geographically close neighbours is also presented here.

- `Chapter 6`: here we aim to present MORAI's experimental evaluation. The chapter describe our simulation environment, compare and evaluate different approaches from the state of art and from our own design. We also present and analyse the results obtained.

- `Chapter 7`: this last chapter will review our contributions and describe some of the perspectives opened by our approach as well as the remaining challenges.

Chapter 2

Peer-to-peer and Semantic Data Exchange

In this chapter, we aim to survey existing solutions (models, architectures, protocols, etc.) that can contribute to solving the problem of intermittent access to the Web of Data by mobile contributors. These solutions are discussed here in relation to the network architectures. Our contribution here is a comparison of the functional approaches of gossip protocols based on the underlying adhesion mechanism. For these approaches we also identify a set of existing applications and future research trends that we consider relevant.

The chapter is organized as follows: Section 1 presents a general overview of gossip protocols. Section 2 discusses gossip protocols with basic membership mechanisms that can adopt random or deterministic approaches. Gossip protocols with two overlay membership mechanisms are described in section 3. Section 4 concludes the chapter with a summary of the main conceptual aspects of gossip protocols and research orientations that may be useful and relevant for further works.

2.1 Introduction: a contribution based on peer-to-peer resilient architecture

Research on RDF data exchange on peer-to-peer networks has made great progress. Many of these progress, particularly those dedicated to dynamic architectures, are based on gossip based (or epidemic) protocols. These protocols have an interesting approach in the sense that they are very resistant (having an intrinsic redundancy degree allowing to mask network and node failures) and scalable (load distribution across all system nodes) [1]. The operating principle is conceptually as follows: when a node (contributor) needs to send information via the network, it randomly selects t nodes among its neighbors and transmits them the message (t is a parameter named *fanout*). Each node repeats this process once when it receives the message.

Challenges and operating principles of these protocols are very diverse. They are very suitable for designing peer-to-peer communication systems because of their scalability, ease of deployment, but above all their resistance to network and process failures. They have been successfully applied in several areas: aggregate calculation (mean, variance, minimum, maximum, etc.) [13–15], load balancing [16], network management [17]. The shared characteristic of such protocols is that each node periodically or reactively exchanges information with certain of its peers. A gossip protocol assumes the existence of an underlying membership mechanism, a fundamental component, which provides each node a complete or partial system knowledge. This consists of a list of node identifiers or profiles commonly called a *view*.

The costs in terms of memory and network traffic to ensure overall system knowledge are generally elevated for dynamic networks where nodes continuously leave and join. These constraints lead us to mechanisms that provide partial knowledge of the system. These mechanisms are more adapted to the dynamic characteristics of the system and are less constraining in terms of resources. Several gossip protocols were then implemented on top of these mechanisms. The latter can be classified into two families [1]: basic membership mechanisms and two-layer membership mechanisms.

2.2 Gossip protocols with basic membership mechanisms

With this type of mechanism, the view owned by each node is composed of peers dispersed across the network. No characteristics are considered on a node to integrate it into another node's view. Two system models can be identified among these protocols [1]: centralized systems that use a set of central servers whose main task is to provide each node with a random view, and decentralized systems whose main characteristic is self-organization. We will focus here on protocols that adopt decentralized architectures that avoid in particular problems related to storage memory size and central server failures (Figure 2.1: On each node, the outgoing arrows indicate the neighboring nodes that make up its local view. The incoming arrows represent the views of the neighboring nodes in which the node is integrated). These protocols can be divided into two groups: (1) protocols using a random selection of peers at runtime and (2) those using deterministic selection.

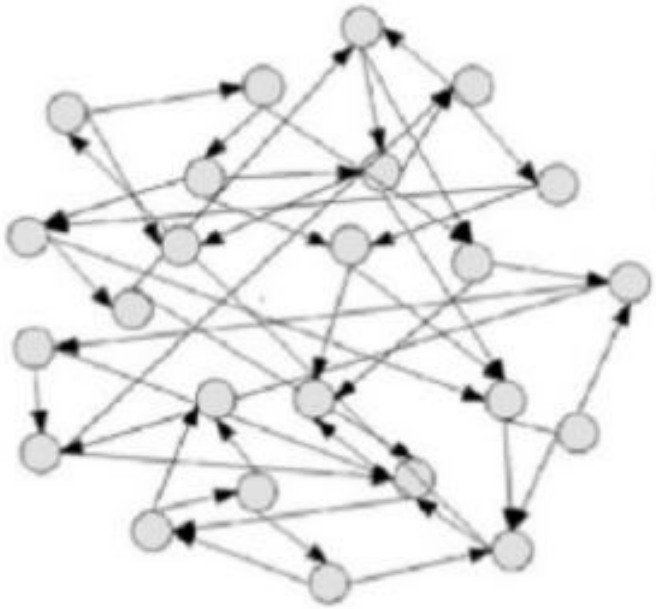

Figure 2.1: Example of a basic membership mechanism [1].

2.2.1 Protocols with random peer selection approach

These gossip protocols have the particularity of being totally based on random choices. More precisely, at runtime, the peers selection for information dissemination is done randomly among the peers composing the node view. All the nodes composing the view have in principle the same probability of being requested for the exchange. This keeps the architecture connected and close to the random graph properties. The challenge is therefore to build and maintain the graph connected even with node failures. Choosing the *fanout* parameter to be applied to the protocol is decisive in the sense that it has a particular impact on the speed of message propagation. Hence the importance of finding a good value for the *fanout*. The following work is based on this principle.

Ganesh et al. [24] presents SCAMP, a peer-to-peer membership service that operates in a completely decentralized manner, where no node has any overall knowledge of members. The system is totally self-organized, the size of local views naturally converges to the "right" value for gossips to succeed. This value depends on the size of the system.

Jelasity et al. [16] introduces a generic scheme, generalizing gossip protocol based on sampling services. Sampling services aim to give each node a set of peers with which to exchange information. Authors show that unstructured dynamic overlay networks based on gossip protocols are natural candidates for the implementation of sampling services for their reliability and scalability. Two methods are presented in the Peer Sampling Service API: Init() and GetPeer(). Init() starts the service for a given node if it was not previously done. GetPeer() retrieves a peer address when the group includes several nodes.

Jelasity et al. [25] introduces a generic framework implementing decentralised peer-sampling service by building and maintaining unstructured dynamic architectures based on information about peer contributors. The basic principle underlying the proposed framework for designing the peer sampling service itself relies on gossip paradigm. In fact, each node manages a local small table providing a partial view on the entire set of nodes and periodically updating it by a gossip process.

Leitao et al. [26] presents HyParView, a membership protocol to support gossip broadcasting that ensures elevated reliability levels, despite high node failure rate. HyParView

is based on an approach based on the use of two distinct partial views: small active view, and a bigger passive one. All nodes' active views build together an architecture used for message transmission. The purpose of the passive view is to maintain a list of nodes that can be used to substitute active view's failed nodes.

Bortnikov et al. [27] presents Brahms, a random peer sampling algorithm for large dynamic systems that are pronned to malicious behaviour. Brahms provides a view to each and also overcomes Byzantine attacks (e.g. attacks where adversaries have full control to certain authentic nodes from which they disrupt the network).

2.2.2 Protocols with deterministic peer selection approach

These protocols differ from random selection protocols in the deterministic nature of their peer selection procedures during data exchange. In previous protocols, although the architecture remains effectively connected despite its dynamic nature, peer-to-peer links are by default meaningless. In the case of a deterministic choice, the selection is guided by the application of mechanisms based on characteristics such as peer "age" (time spent in the view), metric distance, similarity (shared interests, semantic proximity, profile, etc.), scheduling (Round Robin), etc. This allows to have virtual but relevant links between peers in the architecture, thus improving the quality of the views. In this category, the following related works have attracted our attention.

Voulgaris et al. [17] describes CYCLON, a low-cost, full membership management framework. CYCLON improves on the basic shuffling protocol [28]. Shuffling operation is a swapping process of a neighbour subset among a couple of nodes. It originates from any of these two nodes. The basic shuffle protocol guarantees that overlay connectivity remain intact until membership changes. CYCLON uses a similar design like the basic shuffle. However, nodes do not randomly select the neighbour with whom to exchange informations, they choose the oldest neighbor.

Nedelec et al. [29] proposes a random peer sampling protocol called Spray, based on Scamp and CYCLON. The protocol is designed to avoid the constraints introduced by

WebRTC framework. WebRTC allows communication channels between browsers to be established. However, it does not manage addressing and routing. Browsers connect by exchanging offers and receipts using a shared mediator like couriers, specialised signalling services, available WebRTC links, etc. Spray avoids these constraints through its three-part connection establishment procedure, by using only neighbor-to-neighbor interactions. Spray: 1- adapts dynamically each peer's neighborhood. So, connection volume logarithmically increases with network size; 2- only uses interactions between neighbours to establish connections. Thus, connections are established in constant time; 3-quickly converges towards a topology with similar properties as a random graph. As a result, the network gains robustness against large-scale failures and efficiently disseminates information.

In **[30], Alromih et al.** proposes EEGossip, an Energy-Efficient Gossiping protocol to route data towards sink. Using a selection procedure, the protocol determines the best path for each neighbor. The selection function uses the next node residual energy, next node distance (the distance between the current node and its neighbor) and the sink distance (the distance between the sink and the next node). For the calculation of the distance, EEGossip uses the Chebyshev distance [31] which overcomes the Euclidean distance regarding both processing complexity and execution time [32].

2.2.3 Synthesis on basic membership mechanism

In summary, gossip protocols adopting a basic membership mechanism have several advantages for the decentralized peer-to-peer systems. They make it possible to maintain the architecture connected despite the mobility (arrival/departure) of the nodes. The failure resistance of nodes and network is ensured in particular by the level of data redundancy. The amount of information passing through the architecture can be controlled by choosing the size of the views, the *fanout* parameter, the type of protocol execution (cyclic or reactive) and also the type of message propagation (push/pull). WebRTC also has an important advantage for these protocols through its signalling and connection services. It enables gossip protocols to be deployed on mobile phone or tablet Web browsers, enabling

direct/indirect connections between mobile users. Table 2.1 provides a summary of the different gossip protocols surveyed in this section. The following criteria were used to compact their comparison in one table:

- `Push-propagation`: the protocol is based on a push stream. In this type of propagation, nodes transmit data to randomly selected neighbors without expecting responses from them. The mechanism is very suitable for disseminating information as it is unnecessary to have receivers responding to originators.
- `Pull-propagation`: the protocol is based on a pull stream. Pull propagation guarantees data is sent when required. This allows to reduce network load whenever data size is important.
- `Cyclic Execution` : the protocol is executed in a cyclic manner at each peer level.
- `Biased peer selection` : the choice of a peer when executing the protocol is made in a deterministic way. A selection mechanism is applied to select the most suitable node (oldest (age), closest (metric distance), similarity metric, etc.)
- `Defense mechanism` : the protocol integrates one or more security mechanisms to maintain the architecture connected and/or preserve the anonymity of each peer.
- `WebRTC` : the protocol is based on the WebRTC framework which allows communication channels between browsers to be established.

Table 2.1: Summary of basic gossip protocols.
The total support of a characteristic is indicated by the sign √,
the partial support by the sign ∘

	Scamp [24]	CYCLON [17]	HyParView [26]	Brahms [27]	Spray [29]	EEGossip [30]
Push-propagation	√	√	∘	√	√	
Pull-propagation			∘	√	√	√
Cyclic execution		√	∘	√	√	
Biased peer selection		√			√	√
Defensive mechanism				√		
WebRTC		√			√	

2.3 Gossip protocols with two overlay membership mechanisms

A second family of protocols is designed on top of the membership protocols described in the previous section by adding clustering mechanisms to form two-level architectures (Figure 2.2). Here, the role of basic gossip protocols is to build and maintain a connected architecture on top of which an appropriate clustering mechanism is then applied to build even more efficient overlay networks.

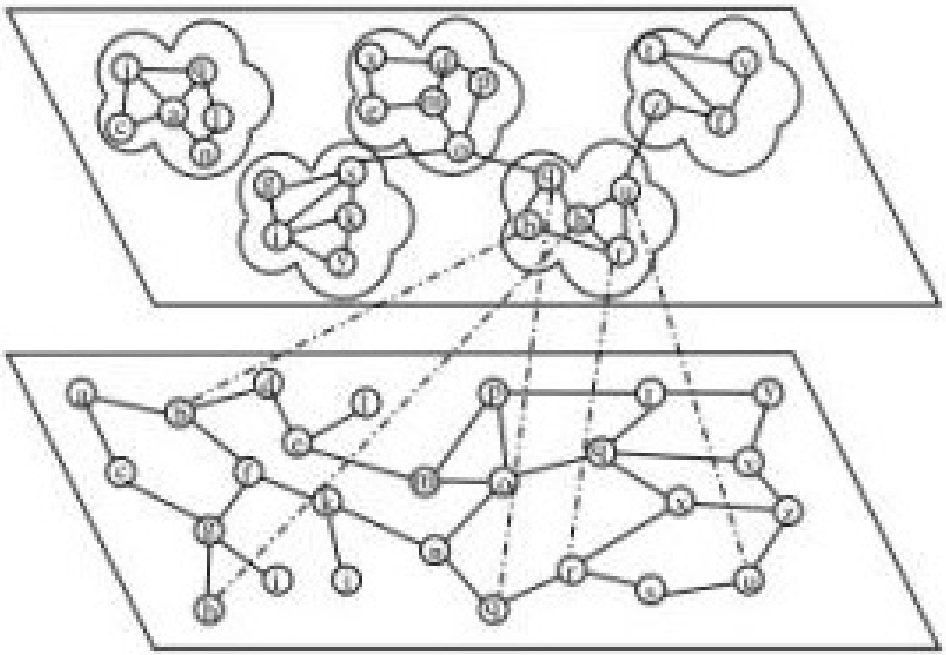

Figure 2.2: Example of two overlay membership mechanisms [2].

The common objective of these architectures is to cluster nodes according to a geographical, semantic, profile or network proximity criteria and take this proximity into account in order to provide participants with local neighbor lists exclusively made of members of same cluster. Some models offer mechanisms that allow a few selected nodes in the cluster to be equipped with a remote view composed of nodes from other clusters in order to keep the entire system connected (Figure 2.2). Kermarrec et al. [1] show that this connectivity

is achieved through a limited set of links amoung clusters.

2.3.1 General conception approaches

Several models of two overlay protocols have been proposed. These protocols have the particularity of being completely autonomous, in a way that all the nodes have the same roles. The architecture adhesion procedure for a newly arrived node does not require contact with a particular node (contact server). Each node can be used as a contact for a new one. The membership mechanisms differ from one protocol to another. The aim is to find a first peer who responds favourably to the membership request. The latter peer will transfer the information relating to his view according to the type of flow (push and/or pull) adopted by the protocol. This information will allow the node that initiated the request to initialize its view. On a number of these protocols, clusters are also formed by adopting the operating principle of the basic gossip protocols. This is the case of Gossple, Vicinity and Behave [2, 22, 33]. To form the second overlay, exchanges between peers are performed according to the gossip model guided by the clustering metric considered. In Gossple, Behave, and Cyclades for example, the cluster is represented by a second view held by each node. This second view is made up of the best neighbours selected on the metric basis. Clusters are improved as they are updated by reactive or cyclical update operations.

A number of these solutions also include caching mechanisms to speed up data access [22, 33, 34]. This type of mechanism allows relevant content to be temporarily stored on the architecture. Thus, all local caches form a common cache accessible to system contributors. A partial/total caching of the data passing through the system is performed by these contributors in order to process a large part of the requests from the local cache or from the caches of neighboring peers.

2.3.2 Selected well-known tools

Voulgaris et al. [35] proposes a proactive approach to build epidemic protocol-based semantic overlay gathering same content peers. Each peer manages a semantic neighbor list and first queries its semantic neighbors to find a file. The model assumes a semantic proximity function that provides numeric semantic proximity metrics among peers regarding their lists of files (type of files: music,video, documents etc).

Jelasity et al. [36] proposes T-MAN gossip protocol which allows to build a variety of network architectures. T-MAN is based on a peer sampling service [25] generating a starting network architecture of random links. T-Man primarily depends on three parameters: message size m, sampling parameter w and ranking method $RANK$. Nodes rank their descriptor set (composed of descriptors from random links) with the ranking method and pick first k as neighbours. This results in a structure named *the target graph.*

Mordacchini et al. [37] proposes a general system architecture dedicated to take advantage of collaborative peer-to-peer information exchange. The idea is to gather similar users and disseminate relevant recommendations across them. The protocol uses a clustering mechanism to group similar users. Each peer firstly independently determines which peers they are linked to. These individual connections are selected based on interest-based metric distance, among the encountered peers. Each time it meets a new peer, it learns from and communicates with potential new neighbours (i.e. similar users). When this process is stabilized, a peer may consider his neighbourhood as a representation of a community of common interest.

Bertier et al. [2] describes the construction of an anonymous social knowledge network using a gossip protocol called Gossple. Periodically, Gossple nodes exchange on their interest and calculate their interest-related distances. They propose to improve navigation in Web 2.0 systems through implicit personalization: an anonymous network of knowledge interested by similar topics is associated with each user, independently from the way they expressed their interests. There is an implied use of this knowledge by users in order to drive and refine searching actions.

Voulgaris et al. [22] presents a self-organizing, generic overlay management framework,

VICINITY. Given the node descriptor p and a set of node descriptors D, a select function *SELECT(p, D, k)* is applied, to return the set of k descriptors that are most closely related to the outgoing links from p in the target structure. Such function is often built on a proximity-specific measurement globally defined. The protocol relies on two layers. The base layer is the peer sampling service. It is in charge of keeping the architecture connected and of periodically providing the upper layer with candidate nodes. Candidates are randomly and uniformly sampled throughout the system. The upper layer protocol, named VICINITY, determines which nodes to foster by use of the selection function.

Frey et al. [33] presents Behave, a decentralized caching architecture based on behavioral positions and using gossip protocols to construct superposed clusters with similar interest peers. Behave nodes adopt a gossip protocol based on the similarities between their navigation histories to form an interest-based topology. Each node thus has a set of neighbours whose browsing history is closest to their own. From this topology, Behave's behavioral cache appears as the merging of a node's neighbors' local caches.

Folz et al. [34] presents CyCLaDEs, a network architecture based on LDF (Linked Data Fragment) similarities. Using LDF client similarities, CyCLaDEs intends to provide a decentralized behavioral cache for LDF query processing. A predefined number of best customers is identified for each customer and a one-to-one link is established with each one. In case of a given user's process, first the local cache is checked for each sub-request triple, followed by its neighbors' cache, and if necessary, the LDF server. The network architecture relies on random peer sampling model for member composition management and a clustered architecture to handle the k-best neighbors.

Boutet et al. [38] presents HyRec, a scalable and cost-effective online system dedicated to customizing user-centric collaborative filtering. HyRec loads recommendation tasks on users' Web browsers, while the process is driven by a server which also controls user profile relationships. Each user receives from the HyRec server a set of candidate profiles. Each browser then calculates the KNN (k-nearest-neighbor (KNN) or k-best neighbors,) of its user and the most relevant elements based on this sample. Using a sampling approach, both KNN selection and article recommendation are delegated to users' Web browsers. The sampling approch follows the same principle as gossip protocols.

Carvajal et al. [39] introduces a WebRTC-based library called WEbGC. WebGC enables Web browsers to communicate using gossip protocol and to interact with node-JS applications. WebGC is also based on the SimplePeer framework, which operates as a JavaScript library and serves as a layer for WebRTC facilitating peer-to-peer data connections.

Nedelec et al. [40] presents CRATE, a real-time decentralised collaborative editor. CRATE directly operates on Web navigators using WebRTC. By using an optimistic replication process, CRATE ensures documents accessibility and responsiveness. For the browser network construction, CRATE uses SPRAY random peer sampling protocol with WebRTC technology. With SPRAY, a locally based neighborhood table is provided to editors allowing communication within a subset of editors. CRATE uses SPRAY protocol to evolutively broadcast any replicated sequence operations to all co-workers.

Pilet et al. [41] introduces a peer-to-peer protocol enabling user privacy protection during decentralized averaging. Many limitations of existing solutions, such as eavesdropping attacks, restrict peer exchange coordination, or use of expensive cryptographic primitives such as homomorphic encryption, are overcome by the protocol. The protocol relies on an attack-resilient Random Peer Sampling (RPS) service: Brahms. Each peer is provided with a sample of remaining network peers by the RPS. During several rounds, the protocol exchanges noise before starting to transmit actual data. This makes it hard for an honest but curious attacker to know whether a user is transmitting noise or actual data. This attacker is considered honest but curious in the sense that he observe exchanged values, but he do not inject fake values or try to prevent the algorithm from computing a correct result.

Meiklejohn et al. [42] presents PARTISAN, an actor's application operating platform that enhances evolutivity and decreases latency. In [43], an actor is defined as an computational entity that, in response to a message it receives, can concurrently: send a finite number of messages to other actors, create a finite number of new actors, designate the behavior to be used for the next message it receives. Actor application is one in which actors have the option to stay on a variety of nodes and be able to seamlessly interact with other nodes' actors. PARTISAN offers greater evolutivity by giving developers the ability to define the used network tier at execution point while not modifying application

semantics. PARTISAN offers four overlays to developers, including a peer-to-peer overlay based on HyParView [26] and Plumtree broadcast protocol [44]. The peer-to-peer protocol allows PARTISAN to provide both membership processes used to add/delete members from cluster and sending processes for asynchronous messaging.

Leonardi et al. [45] presents NAPA-WINE a P2P television system (P2P-TV) architecture. P2P-TV is defined as a system in with a source dividing video stream in pieces of data, exchanged with nodes and then distributed to all network members. The goal of NAPA-WINE is to push chunks into layer where peers collaborate to broadcast them, with no requirement for huge resources and bandwidth to sustain the service. Gossip protocols such as Newscast [16] or CYCLON [17] are used for the underlying topology management. This allows peers discovery in overlay topologies mapped over the network.

2.3.3 Synthesis on overlay membership approaches

These two-overlay protocols benefit from the advantages offered by the basic protocols on top of which they are built. Therefore the architecture is connected and resistant to failures. The second overlay is particularly useful for creating links between nodes that make sense in relation to the context. It therefore reduces latency time during search scenarios. On certain solutions, the clustering algorithm is executed at the same time as the the underlying base protocol. This limits the amount of information transiting on the architecture. Table 2.2 provides a summary of the different solutions presented in this section. The following characteristics allowed us to compare these different models:

- `Similarity metric` : the model uses a similarity metric to estimate the distance (in terms of interest, metric, semantics or content) between peers;
- `Clustering by gossip` : the clustering algorithm used is based on the same mechanism as a gossip protocol at runtime;

- `Compression of exchanged data` : a compression strategy is applied on data to maintain fluid exchanges in the system;
- `Caching` : the model integrates one or more caching techniques for data storage and thus ensures availability and local access;
- `Semantic clustering` : the clustering algorithm is based on semantic criteria; more precisely, the similarity metric used estimates the semantic distance between peers;
- `Membership Management Protocol (MMP)` : this is the base protocol (gossip protocol) used in the model to build and maintain the architecture's connectivity;
- `Contact server` : the arrival of a new peer in the architecture is done through a contact server whose main role is to store all or a part of each peer's information and to provide each newcomer with the necessary data for its integration (view, IP address, identifier, etc.).

Table 2.2: Summary of the two overlay membership protocols. Full support for a feature is indicated by the sign √.

	Voulgaris [35]	T-man [36]	Mordacchini [37]	Gossple [2]	Vicinity [22]	Behave [33]	HyRec [38]	Webgc [39]	Cyclades [34]	Crate [40]
Similarity metric	√		√	√	√	√	√	√	√	
Gossip clustering			√	√	√	√		√	√	
Double view	√			√	√	√		√	√	
Data compression				√		√				
Caching	√					√			√	
Semantic clustering	√	√						√		
Used MMP	CYCLON [17]	Jelasity [25]	CYCLON [17]	Brahms [27]	CYCLON [17]	Brahms [27]		CYCLON [17]	Brahms [27]	Spray [29]
Contact server							√		√	√

2.4 Conclusion: gossip protocols in overlay networks conception

Our objective in this chapter was to survey the existing solutions that could contribute to solve the problem of intermittent access to the Web of Data by mobile contributors. We confirmed that gossip protocols offer well adapted approaches to the design and maintenance of decentralized and dynamic peer-to-peer architectures. Consequently, our prospection was guided by the analysis of solutions based on gossip protocols dedicated to the design and management of peer-to-peer overlay networks.

We distinguished two architectures types. The first *(gossip protocols with a basic adhesion mechanism)* refers to basic systems dedicated to designing and maintaining the connectivity of the underlying architecture on which the various exchanges between peer members are based. The second type of architecture *(gossip protocols with two overlay adhesion mechanisms)* is dedicated to the automatic formation of clusters of interest or proximity aimed to group together peer members of the architecture that have one or more common interests or that are close according to a given similarity metric. These two overlay systems are built on top of basic protocols. We also note the possibility of taking into account the location of peers when selecting neighbours [22]. The neighbourhood will then be formed by semantically close peers, having a certain geographical proximity, i.e. a maximum limit in terms of geographical distance is set between the peers considered as neighbours. Based on parameters such as *fanout, view size,* and *node degree,* gossip protocols are generally evaluated by comparing the different architecture behaviors at runtime. Protocol properties such as *failure resistance, convergence time, load distribution and graph properties (clustering coefficient, average of shortest paths, balanced distribution)*, are analyzed and interpreted following test scenarios on simulation platforms such as PeerSim [46].

The works surveyed in this chapter present many trends relevant for future work on gossip protocols. Among them, we especially identified: the consideration of the geographical parameter, in particular by applying a deterministic selection during sampling and exchanges; the integration of the adaptive *fanout* to take into account the evolution of the

architecture size; the implementation of JavaScript solutions to facilitate the deployment of protocols and peers inside Web browsers; and the exploitation of the advantages of explicit friends' social networks.

Chapter 3

Data graph sharing and collaborative modification

This chapter focuses on collaborative data sharing systems. Here, we do not look at approaches related to the semantic Web in general, but more specifically we look at work related to collaborative sharing and modification. Our contribution in this chapter is a classification of approaches dedicated to designing data sharing systems adopting an RDF data model. For each of these approaches we also identify a set of existing applications and the future research trends that we consider relevant.

The chapter is organized as follows: Section 1 classifies the different approaches. In Section 2, we present RDF graph structuring, storage and querying system. Section 3 discusses replication approach. Sections 4 and 5 describe respectively graph distribution systems over structured peer-to-peer and semantic overlay networks. In section 6 we present approaches applied by cloud and Fog computing systems. In sections 7 and 8 we present respectively general syntheses on all these approaches and the conclusion of this chapter.

3.1 Introduction: classification of approaches for distributed and collaborative sharing and modification.

Much work has been done to build infrastructures consisting of centralized nodes, dedicated to data sharing and capable of ensuring the availability and consistency of these data. These client-server infrastructures of the Web of Data allow the implementation of light clients for users, thus transferring processing and calculation loads to the servers. They are also very suitable (compared to decentralized infrastructures) in terms of speed when searching for information on large amounts of data. Central nodes can cooperate to share knowledge. But the major difficulty in this method is that query results may change over time, depending on node availability. If a node becomes unavailable, it is not relayed by any other node in the network. Thus, any node can be a point of failure for the system [47].

To overcome these centralized access-points that could become bottlenecks, several approaches in the literature consider distributed scenarios based on peer-to-peer networks. The network provides high resistance to technical failures of nodes. These peer-to-peer systems offer many advantages of decentralized distributed systems but suffer from problems related to the availability and reliability of sources and data. To overcome these constraints, decentralized peer-to-peer systems rely on mechanisms to mitigate the impact of disconnection scenarios on data availability, but also to balance the system's processing loads on all peers. In the case of decentralized RDF data sharing systems, these mechanisms allow a graph to be partially or totally replicated, shared and modified in a collaborative way through the peers involved in the architecture. To this end, several solutions have been proposed in the literature adopting each of the following principles, which are conceptually different:

- `Graph replication`: the graph is stored on all peers in the network that can modify and/or delete portions of it independently. These addition or deletion operations will then be propagated through the architecture using variable techniques to ensure the

consistency of the different copies.

- `Distributed graph and structured peer-to-peer system (case of DHTs)`: the graph is distributed across all peers on a structured peer-to-peer environment (a DHT). In this case, requests are routed directly to peers holding the requested data.
- `Distributed graph and semantic overlay`: the graph is distributed over all participants by adopting a semantic peer-to-peer network architecture. Also here, requests are routed to neighboring nodes and nodes belonging to other clusters if necessary.
- `Cloud and Fog computing`: the graph is hosted on the cloud. A collaborative cache on browsers is then set up on the architecture to overcome disconnection scenarios.

3.2 RDF Graph : structure, storage and querying

The Resource Description Framework (RDF) is a framework for representing information about resources. Resources can be anything, including documents, people, physical objects, and abstract concepts. RDF 1.1 Concepts and Abstract Syntax [48] defines an abstract syntax (a data model) which serves to link all RDF based languages and specifications. The core structure of RDF data model consists of entities, represented by unique identifiers, and binary relationships, or statements, between those entities. In the graph-based representation of an RDF statement, a triple *subject-predicate-object* element is the atom of knowledge. It is a relationship in which the source is called the *subject*, the labeled arc is the *predicate* (also called property), and the destination is the *object*. Given disjoint and infinite sets I, L and B denoting the IRIs, literals and blank nodes respectively, an RDF triple is any element of the set $(I \cup B) \times I \times (I \cup L \cup B)$. Therefore a RDF graph G can be formally said to be a set of RDF triples.

3.2.1 RDF Data Structure

In an RDF graph, different entities are vertices in the graph and relationships between them are represented as edges. For example in Figure 3.1, edges in the graph indicate that the entity "Messi" is of type "footballer", was born in Rosario, and plays striker for FC Barcelona. Each of the entities that "Messi" is connected to in this graph can have their own set of connections; for example, FC Barcelona is shown to be connected to the Barcelona entity through the region relation. Information about an entity is represented by directed edges emanating from the vertex of that entity (labeled by the property type), where the edge connects the vertex to other entities, or to literal vertices that contain the value of a particular attribute for that entity.

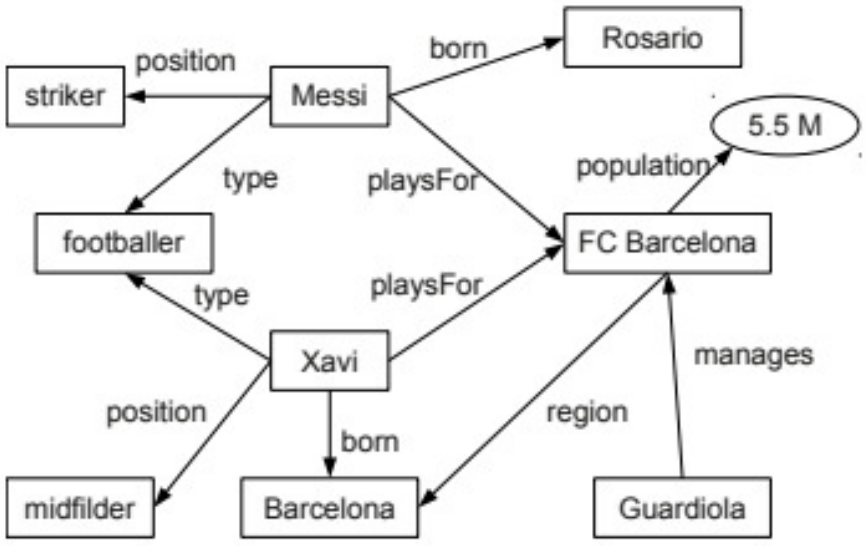

Figure 3.1: Example of RDF Graph Data from DBpedia [3].

RDF Schema is a special vocabulary (within RDFS namespace), a meta-ontology, that supports the declaration of lightweight vocabularies by providing some elementary bases. It is common practice to distinguish an RDF Graph into a data part and a schema part. When looking at the storage [49] and querying [50] in databases, a natural distinction is made from schema definition and data statements.

3.2.2 RDF Graph Storage in collaborative sharing systems

RDF has different representations such as RDF/XML, N-Triples, N3, Turtle and JSON-LD. These syntaxes are generaly used for storing and exchanging RDF data; However, we address here software storage solutions known in collaborative sharing and modification approaches. [51–53].

Jena [51] is a Semantic Web Framework, offering a Java programming interface, a database system, and query languages (RDQL, SPARQL) [54, 55]. Jena's original design (Jena-1) used two alternative approaches to store an RDF Graph: (1) Three tables: one for statements, one for literals and one for resources. Here the main problem was the heavy use of joins to answer queries. (2) One statement table, with indexes by subject, by predicate and by object.

Sesame [52] is an architecture for efficient storage and expressive querying of large quantities of data in RDF and RDF Schema. The main feature of Sesame is that it provides query languages, and a subset of RQL which incorporate the RDF Schema semantics. The concrete data storage is implemented differently according to the underlying database system [52]: PostgreSQL and MySQL.

Virtuoso [53] is a multi-protocol server providing access to relational data stored either within Virtuoso itself or any combination of external relational databases. Virtuoso's initial storage solution is fairly conventional: a single table of four columns holds one quad, i.e. triple plus graph per row. Recent version offers three options for each table: partitioned, replicated or local. Partitioning is based on partition columns specified by the administrator, which are used for hash-based partitioning; partitions can also be replicated, if specified. Replication copies the full table to each machine, which can be used for query-based partitioning, or to store a global schema that is frequently accessed by queries. Local tables are only accessible to the individual machine, and are typically used for local configuration.

3.2.3 RDF Graph Querying in collaborative sharing systems

For approaches related to collaborative sharing systems, RDF data and RDF schemas can be considered at three different levels of abstraction [52]: at the syntactic level (e.g. XML documents), structure level (they consist of a set of triples) and semantic level (they constitute one or more graphs with partially predefined semantics).
Clearly, the requirement is to query at the semantic level (i.e: querying the full knowledge and not only explicitly stated assertions). Two options exist to achieve this goal: (1) Calculate and store the entailments in the graph base, (2) Enable inference of new statements via the query processor as required.

3.3 Graph replication : criteria and approach

Data replication methods are gaining popularity in peer-to-peer computer systems as a way to achieve high availability and reliability. Data replication is now a relevant design principle to ensure reliability, load balancing, and mainly the high availability of data and services in distributed systems. It improves the performance and availability of information sharing in a large-scale network [56]. Data replication consists precisely of storing in separate sites several data object copies. The graph is stored on all (or part of) the peers of the network which can modify and/or delete portions of it autonomously. These addition or deletion operations will then be propagated across the architecture to ensure the consistency of the different replicas. Consistency should be ensured between replicas once all operations are executed on all sites. This ensures a high availability of locally relevant data, especially in peer-to-peer architectures where data storage and processing is distributed among peers which may have dynamic behaviours.

3.3.1 Replication criteria

Replica control mechanisms can be classified according to three criteria [56]: where updates take place (single-master or multi-master), when updates are propagated to replicates (synchronous vs asynchronous) and how replicas are distributed over the network (full or partial replication).

- `Update placement`: For Single-master method, each reproduced object has only one primary copy. With this method, only one site is given reading and writing access whereas only reading authorization is granted to others [56]. This is often referred as the master-slave method for master nodes interaction with other "slave" nodes hosting a copy [56]. Centralizing updates on a single point make it easier to manage concurrent access. However, centralization introduces potential bottleneck and point of failure. Multi-master method allows several sites to hold a primary replica of a single object. These copies are all updatable. Each site is allowed to update its copies. Multi-master provides more agility over single-master, as in case of a master crash, another one may handle replicas [56].

- `Distribution of replicas`: For replicas placement, there are two basic approaches, full replication and partial replication [57]. With full replication, a shared object is copied to each member's site. It provides load balancing simplicity as all sites offer equal capacity. It also offers maximum availability since any site can replace any other site in case of failure. With partial replication, each site holds a copy of a subset of shared objects, so that the replicated objects may be different from one site to another. This approach requires less storage space on the sites and updates only spread to the sites concerned. Thus, these updates produce a reduced load for the network and sites compared to full replication [57].

- `Update propagation`: Updates can be propagated on a replica control system using a synchronous or asynchronous approach [58]. The synchronous update propagation approach applies changes across replicas within the update producing operations context. Therefore, once operation performed, all replicas have the same state [56].

The source node of the transaction (set of update operations) propagates the update operations in the context of the transaction to all other replicas before executing this transaction. There are several algorithms and protocols to carry out this process [59, 60]. Thus, synchronous propagation increases mutual consistency between replicas. With the asynchronous approach, the transaction is first executed at the local site and then updates spread to remote sites. Asynchronous propagation has the advantage that replica unavailability will not interrupt updating processes, thus improving availability. It is possible to classify asynchronous replication solutions as optimistic or pessimistic based on their assumptions about conflicting updates [61, 62]. Pessimistic asynchronous replication is based on predictions that updating conflicts will occur and introduce propagation methods to prevent it. In contrast, optimistic asynchronous replication is based on the positive hypothesis that conflicting updates arise rarely, if at all. Updates propagation is done in the background. Conflicting updates will be processed at the next stage, accepting a certain degree of divergences between sites. Most of these optimistic replication systems ensure mutual consistency between replicas [61, 63].

The degree of replication (complete or partial), as well as the source (single or multi Master) and propagation mode (synchronous or asynchronous) of updates in the system are fundamental characteristics for data replication systems.

3.3.2 Selected Graph Replication approaches

BAYOU [64] is a mobile data search solution allowing to reproduce a database on a computer, edit it offline and then synchronize to any other replicas. In Bayou, a single main site determines actions to execute or interrupt and informs other sites.

RDFGrowth [65] focuses on semantic data sharing where a single peer can modify the shared knowledge, while the others have the right to read. RDFGrowth targets a particular scenario where peers participate in interest groups to develop their internal knowledge on one or more thematic areas.

The concept of OT (Operational Transformation) [66–68] has been developed for collaborative publishers. OT is based on the principle that operations are directly executed at the local site and commands are then forwarded to other sites. Therefore, the same operations sequence is executed by all sites, possibly in different orders. OT's objective is to maintain operations intent and ensure replicas converge. This is done by using a rewrite rule for each simultaneous operations pair.

Skaf et al. [69] presents the first semantic peer-to-peer wiki, SWOOKI. SWOOKI's network is composed of a set of interconnected autonomous semantic wiki nodes that can join and leave the network dynamically. It is an unstructured and decentralized P2P system that requires no central coordination or knowledge. It is based on a symmetrical communication model where each peer can act as both server and client. Data management is based on optimal data replication where each peer hosts a copy of the wiki pages and the associated semantic data. Each peer can autonomously offer all the services of a semantic wiki server, access, search and requests are executed locally without any routing of requests on the network (full replication).

Weiss et al. [70] presents Logoot-Undo CRDT (Commutative Replicated Data Type) algorithm that incorporates the *undo anywhere, anytime* functionality. For reasons of efficiency and fault tolerance, the network content is replicated. This replication can be either total or partial. The Logoot-Undo approach belongs to the CRDT framework whose main idea is to provide real commutability between simultaneous operations. An operation is either an insertion, an update or a deletion. Operations are grouped into patches and sent to all other replicas for integration. Each patch is delivered once and only once for each replica.

In **[71], Spaho et al.** consider peer-to-peer systems with an architecture that includes super-peers. A peer group consists of several peers that may be geographically distant from each other but have a common objective. The work is comprised of tasks to be performed by group peers. Replicating peer group documents is a major task. Peer group has a central manager referred to as the super-pair. It assigns tasks to the group's peers and keeps track of how the work is being done. It facilitates interaction with other peer communities. Super-peer connects group peers with other network peers and super-peers.

If any portion or all of a peer's document evolves, remaining peers holding the document's replica will apply changes.

Ibanez et al. [72] presents SU-Set (SPARQL-Update Set), a CRDT (Commutative Replicated Data Type) for RDF graphs updated with SPARQL Update 1.1 operations. A CRDT is a type of data whose operations, when simultaneous, give the same result regardless of the execution order. This is an emerging formalism for optimistic replication. The authors design a CRDT for RDF graphs updated with SPARQL Update 1.1 operations, thus ensuring the possible consistency of a Live Linked Data (LLD) social network. The latter is considered here as a cooperative publishing system with low latency needs and no editing constraints. The CCI (Convergence, Causality, Intention prevention) coherence model of [67] is used. SU-Set extends the insertion and deletion operations of OR-Set [73] (Observed-Removed Set) to union and difference. In OR-Set, each inserted element is labelled with a unique identifier. This ensures that the elements stored in the payload are always unique, and therefore, added and deleted only once. SPARQL operations on the triples are performed by the user, then rewritten into SU-Set operations with pairs *(triple, id)* in a transparent manner for the user and sent forward to the other nodes, where they are re-performed at the reception.

Crate [40] is a decentralized, real-time collaborative editor that runs directly in Web browsers using WebRTC. To provide document availability and responsiveness, Crate follows the optimistic replication scheme. Each publisher reproduces the document locally and performs directly its operations. Then, the publisher distributes its changes to all other participants. The system is correct if editors with the same set of changes have convergent replicas to an equivalent state, i.e. users read the same document. This property is the strong eventual consistency [74].

3.4 Distributed graph and structured peer-to-peer system (case of DHT)

Significant progress has been made in the research effort to build high-performance RDF data management systems that run on a single machine. These systems have demonstrated high performance on a single machine for data sets containing millions, if not billions of triples [3]. However, since the amount of RDF data continues to evolve, it is no longer possible to store complete data sets on a single machine and be able to access the data with reasonable performance.

Partition-based approaches allow an RDF graph to be partitioned into several fragments and placed on different sites in a parallel/distributed system. Each site hosts a centralized RDF store. The adaptation of the Semantic Web for personal information management and the growing desire for mobility are often accompanied by situations where no network connectivity is available and remote access to data is limited. Such situations could be avoided when mobile devices can operate on offline data replicas and synchronize changes when connectivity is restored. However, due to the still limited storage capacity and computing power of mobile devices, careful selection of the information to be reproduced becomes essential. This selection must be automatic, transparent and adaptive.

3.4.1 Distributed Hash Table (DHT)

Centralized RDF database management systems and triple stores such as RDFStore [75], Jena [54] have been designed to support the storage and retrieval of RDF data. Although these systems are of simple design, they suffer from the traditional limitations of centralized approaches. As an alternative to these centralized systems, peer-to-peer approaches have been proposed to overcome some of these limitations by creating (fully) decentralized data storage and retrieval systems [76–79].

A distributed hash table (DHT) is a technology that allows the setting up of a hash table

in a distributed system. The structured peer-to-peer networks using DHT maintain a structured overlay network on contributors and use message routing instead of flooding [79]. The basic functionality they offer is the search by key, which returns the identity of the node storing the object with this key. The DHT provides a generic interface, which facilitates its adoption by a wide variety of applications as a storage substrate: data storage and retrieval through keys. DHTs store data blocks on hundreds or thousands of machines connected to Internet, replicate data for greater reliability and locate them quickly despite running on high latency extended links. To ensure the availability of objects despite saturation, DHT relies on replication: several copies of each object are stored on different nodes. A replication protocol is responsible for ensuring that at all times, each object is replicated on a sufficiently large number of replicas (called a set of replicas). The replication protocol is also responsible for deciding where replicas should be located. Several replication strategies have been proposed [80].

- `Neighbor Replication`: In all DHTs, nodes store and maintain a list of remote nodes in their routing table according to the DHT selection algorithm. In Chord [81], for example, since the DHT naturally stores and maintains a list of successors, the neighbourhood refers in this case to successor nodes.
- `Path Replication`: Path Replication replicates a data object along the search path through which the request passes, from the source to the root node. It allows objects to be replicated on nodes that are on the same path and, as a result, shortens paths and speeds up searches. Tapestry [82] uses this technique.
- `Multi Publication Key Replication`: To improve data availability, a key is associated with n points in the coordinate space and, as a result, replicated to n separate nodes of the system. A key pair is then unavailable only when all n replicas and corresponding n routes are simultaneously unavailable.

3.4.2 Reference approaches

Freenet [83] is a file sharing application (text, sounds, data, etc.) that also protects the anonymity of authors and readers. Freenet nodes contain 3 types of information: *keys* (which are similar to Web URLs), *addresses* of other Freenet nodes that are also likely to know similar keys and possibly *data* corresponding to these keys. A node that receives a request for a key for which it does not know the exact location transmits the request to a known Freenet node whose keys are closer to the requested key. If a node fails to locate the desired content, it sends a failure message back to its predecessor node, which will then try the alternative successor node that is the next best choice.

PAST [76] is a large-scale peer-to-peer persistent storage utility using Pastry [84]. In PAST, replicas of a file are stored on the nodes closest to the file ID. The insertion message is routed using the file ID as the destination. When the message reaches the first of the nodes closest to the file ID, the node accepts responsibility for a replica of the file and forwards the insertion request to the other n-1 nodes with the ID of the node closest to the file ID.

The Cooperative File System (CFS) [77] is a peer-to-peer system that provides a guarantee of the efficiency, robustness and load balance of file storage and retrieval (such as mp3). CFS provides distributed storage of read-only files. It is structured as a collection of servers that provide block-level storage (block-level). The core of the CFS software consists of two layers, DHash [77] and Chord [81]. The DHash layer (distributed hash) performs block extractions for the client, distributes blocks between servers and maintains cached and replicated copies. DHash uses the distributed search system Chord to locate the servers responsible for a block. Chord implements a hash operation that maps block identifiers to servers.

Nejdl et al. [78] propose an RDF-based P2P architecture called Edutella that uses super peers. In this type of topology, a set of nodes is selected to form the super-peer network, building the backbone of the P2P network, while the other peers connect to the super-peers in a star-shaped topology. Super-peers form a topology called HyperCuP and are responsible for processing requests. In this organization, each super-peer can be

considered as the root of an extension tree that is used for query routing, distribution and index updating. Each peer sends indices of his data to his super-peer.

RDFPeers [79] is one of the first RDF store solutions with a structured peer-to-peer system. The main idea is to use a MAAN (Multi-Attribute Addressable Network) overlay [85] to index a triple three times: once based on the subject, once on the predicate and once on the object. When an RDF triplet is inserted into the network, it will be stored three times, applying a hash function to its values representing the subject, predicate and object. Requests can then be efficiently routed to the network nodes where the triples in question are known to be stored if they exist.

GridVine [86] is a distributed data management infrastructure based on P-Grid DHT [87] on the overlay layer and which also maintains a semantic mediation layer. GridVine allows distributed searching, persistent storage and semantic integration of RDF data. GridVine indexes the triples three times on the P-Grid layer according to their subjects, objects and predicates. In order to integrate all heterogeneous data that are semantically linked but syntactically shared by peers at the P-Grid level, GridVine supports the definition of semantic alignments in peers.

The BRICKS project [88] offers high data availability in DHT by replication. For data replication, BRICKS stores the data in the DHT using several keys, which are correlated to the data key. To be able to retrieve an updated replica, BRICKS uses versioning. Each replica has a version number that is incremented after each update. However, due to simultaneous updates, two different replicas may have the same version number, making it impossible to detect the last replica.

PAGE (Put and Get Everywhere) [89] is a distributed RDF repository based on Bamboo DHT [90]. The RDF data model extends the standard RDF data model by introducing the notion of context. In PAGE, each RDF triple, designated by $t = (s, p, o)$, is associated with a context, labelled c. The RDF triple combined with the context forms a quad. Each quad is indexed six times and identified by an identifier built by concatenating the hash values of its elements (s, p, o, c).

3.5 Distributed graph and semantic overlay

The fundamental problem that makes it difficult to search in existing peer-to-peer systems is that, in terms of semantics, documents are randomly distributed. Considering a request, either the system must search on a large number of nodes or the user runs the risk of missing relevant documents. To solve this problem, the notion of semantic superposition is introduced: a logical network where contents are organized around their semantics, so that the distance between two documents in the network is proportional to their semantic dissimilarity. Peer-to-peer semantic systems (SP2P) combine two complementary technologies: peer-to-peer networking and ontologies. In addition to unstructured and structured peer-to-peer networks based on DHT, the semantic peer-to-peer network retains the semantic characters of node roles in communities. Nodes are identified as domain names classified by the semantic meaning of roles in organizations. Nodes build the semantic peer-to-peer network according to their domains classified as a hierarchical tree. In recent years, several peer-to-peer data management infrastructures allowing semantic data sharing have emerged. These systems are based on a decentralized integration approach: each peer represents an autonomous information system and semantic data integration is achieved by establishing direct alignments between the different peers. Although these systems share schema integration by adopting different formalisms, they all boil down to the underlying infrastructure and consider the system as a graph of interconnected data sources (Figure 3.2: One node is connected to another by a logical link. Nodes with the same kind of music form a semantic superposition. We have the following overlays: Rock, with nodes A, B and C; Rap, with B, E, and F; Jazz, with E, F and G; Techno, with F, D and H. Source). The use of Semantic Web techniques in peer-to-peer systems dates back to the SWAP project [91], coordinated by the University of Karlsruhe.

3.5.1 Semantic overlay networks

Crespo et al. [92] proposed the concept of Semantic Overlay Networks (SON) in which peers are grouped by the semantic relationships of the documents they store. Semantic

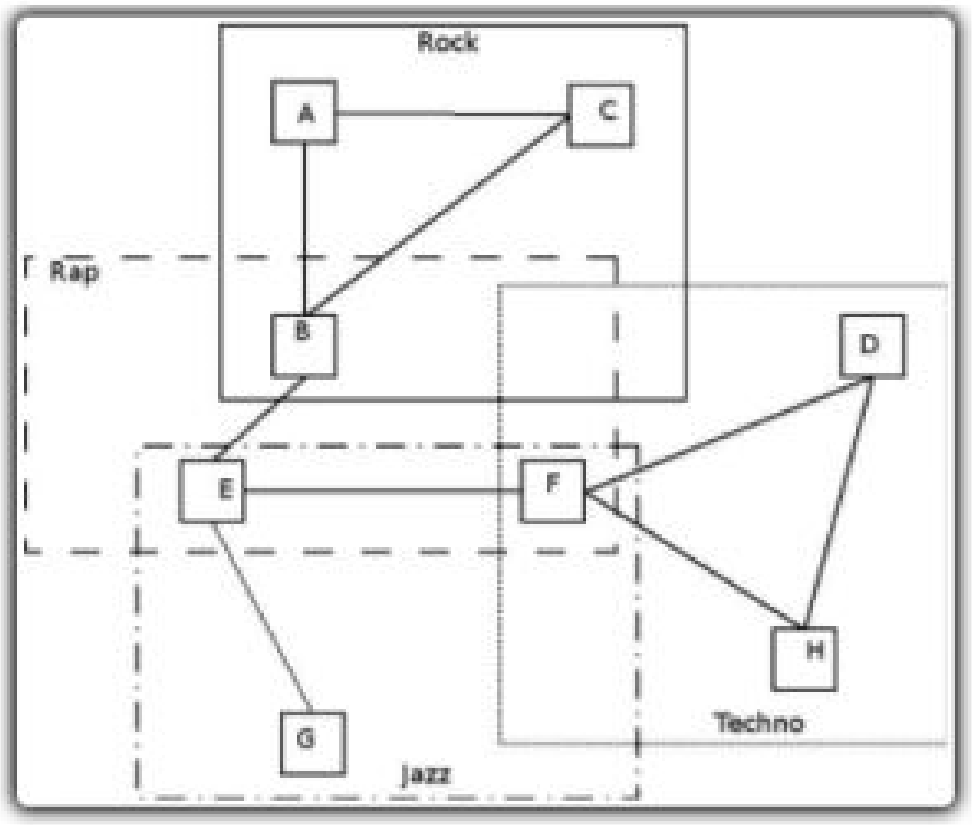

Figure 3.2: Example of P2P semantic overlay with 8 nodes classified by music genre [4].

superposition networks have been proposed as a means of organizing content in peer-to-peer networks. Each peer stores additional information on the classification of content and route requests to the appropriate SONs, which increases the chances that the corresponding objects will be found quickly and reduces the search workload. There are many challenges when creating SON, which include how nodes are assigned to SONs and to which SONs a request should be sent. Peers should be evenly distributed among the SONs, in order to be able to respond quickly to requests, as fewer peers as possible should be interviewed, and each peer should belong to a small number of SONs, so that each peer only has to manage a few connections. However, in practice, the distribution of peers across semantic classes may be unreliable and dynamic. Many peers will belong to classes with very popular subjects that are constantly changing, while some uncommon classes will be less populated [93]. In addition, in most early approaches, nodes decide which SONs to

join based on the classification of their documents. This leads to a fixed configuration of semantic overlay networks, so that performance highly depends on a good choice of classification algorithm and requires an identical classifier for all peers. The classification of documents and queries can be done automatically, manually or by a hybrid process. Several examples of automatic classifiers exist in the literature: text matching [94], Bayesian networks [95], clustering algorithms [96], etc. Manual classification can be achieved by requiring users to mark each query with the style or subtype of the expected results. If the user does not know the subtype or style of the potential results, he can always select the root of the hierarchy to have all the nodes queried. Finally, hybrid classifiers aid the manual classification with databases.

3.5.2 Semantics in peer-to-peer systems

Piazza [97] includes an infrastructure and alignment language for semantic alignment and data management in a peer-to-peer environment. The system takes into account the structure of the domain and the document. In Piazza, peer requests can be reformulated and sent to semantically related peers. Transitive closure of peer-to-peer translations is used to respond to requests.

Loser et al. [98] introduce the concept of semantic superposition clusters in super-peer networks. SOCs (Semantic Overlay Clusters) are designed for very large, highly distributed networks that improve semantic search and interoperability. In particular, the super-pair topology, consisting of a super-pair backbone with powerful computers and smaller clients that are linked to these super-pairs, is very appropriate for this approach.

In **[99], Voulgaris et al.** present a semantic overlay linking semantically related nodes. This semantic overlay provides the main search mechanism, while the initial peer-to-peer system provides the failover search mechanism. They are studying implicit techniques to group nodes with similar interests together to improve content search in file sharing systems. Each node, based on its query/response history, creates a list of semantic neighbors to which its queries are sent before using the standard search algorithm as a failover.

P2PDating's [93] idea is to create semantic overlay networks in a P2P environment,

where a peer decides independently which SONs he wants to join. The approach works by forcing peers to come into contact with other peers they are not yet aware of. Authors consider high quality peer caching to be the natural way to create semantic overlay networks. It can be considered as the criterion for creating network links between peers, or be used in existing P2P systems.

Himali et al. [100] propose a semantic clustering and routing scheme that aims to improve the quality and efficiency of research in P2P systems. The basis of semantic clusters are concepts of a shared semantic ontology already agreed. This semantic ontology is used to improve information retrieval and facilitate interoperability. Peers discover semantic neighbours by taking into account the structural relationships of concepts in ontology. Himali et al. also propose a new query routing mechanism that exploits the conceptual hierarchy to quickly route a query to the target cluster.

3.6 Distributed graph: cloud and Fog computing

This section discusses on the evolving paradigm of Cloud Computing, which aims to provide reliable, customized and QoS (Quality of Service) guaranteed dynamic computing environments for end-users. Although this paradigm is mainly driven by the high availability of Internet access, it presents a relevant approach which consists in providing power (storage, calculation, processing, etc.) for end-users via Internet instead of obtaining this power by acquiring hardware and software. We are particularly interested in this approach as it could be envisaged in a local environment with limited Internet access. Mobile contributors could form a local common storage and possibly benefit from power of local super-contributors (i.e more powerful contributors in terms of storage, calculation, processing, etc.) via a peer-to-peer connection.

The size, diversity and increased complexity of RDF reasoning makes it difficult to maintain huge RDF data volumes. Distributed data store architectures are required to overcome volume related challenges. Cloud computing has become a widely adopted paradigm for scalability, fault tolerance and elasticity features offered to many applications, facil-

itating distributed and parallel architectures deployment. Semantic Web community researchers focus on solving the scalability and performance problems of traditional Semantic Web tools by leveraging cloud computing technologies. The advent of Cloud Computing has paved the way for a distributed ecosystem of RDF triple stores that has the potential to provide very large scale storage and distributed query processing capabilities. The MapReduce paradigm, which is built on Google's file system [101], is the dominant parallel and distributed programming paradigm in the cloud computing community because of its high performance and fault tolerance capability [102]. MapReduce is a programming model for parallel processing of huge data volumes. It is an evolutionary technology welcomed by the scientists. This is used by Google for Web indexing, data storage, social networks. Apache also implements MapReduce in the open-source framework Hadoop [103], which is successfully applied to solve data intensive problems in different domains. This is a distributed file system in which files are stored using replication. Hadoop offers a high degree of reliability and fault-tolerance.

3.6.1 Cloud for distributed RDF data

In recent years, cloud computing provided many opportunities for companies by offering their customers a range of IT services. The current cloud computing *pay-as-you-go* model is becoming an effective alternative to owning and managing private data centers for customers facing Web and batch applications processing [104]. Cloud computing frees organizations and their end users from having to plan for many details, such as storage resources, computational limitations and the cost of network communication.

Cloud computing is becoming the general Internet approach to information storage, retrieval and management. At the same time, mobile devices are emerging as the main service applications. Successful integration of cloud computing and mobile devices is therefore the key task of the next generation network. However, this integration faces several fundamental challenges: service agility, real-time response, long-term connection [105]. To address these challenges between cloud and mobile applications, Fog computing has re-

cently emerged as a more practical solution to enable seamless convergence between cloud and mobile for content delivery and real-time data processing [106]. Fog computing can address these issues by providing resources and services that are accessible to end users at the edge of the network, while cloud computing is more about providing distributed resources over the main network.

3.6.2 Fog/edge for distributed RDF data

Fog computing is a distributed computing paradigm acting between cloud data centers and devices/sensors (users) as a middle tier [107]. The Fog computing concept was introduced by Cisco in 2012 to address the challenges of IoT (Internet of Things) applications in the conventional cloud computing [108]. A Fog computing system consists of conventional network devices like routers, switches, decoders, proxy servers, base stations, etc. (Figure 3.3) placed near peripherals/sensors [107]. These components have various computation, storage, networking, and other features, and can support the execution of service applications. As a result, functional components allow Fog based services to build widespread geographic cloud-based service distributions. In addition, Fog computing eases positioning support, enhanced mobility, live interaction, interoperability and evolutivity. Thus, it can operate efficiently in terms of service latency, energy consumption, network traffic, capital and operating expenses, content distribution, etc. In this sense, Fog computing better meets the requirements of IoT applications as opposed to the single use of cloud computing [109].

Fog computing is often assimilated to Edge computing particularly due to the fact that Edge takes up the idea of Fog computing, i.e. bringing computing resources closer to end users [110]. However, there is a particular difference that is based on the location of computing resources. Fog computing is based on small data centers spread over different sites located on the periphery of the network [111]. These data centers typically have several servers and provide computing and storage resources to customers located at the edge of the network. Edge computing, on the other hand, allows data processing on the

peripheral network, which consists of end devices (mobile phones, smart objects, etc.), peripherals (edge routers, decoders, bridges, base stations, wireless access points), etc. [110].

There are similar concepts such as Mobile Cloud Computing (MCC) and Mobile-Edge Computing (MEC) that identify themselves in Fog concept [107]. MCC describes an infrastructure where the storage and processing of data take place outside of mobile equipment's [107]. Mobile cloud-based services transfer processing and data storage power from smartphones to the cloud, providing mobile services and applications not only to the users themselves, but to a wider set of mobile subscribers as well [112]. MEC may be considered like a cloud service operating on a dynamic network and accomplishing particular functions not achievable using a conventional network architecture [113]. In the context of Internet of Things, Fog computing appears to be a mix of MCC and MEC, while it stands out as an increasingly promising and widespread computing paradigm.

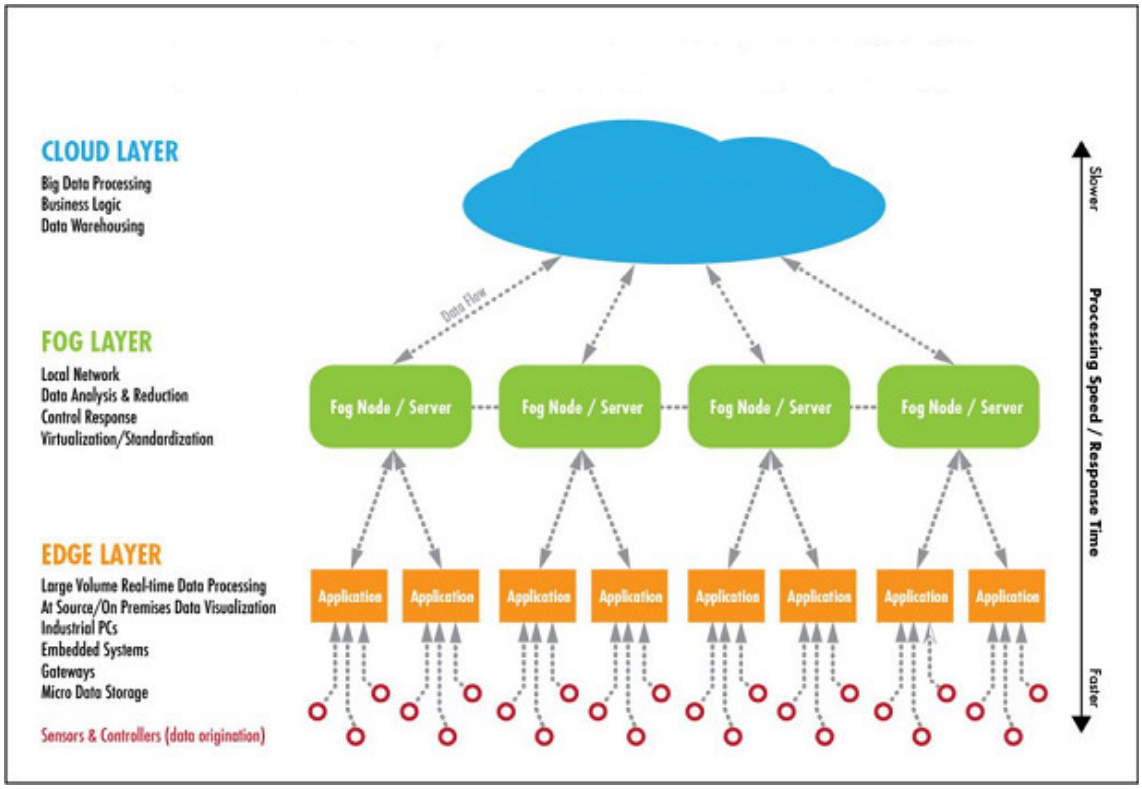

Figure 3.3: Example of Fog/Edge architecture [5].

3.6.3 Selected Fog/Edge approaches

Cloudlet [114] is considered as an exemplary implementation of resource-rich Fog nodes. Three layers made up its architecture. The lower layer consists of both Linux kernel and cloud data cache, the middle layer is virtualization with a set of cloud software such as OpenStack [115], and the upper layer consists of applications isolated by different virtual machine (VM) instances. Cloudlets have been specifically designed to provide services to mobile users with limited local resources that can act as lightweight clients and access cloud resources that are one-step away via a wireless network.

Zhu et al. [116] apply existing methods for Web optimization in an innovative way. In the context of Fog computing, these methods can be combined with unique knowledge that is only available on Fog devices. A more dynamic adaptation to the user's conditions can also be achieved with specific knowledge on the network periphery. Consequently, the rendering performance of a user's Web page is improved beyond that obtained by simply applying these methods on the Web server.

Hong et al. [117] presents a PaaS (Platform as a Service) programming model, called Mobile Fog, that provides simplified programming abstraction and supports dynamically scaled applications at runtime. In this model, an application consists of distributed mobile Fog processes that are aligned with computing instances distributed over the Fog and cloud, as well as various equipment at the network periphery. At runtime each process performs application-specific tasks such as detection and aggregation in relation to its location and level in the network hierarchy. Each Mobile Fog process manages the workload of a certain geo-spatial region.

DiploCloud [118] is an efficient and scalable distributed RDF data management system for the cloud.Three major structures constitute the system: clusters of RDF molecules considered as hybrid structures using both property tables and RDF sub-graphs, model lists (storage of literals in compact lists as in a column-oriented database system) and an efficient key index that indexes URIs and literals based on the clusters to which they belong. The system design follows the architecture of many modern distributed cloud-based systems (for example, Google's BigTable [119]), where a main node is responsible for inter-

acting with customers and orchestrating operations performed by other nodes (Worker).

Faruque et al. [120] introduces Fog computing as an innovative platform for energy management. The proposed platform uses interoperability, scalability, adaptability and connectivity between intelligent platforms on the Fog computing platform, which is a low-power, low-cost device for computing, storage and communication. Energy management or control software is implemented as a service based on Devices Profile for Web Services (DPWS) also used for discovery to provide plug-n-play functionality. This service-oriented architecture also summarizes the heterogeneity of communications and hardware. With a huge number of vehicles in urban areas, bringing underused vehicle resources into service offers an excellent opportunity and value. Hou et al. [121] thus conceive the idea of using vehicles as communication and computing infrastructures, called *Vehicle Fog Computing (VFC)*, which is an architecture that uses a multitude of collaborative customers/end users or onboard devices close to the user to perform communications and calculations, based on a better use of each vehicle's individual communication and computing resources.

Rahman et al. [122] presents a Fog-based distributed semantic model named Semantic-Fog allowing semantic services in the vicinity of IoT devices. The semantic Fog structure comprises IoT features in the perception level, Fog units in the treatment level, and cloud infrastructure for the treatment and application level. The Fog nodes are organized according to their functionalities. Layer 2 Fog unit gathers raw sensory data and execute certain functions like filtering and clustering, and pass them on to the higher level Fog unit. Layer 1 nodes obtain this aggregated data of high quality and carry out compiling, modelling and mapping processes, and initiate suitable measures. Then, data are transmitted to remote cloud for storing, viewing or complex treatment.

Mehmood et al. [123] proposes a cloud-centric IoT platform for virtual object registration and initialization. For security reasons, permission and control are required for registration procedure. Only authorized persons will be approved by the authorization mechanism to record IoT equipment and prevent unneeded IoT platform exploitation. This differs from traditional IoT platforms as they offer material and application services within a single platform and allow users to connect and use them. RDF is used for finding, exchanging and presenting data on IoT marketplace.

In **[124], Farnbauer-Schmid et al.** introduces the Semantic Edge Computing Runtime (SECR), an edge computing tool developed to provide a background for IoT peripherals. SECR combines two concepts: data integration to support the diversity of IoT applications, and edge technology to minimize data volume to be transmitted for distant processing. By using data integration, edge processing can be carried out at a greater degree of abstraction. All outputs from SECR services are available as RDF graphs allowing edge level interoperability.

3.7 Synthesis on collaborative graph sharing and modification approaches

The four concepts previously presented (Graph Replication, Distributed Graph and DHT, Distributed Graph and Semantic Overlay, Cloud and Fog Computing) constitute a set of approaches that are relevant for decentralized architectures. Their use in the design of mobile solutions for sharing and contributing Data to the Web could solve the problem of high availability of relevant data in contexts where access to remote sources is constrained by limited connectivity to Internet. The approach of intelligent replication of the graph on the architecture is particularly interesting in this context. The challenge then concerns the preservation of data consistency as update operations are produced at the contributing nodes. Optimistic asynchronous replication effectively addresses this challenge with the assumption that there are few conflicting updates, which, if they occur, are processed at the end of the process. This means that the system tolerates a certain level of divergence between replicas. This compromise may be acceptable for collaborative sharing and contribution solutions. Table 3.1 summarizes different solutions presented. These models are compared according to the following criteria:

- `Data type`: Exchanged data types.
- `DHT`: The architecture relies on DHT.
- `Partial replica`: The system adopts a partial replication mechanism on the graph.

- `Semantic Overlay`: The system builds a semantic overlay on the nodes.
- `Total replica`: The system adopts a total replication mechanism on the graph.
- `Caching`: The system has a caching service.
- `Fog layer`: The architecture has a Fog layer.

Table 3.1: Summary table of solutions dealing with mobile access to data:
The support of a characteristic is indicated by the sign √

	Semantic-Fog [122]	SECR [124]	Swooki [69]	Cloudlet [114]	Logoot-Undo [70]	Crate [40]	DiploCloud [118]
Data Type	RDF	RDF	RDF	file	file	file	RDF
DHT					√		
Partial Replica	√	√			√		√
Total Replica			√		√	√	
Caching							
Fog Layer	√	√		√			√

The literature shows that the approaches identified above have been successfully applied on architectures based on gossip protocols. The decentralized peer-to-peer architecture is built using the underlying gossip protocol that also ensures connectivity despite node arrivals and departures. The protocol also manages, in some cases, the exchange of update operations between nodes. The data structure is then hosted on the architecture according to one of the previous approaches. We can cite a few examples.

In Crate [40], a decentralized real-time collaborative editor that runs directly on Web browsers using WebRTC, the Spray gossip protocol [29] builds and maintains a mesh of contributing Web browsers. Spray provides each contributor with a local neighborhood table (the view) that allows communication in a subset of editors. Crate adopts the graph replication approach. The graph represents a document shared by all publishers (contributors). Each publisher reproduces the document locally and performs its operations directly. To also ensure the consistency of the graph on each editor, Crate uses the Spray protocol to distribute all update operations to all collaborators in an evolutionary way.
Swooki [69] is a peer-to-peer semantic wiki that combines the wiki approach of ontologies such as Semantic MediaWiki [125] and a peer-to-peer wiki based on total replication and CCI (convergence, causality preservation and intention preservation) model such as Wooki [126]. On the one hand, Swooki is based on the graph replication approach. The graph here represents a semantic wiki page (combination of text and RDF data). On the other hand, update operations will be routed to collaborating nodes using the gossip protocol [127] combined with an anti-entropy protocol [128].
Voulgaris et al. [99] use the semantic superposition approach to exploit the semantic structure present in document sharing systems to improve search performances. They propose an architecture with contributor nodes that are semantically close. Each node maintains a list of semantic neighbors to which requests are submitted first, before using a default search mechanism if no semantic neighbors can respond to the request. To build and maintain the semantic neighborhood, the authors assume the existence of a peer-to-peer system supporting semantic searches. This system is then built with the SCAMP gossip protocol [24] that generates an unstructured overlay network.

3.8 Conclusion: data sharing systems with RDF model

In this chapter, we have focused on approaches dedicated to data sharing systems construction according to the RDF data model. The structured analysis of these approaches is based on the layout of the architecture graph, i.e. whether it is replicated (partially or totally), or distributed, or shared and modified in a collaborative way through the involved peers in the architecture.

Existing solutions are generally boosted by local caching, source indexing and synchronization mechanisms. This last element raises the issue of data consistency during system execution, in particular the divergence of states between remote and local sources when all synchronization operations are performed. To overcome this constraint, in some cases, such as [69], the CCI (Convergence, Causality Preservation and Intention preservation) model is used to ensure system coherence. This model allows the system to maintain the following three properties [67]: Convergence: when the same set of operations is executed on all sites, they will all have the same state; Causality: if an operation O is executed before another operation O', the same execution order is respected on all sites ; Intent: for any operation O, the effects of the execution of O on all sites are the same as the intentions of O, and the effect of the execution of O does not change the effects of the independent operations. Other solutions [83, 93, 99] also rely on mechanisms for caching relevant data according to the meta-data of peer requests. Replacement (or deletion) policies identify which data to move to persistent storage or permanently delete for proper cache management.

Using metrics such as complexity in time and space and the traffic effect on network architecture, existing solutions are compared by analyzing their performance in relation to their own properties. These include data quality metrics (exhaustiveness, conciseness, consistency), cycle number, query load, scalability. Here, we also noted some interesting points for future work. They can be summarised mainly in two directions. The first concerns the problem of synchronization between local and remote sources. It targets the design of relevant mechanisms to improve the consistency and reliability of the exchanged

data. The second direction aims to ease the constraints associated with the implementation of CRDT (Commutative Replicated Data Type) by eliminating the requirement for the underlying network to ensure causal ownership. One additional relevant approach that we can consider in future work is cooperative cache systems. These systems are also relevant to ensure local access, especially in cases where Web browsers represent the contributors. The next chapter present a set of mechanisms that can lead us to mobile and restricted access to Linked and shared peer-to-peer data.

Chapter 4

Towards an Architecture and Model for Limited and Local Mobile Access to Web of Data

In this chapter we propose a solution to the problem of RDF data exchange by mobile contributors in areas where Internet access is intermittent. We introduce a logical architecture consisting of three levels. It allows contributors to co-construct locally and automatically a warehouse of RDF triples. Two main axes will be considered for the implementation of this architecture: (1) the construction and maintenance of connectivity ensured by the gossip protocol, and (2) the high availability of data ensured by a replication mechanism. To evaluate our architecture, we will consider different aspects such as robustness, efficiency, reliability.

The chapter is organized as follows: section 1 presents a set of mechanisms that can lead us to mobile and restricted access to Linked and shared peer-to-peer data. The architecture, the chosen configuration and the functioning principle are presented in section 2. Section 3 provides the architecture theoretical conception details. The evaluation plan is described in section 4. Section 4 concludes the chapter.

4.1 Introduction: Linked Data sharing architecture on a gossip protocol

We have seen in the state of the art several approaches that deal with solutions that we can consider to build our system. Although these solutions are in some cases specific to scenarios and environments different from ours, we can draw inspiration from them to propose relevant and effective alternatives.

In particular, for our scenario, several of these solutions based on peer-to-peer architectures provide important points to circumvent the permanent need for Internet connection. In the following, we propose a set of mechanisms that can lead us to mobile and restricted access to Linked and shared peer-to-peer data. We are primarily interested in the peer-to-peer architecture that will support our solution. An hybrid peer-to-peer architecture is a good option in the sense that the scalability of the system is an important aspect in our case. Indeed, we consider in our scenario, the existence of particular nodes with more important characteristics compared to the others: storage memory, processors, RAM, Internet access. These nodes, placed for example on public buildings such as the town hall, the university or the stadium, may be considered as super-peers in our architecture. Thus, each new peer wishing to join the architecture, will be integrated in one of the subsets connected to one of the super-peers. As super-peers can be subject to disconnection or crash, we assume that they can join and leave the architecture at any time. However, this should not lead to a total shutdown of the read and write access processes at the local level, i.e. at the level of the nodes connected to a failed super-peer (momentarily disconnected from the architecture). For a node wishing to join the architecture, we will assume that the integration process will take place according to its geographical location. Specifically, we will consider that the new node will be integrated with the set of peers connected to the nearest super-peer in relation to its geographical location. This requires, notably, that each node integrates its geographical coordinates into its profile.

We will build our architecture on a gossip protocol with a two overlays adhesion mechanism. As we have seen in the second chapter of this document, these protocols have

the particularity of being self-organised, resistant to failures and especially adapted to dynamic environments where nodes join and leave the architecture without any conditions. We are particularly interested in a hierarchical protocol without a contact server. Although the super-peers in the architecture can also act as contact server, their dynamic nature can lead to their unavailability and thus to a blockage of the integration process of a new node. Therefore, we will consider that any request from a node wishing to integrate the architecture can be handled by any other node.

The hierarchical nature of the underlying gossip protocol will allow us to apply a clustering mechanism on the nodes creating virtual links between peers according to proximity criteria that we will define. Through this mechanism, we want to achieve an architecture in which each super-peer will be connected to peers organised in small clusters representing common interest groups. We will also assume the existence of inter-cluster and intra-cluster links to ensure a locally connected architecture. Once a node has initialized its view with data received from a random contact node, the cyclic execution of the gossip protocol will allow to improve its view, which will progressively be composed of information on nodes with similar points of interest which collectively form one or more clusters.

As the Internet connection is unreliable in our environment, a mechanism for local access to data must also be implemented. We consider precisely that all the super-peer nodes of the architecture perform local and intelligent replication of relevant data from online sources when connected to Internet. They form a distributed local data source that is queried before the online sources when processing queries. Data can be modified locally and it is relevant to implement a real-time update process to maintain consistency between local and remote sources as well as between local sources of different super-peers. When no update process between super-peers is in progress, the different local sources will be assumed to be identical. As a result, no scenario of routing requests to neighbouring super-peers can occur. We will also assume that in each cluster, data related to the considered point of interest is replicated at the super-peer level.

4.2 Architecture and Configuration: theoretical analysis

In this section we present our architecture consisting of three levels.

Level 1: Composed of the different peers (mobile devices) involved in the architecture. Here we will implement the following elements.

- Gossip protocol with two overlay adhesion mechanism;
- Construction of overlay 1 based on the metric distance between peers;
- Construction of overlay 2 based on a semantic distance between the peers' centers of interest;
- Construction of common cooperative caches.

Level 2: This level consists of the super-peers (servers) participating in the architecture. Two types of configurations are possible on this level

- `Architecture 1` : one super-peer on each zone
- `Architecture 2` : a cluster of super-peers in each zone

Levels 1 and 3 are common to architectures 1 and 2. The difference between these architectures is therefore mainly due to the cluster disposition or not of the super-peers on level 2. We detail these two architectures in the following.

Level 3: this level represents the remote data sources. Its main goal here is to process requests not satisfied by level 2.

For this architecture, we consider the following properties on the replica control mechanism.

- `Update location (P1)`: we go for the multi-master approach. This approach is more flexible than the single-master approach, as in the event of a failure of one master, other masters can manage the replicas. This improves data availability.

- `Update propagation moment (P2)`: we opt for the asynchronous approach. Although the synchronous approach ensures good convergence of the replicas, it has a major weakness, namely that an update operation is only applied when all the replicas have received it. This means that if a replica is no longer available, the update process is blocked. In addition, the latency of the requests and the load on the network increases with the number of replicas. In contrast to this approach, the asynchronous model ensures a reduced load on the network. This reduces latency and improves the scalability of the architecture. More precisely we opt for the optimistic asynchronous approach. This model is more flexible than the pessimistic model as it allows the system to decide the optimal period to propagate update operations. This allows the system to support nodes' dynamic character.

- `Replication mode (P3)` : for this property we consider both partial and complete approaches. More precisely, for the purposes of the theoretical evaluation we will consider several models of the architecture differentiated mainly by level 2. On each model we will consider either a partial, total or hybrid replication mode (combination of partial and total modes). This last mode concerns exclusively the models of architecture 2 (described below) which has the particularity of having super-peer clusters on each zone. The two replication modes will therefore be applied above and within the clusters.

4.2.1 Architecture 1 : one super-peer per zone

Here we present the configuration of architecture 1 (Fig 4.1), which is characterised by the existence of a super-peer (SP) in each zone.

In table Tab 4.1 we present the different possible configurations of architecture 1 related to the replication mechanism applied on the super-peers, 1.A: "DHT-superPeer/zone-

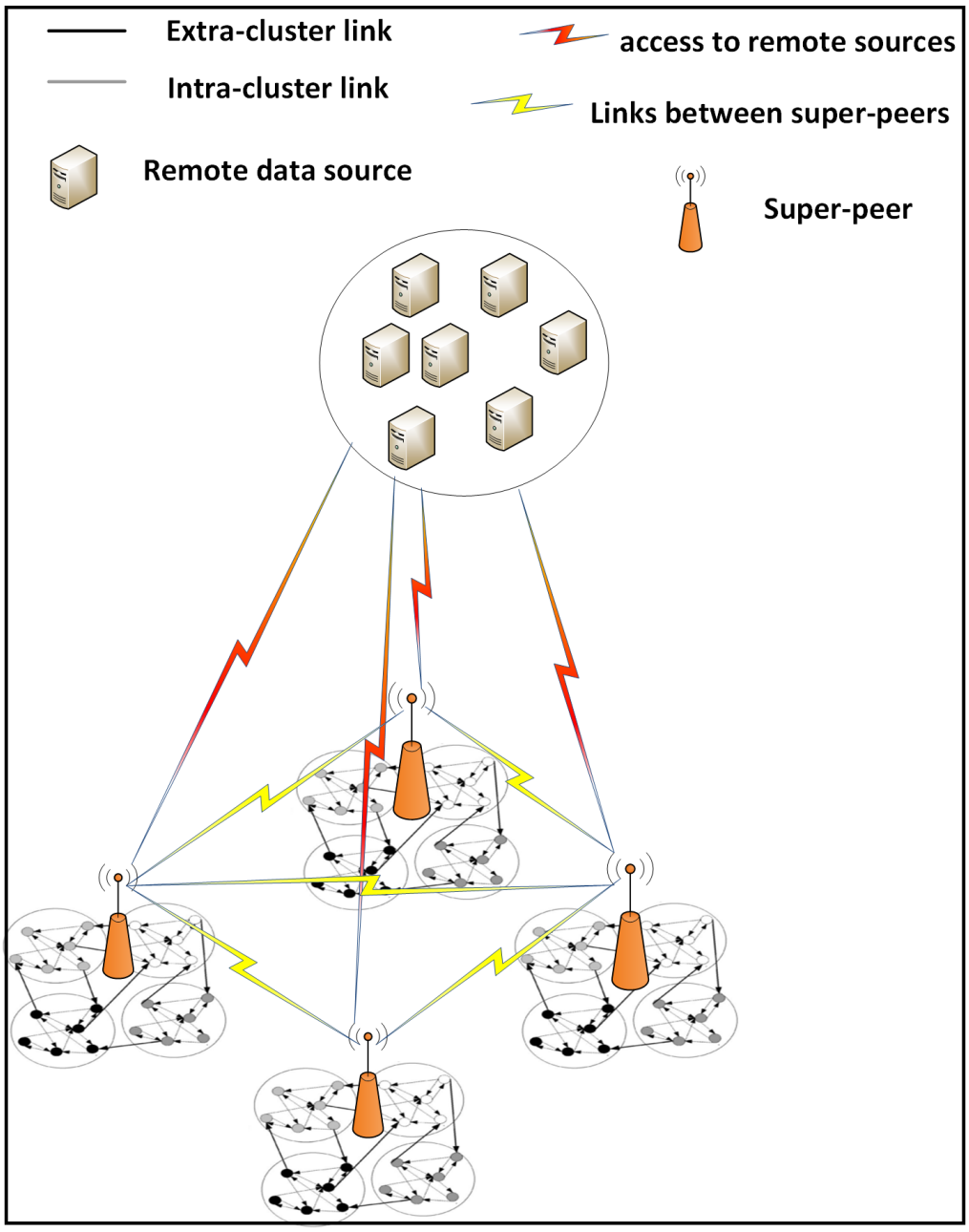

Figure 4.1: Architecture 1, one super-peer per zone.

	Structured Architecture (DHT)		Unstructured Architecture	
	Partial Replication	Full Replication	Partial Replication	Full replication
	1.A	1.B	1.C	1.D
Request Routing	✓			
Request Diffusion			✓	
Updates Routing	✓		✓	
Updates Diffusion		✓		✓

Table 4.1: General summary table of possible models with architecture 1.

partial", 1.B: "DHT-superPeer/zone-total", 1.C: "unstructured-superPeer/zone-partial", 1.D: "unstructured-superPeer/zone-total. We compare them here according to whether the requests and update operations (Update) are routed or broadcasted on the network.

4.2.2 Architecture 2 : a cluster of super-peers per zone

Here we present the configuration of architecture 2 (Fig 4.2), which is mainly characterised by the existence of a cluster of super-peers in each zone. Each cluster ensures the coverage and data availability in its zone.

On table Tab 4.2 we present the different possible configurations of architecture 2 related to the replication mechanisms applied above and inside the clusters: 2.A (DHT-cluster/zone-partial:partial), 2.B (DHT-cluster/zone-partial:total), 2.C (DHT-cluster/zone-total:partial), 2.D (DHT-cluster/zone-total:total), 2.E (unstructured-cluster/zone-partial:partial), 2.F (unstructured-cluster/zone-partial:total), 2.G (unstructured-cluster/zone-total:partial), 2.H (unstructured-cluster/zone-total:total). We compare them here according to whether requests and update operations (MAJ) are routed or broadcast within the cluster or outside

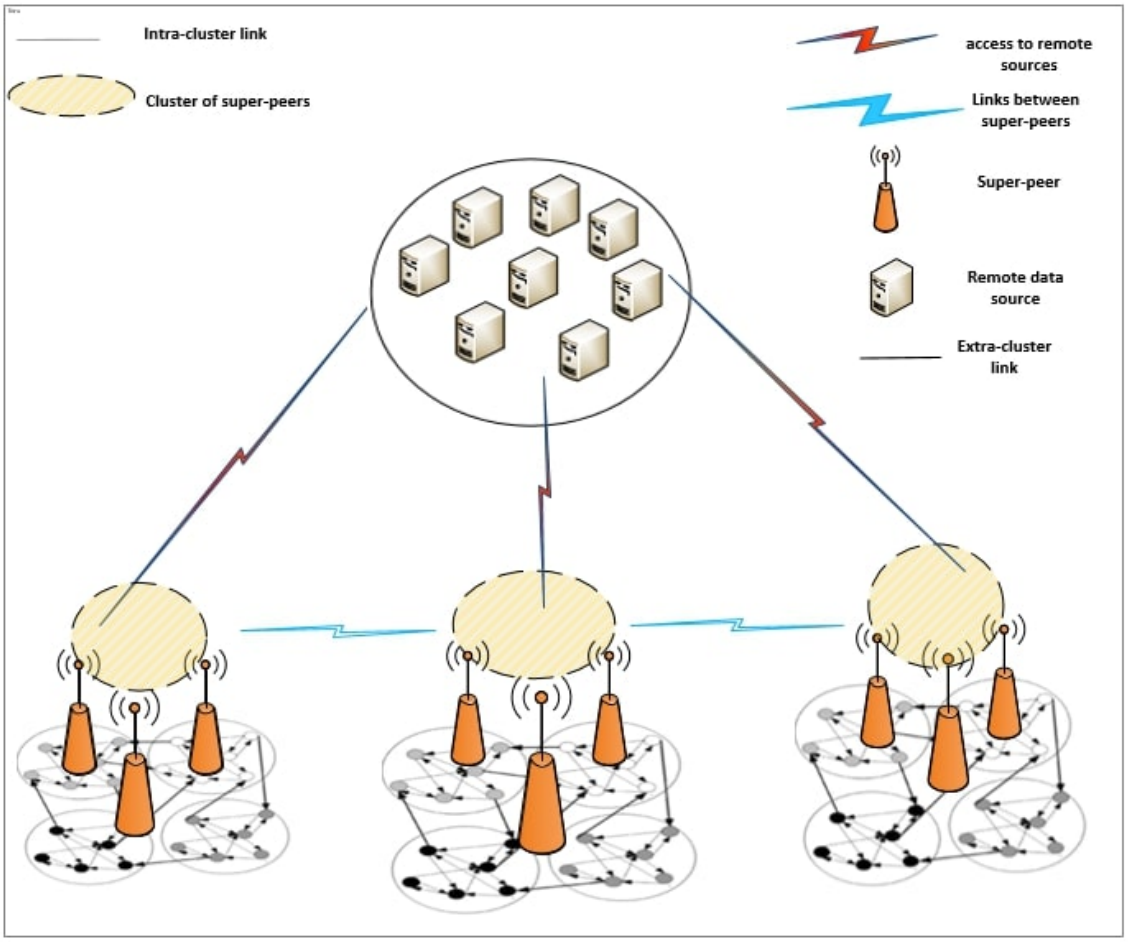

Figure 4.2: Architecture 2, mobile peers (black dot), super-peers (red) and remote sources.

Above the cluster	Structured Architecture (DHT)				Unstructured architecture			
	Partial replication		Full replication		Partial replication		Full replication	
Inside the cluster	Partial replication	Full replication	Partial replication	Full replication	Partial replication	Full replication	Partial replication	Full replication
	2.A	2.B	2.C	2.D	2.E	2.F	2.G	2.H
Routing requests on the cluster	✓		✓					
Routing requests outside the cluster	✓	✓						
Diffusion of requests on the cluster					✓		✓	
Diffusion of requests outside the cluster			✓		✓	✓		
Update routing on the cluster	✓		✓		✓		✓	
Update routing outside the cluster	✓	✓			✓	✓		
Update diffusion on the cluster		✓		✓		✓		✓
Update diffusion outside the cluster			✓	✓			✓	✓

Table 4.2: Architecture 2, Possible configurations.

the cluster (across the whole architecture).

4.2.3 General Synthesis: comparing alternative architectural models

In table 4.3 we make a comparison of the different architectural models that can be considered in relation to our objectives. These models vary according to whether they adopt a structured network or not, depending on the type of replication implemented on level 2 (partial or total replication). A green cell with a + sign indicates a perfect fit of the model with the property concerned, a yellow cell with a +/- sign indicates a medium fit and a red cell with a - sign indicates a weak or poor fit. We have eliminated the models presented with red columns in the previous tables 4.1 and 4.2 (1.B, 1.D, 2.C, 2.D, 2.G and 2.H). As these models are based on total replication over super-peers, we consider that they are not suitable for our architecture because with total replication all update operations are spread over the entire level 2. This naturally acts on the network load on level 2 and can in some cases slow down or even interrupt the architecture's operation.

We compare the considered models regarding the following properties :

- `Scalability` : the model is able to scale without impacting on the quality of the architecture.
- `Resistance to failure`: architecture resistance to super-peers failure. Does the unavailability of a super-peer have an impact on the architecture (especially on level 1)?
- `Availability`: The model ensures high, medium or low data availability for level 1.
- `Latency`: The latency during search scenarios is considered low, medium or high.
- `Network load`: The model has a low, medium or high network load.

	1.A	1.C	2.A	2.B	2.E	2.F
Scalability	+	+	+	+	+	+
Failure Resistance	-	-	+	+	+	+
High Availability	+/-	+/-	+/-	+	+/-	+/-
Latency	+	-	+/-	+	-	+/-
Network Load	+	+/-	+	+	+/-	+/-

Table 4.3: General summary of the different architectures.

We can see from table 4.3 that extensibility is well covered by the different models. Replication does indeed ensure the scalability of the system. The decentralised nature of our architecture also facilitates scalability.

Models 2.x have a good resistance to failures, unlike models 1.x. With the latter, when a super-peer becomes unavailable (crash, disconnection, ...), all connected single nodes are impacted. That is to say that requests not satisfied by level 1 common cooperative cache will no longer be able to be processed. Moreover, in the case of partial replication (1.A and 1.C), the sub-graph hosted by this super-peer also becomes unavailable, even if the replication is multi-master, this can have an impact on latency.

The high availability of data hosted at super-peer level is also an important feature of our architecture. From the table we also expect that full replication will provide the architecture with better data availability, this is the case with model 2.B. In this model the cluster on each zone ensures near availability of single nodes. On the other hand, the other models (1.A, 1.C, 2.A, 2.E and 2.F) offer lower availability because they adopt partial replication on super-peers but also because of the unstructured architecture for 2.E and 2F.

With regard to latency, we consider that models 1.A and 2.B provide better latency. Both models adopt a partial replication over super-peers. 2.B applies in addition a total replication within the cluster. 2.A has average latency compared to the others due to partial replication above and within the clusters. The routing table here allows requests and updates to be routed to the relevant super-peers. Models 1.C, 2.E and 2.F have higher latency. This is mainly due to the fact that these models are based on an unstructured architecture. This means that during search scenarios, queries are distributed over the entire architecture.

We consider that 1.A and 2.A models are very well adapted regarding the network load, we have a structured architecture and a partial replication on the super-peers and in the cluster too (for 2.A). This means that updates are only routed to the relevant nodes. This is also the case for 2.B model, which differs from 1.A and 2.A mainly by the total replication in the cluster. However, as the cluster size is assumed to be small, the network load can be considered low for this model.

4.3 Approach : Architecture theoretical conception details

Based on the state of the art we have identified several interesting approaches for our project. The main objective of the project is to propose a solution (model, architecture, algorithm, etc...) that can allow mobile users to locally co-construct a triple RDF data warehouse in areas where Internet access is very limited. In this perspective, we decide

to consider a three-level architecture that can ensure good data availability, resistance to failures (crash, disconnection, etc...) of peers, reasonable query processing time, acceptable network load.

In a first step we will focus on level 1 of the architecture. This will host mobile contributor nodes with limited capacities (Internet access, storage memory, energy autonomy, RAM, processor, etc...). Each node implements a gossip protocol which mainly allows to ensure data access to all connected nodes. We will use a basic gossip protocol with deterministic peer selection that will allow us to consider the notion of distance between peers when updating local views. We will then build on top of this gossip protocol an overlay based on the points of interest of each peer. This last overlay will also allow us to organise peers into clusters of points of interest with extra-cluster links allowing data and queries routing between clusters. As each peer has a local cache, requests will first be processed at local cache level, then they will be processed at neighbours caches (cooperative caching) belonging to the same clusters before finally being routed to nodes of other clusters. At this stage, if request is still not satisfied, it will be forwarded to the top level of the architecture (level 2).

The next step is to set up level 2 of the architecture. This level is made up of super-peer nodes on different areas covered by the architecture. We consider that super-peers have no constraints of Internet connection or storage memory. Each super-peer also has a wifi network to which all contributors in the area are connected. Super-peers are organised according to a DHT offering a geographical squaring of the area to be covered. In a given area, the super-peers participating in the grid form a cluster.

On this level we will apply a replication mechanism that will allow us to maintain good data availability and the architecture's resistance to super-peer failures. This mechanism will also ensure a reasonable processing time for unsatisfied level 1 requests. We will apply partial replication over clusters and full replication between super-peers within clusters (model 2.B). For updates, we will opt for a multi-master approach which offers better data availability, and optimistic asynchronous approach which also allows for better management of super-peer failures (crashes and disconnections).

The last step will be to evaluate our architecture.

4.4 Evaluation Plan and Validation Metrics

To evaluate our architecture, we consider different aspects such as robustness, efficiency and reliability. We will study these aspects in relation to different metrics: network load and failure resistance for robustness, latency for efficiency, high data availability for reliability.

- `Network load` : we consider the number of messages transiting on the architecture during requesting and updating scenarios. We will also be able to compare the value of this metrics with those from studies on similar architectures.
- `Failure resistance`: using this metric we will evaluate the robustness of our architecture during disconnection or super-peer crash scenarios. These scenarios will be simulated on the architecture. The objective here is to show that when a super-peer failure occurs on level 2, the impact is almost not perceived on level 1.
- `Latency` : response processing time during search operations will be measured on the architecture. We will consider requests not satisfied by level 1 cooperative cache. As these requests are transmitted to level 2, the latency will be directly impacted by the replication mechanism implemented on this level.
- `High availability` : we will evaluate the high availability of data by acting directly on level 1. The objective here is to show that our architecture ensures good availability of locally relevant data despite the mobility and dynamic nature of the contributors. We will therefore simulate scenarios of mobility (the node changes cluster) and disconnection of the contributor nodes and show that the data from these nodes remains available on the architecture after the scenarios.

We will also test the whole architecture considering that level 1 has no Internet connection. We will run the different scenarios considered on our initial use case example, the Saint

Louis International Jazz Festival. The objective will therefore be to show that these different scenarios are feasible on the architecture we have proposed.

4.5 Conclusion: a proposed three-layer logical architecture

Our objective in this chapter was to propose a solution to the problem of RDF data exchange by mobile contributors in areas where Internet access is intermittent. We have therefore proposed a logical architecture consisting of three levels. It allows contributors to co-construct locally and automatically a warehouse of RDF triples. Two main axes were considered for the implementation of this architecture: the construction and maintenance of connectivity ensured by the gossip protocol, and the high availability of data ensured by the replication mechanism implemented on level 2 as well as the cooperative caching system built by the level 1 contributors. To evaluate our architecture, we consider different aspects such as robustness, efficiency, reliability. We will study these aspects in relation to different metrics: network load and failure resistance for robustness, latency for efficiency, high availability of data for reliability. For experimental purposes, we consider the different models used in the theoretical analysis.We also consider SNOB protocol [129] that we adapt to our context by integrating geolocation to build and maintain the second overlay.
The next chapter describe MORAI (Mobile Read Access in Intermittent), a distributed peer-to-peer architecture organized in three levels dedicated to RDF data exchanges by mobile contributors.

Chapter 5

MoRAI: Geographic and Semantic Overlay Network for Linked Data Access with Intermittent Internet Connectivity

In this chapter, we propose MoRAI (Mobile Read Access in Intermittent), an architecture dedicated to RDF data exchange through mobile applications. MoRAI allows users to access and share data in environments where hardware resources are limited and Internet access is intermittent. Intermittent Internet access refers here to cases where the contributors frequently only have a local wifi or data connection with TCP/IP but do not have access to remote Internet resources especially outside the country or continent. In this case contributors continue to have local access but the failure of a national or international link prevents them from accessing external resources such as DBpedia or Wikidata for example.

In this context, MoRAI allows locally connected systems to continue operating. In this P2P architecture, peers are applications running in a Web browser. To connect to the

Linked Data sharing network the user just has to open the Web application to become a new peer.

This chapter is organized as follows: Section 1 introduces MoRAI. Section 2 introduces our motivating scenario. Section 3 positions our contribution in the state of the art. Sections 4 and 5 respectively present the model and algorithms details. We conclude with the implementation overview in Section 6.

5.1 Introduction to MoRAI : Mobile Read Access in Intermittent Networking Conditions

Our work builds on a joint interest in combining semantic Web approaches with P2P Web overlay networks: (1) overlay networks provide more resilient and efficient connectivity for Web agents to exchange their data and, in return, (2) Web formalisms and protocols provide the needed standards to ensure interoperability between heterogeneous peers, other architectures and heterogeneous resources.

MoRAI's architecture is organized in three levels. The first level consists of the single peers, representing the mobile devices participating to the RDF data sharing system. Such data are stored by all participants and are considered as a data warehouse according to an RDF graph structure. We use a gossip protocol to ensure peer connectivity and data exchange. The second level consists of super-peers, with important resources (Internet access, storage memory, computing power), strategically placed and whose main objective is to ensure data availability. The third level represents remote data sources, in particular SPARQL Endpoints. This chapter details the following contributions:

1. MoRAI, a peer-to-peer logical architecture dedicated to environments with limited Internet access and hardware resources and ensuring participant connectivity and data availability.

2. A geographical overlay network based on peer locations to consider geographically

close neighbours and combined with a semantic overlay in order to provide each participant with a relevant and available knowledge base.

5.2 Motivating scenario: ensuring a reliable sharing of information at a local event

We aim at scenarios where users participate to an event at a given location particularly characterized by intermittent Internet access. During this event, participants represented by browsers on their mobiles form a virtual peer-to-peer network. The generated architecture allows participants to exchange information as RDF data triples. Such exchanges are feasible thanks to the use of WebRTC [29, 129] which allows to establish direct links between Web browsers.

As stated in Chapter 2, a real example of such a scenario that motivated our work is the International Jazz Festival of Saint-Louis in Senegal. The targeted architecture must allow participants to exchange data related to the festival via their phones or tablets.

More precisely, each user must have the ability to exchange data with semantically close neighbours i.e. those with whom the user shares requests for the same types of information and therefore sends the same types of queries to the network. For example, in the case of the Jazz Festival, some participants will be interested by data related to music such as concerts and others by data related to cultural activities such as art expositions.

A second objective is also to enable participants to exchange with geographically close neighbours. The idea is that, during data exchange operations, several peers will be geographically near their points of interest. In the case of the Jazz Festival for example, we assume that a number of the participants requesting information about expositions will be near the art museum. Considering the infrastructure, geographical closeness may also ensure a better chance for two peers to setup a network connection.

This specific motivating scenario is used to scope the design of the architecture and pa-

rameterize the simulations for its evaluation.

5.3 MoRAI's specificities: positioning the architecture in the state of art

In P2P networks, regardless of the topology connecting the peers, for unstructured overlay networks, links between peers are created randomly or are based on proximity measurements. Gossip-based networks are a good example [13–15]. These protocols aim to build and maintain an unstructured topology with random graph properties. They balance the load between peers, are fail-safe, and are used as building blocks in many network management applications, especially when highly dynamic environments are expected [15].

In order to increase availability and reliability, data replication techniques became "commonplace" in P2P as they ensure the high availability of data and services in distributed systems [56]. The resource is stored on all (or part of) the peers of the network that can modify and/or delete portions of it autonomously.

CyCLaDEs [34] aims to build a decentralized behavioral cache for processing LDF requests based on the similarities of LDF (Linked Data Fragment) client profiles. When a request is processed by a given client, each sub-request pattern triple is searched first on the local cache, then in the cache of its neighbors and finally on the LDF server if necessary. CyCLaDEs relies on a random peer sampling architecture for member composition management and a clustered architecture to handle the k-best neighbors.

Behave [33] is a decentralized caching architecture based on behavioural positions. Behave nodes adopt a gossip protocol based on the similarities between their navigation histories to form an interest-based topology. It's behavioral cache emerges from this topology as the federation of the local caches of a node's neighbors. However, Behave has not proved its efficiency in the context of the semantic Web, precisely in SPARQL query processing.

FireChat [130, 131] by Open Garden is a messaging software enabling users to commu-

nicate offline regardless of Internet connectivity or phone service. FireChat relies upon open-access mesh network and is able to use phone-to-phone bluetooth signals to connect whenever mobile phone coverage is unavailable. However, FireChat does not give users the ability to query their neighbors, or to access data preceding its arrival in the community.

Solid [132] is a decentralized platform for social Web applications. The user's data is stored in a Web-accessible personal online datastore (pod). To share information with others (individuals or organisations), user gives permission to access the appropriate information in his pod. The data in the pod remain owned by the user owner. Querying several users' pods however remains an open question on Solid.

The entity registry system (ERS) [133] aims to replace the Web as a platform for publishing Linked Data when the latter is not available (i.e. in cases where internet connectivity is not guaranteed). It allows for Linked Data without using the centralised components that make up the Web infrastructure. ERS is compatible with the RDF data model. But the availability of data in case of a contributor crash is not guaranteed. When a contributor crashes, its data is no longer available to its neighbors unless it was replicated before the crash.

Snob [129] is a query execution model for SPARQL query over RDF data hosted in a network of browsers. Direct neighbours in the network are the data sources and results received from neighbours are stored locally as intermediate results. To speed up query execution, browsers processing similar queries are connected through a semantic overlay network.

In this chapter we combine and extend on the one hand the Cyclades [34] approach regarding the hierarchy of requests processing : first in the local datastore, then in the neighbors' datastore and finally in the super-peers' datastores and on the other hand, the MoRAI request execution model is close to the Snob model (Fig. 5.1). MoRAI allows users to query several data sources at once. The difference comes from taking into account the location of the neighbours to increase the search perimeter and super-peer integration on which the data graph is replicated. This allows us to improve data availability, thus providing

Figure 5.1: MoRAI conception

the system with a better completion rate.

5.4 MoRAI Approach: a 3-level semantic and geographic overlay network

Our logical architecture is adapted to constraining environments and allows mobile contributors to share locally, via a browser network, an RDF dataset. It consists of 3 levels (Fig. 5.2): single peers, super peers and remote sources.

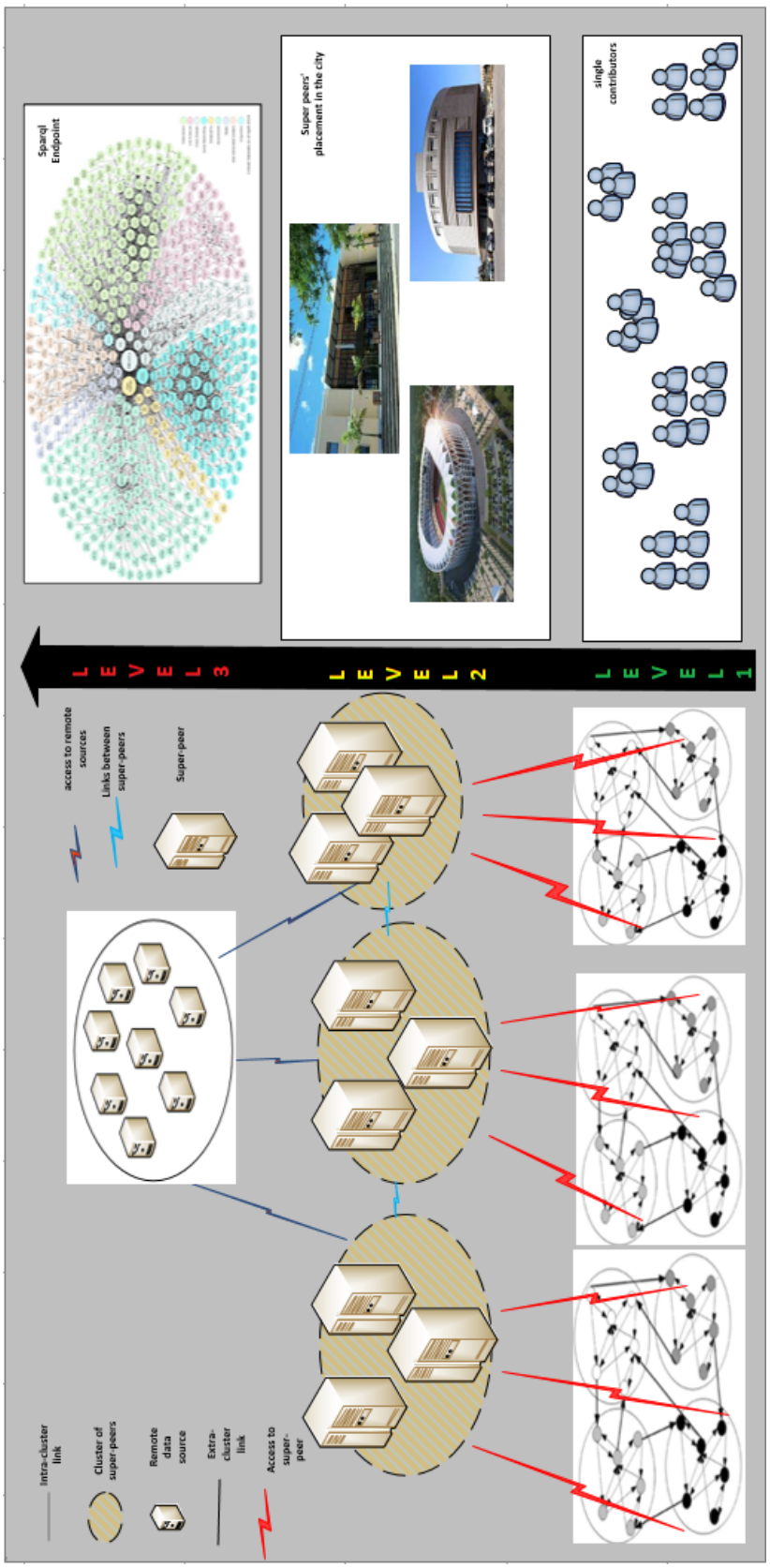

Figure 5.2: MoRAI Architecture: mobile peers (black dot), super-peers and remote sources.

Peers: we consider each browser as a peer participating in the architecture and they form the level 1. To each peer is attached a profile defined as a set of triple pattern of queries (Fig. 5.3). The profile is built with the ability to store several requests which can represent several interests. Each peer also holds a local triple store containing RDF triples. As these peers represent Web browsers, RDFStore-js [134] is used as a local triple store and query engine. Each browser hosts an RDFStore-js and is able to integrate MoRAI as a NodeJs or JavaScript client. To take mobility in consideration, a pair of geographic coordinates is assigned to all peers and is used for calculating the Euclidean distance between them.


```
{
    "filename":"q55.json",
    "GeoCoord":
        {
            "coordonnee":[{"lattitude":"80.2"},{"longitude":"80.0"}]
        },
    "query":"SELECT DISTINCT  ?x4 ?x5
             WHERE
                {
                    <http://data.linkedmdb.org/film/334> <http://xmlns.com/foaf/0.1/based_near> ?x2 .
                    <http://data.linkedmdb.org/film/334> <http://www.w3.org/2004/02/skos/core#subject> ?x3 .
                    <http://data.linkedmdb.org/film/334> <http://xmlns.com/foaf/0.1/page> ?x4 .
                    <http://data.linkedmdb.org/film/334> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> ?x5
                }",
    "name":"linkedmdb",
    "results":
            {
            "bindings":
                [
                    {
                        "x4":{"type":"uri","value":"http://www.freebase.com/view/guid/9202a8c04000641f800000000010f1c5"},
                        "x5":{"type":"uri","value":"http://data.linkedmdb.org/movie/film"}
                    },
                    {
                        "x4":{"type":"uri","value":"http://www.rottentomatoes.com/alias?type=imdbid&s=0212720"},
                        "x5":{"type":"uri","value":"http://data.linkedmdb.org/movie/film"}
                    },
                    {
                        "x4":{"type":"uri","value":"http://www.imdb.com/title/tt0212720"},
                        "x5":{"type":"uri","value":"http://data.linkedmdb.org/movie/film"}
                    }
                ]
            },
    "card":3
}
```


Figure 5.3: Example of profile content with queries and exchanged RDF triples.

- **Level 1:** On this level, we build 3 overlays for each peer to have 3 views.
  - *The first overlay RPS (Random Peer Sampling)* is generated by a random peer sampling service to initialize and keep connected the level 1. Each peer maintains a random view of size s_r.
  - *The second overlay SON (Semantic Overlay Network)* is built from semantic distance between peers to connect contributors with close semantic profiles.

Each peer maintains a semantic view of size s_s.

- *The third overlay GON (Geographic Overlay Network)* is built from the geographic distance between contributors. It allows each peer to have a view on its geographically closest neighbors. Each peer maintains a geographic view of size s_g.

- **Level 2:** this level is made up of the super-peers i.e. special peers [135] connected to Internet and endowed with more resources (storage, RAM, processor) than mobile contributors. They are organized to have a cluster of super-peers on each covered area of the architecture. This allows to ensure better data availability by anticipating possible super-peer crashes.

- **Level 3:** it consists of SPARQL Endpoints online. It is accessible to level 2 super-peers connected to Internet. It receives requests not satisfied at level 2 and follows a classic Web client-server approach.

Requests are first processed by the neighbours on level 1. In case of failure or unsatisfactory response the request is transmitted to level 2 of the architecture where it will be processed by a super-peer covering the area. The latter can also route the request to another cluster in case of failure before routing as a last option to level 3 of the architecture i.e. SPARQL Online Endpoints. A response is considered unsatisfactory or incomplete when the expected number of responses is different from the one received. Expected number of response refers here to the cardinality of each request held by a peer. For example in our simulation (section 6) the cardinality of the request represents the number of RDF triples on the graph that match the request. Thus each request is associated with a cardinality, which allows to calculate for each pair and each round the local completion rate of the request defined by the ratio between the number of responses received and the cardinality.

To form their views, peers rely on two proximity functions $S(P, Q)$ and $G(P, R)$ which, given peers P, Q and R, give respectively a metric of the semantic proximity between P and Q and a metric of the geographical proximity between P and R. The more P and Q send semantically similar requests, the higher $S(P, Q)$ is. The closer P is geographically

to R, the higher $G(P,R)$ is. We aim to provide P with: (a) a semantic view composed of the peers $Q_1, Q_2, ...Q_{s_s}$ such that $\sum_{i=1}^{s_s} S(P,Q_i)$ is maximum and (b) a geographical view composed of the peers $R_1, R_2, ...R_{s_g}$ such that $\sum_{i=1}^{s_g} G(P,R_i)$ is maximum.

5.5 Algorithms and processes for each level

In the rest of this chapter, we focus on the construction and evaluation of level 1 and 2 of the MoRAI model. Level 3 is relying on classical SPARQL protocol connections and will not be covered here. In MoRAI, member peers' profiles are initialized according to the three views they hold. On each level and at each cycle, a predefined process is executed (GeoRank Algorithm 1).

Algorithm 1 GeoRank Algorithm

Require: P a peer profile, Candidates a set of peers profiles of size c

Ensure: G a view of size s_g

1: $G \leftarrow \emptyset$
2: **for** $i \leftarrow 1$ to c **do**
3: $\quad distance \leftarrow ComputeDist(P, Candidates(i))$
4: $\quad SetScore(P, Candidates(i), distance, score_{P,C_i})$
5: **end for**
6: $bestCandidates \leftarrow RankCandidatesByScore(score_{P,C_1}, score_{P,C_2}....score_{P,C_c})$;
7: **for** $j \leftarrow 1$ to s_g **do**
8: $\quad G \leftarrow G \cup \{bestCandidates(j)\}$
9: **end for**
10: **return** G

11: $SetScore(C_1 \in Candidates, C_2 \in Candidates, distance\ d, score_{C_1,C_2})$
12: $score \leftarrow 0$
13: **if** $d > 0$ and $d <= 15000$ **then**
14: $\quad score \leftarrow score + 15$
15: **end if**
16: **if** $d > 15000$ and $d <= 30000$ **then**
17: $\quad score \leftarrow score + 10$
18: **end if**
19: **if** $d > 30000$ and $d <= 45000$ **then**
20: $\quad score \leftarrow score + 5$
21: **end if**
22: **if** $d > 45000$ **then**
23: $\quad score \leftarrow score + 0$
24: **end if**
25: $score_{C_1,C_2} \leftarrow score$

5.5.1 Level 1: construction and evolution of simple peers overlays

To build the RPS and SON overlays we reproduced and extended the approaches of [2,34, 129]. We added a method to build the GON and modified the execution phase of these protocols to allow each peer to query a super-peer.

- `Random Peer Sampling (RPS) overlay` is initialized and maintained with the CYCLON [17] protocol. To update the RPS, a peer periodically performs a shuffling i.e., selects the oldest peer from its view and they exchange parts of their views (a subset of their entries). This view is used for the bootstrap and the maintenance of the semantic and the geographic overlays. The random choice also avoids cases of peer isolation.

- `Semantic Overlay Network (SON)` is built on top of the RPS using the ranking algorithm defined in Snob [129]. It uses triple patterns containment to define similarities among profiles and assigns weights to the containment to rank neighbours. SON clusters browsers according to their profile. The maintenance of s_s-best neighbours is performed at RPS exchange time applying a similarity metric to the queries. To compare peer profiles (defined in section 5.4), we test triple pattern containment by searching for substitution of variables in the triple patterns.

- `Geographic Overlay Network (GON)` the GON is also built on top of the RPS. To calculate distance between peers we use the geographical coordinates embedded in each profile. At the execution of the shuffling each peer also updates its geographical view. This update is done by replacing a neighbour in the GON with the oldest one from the RPS. The *GeoRank* algorithm then allows to order all these neighbors and to assign a score to each distance between neighbors. The *RankCandidatesByScore* function returns Canditates with peers ranked by their computed score $score_{P,C_i}$. The *ComputeDist* function returns the Euclidean distance between two peer profiles. The *SetScore* function assigns scores to distances.

In Level 1 the predefined process consists of three sub-processes executed in each cycle

by each participating peer: Periodic Shuffling, DataStore Update and Mobility Execution (Figure 5.4).

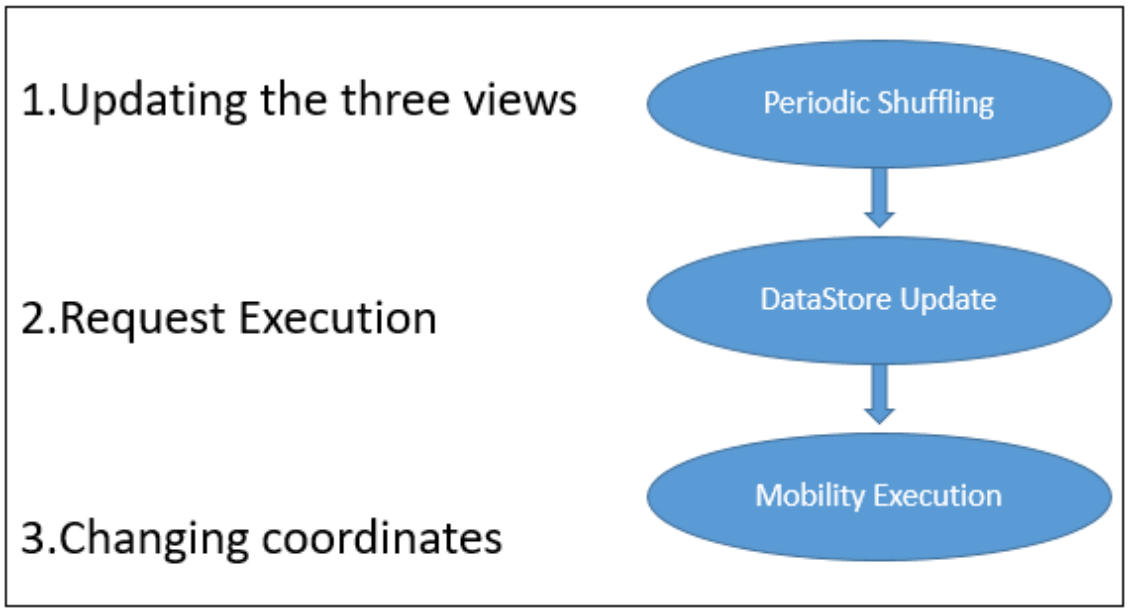

Figure 5.4: Peer executed sub-processes

Periodic Shuffling (Figure 5.5): in this sub-process the three RPS, SON and GON views held by the peer are updated. For each view, the update is started with a choice of a neighbor. Suppose that a peer P participating in the architecture is initialized with a view $V = \{V_1, V_2, ...V_{s_v}\}$ and V_i ($1 \leq i \leq s_v$ with s_v the corresponding view size) a neighbor of P belonging to V. The update of V starts with a choice of a V_i. For the RPS this choice is made in a random way, while it is based on profiles similarity for the SON case, and for the GON it is based on the geographical closeness. Finally a data exchange is started with V_i in case it is active, otherwise V_i is removed from V.

DataStore Update (Figure 5.6): this sub-process updates the local RDF triplestore of each peer. The update starts by sending all local SPARQL templates to all neighbors (RPS, SON and GON). Each neighbor receiving these SPARQL queries executes them locally and returns responses to the initiating peer. All RDF results are then stored in the initiator local RDF triplestore.

Mobility Execution (Figure 5.7): this sub-process permits to consider peer mobility. Dur-

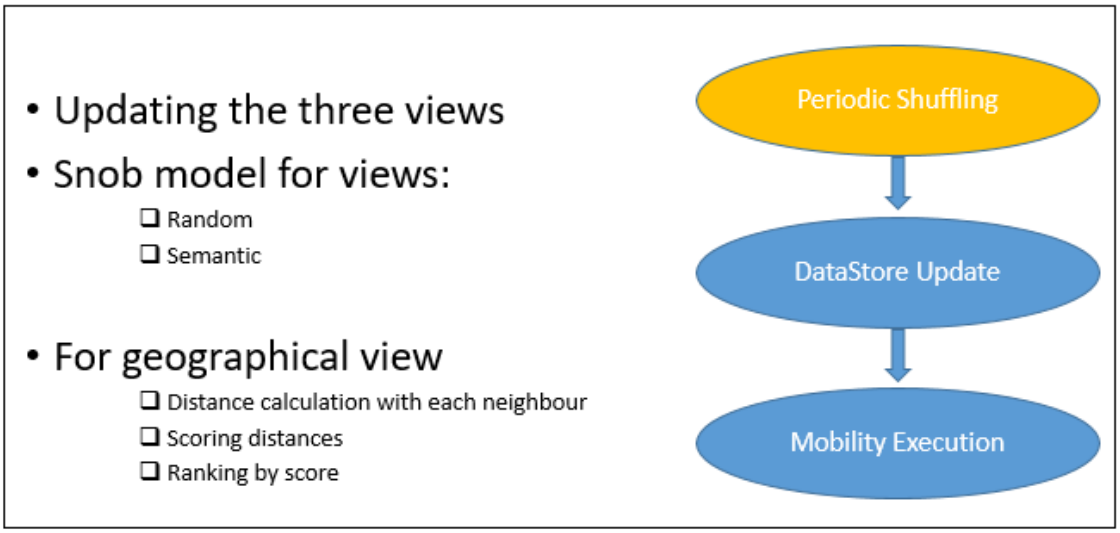

Figure 5.5: Periodic Shuffling sub-process

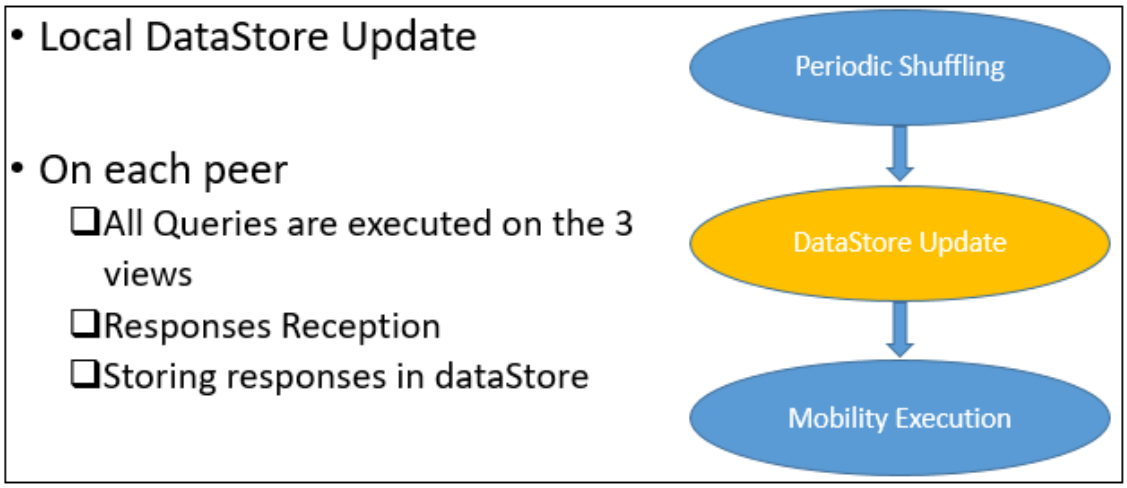

Figure 5.6: DataStore Update sub-process

ing its execution, peer's geographical coordinates may be updated to a new position. This makes it possible to prepare the next GON shuffling. Considering peers' mobility, displacement induces a coordinate update which then implies a GON update. This provides opportunity to take advantage of new neighbors to acquire new data responding to local requests.

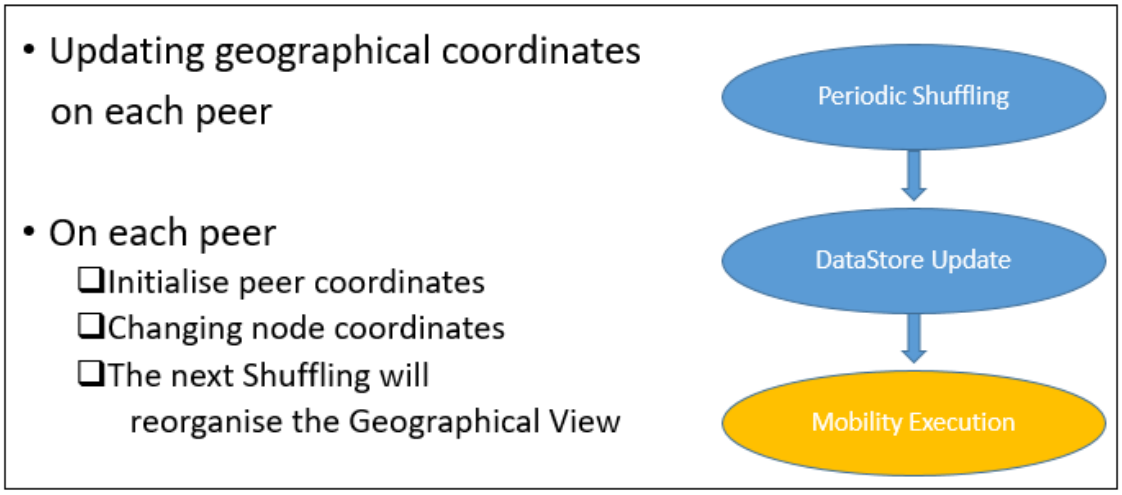

Figure 5.7: Mobility Execution sub-process

5.5.2 Level 2 : super-peers organisation and processing schemes

This level consists of super-peers covering the architecture's target zone. By analogy to our general motivating scenario in Section 1.3, the zone could be the city of Saint-Louis. They are set up according to the structure of a structured peer-to-peer architecture allowing requests to be directly routed between super-peers.

Super-peers are organized so as to cover the entire area and grouped into clusters on each zone. They are physically supported by public buildings that can ensure stable Internet connectivity and energy autonomy. We make this hypothesis because in reality, in large African cities in general, Saint Louis in particular, public structures such as the prefecture, the governance, the art museum, etc., are provided with electrical generators to compensate probable power outages. These structures are also connected to Internet by special lines provided by operators to ensure permanent service stability and continuity. Installing WiFi terminals on these structures allow us to cover the different areas targeted by MoRAI and to offer local connectivity to users.

In the initial state, the data graph is distributed over all super-peers according to the different thematic. This means that on each cluster of super-peers is applied a total

replication [56] of a part of the graph (a sub-graph) representing a given thematic. The super-peers belonging to the same cluster thus locally store (in local triplestore) the same sub-graph. This allows the system to reduce data exchange in the cluster. On the other hand, extra-cluster links are established between different cluster of super-peers. This allows the routing operation between clusters to recover the totality of possible answers for a given request.

On level 2, apart from the initialization phase, two sub-processes (Figure 5.8) are executed by super-peers for each new request received from level 1 peers: Execution/Routing and Compilation/Restitution sub-processes.

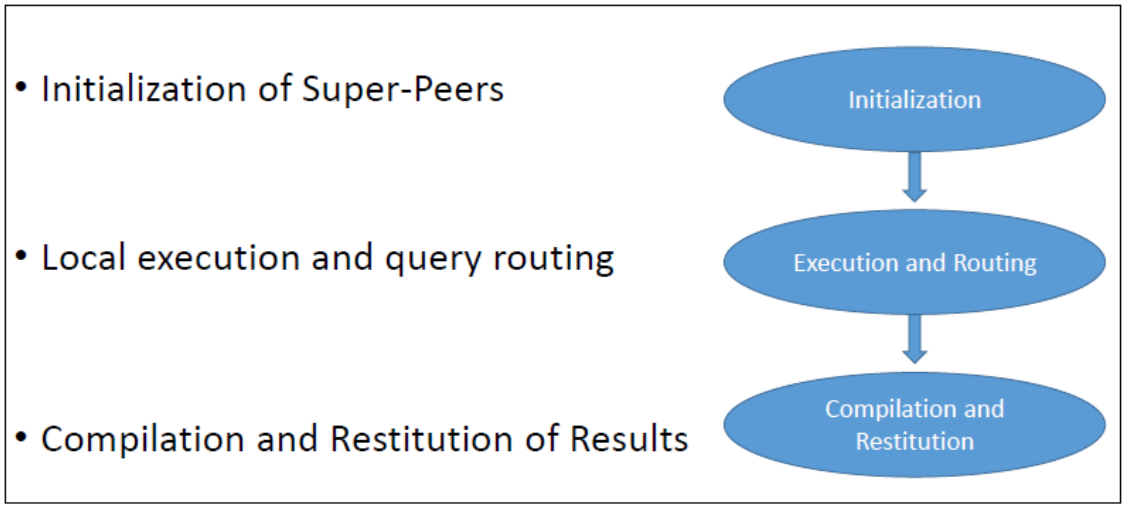

Figure 5.8: Super-Peer executed sub-processes

Execution/Routing (Figure 5.9): each peer of level 1 chooses, at each cycle, its super-peer contact according to the zone where it is located and its distance from the super-peer. The nearest super-peer according to distance is chosen as contact and the request is sent. The contact super-peer then locally processes the request and routes it to neighboring super-peers from different clusters. Routing operation is carried out through the existing extra-cluster links.

Compilation/Restitution (Figure 5.10): each neighboring super-peer receiving a request through an extra-cluster link performs a local processing before sending its answers back to the initiating super-peer. Once all responses are received from routing operation, the

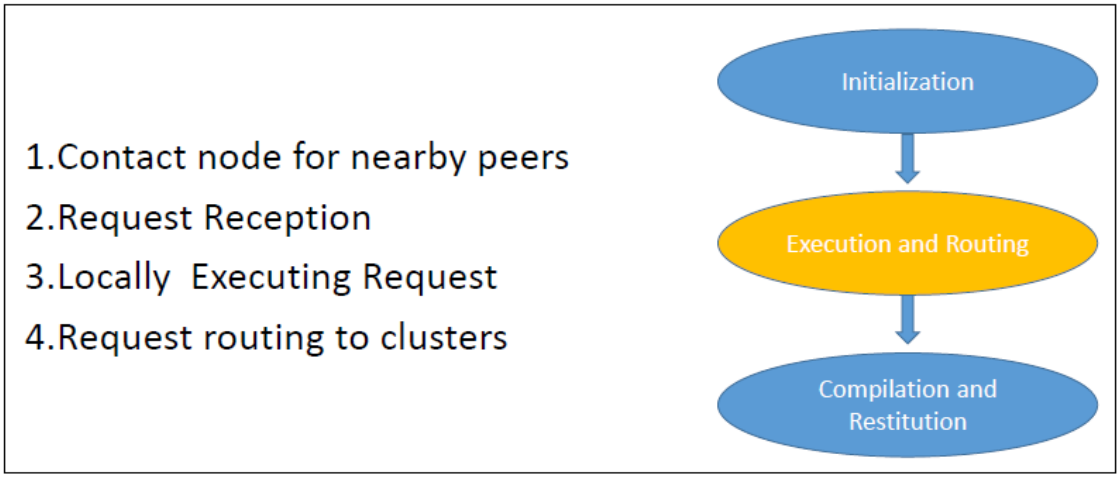

Figure 5.9: Execution/Routing sub-processes

contact super-peer compiles all these responses, updates the local datastore (the local sub-graph) and forwards data to level 1 peer initiating the request.

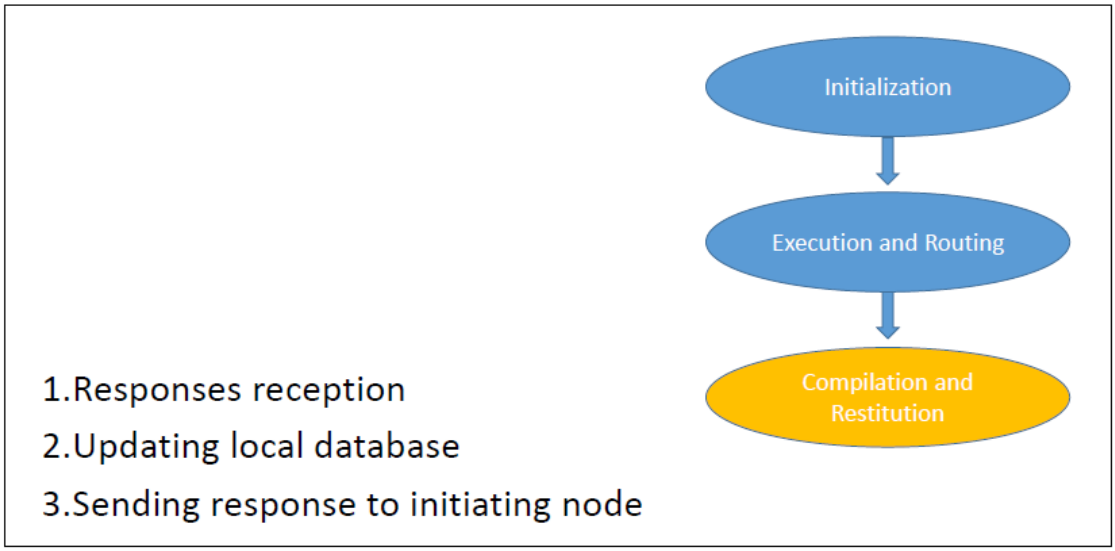

Figure 5.10: Compilation/Restitution sub-processes

5.6 Conclusion: MoRAI implementation overview

In this chapter, we proposed MoRAI (Mobile Read Access in Intermittent Internet connectivity), a distributed peer-to-peer architecture organized in three levels dedicated to data exchanges by mobile contributors. We designed MoRAI to support the local exchange of RDF data. We presented the conceptual and technical aspects of the two main levels of the architecture consisting of single peers, representing mobile devices participating to the RDF data sharing system, and super-peers connected to Internet, endowed with more resources and organized to constitute a cluster on each covered zone of the architecture.

Our next step is to deploy and evaluate this approach in an application allowing data sharing during a real event. We also plan to design a more advanced version of MoRAI which will integrate the write access of peers so they can collaboratively modify the RDF graph they host. This will allow us to consider in the future the generalization to other scenarios such as distributed semantic tagging in museums or tourism information sharing.

Chapter 6

Experimental Evaluation of MoRAI Through Simulation

6.1 Introduction: MoRAI's evaluation process

To evaluate MoRAI, we experimented the architecture in simulated environments configured from the characteristics of our motivating scenario. Our objective is to have a simulation environment to reproduce, compare and evaluate different architectures from the state of art and from our own design.

We compared the performance of MoRAI to Snob model [129] with the all usual metrics: request completion rate, network load (number of messages on the network) and resistance to members' failures. Our general hypothesis is that the introduction of super-peers and GON will lead to a better query completion rate than Snob without a significant increase (or acceptable increase) of network load. This will allow us to maintain the limit on the number of messages which is one of the strong points of Snob ($number\_of\_messages < (all\_overlays\_size * |query|)$). We also hope to maintain an optimal completion rate when MoRAI faces failure scenarios. We are also looking at the number of requests received and processed by Level 2 super-peers to determine this level's impact on the completion rate.

Experimental results showing the relevance of: a) considering geographical positions during exchanges and b) integrating RDF graph replication. Results precisely show the impact of these two points on the completion rate and the system's resistance to crash scenarios. The next section presents our experimentation environment. Section 3 provides experimental results. Section 4 concludes the chapter.

6.2 Experimental Setup on top of Peersim

We build our experimentation environment with the Peersim [46] tool which allows us to simulate scalable and dynamic P2P environments and which is particularly dedicated to epidemics protocols. We parameterize the environment with 196 contributor peers and 4 super-peers. We use real datasets used in the evaluation of Snob: Diseasome[1] and Linked Movies Database[2] (LinkedMDB). Diseasome is a network of 4300 disorders and disease genes linked by known disorder-gene associations for exploring all known phenotype and disease gene associations. LinkedMDB is an RDF graph containing a linked-data collection of movies, actors, directors, and the relationships between all of those entities. These datasets represent two points of interest for the participating peers. We add to this dataset a manually created dataset (GeoCoord) containing geographic coordinates. These coordinates are representative of the initial positions of the different participating peers. They are chosen so that the distances between peers range from 1 to 50000 meters. We choose these parameters to get closer to our example scenario of Saint Louis festival.

In this scenario the 196 peer contributors would represent people attending the festival. As we do not have real and reliable data on the festival's topics, we are using data related to Disesomes and LinkedMDB, which will represent the two major topics of the festival, namely concerts and cultural activities. This allows us to perform the simulation with two different points of interest.

[1]https://old.datahub.io/dataset/fu-berlin-diseasome

[2]http://data.linkedmdb.org/

Our 4 super-pairs organized in two clusters of two super-pairs represent in the scenario the super-pairs installed in public buildings with reliable Internet access and high storage capacity. The two clusters would be located, for example, at the city's Stadium where concerts will take place and at the Arts Museum where cultural activities will be held. We have chosen a maximum distance of 50000 meters between participants to contain all participants within a limited perimeter. For reproducibility the complete dataset is available online[3].

Each peer is thus initialized with a set of randomly selected triples in Diseasome and LinkedMDB. Each peer is also initialized with random geographic coordinates from GeoCoord. The super-peers are parameterized to form two clusters of two super-peers. This means that the graph is split into two parts (sub-graph), each part is stored by a cluster. The super-peers belonging to the same cluster therefore hold the same sub-graph. We also note that we set the system such that the two sub-graphs are representative of the two topics considered (Diseasome and LinkedMDB), which also implies that the clusters are thematic.

We reuse the generated query set for Snob [129] evaluation. These are 100 queries for LinkedMDB and 96 queries for Diseasome. To simulate architecture points of interest, we associate geographic coordinates to each request. This allows to consider the position of the request holder peer in relation to the point of interest. Peers are initialized randomly with zero or two queries. To calculate the completion rate on each execution round, we also associate each query with a cardinality representing the size of the set constituting the data graph triples that match the query.

To simulate the mobility of the participating peers, on each execution round, an update of the geographical coordinates is done by a random choice between three pairs of coordinates which are: the initial coordinates (the peer is motionless since the last round), the coordinates carried by one of the stored requests (the peer moves towards one of its points of interest), a pair of random coordinates (the peer moves towards an arbitrary place).

[3]https://github.com/MmoooGit/MoRAI

We perform two phases of simulation: the first one with a normal execution of each cycle for all participating peers; the second one with an execution disrupted by crash scenarios to observe the impact of peer failures on the different metrics.

Normal Execution: A total of 196 participant peers are initialized. 98 of them hold 2 requests each and the remaining 98 hold 0 request. That is a total of 196 requests loaded for the simulations. Given that the Snob model (RPS+SON) has already made comparisons with the RPS model and has proven its efficiency, we start here from the Snob model which we compare with the RPS+SON+GON and RPS+SON+GON+SP models.

- `RPS+SON`: This is Snob model simulated in our environment for reproducibility and comparison. We modify the initial model (RPS+SON) by initializing the peers with 2 queries (2Q) instead of 1 for the initial model.
- `RPS+SON+GON`: idem and we add a geographical view (GON) to take into account the location of the peers during execution. We analyze this scenario to see the impact of the addition of the geographical view on the query completion rate when the protocol is executed only on level 1.
- `RPS+SON+GON+SP`: idem and we add the level 2 of the architecture represented by the super-peers (SP). This scenario is chosen in order to have a global view of the effectiveness of MoRAI. It also allows us to appreciate the relevance of integrating super-peers on the system.

Crash Execution: to simulate peers arrivals and departures, we introduced the crash scenarios. We are interested here in peer failures since in our context they are the most volatile compared to super-peers. Our crash simulations will focus only on peer failures. In these scenarios we choose a series of execution rounds during which we drop a specific number of peers. In order to get an accurate view of the impact of the requests, these peers are chosen from those that hold requests.

6.3 Experimental Results

In this experiment, we focused on the conceptual aspect of MoRAI by means of a simulation with Peersim. Peersim allows us to simulate MoRAI's execution while abstracting from the aspects related to connectivity between nodes. In a real environment, the connectivity between nodes (represented by browsers) would rely on WebRTC.

6.3.1 Completion rate: real impact of localisation and super-peers

We have observed the completion rate on both simulation phases (normal and crash). We notice a clear improvement of the completion of the RPS+SON+GON model compared to the RPS+SON model (see Figures. 6.1 and 6.2). This can be mainly explained by the fact that, at each execution round, the GON brings new data to the initiator peer. Based on the hypothesis that geographic neighbours can be an important reserve of responses, choosing to query them increases the number of neighbours potentially holding favourable query responses. An increase of the number of neighbours to be requested in the SON could also improve completion but, considering the unstable and unreliable nature of connectivity in the context of our study, requesting geographical neighbours would be less time consuming since closer peers are to each other, lower is latency. Moreover, our hypothesis on the geographical neighbors does not exclude the fact that they can be semantically close to the initiator peer.

The completion rate gains a much greater improvement when we add the super-peers to the architecture. Indeed we notice on the curve that the completion rate reaches 100% after 12 rounds on average against 25 rounds for the RPS+SON+GON model and 45 rounds for the RPS+SON model (Figures. 6.1 and 6.2). This is explained by the fact that the data graph is fully replicated by the two clusters of super-peers and that the inter-cluster links allow super-peers from different clusters to exchange. This allows each participating peer to receive the maximum possible response to its requests. In order to have an accurate estimation of the impact of super-peers, we have configured the simulation environment

so that each participating peer has only one access to the super-peers. We restricted this access to super-peers to the first round to show their impact as architecture starter. From this single access we can see the impact of super-peers on request completion rates in the next rounds. This is visible from the second round with a peak on the completion curve. In real scenario, the same impact will be observed each time super-peers are requested.

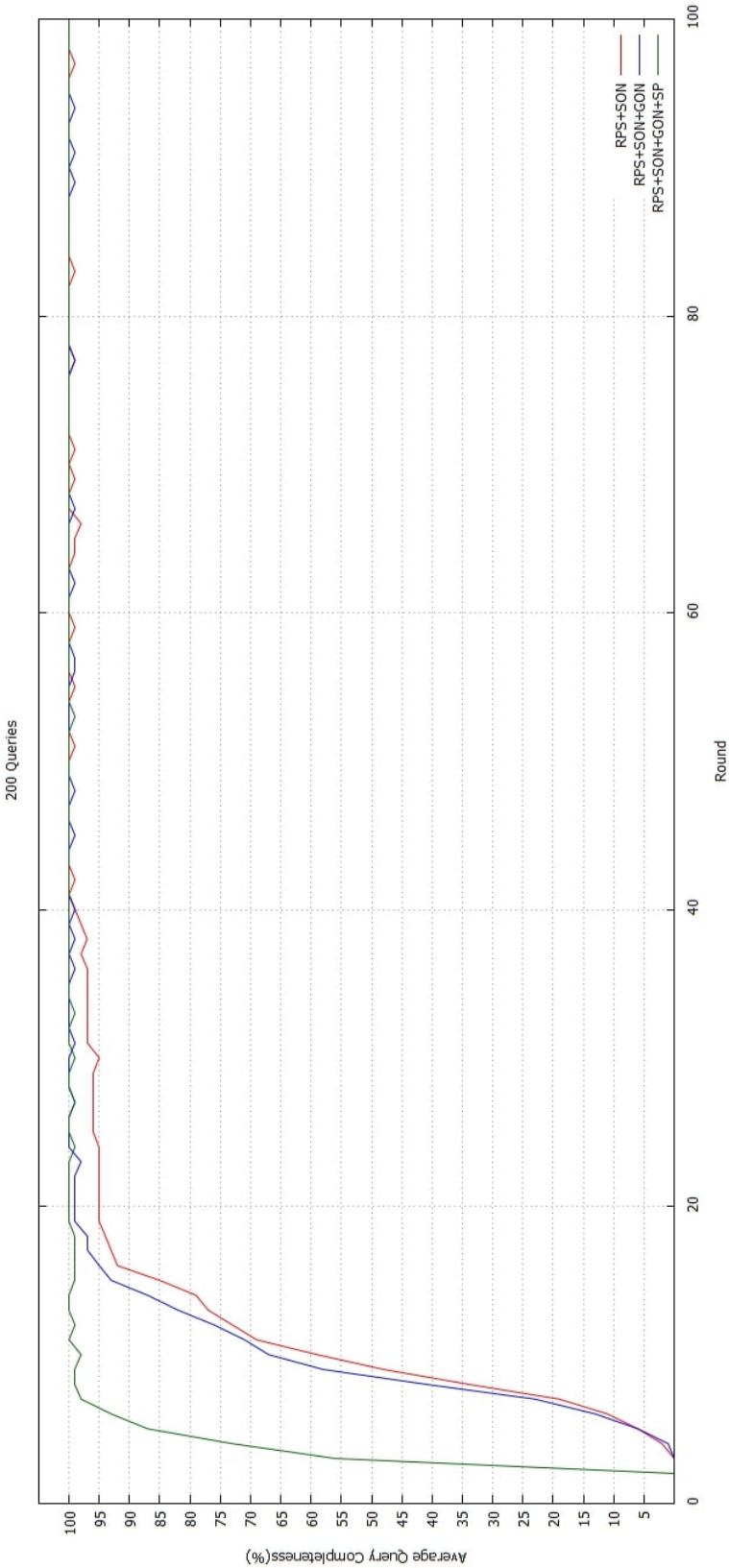

Figure 6.1: Average query completeness by round for RPS+SON, RPS+SON+GON and RPS+SON+GON+SP, with 196 peers, 196 queries, 2 clusters of 2 super-peer. RPS, SON and GON size: 10 (5 for shuffling size)

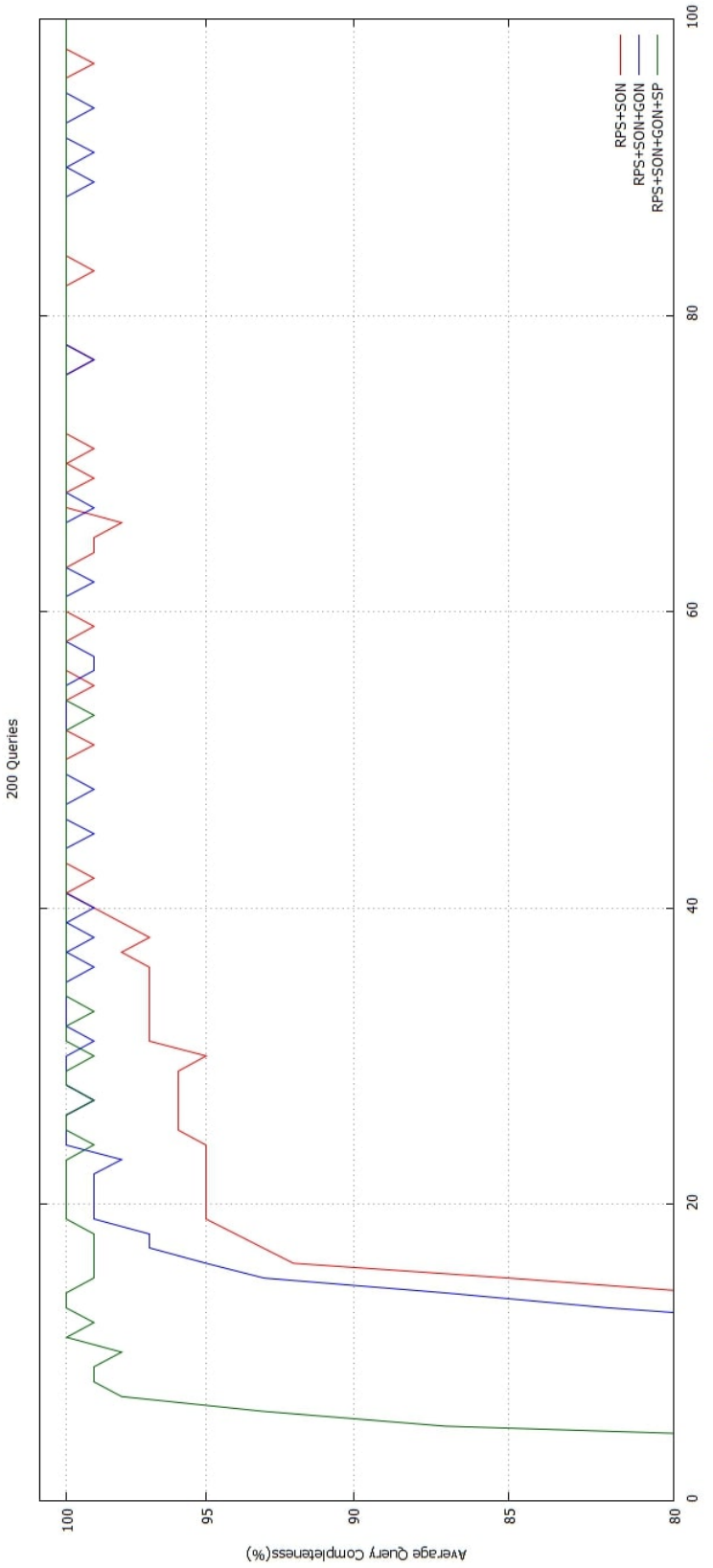

Figure 6.2: Average query completeness by round for RPS+SON, RPS+SON+GON and RPS+SON+GON+SP, with 196 peers, 196 queries, 2 clusters of 2 super-peer. RPS, SON and GON size: 5 (5 for shuffling size)

6.3.2 An acceptable network load

The load observed for the RPS+SON+GON and RPS+SON+GON+SP models is 33% higher than the load of the RPS+SON model (see Figures. 6.3 and 6.4). This is explained by the fact that GON involves additional exchanges on the network. Indeed, for each participant, in addition to the exchanges generated during the shuffling at the RPS and SON level, the GON is called upon for local processing of requests during the periodic shuffling. This load increase is, however, compensated by the completion clearly visible on the curve. The two models RPS+SON+GON and RPS+SON+GON+SP generate approximately equal amounts of messages. This is due to the fact that exchanges between level 1 and level 2 generate few messages since we have limited them to a maximum of one message per peer. This translates into a maximum increase of 96 messages over the 100 execution rounds, or 4,8% of the load of the RPS+SON+GON model.

Although this load increase is not negligible, it is nevertheless acceptable in our scenario because if we consider that the limit of the total number of messages is defined by $(R_s + S_s + G_s) * |queries|$ i.e (5+5+5)*196= 2940 messages, we note on the curve that the maximum of the number of messages is 2003 messages thus lower than the limit 2940 messages. We can suppose this limit of the total number of message as only one triple pattern query is sent to each neighbour. Also the load induced by the exchanges with super-peers is acceptable since we have a limited number of super-peers and the objective is that they be requested as less as possible.

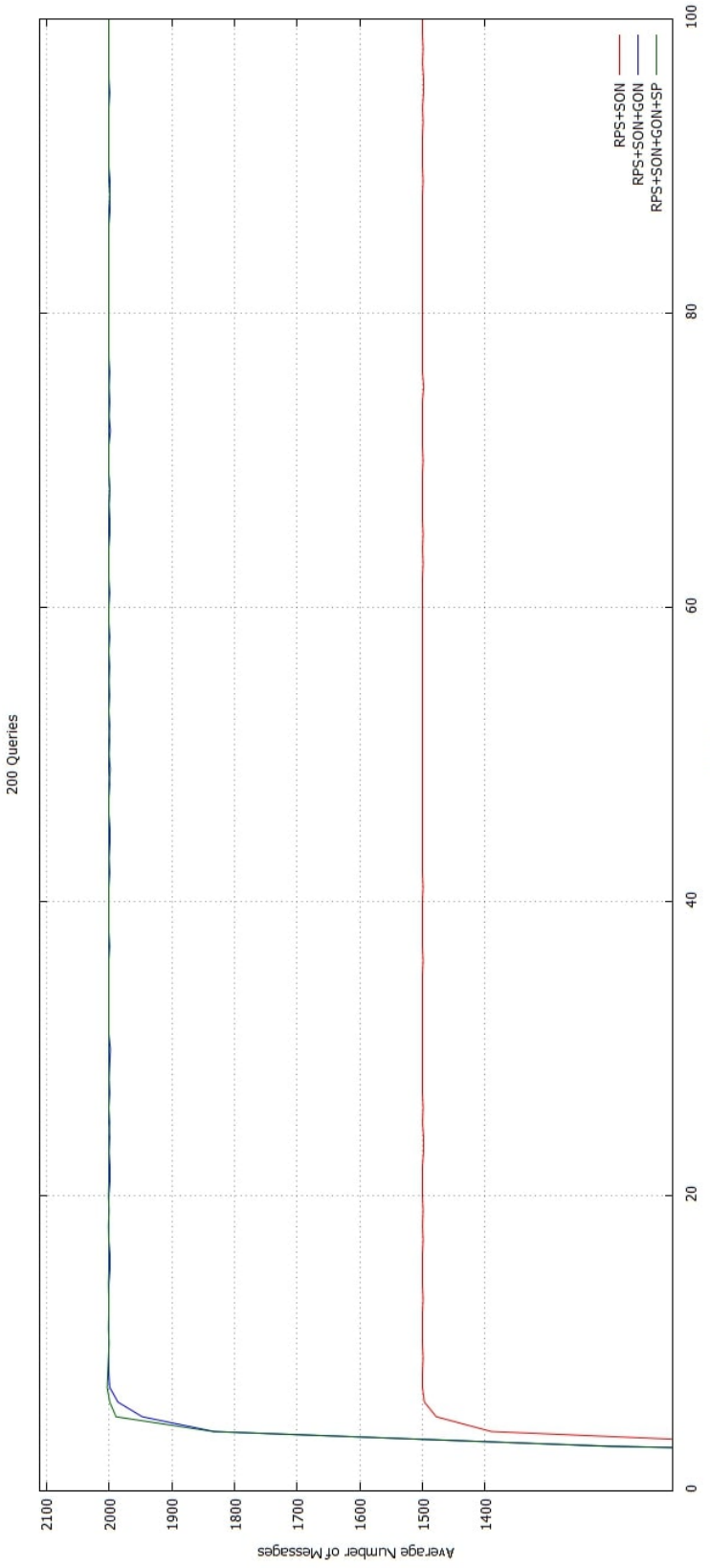

Figure 6.3: Average number of message by round for RPS+SON, RPS+SON+GON and RPS+SON+GON+SP with 196 peers, 196 queries executed, 2 clusters of 2 super-peer. RPS, SON and GON size: 10 (5 for shuffling size).

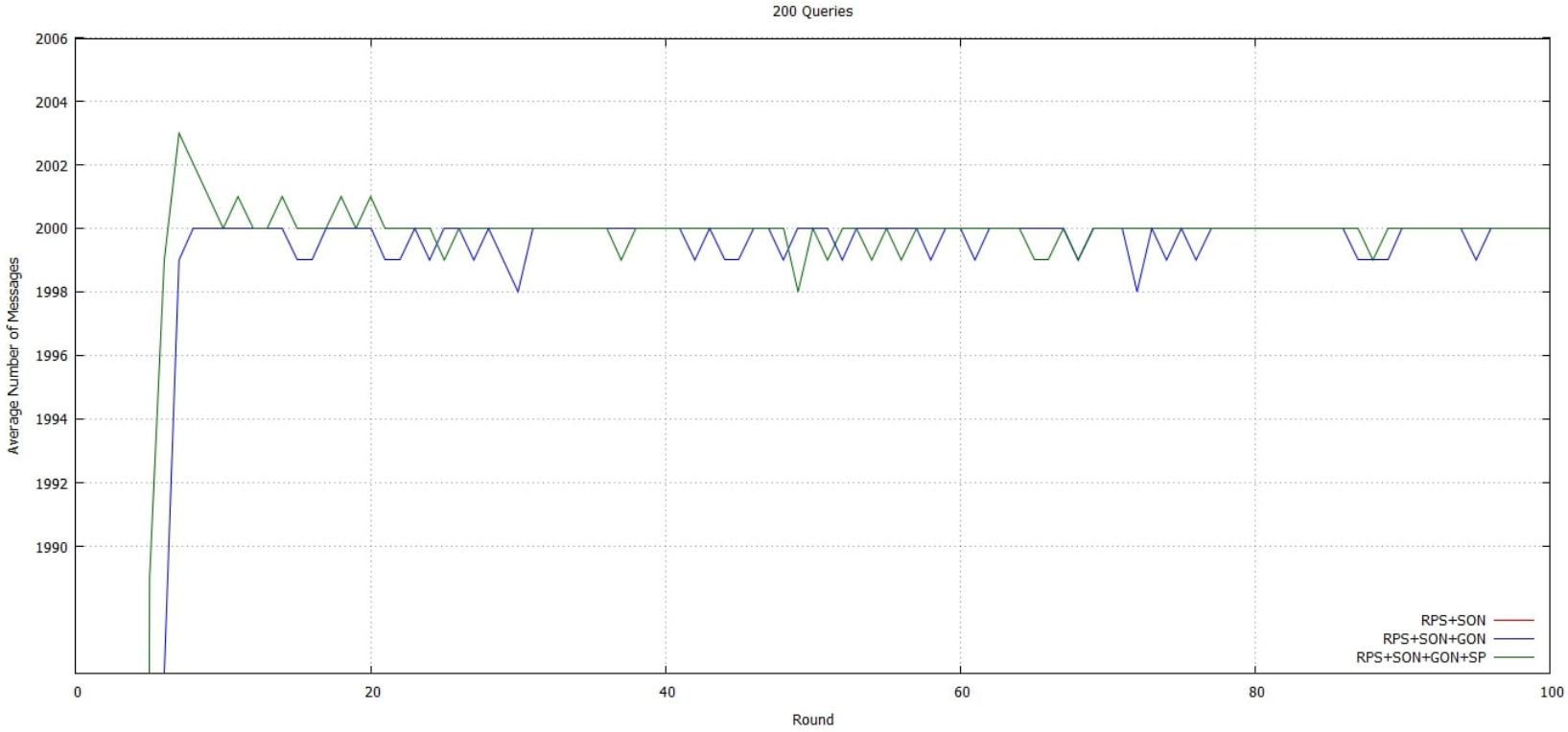

Figure 6.4: Average number of message by round for RPS+SON+GON and RPS+SON+GON+SP with 196 peers, 196 queries executed, 2 clusters of 2 super-peer. RPS, SON and GON size: 5 (5 for shuffling size)

6.3.3 A negligible impact of failure

We observe a global resistance of the system to the provoked failures. On the three models RPS+SON, RPS+SON+GON, RPS+SON+GON+SP the curve (see Figures. 6.5 and 6.6) shows a fall of the completion approximately equivalent over crash intervals. Indeed the three models show similar reactions to the crash scenario. The curves reach the same completion rates within the predefined crash intervals. This is precisely explained by the fact that the completion rate is calculated as a function of the number of living (up) peers in the system and the local completion rate of each of these peers. The local completion rate of a peer is obtained as a function of the total number of potential responses available on the entire graph to answer its local requests. Since the number of peers falling on the different crash intervals are equivalent across all three models, completion follows the same equivalence. However, we notice on the curve a slight completion difference at the intervals [25-30] and [33-38]. This is caused by the overall completion level of the RPS+SON+GON+SP model which, for these intervals ([25-30] and [33-38]) had already reached 100% of completion against (respectively) 98% and 99% for the two others.

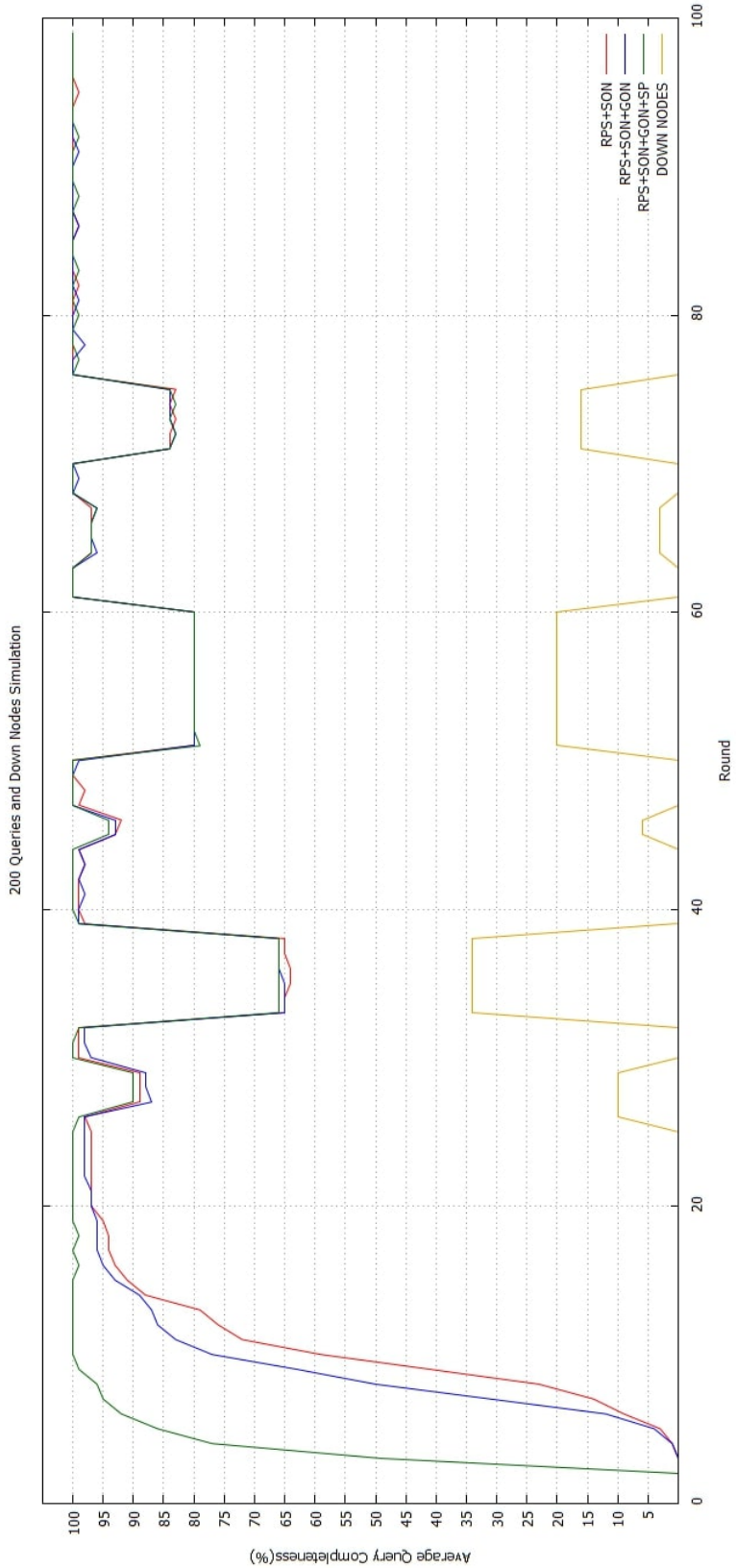

Figure 6.5: Average query completeness by round with crash scenario. DOWN NODES yellow curve at the bottom shows the number of nodes down per round. RPS+SON, RPS+SON+GON and RPS+SON+GON+SP are tested with 196 nodes, 196 queries executed, 2 clusters of 2 super-peer. RPS, SON and GON size: respectively 10 (5 for shuffling size)

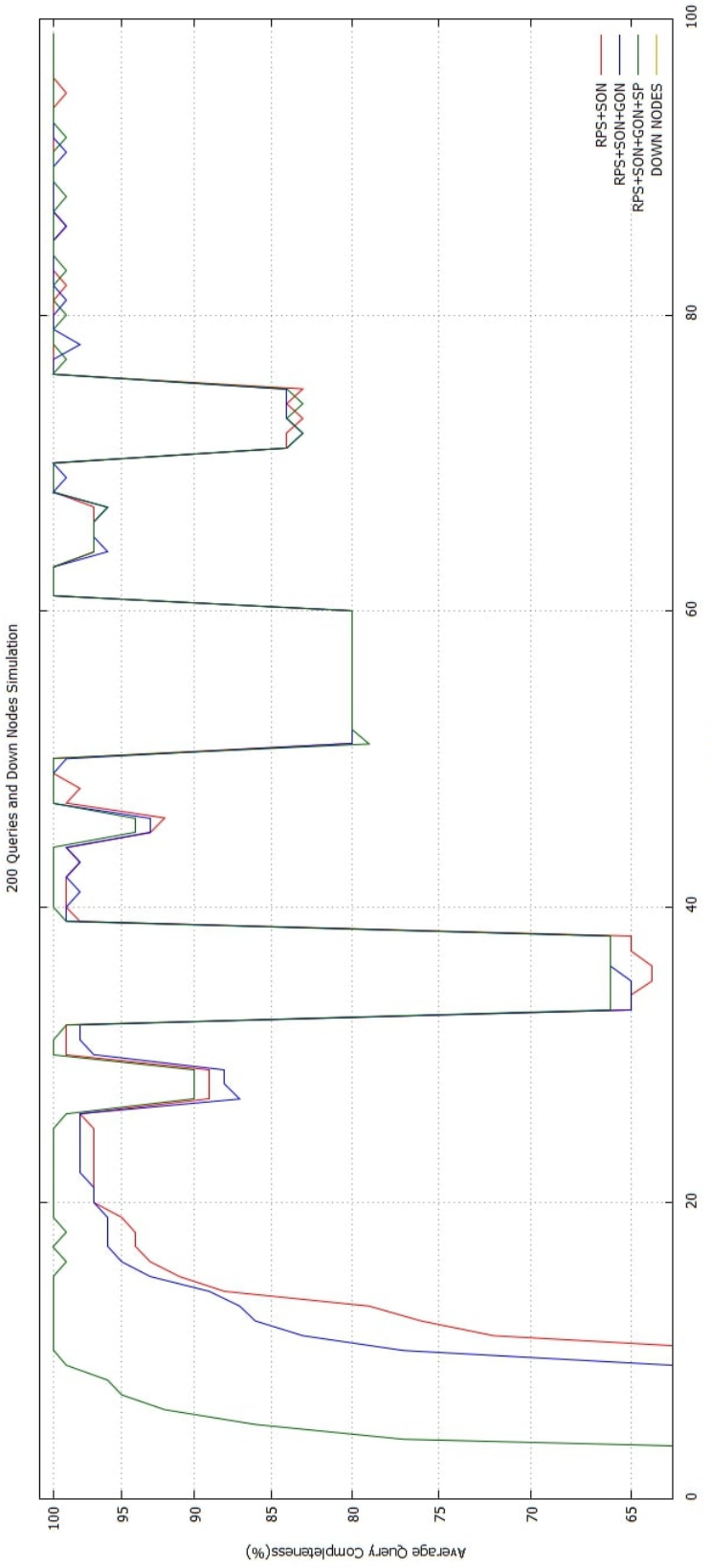

Figure 6.6: Average query completeness by round with crash scenario. RPS+SON, RPS+SON+GON and RPS+SON+GON+SP are tested with 196 nodes, 196 queries executed, 2 clusters of 2 super-peer. RPS, SON and GON size: 5 (5 for shuffling size).

6.4 Conclusion: the relevance of MoRAI

In this chapter we carried out an experimental evaluation of MoRAI through extensive simulation configured to capture key aspects of our motivating scenario of supporting data exchange between the participants of a local event.
In this simulation phase, we tested the architecture design mode which allowed us to generate an architecture with 3 levels organised in clusters. We also tested MoRAI execution process. This allowed us to evaluate the performance of MoRAI in terms of completion rate, failure resistance and network load. We used real data from Diseasome and LinkedMDB, and Sparql queries adapted to this dataset. This allowed us to align our simulations to Linked Data access scenarios. In a eventual deployment phase in real environment, MoRAI client could be a NodeJS or JavaScript client running on a browser. Connectivity between browsers should be supported by the WebRTC layer.
In terms of the requests completion rate and the system's resistance to crash scenarios, the results show the relevance of considering geographical positions during data exchanges and of integrating RDF graph replication to ensure data availability on the architecture.

Chapter 7

Conclusions and perspectives on MoRAI, a Mobile Read Access in Intermittent Network Access Contexts

In a context of significant growth in the number of smartphone users on the African continent, the Internet penetration rate is still relatively low compared to other continents, especially Europe, North America and Asia. GSM Association (GSMA) 2020 report on mobile economy in Sub-Saharan Africa [11] predicted that over the next five years, the number of smartphone connections in Sub-Saharan Africa will almost double to reach 678 million by the end of 2025, representing an adoption rate of 65% of the population. Internet Words Stats data [10] of December 2020/June 2021 estimated Internet penetration rate at 46.2% in Africa compared to 93.9% in North America, 87.1% in Europe and 63.8% in Asia. This general observation led us to address, in this thesis, the issue of data access in environments with resource-limited networks. More precisely, our objective in this manuscript has been to answer the question of the feasibility of data sharing by mobile

contributors, in situations where Internet access occurs intermittently. We targeted a scenario where users form a P2P network such that anyone can generate information and make it available to everyone else on the network.

7.1 Thesis summary: an efficient geography and semantics based sharing architecture

Starting with chapter 2, we have reviewed the solutions (models, architectures, protocols, etc.) related to network architectures that, in the current state of the art, are relevant to our general objective. We proceeded to a comparison of the functional approaches of the gossip protocols based on the adopted membership mechanism. The gossip protocols with a two-overlay membership mechanism have proven to be very efficient in contexts similar to ours due to their ability to automatically form clusters of interest or proximity aimed to group together peer members of the architecture that have one or more common interests or that are close according to a given similarity metric.

This led us to consider the idea of integrating the location dimension into the process of building a gossip protocol. In our proposal the neighbourhood of each contributor is constituted by semantically close peers, having a certain geographical proximity, i.e. a maximum limit in terms of geographical distance is set between the peers considered as neighbours.

Then, in chapter 3, we addressed the issue of collaborative data sharing systems with a particular focus on the data model adopted by these systems. In this chapter, we have proposed a classification of approaches dedicated to designing decentralized data sharing systems adopting an RDF data model. This classification led us to distinguish four different approaches with conceptually different principles:

- Graph replication: the graph is stored on all peers in the network that can modify and/or delete portions of it independently.

- Distributed graph and structured peer-to-peer system (case of DHTs): the graph is distributed across all peers on a structured peer-to-peer environment.
- Distributed graph and semantic overlay: the graph is distributed over all participants by adopting a semantic peer-to-peer network architecture.
- Cloud and Fog computing: the graph is hosted on the cloud.

The comparative analysis of these approaches we performed was based on the layout of the architecture graph, i.e. whether it is replicated (partially or totally), or distributed, or shared and modified in a collaborative way through the involved peers in the architecture.

We also saw that these approaches have been successfully applied on architectures based on gossip protocols (Example: Crate [40], Swooki [69] and Voulgaris et al. [99]). Crate [40] adopts the graph replication approach with a graph represented by a document shared by all contributors. Crate also uses Spray gossip protocol [29] to distribute update operations to all collaborators. Swooki [69] is based on the graph replication approach with the graph represented by a semantic wiki page and update operations are routed to collaborating nodes using a gossip protocol [127]. Voulgaris et al. [99] use the semantic superposition approach to exploit the semantic structure present in document sharing systems to improve search performances. This system rely on SCAMP gossip protocol [24] to build and maintain the overlay network.

Building on this analysis, in chapter 4, we introduced and proposed a theoretical analysis of an approach consisting of combining, on the one hand, a logical peer-to-peer architecture whose construction and maintenance are ensured by a gossip protocol, and, on the other hand, a replication system allowing to ensure local data availability. This approach led us to a three-level architecture ensuring that contributors can build and share a local knowledge base of RDF triples.

We theorised the construction of level 1 consisting of mobile contributors by designing a two-overlay membership gossip protocol that provides a semantic overlay based on points of interest and a geographical overlay based on physical distances. The level 2 of our

architecture is dedicated to the super peers participating in the architecture and organised according to a cluster arrangement of super peers on each zone. Level 3 represents here the remote data sources (e.g. SPARQL Endpoint).

Still in chapter 4 we also theorised a data replication system with a control mechanism whose properties are the following:

- multi-master and asynchronous approaches for respectively the localisation mode and the propagation mode of update operations;
- hybrid replication system (partial and complete) for the data replication mode.

We concluded this chapter by setting out an evaluation plan for our approach with the following metrics: network load and failure resistance for robustness, latency for efficiency, high data availability for reliability.

In chapter 5, we have provided a concrete approach to our proposal of chapter 4 by conceptualizing the execution process of MoRAI (Mobile Read Access in Intermittent), an architecture dedicated to RDF data exchange through mobile applications allowing users to access and share data in environments where hardware resources are limited and Internet access is intermittent.

MoRAI rely on one hand on Cyclades [34] approach regarding the hierarchy of requests processing and on the other hand on Snob query execution model [129]. The difference comes from taking into account the location of the neighbours to increase the search perimeter and the integration of super-peers on which the data graph is replicated allowing data availability improvement.

As stated above, MoRAI concretely consists of three levels. In our proposal, in level 1 we build 3 overlays for each peer to have 3 views:

- the first overlay RPS (Random Peer Sampling) is generated by a random peer sampling service;

- the second overlay SON (Semantic Overlay Network) is built from semantic distance between peers;
- the third overlay GON (Geographic Overlay Network) is built from the geographic distance between contributors.

On level 2, we have super-peers connected to Internet, endowed with more resources (storage, RAM, processor) and organized to have a cluster of super-peers on each covered area of the architecture. Level 3 consists of SPARQL Online Endpoints and follows a classic Web client-server approach ensuring the link with the classical architecture of Linked Data on the Web.

To evaluate and validate our architecture, chapter 6 describes a simulation environment built with Peersim tool which allows to simulate scalable and dynamic P2P environment. For comparison and reproducibility, we reused a real dataset from the state of the art and previously introduced in the evaluation of Snob: Diseasome and Linked Movies Database (Linked-MDB). We added to this dataset a manually created extension (GeoCoord) containing geographic coordinates. We also reused the query set generated for the Snob evaluation [129]. It contains 100 queries for LinkedMDB and 96 queries for Diseasome. To simulate points of interest in evaluating our architecture, we associate geographic coordinates to each request. This allows us to consider the position of the request holder peer in relation to the location of a point of interest.

Chapter 6 also presented and analysed the results obtained from MoRAI's experimental evaluation. Two simulation phases were performed:

- the first one with a normal execution of each cycle for all participating peers;
- the second one with an execution disrupted by crash scenarios to observe the impact of peer failures on the different metrics.

These two phases were performed on the following models: RPS+SON (Snob), RPS+SON+GON and RPS+SON+GON+SP (with SP= Super-Peer). Results show a clear improvement of

the completion of the RPS+SON+GON model compared to the RPS+SON model. This completion rate gains a much greater improvement when we added the super-peers to the architecture (RPS+SON+GON+SP) with a rate of 100% after 12 rounds on average against 25 rounds for the RPS+SON+GON model and 45 rounds for the RPS+SON model. Results also show that network load observed for the RPS+SON+GON and RPS+SON+GON+SP models is 33% higher than the load of the RPS+SON model. Although this load increase is not negligible, it is nevertheless acceptable in our scenario because if we consider the limit of the total number of messages (defined on chapter 6) we noted that the maximum number of messages is lower than the limit.

By analysing failure impact on the three models RPS+SON, RPS+SON+GON and RPS+SON+GON+SP, results showed a fall of the completion approximately equivalent, thus, these models presented similar reactions to the crash scenario. However, we noted a small variation of the completion rate on the predefined crash intervals which can be explained by the level of the completion rate at the time of the crash, i.e. when the completion rate has already reached 100%, the fall of the curve is relatively less important (1 to 2% variation).

In the end we can conclude that MoRAI would be an efficient architecture to implement and evaluate a real application of Linked Data sharing in the context of limited or unreliable Internet connectivity.

7.2 Future directions

In this thesis, we have worked to provide an answer to the problem of access to the Web of data by mobile contributors in a highly distributed situation. Our MoRAI solution has several advantages that we have seen in the previous chapters, notably the resistance to crashes and the quality of the completion rate.

We are interested in several perspectives in order to extend the capabilities of the solution, to improve its performance and finally to offer users a quality tool that meets their

expectations, particularly in terms of autonomy, transparency and efficiency.

- **Real-life application**: One of our short term perspectives in relation to this work is to implement and evaluate MoRAI in a real application for an event such as the Saint Louis Jazz Festival. To do this we will need to work on integrating MoRAI into participants' phones or tablets via Web browsers. We will be able to use RDFStore-js [134] as a data store and a query engine that we will extend with MoRAI. Each browser will host a NodeJS or JavaScript client of MoRAI. The use of the neighbourhood of geographically close nodes will reduce the link establishment costs of WebRTC routing.
- **Write Access**: MoRAI provides read access to local and remote data sources. Write access remains a challenge in our context. We plan to extend the capabilities of MoRAI by offering users the possibility to update local or remote data sources. Users will be able to add or modify RDF triples and share the generated update operations. In Chapter 4, we theorised the adoption of the multi-master approach which will allow multiple participants to store the same replica of the graph and update it. We have also theorised the adoption of the optimistic asynchronous approach for sharing update operations. This approach will allow us to deal with the constraints related to the volatility of the participants that could interrupt the process.
- **CCI Models (Convergence, Causality preservation and Intention preservation)**: In order to ensure perfect consistency between the local and remote sources with which our participants interact, we plan to study the prospects of adapting the CCI model [67] in MoRAI after the integration of write access. As mentioned in chapter 3, this model has proven its efficiency in the literature (e.g. Su-Set [72] and Swooki [69]). The updating operations can generate inconsistencies especially in our context of intermittent access to Internet and contributors' mobility. On local sources these inconsistencies can be materialized mainly by diverging states of locally replicated subgraphs. This would imply different responses obtained from identical queries depending on the location and neighbourhood of the initiating participant.

The CCI model will allow us in this context to prevent source divergence (replicated graph), causality violation and intentional update operations on each source.

As a concluding note, we wanted to recall the reader that, in the past, many solutions designed for users with disabilities or for usage contexts with specific restrictions proved to be efficient approaches for more general, mainstream or unforeseen usages. For instance, subtitling methods were initially designed for people with hearing impairment but are now used by everyone in noisy contexts (e.g. pubs, airports) or places requiring a silent environment. We hope this could also be the case for MoRAI in a longer-term.

References

[1] A.-M. Kermarrec, L. Massoulié, and A. J. Ganesh, "Probabilistic reliable dissemination in large-scale systems," *IEEE Transactions on Parallel and Distributed systems*, vol. 14, no. 3, pp. 248–258, 2003.

[2] M. Bertier, D. Frey, R. Guerraoui, A.-M. Kermarrec, and V. Leroy, "The gossple anonymous social network," in *ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing*. Springer, 2010, pp. 191–211.

[3] J. Huang, D. J. Abadi, and K. Ren, "Scalable sparql querying of large rdf graphs," *Proceedings of the VLDB Endowment*, vol. 4, no. 11, pp. 1123–1134, 2011.

[4] F. Schuler, "Étude et utilisation des technologies des p2p," *Internet, Février*, 2005.

[5] WINSYSTEMS, "Cloud, fog and edge computing – what's the difference?" https://www.winsystems.com/cloud-fog-and-edge-computing-whats-the-difference/, 2017.

[6] T. Berners-Lee and J. Timothy, "Information management: A proposal," Tech. Rep., 1989.

[7] T. Berners-Lee, "W3 future directions," https://www.w3.org/Talks/WWW94Tim/, 1994. [Online]. Available: https://www.w3.org/Talks/WWW94Tim/

[8] ——, "Linked data," https://www.w3.org/DesignIssues/LinkedData.html, 2006.

[9] C. Bizer, T. Heath, and T. Berners-Lee, "Linked data: The story so far," in *Semantic services, interoperability and web applications: emerging concepts.* IGI Global, 2011, pp. 205–227.

[10] I. W. Stats, "Internet world stats," https://www.internetworldstats.com, 2021.

[11] GSMA, "The mobile economy sub-saharan africa," https://www.gsma.com/mobileeconomy/sub-saharan-africa/, 2020.

[12] A. Ismail, "Communautés dans les réseaux sémantiques pairs-à-pairs," Ph.D. dissertation, Aix-Marseille 2, 2010.

[13] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," in *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.* IEEE, 2003, pp. 482–491.

[14] A. Montresor, M. Jelasity, and O. Babaoglu, "Robust aggregation protocols for large-scale overlay networks," in *International Conference on Dependable Systems and Networks, 2004.* IEEE, 2004, pp. 19–28.

[15] M. Jelasity, A. Montresor, and O. Babaoglu, "Gossip-based aggregation in large dynamic networks," *ACM Transactions on Computer Systems (TOCS)*, vol. 23, no. 3, pp. 219–252, 2005.

[16] M. Jelasity, R. Guerraoui, A.-M. Kermarrec, and M. Van Steen, "The peer sampling service: Experimental evaluation of unstructured gossip-based implementations," in *ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing.* Springer, 2004, pp. 79–98.

[17] S. Voulgaris, D. Gavidia, and M. Van Steen, "Cyclon: Inexpensive membership management for unstructured p2p overlays," *Journal of Network and systems Management*, vol. 13, no. 2, pp. 197–217, 2005.

[18] N. Nova, F. Girardin, and P. Dillenbourg, "Etude empirique de l'utilisation de la géolocalisation en collaboration mobile," in *Proceedings of the 17th Conference on l'Interaction Homme-Machine.* ACM, 2005, pp. 207–210.

[19] A. G. Dimakis, A. D. Sarwate, and M. J. Wainwright, "Geographic gossip: Efficient aggregation for sensor networks," in *Proceedings of the 5th international conference on Information processing in sensor networks.* ACM, 2006, pp. 69–76.

[20] A. Fiorese, P. Simões, and F. Boavida, "Peer selection in p2p service overlays using geographical location criteria," in *International Conference on Computational Science and Its Applications.* Springer, 2012, pp. 234–248.

[21] M. Picone, M. Amoretti, and F. Zanichelli, "Geokad: A p2p distributed localization protocol," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on.* IEEE, 2010, pp. 800–803.

[22] S. Voulgaris and M. Van Steen, "Vicinity: A pinch of randomness brings out the structure," in *ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing.* Springer, 2013, pp. 21–40.

[23] V. N. Soares, J. J. Rodrigues, and F. Farahmand, "Geospray: A geographic routing protocol for vehicular delay-tolerant networks," *Information Fusion*, vol. 15, pp. 102–113, 2014.

[24] A. J. Ganesh, A.-M. Kermarrec, and L. Massoulié, "Peer-to-peer membership management for gossip-based protocols," *IEEE transactions on computers*, vol. 52, no. 2, pp. 139–149, 2003.

[25] M. Jelasity, S. Voulgaris, R. Guerraoui, A.-M. Kermarrec, and M. Van Steen, "Gossip-based peer sampling," *ACM Transactions on Computer Systems (TOCS)*, vol. 25, no. 3, p. 8, 2007.

[26] J. Leitao, J. Pereira, and L. Rodrigues, "Hyparview: A membership protocol for reliable gossip-based broadcast," in *37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07).* IEEE, 2007, pp. 419–429.

[27] E. Bortnikov, M. Gurevich, I. Keidar, G. Kliot, and A. Shraer, "Brahms: Byzantine resilient random membership sampling," *Computer Networks*, vol. 53, no. 13, pp. 2340–2359, 2009.

[28] A. Stavrou, D. Rubenstein, and S. Sahu, "A lightweight, robust p2p system to handle flash crowds," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 1, pp. 6–17, 2004.

[29] B. Nédelec, J. Tanke, D. Frey, P. Molli, and A. Mostefaoui, "Spray: an adaptive random peer sampling protocol," Ph.D. dissertation, LINA-University of Nantes; INRIA Rennes-Bretagne Atlantique, 2015.

[30] A. Alromih and H. Kurdi, "An energy-efficient gossiping protocol for wireless sensor networks using chebyshev distance," *Procedia Computer Science*, vol. 151, pp. 1066–1071, 2019.

[31] Wikipedia contributors, "Chebyshev distance — Wikipedia, the free encyclopedia," https://en.wikipedia.org/w/index.php?title=Chebyshev_distance&oldid=932176238, 2019, [Online; accessed 26-February-2020].

[32] S. Dahal *et al.*, "Effect of different distance measures in result of cluster analysis," 2015.

[33] D. Frey, M. Goessens, and A.-M. Kermarrec, "Behave: Behavioral cache for web content," in *IFIP International Conference on Distributed Applications and Interoperable Systems*. Springer, 2014, pp. 89–103.

[34] P. Folz, H. Skaf-Molli, and P. Molli, "Cyclades: a decentralized cache for linked data fragments," in *ESWC: Extended Semantic Web Conference*, 2016.

[35] S. Voulgaris and M. Van Steen, "Epidemic-style management of semantic overlays for content-based searching," in *European Conference on Parallel Processing*. Springer, 2005, pp. 1143–1152.

[36] M. Jelasity, A. Montresor, and O. Babaoglu, "T-man: Gossip-based fast overlay topology construction," *Computer networks*, vol. 53, no. 13, pp. 2321–2339, 2009.

[37] M. Mordacchini, R. Baraglia, P. Dazzi, and L. Ricci, "A p2p recommender system based on gossip overlays (prego)," in *2010 10th IEEE International Conference on Computer and Information Technology*. IEEE, 2010, pp. 83–90.

[38] A. Boutet, D. Frey, R. Guerraoui, A.-M. Kermarrec, and R. Patra, "Hyrec: Leveraging browsers for scalable recommenders," in *Proceedings of the 15th International Middleware Conference*. ACM, 2014, pp. 85–96.

[39] R. Carvajal-Gómez, D. Frey, M. Simonin, and A.-M. Kermarrec, "Webgc gossiping on browsers without a server," in *International Conference on Web Information Systems Engineering*. Springer, 2015, pp. 332–336.

[40] B. Nédelec, P. Molli, and A. Mostefaoui, "Crate: Writing stories together with our browsers," in *Proceedings of the 25th International Conference Companion on World Wide Web*. International World Wide Web Conferences Steering Committee, 2016, pp. 231–234.

[41] A. B. Pilet, D. Frey, and F. Taiani, "Robust privacy-preserving gossip averaging," in *International Symposium on Stabilizing, Safety, and Security of Distributed Systems*. Springer, 2019, pp. 38–52.

[42] C. S. Meiklejohn, H. Miller, and P. Alvaro, "{PARTISAN}: Scaling the distributed actor runtime," in *2019 USENIX Annual Technical Conference (USENIX 19)*, 2019, pp. 63–76.

[43] Wikipedia contributors, "Actor model — Wikipedia, the free encyclopedia," https://en.wikipedia.org/w/index.php?title=Actor_model&oldid=939106706, 2020, [Online; accessed 26-February-2020].

[44] J. Leitao, J. Pereira, and L. Rodrigues, "Epidemic broadcast trees," in *2007 26th IEEE International Symposium on Reliable Distributed Systems (SRDS 2007)*. IEEE, 2007, pp. 301–310.

[45] E. Leonardi, M. Mellia, C. Kiraly, R. L. Cigno, S. Niccolini, and J. Seedorf, "Network friendly p2p streaming: The napa-wine architecture," 2020.

[46] A. Montresor and M. Jelasity, "Peersim: A scalable p2p simulator," in *2009 IEEE Ninth International Conference on Peer-to-Peer Computing*. IEEE, 2009, pp. 99–100.

[47] L. Liu and N. Antonopoulos, "From client-server to p2p networking," in *Handbook of Peer-to-Peer Networking*. Springer, 2010, pp. 71–89.

[48] W. W. W. Consortium *et al.*, "Rdf 1.1 concepts and abstract syntax," 2014.

[49] A. Matono, T. Amagasa, M. Yoshikawa, and S. Uemura, "An indexing scheme for rdf and rdf schema based on suffix arrays." in *SWDB*, 2003, pp. 151–168.

[50] G. Karvounarakis, S. Alexaki, V. Christophides, D. Plexousakis, and M. Scholl, "Rql: a declarative query language for rdf," in *Proceedings of the 11th international conference on World Wide Web*. ACM, 2002, pp. 592–603.

[51] B. McBride, "Jena: Implementing the rdf model and syntax specification," in *Proceedings of the Second International Conference on Semantic Web-Volume 40*. CEUR-WS. org, 2001, pp. 23–28.

[52] J. Broekstra, A. Kampman, and F. Van Harmelen, "Sesame: A generic architecture for storing and querying rdf and rdf schema," in *International semantic web conference*. Springer, 2002, pp. 54–68.

[53] O. Erling and I. Mikhailov, "Virtuoso: Rdf support in a native rdbms," in *Semantic Web Information Management*. Springer, 2010, pp. 501–519.

[54] J. J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, and K. Wilkinson, "Jena: implementing the semantic web recommendations," in *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*. ACM, 2004, pp. 74–83.

[55] K. Wilkinson, C. Sayers, H. Kuno, and D. Reynolds, "Efficient rdf storage and retrieval in jena2," in *Proceedings of the First International Conference on Semantic Web and Databases*. Citeseer, 2003, pp. 120–139.

[56] E. Spaho, L. Barolli, and F. Xhafa, "Data replication strategies in p2p systems: A survey," in *2014 17th International Conference on Network-Based Information Systems*. IEEE, 2014, pp. 302–309.

[57] V. Martins, "Data replication in p2p systems," Ph.D. dissertation, Université de Nantes, 2007.

[58] J. Gray, P. Helland, P. O'Neil, and D. Shasha, "The dangers of replication and a solution," *ACM SIGMOD Record*, vol. 25, no. 2, pp. 173–182, 1996.

[59] B. Kemme and G. Alonso, "A new approach to developing and implementing eager database replication protocols," *ACM Transactions on Database Systems (TODS)*, vol. 25, no. 3, pp. 333–379, 2000.

[60] P. A. Bernstein, V. Hadzilacos, and N. Goodman, "Concurrency control and recovery in database systems," 1987.

[61] Y. Saito and M. Shapiro, "Optimistic replication," *ACM Computing Surveys (CSUR)*, vol. 37, no. 1, pp. 42–81, 2005.

[62] E. Pacitti, P. Minet, and E. Simon, "Fast algorithms for maintaining replica consistency in lazy master replicated databases," Ph.D. dissertation, INRIA, 1999.

[63] M. Shapiro, K. Bhargavan, and N. Krishna, "A constraint-based formalism for consistency in replicated systems," in *International Conference On Principles Of Distributed Systems*. Springer, 2004, pp. 331–345.

[64] D. B. Terry, M. M. Theimer, K. Petersen, A. J. Demers, M. J. Spreitzer, and C. H. Hauser, "Managing update conflicts in bayou, a weakly connected replicated storage system," in *SOSP*, vol. 95, 1995, pp. 172–182.

[65] G. Tummarello, C. Morbidoni, J. Petersson, P. Puliti, and F. Piazza, "Rdfgrowth, a p2p annotation exchange algorithm for scalable semantic web applications." *P2PKM*, vol. 108, 2004.

[66] C. A. Ellis and C. Sun, "Operational transformation in real-time group editors: issues, algorithms, and achievements," in *Proceedings of the 1998 ACM conference on Computer supported cooperative work*. Citeseer, 1998, pp. 59–68.

[67] C. Sun, X. Jia, Y. Zhang, Y. Yang, and D. Chen, "Achieving convergence, causality preservation, and intention preservation in real-time cooperative editing systems," *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 5, no. 1, pp. 63–108, 1998.

[68] C. A. Ellis and S. J. Gibbs, "Concurrency control in groupware systems," in *Acm Sigmod Record*, vol. 18, no. 2. ACM, 1989, pp. 399–407.

[69] H. Skaf-Molli, C. Rahhal, and P. Molli, "Peer-to-peer semantic wikis," in *International Conference on Database and Expert Systems Applications*. Springer, 2009, pp. 196–213.

[70] S. Weiss, P. Urso, and P. Molli, "Logoot-undo: Distributed collaborative editing system on p2p networks," *IEEE transactions on parallel and distributed systems*, vol. 21, no. 8, pp. 1162–1174, 2010.

[71] E. Spaho, A. Barolli, F. Xhafa, and L. Barolli, "P2p data replication: Techniques and applications," in *Modeling and Processing for Next-Generation Big-Data Technologies*. Springer, 2015, pp. 145–166.

[72] L. D. Ibáñez, H. Skaf-Molli, P. Molli, and O. Corby, "Live linked data: synchronising semantic stores with commutative replicated data types." *IJMSO*, vol. 8, no. 2, pp. 119–133, 2013.

[73] M. Shapiro, N. Preguiça, C. Baquero, and M. Zawirski, "Conflict-free replicated data types," in *Symposium on Self-Stabilizing Systems*. Springer, 2011, pp. 386–400.

[74] ——, "A comprehensive study of convergent and commutative replicated data types," Ph.D. dissertation, Inria–Centre Paris-Rocquencourt; INRIA, 2011.

[75] A. Reggiori, D.-W. van Gulik, and Z. Bjelogrlic, "Indexing and retrieving semantic web resources: the rdfstore model," in *SWAD-Europe Workshop on Semantic Web Storage and Retrieval*. Citeseer, 2003, pp. 13–14.

[76] A. Rowstron and P. Druschel, "Storage management and caching in past, a large-scale, persistent peer-to-peer storage utility," in *ACM SIGOPS Operating Systems Review*, vol. 35, no. 5. ACM, 2001, pp. 188–201.

[77] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica, "Wide-area cooperative storage with cfs," in *ACM SIGOPS Operating Systems Review*, vol. 35, no. 5. ACM, 2001, pp. 202–215.

[78] W. Nejdl, M. Wolpers, W. Siberski, C. Schmitz, M. Schlosser, I. Brunkhorst, and A. Löser, "Super-peer-based routing and clustering strategies for rdf-based peer-to-peer networks," in *Proceedings of the 12th international conference on World Wide Web*. ACM, 2003, pp. 536–543.

[79] M. Cai and M. Frank, "Rdfpeers: a scalable distributed rdf repository based on a structured peer-to-peer network," in *Proceedings of the 13th international conference on World Wide Web*. ACM, 2004, pp. 650–657.

[80] S. Ktari, M. Zoubert, A. Hecker, and H. Labiod, "Performance evaluation of replication strategies in dhts under churn," in *Proceedings of the 6th international conference on Mobile and ubiquitous multimedia*. ACM, 2007, pp. 90–97.

[81] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: a scalable peer-to-peer lookup protocol for internet applications," *IEEE/ACM Transactions on Networking (TON)*, vol. 11, no. 1, pp. 17–32, 2003.

[82] B. Y. Zhao, J. Kubiatowicz, A. D. Joseph *et al.*, "Tapestry: An infrastructure for fault-tolerant wide-area location and routing," 2001.

[83] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong, "Freenet: A distributed anonymous information storage and retrieval system," in *Designing privacy enhancing technologies*. Springer, 2001, pp. 46–66.

[84] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *IFIP/ACM International Confer-*

ence on Distributed Systems Platforms and Open Distributed Processing. Springer, 2001, pp. 329–350.

[85] M. Cai, M. Frank, J. Chen, and P. Szekely, "Maan: A multi-attribute addressable network for grid information services," *Journal of Grid Computing*, vol. 2, no. 1, pp. 3–14, 2004.

[86] K. Aberer, P. Cudré-Mauroux, M. Hauswirth, and T. Van Pelt, "Gridvine: Building internet-scale semantic overlay networks," in *International semantic web conference.* Springer, 2004, pp. 107–121.

[87] K. Aberer, P. Cudré-Mauroux, A. Datta, Z. Despotovic, M. Hauswirth, M. Punceva, and R. Schmidt, "P-grid: a self-organizing structured p2p system," *Sigmod Record*, vol. 32, no. ARTICLE, pp. 29–33, 2003.

[88] T. Risse, P. Knezevic, C. Meghini, R. Hecht, and F. Basile, "The bricks infrastructure-an overview," in *8th International Conference EVA.* Citeseer, 2005.

[89] E. Della Valle, A. Turati, and A. Ghioni, "Page: A distributed infrastructure for fostering rdf-based interoperability," in *IFIP International Conference on Distributed Applications and Interoperable Systems.* Springer, 2006, pp. 347–353.

[90] S. Rhea, D. Geels, T. Roscoe, J. Kubiatowicz *et al.*, "Handling churn in a dht," in *Proceedings of the USENIX Annual Technical Conference*, vol. 6. Boston, MA, USA, 2004, pp. 127–140.

[91] M. Ehrig, P. Haase, R. Siebes, S. Staab, H. Stuckenschmidt, R. Studer, and C. Tempich, "The swap data and metadata model for semantics-based peer-to-peer systems," in *German Conference on Multiagent System Technologies.* Springer, 2003, pp. 144–155.

[92] A. Crespo and H. Garcia-Molina, "Semantic overlay networks for p2p systems," in *International Workshop on Agents and P2P Computing.* Springer, 2004, pp. 1–13.

[93] J. X. Parreira, S. Michel, and G. Weikum, "p2pdating: Real life inspired semantic overlay networks for web search," *Information Processing & Management*, vol. 43, no. 3, pp. 643–664, 2007.

[94] R. Baeza-Yates, B. Ribeiro-Neto *et al.*, *Modern information retrieval*. ACM press New York, 1999, vol. 463.

[95] E. RICH and K. KNIGHT, "Artificial intelligence. 1991. ed."

[96] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[97] Z. G. Ives, A. Y. Halevy, P. Mork, and I. Tatarinov, "Piazza: mediation and integration infrastructure for semantic web data," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 1, no. 2, pp. 155–175, 2004.

[98] A. Löser, F. Naumann, W. Siberski, W. Nejdl, and U. Thaden, "Semantic overlay clusters within super-peer networks," in *International Workshop on Databases, Information Systems, and Peer-to-Peer Computing*. Springer, 2003, pp. 33–47.

[99] S. Voulgaris, A.-M. Kermarrec, and L. Massoulié, "Exploiting semantic proximity in peer-to-peer content searching," in *Proceedings. 10th IEEE International Workshop on Future Trends of Distributed Computing Systems, 2004. FTDCS 2004.* IEEE, 2004, pp. 238–243.

[100] D. R. Himali, S. B. Navathe, and S. K. Prasad, "Sas: Semantics aware search in p2p networks," in *2013 IEEE 33rd International Conference on Distributed Computing Systems Workshops*. IEEE, 2013, pp. 178–183.

[101] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The google file system," 2003.

[102] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[103] K. Shvachko, H. Kuang, S. Radia, R. Chansler *et al.*, "The hadoop distributed file system." in *MSST*, vol. 10, 2010, pp. 1–10.

[104] A. D. JoSEP, R. KAtz, A. KonWinSKi, L. Gunho, D. PAttERSon, and A. RABKin, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, 2010.

[105] T. H. Luan, L. Gao, Z. Li, Y. Xiang, G. Wei, and L. Sun, "Fog computing: Focusing on mobile users at the edge," *arXiv preprint arXiv:1502.01815*, 2015.

[106] I. Stojmenovic, S. Wen, X. Huang, and H. Luan, "An overview of fog computing and its security issues," *Concurrency and Computation: Practice and Experience*, vol. 28, no. 10, pp. 2991–3005, 2016.

[107] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *2012 Proceedings IEEE Infocom*. IEEE, 2012, pp. 945–953.

[108] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. ACM, 2012, pp. 13–16.

[109] S. Sarkar, S. Chatterjee, and S. Misra, "Assessment of the suitability of fog computing in the context of internet of things," *IEEE Transactions on Cloud Computing*, vol. 6, no. 1, pp. 46–59, 2015.

[110] R. Mahmud, R. Kotagiri, and R. Buyya, "Fog computing: A taxonomy, survey and future directions," in *Internet of everything*. Springer, 2018, pp. 103–130.

[111] B. Confais, "Conception d'un système de partage de données adapté à un environnement de fog computing," Ph.D. dissertation, Université de Nantes, 2018.

[112] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," *Wireless communications and mobile computing*, vol. 13, no. 18, pp. 1587–1611, 2013.

[113] M. Patel *et al.*, "Mobile-edge computing-introductory technical white paper, etsi mec white paper," V1 18-09-14, 36 pages, Tech. Rep., 2014.

[114] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE pervasive Computing*, no. 4, pp. 14–23, 2009.

[115] R. C. Computing, "Openstack cloud software," 2012.

[116] J. Zhu, D. S. Chan, M. S. Prabhu, P. Natarajan, H. Hu, and F. Bonomi, "Improving web sites performance using edge servers in fog computing architecture," in *2013 IEEE Seventh International Symposium on Service-Oriented System Engineering*. IEEE, 2013, pp. 320–323.

[117] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwälder, and B. Koldehofe, "Mobile fog: A programming model for large-scale applications on the internet of things," in *Proceedings of the second ACM SIGCOMM workshop on Mobile cloud computing*. ACM, 2013, pp. 15–20.

[118] M. Wylot and P. Cudré-Mauroux, "Diplocloud: Efficient and scalable management of rdf data in the cloud," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 3, pp. 659–674, 2016.

[119] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, "Bigtable: A distributed storage system for structured data," *ACM Transactions on Computer Systems (TOCS)*, vol. 26, no. 2, p. 4, 2008.

[120] M. A. Al Faruque and K. Vatanparvar, "Energy management-as-a-service over fog computing platform," *IEEE internet of things journal*, vol. 3, no. 2, pp. 161–169, 2016.

[121] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin, and S. Chen, "Vehicular fog computing: A viewpoint of vehicles as the infrastructures," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 6, pp. 3860–3873, 2016.

[122] H. Rahman and M. I. Hussain, "Fog-based semantic model for supporting interoperability in iot," *IET Communications*, vol. 13, no. 11, pp. 1651–1661, 2019.

[123] F. Mehmood, S. Ahmad, and D. Kim, "Design and implementation of an interworking iot platform and marketplace in cloud of things," *Sustainability*, vol. 11, no. 21, p. 5952, 2019.

[124] M. Farnbauer-Schmidt, J. Lindner, C. Kaffenberger, and J. Albrecht, "Combining the concepts of semantic data integration and edge computing," *INFORMATIK 2019: 50 Jahre Gesellschaft für Informatik–Informatik für Gesellschaft*, 2019.

[125] M. Völkel, M. Krötzsch, D. Vrandecic, H. Haller, and R. Studer, "Semantic wikipedia," in *Proceedings of the 15th international conference on World Wide Web*. ACM, 2006, pp. 585–594.

[126] S. Weiss, P. Urso, and P. Molli, "Wooki: a p2p wiki-based collaborative writing tool," in *International Conference on Web Information Systems Engineering*. Springer, 2007, pp. 503–512.

[127] P. T. Eugster, R. Guerraoui, S. B. Handurukande, P. Kouznetsov, and A.-M. Kermarrec, "Lightweight probabilistic broadcast," *ACM Transactions on Computer Systems (TOCS)*, vol. 21, no. 4, pp. 341–374, 2003.

[128] A. Demers, D. Greene, C. Houser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry, "Epidemic algorithms for replicated database maintenance," *ACM SIGOPS Operating Systems Review*, vol. 22, no. 1, pp. 8–32, 1988.

[129] A. Grall, H. Skaf-Molli, and P. Molli, "Sparql query execution in networks of web browsers," in *THE 17TH INTERNATIONAL SEMANTIC WEB CONFERENCE, WORKSHOP ON DECENTRALIZING THE SEMANTIC WEB*, 2018.

[130] Hong Kong Protests Propel FireChat Phone-to-Phone App, "Hong kong protests propel firechat phone-to-phone app," https://www.nytimes.com/2014/10/06/technology/hong-kong-protests-propel-a-phone-to-phone-app-.html, 2014, [Online; accessed 22-May-2020].

[131] FireChat – the messaging app, "Firechat – the messaging app," https://www.theguardian.com/world/2014/sep/29/firechat-messaging-app-powering-hong-kong-protests, 2014, [Online; accessed 22-May-2020].

[132] A. V. Sambra, E. Mansour, S. Hawke, M. Zereba, N. Greco, A. Ghanem, D. Zagidulin, A. Aboulnaga, and T. Berners-Lee, "Solid: A platform for decentralized social applications based on linked data," *MIT CSAIL & Qatar Computing Research Institute, Tech. Rep.*, 2016.

[133] M. Charlaganov, P. Cudré-Mauroux, C. Dinu, C. Guéret, M. Grund, and T. Macicas, "The entity registry system: Implementing 5-star linked data without the web," *arXiv preprint arXiv:1308.3357*, 2013.

[134] A. G. Hernández and M. GARCıA, "A javascript rdf store and application library for linked data client applications," in *Devtracks of the, WWW2012, conference. Lyon, France.* Citeseer, 2012.

[135] H. T. Shen, Y. Shu, and B. Yu, "Efficient semantic-based content search in p2p network," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 7, pp. 813–826, 2004.