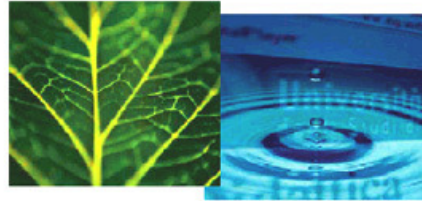


PhD Dissertation

---



**International Doctorate School in Information and  
Communication Technologies**

DISI - University of Trento

**DISTRIBUTED CONTACT AND IDENTITY MANAGEMENT**

Alethia Graciela Hume LLamosas

Advisor:

Prof. Fausto Giunchiglia

Università degli Studi di Trento

Co-Advisor:

Prof. Luca Cernuzzi

Universidad Católica "Nuestra Señora de la Asunción"

---

April 2014



# Abstract

*Contact management is a twofold problem involving a local and global level where the separation between them is rather fuzzy. Locally, users need to deal with contact management, which refers to a local need to store, organize, maintain up to date, and find information that will allow them contacting or reaching other people, organizations, etc. Globally, users deal with identity management that refers to peers having multiple identities (i.e., profiles) and the need of staying in control of them. In other words, they should be able to manage what information is shared and with whom.*

*We believe many existing applications try to deal with this problem looking only at the data level and without analyzing the underlying complexity. Our approach focus on the complex social relations and interactions between users, identifying three main subproblem: (i) management of identity, (ii) search, and (iii) privacy.*

*The solution we propose concentrates on the models that are needed to address these problems. In particular, we propose a **Distributed Contact Management System (DCM System)** that:*

- *Models and represents the knowledge of peers about physical or abstract objects through the notion of entities that can be of different types (e.g., locations, people, events, facilities, organizations, etc.) and are described by a set of attributes.*
- *By representing contacts as entities, allows peers to locally organize their contacts taking into consideration the semantics of the contact's characteristics.*
- *By describing peers as entities allows them to manage their different identities in the network, by sharing different views of themselves (showing possibly different information) with different people.*

*The contributions of this thesis are, (i) the definition of a reference architecture that allows dealing with the diversity in relation with the partial view that peers have of the world, (ii) an approach to search entities based on identifiers, (iii) an approach to search entities based on descriptions, and (iv) the definition of the DCM system that instantiates the previously mentioned approaches and architecture to address concrete usage scenarios.*

**Keywords:** Contact Management, Identity Management, P2P, Entity Search



## Acknowledgements

First I would like to express my special appreciation and thanks to my co-advisor Professor Dr. Luca Cernuzzi, this would not have been possible without your encouragement. I would also like to thank my advisor Professor Dr. Fausto Giunchiglia, you have been a great mentor for me. I would like to thank you for your trust, for encouraging my research and for allowing me to grow as a research scientist. Your advice on both research as well as on my career have been priceless. I would especially like to thank my colleagues from the KnowDive group. All of you have been there to support me with good advice and interesting discussions. You have contributed significantly to my personal and professional growth.

A thought to all my friends here in Trento who supported me in writing, and pushed me to strive towards my goal. Thanks also to my friends from Paraguay who, from the distance, were always present and knew how to give me the extra strength that I needed. A special thanks to my family for all their love and encouragement. Words cannot express how grateful I am to my two pillars, my mother and my husband. To my mom, thank you for all of the sacrifices that you've made on my behalf, your prayer for me was what sustained me thus far. To my beloved husband Eduardo, whose faithful support during the final stage has been of key importance. You spent sleepless nights and you were always my support in the most difficult moments.

*Alethia Hume*  
University of Trento  
April 2014

---

*The work compiled in this thesis has been partially supported by the European Community's Seventh Framework Program (FP7/2007-2013) under grant agreement n. 600854 Smart Society: hybrid and diversity-aware collective adaptive systems: where people meet machines to build smarter societies <http://www.smart-society-project.eu/>*



# Contents

<b>I</b>	<b>General Notions</b>	<b>1</b>
<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	The context . . . . .	3
1.2	The problem . . . . .	4
1.3	The solution . . . . .	5
1.4	Structure of the thesis . . . . .	6
<b>2</b>	<b>The problem</b>	<b>9</b>
2.1	Problem setting scenarios . . . . .	9
2.2	Discussion . . . . .	11
<b>3</b>	<b>Ground knowledge</b>	<b>15</b>
3.1	Fundamental notions . . . . .	15
3.2	Knowledge schema . . . . .	17
3.3	Instantiation of knowledge . . . . .	18
3.3.1	Entity identifiers . . . . .	19
3.3.2	Entity instances . . . . .	20
3.4	Contacts classification . . . . .	21
3.4.1	Classification of subjects . . . . .	22
3.4.2	Classification of objects . . . . .	24
3.5	Summary . . . . .	25
<b>II</b>	<b>The Proposed Approaches and System</b>	<b>27</b>
<b>4</b>	<b>Reference architecture</b>	<b>29</b>
4.1	Requirements . . . . .	30
4.2	High-level design . . . . .	32
4.3	System logical view . . . . .	34

4.3.1	Contact management portal . . . . .	35
4.3.2	Contact management application . . . . .	37
4.3.3	Contact management network . . . . .	38
4.4	System dynamic view . . . . .	39
4.5	Summary . . . . .	44
<b>5</b>	<b>A name-based approach to search in the DCM system</b>	<b>45</b>
5.1	Motivating example . . . . .	47
5.2	A name-based overlay for linking directories . . . . .	48
5.2.1	Formalization of the model elements . . . . .	48
5.2.2	Building the name-based overlay . . . . .	50
5.3	Name matching . . . . .	52
5.4	Algorithms . . . . .	53
5.5	Summary . . . . .	56
<b>6</b>	<b>A description-based approach to search in the DCM system</b>	<b>57</b>
6.1	Motivating example . . . . .	59
6.2	A semantic overlay for linking directories . . . . .	61
6.3	Semantic link discovery . . . . .	65
6.4	Algorithms . . . . .	66
6.4.1	Identifying semantically relevant peers . . . . .	67
6.4.2	Searching inside a relevant peer . . . . .	69
6.4.3	Aggregation of search results . . . . .	70
6.5	Summary . . . . .	70
<b>7</b>	<b>The distributed contact management (DCM) system</b>	<b>73</b>
7.1	Presentation Cards . . . . .	74
7.2	DCM Users . . . . .	77
7.3	Usage scenarios . . . . .	78
7.3.1	Initialization Scenarios . . . . .	78
7.3.2	Sharing Scenarios . . . . .	85
7.3.3	Search Scenarios . . . . .	91
7.4	Summary . . . . .	97
<b>III</b>	<b>Evaluation</b>	<b>99</b>
<b>8</b>	<b>Experimental evaluation of name-based search</b>	<b>101</b>
8.1	Implementation . . . . .	101



8.2	Evaluation . . . . .	105
8.3	Summary . . . . .	107
<b>9</b>	<b>Experimental evaluation of description-based search</b>	<b>109</b>
9.1	Implementation . . . . .	109
9.2	Evaluation . . . . .	110
9.2.1	Data-set generation . . . . .	110
9.2.2	Evaluation of results . . . . .	112
9.3	Summary . . . . .	114
<b>IV</b>	<b>Conclusions</b>	<b>117</b>
<b>10</b>	<b>Related work</b>	<b>119</b>
10.1	Contact and Identity Management . . . . .	119
10.2	Distributed entity directory . . . . .	120
10.3	Semantic flooding . . . . .	122
<b>11</b>	<b>Conclusions and future work</b>	<b>129</b>
11.1	The context . . . . .	129
11.2	The contributions . . . . .	130
11.3	The evaluations . . . . .	133
11.4	Future work . . . . .	134
	<b>Bibliography</b>	<b>137</b>



# List of Tables

8.1	Implementations of Kademlia protocol. . . . .	102
8.2	Average query time . . . . .	106
10.1	Search Methods in P2P networks. . . . .	125



# List of Figures

3.1	An example of an <i>ET</i> to describe entities of type “Person” and two instances showing the different descriptions of the same entity from the points of view of peers P1 and P2 respectively. . . . .	22
3.2	An example of a classification of contacts as subjects. . . . .	23
3.3	An example of a classification of contacts as objects. . . . .	24
4.1	General system view . . . . .	32
4.2	Third party platforms . . . . .	33
4.3	Logical Architecture . . . . .	35
4.4	Sequence diagram of the initialization process of a new peer in the system. . . . .	40
4.5	Sequence diagram of a peer sharing its contact information. . . . .	41
4.6	Sequence diagram of a peer searching the system. . . . .	43
5.1	Example of Contact Lists Related by Identifiers . . . . .	47
6.1	DCM Network of User-Generated Classifications . . . . .	58
6.2	Example of Contacts with Similar Characteristics . . . . .	60
6.3	Classification . . . . .	62
6.4	A Semantic Overlay Network . . . . .	64
7.1	Example of Person Presentation Cards . . . . .	76
8.1	Query time of different networks . . . . .	106
9.1	Topics Popularity Distribution . . . . .	112
9.2	Evaluation Results: Random queries . . . . .	113
9.3	Evaluation Results: Popular vs. Unpopular Queries . . . . .	114



## Part I

# General Notions





# Chapter 1

## Introduction

### 1.1 The context

In the general context of this thesis, we see Internet as a network of peers (a P2P network) organizing their content in directories (e.g., contact lists, document directories and event directories, like calendars or agendas, etc.), which store information about a number of “things” that are of their interest. We refer to these things that exist in the real world as *entities*. They can be of different types (e.g., locations, people, events, facilities, organizations, etc.) and are described by a set of attributes. In particular, we focus the attention in contact information, hence directories of contacts or contact lists. Within this context, a contact is seen as a profile of an entity where the type of entity may be constrained to certain types (i.e., corresponding to entities that can be contacted).

Different profiles of an entity can show different aspects of such entity. When talking about directories of contact, this can be reflected in different contact information. For example, a professor at the University of Trento may have a profile as a professor including his university email address and university home page. Another profile of the same person may include his mobile phone number, home address, and personal email address. As we can see, the former shows a professional (or work) aspect while the latter shows a more social aspect of the same person.

These different profiles exists as a consequence of different information being shared with different people (e.g., work related information may be shared with students and other information may be shared with family members). In the network of peers that we are describing, this means that the different profiles will be stored in directories corre-

sponding to different peers. On the other hand, all his students will have the same information (i.e., the same profile as a professor), and this also means that the same profile can be stored in different peers' directory. Further, we see the root of the difference between peers storing one profile or the other in the ties or relations (e.g., student-professor, colleagues, family ties, etc.) between each peer and the entity being described.

This thesis looks at many situations in everyday activities requiring managing such profiles (i.e., contacts), including how they are created, shared, updated, searched and organized in contact lists (also called contact directories).

## 1.2 The problem

Within the context described in Section 1.1, the problem this thesis address is twofold, (i) management of contacts stored in peer's contact list, and (ii) management of identities or profiles shared by peers in the network.

- The **contact management** refers to a local dimension of peers that need to store, organize, maintain up to date, and find information that will allow them contacting or reaching other people, organizations, etc.
- The **identity management** refers to a global dimension of peers having multiple identities (i.e., profiles) and the need of staying in control of them. In other words, they should be able to manage what information is shared and with whom.

However, both part of the problem can not be separated because a profile shared by a peer needs to be managed (i.e., stored, organized) as a contact in the contact list of another peer. Also, the need of finding contact information about someone can be seen as a sharing request from the point of view of the owner of the information and so on. This evidence the inherently distributed nature of the problem.

On the other hand, we mentioned before that the contact list of different peers can be related, for example by describing the same contact or by having contacts with similar characteristics. A concrete example of the first case would be that different students working with the same Prof. G. Lombardi may have his contact in their contact lists. On the other hand, G. Lombardi is a professor in the area of Information Technology (specialized, in particular, in distributed systems), and most likely he knows and has the contact of many colleagues working in the same or related areas. This, in turn, shows that the relation between data from different contact directories (i.e., corresponding to different peers) can be of different nature and may be seen as a consequence of the relations between peers themselves.

Taking into consideration these different aspects we believe that there is a complexity associated to the **Distributed Contact and Identity Management** problem, which requires the definition of appropriate models addressing the underlying subproblems rather than looking only the data level. We identify the following subproblems:

1. *Management of identity.* On one hand, the identity of contacts has to be managed in order to avoid duplicate contacts in local directories of peers. On the other hand, the identity of peers has to be managed in order to allow them to stay in control of their contact information globally (i.e., what it is shared and with whom).
2. *Search.* In the context described above, peers may have different information needs. On one hand, the search problem refers to the need of taking into consideration the different nature of possible search queries (e.g., search a specific contact or search contacts with certain characteristics). On the other hand, it refers to the need of taking into consideration different scopes for the search (e.g., search locally, search among friends, search globally in the network, etc.).
3. *Privacy.* The problem of privacy refers to the need of taking into consideration the privacy concerns that may appear as a consequence of manipulating contacts which can include personal sensitive information. In this sense, peers should be able to define what information can be shared and with whom.

### 1.3 The solution

In this PhD Thesis we propose a **Distributed Contact Management System (DCM System)** with the following features:

1. It takes into account that different peers can describe physical or abstract object (called *entities*) from different points of view showing possibly different information about them.
2. It allows peers to locally organize their contacts in a meaningful manner, taking into consideration the semantics of the contact's characteristics.
3. It allows peers to manage their different identities in the network by defining and sharing different contact profiles of themselves with different people.
4. It takes into account that peers directories in the system are inherently connected by links of different nature and provides models that can formalize these links in order to exploit them through different types of services.

5. It allows peers to exploit links connecting different peers directories to search information about entities (possibly contacts) based on identifiers.
6. It also allows peers to exploit links connecting different peers directories to search contacts based on descriptions.
7. It takes into account the importance of privacy with regard to contact information and proposes a privacy-friendly design that can facilitate the adoption of privacy enhancing technologies.

In short, the contributions of the thesis are:

**A reference Architecture** that takes into consideration the different actors interacting with the system, identifies different system components and define how they interact with each other. This architecture allows dealing with the diversity in relation with the partial view that peers have of the world, by accommodating multiple representations (from the perspectives of different peers) for the same real world entity. This includes also the case of peers describing other peers from different perspectives as part of their contact directories.

**An approach to search based on Identifiers** that proposes a model to build the connecting links between local directories of peers based on different types of identifiers that are used to refer to the entities that are stored in those directories.

**An approach to search based on Descriptions** that proposes a model to build semantic links in order to connect local directories of peers that store information about entities with similar or related characteristics.

**The DCM System** that has two roles: (i) on one hand, it is a case study that evaluates the three above-mentioned contributions by integrating them into a concrete system that shows the added value of the three elements as a whole; and (ii) on the other hand, the definition of the system is in itself a contribution of this thesis, including the definition of models that are application dependent (i.e., custom) and concrete usage scenarios.

## 1.4 Structure of the thesis

The rest of the thesis is organized as follows:

- The **Chapter 2** introduces some situations exemplifying different problems that are related to management of contacts in personal devices and that people have to face

in everyday activities. It also identifies and discusses a set of subproblems that are motivated by these situations.

- The **Chapter 3** presents basic notions and models that are adopted in this thesis for the representation and organization of information.
- The **Chapter 4** presents the reference architecture proposed by this thesis for the Distributed Contact Management System (DCM System). This includes the identification of requirements for the architecture; the discussion of the high-level system design identifying also actors interacting with it; the introduction of the system logical view discussing the different system components; and the discussion of system component interactions as a consequence of main system functionalities.
- The **Chapter 5** presents an approach to build a distributed directory that can link data from peers directories based on their identifiers in order to provide search services analogous to those of telephone book white pages. This includes the definition of models for the directory and the algorithms for the search.
- The **Chapter 6** presents an approach to build a semantic overlay that can link data from peers directories based on their characteristics in order to provide search services analogous to those of telephone book yellow pages. This includes the definition of models for the semantic overlay as well as the algorithms for the search.
- The **Chapter 7** presents the DCM System, introducing more application specific models and an extensive description of different usage scenarios, which include the actions performed by the system in order to support them.
- The **Chapter 8** presents details of a preliminary evaluation of the approach proposed in Chapter 5.
- The **Chapter 9** presents details of a preliminary evaluation of the approach proposed in Chapter 6.
- The **Chapter 10** presents the related work in the different related areas.
- Finally, **Chapter 11** and **Chapter ??** present the conclusions and future work, respectively.



## Chapter 2

# The problem

The goal of this thesis is to define a distributed infrastructure for managing contacts, which enables peers to manage their knowledge about the contact information of other peers and to easily share their own contact information.

Although the contact management problem is essentially not new and it might look rather simple, we believe that it has not been solved in an efficient manner. Moreover, we consider that there is an underlying complexity that has been overlooked by many existing applications and it is related to the fact that managing contacts is at the core of managing/understanding social relations and social interactions. In order to deal with such complexity, the attention has to focus on appropriate models to address the underlying subproblems rather than looking only the data level.

In this chapter, first, we introduce some problem scenarios related to the management of contacts that depict some issues that users usually face. Second, we identify and discuss subproblems that are motivated by these scenarios.

### 2.1 Problem setting scenarios

Many situations in everyday activities require users of different types of devices (e.g., smart-phone, notebooks, PDAs) to deal with the management of their contacts. We present in this section the description of a set of scenarios showing the type of difficulties that users usually have in relation to the management of contact lists. The scenarios are the result of a creative thinking work, we present them in a narrative way and we use an informal language in order to illustrate realistic situations.

**Problem setting scenario 1.** *Andrea is a 40 years old clerk. Recently he moved from mobile to a smartphone. He wishes his contacts in Facebook, Twitter and Skype automatically in his agenda. He finds out that it is possible to import such contacts to his smart phone. The problem with this is that instead of having a single coherent contact list, he got an extended list with a lot of contacts that seem to be repeated. Instead he has to switch from a social network to another and manually copy his contacts, or he has to import and then check manually for duplicates in order to merge the information from different accounts.*

**Problem setting scenario 2.** *Liza is a PhD student of DISI, recently she moved from iPhone to Android and she bought a recent HTC. For some reason she ignore, she wasn't able to maintain phone numbers and email addresses in her SIM card, but she still have the email addresses of some people on Gmail and some of her contacts are also friends of her on Facebook. At the moment the only solution she has found is to spread on Facebook, Twitter and Skype a message telling friends about this problem and asking them back their phone numbers. She is upset by having to spread this message so widely but this is the easiest way that found to do it. Now she has to memorize again all the information that she is receiving in her contact list. She is also upset because she has to put the addresses, emails and phone numbers manually. Moreover, she recently has made new friends and she has no clue about how to rescue their mobile numbers.*

**Problem setting scenario 3.** *Giovanni was a Master student of DISI. At present he has a contract in a local company as programmer. He never changed phone company and always has had the same brand of smartphone (Samsung). Anyways, as years go by, his friends have been messing his contact data, some moved house, some changed mobile phone number, and some abandoned their email for a new one. Just few of them let him know about those switches, therefore he regularly find his contact data as old. He his irritated by this, because he often finds himself sending message to people that never receives them. He is also irritated by having so often to put his hands on his contact data in order to update them when he would have thousand of other things to do.*

**Problem setting scenario 4.** *Anna is a new PhD student in Trento; she just arrived from Austria last week. She is making new friends very quickly and soon got involved in the students activities of her students' residence, San Bartolommeo. Last Friday there was a party in the main hall of the residence. She chatted with a lot of people but in particular she met a guy called Carlo, that she discovered having a lot of her research interests. They promised to get in touch and he gave her his (printed) business card. At the party she didn't had time to memorize in her phone Carlo's contact information. She had to leave the party at 11pm. The next day, when she wishes to contact him to send him*



some references and start working together for a manuscript, she can't find his card. She doesn't know how to get his email or phone number. She saw Carlo speaking with other people she knows, but she does not know who of them (or if any of them) may have a way to reach him.

**Problem setting scenario 5.** *Carlo is a PhD student at the University of Trento. He is very social, he likes sports and he enjoy participating at students activities. Last Friday, he went to a party in the main hall of the students' residence, San Bartolommeo. At the party he first met Anna, a researcher, they agreed to discuss ideas for working together and he gave her his business card. Right before leaving the party, Carlo met also Peter and his friends. After some minutes chatting with them he learned that they also enjoy playing at soccer as much as he does. In fact, they play almost every week and invited Carlo to join them. Carlo wishes to give them his phone number or personal email so they can contact him for next week' match. He has his business card, which contains only his work email address and does not contains the information he wishes to share. He then decides to take note of the phone number of one of the guys with whom he agrees to get in touch during the week. Unfortunately, when Carlo tries to contact the guy, he realizes that he has the wrong number. He is upset because he was really looking forward to play soccer and now he has no idea of how to contact those guys.*

**Problem setting scenario 6.** *John is a 45 years old father. He started to use a smart-phone a couple of years ago. He is italian and lives with his family in Milan, where John and his wife recently found out that one of their daughter has a very rare skin disease. The doctor that diagnosed her in Milan was very clear with them, he explained that although hi recognized the disease, he was not an expert on it. They are now trying to find a doctor that can treat their daughter. They started by asking their current doctor for references to other doctors. They are also searching information in internet and posting messages on social networks (e.g., Facebook, Twitter) asking if anyone knows an specialist on skin conditions. His next step is to talk to his friends and ask them if they know any expert on the disease (or any dermatologist) but looking at his contact list, he does not know where to start, who of his contacts is more likely to have the information he needs. John wishes to have an easier way to search for the contact of a doctor that can help them. He is upset and worry about his daughter.*

## 2.2 Discussion

We model contacts as *entities* of different types (e.g., people, restaurants, hotels, universities and others) whose descriptions include different ways to reach them. The problem

setting scenarios presented above show that the management of contact information includes local and global dimensions.

- The *Local* dimension refers to managing the local contact list of peers. This includes enabling basic operations that allow peers to create, update, delete and search contacts. We have to take into consideration that the creation of new contacts can often be the result of importing information from different sources (e.g., Facebook, Gmail, Skype, etc.) and these sources may contain (possibly different) information about the same contact. An example of this situation is what happen to Andrea in problem setting scenario 1. In this sense, avoiding duplicate contacts in the local contact lists of peers is an *identity management problem*.

On the other hand we also consider the *search problem* as part of this dimension because it represents a need that is local (for the peer) regardless of the scope of the search (i.e., if the peer wants to search in its local contact list or search in whole network). We can distinguish between two types of search, which are analogous to the white and yellow pages from phone book directories<sup>1</sup>.

In the first case (i.e., white pages), the peer knows exactly what is the (one) target contact and the *search is based on a given identifier*. For example, in the problem setting scenario 4, Anna knows exactly who she wants to contact (i.e., Carlo), she knows his name and if a description of him is presented to her she will be able to recognize him. This is also similar to the problem of Liza in problem setting scenario 2. In the second case (i.e., yellow pages), the target of the search is not one particular contact but a set of contacts fulfilling a given set of characteristics. We say that *search is based on a given description* in this case. An example of this is the problem of John in problem setting scenario 6.

- The *Global* dimension refers to the awareness of an intrinsic connection between peers that may be storing information about the same contact. Moreover, people often need to share different contact information (e.g., in different contexts or with different people) showing possibly different profiles of themselves. An example of this is the situation of Carlo in problem setting scenario 5, he wishes to show different profiles to Anna and Peter. In the context of contact management, we need to allow peers (such as Carlo) to stay in control of their own contact information, deciding what information is shared and with whom. On one hand, this is related with the peers' right to privacy and, to data protection as way to guarantee it. On the other hand, the models to manage contacts identity at a global level has to enable the

---

<sup>1</sup>[http://en.wikipedia.org/wiki/Phone\\_book](http://en.wikipedia.org/wiki/Phone_book)

description of the same contact (i.e., of Carlo) from diverse points of view (i.e., from Anna and Peter).

As this discussion shows, there are many subproblems that can be derived from the analysis of different aspects of the Contact Management problem. In particular, we distinguish the following:

1. **Identity management** from a local and a global perspective. On one hand, the identity of contacts has to be managed in order to avoid duplicate contacts in local directories of peers. On the other hand, the identity of peers has to be managed in order to allow them to stay in control of their contact information globally (i.e., what it is shared and with whom).
2. **Search** based on identifiers and descriptions (i.e., similar to white pages and yellow pages). When dealing with management of contacts and in particular with search, peers have different information needs in different scenarios. On one hand, the search problem refers to the need of taking into consideration the different nature of possible search queries (e.g., search a specific contact or search contacts with certain characteristics). On the other hand, it refers to the need of taking into consideration different scopes for the search (e.g., search locally, search among friends, search globally in the network, etc.).
3. **Privacy** and data protection. The problem of privacy refers to the need of taking into consideration the privacy concerns that may appear as a consequence of manipulating contacts which can include personal sensitive information. In this sense, peers should be able to define what information can be shared and with whom.



## Chapter 3

# Ground knowledge

This chapter aims at introducing general notions in order to set the basis (i.e., ground knowledge) for the discussions and approaches that are presented throughout this thesis. Therefore, we define here basic elements that serve as a building blocks for the Distributed Contact Management System (DCM system).

We start by presenting the notions of contact, peer and entity, which are fundamental for the definition of our system. Then, we present the semantic schema we adopt in this thesis for the representation of data. Next, we show how this semantic schema is instantiated into actual data describing peer's knowledge. Finally, we show how contacts can be classified in hierarchical structures based on characteristics of different nature.

**Acknowledgement.** Some of the notions and definitions presented in this chapter are the result of previous work developed by members of the Knowdive<sup>1</sup> group. In particular, some of the notions we adopt are based in the definitions introduced in [Pane, 2012] and D1.1 (deliverable 1.1) from SmartSociety<sup>2</sup> project.

### 3.1 Fundamental notions

Three fundamental notions for our system are:

**Contacts.** When thinking in contact information, people may often think in name, address, phone numbers and maybe email address. However, with the extensive use of personal devices (e.g., notebooks, smart- phones, PDAs) in combination with many

---

<sup>1</sup><http://disi.unitn.it/~knowdive/>

<sup>2</sup><http://www.smart-society-project.eu/>

web-based technologies and services (e.g., social networks, instant messaging applications and others), the notion of contact information becomes more complex. A contact in our system refers to an entity from the real world that is somehow “contactable” (i.e., is capable of getting involved in communication activities), could be a person, a facility, an organization, etc. Then, the representation of a contact is seen as a profile describing characteristics that are known of such entity. This may include information about different ways to reach the contact (e.g., phone number, mobile number, skype user, and others), as well as other general characteristics (e.g., name, age, place of birth in the case of people) that can help distinguishing contacts from one another.

**Peers.** We discussed in Chapter 2 the contact management at a large scale as a problem that is inherently distributed. Therefore, when thinking in users of the DCM system, we see a network of interacting peers. A peer refers to a user of the system that maintains a contact list, is capable of acting, making decisions and participating in communications activities (i.e., is contactable).

**Entity.** We use the notion of entity to refer to a “thing” that exists in the real world. Entities are defined as abstract or physical objects, can be of different types (e.g., person, location, event, etc.) and are described by attributes (e.g., name, birth date, latitude-longitude, size, duration, etc.), which can be different for different types of entities. Within the system, we formalize the notion of entities and we use it to represent structured information about contacts and peers. In other words, we adopt an entity-centric approach that uses entities as the basic element of knowledge.

We say then that the peer’s contact list contains its knowledge about known entities, including a description of itself as an object (i.e. a person) from the real world. In order to understand the semantics of entities within the context of distributed contact management, we need to take into consideration that:

- Peers represent their own “versions” of entities. As a consequence, different peers may describe different points of view, showing possibly different aspects of the same entity.
- Peers have a partial view of the world. This is a consequence of the fact that different peers may know different subset of entities that exist in the world.
- Peers can describe entities that refer to other peers. This happens for example when the contact of a peer, in turn, is also a user of the system.

To reason about contacts and peers, we need to represent entities in such a way that will allow peers to understand each other (i.e., interoperability). For example, a peer should be able to find all the different versions of an entity describing the same contact that are available even if different peers generated them. Another example is the case of a peer sharing its (own) contact with another peer; they need certain level of agreement regarding the structure they use to represent the data.

Achieving interoperability between peers at the data level requires an agreement on the formal models that they will follow for the representation of data (i.e., their knowledge). In other words, in order to make the knowledge comparable, the same format should be followed by different peers in the system. The entity-centric approach adopted in this thesis distinguishes between a *schema level* that defines this “format” and a *knowledge level* that defines how to instantiate the schema into actual knowledge. In the following sections, we formally present the elements that are part of these two levels.

### 3.2 Knowledge schema

The Schema.org<sup>3</sup> initiative defines schemas as “A set of types, each associated with a set of properties and where the types are arranged in a hierarchy”. We adopt an approach that is align with this idea and allows the definition of templates for each type of entity used in the system. These templates serve to establish restrictions on the set of attributes that can be used to describe a given type of entity. The meaning is further specified by mapping single elements from the schema (i.e., types of entities, the names of attributes and their values) to concepts from a knowledge base.

A *concept* is defined as “An abstract or general idea inferred or derived from specific instances” in WordNet<sup>4</sup>; and as “An idea, something that is conceived in the human mind” in Wikipedia<sup>5</sup>. In general, in the area of knowledge representation, concepts are used to formalize and represent the meaning of words in a language independent manner.

In what follows, we formalize our notion of entity type and other related notions (i.e., basic schema elements) in a recursive manner.

An **Entity Type (ET)** is formalized as the tuple

$$ET = \langle C, \{AD\} \rangle$$

where,

- $C$  represents a concept associated to the name of the entity type and which defines the class of entities that are describe by it;

---

<sup>3</sup><http://schema.org/>

<sup>4</sup><http://wordnet.princeton.edu>

<sup>5</sup><http://en.wikipedia.org/wiki/Concept>

- $\{AD\}$  is a non-empty set of attribute definitions denoting the type of attributes that can be used to describe an entity of the corresponding  $ET$ . We assume that a distinction can be made between those mandatory and optional attributes but for the sake of simplicity we avoid going into more details about this in the models.

The notion of **Attribute Definition (AD)** is aimed to explicitly state constraints regarding how can be describe certain property of an entity. It is formally defined as the tuple

$$AD = \langle C, AT \rangle$$

where,

- $C$  is a concept associated to the name of the attribute, which provides a meaning for the property that an instance of the corresponding AD is describing;
- $AT$  is a data type that establishes constraints on the values for the definition of the attribute. We can distinguish among those that are natively supported by the system (e.g., integer, string, float, date, etc.), complex concepts from a knowledge base, and the application defined  $ETs$ .

### 3.3 Instantiation of knowledge

In order to actually represent knowledge about *entities* from the real world, a schema defined following the models presented in Section 3.2 has to be instantiated. An entity type  $ET$  is instantiated to represent a particular description of a *real world entity*, which we call *Digital Entity (DE)*. In turn, an attribute definition  $AD$  is instantiated as part of a  $DE$  to represent a property (or characteristic), called *Attribute (A)* of the entity.

The representation of entities may look rather similar to each other when they describe entities having similar characteristics (e.g., people with similar interests, born in the same city maybe also in the same month, etc.). On the other hand, in the DCM system the peers describe entities from their own point of view and therefore it may also happen that two descriptions (from different peers) that appear to be different actually refer to the same entity. We distinguish an entity from others in the system by mean of *identifiers*. These identifiers are defined as labels assigned to entities and used as a reference to them (used to “call” them) <sup>6</sup>. Moreover, from our point of view, entities can be assigned with multiple identifiers that serve different purposes.

Let us first discuss the entity identifiers and then we will formally define the entity instances.

---

<sup>6</sup><http://en.wikipedia.org/wiki/Identifiers>



### 3.3.1 Entity identifiers

The identity of an entity encodes its uniqueness within certain context and allows to distinguish it from other entities. It is defined by characteristics of the entity that can be intrinsic (i.e. that belongs by nature) or extrinsic (i.e. acquired from the outside) [Do Van Thanh, 2007]. Identity is as a fundamental notion when reasoning about entities, it allows position entities (that can be individuals or objects from the real world) and understand their relations with other objects in the environment [Windley, 2005; Camp, 2004].

Following the notion of identity, *identifiers* are used in order to refer (i.e., identify) a person, an organization or any type of entity within a context. One entity can have, in fact, multiple identifiers that serve for different purpose or in different contexts. In the DCM system, we can distinguish between identifiers used by humans (called human-understandable identifiers) and identifiers used by computers (called machine-understandable identifiers).

People usually refer to an entity by a name, for example, when talking about the entity. In their minds this name is uniquely mapped to the description they have (their view) of such entity [Pane, 2012]. The context is implicit in the conversation in this case. An entity can be called by multiple names (e.g., the same person being identified by the names: Anne Smith, Anne Elizabeth Smith and Little Annie) and different entities can be referred by (called using) the same name (e.g., Anne Smith and Alice Smith can be also identified by the name A. Smith) as a consequence of being arbitrarily assigned. This does not change their importance as human-understandable identifiers but makes impossible for the machine to dereference names into the entity they represent. Therefore, we need machine-understandable identifiers to allow computers to refer to entities.

On the other hand, a machine-understandable identifier is one that can be uniquely solved by computers. Many standards were proposed in the WWW for digital identifiers, among them the most widely known are URIs<sup>7</sup>, URLs<sup>8</sup> and URNs<sup>9</sup> In the DCM system we need to distinguish the different descriptions of the same real world entity (i.e., from local directories of peers) while still maintaining the track about what entity from the real world the peer is describing (i.e., global identification). With the purpose of distinguishing between local and global identifiers, the work of Pane [2012] creates two new identifiers, called *SURL* and *SURI*.

A *SURL* is defined as a semantic URL that represents a particular description (in local directories) of a real world entity. A *SURL* is created in local directories for each

---

<sup>7</sup>RFC1630 - Universal Resource Identifiers

<sup>8</sup>RFC1738 - Uniform Resource Locators

<sup>9</sup>RFC1737 - Uniform Resource Name

entity being described in it, it is globally unique and can be dereferenced to obtain the full description of the entity. In other words, it encodes the location of a particular description of a real world entity.

A *SURI* is defined as a semantic URI that represents a real world entity without attaching it to a particular description. The same SURI is shared by different directories describing the same real world entity, it is also globally unique. A SURI cannot be directly used to retrieve an entity description, because it does not commit to one single description and it rather includes the different points of view from which an entity is described.

Differently from other approaches from the Semantic Web that combine URIs and URLs to identify entities in the Web (e.g., OKKAM, semanticweb.org<sup>10</sup>, www.w3.org<sup>11</sup>), the separation between local and global identifiers allow us to split the identification of a real world entity and its description(s). Further, other approaches implicitly impose a description for the real world entity when re-using the identifier, while we (by adopting the local/global identifiers) embrace diversity with regard to the point of views represented in different directories, enabling also the creation of a network of interconnected directories (we discuss in more details this in Chapter 5).

### 3.3.2 Entity instances

As we described before, knowledge in the system is represented through the instantiation of the models presented in Section 3.2.

A **Digitally Entity (DE)** instantiates an entity type ET and describe a real world entity from a particular point of view (i.e., the point of view of the directory's owner). This description represents known characteristics of the entity through a set of attributes. Then, the description is also attached with the different types of identifiers that are used to refer to it (i.e., human-understandable and machine-understandable identifiers). Formally, it is defined as the tuple

$$DE = \langle SURL, SURI, \{N\}, ET, \{A\} \rangle$$

where,

- *SURL* is unique identifier of this particular *DE*;
- *SURI* is a unique identifier of the real world entity that the corresponding *DE* is describing;
- $\{N\}$  is a set of strings representing names used by the corresponding description *DE* to identify a real world entity;

<sup>10</sup>[http://semanticweb.org/wiki/Uniform\\_Resource\\_Identifier](http://semanticweb.org/wiki/Uniform_Resource_Identifier)

<sup>11</sup><http://www.w3.org/TR/cooluris/#semweb>

- $ET$  is the entity type among those defined for the corresponding system;
- $\{A\}$  is a non-empty set of attributes describing the characteristics of the entity.

An **Attribute (A)** instantiates an attribute definition  $AD$  to represent a particular characteristic of the entity within a  $DE$ . Some attributes may have multiple values, its values may be mapped to a meaning in some knowledge base (i.e., semantic values) or can represent a relation to another entity when the value is a reference to another  $DE$  (i.e., relational attribute). Formally, it is defined as the tuple

$$A = \langle AD, \{V\} \rangle$$

where,

- $AD$  is an attribute definition among those defined for the corresponding system and denotes constraints on possible values for this attribute;
- $\{V\}$  is a set of attribute values of the type  $AT$  of the corresponding  $AD$ . Note that, if the corresponding  $AT$  is an  $ET$ , then  $A$  is called a relational attribute as it defined relations between two entities. For example, relations like friend-of, colleague-of, mother-son, etc., would be defined as relational attributes.

A simplified example of the schema and its instantiation is shown in Figure 3.1. It is important to note, (i) first, that the specific definition of entity types depends of the domain; (ii) second, that this example shows only one entity type (i.e., Person) but in our system we will need to describe other types of entities (e.g., locations, events, organizations, facilities, etc.) in order to represent general knowledge of peers about contacts and other related entities.

### 3.4 Contacts classification

Classifications has been used for a long time as a mechanism to organize different types of objects. Some well known examples of its extensive use and effectiveness are web directories, file systems, email directories, business catalogs, among others. These classifications are tree-like structure hierarchies used to organize objects of different types depending on their characteristics and the purpose of the classification in the target application [Giunchiglia et al., Winter 2006].

The hierarchical structure of classifications encodes subsumption relationships between the nodes in the hierarchy, which means that elements (or objects) that can be classified at a child node form a subset of the elements that can be classified at the parent node [Giunchiglia et al., Winter 2006; Giunchiglia and Zaihrayeu, 2008]. However, the

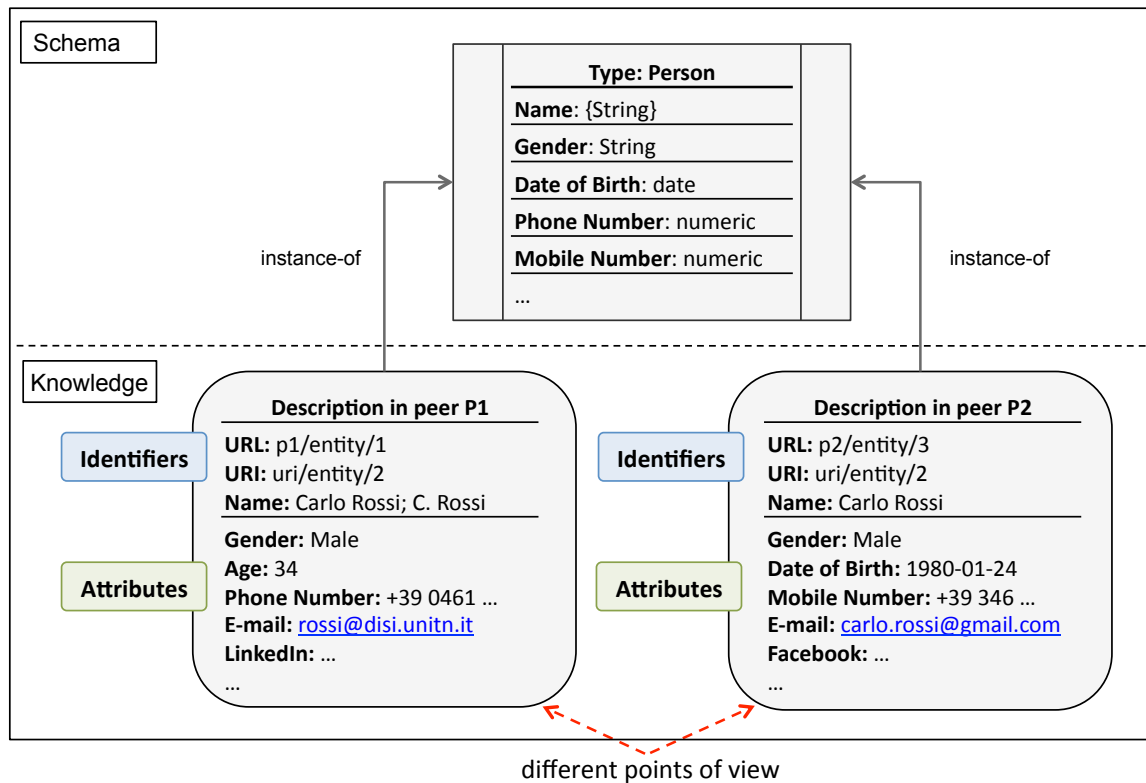


Figure 3.1: An example of an *ET* to describe entities of type “Person” and two instances showing the different descriptions of the same entity from the points of view of peers P1 and P2 respectively.

same object can be classified at different classification nodes or at different nodes from different classifications. This means that the same object can be classified from different perspectives, (i.e., considering different subsets of characteristics), which in turn allows finding objects by following different paths (from different perspectives) in the hierarchies.

Within the context of this thesis, we are interested in using classifications (also called lightweight ontologies [Giunchiglia and Zaihrayeu, 2008]) to organize contacts from peer’s contact lists (i.e., the entities describing contacts) based on certain properties. Two main dimensions can be distinguished for the organization of contacts, (i) contacts organized as subjects with whom the user is connected (or linked) through a social relation; and (ii) contacts organized as objects that are known by the peer and have certain characteristics.

### 3.4.1 Classification of subjects

As *subjects* of some social relation with the peer, contacts can be organized in social groups. For example, friends of the peer, family of the peer, colleague of the peer, etc.

Then, the organization of the groups themselves into a hierarchy defines a *classification of subjects*.

An abstract example of a possible classification of contacts as subjects is shown in Figure 3.2. In this example, social groups are mainly separated between groups of “Family” members, “Work” related contacts and “Friends”, while nodes that are lower in the hierarchy further separate the groups of the nodes that are one level above. As we also mentioned before, contacts can be classified at more than one node. For example, a close friend who happens to be also a colleague at work will be classified at two nodes, namely, “Colleagues” and “Close friends”.

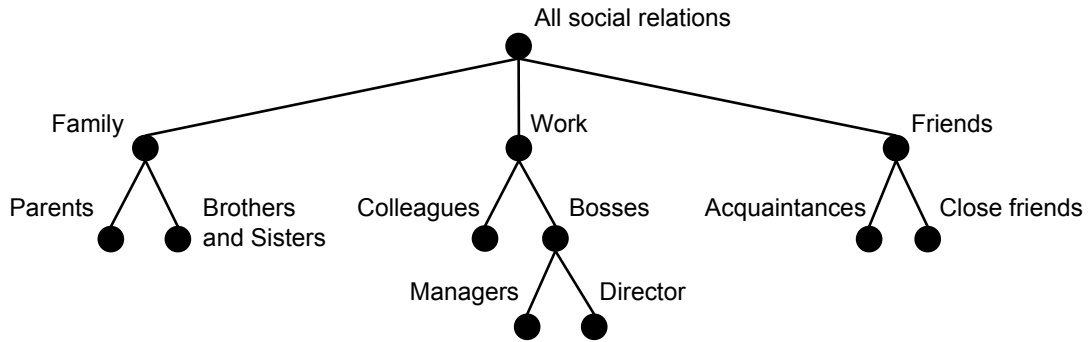


Figure 3.2: An example of a classification of contacts as subjects.

Formally, we can define a classification of subjects as a rooted tree  $CS = \langle \{n\}, \{e\}, \{l\} \rangle$ , where  $\{n\}$  is a set of nodes,  $\{e\}$  is a set of edges on  $\{n\}$ ,  $\{l\}$  is a set of labels, and for any node  $n \in \{n\}$  there is label  $l \in \{l\}$  associated with  $n$ . In  $CS$ , the label of a node represents an intended relation between entities contained by the node and the peer that owns the classification (i.e., parents of the peer, colleagues of the peer, etc.). Note that the actual meaning of a node should be understood as the meaning of all the nodes in the path to the root.

These classifications of subjects can be used in the assignment of permissions and access control rules, such as it is done in RelBAC [Giunchiglia et al., 2008]. For example, each node in the classification can be mapped to a specific set of contact information (i.e., profile or identity) to be shared by the peer with the contacts that are classified under such node. In the DCM system, in general, this type of classifications may serve the purpose of limiting the scope of certain services, among others we can mention, search contact, share or publish contact information, send contact, etc.

### 3.4.2 Classification of objects

As *objects* that exists in the real world and are described by certain attributes, contacts can be organized in groups with similar characteristics. For example, those contacts that have the same profession, live in the same city, have the same nationality, etc. Such groups can be formed by specifying only one constraint regarding the characteristics of objects or by combining one or more constraints. Then, groups can be organized in a hierarchy, from more general to more specific groups, to define a *classification of objects*.

An abstract example of possible classifications of contacts as objects is shown in Figure 3.3. Given that contacts are represented in our system through the notion of entities, the characteristics used to classify them include their types and attributes. In this example, we can see two classifications that correspond to different types, namely, “Facility” and “Person”. In the case of contacts that are persons, we can see that the peer is interested in distinguishing between those that are “Doctors” and those that are “Professors” (i.e., based on their profession). Among the doctors, we can also see that the peer is interested in “Dermatologists”, “Dermatologists” located in “Italy”, “Dermatologists” located in “Paraguay”, and “Allergists” (without distinguishing where they are located).

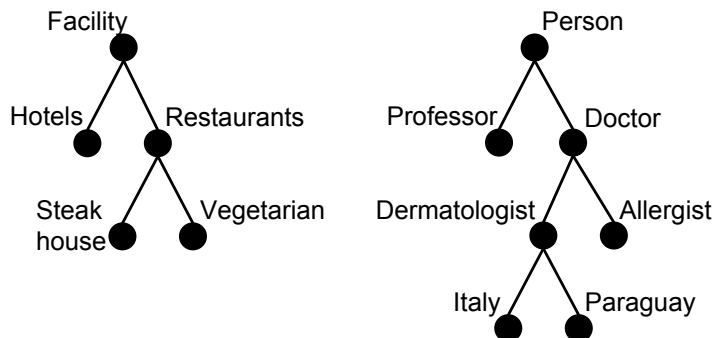


Figure 3.3: An example of a classification of contacts as objects.

Formally, we can define a classification of objects as a rooted tree  $CO = \langle \{n\}, \{e\}, \{l\} \rangle$ , where  $\{n\}$  is a set of nodes,  $\{e\}$  is a set of edges on  $\{n\}$ ,  $\{l\}$  is a set of labels, and for any node  $n \in \{n\}$  there is label  $l \in \{l\}$  associated with  $n$ . In  $CO$ , the label of a node is used to describe a characteristic that is intended on entities contained by the node. Also in this case the actual meaning of a node should be understood as the meaning of all the nodes in the path to the root.

These classifications of objects can be used in the DCM system to build catalogs of contacts, such as the yellow pages from phonebooks, and to search contacts by descriptions. Moreover, classifications from different peers can be connected by linking nodes that encode related meaning in order to build distributed yellow pages for the system.

### 3.5 Summary

This chapter introduced the foundational notions for this thesis, which will be used in the following chapters to build the Distributed Contact Management System (DCM system) on top of them. First, the notions of contact, peer and entity were defined. A *contact* is defined as an entity from the real world that is somehow contactable by possibly diverse means; A *peer* is defined as a user of the system, maintaining a contact list and capable of participating in communication activities; and An *entity* represents an abstract of physical object that exist in the real world, it has a type and is described by a set of attributes. Moreover, it was established the adoption of an entity-centric approach, which uses entities to represent contacts and peers.

Next, the notion of a knowledge schema was presented as a mean to achieve interoperability between peers. It defines templates for the different types of entities used in the system, establishing restrictions on the set of attributes used to describe a given type. These templates are then instantiated into Digital Entities (*DEs*) and their Attributes (*As*) to actually represent knowledge about entities from the real world. Two types of identifiers were introduced in association to entities in order to distinguish among many possibly different descriptions of the same real world entity. A semantic URL (*SURL*) represents a particular description, while a semantic URI (*SURI*) represents the actual entity without attaching any specific description to it.

Finally, it was discussed how the hierarchical structure of classifications (also called lightweight ontologies [Giunchiglia and Zaihrayeu, 2008]) can be exploited to organize contacts. On one hand, the notion of classification of subjects was presented as a mean to organize contacts with whom the user is connected through social ties. On the other hand, the classification of objects was presented to organize contacts based on their characteristics (i.e., as objects described by certain attributes).





## **Part II**

# **The Proposed Approaches and System**



## Chapter 4

# Reference architecture

In this chapter we present an architecture of reference for the Distributed Contact Management System (DCM System), which integrates different system components that interact in a complex manner. Moreover, we discuss the different roles played by each component in addressing the problem tackled in this thesis. This architecture is one of the contributions of this thesis.

The first step for the definition of the architecture was the elicitation of high-level requirements through the analysis of the subproblems identified from problem setting scenarios. Requirements are organized around dimensions considered of key importance in the design of such a complex system (i.e., dynamic, with socio-technical implications, inherently distributed), and cover the properties that should be met by it.

After the requirements analysis, the overall system design was thought with the following main objectives in mind:

- To meet the requirements identified in the first step of the architecture definition. These requirements imply functional as well as non-functional properties that the system should have.
- To provide an architecture that is generic enough, and that can therefore be abstracted from application specific details in order to be applicable to diverse knowledge management scenarios.

Taking into consideration the above mentioned objectives, the second step was to continue with the definition of the system architecture itself. This included:

1. The identification of actors interacting with the system and the discussion of their roles in a high-level system design (Section 4.2).
2. The identification of system components in a logical view that analyzes the role of each of them. Such analysis must include also the modules that are needed within each component (Section 4.3).
3. The identification of interactions that happen between the different components as a consequence of different types of functionalities provided by the DCM system (Section 4.4).

Finally, it is important to note that the approaches presented in subsequent chapters of this thesis, namely Chapters 5 and 6 as well as the case study from Chapter 7 will be framed within this architecture.

## 4.1 Requirements

The requirements analysis for the architecture of the DCM system takes as a starting point the problem setting scenarios from Chapter 2. In particular, we focus on deriving requirements from the different subproblems that were identified from such scenarios. In the following, we report the output of this analysis.

**Data Storage.** We follow a model that is centered in the notion of entities to represent data (i.e., knowledge)<sup>1</sup>, which means that data is stored in the DCM system using *entity bases*. The DCM system deals with a scenario that is inherently distributed. The direct impact on data storage of this inherent distribution is twofold:

- First, storage of data is also distributed among peers (i.e., peers have their own local contact list). Therefore, the system will support the storage of peers knowledge locally in their personal devices. This means that each peer should be able to maintain its own entity base.
- Second, peers need to find a minimal agreement about their entity representations in order to understand each other (i.e., for communication and interactions). Therefore, the system should provide a point of reference to get a basic (and extendable) schema as well as general purpose knowledge (i.e., information about entities of general interest).

---

<sup>1</sup>As it is presented in Chapter 3

**Peers interaction and linking.** Peers in the DCM system are also inherently connected. The links that connect peers in this network can be of different nature: (i) peers can be explicitly connected because they know each other (i.e., one peer is in the contact list of the other and vice versa), (ii) they can be indirectly connected by knowing the same people (i.e., having the contact information about the same person, facility, etc.) even if they do not know each other; (iii) peers can also be implicitly connected because they have similar characteristics, interests, or needs (e.g., one peer may have the information that another peer needs). As a consequence,

- The DCM system should support the identification and representation of different type of links between peers.
- The system have to provide mechanisms that allow exploiting these links, for example to search.

**Services.** Through the DCM system it should be possible to access different services allowing peers to manage their local contact information, search for new contacts, match existing contacts to other peers in the network, and publish (or share) their own contact information. This means that:

- The system should provide search services that can run at different levels and with different scope.
- Matching services should be provided in the system, (i) at a local level to avoid duplicates, and (ii) at a global level to link peers and contacts.
- The system should support publishing services, connected to search mechanisms in order to allow peers to be “findable” (when this is desired).
- Direct exchange services should also be supported by the system.

**Privacy.** In the DCM system, the mechanisms used for representing and searching contacts might involve the manipulation of sensitive information, for example human personal data. This may raise privacy concerns that we need to take into account. Fully addressing privacy in the system will require an extensive study of the state of the art as well as the adoption/implementation of privacy enhancing technologies that can technically enforce basic privacy principles<sup>2</sup>. Moreover, the study of privacy in the context of the DCM system represents another big research area and it is out of the scope of this thesis.

---

<sup>2</sup>as defined in the EU-Directive 95/46/EC [European Commission, 1995] and the newly proposed GDPR [European Commission, 2012]

However, we consider that privacy is a very important dimension in this type of complex socio-technical systems. To partially deal with privacy, the DCM system will provide a privacy-friendly design. This means that the system design should take into consideration main privacy principles in order to facilitate the adoption of privacy enhancing technologies.

**Performance.** The system is expected to be designed to handle a potentially large number of peers managing information about an even larger number of objects. This is reflected in the following architectural requirements:

- The system should be able to scale in the number of peers that can manage while maintaining a coherent system behavior.
- The system should also scale to handle diversity of peers, in terms of their different points of view, different needs and different interests; while still providing good quality services.

## 4.2 High-level design

The design of the DCM system aims at defining an architecture that supports the management of peers' contacts and identities in a distributed manner. In this section, we present a general view of this system that takes into consideration the different (external) actors that can interact with the DCM system in order to define the nature and mechanisms for these interactions.

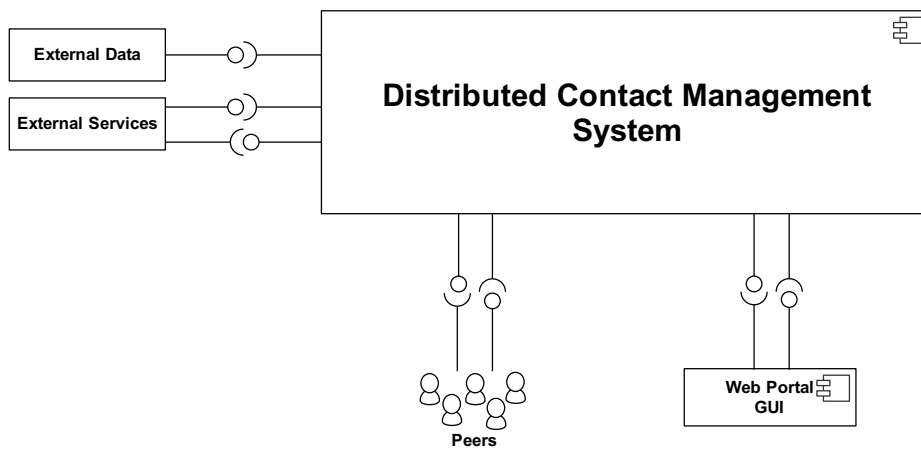


Figure 4.1: General system view

From this general perspective, the actors that interact with the system are (i) peers that use the system to manage their contacts and identities; (ii) a system administrator, who is responsible for verifying that the system works properly; and (iii) external systems possibly acting as data providers, service providers, or both. The Figure 4.1 introduces the general view of the system.

The term *Peer* refers to users of the system, which has the ability to make decisions. The Peers in the DCM system will always act on behalf of someone, might be a person, a company, organization, etc. Further, peers can decide to what extend the identity of this someone is revealed and to whom (e.g., a central authority in the system, other peers, etc.). Peers will interact with the system through applications running on personal devices (e.g., smart-phones, tablets, laptops, etc.). When interacting with the system, the peers will identify using a pseudonym (i.e., a user name), which implies the creation of an user account. The notion of a user account enables the system to link the actions and data of the same peer thereby maintaining a long-term relationship. This relation could, in turn, enable different types of services, which are available only for system users. We will discuss services in more details in the following sections.

A *Web Portal GUI* has a twofold goal. First, it will provide an interface that can be used by peers to create an user account (i.e., when interacting with the system for the first time) and to download an application for their personal device. Second, it will also provide an interface that can be used by the administrator(s) of the system to very an monitor that everything is working properly (i.e., to monitor system behavior). Note that this presupposes the existence of a central authority that stores user accounts, allow authentication, and (in general) deploys the different components of the system. This is further discussed as part of the system logical view (Section 4.3)

The *external data* and *external services* are access by the DCM system through external systems that we call *3rd Party Platforms*. They are systems over which we have no control, but that are part of the context in which our system has to function and therefore can not be ignored.

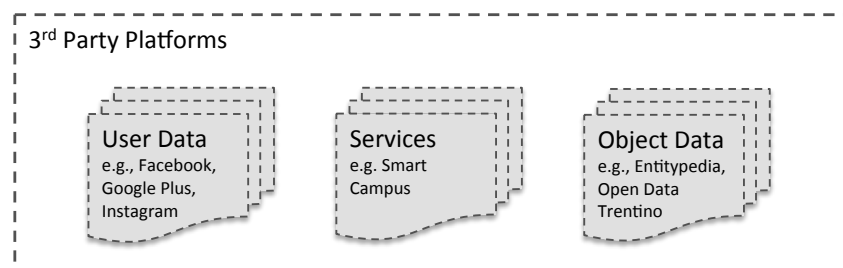


Figure 4.2: Third party platforms

According to their characteristics (information they store and the functionalities they have), the 3rd party platforms can be distinguished in three types (as shown in Figure 4.2):

- First, there are systems that are mainly based on *User Data*, all social networks are a good examples of this category. They store user information, which include personal data, preferences, and tracking of users actions. Their added value is in their user base, which can be used as target for, for example, advertisement.
- Second, we can identify other systems that are based on *Object Data*, they can be seen also as information providers. They center the attention in building big dataset of information about different types of entities from the real world (i.e., entity bases). Usually, they invest a lot of effort in producing high quality data aiming to become a referent source for a particular domain(s). Some examples of this type are, Wikipedia<sup>3</sup>, Freebase<sup>4</sup>, Entitypedia<sup>5</sup>, DBpedia<sup>6</sup>, Open Data Trentino (ODT)<sup>7</sup>.
- Third, we can distinguish those that serves as *Service* providers. These are systems that usually connect user and object data, i.e., are in the middle. An example of this are systems that process information from object data systems based on user data in order to provide customized services to users, like in SmartCampus<sup>8</sup> project.

### 4.3 System logical view

A logical view of the architecture is presented in Figure 4.3. This view shows the different components that are part of the system architecture.

The entry point and a point of reference in the system is the *DCM Portal* component. The portal can be access by a new peer through a web interface, as we mentioned before, in order to create an account and download an application that will run locally in its device. This application that we call *DCM App* represents the second component of the system. The app allow peers to interact with each other and with the portal in order to have access to different type of services (like search). Finally, the peers interacting through the app represent the third component of the system, namely, the *DCM Network*. In what follows we present each component of the system in more details.

---

<sup>3</sup>[www.wikipedia.org](http://www.wikipedia.org)

<sup>4</sup><http://www.freebase.com/>

<sup>5</sup><http://entitypedia.org>

<sup>6</sup><http://dbpedia.org>

<sup>7</sup><http://dati.trentino.it/>

<sup>8</sup><http://www.smartcampuslab.it>



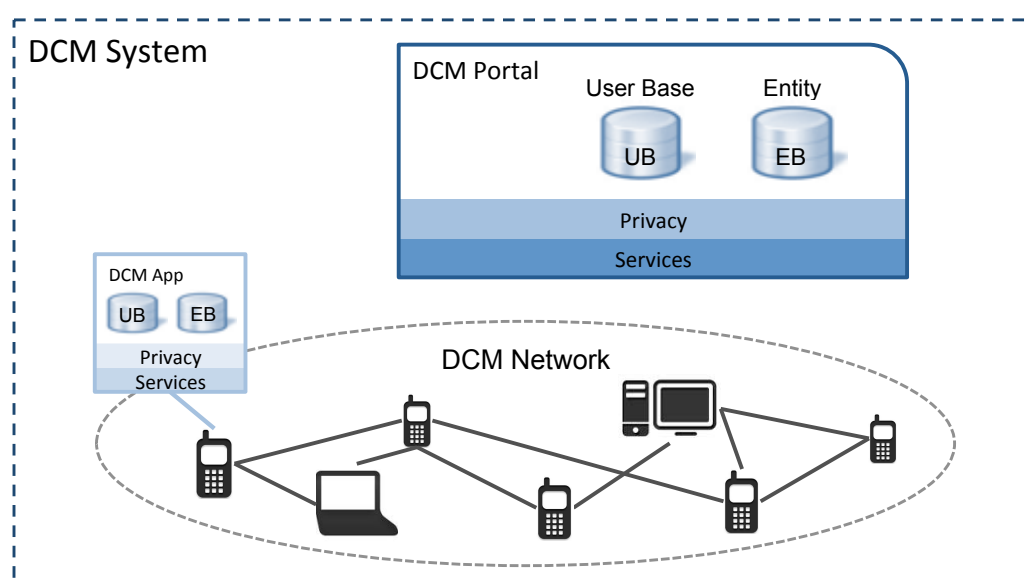


Figure 4.3: Logical Architecture

### 4.3.1 Contact management portal

The *DCM Portal* is composed by three layers, namely, (i) a data layer, (ii) privacy layer and a (iii) service layer.

The *Data layer* stores information of users in a *User Base (UB)* and information about entities that are known to the portal in an *Entity Base (EB)*:

- The *UB* of the portal is intended for identifying users, called peers in the DCM system. It stores information about the pseudonyms (i.e., user names) under which the system recognizes peers, regardless of the real world entity on behalf of whom they are acting. The *UB* can also maintains other peer-related information, such as, the last time the peer was seen, available mechanisms to send them notifications, etc. The identification of a peer in the *UB* is based on a user name, which can be any arbitrary unique combination of characters (i.e., letters, numbers and symbols). Moreover, user names are not mapped to a concepts from a knowledge base (e.g., WordNet<sup>9</sup>). However, a user name can be linked (although it is not mandatory) to an entity in order to show that the corresponding peer is acting on behalf of such real world entity.
- On the other hand, an *EB* is intended to store the knowledge of the portal, namely, descriptions of the real world entities (e.g., persons, organizations, facilities, loca-

<sup>9</sup><http://wordnet.princeton.edu/>

tions, events, etc). This include, first, the definition of the semantic schema<sup>10</sup> (i.e., set of templates) used to describe entities in the DCM system. Second, it includes the description of entities on behalf of whom peers act, called the profile of the peer. Last, it also includes other related entities that are needed to describe the general knowledge that the portal has about the real world. The place of birth of a person or the location where he/she lives; the place where an organization has its headquarters, or even the institution where a person works are all examples of general interest entities that can be part of the portal knowledge and can also be used to describe the profile of a peer.

The *Privacy layer* plays the role of a filter that knows who can access what and under which circumstances. We define a privacy layer as part of a privacy-friendly design. This layer is intended to account for the need of an agreement between the peers and the portal regarding how peers data will be manipulated in the portal. Such agreement could be achieved and technically enforced in an automatic (or semi-automatic) manner with the adoption of appropriate privacy-enhancing technologies. As we mentioned before, providing concrete solutions for this layer is out of the scope of this thesis and we leave it as part of our future work. However, we invite the interested reader to find more information in [PrimeLife, 2011].

The *Service layer* encapsulates a number of services that are supported by the system. They include, among others: creation of user accounts, downloading the DCM App, searching users (for example, to find if a local contact is also a user of the system), searching contacts that were published in the portal, etc. The portal also offers services for the protection of user's data in the network. It is important to note that between this layer and the data layer (as shown in Figure 4.3) we have the privacy layer. This means that all services provided by the portal presuppose the evaluation of privacy (what data can be revealed by the services, for what purpose and to whom).

Finally, It is important to point out that the separation among layers is mainly conceptual and the actual implementation of privacy policies or services might be transversal to different layers. For example, the privacy policies could require additional metadata to be stored with the data they protect, in the data layer (UB and EB). Similarly, different services might also require storing some additional information (e.g., indexes, statistics, etc.).

---

<sup>10</sup>The notion of a semantic schema is discussed in Chapter 3

### 4.3.2 Contact management application

Another key component in the architecture is the *DCM Application (DCM App)* running on local devices of peers. Once a peer creates a user account and installs the DCM app, this will be its everyday interface to into the system. In fact, we say peers become part of the system through the DCM app. As it can be seen in Figure 4.3, the internal structure of the DCM App is in principle the same as the DCM Portal (i.e., composed by three layers). It can also be seen that a slightly different color is used for the DCM App, which is aimed to point out that they work at different levels (i.e., the portal at a global level and the app at a local level).

The *Data Layer* at peer  $P_i$  maintains a *User Base*  $UB_{P_i}$  and a *Entity Base*  $EB_{P_i}$ , which contain the following information:

- The  $UB_{P_i}$  at peer  $P_i$  stores information regarding the DCM user account of the local peer. This information will be used, for authentication purposes, in any interaction that the peer has with the DCM portal. Additionally, the  $UB_{P_i}$  stores information about others (external) user accounts (e.g., Facebook, Google Plus, Skype, etc.) of the peer and relevant user information related to these accounts (for example, contacts of the peer in external accounts).
- The  $EB_{P_i}$  at the same  $P_i$  stores only the information about the entities of the user(s) in  $UB_{P_i}$ , i.e., entities that  $P_i$  knows or that are somehow relevant for  $P_i$ . We say that the  $EB_{P_i}$  represents  $P_i$ 's point of view or knowledge about objects from the real world.

Let us give an example to clarify the difference between the information stored in the different bases. Take the example of Anna and Liza, they are friends in Facebook and Anna is a user o the DCM system. Let us call  $UB_{P_a}$  and  $EB_{P_a}$ , respectively, to the user base and entity base in Anna's DCM app. If Anna has the contact information (i.e., name, cellphone number, address, etc.) of Liza in the DCM app, the description of Liza as a person will be in  $EB_{P_a}$ . This description may include attributes like, name, gender, nationality, cellphone number, address, and Liza's user name in Facebook. However, knowing that Liza has a Facebook account does not imply that they are Facebook friends. If Anna imports information about her Facebook friends into the DCM app, then Liza's user name from Facebook will be stored in the  $EB_{P_i}$  as a contact associated to an external account of Anna.

The *Privacy layer* at peer  $P_i$  filters who can have access to the local information in  $UB_{P_i}$  and  $EB_{P_i}$ . Following the same reasoning as in the case of the DCM portal, this layer is included in the DCM app as part of a privacy-friendly design. Moreover, the

privacy layer at the DCM app is intended to negotiate and agree with corresponding privacy layers, at the DCM portal and at other peers.

The *Service layer* at peer  $P_i$  is intended to communicate with the corresponding service layers at the DCM portal or at other peers, in order to have access to services that they can offer to  $P_i$ . At the same time, other peers can access to services provided by  $P_i$  through this layer (for example, in a peer-to-peer manner). In general, this service layer can be seen as the set of APIs through which the DCM app communicates with other components of the system. This presupposes that the service layer, at the portal and at different peers, speak the same language and can therefore understand each other. However, the computation of a particular service is subject to privacy constraints regarding the data they are allowed (or not) to reveal.

### 4.3.3 Contact management network

The third component of this architecture is the network formed by peers running the DCM app and interacting with each other (i.e., a P2P network), namely, the *DCM Network*. There are different types of links that can connect peers in the DCM system. For example, we can consider that two peers are linked at the data level if they store information about the same entity. However, we could say that they are linked at the service level if they interact providing services to each other or a general distributed service in the network (e.g., distributed search). In this thesis, we propose different approaches that model these connecting links between peers.

We believe the added value of exploiting this network is twofold,

- First, we can make the system more scalable by leveraging on intrinsic characteristics of P2P networks, which increase their capabilities as the network grows. In fact, distributed solutions can directly exploit the distributed nature of the contact management problem.
- Second, by providing services that are based on distributed solutions (e.g., distributed search, direct contact exchange) we can better support privacy. In other words, for the system to work there is no need to reveal everything to some central authority (in this case to the DCM portal). The data can remain under the control of the peer, which means that the peer can decide what information to reveal and to whom (through the privacy layer at the DCM app). Note that when talking about contact information of people, we are referring to manipulation of potentially sensitive information.

Finally, the importance of this component for the system goes beyond the simple sum

of the number of peers (DCM apps) participating in the network. In this thesis, the models that are needed to build such network are part of the contributions.

## 4.4 System dynamic view

The dynamic view of the system outlines the interactions among the different system components. These interactions can be very complex as some activities may require the many parallel threads of execution or iterative component interactions. However, we believe that a simplified view of such interactions can help in understanding the overall system behavior. We organize the description of these interactions around the peers, who we consider as main actors in our system.

At a very high-level, we identify three main use cases for the peers: (i) initialization, (ii) sharing, and (iii) search. Each of them, in turn, can include smaller use cases that can also be executed independently. For the sake of making the dynamic view of the system more understandable, we avoid going into the details of more atomic use cases. Instead, we present only general sequence diagrams for each of the main uses cases in this section. The more atomic cases are covered by the scenario descriptions in Section 7.3.

The *initialization diagram* presented in Figure 4.4 shows the interactions that take place among the various system components when a new peer starts using the system. The whole process begins with the peer accessing to the DCM portal, creating an account and downloading the DCM application. Such user account is returned to the peer, who can now install the DCM app on its personal device and register its account. The account registered in the DCM app tells the system the pseudonym on behalf of whom the application is acting. In order to register the account, the DCM app asks the DCM portal to authenticate the user. Without registering an account, the peer will not be able to join the network. The future interactions of the peer with the system will be now done through the DCM app.

A this point, the DCM app asks the peer to input some initial configuration parameters, which may include defining different profiles to be used in the system and setting privacy preferences or at least verifying the default settings. Finally, the DCM app connects to other peers in the system and joins the DCM network.

The *sharing diagram* presented in Figure 4.5 presents the system components interactions taking place when a peer shares its contact information (i.e., its profile). Peers will be able to share their contact information within the system using different tools. In the diagram we show a high-level view outlining that different mechanisms will require interaction between different components. First, we see that the peer directly interact only with the DCM app. Then, the DCM app identifies which mechanism should be

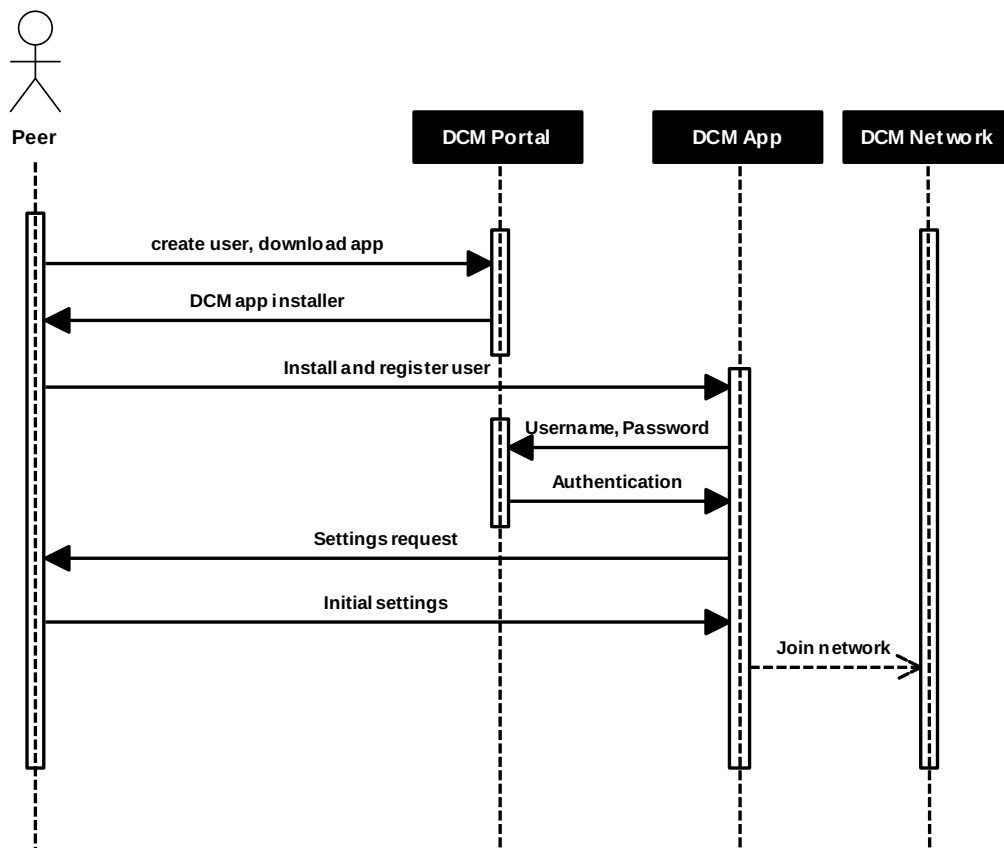


Figure 4.4: Sequence diagram of the initialization process of a new peer in the system.

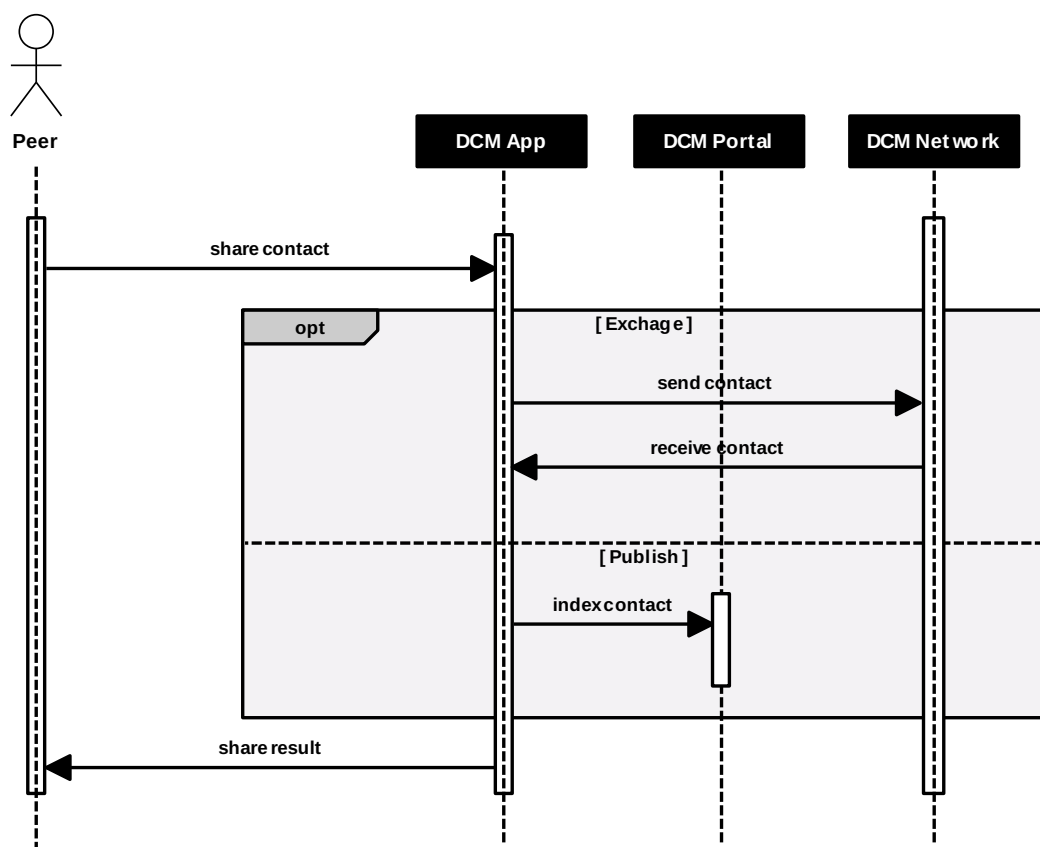


Figure 4.5: Sequence diagram of a peer sharing its contact information.

used. Such identification will be based on the request received from the peer, different settings previously established by the peer and any other information that the DCM app may have.

As we can see in the diagram, the DCM app will interact with the DCM network for exchanging contact information with other peers. On the other hand, the peer may rely on the DCM portal and publish its contact information there. In fact, the peer may decide to share its profile with the portal for different reasons. For example to be included in some sort of public directory (e.g., hotels and restaurants would be interested on this) or to prove its identity in order to ask the system to protect its personal information in the network (e.g., people may be interested on this)<sup>11</sup>. Finally, the result in this diagram refers to any notification or confirmation message that need to be shown to the peer as an outcome of the sharing activity.

The *search diagram* presented in Figure 4.6 shows the interactions that take place among the various system components during search. Peers can search in the DCM system with different purposes and, therefore, providing as input different parameters (i.e., different types of query). We see in the diagram that also in this case the search process is triggered at the DCM app. The first distinction to be made by the DCM app is between searching of users and searching contacts. The aim of searching a user would be to find if someone's known profile match to an exiting user of the system. In order to perform such type of search, the DCM app have to interact with the DCM portal as the portal manage the list of users of the system (i.e., through the portal's UB) and their associated profiles (i.e., through the portal's EB). Notice that the DCM portal should return the user name only if the corresponding peer has authorized (i.e., has decided to be findable by a given profile).

On the other hand, the aim of searching a contact is to actually find the profile (i.e., contact information) of someone. However, again here the DCM has to distinguish between the case of a precisely known target (i.e., search in white pages) and the case of an unknown target that should have certain characteristics (i.e., search in yellow pages). The search process in these cases is characterized by the initialization of a new thread of execution that interacts with the DCM portal and DCM network in the first case, while it only (or mainly) interacts with the DCM network in the second case. The new execution threads shown in the diagram are aimed to point out that these search mechanisms require asynchronous (and maybe also iterative) communications between components. Moreover, both types of search can be combined by executing them in parallel or sequentially. In this thesis we present an approaches to address each of these two types of searches. They

---

<sup>11</sup>We will further discuss how these types of functionalities can be supported by the approaches presented in the following chapters.



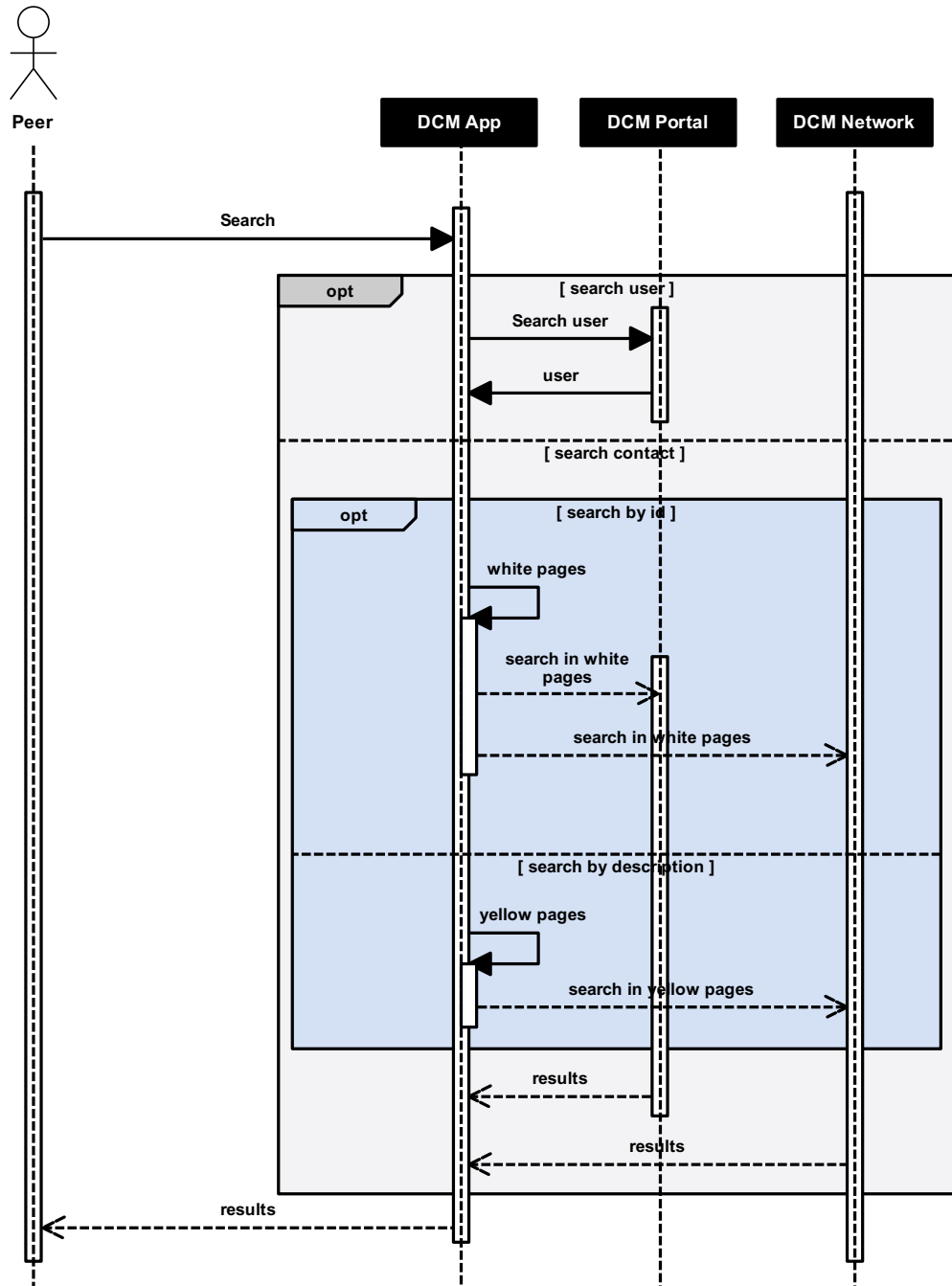


Figure 4.6: Sequence diagram of a peer searching the system.

are discussed in details in Chapters 5 and 6, respectively.

## 4.5 Summary

The reference architecture for the Distributed Contact Management System (DCM System) was presented in this chapter. First, the analysis of the system requirements was presented. The outcome of such analysis was discussed in terms of: (i) data storage for an inherently distributed scenario, (ii) peers interaction and linking, (iii) services that the system needs to offer, (iv) possible privacy concerns that may arise, and (v) the system performance.

Taking into consideration the identified requirements a *general view* of the system was presented. In it, the different (external) actors that can interact with the DCM system were defined, as well as the nature and mechanisms for these interactions. Next, the system *logical view* introduced the DCM Portal, DCM App and DCM Network as key system components allowing peers to create accounts, interact with each other and access to different type of services. Last, a *dynamic view* of the system was presented, showing a simplified view of different interactions among system components. Three types of interactions were distinguished: initialization, sharing and search.

## Chapter 5

# A name-based approach to search in the DCM system

We see DCM network of peers (a P2P network) organizing their content in directories, which digitally represent their own versions of *entities* that exist in the real world. Entities as defined in Chapter 3 (Section 3.3.2) can be of different types (e.g., person, location, event and others), they have a name, and are described by attributes (e.g., latitude-longitude, size, birth date), which are different for different entity types [Bazzanella et al., 2008]. Different versions of an entity can represent different points of view, they could show different aspects of the entity or the same aspects with different level of details. In a way, the local representations from peers can be seen as pieces of information about a particular entity that are stored in a distributed manner in the network.

In this network, the different directories contain related data and, to some extent, they can complement each other. One problem that prevents us from exploiting the relation between these data is that there are no links connecting the local directories from peers. An effort to connect related data on the web is that of Linked Data<sup>1</sup>, which allowed linking important datasets like, dbpedia, Freebase, DBLP, ACM, and others. Nevertheless, this approach leaves out of the semantic web the individual users (i.e., simple normal peers) and the data from their local directories stored in personal devices (e.g., smart-phones, PDAs, notebooks, etc.). We propose an approach to build a distributed directory that constructs the connecting links among the local directories of peers from the DCM network. It is important to note that the model applies to any entity in the *EB*

---

<sup>1</sup><http://linkeddata.org/>

of peers that describe a real world object and not only to those describing a contact of the peer. The aim of this directory is to become a bridge allowing to link peers in the DCM network that are interested in the same entity (maybe even from different perspectives).

As in any directory, one way in which peers normally identify and distinguish an entity from others is by means of names (e.g., George Lombardi, Trento, Italy, University of Trento), which play a different role from the other attributes because they are identifiers rather than descriptions [Holloway and Dunkerley, 2004]. The values of other types of attributes have a meaning that can be understood, e.g., by mapping them to concepts from a knowledge base, like WordNet<sup>2</sup>. Names, on the other hand, are strings that behave similarly to keywords. Real world entities can be called by multiple names as a consequence of variations and errors. Moreover, the set of names used in different local representations to identify the same real world entity can be different, at the same time that the sets of names used to identify different real world entities can overlap.

The approach we propose incorporates the notion of a real world entity described by different local representations from peers. This notion is used to organize the references to the local representations in order to build a distributed entity directory that allows finding all the available information about entities. Our system offers two main features:

- First, it takes into consideration that multiple, possible different, names can be used to identify the same real world entity (e.g., George Lombardi vs. G. Lombardi and Italy vs. Italia).
- Second, it allows peers to have control over the privacy of their data because the entity directory stores only the names of the entity and a link to the local representation and not the data itself.

As a result, any name that is used in some local representation to identify an entity can be used to find the different versions of that entity that are stored in the network of peers.

The rest of this chapter is structured as follows. Section 5.1 presents a motivating example, while Section 5.2 formalizes the basic notions that link the different directories. Further, the Section 5.3 discusses the name matching problem that arises when linking different directories and the algorithms to perform search over the proposed name-based overlay are presented in Section 5.4. Finally, a summary of the chapter is presented Section 5.5.

*An initial version of the work presented in this chapter was published in [Giunchiglia and Hume, 2012], while an extended version was published in [Giunchiglia and Hume, 2013b] and [Giunchiglia and Hume, 2013a].*

---

<sup>2</sup><http://wordnet.princeton.edu/>

## 5.1 Motivating example

Nowadays, most of the organization of our data is done in terms of directories. A well known (and old) example is the telephone book directory, used to organize address and phone numbers of people and companies. Newer forms of directories can be seen, for example, in contact lists, document directories, event directories (i.e., calendars or agendas) used by peers in current devices (e.g., computers, PDAs, smart-phones) to organize the local representation of entities of their interest. Moreover, the data from different directories (possibly from different peers) can be related. Different peers attending to the same event might store local representations of the event. Each of them might also have the contact information of the other peers attending to the event, e.g., a meeting.

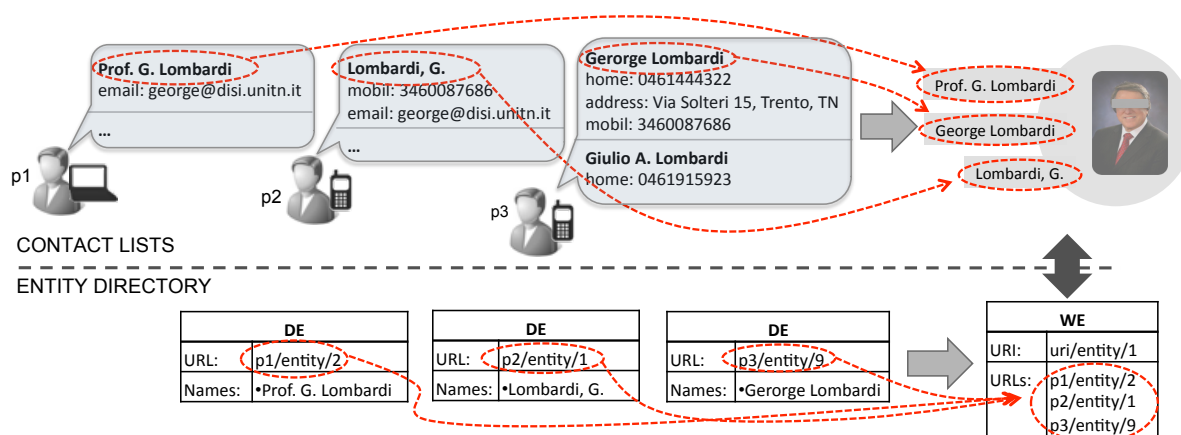


Figure 5.1: Example of Contact Lists Related by Identifiers

Let us consider in details the example of contact lists in different devices from the peers of a network that connects students, researchers and professors among them (e.g., SmartCampus<sup>3</sup>), and with their family members. The first part of Figure 5.1 (upper part) shows that the contact list of each device can be seen as a local directory of people. Different peers in this network can have different information about the people in their contact lists, like phone numbers, email addresses, skype user and others, which show different ways to get in touch with them. For example, suppose that  $p_1$  is a student that is taking a course with prof. George Lombardi and therefore  $p_1$  has, in its contact list, the university email address of the professor. A researcher  $p_2$  that is working with him could have more information, like his email and mobile phone number. On the other hand, a family member  $p_3$  may have his home address and phone number but not the university email (because such information is not relevant for  $p_3$ ).

<sup>3</sup><http://www.smartcampuslab.it>

Now, suppose that another researcher in the network, let us call it  $p_4$ , hears about prof. Lombardi work and wants to contact him. We can see that:

1. First, the information that  $p_4$  needs is distributed in the network and the problem is knowing where the different pieces are stored
2. Second, the different peers can call the same person using different names, e.g., Prof. Lombardi, George Lombardi, G. Lombardi. In our example, this means that  $p_4$  need to be sure that the other peers (i.e.,  $p_1$ ,  $p_2$  and  $p_3$ ) are all referring to the same person as he is.
3. Third, the contact information can change in time. The work email of Prof. Lombardi will change if his affiliation changes, his phone numbers can change at any time, and his address will change if he changes residence.
4. Finally, the privacy and the sensitiveness of the information have to be considered. Most likely the phone number and address of the home of prof. Lombardi would be more private than the university email. As a consequence,  $p_3$  will not share such information with everyone.

## 5.2 A name-based overlay for linking directories

A name-based overlay can be build by formalizing a model that link data from different directories. We propose a model that distinguishes between a Digital Entity ( $DE$ ) and a Real World Entity ( $WE$ ).

### 5.2.1 Formalization of the model elements

A  $DE$  is defined as a local representation of an entity that exist in the real world. Its formal definition was introduced in Chapter3, however, for the sake of simplicity we use a minimal version of such definition in this chapter to represent  $DEs$ . A  $URL$  (Uniform Resource Locator) is used in order to uniquely identify a  $DE$  and it can be used (by dereferencing) to obtain the full local description (i.e., based on attributes). We also consider a set of names  $\{N\}$  as the human readable identifiers used in  $DEs$  to refer to a  $WE$  and distinguish it from others. Formally,

$$DE = \langle URL, \{N\} \rangle \quad (5.1)$$

On the other hand, we referred before to real world entities without formally defining them. In this chapter, we introduce a formal representation for  $WEs$  as part of the model that is proposed in this section. A  $WE$  represents the real world entity and is modeled

as a class of *DEs*. We use a *URI* (Uniform Resource Identifier) to uniquely identify each *WE*. Formally,

$$WE = \langle URI, \{URL\} \rangle \quad (5.2)$$

where  $\{URL\}$  is a non-empty set of identifiers of different *DEs* that describe *WE*. As a consequence of the composition of these definitions we can see that multiple sets of names are given to a *WE* through *DE* definitions from different peers that describe the same *WE*.

In the second part of Figure 5.1 (lower part) we show how the example from Section 5.1 can be formalized in terms of these notions (i.e., *DEs* and *WEs*). We can see a *one-to-one* mapping between the *WE* from an entity directory and the real person represented in different contact lists. Moreover, we see that an entry from a contact list is translated into a *DE* in the directory (i.e., also a *one-to-one* mapping). There is a *one-to-many* relation between *WEs* and *DEs* which shows that each single entry in a contact list correspond to one person but one person can be described in many different entries (possibly from different peers). Finally, the relation between *Names* and *WEs* introduces a name matching problem that is better discussed in the following section.

Note that these notions allow the separation between “what” is being represented and “where” is being represented. This separation is needed in order to model the issues stated in items 1 and 2 from the example of Section 5.1. The *DEs* model the different pieces of information that  $p_4$  needs and their *URLs* tell us where they are. The *WE* models the link that connects different *DEs* and its *URI* identify what they represented. Regarding item 2, we can see that different sets of names are given in *DEs*, which models the fact that  $p_1$ ,  $p_2$  and  $p_3$  can define the different names that they use to call an entity.

On the other hand, the distinction between the two notions (*DE* and *WE*) also provide the infrastructure to deal with the issues introduced by the other two items (i.e., items 3 and 4 in Section 5.1). The dynamism of the information about the entities and the privacy of local data are constrained to affect *DEs*. In this way, when the email of Prof. Lombardi changes (see Figure 5.1),  $p_2$  (the researcher) updates its local representation (i.e., the *DE*). The corresponding *WE* definition is not affected by this update, nevertheless the information (available in the P2P network) about Prof. George Lombardi is updated. Similarly, access control can be implemented over the data associated to each single *DE* representation, which do not affect *WE* definitions. Note that such implementation (i.e., access control implementation) is out of the scope of this paper, but the interested readers are invited to see (for example) [Giunchiglia et al., 2008].

### 5.2.2 Building the name-based overlay

After the elements of the name-based overlay has been formally defined, the information about entities can be organized by incorporating the notions of *WE* and *DE*. These notions allow the separation of the problem of finding the *DEs* that represent different versions of a *WE* from the problem of finding *WEs* that are identified with multiple names. We exploit this separation by building two different indexes, one to deal with each problem.

A *DEindex* is created to map *WEs* (i.e., *URIs*) to *DEs* (i.e., *URLs*) and can be formally defined as,

$$DEindex = \{WE \rightarrow DE \mid \nexists WE' \rightarrow DE \in DEindex \text{ s.t.}, WE' \neq WE\} \quad (5.3)$$

We can see that this index encodes the *one-to-many* relation between *WEs* and *DEs* because the mapping of different *WEs* to the same *DE* is not allowed. On the other hand, a *WEindex* is created to map the names that are given (in local representations) to *WEs* (i.e., *URIs*). Let us call  $\{N^{DE}\}$  to the set of names of a digital entity *DE*. Then, the *WEindex* can be formally defined as,

$$WEindex = \{N \rightarrow WE \mid \exists WE \rightarrow DE \in DEindex \text{ s.t.}, N \in \{N^{DE}\}\} \quad (5.4)$$

We can see that this index encodes the *many-to-many* relation between *Names* and *WEs* because the only constraint on the mappings is related to the existence of a local representation that gives “support” to such mapping.

Let us now discuss in more details how the publication, maintenance and search of entities are done over this *name-based overlay* (also called *EntityDirectory*):

- The *publication and deletion of DEs* in the network are the two main events that modify the *EntityDirectory* by affecting the content of the indexes defined above. The publication of a *DE* affects both indexes in a straightforward manner. First, the *DE* is associated to the *WE* that it represents by adding the corresponding mapping (i.e.,  $WE \rightarrow DE$ ) to the *DEindex*. Second, the mappings  $N_i^{DE} \rightarrow WE$ , of each name  $N_i^{DE}$  in  $\{N^{DE}\}$  to the *WE* that is associated to the *DE*, are added to the *WEindex*. In order to do this, we assume that the peer locally caches the identifier (i.e., the *URI*) of the *WE* that is represented by its *DE*<sup>4</sup>. On the other hand, when a *DE* is deleted from the network, only the *DEindex* is directly affected. The same mapping  $WE \rightarrow DE$  that is added when the *DE* is published, is then removed from the *DEindex* when the peer deletes the *DE*. Regarding the *WEindex*, we say

<sup>4</sup>Note that the initial identification of the *WE* described by a *DE* is a problem of identity management and is out of the scope of this work. See for example [Hogan et al., 2012; Bouquet et al., 2008]



that it is not directly affected because the mappings of names can be removed only after verifying that they are no longer valid to identify the corresponding *WE*. Such verification is further discussed as part of the *EntityDirectory* maintenance.

- The *maintenance* of the *EntityDirectory* is performed through periodic checks over the indexes in order to detect and remove entries that are no longer valid. In the *DEindex*, an entry can be considered invalid if it contains mapping to a *DE* that has been unreachable for a long time. In order to detect this situation, each entry is attached with a timestamp corresponding to the last time when the *DE* was reachable. This timestamp is updated in every periodic check. When the *DE* is not reachable, the interval between the last reachable time and the current time is verified. The corresponding entry is removed from the *DEindex* if such interval exceeds a given threshold. An entry from the *WEindex*, on the other hand, is considered invalid if it contains a mapping that do not complies with the constraint established by the index definition presented in equation 5.4. This means that a mapping between *N* and *WE* has to be removed from the *WEindex* when there are no *DEs* in the network using the name *N* to refer to such *WE*. In other words, when none of the available entities provide support to such mapping.
- *Search* in the *EntityDirectory* can be performed using two different types of identifiers, *URIs* and *names*. In this context, having as input a *URI* means that the target *WE* has been uniquely and fully identified. Therefore, the goal of the search is to obtain all the different representations (i.e., the *DEs*) of the *WE*. On the other hand, in a search based on names, we need to find the candidates *WEs* (to be the right answer) as a consequence of the *many-to-many* relation between names and *WEs*. After the candidates *WEs* has been found, we can use the search by *URI* to find the different representations of them. In what follows, the search by names is considered in more details while the search by *URIs* is included as a part of the former.

A query is formally defined as  $Q = \{N^Q\}$ , where  $\{N^Q\}$  is the non-empty set of names used to identify one target *WE*. Then, the problem of searching entities based on their names can be seen as retrieving *WEs* that are described in the network by at least one *DE*, such that, the intersection between  $\{N^{DE}\}$  and  $\{N^Q\}$  is not empty. This definition considers a partial matching between  $\{N^{DE}\}$  and  $\{N^Q\}$  in order to allow finding a *WE* from any of the names given to it on different *DEs*. In turn, this can be translated in the formal definition of the Query Answer (*QA*) as follows:

$$QA = \{\langle WE, \{DE\} \rangle \mid \exists N' \in \{N^Q\} : N' \rightarrow WE \in WEindex \wedge \forall DE' \in \{DE\} : WE \rightarrow DE' \in DEindex\} \quad (5.5)$$

As we mentioned before, this answer is build in two steps. The algorithms that perform the two steps are presented in Section 5.4.

### 5.3 Name matching

Names are human readable identifiers that serve the purpose of distinguish an entity from others. They are labels composed by a combination of words, numbers and symbols [Holloway and Dunkerley, 2004]. In the context of our entity directory, we define the set of names that identify a *WE* as the union of the names used in *DEs* that locally represent that *WE* in different peers. Names are different from other attributes because they play the role of keywords rather than been mapped to concepts from a knowledge base. As such, names can suffer from different types of variations. Following the results from the study performed in [Bignotti, 2012], we can distinguish among the following types:

- **Format.** The format variations have a strong dependence with entity type and affect mostly to people names. They include the variation of the order in which the words of a name can be written (e.g., *George Lombardi* and *Lombardi, George*) and the multiple abbreviations that can exist for the same full name (e.g., *Giulio Augusto Lombardi* can be abbreviated as *G. A. Lombardi*, *Giulio A. Lombardi* and others). It is also important to notice that the abbreviation of a name can be a valid reference to many different full names (e.g., *G. Lombardi* is valid for *George Lombardi* but also for *Giulio Lombardi*).
- **Full translations.** Names sometimes are written differently in different languages (e.g., *Trento* in Italian, *Trient* in German or *Trent* in English).
- **Part-of translations.** In other cases only one part of the name changes in different languages. This is the case of names composed by common and proper nouns, where the common noun is called trigger word in [Bignotti, 2012] and is the only part that is affected by the translation (e.g., *University of Trento* vs. *Università di Trento*).
- **Misspellings.** Names can be misspelled, either in the definition of a *DE* or during the specification of a search query. The misspellings can be a consequence of variations in the punctuation, capitalization, spacing, omissions, additions, substitutions, phonetic variations (e.g., *Fasuto* vs. *Fausto*, *G Lombardi* vs. *G. Lombardi*).
- **Pseudonyms.** Entities also have pseudonyms that are not (necessarily) variations of a name but rather alternative names for an entity, which can be defined (and used) in different contexts. This is the case for some arbitrary nicknames that are sometimes used by peers to refer to a *DE* (e.g., *Fede* is commonly used as a nickname

for *Federico* or *Federica* and *The King of Rock and Roll* is a common nickname for *Elvis Presley*).

The name variations together with the *DE* definition presented above, show that the relation between names and *DEs* is of the type *many-to-many*. In turn, this leads to a name-matching problem when we intend to search an entity based on its names [Holloway and Dunkerley, 2004]. This problem, in the context of the entity directory, can be decomposed in:

1. The problem of matching names inside the network: A name used in a *DE* can be a variation of the name used in another *DE* that represent the same *WE*. We need to take into consideration all the multiple names (including name variations) used in the network to identify a *WE* and match them to all the different *DEs* that describe *WE*. In the example from Figure 5.1, if the user is searching an entity with the name “*George Lombardi*”, the directory should be able to return all the *DEs* (i.e., *p1/entity/2*, *p2/entity/1* and *p3/entity/9*) that represent the different versions of *uri/entity/1* rather than only returning the one that give it such name (i.e., *p3/entity/9*).
2. The problem of matching queries with the names used in the network: This case considers query names that are unknown to the entity directory, but that are however variations of one or more known names. We say that a name is unknown to the directory if there is no *DE* in the network that uses such name to identify a *WE*. The easiest example is a query name that is misspelled with regard to the *DEs* of the directory. In the example from Figure 5.1, if the user input the query “*Goerge Lombardi*”, the search should be able to find that “*George Lombardi*” is a candidate match.

## 5.4 Algorithms

We assume that the indexes offer non-blocking APIs (to allow the parallelization of index lookups), which mean that a call to the *GET* function on the indexes returns immediately a reference to an object that will be filled with the results from the index lookup. In Algorithm 1, we define the global data structures, which are strictly related to the indexes. They are used across the different functions involved in the search. We use the statement *for all* (line 6 in Algorithm 2 and line 8 in Algorithm 3) to denote the concurrent execution of the statements that are in its body (i.e., line 7 in Algorithm 2 and lines 9 to 24 in Algorithm 3).

The Search Entity function is presented in Algorithm 2 and is the main entry point for the search by names. This function receives the query names and returns a set of candidate *WEs* according to the constraints given in Equation 5.5. In order to measure how relevant each candidate *WE* is, we count the number of query names that match with the names associated to the *WE*. This relevance is associated to each candidate *WE* and included in the resultset. In line 7, the first step of the search by names is initiated with the call to the *GetWEindex* function of the *WEindex*. The object returned by the function is given to the corresponding handler function, which knows how to process it.

---

**Algorithm 1** Global Data Structures
 

---

- 1: WEAnswer :  $\langle \text{isComplete}, \text{name}, \text{weAnsValues} \rangle$
  - 2: DEAnswer :  $\langle \text{isComplete}, \text{URI}, \text{deAnsValues} \rangle$
  - 3: isComplete : boolean ▷ TRUE when the index lookup is finished
  - 4: weAnsValues : NULL OR {URI} OR {URL} OR {{URI}  $\cup$  {URL}}
  - 5: deAnsValues : {URL} ▷ not empty set of URLs
- 

---

**Algorithm 2** Search Entity
 

---

- 1: **function** SEARCHENTITY(names : {name})  $\rightarrow$  { $\langle$ WE, relevance $\rangle$ }
  - 2:   WES : { $\langle$ WE, relevance $\rangle$ } ▷ stores search results
  - 3:   WE :  $\langle$ URI, {URL} $\rangle$  ▷ {URL}.size == 1 when URI == NULL
  - 4:   relevance : integer
  - 5:   WES := {}
  - 6:   **for all** name  $\in$  names **do** ▷ Parallel threads
  - 7:     HANDLEWEANSWER(GetWEindex(name), WES)
  - 8:   **end for**
  - 9:   **return** WES
  - 10: **end function**
- 

The Algorithm 3 shows the *HandleWEAnswer* function, which is in charge of processing the values retrieved from the *WEindex*. We can see from lines 4 to 6 the loop that waits until the answer is completed. Then, in line 8, we start one execution thread to process each retrieved value. A value returned from the *WEindex* represents a *WE*, it can be a *URI* or a *URL* (see line 4 from Algorithm 1). In the former case, we say that the *WE* identity is known. The corresponding instance is created (line 10 in Algorithm 3) with the global identifier and an (up to now) empty set of *DEs*. In the later case, the *URL* identifies a *WE* with no global identifier and we assume that there is only one *DE* that describes it (line 18 in Algorithm 3).

In lines 11 and 19, we check whether the *WE* is already in the result-set. If it is, we

call the function *relevanceWE++*, which increments the count of the relevance that is associated with the *WE*. Otherwise, we *add* the *WE* to the result-set with a relevance count initiated to 1 (lines 14 and 22). At this point, if we are in the case of a *WE* with global identifier (i.e., with a *URI*), the second step of the search is initiated with the call to the *GetDEindex* function of the *DEindex* (see line 15). The object returned by the function is given to the *HandleDEAnswer* function, which then process it.

---

**Algorithm 3** Handler of the WE Answers
 

---

```

1: function HANDLEWEANSWER(weAnswer : WEAnswer, WEs : {⟨WE, relevance⟩})
2:   waitingTime : integer
3:   waitingTime := 5                                     ▷ parameterizable waiting time
4:   while weAnswer.isComplete = FALSE do
5:     WAITMS(waitingTime)                               ▷ specified in milliseconds
6:   end while
7:   if weAnswer.weAnsValues ≠ NULL then
8:     for all weAnsValue ∈ weAnswer.weAnsValues do     ▷ Parallel threads
9:       if ISURI(weAnsValue) then
10:        wEntity := ⟨weAnsValue,{⟩
11:        if wEntity ∈ WEs then
12:          RELEVANCEWE++(WEs, wEntity)
13:        else
14:          ADD(WEs,⟨wEntity,1⟩)
15:          HANDLEDEANSWER(GetDEindex(weAnsValue), WEs)
16:        end if
17:      else
18:        wEntity := ⟨NULL,{weAnsValue⟩}
19:        if wEntity ∈ WEs then
20:          RELEVANCEWE++(WEs, wEntity)
21:        else
22:          ADD(WEs, ⟨wEntity,1⟩)
23:        end if
24:      end if
25:    end for
26:  end if
27: end function

```

---

Finally, the Algorithm 4 shows how the values retrieved from the *DEindex* are handled. First, we wait until the answer is completed (see the loop from line 4 to line 6) and then the values are used to update the resultset. Note that the function *addDE2WE*

**Algorithm 4** Handler of the DE Answers

---

```

1: function HANDLEDEANSWER(deAnswer : DEAnswer, WEs : {⟨WE, relevance⟩})
2:   waitingTime : integer
3:   waitingTime := 5
4:   while deAnswer.isComplete = FALSE do
5:     WAITMS(waitingTime)
6:   end while
7:   ADDDE2WE(WEs, deAnswer.key, deAnswer.deAnsValues)
8: end function

```

---

takes the key (i.e., the *URI*) to identify, in the resultset, the *WE* that has to be updated. The values (i.e., the *URLs*) are then associated to such *WE* in order to complete the *QA*. We say that this function (called in line 7 in Algorithm 4) adds *DEs* to a given *WE* from a given set.

## 5.5 Summary

We presented an approach to build a distributed directory of entities in the DCM system that distinguishes between the notions of Digital Entity (DE) and Real World Entity (WE) in order to link local directories of different peers. The directory provides search services based on entity identifiers. In particular, we presented the algorithms for searching entities based on their names. We discussed the name matching problem that appears as a consequence of the *many-to-many* relation between names and (WEs). Then, we showed that, by its design, our approach deals with the problem of matching names inside the network (i.e., the first part of the name matching problem).

The data from peers are stored locally, only the identifiers and the links to the local representations are indexed. This infrastructure allows the implementation of access control mechanisms on the local representations in order to deal with privacy issues. At the same time, the changes made by peers in local representations, are available in the directory in a straightforward manner. Moreover, these features of the approach are independent from the specific underlying implementation of the indexes. In other words, the indexes can be stored in a centralized or distributed manner, while data will be still distributed.

## Chapter 6

# A description-based approach to search in the DCM system

Classifications are trees where links between nodes are commonly used to codify the fact that a node lower in the hierarchy describes a topic (and contains information about this topic) which is more specific than the topic of the node one level above. Some well known examples of these type of hierarchical structure are, email directories, file systems, web directories, and so on. In most of this examples classifications are usually intended to classify (i.e., organize) documents based on the topics that are included in their content. In our approach, we generalize this notions by looking at a document as a type of entity where topics described in its content and the content itself are modeled as attributes. Further, we use this generalized notion of classification and apply it to classify entities describing contacts in the DCM system.

In this chapter we see the DCM network (described in Chapter 4) as a network of peers where each peer stores various objects (i.e., entities) with certain characteristics that are of interest to its users. Moreover, we see these objects organized in tree-like hierarchies or classifications (i.e., classification of objects *CO* as we defined in Section 3.4.2). An abstract example of user generated classifications of several peers in the DCM system can be seen in Figure 6.1.

These classification hierarchies, also called lightweight ontologies [Giunchiglia and Zaihrayeu, 2008], are very common in knowledge organization systems as an effective and intuitive way to organize knowledge of humans according to their subjective view of the world [Giunchiglia and Zaihrayeu, 2008; Giunchiglia et al., Winter 2006]. On one hand,

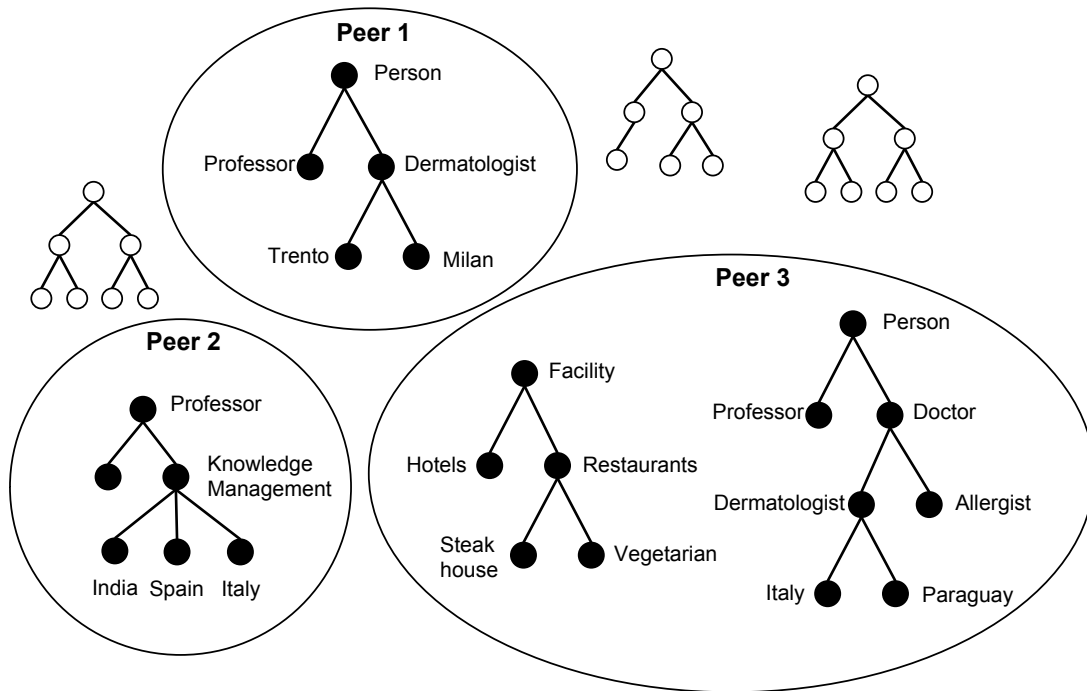


Figure 6.1: DCM Network of User-Generated Classifications

single nodes in the classification describe characteristics of entities in which the peer is interested. For example, the user of *Peer1* is interested in entities that refer to *Dermatologists* (i.e., entities with the profession dermatologist). On the other hand, the whole classification specifies the user interest profile. For example, the user of *Peer 3* is interested in doctors with various kinds of specializations, and the user of *Peer 2* is more interested in professors from different parts of the world that are specialized in knowledge management. Notice, that a user can build more than one classification in order to represent the diversity of its interests. For instance, the user of *Peer 3* classify facilities and people separately.

The goal of this chapter is to show how multiple classifications can be exploited to help the peer in finding contacts with certain characteristics that are of its interest. For example, a peer that is interested in finding a doctor that is an expert on some rare skin disease might benefit from finding peers who know many dermatologists. Moreover, even a peer that already knows many dermatologists may be interested in contacting not only those doctors that he knows, but also other doctors that are known by them or by any of their friends. In this work we also aim to avoid the imposition of a global structure for the classification of different peers in the network (i.e., a pre-established and shared ontology).



In the following sections, we present models that allow us to build distributed yellow pages for contacts of peers in the DCM network and propose an approach to search contacts based on their descriptions in a distributed manner. The models are based on the following key ideas:

1. The first is that the links connecting nodes inside a classification can be combined with links that codify semantic relations among classifications in order to form a *semantic overlay network* that can be exploited to perform a semantic search on nodes.
2. The second is that semantic search on nodes is implemented by flooding the links of the semantic overlay network in order to propagate the query to those peers having relevant nodes in their classifications. Differently from “normal” flooding as it happens, for instance, in Gnutella [Gnu], these links carry meaning and more precisely, codify the semantic relation (i.e. equivalence, more or less general) holding between any two nodes and allow, therefore, for “more informed” query propagation.
3. The third and last is that semantic search inside a peer is performed by extending to notions of concept search (*C-Search*) [Giunchiglia et al., 2009b] for entities (a semantics enabled information retrieval approach) thus exploiting as much as possible the advantages of a syntactic search and also a semantic search, as a function of the available background knowledge [Giunchiglia et al., 2006].

The rest of the chapter is organized as follows. We start by introducing a motivating example in Section 6.1. Then, we define a semantic overlay across the directories of peers from DCM network in Section 6.2. In Section 6.3, we discuss the problem of discovering links across classifications, while in Section 6.4 we show how these links can be exploited to perform search. Finally, in Section 6.5 we conclude the chapter with a summary.

**Acknowledgement.** *The work presented in this chapter was performed in collaboration with Uladzimir Kharkevich and Prof. Fausto Giunchiglia. Most of the content was published in [Giunchiglia et al., 2010] and [Giunchiglia et al., 2011].*

## 6.1 Motivating example

Let us consider the example of contact lists in devices from different peers of a network connecting friends among each other, their respective family members, etc. In Figure 6.2 we see again the contact list of each device as a local directory of people, where peers  $p_1$  and  $p_2$  are linked by their friendship; and peers  $p_2$  and  $p_3$  are linked by a family tie (i.e., they are cousins). In these directories, we also see that different peers can have

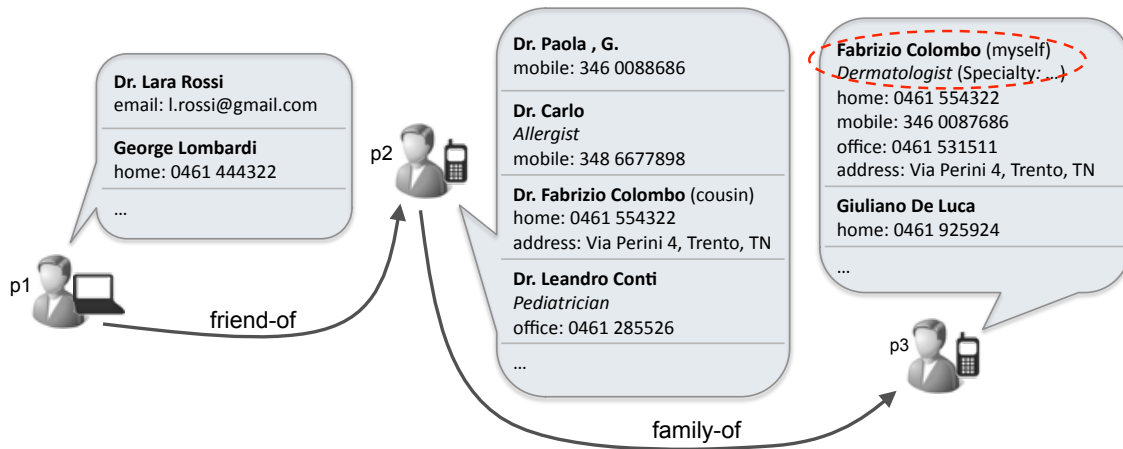


Figure 6.2: Example of Contacts with Similar Characteristics

information about people with similar characteristics (e.g., same profession, living at the same city, etc.). At the same time peers can have different level of details about the people in their contact lists.

For example, suppose that peer  $p_1$  needs to find a doctor that is specialized in a rare skin disease for his daughter. She has been recently diagnosed by her pediatrician but he is not specialized in that area and can not help her or guide the treatment. On the other hand,  $p_2$  (a friend of  $p_1$ ) is also interested in doctors and has the contact of a number of them. In  $p_2$  contact list we can find the contact information of his general doctor (Dr. Paola G.), his allergist (Dr. Carlo), his cousin (Dr. Fabrizio Colombo), and his son's pediatrician (Dr. Leandro Conti). Finally,  $p_3$  (the cousin of  $p_2$ ) is a doctor (Dr. Fabrizio Colombo) and he is actually dermatologist with a lot of experience in many rare diseases that affect the skin. However,  $p_2$  (Dr. Fabrizio's cousin) do not know this level of detail about Fabrizio's profession. For  $p_2$  contact list is enough to say that Fabrizio is a doctor.

Taking into consideration the original problem that  $p_1$  has (i.e., finding an specialist for his daughter), we can see that:

1. The contact information that  $p_1$  needs is in the network, in fact, the person he needs to contact is in the network.
2. The peers that can help to solve the problem might be already connected, but the lack of meaningful links prevents us from exploiting these connections in an efficient manner. In our example, this means that the fact that  $p_2$  is a friend of  $p_1$  and the cousin of  $p_3$  is not enough to realize that he can help solving  $p_1$ 's problem.
3. The relevance of the peers that can contribute to solve  $p_1$ 's information need is given by the characteristics of the contacts in their contact lists. In our example this

means that the important link between  $p_1$ ,  $p_2$ , and  $p_3$  in this case is that they have the contact information (with different levels of details) of doctors.

4. Finally, relevant peers might not even be aware of the fact that they can contribute to solve the problem. Most likely,  $p_2$  does not even know what is Fabrizio's specialty, he knows only that it is a dermatologist.

## 6.2 A semantic overlay for linking directories

In order to build a semantic overlay linking directories of different peers in the DCM network that have contacts with similar characteristics, we are mainly interested in the notion of classification of objects *COs* as defined in Section 3.4. A *CO* is aimed to classify entities describing contacts based on their characteristics as objects from the real world. It can be defined as,

$$CO = \langle \{n\}, \{e\}, \{l\} \rangle$$

where,

- $\{n\}$  is a set of nodes;
- $\{e\}$  is a set of edges on  $\{n\}$ ;
- $\{l\}$  is a set of labels expressed in natural language, such that for any node  $n \in \{n\}$  there is a label  $l \in \{l\}$  associated with  $n$ .

Moreover, the label of a node in a *CO* is used to describe a characteristic that is intended on entities contained by the node. An example of a user-generated classification is shown in Figure 6.3a.

The limitation of this definition is that its representation lacks of a level of formality that can allow automatic reasoning. In order to enable automatic reasoning about classifications and their content, we convert each *CO* into a Normal Formal Classification (NFC) [Giunchiglia et al., Winter 2006], which is a full-fledged lightweight ontology [Giunchiglia and Zaihrayeu, 2008]. It is important to note that the interpretation given to classifications and their elements (i.e., labels and nodes) is highly-dependent on the context in which they are used. The approach presented in [Giunchiglia et al., Winter 2006] to convert informal classifications into NFCs declares that its context is mainly the classification of documents and therefore the semantics of the classification is defined accordingly.

In our work, we intent to use classifications to organize (i.e., classify and search) real world entities based on different characteristics that describe them. Following the notions

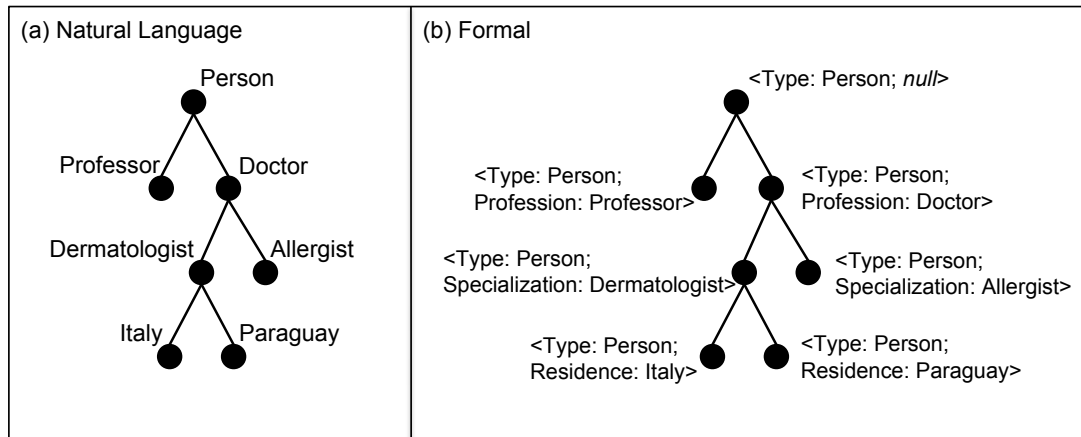


Figure 6.3: Classification

of entities discussed in Chapter 3, we actually consider document as a type of entity where the title, topics in its content and the content itself (among other characteristics) can be represented as attributes. As a consequence, the process to follow for the conversion of a *CO* into a *NFC* is similar to the one discussed in [Giunchiglia et al., Winter 2006; Zaihrayeu et al., 2007] but the details regarding the semantics of classifications are re-defined as follows:

**Labels.** A label can describe a type of entity or a property (i.e., attribute). The meaning of the label in the classification is the set of entities that are of the given type or have the given attribute. For example, the semantics of the label “Person” is the set of entities (i.e.,  $\{DE\}$ ) referring to persons (i.e., are instances of the corresponding *ET* person). On the other hand, the semantics of the label “lives in Italy” is the set of entities having an attributes *A* stating “Place of residence: Italy”. In the case of labels, their semantics is completely captured by the label itself.

**Nodes.** A node represent a complex constraint, similar to the specification of a search request based on an entity description. This complex constraint is formed by combining the constraint in the label of the node with the constraints in the labels of all the nodes that are in the path to the root of the classification. The meaning of a node in the classification is then the set of entities (i.e.,  $\{DE\}$ ) that satisfy the complex constraint represented by the node. For example, the semantics of the node “Italy” in the classification of Figure 6.3(a) is the set of entities that refer to *persons*, who are a *doctors*, are specialized in *dermatology* and live in *Italy*. In the case of nodes, their semantics is defined by the labels of the nodes that are in the path to the root of the classification.

**Classification.** The semantics of all the nodes in the classification and the set of entities that classified by them define the semantics of the classification itself. As defined in [Giunchiglia et al., Winter 2006], the semantics of the whole classification (in the most general case) is defined by the nodes' labels, the structure of the classification and the classification algorithm that is used.

To convert a classification into a NFC, the models for the representation of knowledge presented in Chapter 3 and the *background knowledge (BK)* [Giunchiglia et al., 2006] of the peer are used. The BK represents the knowledge of the peer about concepts and their relationships over a specific domain or a limited set of domains. We assume the BK follows the DERA methodology [Giunchiglia et al., 2014] for the representation of domains.

Then, the conversion is performed by executing the following steps:

- **Step 1:** Translate the label of each node in a classification to a pair  $\langle ET, A \rangle$ . In order to translate each label of a classification node  $n_i$  into a constraint, the label has to be mapped to a concept  $C_{ET}$  of a corresponding entity type  $ET^{n_i}$  or to an attribute  $A^{n_i}$ . If  $ET^{n_i}$  is not specified in the label and  $n_i$  is the root node in the classification, then  $ET^{n_i}$  is set to a generic entity type "Thing". If  $n_i$  is not the root node, then  $ET^{n_i}$  is set as the entity type of  $n_i$ 's parent node (i.e.,  $n_i$  inherits the  $ET$  of its parent node in the classification). On the other hand, if  $ET^{n_i}$  is specified in the label and  $n_i$  is not the root node, then  $ET^{n_i}$  must be compatible with the  $ET$  of its parent node. In fact, the entity type  $ET^{n_i}$  of a node  $n_i$  must be equivalent or more specific than the type of its parent node, otherwise it is invalid (i.e., the classification will not be valid). Finally, if the label does not specifies an attribute, then  $A^{n_i}$  is assigned with *null*. The classification that results from computing the pair  $\langle ET^{n_i}, A^{n_i} \rangle$  for all the nodes  $n_i$  in the classification, is called Formal Classification (FC). The part (b) of the Figure 6.3 shows an example of a FC created from the classification in the part (a) of the same figure.
- **Step 2:** Compute the meanings of nodes in the classification. In order to encode meaning of nodes in the classification, the notion of *concept at node* from [Giunchiglia et al., 2007a] is applied. As a consequence, the meaning  $M^n$  of a node  $n$  is defined as the conjunction of the meanings of the nodes that are in the path to the root from  $n$ . Namely,

$$M^n = \langle ET, A \rangle^{n_1} \sqcap \langle ET, A \rangle^{n_2} \sqcap \dots \sqcap \langle ET, A \rangle^{n_i} = \langle ET^n, \{A\}^n \rangle$$

where  $ET^n$  is the most specific entity type among all  $ET$ s in  $\{ET^{n_1}, ET^{n_2}, \dots, ET^{n_i}\}$  and  $\{A\}^n = \{A^{n_1}, A^{n_2}, \dots, A^{n_i}\}$ . The resulting classification, in which meanings of nodes  $M^n$  are computed for all the nodes in the classification, is a NFC.

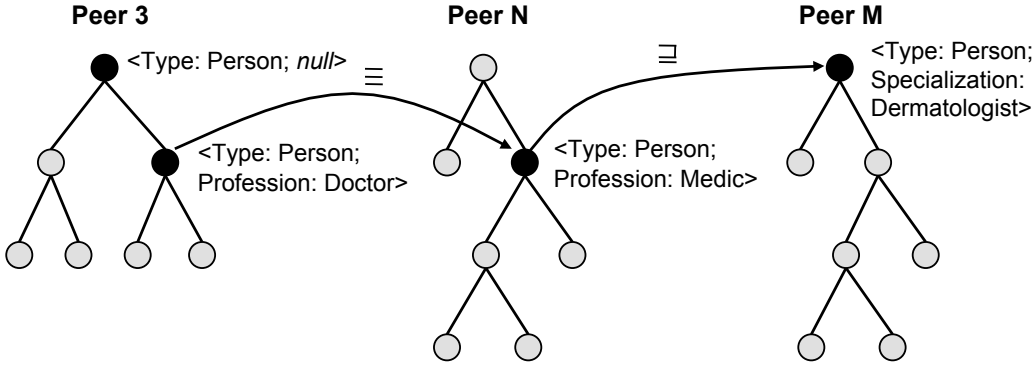


Figure 6.4: A Semantic Overlay Network

After a NFC has been created, entities (i.e., *DEs*) can be automatically classified to nodes in the classification by using the get-specific principle [Giunchiglia et al., 2007b]. In order to follow the get-specific principle, an entity *DE* should be classified under those classification nodes that (i) have meanings that are more general than the entities themselves, and (ii) have no child nodes that can describe the entity more specifically. Formally, a set of entities  $S(n)$  classified in a sub-tree of node  $n$  is defined as follows:

$$S(n) = \{DE \mid DE \sqsubseteq M^n\} \quad (6.1)$$

If node  $n$  has a set of child nodes  $Chilids(n)$ , then a set of entities  $D(n)$  classified to node  $n$  is defined as follows:

$$D(n) = S(n) - \bigcup_{n_i \in Chilids(n)} S(n_i) \quad (6.2)$$

To make the peers in the CDM Network able to reason about the content of each other, semantic links, expressed in the C-OWL language [Bouquet et al., 2004], can be created between related nodes in their classifications. C-OWL envisions a wide range of possible semantic relations that can hold between related nodes in different classifications. For the goals of this approach we concentrate on the following links: (i) equivalence links (represented as  $A \xrightarrow{\equiv} B$ ), (ii) more general links ( $A \xrightarrow{\sqsupseteq} B$ ), and (iii) more specific links ( $A \xrightarrow{\sqsubseteq} B$ ). For example, in Figure 6.4, the link between nodes with labels “Medic” and “Dermatologist” is used to specify that the meaning of the former node ( $\langle Type : Person; Profession : Medic \rangle$ ) is more general than the meaning of the latter node ( $\langle Type : Person; Specialization : Dermatologist \rangle$ ). Note that, according to Equation 6.1, all the entities which are classified in the subtree of the latter node can be classified also in the subtree of the former node.

The set of links which connect nodes inside a classification plus C-OWL links across

classifications constitute a semantic overlay network which can be built on top of any underlying set of peers and their physical connections.

### 6.3 Semantic link discovery

When a new peer joins the network, there are no semantic links connecting the nodes in the classifications of this peer with the nodes in classifications of other peers in the network. Another example where links can be missing is when a new node is created in a classification, e.g. because the peer became interested in a new property or characteristic. In the following, it is discussed how new semantic links can be discovered in these and other similar situations.

If the two classifications which need to be connected are known in advance, then semantic links between these classifications can be created manually by the users or they can be automatically computed by using semantic matching (S-Match) [Giunchiglia et al., 2007a] approach to compare the meaning of nodes from the given classifications.

On the other hand, when the relevant classifications are not known, the initial links that the peer will have to other peers in the DCM network will be given by its contact list. The contacts from this list that are also users of the DCM system, and therefore are also peers in the DCM network, can be identified. Then, S-Match can be executed between the given classification (from the local peer) and all the classifications from the identified known peers. The problem with this approach is that the number of peers can be big and, therefore, running S-Match for all the possible combinations of classifications can become unfeasible.

In order to reduce the number of peers to contact for discovering links, the user can select only a subset of the known peers to run S-Match with their classifications. The selection of the subset can be done in two ways:

- (i) by selecting a node from its classification of subjects  $CS$ , for example the peer may decide to constraint the discovery of relevant links to the group of “close friends”; and
- (ii) by selecting a node from its classification of objects  $CO$ , for example the peer may decide to constraint the discovery of relevant links to a group of contacts with certain characteristics (e.g., contacts who are doctors, contacts that live in Italy, etc.).

The limitation of this approach is that the peer is only able to build links to peers that are already known. This means that in the cases in which there are not known peers, the approach will not be applicable. Moreover, the peer will not be able to discover links

to peers that contain relevant information but are not known for him (i.e., are not in the peers' contact list).

Alternatively, we propose to use the distributed entity directory that was proposed in Chapter 5. In particular, searching an entity by name (i.e., identifiers) in this directory allows the peer to find other peers in the network having information about such entity. When this entity refers to an object (physical or abstract) from the real world in which the local peer is interested, the search result can be used to identify peers with a common interest. Let us consider again the example of a peer trying to find contacts of doctors that are experts in a disease called *mm*. The peer can, first, query the entity directory in order to get the set  $\{SURL\}$  corresponding to other peers that store information about *mm*. Then, for each  $SURL_i \in \{SURL\}$  the peer  $P_i$  owner of it can be identified. Under the assumption that  $P_i$  is also interested in the disease and may know doctors that are specialized on treating it, S-Match can be run to compute the semantic links between the classifications of the local peer and  $P_i$ .

## 6.4 Algorithms

The actual implementation and evaluation of the approach presented in this Chapter was done based on entities of the type *Document*. Therefore, the algorithms described in this section present the search process applied to documents.

Documents are considered a particular type of entity where the main attribute encoding its meaning is the document "subject". We assume that the constraints represented by the nodes in the classification and by the query will mainly refer to this attribute, which encodes the meaning of the content, called  $C^d$  (i.e., the complex concept at the document). With a similar reasoning, the meaning at the node  $M^n$  is called  $C^n$  to represent a complex concept encoded by the node. Additionally, the entity type *ET* is the same for the whole classification (namely, documents) and can be omitted.

The problem of searching documents based on the description of its content (or subject) is then considered as the process of finding documents which are semantically related to the user information needs and which are stored in a document collection distributed among all the peers in the network. When a user searches for documents, she, first, selects a node  $n$  in the classification. The root node of the classification serves as a default node for search if no other node is selected. Second, the user issues the query  $q$ . The query is converted into an expression in  $L^C$  using the same technique used for creation of concept at nodes. Let  $C^n$  be a complex concept at node  $n$  and  $C^q$  be a complex concept extracted from query  $q$ . The goal of the search algorithm is to find documents  $d$  stored in the network, such that, concept of document  $C^d$  is more specific than the



concept at node  $C^n$  and there exists a concept  $C$  described in  $d$  which is more specific than the query concept  $C^q$ . Formally a query answer  $A(C^n, C^q)$  is defined as follows:

$$A(C^n, C^q) = \{d \mid C^d \sqsubseteq C^n \text{ and } \exists C \in d, \text{ s.t., } C \sqsubseteq C^q\} \quad (6.3)$$

The problem of a semantic search in the P2P network can be decomposed into three subproblems:

1. Identifying semantically relevant peers.
2. Searching inside relevant peers.
3. Aggregation of the search results.

Let us consider these three subproblems in detail.

#### 6.4.1 Identifying semantically relevant peers

A peer is considered to be semantically relevant to a query if there are nodes in the peer's classification which are relevant to the node selected by the user. Moreover, some of the documents classified in these nodes should be relevant to the user query. In order to store the information about potentially relevant peers, the initiator peer  $p_I$  creates a peer information list, defined as follows:

$$peerinfos(n) = [\langle p, nodeinfos(p, n), stat \rangle],$$

where  $p$  is a relevant peer,  $stat$  is a status of  $p$ :  $NQ$  - peer is not queried,  $QU$  - peer is already queried, or  $RE$  - response is returned, and  $nodeinfos(p, n)$  is a list which stores information about nodes  $n'$  from peer  $p$  which are semantically related to node  $n$  plus a set  $\{l\}$  of incoming links  $l$  for node  $n'$ :

$$nodeinfos(p, n) = [\langle n', \{l\} \rangle]$$

Initially,  $peerinfos(n)$  contains information only about the peer  $p_I$ :  $peerinfos(n) = [\langle p_I, [\langle n, \emptyset \rangle], NQ \rangle]$ . After  $peerinfos(n)$  is initialized,  $p_I$  starts an infinite loop, where a single iteration is performed as follows:

- Select the first (if any) peer info  $\langle p, nodeinfos(p, n), stat \rangle$  from  $peerinfos(n)$ , such that,  $stat = NQ$ .
- If there are no such peer infos, wait until the  $peerinfos(n)$  list is modified and perform the previous step again.
- Form a query request  $\langle C^n, C^q \rangle$  and submit it to peer  $p$ .

- Change the status of peer  $p$  to  $stat = QU$ .

When peer  $p$  receives the query request, it locally computes a set of links  $L$ , such that, each of the target nodes has a complex concept which is more specific than the complex concept  $C^n$ . Note that, at the same time, the concept of the target node in a link can be equivalent, more specific, or more general than the concept of the source node. All the links in  $L$  are sent back to the initiator peer  $p_I$ <sup>1</sup>. Peer  $p_I$  updates the  $peerinfos(n)$  list by using information from links in  $L$ .  $peerinfos(n)$  list is then sorted in a decreasing order of the number of incoming links. It is assumed that, in this way, peers are queried in a decreasing order of their importance.

Every node  $n'$  in  $nodeinfos(p, n)$  has only the documents with complex document concepts  $C^d$  which are more specific than the complex concept  $C^n$ . This is because, from  $C^d \sqsubseteq C^{n'}$  and  $C^{n'} \sqsubseteq C^n$ , it follows that  $C^d \sqsubseteq C^n$ . In spite of this, links between nodes do not describe all complex concepts  $C$ , which can be found in the documents classified to these nodes. Therefore, it can be the case that node  $n'$  has no documents which are relevant to the query concept  $C^q$ . The portion of such nodes can increase when a concept  $C^n$  becomes more and more general. In the worst case, i.e. when  $C^n \equiv \top$ , all the nodes which can be reached by all the links can be added to  $nodeinfos(p, n)$  and all the corresponding peers  $p$  can be queried. *Semantic Flooding*, in this case, is reduced to normal flooding and, in general, can be very inefficient.

In order to implement a more efficient selection of semantically relevant peers, it is proposed in this paper to use a measure of semantic similarity  $SS(C^{n'}, C^q)$  between complex concepts at node  $C^{n'}$  and the complex query concept  $C^q$  (see, for example, [Borgida et al., 2005]). As a simple example of a semantic similarity measure  $SS(C^{n'}, C^q)$ , let us consider the following measure:

$$SS(C^{n'}, C^q) = \begin{cases} 1 & \text{if } C^{n'} \sqsubseteq C^q \\ 0 & \text{otherwise} \end{cases}$$

Observe that for  $n'$  with  $SS(C^{n'}, C^q) = 1$ , concepts  $C^d$ , for all the documents classified to  $n'$ , are more specific than query concept  $C^q$ . It is because, from  $C^d \sqsubseteq C^{n'}$  and  $C^{n'} \sqsubseteq C^q$ , it follows that  $C^d \sqsubseteq C^q$ . Given that  $C^d$  is built from concepts  $C$  found in the document  $d$ , it is likely that  $d$  is relevant to query  $q$ . Note that the following measure of semantic similarity is actually used:

$$SS(C^{n'}, C^q) = \sum_{A^q \in C^q} \frac{1}{10 \min_{A^{n'} \in C^{n'}} (dist(A^{n'}, A^q))} \quad (6.4)$$

<sup>1</sup>Note that by doing this, the search process by itself can be used to discover new links.

where  $A^q$  is an atomic concept in the concept at query  $C^q$ ,  $A^{n'}$  is an atomic concept in the concept at node  $C^{n'}$  and  $dist(A^{n'}, A^q)$  is the distance between the two atomic concepts in the background knowledge (BK) of the user. The distance is measured by the minimum number of edges that connect the two atomic concepts in the hierarchy of concepts. Now, instead of just the number of incoming links,  $peerinfos(n)$  list is sorted in a decreasing order of the peer scores computed as a sum of node scores  $score(n', q)$ . A node score  $score(n', q)$  is computed as follows:

$$score(n', q) = (N_l + 1) * (SS(C^{n'}, C^m) + SS(C^{n'}, C^q)), \quad (6.5)$$

where,  $N_l$  is a number of incoming links for node  $n'$ . Note that only links for those nodes which are relevant for current search request are considered while sorting  $peerinfos(n)$ .

#### 6.4.2 Searching inside a relevant peer

On receiving a search request  $\langle C^n, C^q \rangle$ , peer  $p$  performs search for relevant documents in a local document collection by using the *C-Search* [Giunchiglia et al., 2009b]. C-Search is an IR approach which is based on retrieval models and data structures of syntactic search, but which searches for complex concepts  $C$  rather than words  $W$ . The key idea is that syntactic matching of words is extended to semantic matching [Giunchiglia et al., 2007a] of complex concepts, where semantic matching is implemented by using positional inverted index. The output of C-Search is a list of documents ordered by their relevance to the query. A list of top  $k$  ranked documents, nodes to which the documents are classified, and the information about frequencies of atomic concepts  $A \in C^q$  in the retrieved documents and in the whole local document collection are sent back to the initiator peer  $p_I$ . Peer  $p_I$  updates the  $peerinfos(n)$ , i.e. the status of  $p$  is changed to  $stat = RE$ . In order to store the information about the relevant documents, the initiator peer  $p_I$  uses a document information list:

$$docinfos(q) = [\langle d, n', [\langle A, tf(A, d) \rangle] \rangle],$$

where  $d$  is a document which is classified to node  $n'$ , and which is also relevant to query  $q$ ,  $tf(A, d)$  is a number which represents the importance of document  $d$  to an atomic concept  $A \in C^q$ . Moreover, in order to store the global information about the importance of atomic concepts  $A \in C^q$ ,  $p_I$  uses term information lists for all  $A$ :

$$terminfos(A) = [\langle p, numDocs_p, docFreq_p(A) \rangle],$$

where  $docFreq_p(A)$  is a number which represents the frequency of atomic concept  $A$  in the document collection of peer  $p$  which has  $numDocs_p$  documents in total. When receiving new results, a peer  $p_I$  updates the  $docinfos(q)$  and  $terminfos(A)$  tables.

The search process terminates when: (i) the required number (e.g. 100) of documents is retrieved; or (ii) all the relevant documents are retrieved; or (iii) the search time exceeds some predefined limits; or (iv) the user terminates the process.

### 6.4.3 Aggregation of search results

After the search process is terminated, the peer  $p_I$  merges query answers from different peers into a single query answer. First, the cosine similarity  $\cos(d, q)$  from the vector space model is computed for every retrieved document  $d$ . Terms are weighted by the *tf-idf* weight measure used in *Lucene* [Luc], where an inverse document frequency  $\text{idf}(A)$  is estimated as follows:

$$\text{idf}(A) = 1 + \log\left(\frac{\text{numDocs}}{\text{docFreq}(A) + 1}\right),$$

where  $\text{numDocs}$  is computed as a sum of all the  $\text{numDocs}_p$ , and  $\text{docFreq}(A)$  is computed as a sum of all the  $\text{docFreq}_p(A)$ . Second, the cosine similarity  $\cos(d, q)$  is combined with the score  $\text{score}(n, q)$  of the node  $n$  to which the document is classified in order to compute the final score of the document  $\text{score}(d, q)$ :

$$\text{score}(d, q) = \text{score}(n, q) + \cos(d, q)$$

Finally, documents are ordered according to the relevance score and presented to the user in the decreasing order of relevance.

## 6.5 Summary

In this chapter it was shown how the notion of classification of objects COs, as defined in Section 3.4, can be exploited to build a semantic overlay linking directories of different peers in the DCM network. The set of links which connect nodes inside a classification plus C-OWL links across classifications constitute a semantic overlay network which can be built on top of any underlying set of peers and their physical connections. Thus allowing peers to semantically search contacts that are distributed in the DCM network and have certain characteristics.

In order to build a semantic overlay, we discussed how new semantic links can be automatically computed by using semantic matching (S-Match) approach between two known classifications. This was shown to be particularly relevant when a new peer joins the network or when a new node is created in a classification. When relevant classifications are not known the user can select a subset of the known peers, by selecting a node from its classification of subjects CS or from its classification of objects CO, and run S-Match

with their classifications. As another alternative we proposed to use the distributed entity directory to find other peers in the network having relevant information but that are not in the local contact list, and run S-Match with their classifications.

Next, we presented an implementation of the approach that is based on entities of the type *Document* and decomposes the problem into three subproblems. The first, identifying semantically relevant peers, defined as those having nodes in their classifications which are relevant to the search request. The second, searching inside relevant peers, which is done by using *C-Search* [Giunchiglia et al., 2009b]. And the third, aggregation of the search results, which includes merging query answers from different peers into a single query answer and computing a relevance score for each answer. The relevance score in the implementation presented is done using cosine similarity and tf-idf weight measure used in *Lucene* [Luc].

Similar to the entity directory, also in this approach the data from peers are stored locally. The links to classifications from other peers are also stored locally. This means that the modifications made by peers to the local representation of their contacts are available in the semantic overlay network also in a straightforward manner. Moreover, this infrastructure can also benefit from the implementation of access control mechanisms on the local representations in order to deal with privacy issues.



## Chapter 7

# The distributed contact management (DCM) system

In previous chapters we have presented the proposed reference architecture (Chapter 4) for the DCM system, an approach to search entities based on their identifiers (called names in general) (Chapter 5) and another approach to search entities based on their descriptions (Chapter 6). In this chapter we present the Distribute Contact Management (DCM) System by framing the two search approaches into the architecture, defining additional application specific notions, and presenting an extensive description of usage scenarios.

In order to frame the search approaches in the architecture we will describe how each type of search is used in the system, which components affect, when is used and what subproblems helps to solve.

Additional application specific notions include: (i) First, the definition of *Presentation Cards* (PCs) as a generalization of the usage of business cards. A PC constitute an abstraction of the contact profile for a peer, who can create many PCs including different information, in order to use them in different contexts. (ii) Second the definition of *DCM Users* in terms of the data structure used to represent them in the system.

Finally, the description of usage scenarios presents the dynamics of the system in more details. The scenarios are grouped according to the three high-level use cases that we identified in Chapter 4 (Section 4.4) and are meant to illustrate the main functionalities and features of the system. We describe them in terms of user actions, which are then mapped onto the set of system actions needed to support the scenario.

## 7.1 Presentation Cards

To support the exchanging of contact information between peers, we introduce the notion of *Presentation Card (PC)*. A peer in the DCM system can create one or more presentation card(s) to be shared in different contexts (i.e., personal, business, party, advertisement, etc). We use these cards as generalizations of the notion of business cards which were extensively used for many years in formal introductions of individual and companies.

Note that with the arrival of electronic communication mechanisms, the kind of information included in business cards evolved to include, for example, e-mail addresses, websites, and social media addresses (e.g., Facebook, LinkedIn, Twitter, etc.). Further, with the evolution of technology, the notion of business cards evolved to electronic formats, which resulted in the definition of new standards for electronic business cards (e.g., vCard<sup>1</sup>, xCard<sup>2</sup>, etc.) to allow representing and exchanging contact information using different types of information systems. In the DCM system, we focus the attention on the model that is behind the creation of presentation cards, their purpose and how to manage them as a mechanism to manage the peers identities within the system.

Within the system the characteristic of peers are represented using the notion of entities (as defined in Chapter 3, Section 3.1). Consequently, we define a **Presentation Card (PC)** to denote a profile created to present one view (perspective) of a particular entity *DE*. Formally, it is defined as the tuple

$$PC = \langle Name_{PC}, SURL, SURI, \{N\}, ET, \{A\}, PP \rangle$$

where,

- $Name_{PC}$  denotes a label (i.e., arbitrary set of characters and numbers) used by the peer to recognize and uniquely identify the card. The peers could use this name as a reminder of the context for which the card was intended, or of the attributes that the card includes. The name of the card is never revealed to other peers;
- $SURL$  denotes the unique identifier of the entity instance *DE* to which this card is presenting;
- $SURI$  denotes the unique identifier of the real world entity *WE* to which the card refers;
- $\{N\}$  denotes the subset of names from the corresponding *DE* that are included in the card;

---

<sup>1</sup>RFC 6350 - vCard Format Specification (<http://tools.ietf.org/html/rfc6350>)

<sup>2</sup>RFC 6351 - xCard: vCard XML Representation (<http://tools.ietf.org/html/rfc6351>)



- $ET$  represents the entity type of the corresponding  $DE$  (i.e., the type of entity that the card is presenting);
- $\{A\}$  denotes the subset of the entity attributes (entity type dependent) of the corresponding  $DE$  that are included in the card;
- $PP$  denotes a set of privacy preferences applicable to the information in the presentation cards.

Furthermore,  $PCs$  exist only in the context of the digital representation of an entity  $DE$  (i.e., the context of the entity they describe). As such, the definition of  $DEs$  that was introduced in Chapter 3 is extended as follows,

$$DE = \langle SURL, SURI, \{N\}, ET, \{A\}, \{PC\} \rangle$$

where,  $SURL$  is unique identifier of the  $DE$ ;  $SURI$  is a unique identifier of the real world entity that  $DE$  is describing;  $\{N\}$  is a set of strings representing names used by the corresponding description  $DE$  to identify a real world entity;  $ET$  is the entity type among those defined in the knowledge schema of the system;  $\{A\}$  is a non-empty set of attributes describing the characteristics of the entity; and  $\{PC\}$  is a set of presentation cards associated to  $DE$ .

Additionally, we define the following set of rules for the above  $DE$  definition:

1. The content of a presentation card can only be modified by its owner peer.
2. The owner of a presentation card  $PC_i$  is defined as the peer that is acting in the system on behalf of an entity from the real world described by  $DE$ , such that  $PC_i \in DE.\{PC\}$  and  $PC_i.SURL = DE.SURL$
3. The  $PC_i.SURI$  should always correspond to  $DE.SURI$  for every  $PC_i \in DE.\{PC\}$
4. The  $PC_i.ET$  should always correspond to  $DE.ET$  for every  $PC_i \in DE.\{PC\}$
5. Conceptually, the set of attributed that describe a  $DE$  is now defined as the union between all the attributes  $DE.\{A\}$  from the digital entity and all the attributes defined by its presentation cards. Formally, it is defined as the set,

$$\{DE.\{A\} \cup \bigcup_{PC_i \in DE.\{PC\}} PC_i.\{A\}\}$$

there should be no difference for the peer, independently from where or how they are stored.

6. In turn, the previous has the following consequence for  $PC_i \in DE.\{PC\}$ :

- If  $DE$  is the owner of  $PC_i$ , we say that  $PC_i$  is constraint to contain only attributes that are in  $DE.\{A\}$  (i.e., a subset of  $DE$ 's original attribute).
- If  $DE$  is NOT the owner of  $PC_i$ , we say that  $PC_i$  can conceptually extend the attribute set of  $DE$ .

The Figure 7.1 shows an example of the representation of a person (i.e., “Mario Rossi”) that has defined two presentation cards, one for business related purposes (called “work”) and the other for a more informal context (called “friends”).

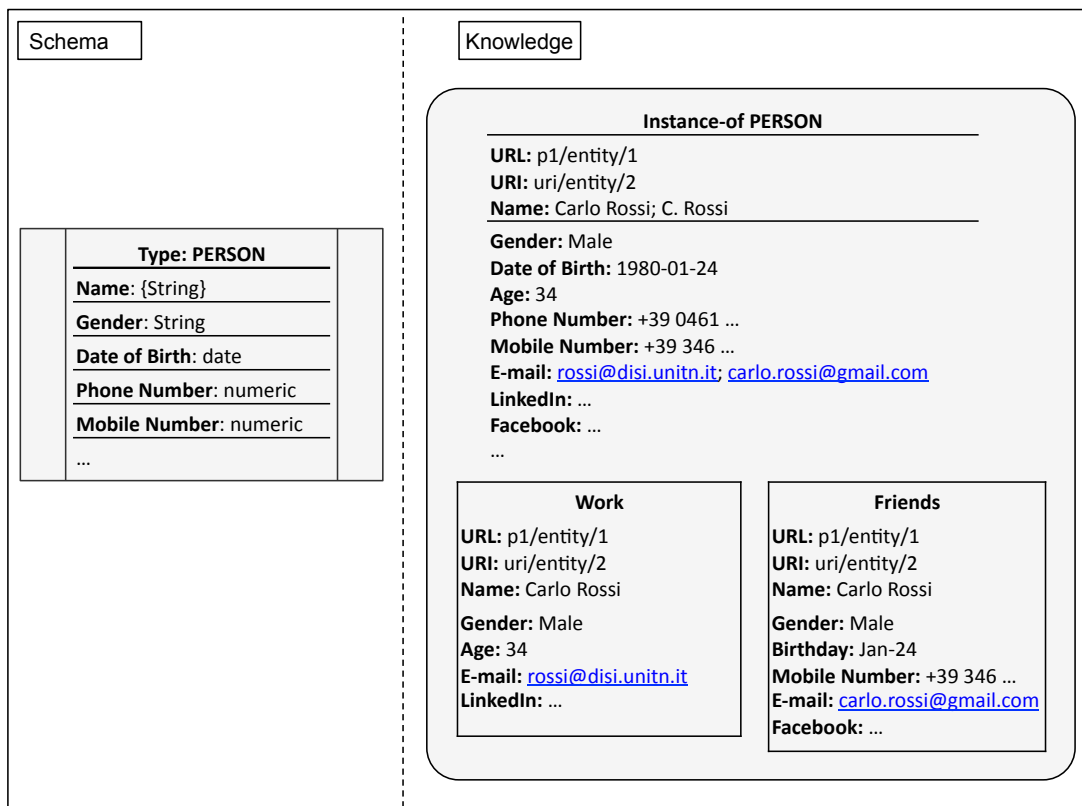


Figure 7.1: Example of Person Presentation Cards

An important feature of presentation cards is the inclusion of privacy preferences. The aim is to encode within the cards the privacy policies that apply to the information that is in it. When the user exchange cards with other users, these policies should travel with the data in the form of sticky policies as defined in [PrimeLife, 2011]. However, the representation of privacy policies and the implementation of privacy enhancing technologies that can enforce them is out of the scope of this thesis and are left as future work. We include the notion of privacy preferences here and we refer to them again in the description of scenarios in order to show that the DCM system proposed here provide a

privacy-friendly design that can facilitate the adoption of appropriate privacy preserving tools in the future.

## 7.2 DCM Users

When a peer creates an account in the DCM platform, he/she becomes a **DCM User**. Formally, we define a DCM User as the tuple  $U = \langle uName^*, password^*, ET^*, DE \rangle$  where:

- $uName^*$  is a unique user identifier.
- $password^*$  is the secret code that allows the user the authentication.
- $ET^*$  is the entity type of the user and it has to be the same as the ET of the corresponding DE (when this is given).
- $DE$  is the digital entity that corresponds to the user, it is optional (i.e., the user might decide to remain anonymous) and the same entity can create more than one user.

In order to become DCM Users, peers are required to provide a minimal set of information that includes only mandatory elements, which are marked with (\*), from the tuple defined above. User interactions with the application are performed on behalf of an entity from the real world. If such entity is linked to the user (i.e., if  $DE$  is not NULL), the user becomes identifiable in the real world. Otherwise they are only identifiable through a pseudonym in the platform, we call these anonymous<sup>3</sup> users. In the context of the architecture presented in Chapter 4, DCM users are stored at the corresponding  $UB$  (at the portal and/or at the peer), however, the  $UB$  stores only the references (i.e., a link or an identifier) to the  $ET$  and  $DE$ , which are actually defined at the corresponding  $EB$ .

The user owns a contact list that digitally represent entities which are known by the user and for which the user has some contact information. We distinguish between three **types of contacts**:

- A *NON-USER*, represents an entity from the real world that is not a DCM user;
- A *USER*, represents an entity from the real world that is a DCM user; and
- A *SYNCH-USER*, represents a real world entity, who is a user of the DCM application and for which the system has contact information that is maintained in synchrony (i.e., a  $DE$  that has  $PCs$  associated to it).

---

<sup>3</sup>Although they are not really anonymous in the strict sense of the term.

## 7.3 Usage scenarios

### 7.3.1 Initialization Scenarios

The scenarios called of *initialization* correspond to the first interactions that peers have with the system, those that happen during the initialization phase, when the peer starts using the DCM system.

#### 7.3.1.1 Creating an user account

**Personas:**

*Alice* is a researcher, she heard about the features of the DCM app and decided to try it out in her smart-phone.

**Description:**

- Alice wishes to download the DCM app.
- In order to download the DCM app, Alice has to be registered as a DCM user.
- After the DCM portal let her know that she does not need to give personal information for the creation of the account, she decides to register.
- Alice creates a DCM user account at the DCM portal.
- Then, she downloads the application and installs it in her smart-phone.

**System Actions:**

- The DCM portal requires a new peer to create a user account.
- For the creation of the account, the portal presents the peer a form where she can input a user name *uName* and *password*. The form also shows to the peer a list of entity types and requires her to select the one that will be associated with the new user account (e.g., Person, Facility, Organization, etc.).
- The system stores the tuple of  $\langle uName, password, ET, NULL \rangle$  in the portal's UB.
- Notice that the entity DE of the peers's user is NULL up to now and the system knows the peer only by a pseudonym (i.e., its user name).
- The DCM portal now allows the peer to login and download the application.

### 7.3.1.2 Starting to use the DCM app

**Personas:**

*Alice* is a researcher, she heard about the features of the DCM app. After creating an user account, she has (recently) installed the DCM app on her smart-phone.

**Description:**

- After installation, Alice is using the DCM app for the first time.
- The DCM app shows a login interface that requests Alice to provide her user name and password.
- A note in the main screen provides an explanation of the purpose for which such information is requested. The DCM app notifies Alice that as part of the initialization of the app, her login information will be send to the DCM portal in order to authenticate the account.
- After authentication, she is invited to define one or more *Presentation Cards* in order to pre-define different profiles that she might want to share with others.
- Alice notices that she can use also the information from her other existing accounts (e.g., Facebook, Skype, Whatsapp, etc) to automatically fill in her information and her presentation cards.
- Next, Alice is invited to initialize her contact list.
- Alice is then invited to join the DCM network. This means connecting with the network of DCM users where she can publish and search information. In fact, she learns that she can decide to work online or offline at any point.
- Initially, all of her contacts are marked as *NON-USER* contacts.
- She then receives a notification regarding a contact matching process that will be initiated in the background over the local contacts, in order to find which of them are actually contacts of type *USER* (i.e., contacts that are also DCM users). However, the result of this process will be seen in future interactions with the app.
- Alice can now start using the application.

**System Actions:**

- The DCM app requires registration of the user when it is started for the first time. It also requires internet connection in order to verify and authenticate the user.

- After authenticating the user, the user information is stored locally in the peer's UB.
- If the peer selects the “create card” option, then the Scenario 7.3.1.3 is started.
- If the peer selects the “initialize contact list” option, then the app shows a list with the available options that include: importing contacts (Scenario 7.3.1.4) and input contacts manually (Scenario 7.3.1.5) by typing their information.
- Next, the DCM app starts a contact matching process in the background, on the local contacts. This process requires querying the DCM portal in order to find if there are local contacts of the peer that match with existing DCM users. The DCM app will update the type of the contacts that match with existing users. The results of this step will be seen in future interactions with the app.

### **7.3.1.3 Creating presentation cards**

#### **Personas:**

*Alice* is a researcher and a new user of the DCM app.

*John* is a researcher, he is a new colleague of Alice.

*Enrico* is also a researcher, he is a colleague and a good friend of Alice.

#### **Description:**

- Alice is interested in sharing some contact information with her new colleague John. She would like also to share more personal contact information (like phone number and social networks accounts) with close friends like Enrico and all her contact information with her family.
- Alice decides to create 3 different presentation cards to share with these different groups of people.
- Alice first creates a “work” card to share it with John, she adds her university email address, office phone number and the url of her linkedln profile.
- She assigns public permissions to her work presentation card (containing only basic information). This means that the application will give access to Alice's work contact information to any requesting peer.
- Then, she creates a “close friends” card (to share it with Enrico and other friends) with her cellphone number, personal email address and her different social network profiles.

- Finally, Alice defines a “family” card, where she adds all her personal and social contact information, including her home telephone number and home address.
- She also assigns permissions so family members can access her family presentation card.

**System Actions:**

For each presentation card,

- The DCM app creates the card for the DCM user that is currently logged in.
- First, the app retrieves the user entity  $DE_u$  (i.e., the entity associated to the user in the peer’s UB). If the user entity is null, then the app creates a new entity instance, stores it in the peer’s EB and associates it to the user in the peers’s UB.
- Next, the peer is requested to input a name for the card.
- Then, the app presents a form to the peer (as a template) that allows the definition (or selection) of the attributes to be included in the card. Notice that:
  - the form is *ET* dependent, which means that it is based on the *ET* of  $DE_u$ ;  
and
  - the peer can select attributes that are already defined in  $DE_u$  or can input new attributes.
- If the peer inputs new attributes or new attribute values, the app has to verify if the identifiers of  $DE_u$  are affected.
- Now the app asks the user to set up the privacy preferences related to the card.
- The privacy preferences may refer to publishing, sharing, re-distributing and others options that will be explored in the following scenarios.
- Finally, when the peer saves the new card,  $DE_u$  is updated accordingly in the peer’s EB.

**7.3.1.4 Importing existing contacts****Personas:**

*Alice* is a researcher and new user of the DCM app. She heard about the features of the DCM app, she installed it, and now she is starting to use it.

*Ana* is an old friend of Alice, they know each other since they were kids.

*Enrico* is also a researcher, he is a colleague and a good friend of Alice.

**Description:**

- Alice wants to initialize her contact list, but she would like to do it in a semi-automatic manner.
- The app allows Alice to import existing contacts from the native contact list of her smart-phone.
- She selects the import option and waits while the system performs the importing.
- Alice can now see notifications about the contact of Ana because the DCM app detected duplicate entries for her. Alice decides to merge the two entries in her contact list.
- Then, she is invited to import contacts also from other accounts (e.g., Facebook, Skype, Google Plus, etc).
- Alice selects the import contacts option and then chooses Facebook and Skype as sources.
- She authorizes the app to access her Facebook and Skype data.
- Alice's contact list is automatically updated with the contacts imported from external accounts. Some of them are also automatically merged, as she has the same person as a contact in different accounts. For example, Alice has the contact of Enrico in the contact list of her phone and he is also a friend of her in Facebook.
- Next, Alice notes that the contact of Enrico is automatically updated with his Facebook user account, i.e., Facebook is added as a possible mean to communicate with Enrico.
- On the other hand, the DCM app shows a notification regarding the contact of Ana, which might be again duplicated. A contact from the native contact list and another contact from Facebook, seems to refer to the same person, i.e., Ana.
- When Alice confirms that both refer to the same person, the system suggests to merge the contacts.
- She now, sees only one contact of Ana with a set of attributes that is the result of merging the attributes from the matching contacts.

**System Actions:**

- Contacts imported from any source are represented as entities in the system.
- If the contacts has to be imported from an external account, the app:



- (i) requests user’s authorization to access the external account, and
- (ii) notifies the user how the data from the external account will be used.
- Then, the app performs the following steps for each imported contact:
  - The imported contact is modeled as a digital entity  $DE_n$ .
  - The app searches in the peer’s EB for existing entities that match with  $DE_n$ .
  - If a perfect match (above some threshold) is found, the exiting entity is updated with the information in  $DE_n$ . This can be also seen as an automatic merge operation of both entities.
  - If no match is found,  $DE_n$  is added as a new entity. This, in turn, implies that:
    - \* First, the DCM app generates a new SURL for  $DE_n$ ;
    - \* Second, if  $DE_n$  has enough information to be globally identifiable<sup>4</sup>, then a new SURI is generated and assigned to it;
    - \* Third,  $DE_n$  is permanently stored as a new entity in the peer’s EB.
  - If partial matches are found,  $DE_n$  and the entities that partially match with it, are included in a list of potential duplicates. Such list will be later presented to the user, who can decide what to do with them.
- When merging contacts, a preferred name is selected and then the set of attributes that describe the merged contacts, in turn, are merged.
- In order to merge two attribute sets, the attribute definitions  $ADs$  from both sets are compared. If two  $ADs$  match (i.e., if  $AD_i.C$  is equivalent to  $AD_j.C$ ), then the set of values of the corresponding attributes are merged.

### 7.3.1.5 Adding a contact by hand

#### Personas:

*John* is a researcher at the university of Trento. He is attending a conference where he has to present a paper. He has a smart-phone and uses the DCM app to manage his contacts. *Federico* is another researcher that is also participating to the conference and he wishes to interact with other people having similar research interests. He also has a smart-phone but does not use any particular app (besides the native one) to manage his contacts. *Paolo* is an old friend of John and he is traveling to the city where John is attending to a conference.

#### Description:

---

<sup>4</sup>if it has at least one identifying set of attributes (as defined in [Pane, 2012])

- John and Federico meet at the conference, they agree on exchanging contact information.
- Federico does not use the DCM app, so he has to give his contact information to John by voice and John has to input the information on his DCM contact list by hand.
- Federico tells John his full name, affiliation and gives him his email address from university because that is the one he uses for work related things.
- Next, John opens the DCM app on his smart-phone, selects the options for creating a new contact and inputs Federico's email address.
- When finishing typing Federico's contact information, John saves the contact in his contact list.
- During the same trip to the conference, John meets Paolo. They are old friends from when they were kids but they haven't seen each other for a long time.
- After talking for a while, John asks Paolo his phone number in order to stay in touch.
- John opens the DCM app and inputs the contact information of Paolo.
- When he tries to save the information, the app shows a notification telling John that he already has a contact (in his contact list), which apparently corresponds to the same person.
- John realizes that he has an old and outdated contact information of Paolo.
- Instead of creating a new contact, John selects the option of updating the information of the existing contact.

**System Actions:**

- The app first asks the peer to select the type of contact among the *ETs* locally defined (e.g., person, facility, etc.).
- Then, an ET dependent form is displayed to the peer. The attributes that are mandatory are highlighted in the form.
- The peer edits the form's fields in order to input the attributes for the contact, while the app verifies that the given attribute values are valid and that all mandatory attributes are provided.

- A new entity instance  $DE_n$  is created with the information provided by the peer (i.e., first, Federico's contact information and then Paolo's).
- The DCM app performs a local search to check if the peer's EB already contains an entity that matches with  $DE_n$ .
- If no match is found,  $DE_n$  is added as a new entity. This, in turn, implies:
  - First, the DCM app generates a new SURL for  $DE_n$ ;
  - Second, if  $DE_n$  has enough information to be globally identifiable, then a new SURI is generated and assigned to it;
  - Third,  $DE_n$  is permanently stored as a new entity in the peer's EB.
- If at least a partial match is found, the peer is notified.  $DE_n$  and the entity that partially matches with it are shown to the peer as potential duplicates. The peer can now decide if they have to be merged, maintained separated (because they actually refer to different entities), etc.
- When merging contacts, a preferred name is selected and then the set of attributes that describe the merged contacts, in turn, are merged.
- In order to merge two attribute sets, the attribute definitions  $ADs$  from both sets are compared. If two  $ADs$  match (i.e., if  $AD_i.C$  is equivalent to  $AD_j.C$ ), then the set of values of the corresponding attributes are merged.

### 7.3.2 Sharing Scenarios

The scenarios presented in this section are called of *sharing*, they include activities related to share, publish and exchange contacts in the DCM system.

#### 7.3.2.1 Sharing the own contact with a non-user of the DCM app

##### Personas:

*John* is a researcher at the university of Trento. He is attending the ISWC conference where he has to present a paper. He has a smart-phone and uses the DCM app to manage his contacts.

*Federico* is another researcher attending the ISWC conference and he wishes to interact with other people having similar research interests. He has a smart-phone and uses only the native application to manage his contacts.

##### Description:

- John and Federico meet at the conference, they agree on exchanging contact information.
- John wishes to provide to Federico his contact information related to work. He wants to give his email address, LinkedIn profile, Skype user name and the phone number of his office at the university, which he has included in a presentation card called “work”.
- From the menu in the DCM app, John selects the “send card” option and then he selects his “work” card.
- John has the contact information of Federico in his contact list because he previously added it by hand (see Scenario 7.3.1.5).
- When the app asks John to input the destination to send the card, he searches in his contact list and selects Federico’s contact.
- John has the email address of Federico, so the DCM app generates an email to send Federico the information contained in John’s work card.
- On receiving the email, Federico can manually add the information to his contact list (i.e., copy/paste). If he decides to start using the DCM app, he should be able to import automatically the contact from the received email to his contact list. Moreover, with the DCM app he will be able to maintain the received information updated.

#### System Actions:

- On selecting the “send card” option, the DCM app retrieves the entity  $DE_p$  associated to the peer.
- Next, the app retrieves the presentation cards  $DE_p.\{PC\}$  associated to  $DE_p$  and presents them to the user (i.e., the list of available predefined PCs are retrieved).
- After the user selects a presentation card  $PC_i \in DE_p.\{PC\}$ , the app shows the list of contacts allowing the peer to select the destination to send it. The peer can also input manually to whom and how to send the card.
- If the peer selects a *NON-USER* contact, the card cannot be sent through the DCM system and the system proceeds to identify other mechanisms that can be used to send  $PC_i$  (based on the contact information of the selected contact). In this case the system identifies that  $PC_i$  can be sent by email.

- Next, the system takes the information in  $PC_i$  and generates a file in the appropriate format standard (e.g., VCard), depending on the mechanism selected to send.
- The DCM app calls another application in the smart-phone (e.g., the preferred email client of the peer) and requests it to send an email, where the contact information can be added as attachment and the destination should be set according to the corresponding attribute (i.e., email address) in the selected target contact.
- After sending the information in  $PC_i$  to a *NON-USER* contact, the system loses control over the sent data.

### 7.3.2.2 Sharing the own contact with other users of the DCM app through the DCM portal

#### Personas:

*NGI* is a company, an internet service provider. The company outsources the installation services for its client.

*Franco* is a technician (independent worker) who works as an outsourced technician for NGI. He also performs reparations of different types of electronic devices. He uses the DCM app to manage his contact list.

#### Description:

- Potential clients for Franco are those referred to him by NGI and other people to whom his current clients recommend him. As an independent worker, Franco wishes to publish his contact information in order to allow clients (or potential clients) to find him.
- Franco opens the DCM app and from the menu selects the option “share contact” which allows him to share his own contact information using the DCM system.
- Then, he selects the option to upload a card into the DCM portal.
- Next, he selects a presentation card called “work” and decides to make it publicly available in the system.
- When publishing, Franco assigns public permits, allowing every user of the DCM system to find his contact information. He also includes the permissions allowing other users to import the contact information (to their own contact list), to share the contact with third parties and to allow synchronization.

#### System Actions:

- On selecting the “share” option, the DCM app retrieves from the local peer’s  $EB$  the entity  $DE_p$  associated to the peer.
- Next, the DCM app shows to the peer the different alternatives that are available to share contact information:
  - To index the  $DE_p$  in the DCM portal but without uploading the actual data;
  - To upload a presentation card to the DCM portal;
- In order to upload a card into the portal, the DCM app retrieves the presentation cards  $DE_p.\{PC\}$  associated to  $DE_p$  and ask the peer to select a card to publish.
- After the peer selects a presentation card  $PC_i \in DE_p.\{PC\}$ , the system retrieves the privacy settings associated to the card allowing the peer to verify, modify it and/or approve them.
- If the privacy settings associated to  $PC_i$  are modified, the modifications are highlighted and the approval of the peer is requested.
- Then, the system starts a thread to upload and index  $PC_i$  in the portal.
- If the peer decides to index without upload a card, then anyone will be able to find the SURL of the peer (i.e., of  $E_p$ ) in the directory and when dereferencing the SURL, the information from some  $PC_j \in DE_p.\{PC\}$  should be returned. Which  $PC_j$  is returned will depend on the privacy settings of the peer.

### 7.3.2.3 Exchanging the own contacts face to face

#### Personas:

*John* is a researcher at the university of Trento. He is attending to a conference where he has to do a presentation of a paper. He has a smart-phone and uses the DCM app to manage his contacts.

*Peter* is another researcher that is also participating to the same conference. He wishes to interact with other people having similar research interests. He also uses the DCM app in his smart-phone to manage his contacts.

#### Description:

- Peter meets John in person and they start discussing some ideas related to John’s presentation.
- They agree to stay in contact for (possible) future collaborations, they need to exchange their contact information for this.

- John opens the DCM app and from a menu selects the option “exchange contact”.
- Because they are close to each other, John’s phone can detect Peter’s phone and send a request to exchange contacts.
- Peter receives the exchange contact request through the DCM app running in his device.
- Peter accepts the request and selects which contact card to send.
- John also selects which card to send and then the exchange is made directly between the two devices.
- Now, they will be able to contact each other.

#### System Actions:

- On selecting the “exchange contact” option, the DCM app broadcasts a contact exchange request and then scans searching for an answer from a DCM user nearby accepting the request. This could be enabled, for example, by NFC<sup>5</sup> technology.
- After the DCM app detects another peer, the two devices can establish a connection<sup>6</sup>.
- Now the app retrieves the cards  $DE_p.\{PC\}$  from the entity of the peer (from the local  $EB$ ).
- After the local peer selects a presentation card  $PC_i \in DE_p.\{PC\}$  to be exchanged, a “contact” message containing the  $SURI$ ,  $SURL$ , and the entity attributes from  $PC_i$  is sent to the target peer.
- On receiving back a “contact” message from the target peer (as respond of the exchange), the message is parsed to retrieve  $PC_n$  the presentation card of the new contact.
- The peer is given the options of “copy” or “link/synchronize” the contact.
- If the peer selects the “copy” option, a new entity instance  $DE_n$  is created from the information in  $PC_n$  where:
  - $DE_n.SURI = PC_n.SURI$ , i.e., the  $SURI$  from  $PC_n$  is assigned to  $DE_n$
  - All the other attributes from  $PC_n$  are imported as attributes of  $DE_n$

<sup>5</sup>[http://en.wikipedia.org/wiki/Near\\_field\\_communication](http://en.wikipedia.org/wiki/Near_field_communication)

<sup>6</sup>using for example Bluetooth (<http://en.wikipedia.org/wiki/Bluetooth>) or Wi-Fi Direct ([http://en.wikipedia.org/wiki/WiFi\\_Direct](http://en.wikipedia.org/wiki/WiFi_Direct))

- If the peer selects the “link/synchronize” option, a new entity instance  $DE_n$  is created where:
  - $DE_n.SURI = PC_n.SURI$ , i.e., the SUR I from  $PC_n$  is assigned to  $DE_n$
  - The DCM app retrieves the names of the received contact from  $PC_n$  and assigns them  $DE_n$ , i.e.,  $DE_n.\{N\} = PC_n.\{N\}$
  - $PC_n$  is added as a presentation card to  $DE_n$
- Then, the DCM app performs a local search to check if the peer’s EB already contains an entity that matches with  $DE_n$ .
- If a perfect match (above some threshold) is found, the exiting entity is updated with the information in  $DE_n$ . This can be also seen as an automatic merge of both entities.
- If no match is found, a new *SURL* is generated for  $DE_n$  and  $DE_n$  is permanently stored as a new entity in the peer’s EB.
- If a partial match is found,  $DE_n$  and the entity that partially matches with it are included in a list of potential duplicates. The list is presented to the user, who can decide what to do with them (e.g., merge, discard, etc.).

#### 7.3.2.4 Sharing the contact of third parties

##### Personas:

*NGI* is a company, an internet service provider.

*Alice* is a new client of NGI. She subscribed to an internet service package and is currently waiting for the installation to be performed.

*Franco* is a technician that performs different types of antenna installations for different companies offering services like internet, sky, and others.

##### Description:

- Franco is hired by NGI as an outsourced technician to perform the installation of the internet antenna at Alice’s apartment.
- Franco and NGI use the DCM application to exchange information about clients.
- Alice gives NGI the permissions to share her contact information with the technician that has to install the internet antenna at her house.
- NGI sends Franco the contact information that Alice gave them, which is needed for the installation.



- Franco calls Alice and they agree on a day and time for the installation.
- When Franco finishes the installation, Alice decides to cancel the permission that Franco has over her contact information.
- When the DCM app in Franco's device receives the update of the permissions, Alice's contact information is removed from his contact list.

#### System Actions:

- On selecting the "send" option, the DCM app retrieves from the local peer's  $EB$  the entity  $DE_p$  associated to the peer.
- Then, the DCM app allows the peer to select a presentation card  $PC_i \in DE_p.\{PC\}$ .
- The selected card  $PC_i$  is assigned with permissions (i.e.,  $PC_i.PP$ ) that allows the recipient of the card to re-distribute the information, but notifying its owner.
- ...
- After some time, the peer might decide to verify the access control list associated to the card  $PC_i$  and remove the permissions that another peer has. When this happens, the DCM app notifies the modification to other peers that are affected by the changes. The DCM app at those peers verify that the access to  $PC_i$  has been cancelled and remove the contact from the contact list of the peers.

### 7.3.3 Search Scenarios

Some examples of usage scenarios showing the envisioned *search* features of the proposed application are:

#### 7.3.3.1 Search a contact by name

##### Personas:

*Enrico* is a researcher at the University of Trento.

*John* is originally from the south of Italy and recently moved to Trento after getting a position at the University of Trento. He is a new colleague of Enrico, he has a smart-phone and he uses the DCM app to manage his contacts information.

The "*Green Tower*" is a restaurant located in the city center of Trento that serves typical local dishes.

##### Description:

- John would like to go out for dinner and try local dishes. Unfortunately, she does not know any restaurant.
- He asks to Enrico for some recommendations, he suggests the restaurant Green Tower.
- John would like to make a reservation, so she asks Enrico if he can give him a telephone number to which he can call for that.
- Enrico do not have the number but he knows that the contact of the restaurant is in the directory of the DCM.
- Enrico also knows that John is a DCM user, so he tells him that the contact of the restaurant can be found in the DCM directory.
- John searches the “Green Tower” restaurant in the directory and imports it to his contact list. He finds that the restaurant has also a web page and a skype account through which clients can make reservations.

**System Actions:**

- On selecting the “search by name” option, the application allows the peer to search for information about an entity based on its name.
- The peer has to input the pair  $\langle N, ET \rangle$ , where  $N$  is the name of the target entity and  $ET$  is its type (e.g., person, facility, etc.), which is not mandatory.
- The DCM app queries the *Entity Directory* to find candidate entities matching the given parameters.
- The directory returns the *SURLs* of the candidates entities.
- On dereferencing a *SURL* the DCM app receives a “contact” message.
- Let us call  $p_i$  to the initiator peer that requests the dereferentiation of the *SURL* and  $p_s$  to the source peer that owns the information.
- At this point, we assume that  $p_i$  provides its credentials when requesting the dereferentiation of the *SURL* and access control rules are evaluated at  $p_s$  based on the credentials given by  $p_i$ .
- If the application can not contact  $p_s$  (the source of information) to dereference a *SURL*, it moves to the next available source. At this point, the next available source may be the DCM portal (if the peer uploaded its presentation card there).

- On receiving the “contact” message, the message is parsed to obtain  $PC_n$  the presentation card of the target entity.
- Now the peer has the options of “copy” or “link/synchronize” the contact (as described in Scenario 7.3.2.3).

### 7.3.3.2 Search based on existing Non-User contact (to match them to DCM Users)

#### Personas:

*Alice* is a researcher, she is also a new user of the DCM app.

*Enrico* is also a researcher, he is a colleague of Alice at the University of Trento and is also good friend of her.

*John* is originally from the south of Italy but he moved to Trento when he got a position at the University of Trento. He is a new colleague of both, Alice and Enrico. He has a smart-phone and he uses the DCM app to manage his contacts information.

#### Description:

- Alice finished setting up the DCM application and she starts to use it.
- She starts a search/matching process that tries to find out which of her contacts are also DCM users.
- After a while, Alice receives a notification from the DCM app. The app shows the number of contacts from her contact list that match with a DCM user.
- By scanning through her contact list, she can visually distinguish that John is a DCM user (through a different icon, different color, etc.)
- Alice is notified about the number of possible matching contacts. She has the option of verifying/confirming them manually.
- By scanning the possible matchings, she finds the contact of Enrico, and she confirms that the matching is correct.
- Now Alice can also distinguish (visually) the contact of Enrico as a DCM user.

#### System Actions:

On starting the process of matching existing contacts, the DCM app starts a new thread that runs in the background and for each *NON-USER* contact  $DE_i$  carries out the following steps,

- The DCM App sends a request to search for a user, in the *UB* of the portal, that is associated to an entity, in the *EB* of the portal, matching with  $DE_i$ .

- If a perfect match is found (i.e., above a predefined threshold), the DCM app receives from the DCM portal the *SURI* of the matching entity and the user name  $uName_i$  of the corresponding user.
- Now, the DCM app updates  $DE_i$  in the *EB* of the peer and makes it a *USER* contact, which means that:
  - $uName_i$  is added as an attribute to  $DE_i$
  - $DE_i.SURI$  is updated using the *SURI* received from the portal
- If only partial matches are found, the peer can be involved in order to evaluate the options.
- When a *NON-USER* contact is updated to a *USER* contact, the peer may decide to “search” available presentation cards based on the existing DCM user contact (see Scenario 7.3.3.3).

### 7.3.3.3 Search based on existing DCM User contact (to get a card and make it a sync contact)

Note that this type of search is an example of a scenario that can also be triggered periodically in the background with the purpose of automatically finding more information about an existing contact and to suggest the peer to link it and maintain the contact updated.

#### Personas:

*Alice* is a researcher at the University of Trento and she is a new user of the DCM app.

*John* is also a researcher, he is a colleague of Alice at the University of Trento.

#### Description:

- John has been using the DCM application for quite some time now, while Alice is a new user of it.
- During the initialization of the app, the contact of John in Alice’s contact list has changed from normal *NON-USER* contact to *USER* contact.
- Alice has only the university email address of John and she would like to get more contact information of him.
- If possible, she would also like to maintain John’s contact updated in the future.
- She searches John’s card in the network.

- Next, Alice receives John’s presentation card and she finds out that he also has a LinkedIn profile, a website and an office phone number.
- On receiving the card, Alice has two options: “copy” or “import and maintain updated”.
- She selects the “import and maintain updated” option.
- Next, Alice can see that John’s contact information is updated. She can also distinguish (visually) that John’s contact information is now synchronized (i.e., is up to date).

### System Actions:

- On selecting the “search card” option for an existing *USER* contact  $DE_i$ , the DCM app queries the DCM portal providing  $DE_i.SURI$  and the credentials of the local peer  $p_l$ .
- If the portal has a card for  $DE_i$  and is authorized to share it with DCM users, then this card is return to  $p_l$ .
- When  $p_l$  receives the card  $PC_i$ , the DCM app starts a new thread that runs in the background retrieving  $PC_i.SURL$  (i.e., the *SURL* of the card) and uses it to request more contact information directly to the owner of the card.
- In parallel, the DCM card gives  $p_l$  the options of “copy” the information from the card or “link/synchronize” it. Then,  $DE_i$  will be updated accordingly in the *EB* of  $p_l$ .
- It is important to note that  $DE_i.SURI$  should be equal to  $PC_i.SURI$ , at this point.
- If the peer selects the “copy” option, all the attributes from  $PC_i$  are imported as attributes of  $DE_i$ .
- If the peer selects the “link/synchronize” option:
  - The DCM app retrieves the names of the contact from  $PC_i$  and adds them to  $DE_i$ , i.e.,  $PC_i.\{N\} \in DE_i.\{N\}$
  - $PC_i$  is added as a presentation card to  $DE_i$
- If at any point the DCM app receives more information about this contact (i.e., directly from the owner), then  $DE_i$  is updated again accordingly and based on the option (“copy” or “link/synchronize”) previously selected by the peer  $p_l$ .

### 7.3.3.4 Search by description

#### Personas:

*Federico* recently found out that his younger daughter has rare skin disease. He is a user of the DCM system.

#### Description:

- Federico do not know where he can find a doctor to treat his daughter.
- He decides to search in the DCM network, to see if someone knows about a dermatologist that is specialized in his daughter's disease.
- He selects the search contact by description option.
- Next, Federico inputs the query where he specifies that it is searching the contact of a "person", with a degree in "medicine", who is specialized in "dermatology", and that is an expert in the disease "sss".
- He selects a node in his classification that correspond people that are doctors, and issues the query.
- Federico finds the contact of a relevant doctor, he imports the presentation card  $PC_d$  of the doctor and the next day he calls to fix an appointment.
- After talking to the doctor, Federico finds out that the doctor is a friend of his cousin. However, his cousin did not know what was the specialization of his friend (he knew only that he was a doctor).

#### System Actions:

- On selecting the "search by description" option, the peer DCM app allows the peer to specify a query  $Q$ .
- Next, the app retrieve the classification of the peer and allows him to select a node.
- The meaning  $M^n$  of the selected node is retrieved.
- A search request  $\langle Q, M^n \rangle$  is, first, evaluated locally in order to find local contacts that can match the request and links to other peers that are relevant to the request. Let us call  $\{l_r\}$  to the set of relevant links.
- From  $\{l_r\}$  the DCM app obtain the set  $\{P_r\}$  of relevant peers.
- The DCM app contacts the DCM network by sending the search request  $\langle Q, M^n \rangle$  to each peer  $P_{r_i} \in \{P_r\}$  (i.e., the search is started following the approach proposed in Chapter 6).

## 7.4 Summary

The Distribute Contact Management (DCM) System was presented in this chapter, enclosing the reference architecture and the search approaches previously discussed. In order to do so, two application specific notions were introduced, Presentation Card (PC) and DCM User.

The PCs were formally defined to support the exchanging of contact information between peers. A PC denotes a profile created to present one view (perspective) of a particular contact, represented in the system as an entity DE. On the other hand, a DCM User was defined as a peer having an account in the DCM platform.

The rest of the chapter presented different types of usage scenarios illustrating the main functionalities and features of the system. Namely,

- The initialization scenarios correspond to the first interactions that the peers have with the system, when the peer is starting to use the DCM system. They describe activities like, creation of an user account, creation of presentation cards, importing contacts, and creating them by hand.
- The sharing scenarios correspond to interactions that peers can have with other peers (users and not users of the system) using the DCM application. They include activities related to sharing, publishing and exchanging the own contact information as well as the contact of third parties.
- The search scenarios, show examples of the envisioned search features of the proposed application including activities like searching contacts by name, based on existing contact information, and by description.





**Part III**

**Evaluation**



## Chapter 8

# Experimental evaluation of name-based search

This chapter is aimed to discuss the evaluation of the distributed directory presented in Chapter 5. We implement the distributed directory on top of a P2P network, where the distribution of the indexes is done using a Distributed Hash Table (DHT). DHTs<sup>1</sup> are distributed systems that allow the peers participating in the network to store and retrieve pairs of key and value. Then, we perform a preliminary evaluation of performance. PlanetLab<sup>2</sup> is used as a testbed for the evaluation, we believe it gives us realistic network conditions.

### 8.1 Implementation

Different DHT protocols and implementations can be found in the literature. In particular, Chord and Kademlia, are two relevant protocols calling our attention.

*Chord*, which has been used in prominent research projects network, provide an efficient routing performance in terms of number of hops. However, in a real network setting “close neighbors” according to the protocol can be physically located far away in the network. Chord builds a ring topology and uses unidirectional metric (clockwise circle metric), which allows the convergence along the same path for all the lookups of the same key.

*Kademlia*, which has been successfully used in real distributed applications (e.g., Bit-

---

<sup>1</sup>[http://en.wikipedia.org/wiki/Distributed\\_hash\\_table](http://en.wikipedia.org/wiki/Distributed_hash_table)

<sup>2</sup><https://www.planet-lab.eu/>

Torrent's distributed tracker, Kad network and others), is able to route queries through low-latency paths making it more suitable for real applications scenarios. It uses XOR metric, which is also unidirectional. The topology has also the property that every message exchanged conveys or reinforces useful contact information. Differently from Chord's ring topology, Kademia's XOR topology is also symmetric and this allows the nodes in Kademia to learn useful routing information from queries they receive. Kademia performs the caching of  $\langle key, value \rangle$  pairs along the lookup path to alleviate hot spots. Finally, Kademia is the first DHT approach that exploits the fact that node failures are inversely related to uptime.

We believe that Kademia's protocol has a number of features that made it more suitable for real applications. Now, the Table 8.1 summarizes characteristics of existing implementations of this protocol that we analyze under the light of our requirements for the implementation of the distributed directory. Given the model for the directory presented in Section 5.2, we require an implementation that supports (or can be easily extended to support): (i) storing multiple values mapped to the same key, and (ii) multiple indexes.

Table 8.1: Implementations of Kademia protocol.

Implementations	Characteristics			
	Features	Documentation	Programing lenguaje	License
MainLine DHT	A plug-in implementation for Azureus. Is the standard DHT used by BitTorrent	No documentation	Java	GNU General Public License (version 2, June 1991)

TomP2P	Implements a XOR-based iterative routing based on Kademia. Extended DHT operations and supports custom operations. Direct and indirect replication.	Unit tests and examples within the source code. Basic manual of use for developers.	Java	Apache License (Version 2.0, January 2004).
Plan-x	Library that allows instantiation of Kademia nodes, supporting the storage and retrieval of serializable objects.	The javadoc of its API is available but there is neither documentation nor examples about how to use the library.	Java	Open Source
LibTorrent	A BitTorrent implementation, which include some extensions to the Mainline kademia protocol (to provide support for trackerless torrents).	The documentation of its API is available at <a href="http://www.rasterbar.com/products/libtorrent/manual.html">http://www.rasterbar.com/products/libtorrent/manual.html</a> .	C++	Released under the BSD-license ( <a href="http://www.opensource.org/licenses/bsd-license.php">http://www.opensource.org/licenses/bsd-license.php</a> ).

Daylight	Library that provides a simple interface to develop distributed applications targeting the .Net and Mono platforms.	No documentation	C#	GNU Library or Lesser General Public License (LGPL).
BitDHT	General purpose DHT library that is compatible with bittorrent's DHT.	Basic unit tests and example code of how to use libbitdht are provided.	C++	GNU Library or "Lesser" General Public License version 3.0 (LGPLv3).
JKad	Implementation of kademia protocol used in the JMule application (a Java based client for eDonkey2000 networks).	There is no documentation about how to use it.	Java	GNU General Public License (GPL)

The implementations developed in Java are more relevant for compatibility reasons with other existing implementations in the Knowdive<sup>3</sup> group. Among them, we discard MainLine DHT as it is not a stand alone application and has no documentation. These issues make its use difficult outside the originally intended environment (i.e., inside Azureus application). Then, in the case of Plan-x and JKad<sup>4</sup> their main drawback is the lack of proper documentation.

For our implementation, we use TomP2P<sup>5</sup>, an advanced DHT library that extends the

<sup>3</sup><http://disi.unitn.it/~knowdive/>

<sup>4</sup>JKad is a java implementation of kademia protocol developed as part of the JMule project (<http://jmule.org/>)

<sup>5</sup><http://www.tomp2p.net/>

basic functions of DHTs. The library supports storing multiple values mapped to the same key and distinguishes between different index domains. The execution of the operations over different index domains can be seen as having different DHTs, i.e., one for the *DEindex* and other for the *WEindex*. This library also offers some basic documentation and has a (responsive and active) community of users and developers providing supports through a mailing list.

## 8.2 Evaluation

We are interested in the evaluation of the approach under realistic network conditions and we want to measure how much the performance decreases when the size of the network grows (i.e., the scalability). The performance is considered here in terms of the time that takes the system to process a query. We use *PlanetLab*<sup>6</sup> as a testbed because we believe it gives us the realistic network conditions that we need. PlanetLab provides a network of computers (i.e., nodes) that are distributed around the world, connect to each other through the internet and are available for research purposes. We perform the evaluations on networks of 50, 100 and 150 peers and the data extracted from the proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)<sup>7</sup> are used to generate the data-sets. We use the titles of publications, names of authors and names of locations related to the conference.

Each data-set is produced by generating triples of  $\langle Name, URI, URL \rangle$ . The names and *URIs* are replicated in order to simulate different *WEs* having the same name and different peers storing *DEs* that describe the same *WE*. Let us call  $p_n$  to the popularity of a name  $n$  (i.e., number of *WEs* that are called by  $n$ ) and  $p_{we}$  to the popularity of a *WE* (i.e., number of *DEs* that represent *WE*). First, for each name  $n$ , we generate  $p_n$  triples with the same name (different *URI* and *URL*). Second, for each *URI*, we generate  $p_{we}$  triples with the same name and *URI* but with different *URLs*. The popularities  $p_n$  and  $p_{we}$  follow a Zipf<sup>8</sup> distribution, which means that there is a long tail of unpopular names and *WEs*. The distribution of both popularities are independent, which means that a popular *WE* do not necessarily has a popular name and vice versa. We assume that the local entity base of each peer contains, in average, 2000 *DEs*. We have overall around 100000, 200000 and 300000 *DEs*. The query set for each peer is generated by randomly selecting a set of 1400 names from the initial set of entity names.

During the evaluation, we first index the data-set for the corresponding network size and then the peers begin the search evaluation process pseudo-simultaneously. In this

---

<sup>6</sup><https://www.planet-lab.eu/>

<sup>7</sup><http://ijcai.org/>

<sup>8</sup>[http://en.wikipedia.org/wiki/Zipf's\\_law](http://en.wikipedia.org/wiki/Zipf's_law)

process, each peer performs the following steps: (i) takes a query from the query set, (ii) runs the search, (iii) measures and logs the time that the system takes to respond to the query, (iv) waits a random interval of time (between 1 and 3 seconds), and (v) go back to step (i). These steps are repeated until the end of the set of queries. Once all the peers end the search process, we compute the average query time for the network. We show the results for the different network sizes in Table 8.2. The values for the average query times

Table 8.2: Average query time

Network Size	50 peers	100 peers	150 peers
Avg. Query Time (in seconds)	2.77	2.75	2.61

are stable with the network growth and we believe this is a promising result regarding the scalability of the directory. On the other hand, when comparing to information retrieval systems (in general), the average times for search are still high.

In order to have better understanding of the query times that contribute to these averages, we analyze the distribution of the query time in the different networks. In Figure 8.1 we show the results of this analysis, where we can see that also the query time distribution is stable with regard to the network growth. Also in Figure 8.1 we can notice that more than 55% of the queries are actually answered in less than a second, while in almost 70% of the cases the response arrives in less than 2 seconds (which is less than the average time). Moreover, only 9% of queries take more than 5 seconds to be answered.

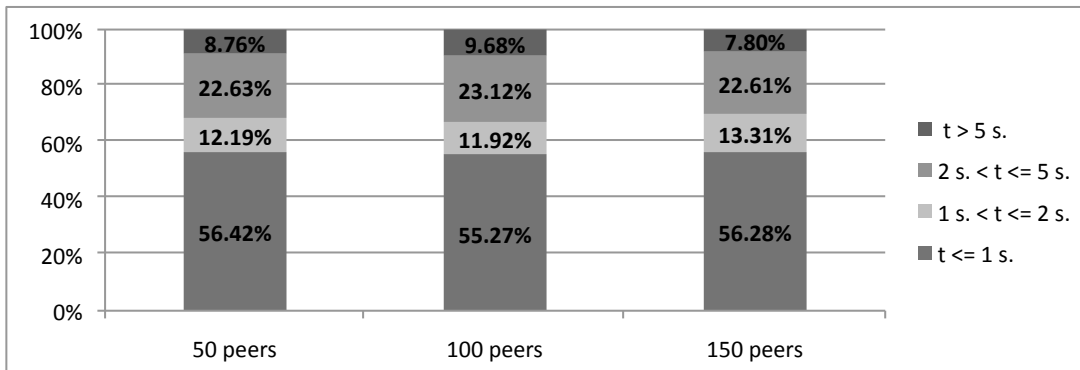


Figure 8.1: Query time of different networks

It has to be noted that the results are returned after the query answer is complete, i.e., once all the lookups involved in the query have ended. This means that a single slow lookup is enough to delay the computation of a query answer and therefore increase the query time. Furthermore, we know that particularly slow peers can produce this problem



when a lookup has to be routed through them. We believe that, in the big picture, the scalability of the approach is a promising and important result. On the other hand, there are some techniques to perform result catching or to avoid routing through slow peers (see for example Rhea et al. [2005]) that can be implemented to reduce the effect of slow peers at query time.

### 8.3 Summary

This chapter reported the outcome of the experimental evaluation of the distributed directory, which was proposed in Chapter 5 to perform name-based search of contacts in a distributed network of peers, i.e., a P2P network.

The implementation was done using a Distributed Hash Table (DHT) for the storage of indexes. Two DHT protocols were mainly considered, Chord and Kademlia, the latter was selected because of several features that make it better for real applications. Next, a comparison of existing implementations of the Kademlia protocol was presented. TomP2P, an advanced DHT library that extends the basic functions of DHTs, was selected.

The evaluation was done using *PlanetLab* as a testbed. PlanetLab provides a network of computers that are distributed around the world, providing a realistic scenario in terms of network conditions. Networks of 50, 100 and 150 peers were configured assuming that the local entity base of peers contains, in average, 2000 *DEs*. When running the evaluation process, the peers are activated pseudo-simultaneously and execute a set 1400 queries randomly selected from an initial set of entity names. Once all the peers ended the search process, the average query time for the network was computed.

It was shown that the average query time (2.7 seconds) was stable with regard to the network growth, which is considered a promising result for the scalability of the directory. Moreover, the distribution of the query time with regard to the network growth was also stable in networks of different sizes. It was also noted that more than 55% of queries were actually computed in less than a second and 70% of them in less than 2 seconds.



## Chapter 9

# Experimental evaluation of description-based search

In this chapter we discuss the evaluation of the approach presented in Chapter 6, which allows peers to search contacts by their description, by conducting simulation experiments. The results of the proposed algorithm, called *Semantic Flooding*, and the centralized *C-Search* algorithm are compared<sup>1</sup>. The key intuition here is to see how much the distributed search approach loses to the centralized one, in terms of the number of results which are retrieved by the centralized approach and which are missing from the results of the distributed approach.

*The work presented in this chapter was performed in collaboration with Uladzimir Kharkevich and Prof. Fausto Giunchiglia. The content was published in [Giunchiglia et al., 2011].*

### 9.1 Implementation

In the experiments, C-Search algorithm was implemented on top of Lucene [Luc] as described in [Giunchiglia et al., 2009b]. A P2P network and the P2P Concept Search [Giunchiglia et al., 2009a] algorithm were simulated on a single machine. *Semantic Flooding* algorithm was implemented on top of this simulation as described in Chapter 6.

The accuracy of search results returned by *Semantic Flooding* algorithm is measured

---

<sup>1</sup>Note that comparing performance of distributed and centralized information retrieval systems is a standard way of evaluating in P2P information retrieval (e.g. see [Tang et al., 2003]).

depending on the number of visited peers, where the accuracy is defined as follows:

$$Accuracy = \frac{|R_{CS} \cap R_{SF}|}{|R_{CS}|} * 100\%,$$

where  $R_{CS}$  are results returned by C-Search and  $R_{SF}$  are results returned by *Semantic Flooding* on the same data-set. In the evaluation, only the first 10 ranked results were used to compute the accuracy. The accuracy measure considers the results returned by C-Search, as the desired results and estimates the ability of *Semantic Flooding* algorithm to approximate these results by querying only a limited number of peers.

The number of queried peers can be used to estimate the number of messages  $M_{num}$  generated to answer a query in the best and the worse case scenarios. In the best case scenario, no link discovery is needed because all the relevant links are already computed. The number of generated messages in this case can be estimated as follows:

$$M_{num} = 2 * m,$$

where  $m$  is the number of queried peers.

In the worst case scenario, the relevant links need to be computed by the link discovery mechanism. If *P2P Concept Search* is used for link discovery, then the number of messages can be estimated by using the following formula:

$$M_{num} = \log p + k + (2 * m),$$

where  $p$  is number of peers in the network,  $m$  is the number of queried peers and  $k$  is the number of atomic concepts that are used by *P2P Concept Search* algorithm (see [Giunchiglia et al., 2009a] for details). In the evaluation,  $k$  was limited to 10.

## 9.2 Evaluation

### 9.2.1 Data-set generation

In order to generate data-sets (which reproduce a realistic scenario) for the evaluation of the proposed *Semantic Flooding* algorithm, the data from the Open Directory Project (also known as *DMoz* [DMo]) and the public tags of about 950000 users from Delicious [Del] (obtained from [Wetzker et al., 2008]) were used. DMoz is a multilingual open content directory of World Wide Web links that is constructed and maintained by a community of over 80000 volunteer editors. The DMoz directory uses a hierarchical structure to organize links into topics and closely related topics are grouped into categories. DMoz contains over 590000 multilingual categories and over 4500000 web sites classified to these categories. Delicious is a bookmarking service allowing users to mark web pages

with tags (words describing the bookmark), store the bookmarks and then share them with other users. The tags were used in this work to identify users who are interested in a web page. The two sources were merged together by matching users from Delicious to nodes in DMoz classification. First, the intersection of URLs from both sources was computed. Second, users which tagged an URL in Delicious were matched to the node  $n$  in DMoz which classifies the document with the same URL. As a result, we obtained 491414 users matched to 45545 nodes.

Four data-sets were generated by randomly selecting sub-sets of 10, 100, 1000, and 10000 users (one user is assigned to one peer). For each user  $u$ , the generation of its classification was performed as follows. A classification hierarchy was formed by the nodes matched to a user. For each node  $n$  in the user's classification, a sub-set of documents  $D_n$  classified to node  $n$  in DMoz were selected. All the documents that were tagged by the user were selected at first, and then a random sub-set of spare documents (i.e. documents classified to  $n$  that were not tagged by user) were also selected. Documents in the data-set were created by concatenating titles and descriptions of web-sites. On average, a classification of each peer had 21 nodes and 385 documents.

For each data-set, a C-Search index  $I_{CS}$  was created. All the documents in the data-set were indexed in  $I_{CS}$ . WordNet was used in C-Search as a background knowledge. Indexes of each single peer were created by filtering  $I_{CS}$ . Distributed Background Knowledge (DBK) [Giunchiglia et al., 2009a], that provides access to the BK of each peer in the P2P network, was used to index concepts and relations from WordNet in the P2P network. By using DBK, a peer can exploit the knowledge of other peers in the network when an atomic concept is missing in the local BK of the peer. The DBK can be seen as the sum of the BKs of all the peers in the network. By using the same BK in both centralized and distributed approaches, the fairness of comparison of the results produced by these approaches is ensured.

A query set was generated by randomly selecting a set of  $N_q$  (100) queries from the AOL query log [Pass et al., 2006] for each data-set. One word queries; queries which contained punctuation, special symbols, or boolean operators (e.g. '+' and '?'); queries which contain the words shorter than 3 letters; and queries which had less than 10 results in  $I_{CS}$  were filtered out. For each query, a node  $n$  in  $DMoz$  classification were randomly selected, such that, a query request  $\langle C^n, C^q \rangle$  have at least 10 relevant documents as computed by *C-Search*.

Given that our data-sets are generated from the data produced by the real users, nodes matched to users provide us with knowledge about real user interest profiles. Note that interests and accordingly classifications of different users can partially overlap, where the overlap has a higher probability for popular topics (e.g. *Top/Computers* and *Top/News*).

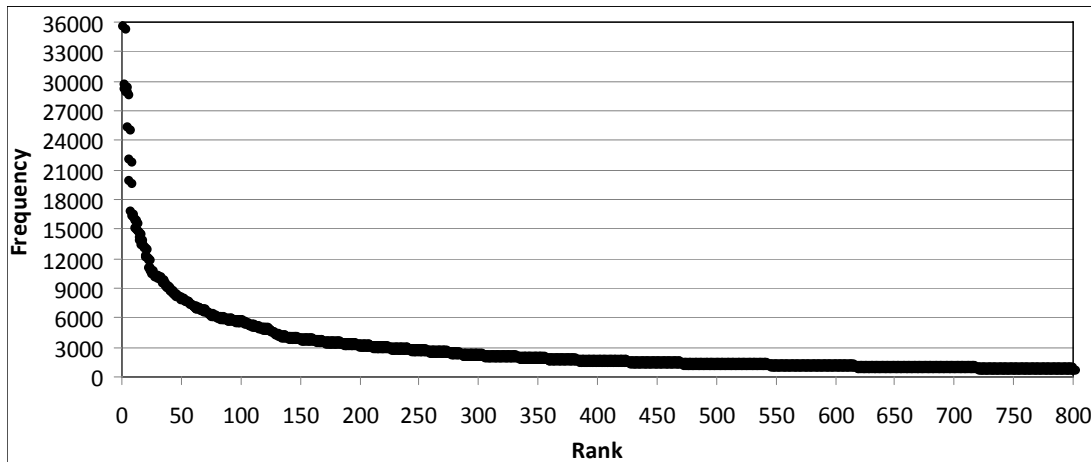


Figure 9.1: Topics Popularity Distribution

In the following the distribution of peers' interests over the set of topics from DMOZ is analyzed. In Figure 9.1, the distribution of the popularity of topics in our data-set is shown. Topics are ranked according to the popularity. The most popular topic occupies the first position in the ranking, followed by the second most popular, and so on. From Figure 9.1, the rapid decrease in the frequency can be observed (from 35503 for the most popular topic, to 29690 for the second most popular). Even more, only moving 50 places down in the ranking the frequency decreases to 8066. And there are only 121 topics that have more than 1% of peers interested on them. This behavior provides evidence of the existence of a “long tail” of unpopular topics.

In order to see how popularity of topics affect the performance of different approaches, two query sets for the data-set consisting of 10000 peers were additionally generated. The first query set consists of popular queries (i.e. queries related to topics that are in the first 200 positions in the popularity ranking, see Figure 9.1) and the second query set consists only of unpopular queries (i.e. queries related to topics in the position 400 or above in the popularity ranking, see Figure 9.1).

### 9.2.2 Evaluation of results

The evaluation results for randomly selected queries are reported in Figure 9.2. The performance achieved by *Semantic Flooding* is compared when: (i) the query request  $\langle C^n, C^q \rangle$  consists of a starting node  $n$  with concept  $C^n$  and of a query  $q$  with a concept  $C^q$ ; (ii) the query request is  $\langle \top, C^q \rangle$ , namely the same as in (i) but with no starting node, i.e.  $C^n \equiv \top$ ; and (iii) the same as (ii) but the semantic similarity  $SS(C^{n'}, C^q)$  is not used. Note that in P2P networks of 10 and 100 peers, the total number of queried peers

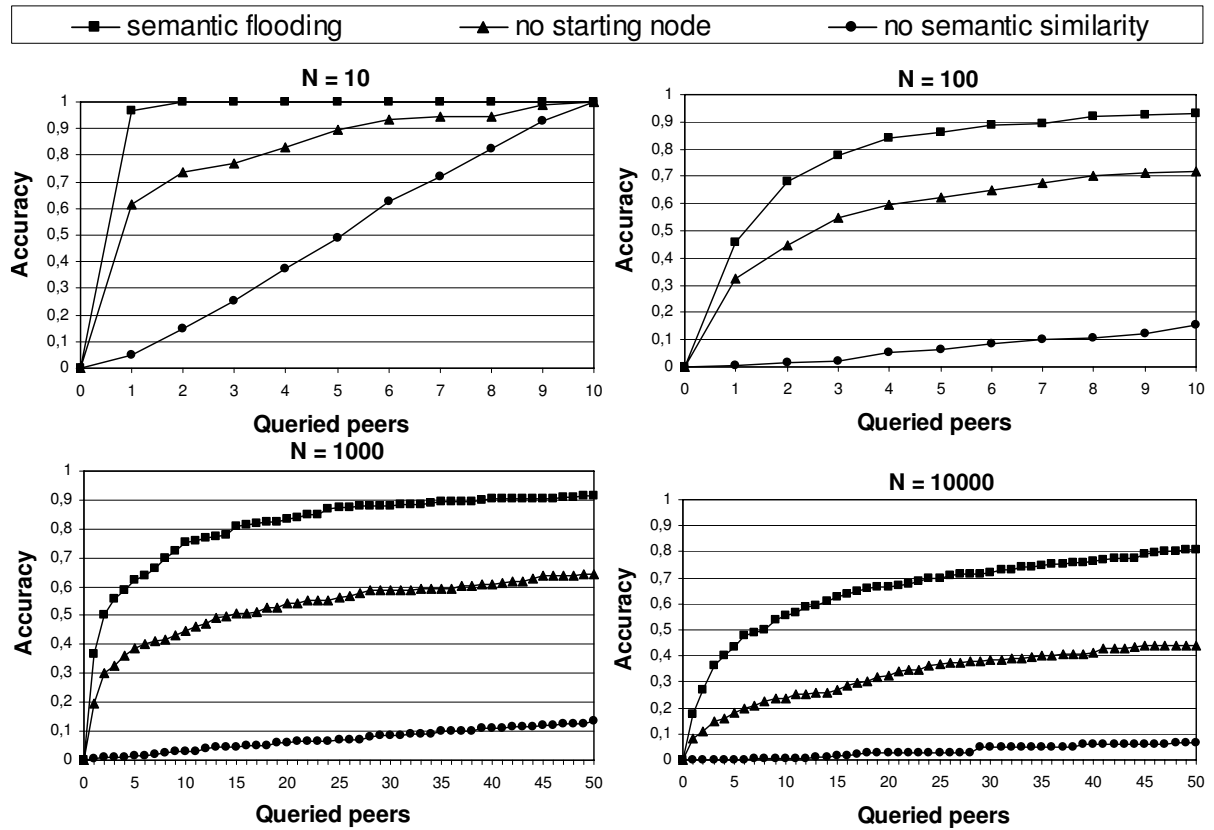


Figure 9.2: Evaluation Results: Random queries

was set to 10, whereas in P2P networks of 1000 and 10000 peers, it was set to 50. It can be seen from Figure 9.2, that when peers are selected without using the similarity function and also without a starting node specified (see “no semantic similarity” lines in Figure 9.2), accuracy decreases very quickly with the total number of peers in the network. The situation improves when semantic similarity is used and only starting node is missing (see “no starting node” lines in Figure 9.2). When the starting node  $n$  is selected, i.e. concept  $C^n$  is provided, the accuracy of *Semantic Flooding* becomes close to the accuracy of the centralized C-Search approach (see “semantic flooding” lines in Figure 9.2). In fact, in the network of 10000 peers, on average only 50 peers need to be queried in order to achieve 70% of accuracy. Note that if we need to retrieve one relevant result (i.e. 10% of accuracy), on average only one peer needs to be queried.

The evaluation of results for popular/unpopular queries are reported in Figure 9.3. From Figure 9.3, it can be seen that even a normal flooding approach can achieve a high accuracy for popular queries. This is because there are many peers which can provide answers to such queries. On the other hand, for unpopular queries the accuracy of results

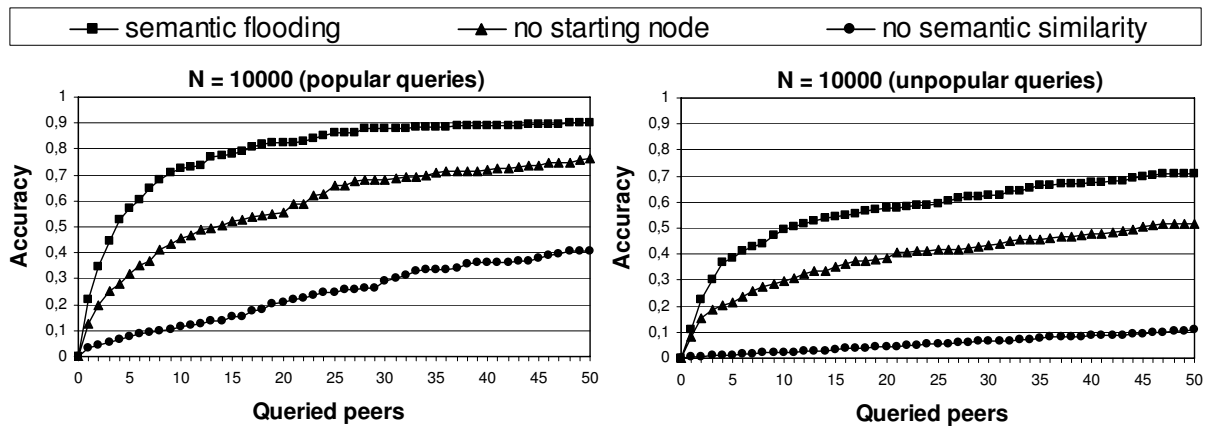


Figure 9.3: Evaluation Results: Popular vs. Unpopular Queries

provided by the normal flooding decrease substantially (i.e. the accuracy has decreased 4 times), whereas the *Semantic Flooding* approach can still provide good accuracy (i.e. the accuracy has only decreased by 22%). Overall, Figure 9.3 shows that the *Semantic Flooding* approach can provide results of high accuracy for both popular and unpopular queries.

### 9.3 Summary

The results of an experimental evaluation for a description-based approach to search in the DCM system was presented in this chapter. The algorithm proposed in Chapter 6 builds distributed yellow pages for contacts of peers by creating a semantic overlay that links directories of different peers in the DCM network. This semantic overlay is then flooded to search contacts based on their descriptions in a distributed manner, hence the name *Semantic Flooding*. It is important to note that the evaluation of the approach presented in this Chapter was applied to documents, i.e., we assumed entities of the type *Document*.

The accuracy of the proposed algorithm was compared against that of the centralized C-Search. In order to achieve this, the C-Search algorithm was implemented on top of Lucene [Luc], while the *Semantic Flooding* algorithm was implemented by simulating the underlying network on a single machine. The dataset for the evaluation was generated by mapping the public tags of about 950000 users from Delicious [Del] to the data from the Open Directory Project (DMOZ) [DMo]. Then, four data-sets were generated by randomly selecting sub-sets of 10, 100, 1000, and 10000 users. On the other hand, AOL query log [Pass et al., 2006] was used for the generation of query sets.



---

During the evaluation, the total number of queried peers was set to 10 for networks of 10 and 100 peers, and to 50 for networks of 1000 and 10000 peers. In fact, it was shown that on average only 50 peers need to be queried in order to achieve 70% of accuracy in a network of 10000 peers. Note that if we need to retrieve only one relevant result (i.e. 10% of accuracy), on average only one peer needs to be queried. Moreover, the accuracy of the approaches were also compared with popular and unpopular queries. Finally, it was shown that the accuracy of results for unpopular queries decreases substantially when using normal flooding while the proposed approach (*Semantic Flooding*) can still provide good accuracy.



## Part IV

# Conclusions



# Chapter 10

## Related work

The work presented in this thesis can be related with approaches from different areas, we focus mainly in three of them. First, we consider the approaches related with *contact and identity management* as a whole. Then, we discuss approaches that can be related with the two proposed search approaches, namely, *a name-based approach to search* and *a description-based approach to search*.

### 10.1 Contact and Identity Management

A number of existing applications deal with contact management on personal devices, focusing their attention on the data level. Some relevant examples are, Duplicate Contact Manager, Contacts+, GO Contacts, SIM Contacts Manager, TAP Contact Exchange, Copy Contacts, Linkle Contact Exchange. Among the variety of operations that they support, we can mention, creation of groups, importing of contacts from different accounts, matching of contacts, information exchange and some form of contact linkage. Although there is some shallow notion of multiple versions for the same contact in the existing applications, we see that most of the operations they support are mainly though in the context of services that integrate data from different devices of the same user in order to maintain them in synchrony. The navigation and search across different contact lists are not only not supported but they are not possible by design. In other words, by focusing on the data level, these approaches are not able to support complex reasoning on contacts and this can be considered the main difference with regard to the solution proposed in this thesis.

We can also find many web-based systems dealing with contact directories. They can support services that are similar to those offered by the contact management applications discussed above, actually they are sometimes integrated as the web versions of the applications for personal devices. However, we can also identify many web-based systems managing contacts with a focus on building datasets, which are then used to offer different types of services. Based on their scope we can distinguish between (i) those that work locally, for instance, within the boundaries of specific institutions (e.g., <https://directory.uchicago.edu/>, <http://web.mit.edu/people.html>); and (ii) those that work globally (e.g., <http://www.whitepages.com/>, <https://www.yellowpages.com/whitepages>, <https://www.yellowpages.com.au/>, <http://www.whitepages.com.au/>, <http://www.anywho.com/whitepages>). Our work differentiates from these approaches because its aim is not massive dataset building but we rather focus on the network of contacts that can emerge from linking individual contact lists of peers. Moreover, these approaches do not focus on the owners of the information, which prevents them from being in control of their data.

On the other hand, we have approaches that deal with identity management from different perspectives. Taking into account the goals that they focus on, we can classify them as follows:

- Focus on generation and assignment of identifiers Pane [2012]; Bouquet et al. [2008, 2007].
- Focus on the identity of individuals with regard to privacy, legal aspects and privacy-enhancing technologies Ahn et al. [2009]; PrimeLife [2011].
- Focus on user authentication and networking issues Jøsang and Pope [2005]; El Malki and Seigneur [2007].

Our approach is complementary to some of these approaches. The most relevant to our work is Pane [2012]. The main difference is give by the introduction of the notion of profiles associated to the representation of digital entities and our focus on search.

## 10.2 Distributed entity directory

In this section we discuss approaches that are capable of managing information about entities in a P2P network. More specifically, those that deals with the distributed indexing and searching of entities based on their identifiers. To the best of our knowledge, at the time of the analysis no approaches that integrates these areas can be found, i.e., that perform search of entities over a P2P network. Nevertheless, we give an overview of related approaches from both areas.

Some entity aware approaches concentrate the attention on the definition of models and structures for the representation of entities [Bazzanella et al., 2008]. In [Bouquet et al., 2008] an entity name system (ENS) is proposed in order to provide support for the generation and reuse of globally unique identifiers for entities across different and independent RDF repositories. The local repository of a single user is not considered as a source of data and the users need a special access permit in order to contribute with the definition of entities. As a first step towards searching, the work presented in [Bazzanella et al., 2009] proposes a model that analyzes the query specification and performs the disambiguation of the desired type of entity. In [Paşca, 2007], named entities are extracted by analyzing queries based on syntactic matching of patterns. These approaches do not directly address the search, but their results are relevant for the definition of the directory proposed in this thesis.

Other approaches that perform search following an entity centric perspective can be found in the literature [Cheng and Chang, 2007; Hu et al., 2006; Hogan et al., 2011]. Entity search engines are proposed in [Cheng and Chang, 2007; Hogan et al., 2011], heuristic rules are used in [Hu et al., 2006] to identify entities appearing in a collection of documents and a service to find documents that contain statements about particular resources is provided in Sindice [Oren et al., 2008]. Most of these approach collect data from multiple web sources (i.e., by crawling) but do not consider distribution at the level of single users (i.e., a P2P network). In particular, [Hogan et al., 2011] automatically aggregates descriptions from the different sources and allows subsequent navigation to related entities. Distribution is considered in terms of clusters of computers that allow parallel processing and scalable storage but the search is centralized (i.e., they build centralized indexes). In contrast to these approaches, our approach performs a distributed search in a P2P network and allows users to maintain their data locally.

On the other hand, we have P2P approaches, which perform distributed search but are not aware of entities [Risson and Moors, 2006; Lua et al., 2005]. They are mainly classified as unstructured and structured approaches. The first unstructured networks (e.g., Gnutella<sup>1</sup>) have scalability problems due to the number of messages generated and do not guarantee that all answers will be found. Other approaches use clustering techniques [Bawa et al., 2003; Cohen et al., 2003; Spripanidkulchai et al., 2003; Crespo and Garcia-Molina, 2005; Joseph, 2002], their goal is to find the best group to answer a query and then send the query to the peers in that group. Our approach can find all available answers and has proven to be promising in terms of scalability.

We can find also more structured approaches that aim to guarantee the location of the content shared on the network (e.g., CAN [Ratnasamy et al., 2001], Chord [Stoica

---

<sup>1</sup><http://en.wikipedia.org/wiki/Gnutella>

et al., 2001] Pastry [Druschel and Rowstron, 2001] and Tapestry [Zhao et al., 2004] They store pairs of  $\langle key, value \rangle$  in a Distributed Hash Table (DHT) and then retrieve the value associated with a given key. Other approaches perform multi-keyword search using DHTs but they can be very expensive in terms of required storage and generated traffic (e.g., see [Li et al., 2003]). Hierarchical structures combine clustering techniques with the structure of DHTs [Ganesan et al., 2004; Janakiram et al., 2011; Papapetrou et al., 2010; Garcés-Erice et al., 2003]. In general, P2P approaches provide the techniques needed in order to build our solution. The novelty of our approach is in the domain of application of such techniques.

### 10.3 Semantic flooding

A number of P2P search approaches have been proposed in the literature (for an overview see [Risson and Moors, 2006]). The algorithm implemented by Gnutella is the classical example of a query flooding algorithm. In early versions of Gnutella, connections between peers were made mainly chaotically. A P2P network was completely *unstructured*, i.e. it did not have any predefined structure. The query sent by a peer was propagated to all the actively connected peers within a predefined number of *hops* from the query sender. The search process was *blind*, i.e. peers have no information related to the resource location. The lack of scalability was recognized as the main problem of the Gnutella. Various techniques were adopted in later versions of the Gnutella protocol in order to make the search process more scalable. *Super-peers* were introduced to utilize the heterogeneity between peers in computer power, bandwidth and availability. *Informed search*, i.e. when peers maintain additional information about resource locations which can be useful for the search, replaced *blind search*. In Gnutella, informed search is implemented by using Query Routing Protocol (QRP). Query Routing Tables (QRT) consisting of hashed keywords are exchanged between peers. During query routing, search request is propagated only to those peers which have all of the query words in its QRT. In [Crespo and Molina, 2002], a peer uses Routing Indices to forward queries to neighbors that are more likely to have answers. Query topics are compared to neighbor's expertise to select relevant peers. In our approach, search for relevant peers is implemented by using semantic links created between nodes in classifications of different peers.

The basic idea of [Bawa et al., 2003; Cohen et al., 2003; Spripanidkulchai et al., 2003; Zhu and Hu., 2004; Crespo and Garcia-Molina, 2005; Joseph, 2002] is to organize peers into Similar Content Groups on top of unstructured P2P systems, i.e. a *peer clustering* approach is implemented. Peers from the same group tend to be relevant to the same queries. A query is guided to Similar Content Group that is more likely to have answers



to the given query and then the query is flooded within this group. For instance, in Semantic Overlay Networks (SONs) [Crespo and Garcia-Molina, 2005], peers that have similar documents are clustered at the same group. A predefined classification hierarchy is used to classify the peers' documents. Thus two peers belong to the same SON if some of their documents classified under the same concept in this global classification. Peers can belong to more than one SON. In our approach, peers with similar content are connected to each other by creating semantic links between nodes in classifications of these peers. Differently from [Crespo and Garcia-Molina, 2005], a global classification is not required and users are free to create their own classification hierarchies.

As mentioned in the previous section, CAN [Ratnasamy et al., 2001], Chord [Stoica et al., 2001], Pastry [Druschel and Rowstron, 2001], and Tapestry [Zhao et al., 2001] use another approach to the routing and topology organization of P2P networks, and they are highly structured. Here we analyze them under the light of a semantic flooding approach. The topology is tightly controlled and documents (or information about documents) are placed at the precisely specified locations defined by their keys. A *data clustering* approach is implemented, i.e. similar data (meta-data) is placed in the same place. Search in these systems is limited to an exact key search. Mercury [Bharambe et al., 2004] supports multi-attribute range queries, e.g. each query is a conjunction of ranges in one or more attributes. Examples of how a full text retrieval can be implemented on top of structured P2P networks are described in [Li et al., 2003; Luu et al., 2008; Bender et al., 2005; Tang et al., 2003]. A straightforward way to implement syntactic search is to use the DHT to distribute peers' inverted indices in the P2P network [Risson and Moors, 2006]. In order to find a set of documents which contain a term, the peer responsible for this term has to be contacted and the corresponding posting list need to be retrieved. In order to search for more than one term, first, the posting list for every single term need to be retrieved, and then all these posting lists have to be intersected. The above approach can potentially be very expensive in terms of required storage and generated traffic (see e.g. [Li et al., 2003]). For instance, posting lists need to be transferred when peers join or leave the network. Searching with multiple terms requires intersection of posting lists, which also need to be transferred. In the case of huge posting lists, a bandwidth consumption can exceed the maximum allowed limits. In [Li et al., 2003], it is shown that the efficiency of DHT can be even worse than the efficiency of a simple flooding algorithm. Some of optimization techniques (e.g. Bloom Filters), which can improve the performance of posting lists intersecting, are summarized in [Li et al., 2003]. In [Luu et al., 2008], indexing is performed by terms and term sets appearing in a limited number of documents. Different filtering techniques are used in [Luu et al., 2008] in order to make vocabulary to grow linearly with respect to the document collection size. In Minerva

[Bender et al., 2005] DHT holds only compact, aggregated meta-information about the peers' local indexes which is used to efficiently select promising peers from across the peer population that can best locally execute a query. In our approach, a DHT based P2P Concept Search is used only for indexing and retrieval of nodes from classifications and not documents. The problem with storage is reduced since inverted indices for nodes are usually smaller than those for documents which are classified to these nodes. Moreover, the generated traffic is reduced because, in P2P Concept Search, a query consisting of a single complex concept do not require the intersection of inverted indices.

All of the described so far approaches are based on syntactic matching of words and, therefore, the quality of results produced by these approaches can be negatively affected by the problems related to the ambiguity of natural language. Some P2P search approaches use matching techniques which are based on the knowledge about term relatedness (and not only syntactic similarity of terms). For instance, statistical knowledge about term co-occurrence is used in [Tang et al., 2003]. Knowledge about synonyms and related terms is used in [Ma et al., 2007]. In our approach, the problem of ambiguity of natural language is dealt with by using semantic matching of complex concepts. Different semantic search approaches are also used in [Zhuge et al., 2005; Xiao and Cruz, 2006; Nejdl et al., 2002; Haase et al., 2004; Löser et al., 2010]. A semantic link P2P network (P2PSLN) [Zhuge et al., 2005] specifies and manages semantic relationships between peers' data schemas. A semantic-based peer similarity measurement is used for efficient query routing. A schema mapping algorithm is used for query reformulation and heterogeneous data integration. Ontology-based P2P data management system (OPDMS) [Xiao and Cruz, 2006] is based on ontology mapping and query processing. Edutella [Nejdl et al., 2002] and Bibster [Haase et al., 2004] are built on JXTA framework and aim to combine meta-data with P2P networks. Each peer is described and published using an advertisement, which is an XML document describing a network resource. For example in the Bibster [Haase et al., 2004] system, these expertise descriptions contain a set of topics that the peer is an expert in. Peers use a shared ontology to advertise their expertise in the Peer-to-Peer network. INGA [Löser et al., 2010] creates personal shortcuts by analyzing the queries issued by the local peer and the queries that are routed through the local peer. Query routing is made by analyzing shortcuts and their similarity to the query and a common topic hierarchy is assumed for the evaluation of this similarity. In our approach, semantic links are created between semantically related nodes in classifications of different peers and not between data schemas of the peers, as in [Zhuge et al., 2005]. Moreover, differently from [Haase et al., 2004; Löser et al., 2010], our approach does not assume a shared ontology.

In Table 10.1, we provide a summary of the search methods discussed in this section.

Table 10.1: Search Methods in P2P networks.

	Network structure	Clustering	Identifying semantically relevant peers	Search method
Gnutella [Risson and Moors, 2006]	Unstructured	-	Blind	Keyword
Routing Indices [Crespo and Molina, 2002]	Unstructured	-	Informed	Keyword
SETS [Bawa et al., 2003]	Unstructured	Peers	Informed	Keyword
Associative overlay [Cohen et al., 2003]	Unstructured	Peers	Informed	Keyword
Interest-based overlay [Spripanidkulchai et al., 2003]	Unstructured	Peers	Informed	Keyword
ESS [Zhu and Hu., 2004]	Unstructured	Peers	Informed	Keyword
SONs [Crespo and Garcia-Molina, 2005]	Unstructured	Peers	Informed	Keyword

NeuroGrid [Joseph, 2002]	Unstructured	Peers	Informed	Keyword
P2PSLN [Zhuge et al., 2005]	Unstructured	Peers	Informed	Semantic
OPDMS [Xiao and Cruz, 2006]	Unstructured	Peers	Informed	Semantic
INGA [Löser et al., 2010]	Unstructured	Peers	Informed	Semantic
EDUTELA [Nejdl et al., 2002]	Hybrid	Peers	Informed	Semantic
Bibster [Haase et al., 2004]	Hybrid	Peers	Informed	Semantic
pSearch [Tang et al., 2003]	Structured	Data	Informed	Semantic
Concept Index in P2P [Ma et al., 2007]	Structured	Data	Informed	Semantic
CAN [Ratnasamy et al., 2001]	Structured	Data	Informed	Key
Chord [Stoica et al., 2001]	Structured	Data	Informed	Key

Pastry [Druschel and Rowstron, 2001]	Structured	Data	Informed	Key
Tapestry [Zhao et al., 2001]	Structured	Data	Informed	Key
Mercury [Bharambe et al., 2004]	Structured	Data	Informed	Keyword
MINERVA [Bender et al., 2005]	Structured	Data	Informed	Keyword
AlvisP2P [Luu et al., 2008]	Structured	Data	Informed	Keyword



# Chapter 11

## Conclusions and future work

### 11.1 The context

This PhD Thesis deals with the problem of contact and identity management in a world of people relying on personal devices for their every day communication activities. In particular, we focus the attention directories of contacts or contact lists used to organize information in a network of peers. Within this context, the foundational notions for this thesis were first introduced in order to use them for building the Distributed Contact Management System (DCM system) on top of them. Namely,

- A *contact* was defined as an entity from the real world that is somehow contactable by possibly diverse means;
- A *peer* was defined as a user of the system, maintaining a contact list and capable of participating in communication activities; and
- An *entity* represents an abstract of physical object that exist in the real world, it has a type and is described by a set of attributes.

In this manner, an entity-centric approach was adopted for the representation of contacts and peers.

Related with the adoption of a model based on entities for representing information in the system, the notion of a knowledge schema was presented as a mean to achieve interoperability between peers. The knowledge schema defines templates for the different types of entities used in the system, establishing restrictions on the set of attributes used to describe a given type. These templates are then instantiated into Digital Entities

(*DEs*) and their Attributes (*As*) to actually represent knowledge about entities from the real world. Two types of identifiers were introduced in association to entities in order to distinguish among many possibly different descriptions of the same real world entity. A semantic URL (*SURL*) represents a particular description, while a semantic URI (*SURI*) represents the actual entity without attaching it to any specific description.

Then, it was also discussed how the hierarchical structure of classifications (also called lightweight ontologies [Giunchiglia and Zaihrayeu, 2008]) can be exploited to organize contacts. On one hand, the notion of classification of subjects was presented as a mean to organize contacts with whom the user is connected through social ties. On the other hand, the classification of objects was presented to organize contacts based on their characteristics (i.e., as objects described by certain attributes).

## 11.2 The contributions

Taking into account the described context, this thesis considered two dimensions of the problem of managing peers contacts. First, the local dimension of peers needing to organize and find information that allow them to contact other peers, and second, the global dimension of peers having multiple identities and needing to stay in control of them. Four contributions were presented to address this twofold problem.

**The reference architecture for the DCM System.** First, the analysis of the system requirements was presented. The outcome of such analysis was discussed in terms of: (i) data storage for an inherently distributed scenario, (ii) peers interaction and linking, (iii) services that the system needs to offer, (iv) possible privacy concerns that may arise, and the (v) the system performance.

Taking into consideration the identified requirements a *general view* of the system was presented. In it, the different (external) actors that can interact with the DCM system were defined, as well as the nature and mechanisms for these interactions. Next, the system *logical view* introduced the DCM Portal, DCM App and DCM Network as key system components allowing peers to create accounts, interact with each other and access to different type of services. Last, a *dynamic view* of the system was presented, showing a simplified view of different interactions among system components. Three types of interactions were distinguished: initialization, sharing and search.

**A distributed directory of entities.** We presented an approach to build a distributed directory of entities in the DCM system that distinguishes between the notions of Digital Entity (DE) and Real World Entity (WE) in order to link local directories of



different peers. The directory provides search services based on entity identifiers. In particular, we presented the algorithms for searching entities based on their names. We discussed the name matching problem that appears as a consequence of the *many-to-many* relation between names and (WEs). Then, we showed that, by its design, our approach deals with the problem of matching names inside the network (i.e., the first part of the name matching problem).

The data from peers are stored locally, only the identifiers and the links to the local representations are indexed. This infrastructure allows the implementation of access control mechanisms on the local representations in order to deal with privacy issues. At the same time, the changes made by peers in local representations, are available in the directory in a straightforward manner. Moreover, these features of the approach are independent from the specific underlying implementation of the indexes. In other words, the indexes can be stored in a centralized or distributed manner, while data will be still distributed.

**A semantic overlay linking directories of different peers.** It was shown how the notion of classification of objects COs, as defined in Section 3.4, can be exploited to build a semantic overlay linking directories of different peers in the DCM network. The set of links which connect nodes inside a classification plus C-OWL links across classifications constitute a semantic overlay network which can be built on top of any underlying set of peers and their physical connections. Thus allowing peers to semantically search contacts that are distributed in the DCM network and have certain characteristics.

In order to build a semantic overlay, we discussed how new semantic links can be automatically computed by using semantic matching (S-Match) approach between two known classifications. This was shown to be particularly relevant when a new peer joins the network or when a new node is created in a classification. When relevant classifications are not known the user can select a subset of the known peers, by selecting a node from its classification of subjects CS or from its classification of objects CO, and run S-Match with their classifications. As another alternative we proposed to use the distributed entity directory to find other peers in the network having relevant information but that are not in the local contact list, and run S-Match with their classifications.

Next, we presented an implementation of the approach that is based on entities of the type *Document* and decomposes the problem into three subproblems. The first, identifying semantically relevant peers, defined as those having nodes in their classifications which are relevant to the search request. The second, searching inside

relevant peers, which is done by using *C-Search* [Giunchiglia et al., 2009b]. And the third, aggregation of the search results, which includes merging query answers from different peers into a single query answer and computing a relevance score for each answer. The relevance score in the implementation presented is done using cosine similarity and tf-idf weight measure used in *Lucene* [Luc].

Similar to the entity directory, also in this approach the data from peers are stored locally. The links to classifications from other peers are also stored locally. This means that the modifications made by peers to the local representation of their contacts are available in the semantic overlay network also in a straightforward manner. Moreover, this infrastructure can also benefit from the implementation of access control mechanisms on the local representations in order to deal with privacy issues.

**The Distribute Contact Management (DCM) System.** The DCM System was presented to enclose the reference architecture and the search approaches previously discussed. In order to do so, two application specific notions were introduced, Presentation Card (PC) and DCM User.

The PCs were formally defined to support the exchanging of contact information between peers. A PC denotes a profile created to present one view (perspective) of a particular contact, represented in the system as an entity DE. On the other hand, a DCM User was defined as a peer having an account in the DCM platform.

Then, different types of usage scenarios illustrating the main functionalities and features of the system were presented. Namely,

- The initialization scenarios correspond to the first interactions that the peers have with the system, when the peer is starting to use the DCM system. They describe activities like, creation of an user account, creation of presentation cards, importing contacts, and creating them by hand.
- The sharing scenarios correspond to interactions that peers can have with other peers (users and not users of the system) using the DCM application. They include activities related to sharing, publishing and exchanging the own contact information as well as the contact of third parties.
- The search scenarios, show examples of the envisioned search features of the proposed application including activities like searching contacts by name, based on existing contact information, and by description.

### 11.3 The evaluations

Experimental evaluations were performed for the two search approaches of the DCM system. Namely, the name-based approach proposed to search for contacts in a distributed directory of entities, and the description-based approach proposed to search contacts in a semantic overlay of linked directories from different peers.

**The evaluation of the name-based approach.** The implementation was done using a Distributed Hash Table (DHT) for the storage of indexes. Two DHT protocols were mainly considered, Chord and Kademia, the latter was selected because of several features that make it better for real applications. Next, a comparison of existing implementations of the Kademia protocol was presented. TomP2P, an advanced DHT library that extends the basic functions of DHTs, was selected.

The evaluation was done using *PlanetLab* as a testbed. PlanetLab provides a network of computers that are distributed around the world, providing a realistic scenario in terms of network conditions. Networks of 50, 100 and 150 peers were configured assuming that the local entity base of peers contains, in average, 2000 *DEs*. When running the evaluation process, the peers are activated pseudo-simultaneously and execute a set 1400 queries randomly selected from an initial set of entity names. Once all the peers ended the search process, the average query time for the network was computed.

It was shown that the average query time (2.7 seconds) was stable with regard to the network growth, which is considered a promising result for the scalability of the directory. Moreover, the distribution of the query time with regard to the network growth was also stable in networks of different sizes. It was also noted that more than 55% of queries were actually computed in less than a second and 70% of them in less than 2 seconds.

**The evaluation of the description-based approach.** It was performed by simulating a distributed yellow pages for contacts of peers, which creates a semantic overlay that links directories of different peers in the DCM network. This semantic overlay was then flooded to search contacts based on their descriptions in a distributed manner, i.e., a *Semantic Flooding* was performed. It is important to note that the evaluation of this approach was applied to documents, i.e., we assumed entities of the type *Document*.

The accuracy of the proposed algorithm was compared against that of the centralized *C-Search*. In order to achieve this, the *C-Search* algorithm was implemented on top of Lucene [Luc], while the *Semantic Flooding* algorithm was implemented by simulating

the underlying network on a single machine. The dataset for the evaluation was generated by mapping the public tags of about 950000 users from Delicious [Del] to the data from the Open Directory Project (DMOZ) [DMo]. Then, four data-sets were generated by randomly selecting sub-sets of 10, 100, 1000, and 10000 users. On the other hand, AOL query log [Pass et al., 2006] was used for the generation of query sets.

During the evaluation, the total number of queried peers was set to 10 for networks of 10 and 100 peers, and to 50 for networks of 1000 and 10000 peers. In fact, it was shown that on average only 50 peers need to be queried in order to achieve 70% of accuracy in a network of 10000 peers. Note that if we need to retrieve only one relevant result (i.e. 10% of accuracy), on average only one peer needs to be queried. Moreover, the accuracy of the approaches were also compared with popular and unpopular queries. Finally, it was shown that the accuracy of results for unpopular queries decreases substantially when using normal flooding while the proposed approach (*Semantic Flooding*) can still provide good accuracy.

## 11.4 Future work

A number of paths with opportunities to extend the scope of this thesis were left for future work<sup>1</sup>, either for lack of time or limitation of resources. In what follows we describe some of these paths.

The first and most obvious is to analyse and implement possible improvements to proposed approaches, as well as their evaluations. For instance, in the name-based search, we would like to integrate techniques that can reduce the effects at query time of having slow peers in the network. We are also interested in returning results in real time, as lookups are completed, in order to avoid waiting until all the lookups in the query have ended. In the case of the description-based search, we performed the evaluation by simulating the network itself. We are interested to see how the approach performs, both in terms of query time and the cost of maintaining the semantic overlay, in more realistic settings.

Another aspect that we are interested is the exploration of other, possibly broader, scenarios. This would include, the application of the proposed models and search approaches. For instance, an interesting scenario to consider would be the extension of the notion of contact to profiles of peers in terms of their knowledge, capabilities, resources and other aspects. In this way, peer search approaches would be enabled to find peers based on characteristics that allow them to perform certain activities or provide a given

---

<sup>1</sup>Future work is highly framed in the context of SmartSociety Project (EU FP7 Grant n. 600854, <http://www.smart-society-project.eu/>).

---

service. On one hand the main challenge to address in this scenario is scalability, since is not only contact information that we will be dealing with. On the other hand, in our approach data from profiles are maintain in the local directories of their owners and the extra load is given only by indexes.

By last, we mentioned throughout the thesis that our work takes into consideration main privacy principles, proposing a privacy-friendly design that facilitate the adoption of privacy enhancing technologies. As part of the future work, we are interested in extending the proposed approaches to integrate privacy preserving tools. This requires a more deeply study of privacy issues in the context of the DCM system, which may in turn require a re-design of some of the proposed models. Moreover, the outcome of such study should result in the integration of mechanisms to represent, understand and enforce privacy policies. Although we are aware of the challenges of this line of work, we also believe it represents the most interesting path for a future work.



# Bibliography

Dmoz: <http://www.dmoz.org/>.

Delicious: <http://www.delicious.com/>.

Gnutella. URL <http://en.wikipedia.org/wiki/Gnutella>.

Lucene: [http://lucene.apache.org/java/2\\_4\\_0/api/org/apache/lucene/search/Similarity.html](http://lucene.apache.org/java/2_4_0/api/org/apache/lucene/search/Similarity.html).

Ahn, Gail-Joon; Ko, Moonam, and Shehab, M. Privacy-enhanced user-centric identity management. In *Communications, 2009. ICC '09. IEEE International Conference on*, pages 1–5, June 2009. doi: 10.1109/ICC.2009.5199363.

Bawa, M.; Manku, G., and Raghavan, P. Sets: Search enhanced by topic segmentation. In *Proceedings of ACM SIGIR Conference*, pages 306–313, 2003.

Bazzanella, Barbara; Chaudhry, Junaid Ahsenali; Themis Palpanas, , and Stoermer, Heiko. Towards a General Entity Representation Model. *5th Workshop on SWAP*, 2008. URL <http://disi.unitn.it/~themis/publications/swap08.pdf>.

Bazzanella, Barbara; Stoermer, Heiko, and Bouquet, Paolo. Searching for individual entities: a query analysis. Technical report, University of Trento, 2009.

Bender, Matthias; Michel, Sebastian; Triantafillou, Peter; Weikum, Gerhard, and Zimmer, Christian. Minerva: Collaborative p2p search. In *Proceedings of VLDB*, pages 1263–1266, 2005.

Bharambe, A.; Agrawal, M., and Seshan, S. Mercury: Supporting multi-attribute range queries. In *Proceedings of ACM SIGCOMM*, 2004.

Bignotti, Enrico. Semantic name matching. Master's thesis, University of Trento, 2012.

Borgida, Alexander; Walsh, Thomas, and Hirsh, Haym. Towards measuring similarity in description logics. In *Proceedings of the 2005 International Workshop on Description Logics (DL2005)*, 2005.

Bouquet, P.; Giunchiglia, F.; van Harmelen, F.; Serafini, L., and Stuckenschmidt, H. Contextualizing ontologies. In *Journal of Web Semantics*, volume 1, pages 325–343, 2004.

Bouquet, Paolo; Stoermer, Heiko, and Giacomuzzi, Daniel. Okkam: Enabling a web of entities. *I3*, 5:7, 2007.

Bouquet, Paolo; Stoermer, Heiko; Niederee, Claudia, and Maña, Antonio. Entity name system: The back-bone of an open and scalable web of data. In *Proceedings of the 2nd IEEE ICSC*, pages 554–561, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-0-7695-3279-0. doi: 10.1109/ICSC.2008.37. URL <http://portal.acm.org/citation.cfm?id=1446294.1446425>.

- Camp, L.J. Digital identity. *Technology and Society Magazine, IEEE*, 23(3):34–41, Fall 2004. ISSN 0278-0097. doi: 10.1109/MTAS.2004.1337889.
- Cheng, Tao and Chang, Kevin Chen-Chuan. Entity search engine: Towards agile best-effort information integration over the web. In *CIDR 2007*, pages 108–113, 2007.
- Cohen, E.; Kaplan, H., and Fiat, A. Associative search in peer to peer networks: Harnessing latent semantics. In *Proceedings of IEEE INFOCOM*, 2003.
- Crespo, A. and Molina, H. Garcia. Routing indices for peer-to-peer systems. In *Proceedings of the 22nd IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2002.
- Crespo, Arturo and Garcia-Molina, Hector. Semantic overlay networks for p2p systems. In Moro, Gianluca; Bergamaschi, Sonia, and Aberer, Karl, editors, *Agents and Peer-to-Peer Computing*, volume 3601 of *Lecture Notes in Computer Science*, pages 1–13. Springer Berlin / Heidelberg, 2005. URL [http://dx.doi.org/10.1007/11574781\\_1](http://dx.doi.org/10.1007/11574781_1). 10.1007/11574781\_1.
- Do Van Thanh, Ivar Jørstad. The ambiguity of identity. *Identity Management*, page 3, 2007.
- Druschel, P. and Rowstron, A. Pastry: scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proc. of ACM SIGCOM*, 2001.
- El Maliki, T. and Seigneur, J. M. A survey of user-centric identity management technologies. In *Emerging Security Information, Systems, and Technologies, 2007. SecureWare 2007. The International Conference on*, pages 12–17, Oct 2007. doi: 10.1109/SECUREWARE.2007.4385303.
- European Commission, . Directive 95/46/ec of the european parliament and of the council of 24 october 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data, November 1995. URL <http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:31995L0046:en:HTML>.
- European Commission, . Proposal for a regulation of the european parliament and of the council on the protection of individuals with regard to the processing of personal data and on the free movement of such data (general data protection regulation), com(2012) 11 final 2012/0011 (cod), January 2012. URL [http://ec.europa.eu/justice/data-protection/document/review2012/com\\_2012\\_11\\_en.pdf](http://ec.europa.eu/justice/data-protection/document/review2012/com_2012_11_en.pdf).
- Ganesan, Prasanna; Gummadi, Krishna, and Garcia-Molina, H. Canon in g major: designing dhds with hierarchical structure. In *ICDCS'04*, pages 263 – 272, 2004. doi: 10.1109/ICDCS.2004.1281591.
- Garcés-Erice, Luis; Biersack, Ernst W.; Felber, Pascal; Ross, Keith W., and Urvoy-Keller, Guillaume. Hierarchical peer-to-peer systems. In *Euro-Par*, pages 1230–1239, 2003.
- Giunchiglia, Fausto and Hume, Alethia. Distributed name-based entity search. In *Workshop on Discovering Meaning On the Go in Large and Heterogeneous Data (LHD-12)*, 2012.
- Giunchiglia, Fausto and Hume, Alethia. A distributed directory system. In *9th International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2013)*, page 97, 2013a.
- Giunchiglia, Fausto and Hume, Alethia. A distributed entity directory. In *The Semantic Web: ESWC 2013 Satellite Events*, pages 291–292. Springer, 2013b.
- Giunchiglia, Fausto and Zaihrayeu, Ilya. Lightweight ontologies. In *The Encyclopedia of Database Systems*, 2008.



- Giunchiglia, Fausto; Shvaiko, Pavel, and Yatskevich, Mikalai. Discovering missing background knowledge in ontology matching. In *Proc. of ECAI*, pages 382–386, 2006.
- Giunchiglia, Fausto; Yatskevich, Mikalai, and Shvaiko, Pavel. Semantic matching: Algorithms and implementation. *Journal on Data Semantics (JoDS)*, 9:1–38, 2007a.
- Giunchiglia, Fausto; Zaihrayeu, Ilya, and Kharkevich, Uladzimir. Formalizing the get-specific document classification algorithm. In Kovács, László; Fuhr, Norbert, and Meghini, Carlo, editors, *ECDL*, volume 4675 of *Lecture Notes in Computer Science*, pages 26–37. Springer, 2007b. ISBN 978-3-540-74850-2.
- Giunchiglia, Fausto; Zhang, Rui, and Crispo, Bruno. Relbac: Relation based access control. In *Proceedings of the 2008 Fourth International Conference on Semantics, Knowledge and Grid, SKG '08*, pages 3–11, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-0-7695-3401-5. doi: 10.1109/SKG.2008.76. URL <http://dx.doi.org/10.1109/SKG.2008.76>.
- Giunchiglia, Fausto; Kharkevich, Uladzimir, and Noori, Sheak Rashed Haider. P2P Concept Search: Some preliminary results. In *SemSearch2009 workshop at WWW*, 2009a.
- Giunchiglia, Fausto; Kharkevich, Uladzimir, and Zaihrayeu, Ilya. Concept search. In Aroyo, Lora; Traverso, Paolo; Ciravegna, Fabio; Cimiano, Philipp; Heath, Tom; Hyvonen, Eero; Mizoguchi, Riichiro; Oren, Eyal; Sabou, Marta, and Simperl, Elena, editors, *The Semantic Web: Research and Applications*, volume 5554 of *Lecture Notes in Computer Science*, pages 429–444. Springer Berlin Heidelberg, 2009b. ISBN 978-3-642-02120-6. doi: 10.1007/978-3-642-02121-3\_33. URL [http://dx.doi.org/10.1007/978-3-642-02121-3\\_33](http://dx.doi.org/10.1007/978-3-642-02121-3_33).
- Giunchiglia, Fausto; Kharkevich, Uladzimir; Hume, Alethia, and Chatvorawit, Piyatat. Semantic flooding: search over semantic links. In *Data Engineering Workshops (ICDEW), 2010 IEEE 26th International Conference on*, pages 191–196. IEEE, 2010.
- Giunchiglia, Fausto; Kharkevich, Uladzimir, and Hume, Alethia. Semantic flooding: Semantic search across distributed lightweight ontologies. *World Wide Web*, pages 1–19, 2011. ISSN 1386-145X. URL <http://dx.doi.org/10.1007/s11280-010-0108-y>. <http://dx.doi.org/10.1007/s11280-010-0108-y>.
- Giunchiglia, Fausto; Dutta, Biswanath, and Maltese, Vincenzo. From knowledge organization to knowledge representation. *Knowledge Organization*, 41(1), 2014.
- Giunchiglia, Fausto; Marchese, Maurizio, and Zaihrayeu, Ilya. Encoding classifications into lightweight ontologies. In *Journal on Data Semantics (JoDS) VIII*, Winter 2006.
- Haase, Peter; Broekstra, Jeen; Ehrig, Marc; Menken, Maarten; Mika, Peter; Plechawski, Michal; Pyszlak, Pawel; Schnizler, Björn; Siebes, Ronny; Staab, Steffen, and Tempich, Christoph. Bibster - a semantics-based bibliographic peer-to-peer system. In *Proceedings of the 3rd ISWC*, pages 122–136, 2004.
- Hogan, Aidan; Harth, Andreas; Umbrich, JÃrger; Kinsella, Sheila; Polleres, Axel, and Decker, Stefan. Searching and browsing linked data with swse: The semantic web search engine. *JWS: Science, Services and Agents on the World Wide Web*, 9(4):365 – 401, 2011. ISSN 1570-8268. doi: 10.1016/j.websem.2011.06.004. URL <http://www.sciencedirect.com/science/article/pii/S1570826811000473>.
- Hogan, Aidan; Zimmermann, Antoine; Umbrich, Juergen; Polleres, Axel, and Decker, Stefan. Scalable and distributed methods for entity matching, consolidation and disambiguation over linked data corpora. *JWS: Science, Services and Agents on the World Wide Web*, 10, 2012. ISSN 1570-8268. URL <http://www.websemanticsjournal.org/index.php/ps/article/view/224>.

- Holloway, Geoff and Dunkerley, Mike. *The Math, Myth and Magic of Name Search and Matching*. Search Software America, 5th edition, 2004.
- Hu, Guoping; Liu, Jingjing; Li, Hang; Cao, Yunbo; Nie, Jian-Yun, and Gao, Jianfeng. A supervised learning approach to entity search. In *AIRS'06*, volume 4182 of *LNCS*, pages 54–66. 2006. URL [http://dx.doi.org/10.1007/11880592\\_5](http://dx.doi.org/10.1007/11880592_5).
- Janakiram, Dharanipragada; Giunchiglia, Fausto; Haridas, Harisankar, and Kharkevich, Uladzimir. Two-layered architecture for peer-to-peer concept search. In *4th Int. Sem Search Workshop*, 2011.
- Jøsang, Audun and Pope, Simon. User centric identity management. In *AusCERT Asia Pacific Information Technology Security Conference*, page 77. Citeseer, 2005.
- Joseph, Sam. Neurogrid: Semantically routing queries in peer-to-peer networks. In Gregori, Enrico; Cherkasova, Ludmila; Cugola, Gianpaolo; Panzieri, Fabio, and Picco, Gian, editors, *Web Engineering and Peer-to-Peer Computing*, volume 2376 of *Lecture Notes in Computer Science*, pages 202–214. Springer Berlin / Heidelberg, 2002. doi: 10.1007/3-540-45745-3\_18. URL [http://dx.doi.org/10.1007/3-540-45745-3\\_18](http://dx.doi.org/10.1007/3-540-45745-3_18).
- Li, Jinyang; Thau, Boon; Joseph, Loo; Hellerstein, M., and Kaashoek, M. Frans. On the feasibility of peer-to-peer web indexing and search. In *IPTPS'03*, 2003.
- Löser, Alexander; Staab, Steffen, and Tempich, Christoph. Semantic social overlay networks. In Shen, Xuemin; Yu, Heather; Buford, John, and Akon, Mursalin, editors, *Handbook of Peer-to-Peer Networking*, pages 189–219. Springer US, 2010. ISBN 978-0-387-09751-0. URL [http://dx.doi.org/10.1007/978-0-387-09751-0\\_8](http://dx.doi.org/10.1007/978-0-387-09751-0_8). 10.1007/978-0-387-09751-0\_8.
- Lua, Eng Keong; Crowcroft, Jon; Pias, Marcelo; Sharma, Ravi, and Lim, Steven. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys and Tutorials*, 7:72–93, 2005.
- Luu, Toan; Skobeltsyn, Gleb; Klemm, Fabius; Puh, Maroje; Žarko, Ivana Podnar; Rajman, Martin, and Aberer, Karl. AlvisP2P: scalable peer-to-peer text retrieval in a structured p2p network. In *Proc. VLDB Endow.*, 2008.
- Ma, Wenhui; Fang, Wenbin; Wang, Gang, and Liu, Jing. Concept index for document retrieval with peer-to-peer network. In *Proc. SNPD '07*, 2007. ISBN 0-7695-2909-7. doi: <http://dx.doi.org/10.1109/SNPD.2007.216>.
- Nejdl, W.; Wolf, B.; Qu, C.; Decker, S.; Sintek, M.; Naeve, A.; Nilsson, M.; Palmer, M., and Risch, T. Edutella: A p2p networking infrastructure based on rdf. In *Proceedings of WWW'02*, 2002.
- Oren, E.; Delbru, R.; Catasta, M.; Cyganiak, R.; Stenzhorn, H., and Tummarello, G. Sindice.com: a document-oriented lookup index for open linked data. *International Journal of Metadata, Semantics and Ontologies*, 3 (1):37–52, 2008.
- Paşca, Marius. Weakly-supervised discovery of named entities using web search queries. In *Proceedings of the sixteenth ACM conference on CIKM '07*, pages 683–690, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-803-9. doi: <http://doi.acm.org/10.1145/1321440.1321536>. URL <http://doi.acm.org/10.1145/1321440.1321536>.
- Pane, Juan. *Distributed Identity Management*. Phd thesis, University of Trento, 2012.
- Papapetrou, Odysseas; Siberski, Wolf, and Nejdl, Wolfgang. Peir: Combining dhds and peer clusters for efficient full-text p2p indexing. *Computer Networks*, 54(12):2019–2040, 2010.
- Pass, Greg; Chowdhury, Abdur, and Torgeson, Cayley. A picture of search. In *InfoScale'06: Proceedings of the 1st international conference on Scalable information systems*, New York, NY, USA, 2006. ACM.

- PrimeLife, . PrimeLife - Privacy and Identity Management in Europe for Life, Policy Languages. Available from <http://primelife.ercim.eu/images/stories/primer/policylanguage-plb.pdf>, 2011.
- Ratnasamy, Sylvia; Francis, Paul; Handley, Mark; Karp, Richard, and Shenker, Scott. A scalable content-addressable network. In *Proc. of SIGCOMM'01*, pages 161–172, NY, USA, 2001. ACM. ISBN 1-58113-411-8. doi: <http://doi.acm.org/10.1145/383059.383072>. URL <http://doi.acm.org/10.1145/383059.383072>.
- Rhea, Sean; Chun, Byung-Gon; Kubiawicz, John, and Shenker, Scott. Fixing the embarrassing slowness of opendht on planetlab. In *Proc. of the 2nd conference on Real, Large Distributed Systems, WORLDS'05*, pages 25–30, Berkeley, CA, USA, 2005. URL <http://dl.acm.org/citation.cfm?id=1251522.1251527>.
- Risson, John and Moors, Tim. Survey of research towards robust peer-to-peer networks: Search methods. *Computer Networks*, 50:3485–3521, 2006. doi: <http://dx.doi.org/10.1016/j.comnet.2006.02.001>. URL <http://dx.doi.org/10.1016/j.comnet.2006.02.001>.
- Spridanidkulchai, Kunwadee; Maggs, Bruce, and Zhang, Hui. Efficient content location using interest-based locality in peer-to-peer systems. In *Proceedings of IEEE INFOCOM*, volume 3, pages 2166–2176, 2003.
- Stoica, Ion; Morris, Robert; Karger, David; Kaashoek, M. Frans, and Balakrishnan, Hari. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proc. of SIGCOMM'01*, pages 149–160, NY, USA, 2001. ACM.
- Tang, Chunqiang; Xu, Zhichen, and Dwarkadas, Sandhya. Peer-to-peer information retrieval using self-organizing semantic overlay networks. In *Proceedings of ACM SIGCOMM*, pages 175–186, 2003.
- Wetzker, R.; Zimmermann, C., and Bauckhage, C. Analyzing social bookmarking systems: A delicious cookbook. In *Mining Social Data (MSoDa) Workshop Proceedings, ECAI*, pages 26–30, 2008. URL [http://www.dai-labor.de/fileadmin/files/publications/wetzker\\_delicious\\_ecai2008\\_final.pdf](http://www.dai-labor.de/fileadmin/files/publications/wetzker_delicious_ecai2008_final.pdf).
- Windley, Phillip. *Digital Identity*. O'Reilly Media, Inc., 2005. ISBN 0596008783.
- Xiao, H. and Cruz, I. F. Ontology-based query rewriting in peer-to-peer networks. In *Proceedings of the 2nd Int. Conf. on Knowledge Engineering and Decision Support*, pages 11–18, 2006.
- Zaihrayeu, I.; Sun, L.; Giunchiglia, F.; Pan, W.; Ju, Q.; Chi, M., and Huang, X. From web directories to ontologies: Natural language processing challenges. In *6th International Semantic Web Conference (ISWC 2007)*. Springer, 2007.
- Zhao, B. Y.; Kubiawicz, J., and Joseph, A. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical report, Computer Science Department, University of California, 2001.
- Zhao, B.Y.; Huang, L.; Stribling, J.; Rhea, S.C.; a.D. Joseph, , and Kubiawicz, J.D. Tapestry: A Resilient Global-Scale Overlay for Service Deployment. *IEEE Journal on Selected Areas in Communications*, 22(1): 41–53, January 2004. ISSN 0733-8716. doi: 10.1109/JSAC.2003.818784. URL <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1258114>.
- Zhu, Yingwu and Hu., Yiming. Ess: Efficient semantic search on gnutella-like p2p system. Technical report, Department of ECECS, University of Cincinnati, 2004.
- Zhuge, Hai; Liu, Jie; Feng, Liang; Sun, Xiaoping, and He, Chao. Query routing in a peer-to-peer semantic link network. *Computational Intelligence*, 21:197–216, 2005.