Web-Based Information Technologies and Distributed Systems

Edited by Alban Gabillon
Quan Z. Sheng - Wathiq Mansoor

Atlantis Ambient
and Pervasive Intelligence
Series Editor Ismail Khalil

ATLANTIS AMBIENT AND PERVASIVE INTELLIGENCE
VOLUME 2

SERIES EDITOR: **ISMAIL KHALIL**

Atlantis Ambient and Pervasive Intelligence

Series Editor:

Ismail Khalil, Linz, Austria

(ISSN: 1875-7669)

Aims and scope of the series

The book series 'Atlantis Ambient and Pervasive Intelligence' publishes high quality titles in the fields of Pervasive Computing, Mixed Reality, Wearable Computing, Location-Aware Computing, Ambient Interfaces, Tangible Interfaces, Smart Environments, Intelligent Interfaces, Software Agents and other related fields. We welcome submission of book proposals from researchers worldwide who aim at sharing their results in this important research area.

All books in this series are co-published with World Scientific.

For more information on this series and our other book series, please visit our website at:

www.atlantis-press.com/publications/books

AMSTERDAM – PARIS

Web-Based Information Technologies and Distributed Systems

Alban Gabillon

University of Polynésie Française
BP 6570
98702 FAA'A
Tahiti
Polynésie française

Quan Z. Sheng

School of Computer Science
University of Adelaide
Adelaide, SA 5005
Australia

Wathiq Mansoor

American University in Dubai, UAE

AMSTERDAM – PARIS

Atlantis Press

29, avenue Laumière
75019 Paris, France

For information on all Atlantis Press publications, visit our website at:

www.atlantis-press.com

Atlantis Ambient and Pervasive Intelligence

Volume 1: Agent-Based Ubiquitous Computing - Eleni Mangina, Javier Carbo, José M. Molina

ISBN: 978-90-78677-28-4 e-ISBN: 978-94-91216-32-9
ISSN: 1875-7669

Preface

The Fourth International Conference on Signal-Image Technology & Internet-Based Systems (SITIS 2008) has been successfully held during the period 30th November to 3rd of December of the year 2008 in Bali, Indonesia. The Track Web-Based Information Technologies & Distributed Systems (WITDS) is one of the four tracks of the conference. The track is devoted to emerging and novel concepts, architectures and methodologies for creating an interconnected world in which information can be exchanged easily, tasks can be processed collaboratively, and communities of users with similar interests can be formed while addressing security threats that are present more than ever before. The track has attracted a large number of submissions; only fifteen papers have been accepted with acceptance rate 27 %. After the successful presentations of the papers during the conference, the track chairs have agreed with Atlantis publisher to publish the extended versions of the papers in a book. Each paper has been extended with a minimum of 30 % new materials from its original conference manuscript.

This book contains these extended versions as chapters after a second round of reviews and improvement.

The book is an excellent resource of information to researchers and it is based on four themes; the first theme is on advances in *ad-hoc* and routing protocols, the second theme focuses on the latest techniques and methods on intelligent systems, the third theme is a latest trend in Security and Policies, and the last theme is applications of algorithms design methodologies on web based systems.

We would like to give our great appreciations to the authors and the PC members of the track to their excellent contributions and effort that makes the creation of this book is achievable. Also, we would like to thank Atlantis publisher who has agreed to publish this

valuable book to the community. Special thanks to Zeger Karssen and Zakaria Maamar for their help and support during the publication of the book.

Alban Gabillon (University of Polynésie Française, France)
Quan Z. Sheng (University of Adelaide, Australia)
Wathiq Mansoor (American University in Dubai, UAE)

Contents

Chapter 1

A Community-based Approach for Service-based Application Composition in an Ecosystem

Elie Abi-Lahoud, Marinette Savonnet, Marie-Noelle Terrasse, Marco Viviani,
Kokou Yétongnon
Université de Bourgogne – Sciences et Techniques, Laboratoire LE2I – Mirande,
Aile de l'Ingénieur, 9, av. Savary, 21078 Dijon cedex, France

The design of composite applications by combining existing services with known semantics is an ongoing topic in current research. Several studies are aimed at providing service description models and standards, service discovery and matching etc. However, service composition in distributed dynamic environments such as P2P ecosystems has received little attention from research communities. In this paper we present a design framework for composing services, taking in particular into account different ways of building peer-communities based on network or services characteristics.

1.1 Introduction

Service oriented computing provides software designer with new concepts and emerging principles for developing loosely-coupled, cross-enterprise business applications. Traditionally, software development approaches rely on CASE tools [1] and modeling concepts to describe and implement software components that can be integrated into applications. Recently, we are witnessing a shift from this static view of software development and deployment towards a dynamic, adaptable service based view of software design in which applications could be realized in a flexible manner to respond to changing needs of users. In this emerging design view, services provide high level functional components that can be shared in open distributed environments. The goal is to design composite applications by combining existing service components with known semantics, spanning organizations and computing platforms.

Many research efforts have been aimed at service oriented computing, ranging from tech-

A. Gabillon et al., *Web-Based Information Technologies and Distributed Systems*,
Atlantis Ambient and Pervasive Intelligence 2, DOI 10.2991/978-94-91216-32-9_1,

nical services to telecommunication services, business process modeling and popular web services. In the information system realm, this research effort has focused to a significant extent on (i) services definition: linguistic constructs and models to define and represent services' behaviors and properties, (ii) services discovery: architectures or protocol suites to allow service sharing and functional matching and (iii) services composition: orchestration of service components into more complex processes [2, 3, 4, 5].

Open computing environments created the needs for virtual cooperating systems to allow resource sharing. Digital enterprise ecosystems emerged as a concept for capturing the interactions of business networks. Ecosystems can comprise autonomous organizations and related services, sharing agreements on overall domain specific components and rules governing interactions and inter-relationships among the participants. Enterprise ecosystems provide some formalization of common models, shared knowledge and global resources to enable loosely coupled interoperability among enterprises. In essence, they can be used, as opposed to open environments, to provide controlled business and enterprise environments delimiting the collaboration scopes to a set of actors respecting business related rules. Ecosystems form a suitable environment for application composition. They provide an environment with identified semantics and business properties wherein peers providing services interact based on a global but not too restrictive agreement. This helps in distinguishing functional needs, relations between them and other business-relevant properties.

1.1.1 *Objectives and Contributions*

In this chapter we address the service-based application composition issue in a peer-to-peer ecosystem. In such an ecosystem, our approach consists of first defining a high level interaction between actors, then refining it to an application defined as a graph of abstract services. The application is realized by substituting abstract services for matching services provided by peers belonging to the ecosystem. We show how the concrete application realization based on service composition can take advantage of the ecosystem's network reorganization into peer communities, in terms of communities' definition and communication protocol by building on top of an unstructured system a hybrid overlay network.

The remainder of the chapter is organized as follows. Section 1.2 exposes literature background, namely service orientation and P2P systems and communities. Section 1.3 exposes ecosystems, peer-communities and services under a multi-layered comprehensive framework for service-based application composition. Section 1.4 focuses on the fourth layer of the framework, describing its organization in a super-peer based overlay network. Sec-

tion 1.5 presents the European Electricity Exchange Market as an ecosystem example. It compares two views of the studied ecosystem, focusing on the process of application realization. Section 1.6 concludes the chapter and presents future work.

1.2 Background

In this Section we discuss two recent developments that are changing the way IT applications are designed, deployed and exchanged: (i) *service oriented computing*, providing a new paradigm for creating applications on demand and (ii) *peer-to-peer systems*, often used for sharing resources. We first describe current work in service oriented computing, then we briefly define P2P systems and review P2P communities-related literature.

1.2.1 *Service Orientation*

Previous work in service oriented systems has focused to a significant extent on 1) constructs and models to define and represent the behaviors and properties of services, and 2) the architectures or protocol suites to allow service sharing and matching and on services' composition into more complex systems.
A *service* can be viewed as a self-contained, modular basic software unit that is described, published and invoked over a network to create new software components or products. It encapsulates functions and modules of an application domain (e.g., business process components, supply chain units). It provides an interface to allow external invocation. Among *service description models* proposed in the literature, the Web Service Description Language (WSDL) [2] has become a de-facto industry standard. It is an XML-based model that allows a syntactical representation of the methods and parameters needed to interact with a service. Other models extend the syntactic representation of services by adding semantics to resolve definition discrepancies and heterogeneities that can hinder service matching and composition. For example, the METEOR-S project [3] extend WSDL with semantic annotations while the OWL-S [4] and WSMO [5] (the Web Service Modeling Ontology) approaches are based on an ontology of web services. The ontology provides a precise description of service components and their inter-relationships. Several *standards* and *architectures* are proposed to enable the integration and sharing of heterogeneous service. For example, *Service Oriented Architecture* (SOA) is a "paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains" [6, 7].

Service discovery is defined by Keller *et al.* [8] as the automatic localization of services corresponding to user's need. Booth *et al.* [9] describe the discovery process as the localization of a machine readable description corresponding to given functional needs. Toma *et al.* [10] define service discovery as a process taking as input a user query and returning as output a list of available resources corresponding to the user's need expressed in the input query. Two major aspects are tackled by service discovery, namely service localization and service matching. Service localization relies on either centralized or distributed architectural models. The UDDI (Universal Description Discovery and Integration [11, 12, 13]) became a widely know standard for centralized service localization. It consists of a set of *UDDI nodes* collaborating to create a global structure. Srinivasan *et al.* [14] extended the UDDI model to support OWL-S semantic description allowing more efficient comparison between the user's need and the available services. Distributed localization models consisted first in setting up a distributed federation of UDDIs [15, 16]. Verma *et al.* [17], Paolucci *et al.* [18] and Schmidt *et al.* [19] discussed other complex models. Service matching is widely addressed in the literature. Ernst *et al.* [20] and Dong *et al.* [21] studied syntactic similarity based on trace data and clustering respectively. Paolucci *et al.* [22], Benatallah *et al.* [23] and the WSMO workgroup [8] tackled the matching based on semantic similarity. Taher *et al.* [24] and Bordeaux *et al.* [25] studied other approaches based on abstract services and labeled transition systems respectively.

Service composition designates the interaction taking place between two or more services in order to accomplish a given goal. The composition process tackles several aspects, such as the interaction description and organization, the message exchange management, the transaction like behavior, the interaction context, the level of automation, the failure recovery, *etc.* The Web Services Business Process Execution Language (WS-BPEL [26]) is the current standard for describing services' compositions. It allows to model compositions as interaction workflows. An alternative for BPEL is the Web Service Choreography Interface (WSCI [27]). Both BPEL and WSCI allow static service composition wherein services are bound at design time. Thakkar *et al.* [28], Casati *et al.* [29] and Sun *et al.* [30] present dynamic composition environments based on composition engines capable of binding selected services at runtime. The automation level of the composition process is also widely studied in the literature. An exhaustive survey on service composition is out of the scope of this work. Useful information is available in [31, 32].

1.2.2 *P2P Systems*

Peer-to-peer (P2P) systems are distributed systems composed of distinct computing elements, called *peers*, with similar resources and capabilities. Peers interact together to share services and resources. P2P systems can be classified into *unstructured* and *structured* systems. In unstructured P2P systems, peers are organized in random graphs with no control over their contents. Each peer controls its contents and the access and sharing of its resources. Unstructured P2P systems can be further classified into (i) *centralized* systems when a central directory is used to store global state information (indexes, data locations, etc.), (ii) *decentralized* systems when no global state information (network state information, context data) is maintained by the peers, and (iii) *hybrid* systems which combine the characteristics of centralized and decentralized by using *super-nodes* (or *super-peers*) [33] to control simple peers with less resources and capabilities. Structured P2P systems keep a tight control over network topology and peer contents by placing data not randomly in peers but at specific locations defined by the overlay network strategy (an indexing strategy).
P2P systems can also be structured by using clustering techniques to group peers based on common properties or interests. Clusters can be viewed as communities belonging to overlays defined on top of unstructured P2P systems. According to Khambatti *et al.* [34], a *community* is a set of active peer members, involved in sharing, communicating and promoting common interests. Significant research is currently targeted at creating community-oriented *overlay networks* in order to avoid query messages flooding and to save resources in handling irrelevant queries over the P2P network. DHT-based techniques [35, 36] guarantee location of content within a bounded number of hops by tightly controlling the data placement. Other techniques based on *clustering* strategies have been proposed to reduce query traffic, grouping peers sharing similar *properties*.
According to Oztopra *et al.* [37], two main strategies are used the literature for clustering peers. The first strategy takes into account network related characteristics while the second focuses on peers *interests*. In the following we review both strategies considering that peers participating in a services' ecosystem are mainly interested in providing, sharing and reusing services.

1.2.2.1 *Using Network Characteristics to Build Peer Communities*

Several research was conducted on clustering of peers based on *network characteristics*. Ratnasamy *et al.* [38] present a scheme whereby nodes partition themselves into groups called bins such that nodes that fall within a given bin are relatively close to one another in

terms of *network latency*. Zhang *et al.* [39] propose a topology aware system constructing an overlay network by exploiting the *locality* in the underlying network using the group concept. Each host in the overlay is running a protocol to communicate with other hosts. In general, each host maintains information about a set of other hosts to communicate with. Two hosts are considered as neighbors if they are connected through the overlay. MetaStream [40] is a content discovery protocol for topology-aware on-demand streaming. In MetaStream, clients choose streaming sources based on *network distance*. For this purpose, they self-organize into a dynamic hierarchy of clusters based on the network topology. Any protocol for constructing a topology-aware hierarchy can be used. Connectivity-based Distributed node Clustering (CDC) [41] implements node clustering based on *node connectivity* in P2P networks, while Zheng *et al.* [42] use an approach based on *link delay* of node communications in the P2P network. Oztopra *et al.* [37] propose to cluster peers based on *time* (communication duration) closeness.

Disregarding the specific adopted technique, building communities based on network characteristics generates an overlay where peers in a community provide different services. In such a scenario, it is highly probable that peers in a community will behave in a *cooperative* manner. When one peer is selected, the possibility of selecting another member of the community is increased. This makes sense considering that it is better for a peer to search for a service among his neighbors before searching among further members.

1.2.2.2 *Using Service Properties to Build Peer Communities*

Service properties are classified in two main categories: (i) *functional* and (ii) *non-functional* [5, 43].

Functional properties represent the functionality provided by the service and its semantic description elements, for example the related input/output parameter list (and conditions if available). Note that the service as a software unit might provide several functionalities. In this case, each functional aspect can be studied as a separate entity. Building communities based on the functionalities provided, allows us to obtain *competitive* communities, where each peer holds services accomplishing the same task, although some service attributes may vary. This way, each peer will compete with others to get selected by a client. The client choice is based on non-functional properties, which are not directly related to the functionality provided by the service.

An exhaustive list or classification of those properties is out of the scope of our project, we note that some of the **non-functional properties** are QoS related and thus in correlation

with the network characteristics such as execution time. Other non-functional properties do not express QoS but might form substrate criteria to build communities on, for instance security-level and trust [44, 45].

1.3 A Framework for Sharing Services

In the following we present and model peers interactions in a services' ecosystem. We define *peer-communities* in such ecosystems and formalize interaction rules. We also describe the multi-layered service-based composition framework under which on-demand application composition takes place.

1.3.1 *Ecosystem, Peer-communities and Services*

In an ecosystem various communities organize business-driven collaboration among groups of service-providing *peers*. We first describe such ecosystems, and then we provide more precise definitions of the relevant terms.

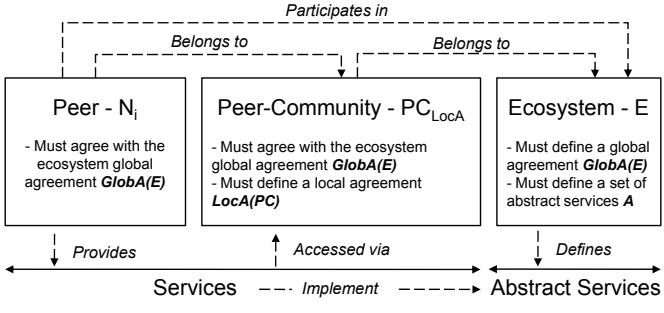

Figure 1.1 Ecosystem's Organization

As described in Figure 1.1, an *ecosystem* is a group of *peer-communities* in which each peer-community accepts a consensual specification of a business area and its related business rules, referred to as *global agreement*. The ecosystem defines a set of abstract services based on the global agreement. A peer respecting the global agreement provides services as implementations of the abstract ones. Peer-communities are groups of peers having a consensual agreement on a minimum set of properties. We denote by *local agreement* this set of the required properties. Each peer must satisfy its ecosystem's global agreement and its communities' local agreements.

We model the ecosystem's organization by the following definitions and rules:

- Given an ecosystem E, its global agreement specification is the set of business-related properties denoted by $GlobA(E)$. $GlobA(E) = \{p^p\}_{p=[1,...,q]}$ where q is the number of relevant business properties in the ecosystem. For example *response time*, which is a widely used property. Properties notation language is chosen depending on the ecosystem domain(s).
- *Abstract services* are defined by the ecosystem in order to disseminate an application domain knowledge. An *abstract service* is an interface defining the abstract operation needed to fulfill the related functionality. An abstract service describes the operation via a semantic business-related description, including but not limited to, its inputs, outputs and an associated set of constraints, typically restrictions. It provides no real implementation of the operation, just the signature. The abstract service interface allows to define also realization constraints to be respected by the interface implementer. An abstract service is designated by A_i. We denote by A the set of abstract services defined by the ecosystem E, and by n the number of described abstract services in the ecosystem such as $|A| = n$.
- For each abstract service A_i the set of defined properties P_i is defined as: $P_i = \{p_i^m\}_{m=[1,...,q_i], i\in\{1,...,n\}}$ subject to $q_i \leqslant q$ and $\exists\ f:\ p_i^m \rightarrow p^p$. For instance, based on the business property *response time*, the abstract service's interface defines the property *execution time*. We note that some business related properties might not be relevant for a given abstract service, thus they are not used in its interface.
- Abstract services defined by an ecosystem E must comply with the $GlobA(E)$ specification:

$$\forall A_i \in A,\ \ P_i \subseteq GlobA(E) \text{ where } i \in \{1,\ldots,n\}$$

- *Services* are defined by peers in order to present their business offers in the ecosystem. They are defined with respect to the ecosystem required functionalities, thus a service is a concrete implementation of an abstract one. A service S_{ij} implementing an abstract service A_i must redefine the abstract operation and respect all the associated constraints. Although having similar functional interfaces, two services S_{ij} and $S_{ij'}$ may differ in their non-functional properties.
- For each service S_{ij} the set of service-related properties is derived from the corresponding abstract service properties and is denoted by $P_{ij} = \{p_{ij}^p\}_{i\in\{1,...,n\},\ p=[1,...,q_i],\ j\in\mathbb{N}}$. Actually a service redefines and implements the properties of its related abstract service. For example, the service redefines the property *execution time* inherited from its

corresponding abstract service and evaluates it. If a service's WSDL description provides several operations related to different functionalities, the service is mapped to the required number of abstract services.

- Within partnerships, services are offered as implementations of abstract services. An implementation relation *IMPD* is defined in order to associate a concrete service with its corresponding abstract service. The implementation dependency is such that:

$$\forall S_{ij},\ \forall A_i \in A,\ i \in \{1,\ldots,n\},\ j \in \mathbb{N}$$
$$IMPD(S_{ij},A_i) \implies \forall p_i^p \in P_i,\ \exists p_{ij}^{p'} \in P_{ij} \text{ where } p, p' \in \{1,\ldots,q_i\}$$

- Given a peer-community PC, its local agreement specification is denoted by $LocA(PC)$. A local agreement is specified either in terms of services properties or any other criteria relevant to the studied ecosystem members (e.g. locality, peer-trust-level, etc.).
- Peers belonging to a community PC_{LocA} must comply with the local agreement specification $LocA(PC)$.

$$\forall N_i \in PC_{LocA},\ N_i\ respects\ LocA(PC)$$

  For instance, given a community PC_{LocA} based on the local agreement $LocA$: *equal trust-level*, all its member peers share the same value for the property trust-level.
- We denote by $S(N_i)$ the set of services provided by the peer N_i and by $S(PC)$ the services available in the community PC. $S(PC)$ is the union of the services whose providing peers comply with the local agreement $LocA(PC)$.

1.3.2 *Multi-layered Service-based Composition Framework*

The multi-layered framework for service-based application composition is illustrated in figure 1.2. It allows dynamic application composition in a given ecosystem. It is composed of five layers. The first layer models the studied business logic in a workflow of activities from which abstract services are described. The second layer allows to define an application modeled by a graph of abstract services. The third layer contains the set of realizations of the application. A realization is defined as a combination of available services on the network capable of executing the application process. The fourth layer is the virtual overlay network in which peers are clustered in communities [1]. The fifth layer represents the underlying peer architecture. At this layer we capture peer related non-functional characteristics that help assessing network related measures. The service binding is deferred until runtime,

[1] For simplification purposes, we do not distinguish hereafter between the terms *peer-community* and *community*.

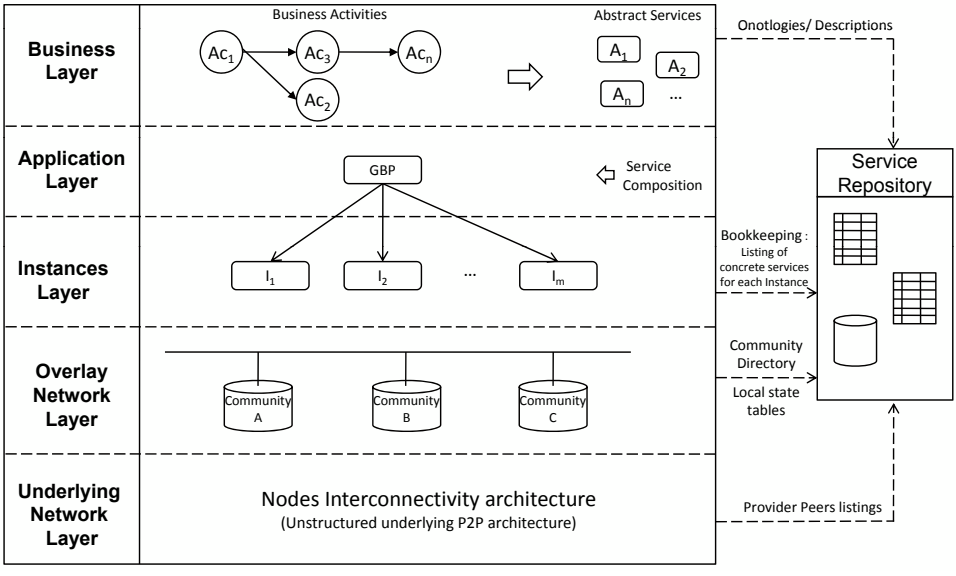

Figure 1.2 Multi-layered Composition Framework

allowing a dynamic cost-based service selection. In the following we briefly describe the framework layers and components.

- The ***Business Layer*** models the business logic in a workflow of required activities. The main purpose is to refine business activities in abstract services modeling functionalities shared by applications in the domain of interest.
- The ***Application Layer*** represents the composite application by a graph of abstract services denoted by *Generic Business Process* (GBP). A GBP is an oriented attributed graph whose vertices represent abstract services and edges represent control sequences indicating functional dependencies between the abstract services. Attributes are associated with the vertices and the edges in order to represent functional and non-functional data and characteristics. Yetongnon *et al.* [46] discusses details about this layer and the following ones.
- The ***Instances Layer*** contains a set of possible service compositions generated from the GBP abstract service graph and based on the available services in the ecosystem. This conversion of a GBP into a set of GBP instances is carried out by an *instantiation process* in which services registered by peers are substituted for the abstract services of the GBP. Thus, a GBP instance is a directed attributed graph whose nodes are registered services, edges connect two services based on the functional dependencies

expressed in the GBP, and the attributes values are derived from the corresponding attributes of both nodes and edges in the GBP graph. The study of the instantiation process is out of the scope of this chapter.

- The ***Overlay Network Layer*** is the peers' organization into a community-based overlay network. The overlay description, organization and communication is detailed the following section.
- The ***Underlying Network Layer*** helps capturing the underlying network characteristics. At this point, the services properties can be evaluated along with the properties of the edges connecting the hosting peers in an instance graph. Peer properties are projected on the corresponding instances graphs. Each enterprise is modeled by a set of peers such as each enterprise application server, providing services or requiring an application instantiation, is a peer.
- The ***Service Repository*** component interacts with the five layers. It provides at each layer the required elements (cf. figure 1.2). For instance at the business layer, it contains the ontologies and the abstract services listings.

1.4 The Overlay Network

The overlay network is a view of the ecosystem filtered by peer-communities local agreements. For instance, local agreements consisting of the property *providing similar functionalities* generate an overlay of peer-communities in which peers providing services implementing the same abstract service are regrouped in the same community.

1.4.1 *Overlay Organization*

We adopt the classical two levels peer organization, consisting of peer groups each managed by a super-peer. Figure 1.3 illustrates an example of a super-peer based overlay network architecture for service oriented application development. It consists of peers whose main goal is to provide concrete implementations of abstract services. Peers are organized in communities managed by super-peers which are in turn organized in a communication topology. For example, peer-community 1 is managed by super-peer `SN1` and includes four peers $\texttt{N1}, \ldots, \texttt{N4}$. Note that a peer can provide implementations for one, several abstract services or none; on the other hand an abstract service can be implemented by more than one peer.

A *peer-community* is a set of peers respecting a local agreement and managed by a super-

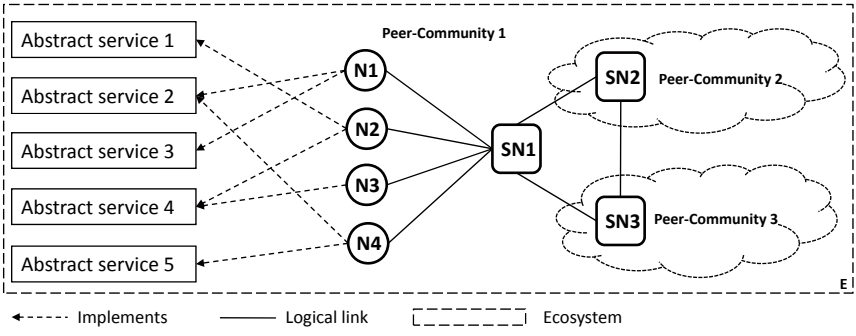

Figure 1.3 Example of the Overlay Network Organization

peer. Formally, it is denoted by

$$PC_{LocA} = \left(SN_k,\ \{N_r\}_{r=1,\ldots,n_k}\right)$$

where SN_k is the super-peer managing the set of n_k peers respecting the local agreement $LocA(PC)$. Peer-communities are based on either (i) network characteristics or (ii) properties connected to services as described in section 1.2.2.

1.4.2 *Super-peers*

Super-peers are selected based on their computing capabilities (in order to handle the GBP instantiation) and/or their trustworthiness[2]. The links between super-peers are chosen as the shortest path from the physical network. Each link between super-peers represents a bidirectional communication path.

Super-peers in the overlay network maintain and manage a *distributed directory structure*. Each super-peer maintains a *local repository*, consisting of two tables: a *local state information table* (for example tables 1.3(a), 1.3(b), 1.4(a)) and a *global state information table* (for example tables 1.2, 1.4(b)). The local state information table contains: (i) the set of peers managed by the super-peer. (ii) A state $St(N_r)$ for each peer. $St(N_r) = \text{ON}$ if the peer N_r is on-line. $St(N_r) = \text{OFF}$ if the peer N_r is off-line. (iii) For each peer N_r, a list of provided services S_{ik} and their related abstract services A_i.

The global state information table, *community directory*, represents for each super-peer SN_k in the overlay, the set of abstract services $\{A_i\}_{i=1,\ldots,n_i}$ that are supported by its community.

[2]For the sake of conciseness we will not detail the super-peers choice and we consider for the rest of the chapter that a super-peer does not depart.

1.4.3 *Event Related Communication*

Two major events need to be considered, first peer join and second peer departure.
When a peer joins the ecosystem three actors or group of actors are implicated. First the peer itself, (i) launches a probing process to discover the closest super-peer in terms of physical distance then (ii) it queries the selected super-peer asking for the list of abstract services and for the community directory table. Then the peer (iii) decides on abstract services to implement [3] or if already implemented it grants network members access to its services. If needed, the peer creates mappings between its existing services and one or more related abstract services. The peer respects indirectly the ecosystem's global agreement *GlobA* by choosing to implement an abstract service or by providing required mappings for its existing services. Finally the peer (iv) sends requests to the super-peers of the communities it is willing to join, notifying them of its presence in the community. Clearly respecting the local agreement $LocA(PC)$ of each of the solicited communities is a join prerequisite. Second, each of the concerned super-peers (i) receives the joining peer request and information, (ii) updates its local copy of the community directory and (iii) sends update notifications to direct super-peers neighbors. Third, other super-peers (i) receive the community directory update notifications and (ii) proceed on updating their global state information.
When a peer departs, the same actors are implicated. First, the peer itself notifies its super-peers before going offline. We adopt clean peer departure considering that peers main goal is to collaborate, improving the network and its added value (generated applications). Second, each of its related super-peers (i) flags the peer as offline in the local state table. Afterward, if the departing peer is the last to provide a given functionality, (ii) the related abstract service is removed for the community directory. Finally the super-peer (iii) sends update notifications to neighbors super-peers containing the community directory new state. Third, other super-peers (i) receive the community directory update notifications and (ii) proceed on updating their global state information.

1.5 Case Study: The European Electricity Market

Produced electricity cannot be stored for long, therefore the market must undergo a regulation process. Market regulation consists of insuring that the quantity of produced electricity is equal to the needed consumption power. Electricity regulation is ensured via exchanges

[3] respecting the corresponding implementation relation *IMPD*.