Registry Composition in Ambient Networks

Fatna Belqasmi

A Thesis
in
The Department
of
Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy at
Concordia University
Montreal, Quebec, Canada

June 2008

Your file Votre référence
ISBN: 978-0-494-42549-7
Our file Notre référence
ISBN: 978-0-494-42549-7

ABSTRACT

Registry Composition in Ambient Networks

Fatna Belqasmi, Ph.D.
Concordia University, 2008

Ambient Networks (AN) is a new networking concept for beyond 3G. It is a product of the European Union's Sixth Framework Program (FP6). Network composition is a core concept of ANs. It allows dynamic, scalable and uniform cooperation between heterogeneous networks. ANs can host various registries. These registries may be of different types (e.g. centralized, distributed), store heterogeneous types of information (e.g. raw data vs. aggregated data), and rely on different interfaces to access the stored information (i.e. protocols or programming interfaces). When ANs compose, the hosted registries need to compose. Registry composition is a sub-process of network composition. It provides seamless and autonomous access to the content of all of the registries in the composed network.

This thesis proposes a new architecture for registry composition in ANs. This overall architecture is made up of four components: interface interworking, data interworking, negotiation and signaling. Interface interworking enables *dynamic* intercommunication between registries with heterogeneous interfaces. Data interworking involves *dynamically* overcoming data heterogeneity (e.g. format and granularity). Interface and data interworking go beyond static interworking using gateways, as done today. The negotiation component allows the negotiation of the composition agreement. Signaling coordinates and regulates the negotiation and the execution of the composition agreement.

Requirements are derived and related work is reviewed. We propose a new functional entity and a new procedure to orchestrate the composition process. We also propose a new architecture for interface interworking, based on a peer to peer overlay network. We have built a proof-of-concept prototype. The interface-interworking component is used as the basis of our new architecture to data interworking. This architecture reuses mechanisms and algorithms from the federated data base area. The thesis proposes as well a new architecture for on-line negotiation. The architecture includes a template for composition agreement proposals, and a negotiation protocol that was validated using SPIN. A new signaling framework is also proposed. It is based on the IETF Next Step in Signaling (NSIS) framework and was validated using OPNET. Most of these contributions are now part of the AN concept, as defined by the European Union's Sixth Framework Program.

ACKNOWLEDGEMENTS

I would like to thank my supervisors Dr. Rachida Dssouli and Dr. Roch Glitho for their support, guidance and motivation. I thank Dr. Dssouli for her warm encouragements, good advices and kindness. Her continuous support was of great help. I thank Dr. Glitho for his assistance and for always being their when I need him. I learned a lot from working with him. Without his continuous encouragement, patience and valuable advices this thesis would not have been possible. His vision and enthusiasm had a great impact on my scientific development and the research work presented in this thesis.

I would like to thank my supervisory committee members, Dr. Agarwal, Dr. Debbabi and Dr. Rilling, for their advices and comments at different stages of this research. I would also like to thank Dr. F. Khendek, his comments and ideas were always helpful.

Special thanks to Concordia University, Ericsson and the Natural Sciences and Engineering Research Council of Canada (NSERC) for their financial support. I thank Ericsson for giving me the opportunity to work in its Research and Development department.

Many thanks as well to all my colleagues from the ambient network project, for their constructive comments. Our participation (as a non EU member) in the ambient network project was sponsored by Ericsson Canada, NSERC of Canada and "Ministere du Developpement Economique, de l' Innovation et de l'Exportation" (MDEIE) of Quebec.

I am also grateful to all my colleagues at the TSE lab at Ericsson Canada, for their help and suggestions. It has been a good pleasure to work with all of you.

DEDICATION

This dissertation is dedicated to my parents -- Fatna and Ali --, and my brothers and sisters -- Fatiha, Fatima, Halima, Mohammed, Amina and Ahmed -- for their unconditional support, encouragement, love and continuous confidence in me. They are a source of energy and motivation in everything I do.

Table of Content

List of Figures

List of Tables

Acronyms and Abbreviations

1G:	First Generation Wireless System
2G:	Second Generation Wireless System
3G:	Third Generation Wireless System
4G:	Forth Generation Wireless System
AN:	Ambient Network
ACS:	Ambient Control Space
ASI:	Ambient Service Interface
AS:	Application Server
ANI:	Ambient Network Interface
ARI:	Ambient Resource Interface
API:	Application Programming Interface
CA :	Composition Agreement
C-FE:	Composition Functional Entity
CM-FE:	Context Management Functional Entity
CA:	Composition Agreement
CAN:	Content-Addressable Network
CASP:	Cross Application Signaling Protocol
CIB:	Context Information Base
CSCF:	Call Session Control Function
C-mode:	Connection mode
D-mode:	Datagram mode
DB:	DataBase
DBMS:	DataBase Management System
DEEP:	Destination Endpoint Exploration Protocol
DHT:	Distributed Hash Table
DNS:	Domain Name System
EGIST:	Extended General Internet Signaling Transport
FDBS:	Federated DataBase Systems
FE:	Functional Entity

GANS:	Generic Ambient Network Signaling
GIST:	General Internet Signaling Transport
GQS:	Global QoS Server
GM:	Group Management
GPRS:	General Packet Radio Service
GSLP:	GANS Signaling Layer Protocol
GTLP:	GANS Transport Layer Protocol
HSS:	Home Subscriber Server database
HTTP:	Hypertext Transfer Protocol
IETF:	Internet Engineering Task Force
I-CSCF:	Interrogating Call Session Control Function
I-Net:	Infrastructure Network
IMS:	IP Multimedia Subsystem
IP:	Internet Protocol
IPDI:	Information Publication and Discovery Interface
IDOQL:	Java Data Object Query Language
ITU-T:	International Telecommunication Union- Telecommunication Standardization Sector
MAA:	Multimedia-Auth-Answer
MAR:	Multimedia-Auth-Request
MANET:	Mobile Ad-hoc NETwork
MIB:	Management Information Base
ML:	Messaging Layer
MRA-FE:	Multi-Radio Access Funtional Entity
MRM:	Message Routing Method
NAD-FE:	Network and Advertisement FE
N-EGIST:	Negotiation Extended General Internet Signaling Transport
NSIS:	Next Step in Signaling
NTLP:	NSIS Transport Layer Protocol
NSLP:	NSIS Signaling Layer Protocol
OSL:	Overlay Support Layer
PAN:	Personal Area Network
P-CSCF:	Proxy Call Session Control Function

P2P:	Peer-to-Peer
PDP:	Pervasive Discovery Protocol
QoS:	Quality of Service
QoS-FE:	QoS Functional Entity
RCE:	Registry Composition Entity
RR:	Resource Registry
RON:	Registry Overlay Network
RSVP:	Resource ReSerVation Protocol
SARC:	Signaling Application for Registry Composition
SIP:	Session Initiation Protocol
S-CSCF:	Serving Call Session Control Function
SATO:	Service-aware Adaptive Transport Overlays
SC:	SATOClients
SS:	SATOServers
SP:	SATOPorts
SNMP:	Simple Network Management Protocol
SOAP:	Simple Object Access Protocol
SQL:	Structured Query Language
SMTP:	Simple Mail Transfer Protocol
SPIN:	Simple PROMELA INterpreter
TCP:	Transmission Control Protocol
TL:	Transport Layer
TLS:	Transport Security Layer
UAA:	User Authentication Answer
UAR:	User Authentication Request
UDDI:	Universal, Description, Discovery and Integration
UDP:	User Datagram Protocol
URI:	Universal Resource Identifier
UMTS:	Universal Mobile Telecommunications System
UpNp:	Universal plug and play
VoIP:	Voice over IP
VR:	Virtual Registry
WLAN:	Wireless Local Area Network

X-adhoc:	eXtended ad-hoc
XR:	eXtended ad-hoc Registry
XML:	Extensible Markup Language

CHAPTER I: Introduction

This chapter starts by motivating the problem. After that, it states the problem and presents the thesis objectives, and summarizes the contributions along with the related publications. It ends with the thesis organization.

I.1 Motivations

Cooperation between networks is no novelty. It emerged with the first generation (1G) of wireless systems, where cellular networks belonging to different operators cooperate to give roaming end-users seamless access to basic services. 2G and 3G wireless systems have further strengthened the concept. However, the cooperation in 1G/2G/3G wireless systems has several drawbacks. It relies on off-line agreements and manual configuration operations. It also enables access to a very limited set of services (i.e. those identified in the agreement). The emergence of new networks (e.g. Mobile ad-hoc networks, wireless sensor networks, personal area networks) and new end users' needs brings new challenges and new requirements. End users are not only interested in accessing new services. They would like to do this ubiquitously, transparently, over any access technology, and over any type of network.

Ambient Networks (AN) is an emerging networking concept for beyond 3G fixed and wireless networks, designed to meet these challenges [1]. It was developed in the context of AN project, a multi-national collaborative project of the European Union's IST Sixth Framework Program (FP6). The project brought together a strong consortium of leading operators, equipment suppliers and research organizations including universities from the European Union and also from different parts of the world. It covers the networking part of the Wireless World Initiative (WWI), whose objective is to define future communication systems that provide users with the best user experience while minimizing the cost of purchase, use and ownership of the systems. Reference [2] gives more details on WWI.

Network composition is one of the key features of Ambient Networks [3], envisioned to allow a level of network cooperation which goes far beyond the static cooperation of today. It provides a uniform, dynamic and scalable cooperation between heterogeneous networks at the control layer, where a network can range from a single node to a full-fledged operator network [4]. The cooperation process via network composition is transparent to the end users, but takes user context and network context into account. It allows seamless and instantaneous access to new services – a significant advance over traditional networks that require extensive manual configuration. In its strongest form, known as network integration, two networks can merge and form a single network.

Dynamic network cooperation at control layer is principally enabled through a new common control plane --called Ambient Control Space (ACS)-- that can be deployed over various existing and future types of networks. The ACS is the set of the functions offered by the control layer and is organized in functional entities (e.g. QoS FE, Mobility FE). Applications can access and use the ACS functionalities through a well defined interface, called Ambient Service Interface (ASI).

Ambient Networks can host several registries. A registry is any authoritative store of information or repository of data. Examples are Management Information Bases (MIB), relational databases and Context Information Bases (CIB) [5]. When Ambient Networks compose, the hosted registries need to compose. Registry composition is a sub-process of network composition. It provides seamless, autonomous and uniform access to the updated content of all of the registries in the composed network. There are several motivations for registry composition. A first motivation is that the registries' content may need to compose. Indeed, when Ambient Networks compose, the content of the hosted registries may be kept as it is; modified or even merged. Content merging can happen for instance when a new service is proposed by the composed network, by combining elementary services provided by the composing networks. A second motivation is that entities in the composed network may need access to a content hosted by a registry that was in a different network before composition. The interface of such registry (e.g. SNMP, SQL) may be different from the one used by the interested entity. The granularity and the format of the registry content may also be different from those supported by the interested entity. A third motivation is that new registries may need to be created in order to store the composed content.

Registry composition supports both the ACS functional entities and the applications running above the ACS. Indeed, each ACS functional entity may use a private registry to store information that is related to its functioning and needs to access this information after composition. The Context Management-FE (CM-FE) for instance collects, filters, aggregates, and provision context information (i.e. user-related and network-related information), and this information may be stored in a private registry. After ANs compose, each FE may also need to access information stored by peer-FEs in other networks.

Furthermore, applications running above the ACS may need to access registries that are part of the real network over which the ACS is deployed. After network composition, the applications may need to access some or all of the registries hosted by the composed network, including those that were part of a different network before composition and those that may be added due to the composition.

I.2 Problem Statement and thesis objectives

The registries to compose may be of different types (e.g. centralized, distributed), they may store heterogeneous types of information (e.g. raw data vs. aggregated data) that is presented using different formats (e.g. Object oriented database, relational database), and they may rely on different interfaces to access the stored information (i.e. protocols such as P2P information discovery protocols [6] or programming interfaces such as UDDI APIs [7]). Two types of problems are therefore related to registry composition: Interface interworking and data interworking. Interface interworking involves autonomously enabling the intercommunication between registries with heterogeneous interfaces. Data interworking involves autonomously overcoming data heterogeneity (e.g. format, type and granularity). Data interworking and interface interworking in the context of registry composition goes beyond static interworking using gateways, as done today. The main difference is that in registry composition both interface interworking and data interworking are done on the fly.

A third problem related to registry composition is negotiation. Indeed, before composing the registries, the entities responsible for the composition in each network should negotiate a composition agreement. Registry composition in Ambient Networks is an autonomous process. Therefore, an on-line negotiation framework is needed. A signaling framework is also needed, to allow the exchange of the negotiation messages between the involved entities.

The objectives of this thesis are as follows:

- **Identify how is the composition initiated and when, what are its main steps, and which entity orchestrates it.**
- **Enable clients to publish and discover information after registry composition**
- **Provide an architecture for the negotiation of registry composition.**
- **Provide a signaling framework for the composition.**

This implies the definition of new functional entities, principles, algorithms and interfaces, and their implementation and evaluation.

I.3 Summary of contributions

This section summarizes the main contributions of the thesis and gives pointers to the related papers we have published. Most of these contributions have been proposed to the AN consortium and are now part of the AN concept, as defined by the European Union's Sixth Framework Program.

- **Critical review of related work:** We have derived general requirements for the composition architecture, and specific requirements that are related to information publication and discovery, negotiation and signaling. We have also reviewed related work in light of the requirements.

- **General architecture for registry composition** ([8], [9])**:** We have proposed a new functional entity and a new procedure to orchestrate registry composition. The new entity is called Registry Composition Entity (RCE). We have identified the sub-functional entities that make up the RCE and the role of each one of them. We have also identified and analyzed the potential approaches to registry composition and interworking between heterogeneous registries. A proof-of-concept prototype was implemented to show the feasibility of our proposal.
- **Interface Interworking** ([9], [10])**:** We have proposed a new architecture for interface interworking, based on a peer-to-peer overlay network. We have chosen the concept of peer-to-peer overlay networks because it enables scalability, full decentralization and self-organizing. The overlay network is created and configured on-the-fly, and its creation process is orchestrated by the RCE.

  The architecture includes procedures for the on-the-fly creation and churn of the overlay network, procedures for information publication and discovery after composition, and an overlay protocol. A proof-of-concept prototype was implemented.
- **Data Interworking** ([9], [11])**:** We proposed a new architecture for data interworking, using as basis the interface interworking solution. The architecture reuses mechanisms and algorithms from the federated database systems and proposes new procedures to solve data composition and interworking autonomy. We have also implemented a proof-of-concept prototype.

- **Negotiation architecture** ([9], [12], [13]): We proposed a new architecture for on-line negotiation. To the contrary of existing negotiation solutions, our architecture allows a third party to create and validate the composition agreement-proposals. This will enable the negotiation to proceed even if one of the negotiating parties does not have the reasoning logic to create the proposals or does not have enough resources to execute this logic.

  The proposed architecture includes a template for the composition agreement proposals, a negotiation protocol, and a discussion of the main steps of the negotiation. The protocol definition includes the protocol entities and messages, and a description of the negotiation-related state diagrams. Correctness requirements for the negotiation protocol were derived and used to validate the protocol using SPIN, a software tool for simulating and validating processes in a distributed system.

- **Signaling Framework** ([9], [14]): We proposed a new signaling framework for registry composition in Ambient Networks. It is a backward compatible extension of the IETF Next Step in Signaling (NSIS) framework. NSIS allows signaling about a data flow along its path. We selected NSIS as the basis for our framework because it is modular and easily extensible. Furthermore, it has already been successfully used as the basis for signaling in different areas (e.g. QoS, signaling through mailboxes such as firewalls).

  The proposed signaling framework is used for both the negotiation and execution of the composition agreement. It was simulated using OPNET -- a software tool for network modeling and simulation--, and measurements was taken regarding the negotiation time delay and the network load in terms of number of exchanged messages. The measurements show that the delays and the network load remain acceptable.

I.4 Thesis organization

The rest of this thesis is organized as follows: Chapter 2 discusses network cooperation in 3G network and gives background information on Ambient Networks and network composition. Chapter 3 presents the identified requirements, and reviews the related work. Chapter 4 is devoted to the general architecture. It describes the new functional entity (i.e. RCE), presents the overall composition procedure, and discusses the potential approaches to registry composition and to interworking between heterogeneous registries. Chapter 5 discusses information publication and discovery after composition. It starts by presenting the architecture for interface interworking, and then describes the data interworking architecture. Chapter 6 describes the proposed negotiation architecture, and Chapter 7 describes the signaling framework. Chapter 8 presents the prototypes, along with performance evaluation. Chapter 9 describes the formal validation process and results of the negotiation protocol. It also presents the simulation models and results for the signaling framework. Chapter 10 concludes the dissertation and discusses items for future work.

CHAPTER II: Background

This chapter starts by introducing network cooperation in 3G networks and its shortcomings in order to motivate the need for Ambient Network (AN) composition. Next, it provides background information on ANs. Then, it presents network composition in the context of ANs and discusses how it overcomes the limitations of 3G network cooperation.

II.1 Network cooperation in 3G networks

IP Multimedia Subsystem (IMS) is the key element of 3G networks [15] . It is an overlay network on top of the packet-switched network, providing multimedia services to mobile end-users. The IMS architecture is organized in two layers: service layer and control layer. This section discusses network cooperation at the control layer of IMS and pinpoints its shortcomings. We have selected IMS because it is the quintessence of what can be done in today's networks when it comes to network cooperation. We start by a brief introduction of the IMS architecture, before discussing network cooperation at the control layer and its shortcomings. This cooperation is also known as interworking in the literature.

II.1.1 IMS Architecture

IMS uses the Session Initiation Protocol (SIP) [16] to provide multimedia services to the end users. Figure II.1 presents the IMS architecture. The service layer includes a set of IMS application servers (AS) that host and execute value-added IMS services (e.g. IP multimedia conference, divert incoming calls to an email address). The control layer comprises the core IMS control nodes responsible for call or session management (e.g. set-up, modification, teardown, charging).

One of the core elements of the control layer is the Call Session Control Function (CSCF) that is a set of SIP servers or proxies, used to process SIP signaling packets in the IMS. The CSCF servers are of three types: Proxy-CSCF (P-CSCF), Interrogating-CSCF (I-CSCF) and Serving-SCSF (S-CSCF). P-CSCF is the first point of contact -- in the signaling plane -- between the IMS terminal and the IMS network. Its main functionality

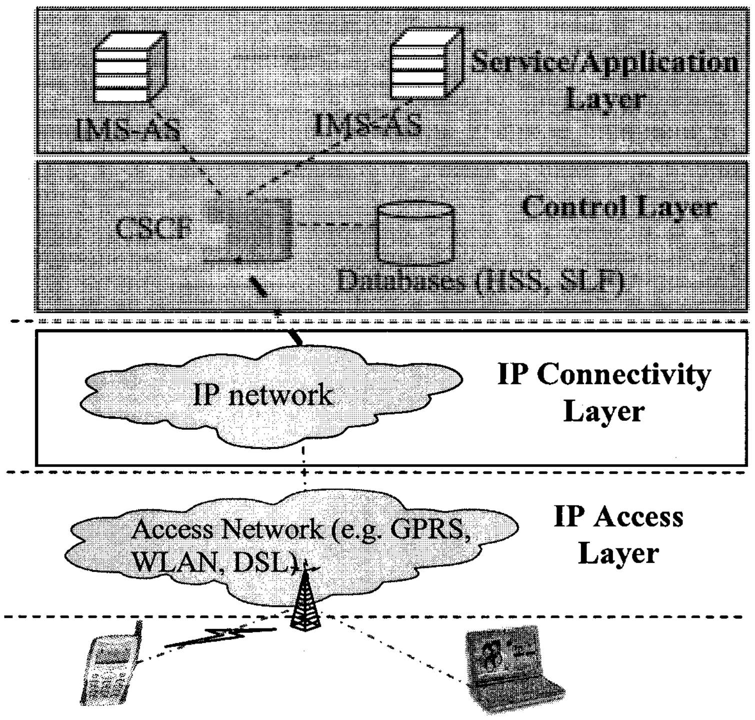


Figure II.1: Simplified IMS architecture

is to authenticate the user and to generate charging information.

The I-CSCF is a SIP proxy located at the edge of an administrative domain and it is usually located in the home network. The S-CSCF is the central node of the signaling plane. It is a SIP server that can act as a SIP registrar. The S-CSCF performs the session control and is always located in the home network. Another element in the control layer is the Home Subscriber Server database (HSS) that stores the user profile, which includes all the user-related data required to handle multimedia sessions (e.g. user's location, telephone records, security information, services to which the user is subscribed). The user can connect to an IMS network through the IP connectivity layer, using various access technologies (e.g. GPRS, WLAN).

II.1.2 Network cooperation at the control layer

Cooperation at the control layer between two IMS networks is principally meant to provide roaming, which enables end users to use their mobile terminals in networks other than their home networks. Figure II.2 presents an example of a roaming scenario. Bob is roaming to a network outside his home network, and he wants to contact Alice, a user in his home network. The sequence of messages exchanged to set up a connection between the two users is presented in Figure II.3. For the sake of clarity, we assume that the two users are served by the same S-CSCF.

The connection establishment procedure has two phases: registration and session set-up. In the registration phase, the IMS terminal requests the authorization to use the IMS service by registering with the IMS network. This is done by sending a SIP REGISTER request to the P-CSCF, which forwards the request to the I-CSCF in the home network. The I-CSCF then contacts the HSS to verify if a roaming agreement exists between the visited network and the home network and if an S-CSCF is already assigned to the user. It sends a User Authentication Request (UAR) to the HSS (step 3), which answers with a User Authentication Answer (UAA) message (step 4). If this is not the first time the user registers, the UAA includes the address of the S-CSCF allocated to him. If this is the first registration, the UAA includes a set of S-CSCF capabilities that the I-CSCF can use to choose an appropriate S-CSCF for the user. The I-CSCF then forwards the REGISTER message to the S-CSCF that proceeds with the user authentication. The S-CSCF sends a Multimedia-Auth-Request (MAR) message to the HSS to save its address for future usage, and downloads the user profile for authentication purposes (step 6). The HSS responds with a Multimedia-Auth-Answer (MAA) (step7). The OK message (step 8) is sent back to the user to indicate the success of the REGISTER request.

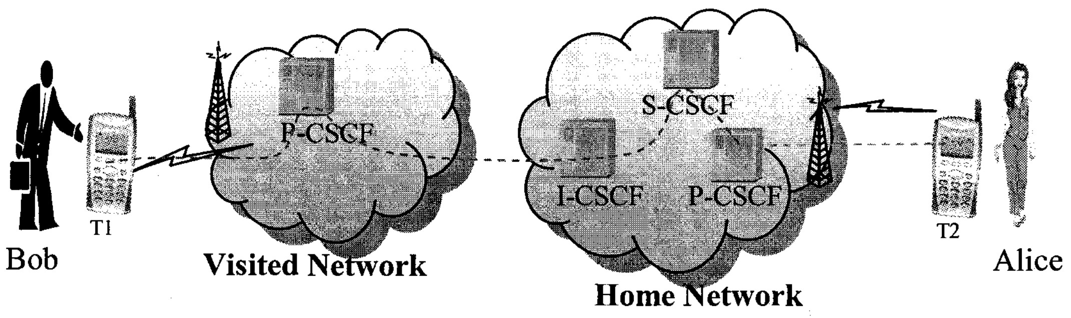


Figure II.2: A roaming scenario in IMS networks

When Bob decides to communicate with Alice (i.e. session set-up phase), his terminal issues an INVITE request that it sends to the P-CSCF (step 11). The P-CSCF forwards the request to the S-CSCF which he had gotten the address for during the registration phase (i.e. in the OK message). The S-CSCF routes the INVITE request to Alice's terminal, thought the P-CSCF serving Alice. Alice's terminal answers with an OK response to inform the caller that it accepted the session. The response traverses the same proxies the INVITE message traversed. The caller confirms the receipt of the OK response via an Ack message. The session set-up Phase ends with the establishment of a communication session with the destination terminal.

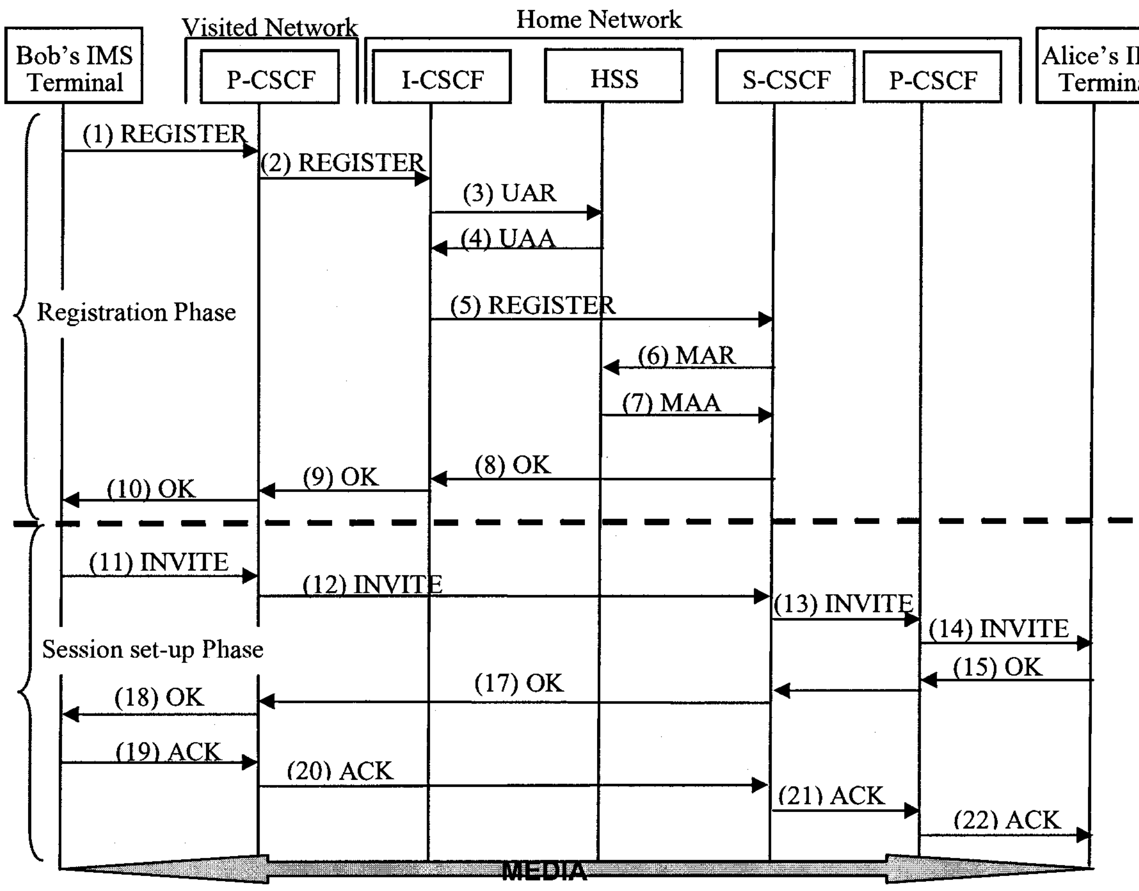


Figure II.3: Simplified sequence diagram for connection establishment

II.1.3 Limitations of network cooperation in 3G

We divide the shortcomings of today's network cooperation at the control layer into two categories: general shortcomings and shortcomings related to specific tools/frameworks used for the cooperation. For the general shortcomings, the cooperation is based on an off-line negotiated roaming agreement between the user and his or her operator and between the latter and the operator of the visited network. In the registration phase of the previous scenario, for instance, an off-line agreement between the visited network and the home network and an off-line agreement between Bob and his operator are required. The HSS is manually configured to reflect these agreements. If there is no off-line roaming agreement between Bob and his operator, Bob will be refused access to the IMS network provided by this operator. If there is no off-line agreement between the two operators, Bob will be refused roaming via this particular visited network, even if he has a roaming agreement with his operator.

The need for off-line agreements allows access to only a limited range of services and makes cooperation time consuming and sometimes impossible. Indeed, this cooperation can only work with pre-arranged and fixed services between a pre-known set of operators, which have been clearly identified in the manually created agreements.

Another general related shortcoming is the lack of session mobility support. Today's network cooperation at the control layer provides support for user mobility, terminal mobility and limited support for service mobility. User mobility is the possibility to have access to one's services independently of the terminal used. Terminal mobility refers to the possibility of having access to ones services when moving in the network. Service mobility is the possibility to have access to one's services when roaming in a different network. In today's networks, the range of services that are "mobile" is pre-determined by the roaming agreement and is usually limited to the basic service (e.g. two-party voice call). Session mobility refers to the user's ability to continue an ongoing session while switching between terminals or changing their attachment point while moving.

The shortcomings related to the tools used are basically SIP related. Indeed, IMS uses SIP as its session control protocol. SIP is a signaling protocol that allows only limited functionalities. It only allows signaling for session set-up, modification and tear down. It cannot, for instance, be used to negotiate an on-the-fly roaming agreement, nor can it be used to dynamically update the HSS to add access to a new service in a user profile. To overcome the off-line-agreement limitation in the previous scenario, there should be a mechanism and related protocols that will allow Bob to establish his call, even if there is no prior roaming agreement between the two operators or between Bob and his operator. When HSS in the home network discovers that there is no agreement with the visited network, it should be possible for the two operators to dynamically negotiate one (e.g. using previous agreements or previously-established agreement template). When the S-CSCF receives Bob's REGISTER request, it could interpret it as a request for the roaming service (if required), and initiate a negotiation with Bob, triggering a dynamic modification of Bob's profile within the HSS, in order to reflect the new status. Ideally, all of these actions should be transparent to the end user or require minimal user intervention.

Ambient network composition aims at overcoming these different shortcomings. It provides a means for on-line agreement creation and execution, and provides an enhanced support for mobility.

II.2 Ambient Networks

ANs include several functional entities [1] [17]. This section presents the overall architecture and the detailed description of media delivery, one of the functional entities (FEs). We have selected media delivery because we will use it later in describing a network composition scenario.

II.2.1 Overall architecture

The overall AN architecture (Figure II.4) includes three main components: the ambient connectivity, the Ambient Control Space (ACS) and the ambient interfaces. Ambient connectivity abstracts the existing network infrastructures to which the AN functionality is added. The ACS encompasses the control layer functional entities, such as Quality of Service FE (QoS-FE), Network Advertisement and Discovery FE (NAD-FE), Context Management FE (CM-FE), Composition FE (C-FE), and Multi-Radio Access FE (MRA-FE). QoS FE allows dynamic control of QoS to be technologically independent. NAD-FE provides an advertisement mechanism to enable an AN and its FEs to be advertised to other ANs and enables the FEs of a given network to discover other ANs and their FEs. CM-FE manages context (i.e. user-related or network-related information) within and across AN borders. It enables the collection, processing and dissemination of the context information to the interested entities. C-FE is one of the key functions of the ACS [4], and is responsible for orchestrating the network composition procedure. MRA-FE is another important FE of AN architecture [18]. It allows transparent and flexible advertisement, discovery and selection of appropriate access networks to serve each particular session, while hiding the heterogeneity of the accesses' technologies. MRA-FE also provides service continuity when the user moves between accesses and simultaneous communication over multiple accesses. Media delivery is discussed in the next sub-section.

The ambient interfaces are divided into three types: Ambient Service Interface (ASI), Ambient Network Interface (ANI) and Ambient Resource Interface (ARI). The ASI enables applications in the service/application layer to exploit the ACS capabilities. The

ANI allows communication between different ANs. The ARI provides the ACS with the necessary control mechanisms to manage the connectivity layer resources (e.g. routers, switches, proxies, media gateways). The AN architectural layers: Service/Application, Ambient Control and Ambient Connectivity, correspond to the Service/Application Layer, Control Layer and IP Connectivity Layer, respectively, of the IMS architecture.

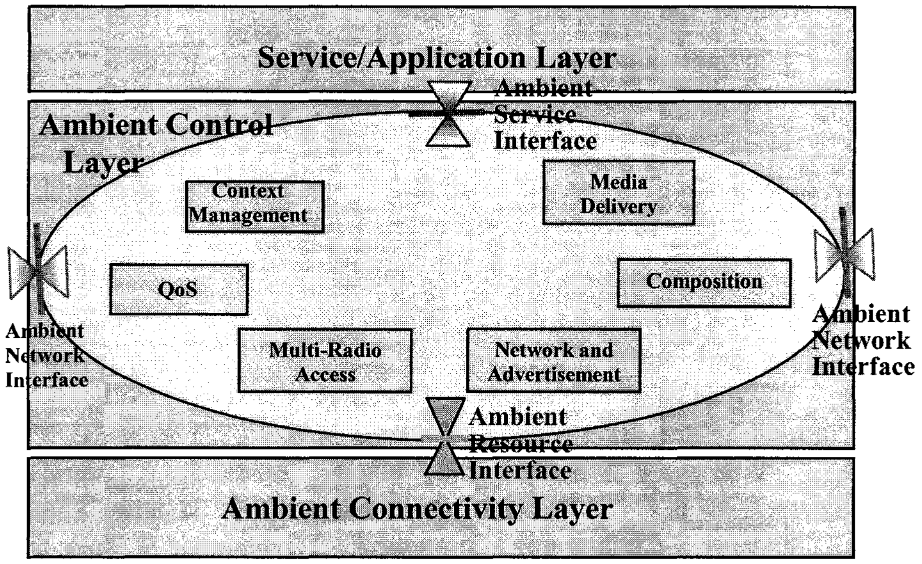

Service/Application Layer
Ambient Service Interface
Ambient Control Layer
Context Management
Media Delivery
QoS
Composition
Multi-Radio Access
Network and Advertisement
Ambient Network Interface
Ambient Network Interface
Ambient Resource Interface
Ambient Connectivity Layer

Figure II.4: Ambient Network architecture

II.2.2 Media delivery

Service-aware Adaptive Transport Overlays (SATO) is a concept developed in ANs that is responsible for the provisioning of advanced and customized media delivery services across heterogeneous networks [19]. It dynamically adapts the content to deliver, according to the user's preferences, network context, the desired end device and the services(s) to support. It can also be used to provide new network value-added services like virus scan, pro-active caching and P2P services such as Voice over IP (VoIP). In sharp contrast to today's content delivery networks and overlay systems that are limited to a specific service (e.g. Skype for voice and instant messaging but not for IPTV), SATO supports all types of services.

Figure II.5 presents the SATO architecture. It is composed of end-devices that are either clients – called SATOClients (SC), or servers – called SATOServers (SS), and a set of intermediate nodes. The intermediate nodes host the so-called SATOPorts (SP) that are the components responsible for data processing (e.g. media transcoding). When a media transport service is needed, an analysis of the required SPs is performed, and a service chain is created to represent the order in which they have to be executed. Next, the actual nodes running the different SPs are chosen based on the service requirements (e.g. QoS, security level). The SATO routing algorithm is then executed to select the best path for end-to-end service delivery, based on the QoS requirement, and the OSL (Overlay Support Layer) routing table is configured accordingly. The OSL is responsible for forwarding the received data to the correct SP that corresponds to the correct SATO.

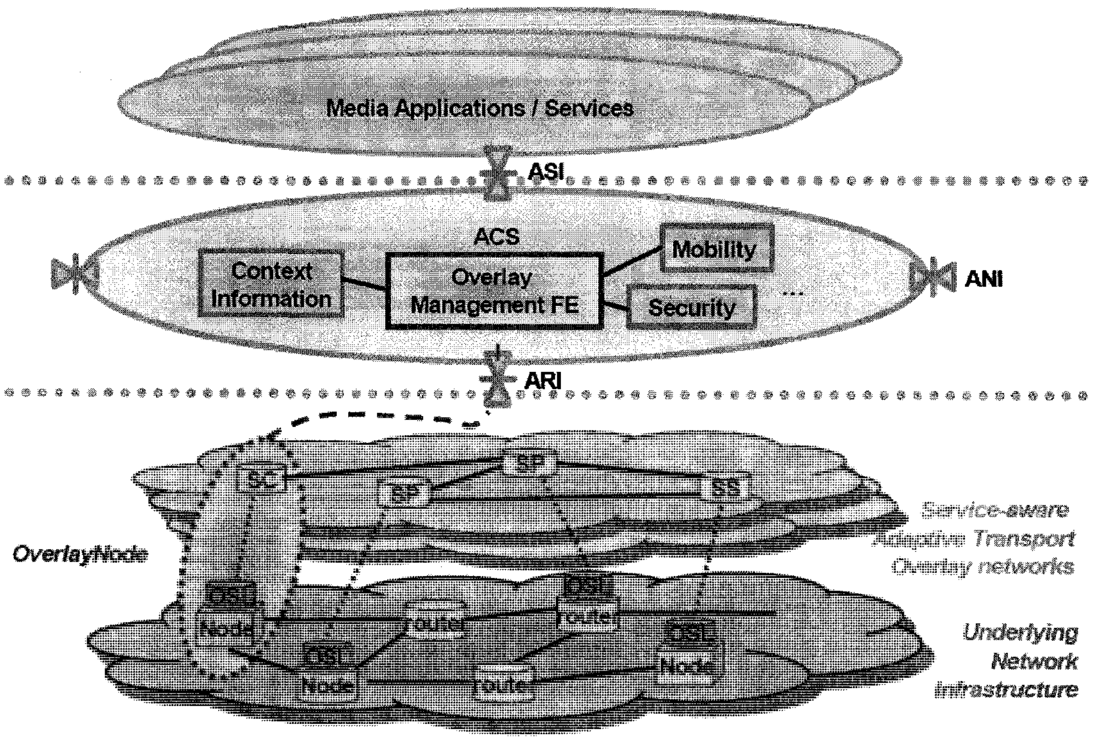


Figure II.5: SATO network architecture

II.3 Network composition

ANs support different degrees of composition, to accommodate a wide range of situations [3][20]. The composition degrees represent the level of cooperation between the composing networks and describe how resources are managed and used after composition. This section presents the network composition degrees, followed by a discussion of the composition steps. We provide a comprehensive survey of network composition in reference [21].

II.3.1 Composition degrees and scenarios

Three degrees of AN composition are possible: network interworking, control sharing and network integration (Figure II.6). They represent the level of cooperation between the composing networks and describe how resources are managed after composition. Network interworking is the most common degree of network composition in real life, where each network keeps control over its resources. Composition allows the coordination of the tasks performed in each of the composing networks. One example is dynamic roaming between two operators that have agreed that users are automatically authenticated in their home network, as opposed to the static roaming of today's networks.

Indeed, unlike roaming as it is today, if no agreement exists between the user and the visited network or between the latter and the home network, the agreement is created on-the-fly as part of the composition process. Another example is to use network interworking to provide dynamic and automatic access to a new service (e.g. Internet access). This is the case of a scenario presented in reference [22]. In this scenario, John's Personal Area Network (PAN-J) enables access to the Internet only via the UMTS interface provided by John's mobile phone. Anne's PAN (PAN-A) provides Internet access via an Ethernet link. This access is more reliable and cheaper than the UMTS access. By composing the two networks, John will have seamless access to this service. Indeed, when approaching PAN-A, PAN-J will automatically detect that accessing the Internet via Anne's Ethernet link is more suitable, and then it will automatically switch from UMTS to WLAN access. The network handles all the issues related to connecting John to the Internet via the Ethernet link transparently and automatically.

With control sharing, composing networks remain separate but share some of their resources. They may exercise joint control over the shared resources, but they maintain control over their individual resources. If common control of certain resources is required, a new AN is created to maintain these resources (Figure II.6.c). One example is when several PANs build a dynamic ad-hoc network for a conference, where they share some files and the same internet access. If the control of only certain resources is delegated to a particular AN, the composition is called control delegation and is a special case of control sharing (Figure II.6.d). A moving network dynamically delegating authentication to an access network is a typical example.

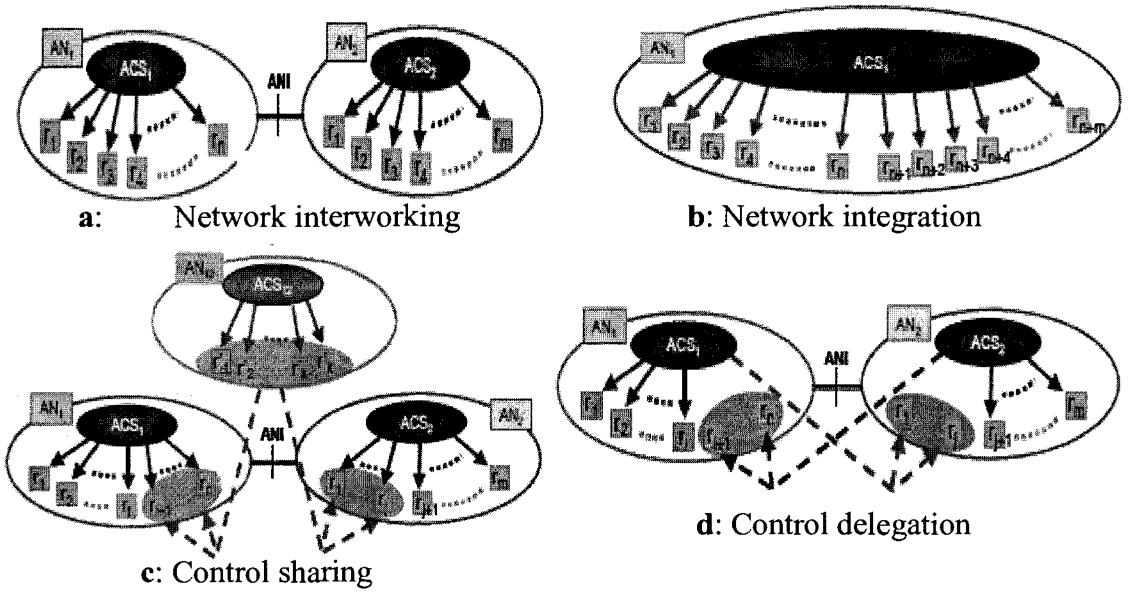


Figure II.6: Network composition degrees

In network integration, all of the participating networks merge into a new common composed network (Figure II.6.b). The composed network consists of all of the logical and physical resources of the composing networks. An example is a step-by-step creation and expansion of a mobile infrastructure network, where groups of equipment are configured and tested as separate networks, then integrated into the infrastructure network. After integration, the individual networks are no longer observable from outside. For more information on AN composition types, the reader can consult reference [23]. Figure II.7 presents a comprehensive network interworking composition scenario [24]. In this scenario, it is 08 AM, Bob is at his home, and he is willing to attend a phone conference with some colleagues at 08h30 AM. Bob owns a Personal Area Network (PAN) that consists of a laptop and a mobile phone. To access the conferencing server, Bob's PAN needs to be connected to Internet.

When Bob's PAN is bootstrapped, it detects the home network and decides to compose with it. The composition is triggered by the need for Internet access. Next, Bob asks to log on to the conference server. This triggers the creation of a SATO overlay network between Bob's laptop and the conferencing server, for end-to-end service delivery. Since the conference has not yet started, Bob decides to watch the news while waiting. Therefore, another SATO network is set up with an IPTV Server (Figure II.7.a). At 08h30, the news session is automatically brought to the background and the conference session is resumed. After some time, Bob has to leave for an appointment at his office. He turns his laptop off, which triggers the hand over of the two running sessions to the mobile phone.

While on the way to his office, the signal connecting him to his home network gets weaker (Figure II.7.b). At the same time, using access discovery, his PAN is aware of the access networks in his vicinity. When the original signal gets too weak, the most suitable access network is selected by the MRA-FE, the PAN composes with the new access network and the ongoing session is adapted to the new context information (e.g. according to the link conditions of the new access network, a new transcoding SATO Port is added to adapt the multimedia streams to the new bit-rate and special resolution).

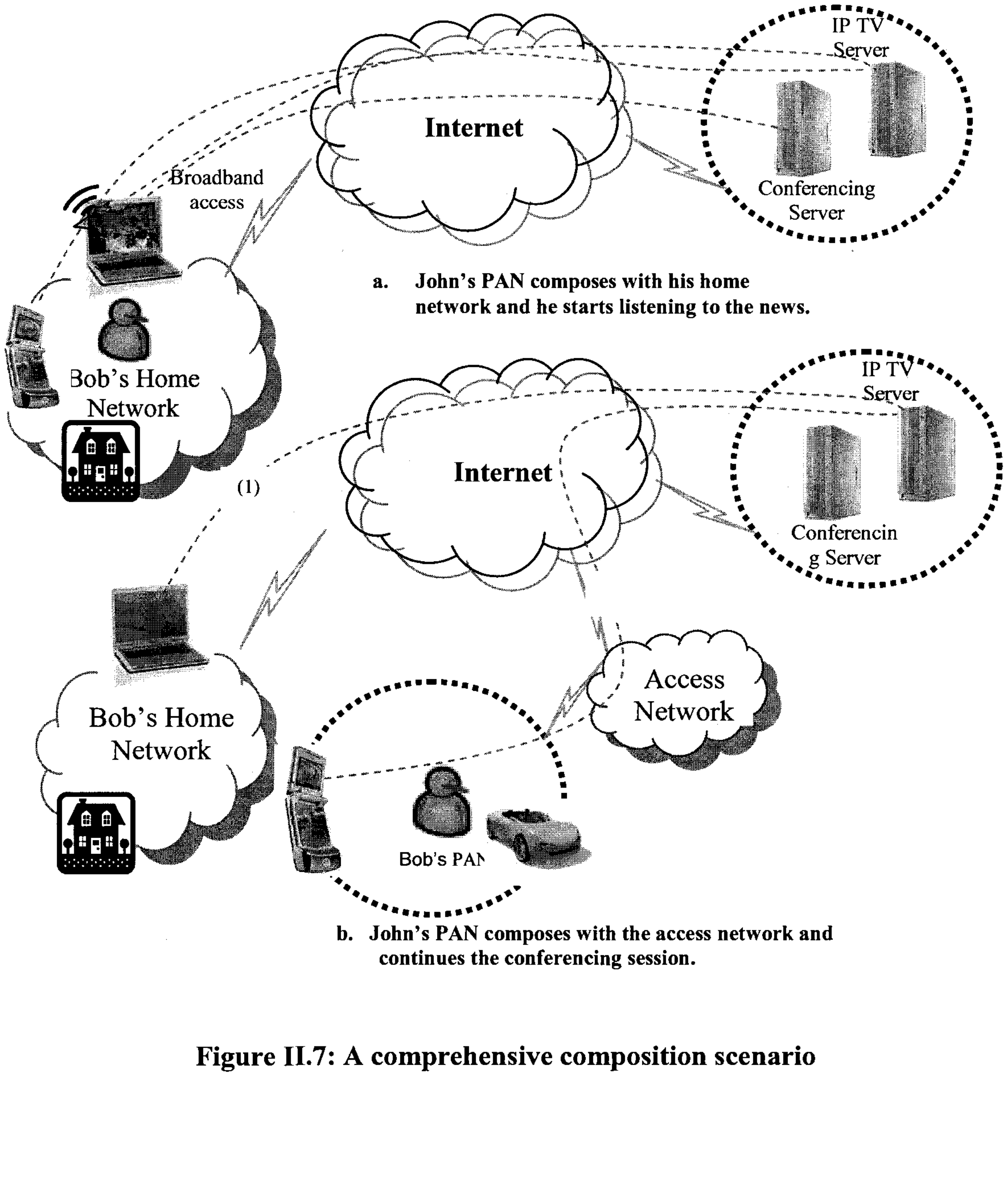

IP TV Server
Internet
Broadband access
Conferencing Server
Bob's Home Network
a. John's PAN composes with his home network and he starts listening to the news.
IP TV Server
(1)
Internet
Conferencin g Server
Access Network
Bob's Home Network
Bob's PAN
b. John's PAN composes with the access network and continues the conferencing session.

Figure II.7: A comprehensive composition scenario

II.3.2 Composition steps

The composition process encompasses a certain number of distinct phases. To realize these different phases, the composing networks need to exchange different types of messages -- in the different phases of the composition -- in order to coordinate and regulate the composition. Therefore, a generic signaling framework is required. In this section, we start by presenting the composition phases and then we describe the signaling framework.

A. Composition phases

The composition process comprises five phases: media sense, discovery/advertisement, security and interworking establishment, composition agreement negotiation and composition agreement execution (Figure II.8). In the first phase, an AN discovers a medium that can allow communication with a neighboring network. This includes the identification of a link to a remote network not in the physical vicinity of the interested composing network. Media sensing may be triggered by different events, such when a PAN needs to compose with an access network to provide Internet access to its owner, or the case where two operator-managed networks are connected for the first time. Another example is where an operator connects a new access point to its network. The composition can also be triggered by a user application (e.g. a user requires composition with a remote network in order to achieve a certain QoS needed by the application at hand).

After a communication medium has been established, a composing network may either pass to an advertisement or a discovery phase. In the first case, the network advertises its resources, capabilities, services and possibly the related pricing information to the other network(s). The advertisement message includes the network identifier, which is used to bind each advertisement to a particular network. In the discovery phase, the network listens to the other networks' advertisements or actively discovers its neighbors by sending a discovery request.

After the candidate ANs for composition have been selected thanks to the discovery/advertisement phase, a basic security and interworking connectivity is established between these networks. This may include authentication and authorization of the different ANs by a trusted third party, and the generation and sharing of a cryptographic session key. The composing networks then negotiate the composition agreement, where they agree – among other items -- on the identifier of the composed network, on how the resources of the composing and composed networks are accessed and managed, and decide on the compensation information. The composition agreement is digitally signed by each network in order to guarantee non-repudiation. The composition process is then completed by the composition agreement execution phase, where the network elements are configured to reflect the content of the negotiated agreement.

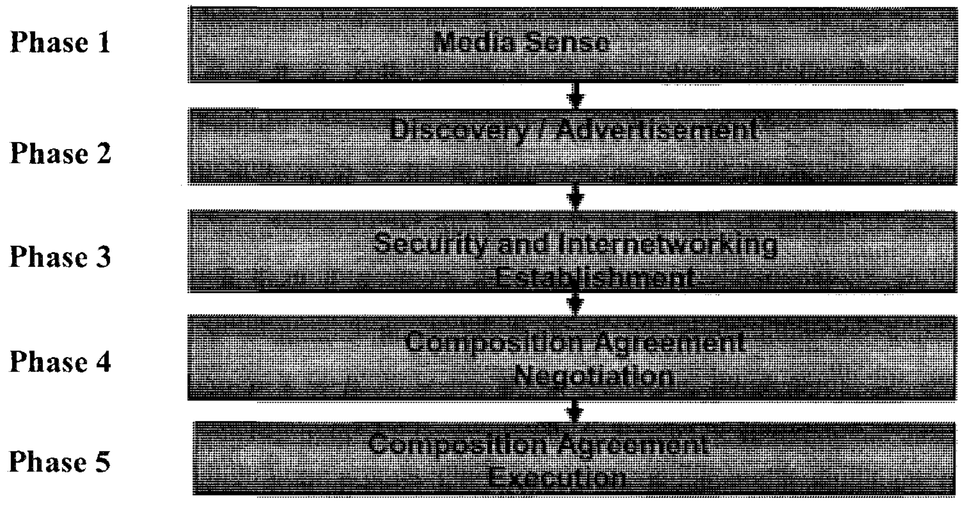

Phase 1
Media Sense
Phase 2
Discovery / Advertisement
Phase 3
Security and Internetworking Establishment
Phase 4
Composition Agreement Negotiation
Phase 5
Composition Agreement Execution

Figure II.8. Network composition procedure

B. Signaling framework

A signaling framework for AN should meet two main requirements: support of symbolic names and support of session mobility. Indeed, to address other peer networks or functional entities, an AN may use symbolic names (e.g. CompositionFE@Net1.com) instead of IP addresses. Furthermore, an AN is a very dynamic environment, where entities may leave and join the network at any time, using the same or a different IP address and port number.

Generic Ambient Network Signaling (GANS) [25] is a generic signaling framework, conceived to support these requirements. It is a backwards-compatible generalization of the IETF Next Step in Signaling (NSIS) [26], a suite of protocols for signaling about a data flow along its path (reference [27] gives tutorial-level information on NSIS). It extends NSIS by allowing control signaling between FEs rather than exclusively along the data path and by supporting symbolic names and session mobility. And, as in NSIS, GANS architecture is composed of two layers: a generic lower layer named GTLP

(GANS Transport Layer Protocol) and a signaling application upper layer called GSLP (GANS Signaling Layer Protocol) (Figure II.9). GTLP provides common transport layer services to higher layer signaling applications, such as locating signaling peers (i.e. a peer FE in another AN), establishing signaling relation and security association between pairs of signaling FEs and maintaining signaling relations if, for example, a peer FE is relocated or reconfigured. The GSLP includes the actual signaling applications (e.g. negotiation of networks' composition).

GTLP comprises two main building blocks: Destination Endpoint Exploration Protocol (DEEP) and Extended General Internet Signaling Transport (EGIST). DEEP is a generic name resolution protocol for heterogeneous environments. It resolves symbolic names into host ID/locator (e.g. IP address), relying on existing name resolution systems such as DNS, multicast DNS [28] and Link Local Multicast Name Resolution [29]. A mechanism is provided to allow dynamic update and storage of the IP address-symbolic name binding. EGIST provides the actual transport framework for the signaling applications messages. It uses existing transport and security protocols (e.g. UDP, TCP, TLS), to provide the transport and security services needed. It has two modes of operation: Datagram mode (D-mode) and Connection mode (C-mode). The transport protocols used by each mode are UDP and TCP, respectively.

When EGIST receives a signaling message, if the destination counterpart is identified by its symbolic name, the latter is passed to DEEP that returns the corresponding address. Then, EGIST creates a message association with the destination end points, encapsulates the received signaling message and sends it to the destination.

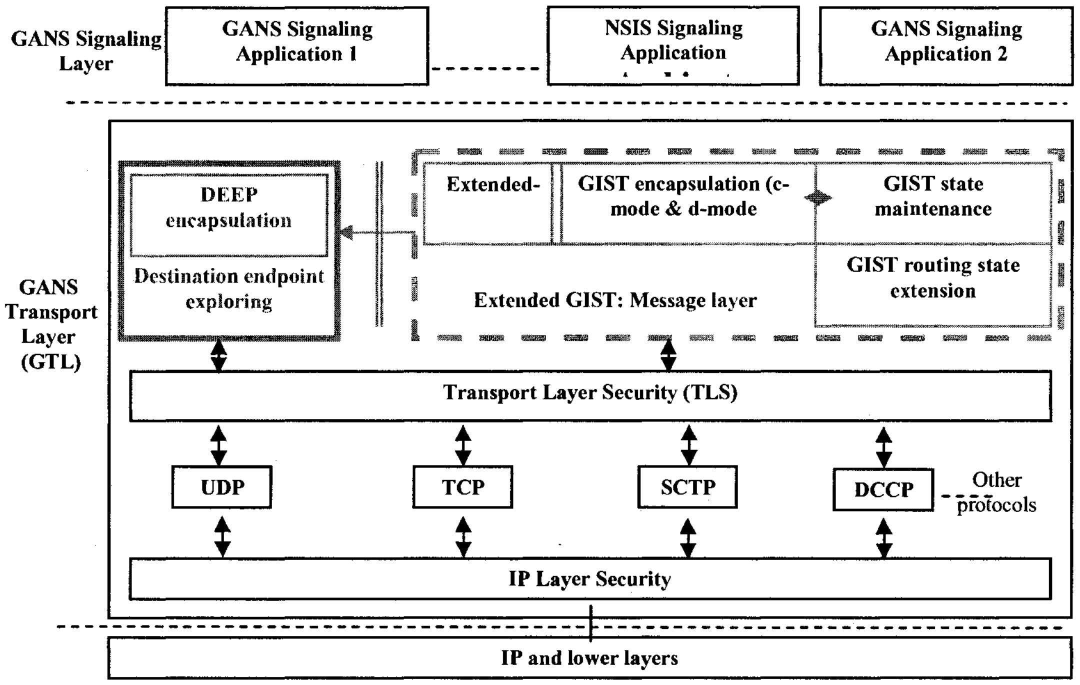

GANS Signaling Layer
GANS Signaling Application 1
NSIS Signaling Application
GANS Signaling Application 2
GANS Transport Layer (GTL)
DEEP encapsulation
Destination endpoint exploring
Extended-
GIST encapsulation (c-mode & d-mode
GIST state maintenance
GIST routing state extension
Extended GIST: Message layer
Transport Layer Security (TLS)
UDP
TCP
SCTP
DCCP
Other protocols
IP Layer Security
IP and lower layers

Figure II.9: GANS protocol stack

II.4 Conclusions

IMS-IMS control layer cooperation is limited to roaming. This is a static cooperation, based on off-line negotiation of the roaming agreement and off-line setup of the required configurations. Furthermore, it offers a limited support for mobility (i.e. session mobility not supported), and provides access to a limited set of services (i.e. those identified in the roaming agreement).

Network composition in the context of ANs enables dynamic cooperation among heterogeneous ANs. It requires no (or minimal) user intervention or off-line configuration. It is an instantaneous process, unlike today's time consuming roaming, which demands off-line roaming agreement and time to manually implement that agreement. The composition process is basically the same, independent of the technology used by the networks to compose, and independent of the type (e.g. PAN, operator network) or the size of these networks. It may be applied recursively, where a composed network may compose again. An AN can also participate in more than one composition process and may be part of different composed networks concurrently, except for network integration. Network composition also provides an enhanced support for the mobility related to today's networks, by supporting, for instance, session mobility and media flow mobility. Network composition is carried out via the ANI, over the Generic Ambient Network Signaling framework (GANS). GANS is a set of protocols that enables signaling among ANs (e.g. to negotiate the composition agreement, to carry out the inter-authentication).

CHAPTER III: Review of the related work

In this thesis, we propose an overall architecture for registry composition in ANs. This overall architecture is made up of three components: an architecture for information publication and discovery after composition, a negotiation architecture and a signaling framework. This chapter presents requirements for the overall architecture and for each of the architectural component. It also critically reviews the related work in light of these requirements.

III.1 Requirements

III.1.1 Requirements on overall architecture

The heterogeneity of the ANs and the hosted registries put stringent requirements on the overall architecture. The first requirement is that it should be independent of the type of composing networks and registries, and independent of the degree of network composition. This will make the architecture support all types of ANs, registries and AN composition degrees (i.e. interworking, control sharing and integration), with a unified composition process.

Furthermore, the architecture should enable a fully autonomous composition. Indeed, network composition is an automated process, which should run without user intervention. Therefore, as a sub-process of network composition, registry composition should be autonomous. In addition, since composed networks may need to decompose, the architecture should be designed in a way to support the decomposition of the composed registries.

Moreover, given that many ANs can compose at the same time and each network can host more than one registry (e.g. a MIB, a CIB and a UDDI registry), the architecture should scale in terms of the number of composing networks and in terms of the number of registries hosted by each network. Additionally, to take advantage of previews works, the architecture should allow the re-use of existing solutions, if any (e.g. allow the use of existing protocols and mechanisms for information publication and discovery).

III.1.2 Information publication and discovery architecture

The first requirement on the architecture for information publication and discovery after composition is that it should deal with both interface interworking and data interworking. To support interface interworking, the architecture should enable clients to access the post-composition registries that are using different interfaces than the ones used by clients. Post-composition registries are the registries that belong to the composed network. The initial registries hosted by the composing networks are called pre-composition registries. To support data interworking, the architecture should allow clients to get information with the level of granularity and the format they need, from any of the post-composition registries. The second requirement is that the interface interworking and data interworking should be done on-the-fly. This means that the interworking solution should be provided only when needed and according to the current situation, which exclude the use of any static solution such as static gateways.

The most common registries are either centralized or peer-to-peer (P2P). Therefore, the third requirement is that the information publication and discovery architecture should be suitable for both centralized and peer-to-peer registries. This means that it should be distributed and should allow dynamic self-organization.

The fourth requirement is that the architecture should be transparent to the clients. It should also not violate the publishing and discovery policies of the composed registries. Moreover, the architecture should insure the discovery of existing information in a timely and efficient manner, with at least the same probability of discovery as the existing popular P2P discovery protocols.

III.1.3 Negotiation architecture

The negotiation of the registry composition is conducted by the RCEs of the composing networks. Therefore, the overall architecture requirement on scalability implies that the negotiation architecture should scale in the number of negotiating parties (each composing network has an RCE). The second requirement is that an agreement should always be reached. This implies among other things that the execution of the negotiation protocol should terminate (e.g. it is loop-free).

Moreover, the negotiation architecture should allow a third party to arbitrate the negotiation and create the Composition Agreement proposals (CA-proposals). This will enable the negotiation to proceed even if none of the negotiating RCEs has the reasoning logic to create the proposals or does not have enough resources to execute this logic. And to take account of the nodes heterogeneity in an AN, the negotiation architecture should allow the third party to be either co-located with an RCE or be an independent entity (e.g. in case the RCE does not have enough resources to support the needed functionalities).

And since entities may leave and join at any time in ANs, the architecture should also not rely on a permanently centralized entity.

III.1.4 Signaling framework

The signaling framework should support the negotiation and execution of the composition agreement, for registry composition. Second, it should support symbolic names and session mobility, not rely on a permanently centralized entity and be lightweight. Indeed, in ANs, entities may leave and join the network at any time -using the same or a different IP address and port number- and they may use symbolic names to address destination entities. Furthermore, an AN can include different types of devices, with heterogeneous capabilities. Therefore, the signaling framework should be lightweight in order to be used by any of theses devices (e.g. including devices with limited resources).

Third, the signaling framework should be independent of the negotiation model: it should support negotiation with or without mediator, support different decision models for the negotiation (e.g. Accept-it-or-leave-it, offer/counteroffer [30])) and support different negotiation approaches. Negotiation can be either one-to-one (i.e. between two entities) or one-to-many (i.e. one entity communicating with more than one entity at the same time) or many-to-many. Therefore, to be independent of the negotiation model, the framework should also allow point-to-point and point-to-multipoint message delivery. Point-to-point message delivery is used for one-to-one negotiation. Point-to-multipoint is used for one-to-many and many-to-many negotiation. The main existing negotiation approaches are parallel negotiation and sequential negotiation [31] . These approaches are defined when the negotiating parties are negotiating multiple issues. They correspond to negotiating parties presenting all their demands/offers at once and one by one respectively.

Fourth, the signaling framework should allow exchange of the negotiation agreements and proposals. Fifth, it should be modular and extensible. Sixth it should separate the semantic of the signaling application (i.e. registry composition) from the message delivery, so that it can be easily extensible. Seventh, it should support flow-dependent signaling applications -- where the signaling messages follow the flow data path, such as in RSVP -- and flow-independent signaling applications (e.g. SIP).

Table III-1 summarizes all of the requirements for registry composition in ANs

Overall architecture	
R1	Independent of the types of composing ANs and registries and of the degree of composition
R2	Enable autonomous composition
R3	Support autonomous registry decomposition
R4	Scalability in terms of the number of composing networks and the number of registries hosted by each network.
R5	Allow the usage of existing protocols, mechanisms and frameworks, if any.
Information publication and discovery architecture	
R6	Deals with both interface interworking and data interworking
R7	Interface and data interworking should be done on-the-fly
R8	Suitable for both centralized and peer-to-peer registries.
R9	Imply no changes on the clients.
R10	The publishing and discovery policies of the composed registries should not be

	violated
R11	Insure the discovery of existing information in a timely and efficient manner
Negotiation architecture	
R12	Scalability in terms of the number of negotiating parties (sub-requirement of R4)
R13	An agreement should always be reached.
R14	Allow a third party to arbitrate the negotiation and create the CA-proposals
R15	Allow the third party to be either co-located with an RCE or be independent.
R16	Do not rely on a permanently centralized entity
Signaling framework	
R17	Support CA negotiation and execution
R18	Support symbolic names and session mobility, not rely on a permanently centralized entity and be lightweight.
R19	Independent of the negotiation model
R20	Allow exchange of the negotiation agreements and proposals
R21	Modular and extensible
R22	Separate the semantic of the signaling application from the message delivery
R23	Support flow-dependent and flow-independent signaling applications

Table III-1: Requirements for registry composition in ANs

III.2 Review of the related work

There is no existing overall architecture related to the overall architecture for registry composition. However there are existing architectural components related to the three registry composition architectural components (i.e. information publication and discovery, negotiation and signaling). Therefore, we discuss and analyse the work related to each component separately. The next sub-section is dedicated to information publication and discovery, followed by sub-sections on negotiation and signaling.

III.2.1 Information publication and discovery architecture

In this section, we review a middleware architecture for inter ad-hoc network communication, network interworking approaches, Distributed Hash Tables (DHT) composition & decomposition, and distributed and federated databases. A DHT is a distributed system that efficiently maps "keys" onto "values", and efficiently routes queries about information to the unique owner of the key related to that information [6] . The mapping of information to numeric keys is done using a hash function.

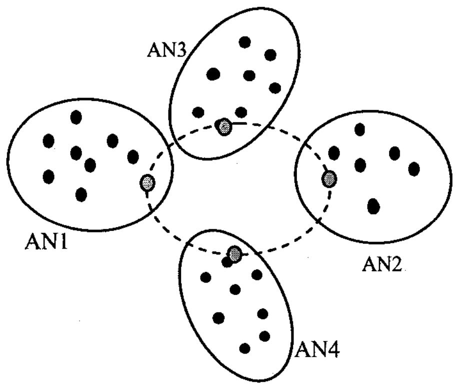
A. A middleware architecture for inter ad-hoc networks communication

Reference [32] describes an approach for the creation and composition of registries on-the-fly to facilitate interface interworking. It focuses on the connectivity between two nodes that belong to distinct and heterogeneous ad-hoc networks. It is based on a resource awareness service that enables dynamic resource discovery. It also defines a new network model, called the Xtended ad-hoc model (X-adhoc), for interface interworking between two nodes.

X-adhoc consists of the collection of the distinct ad-hoc networks that are involved. The networks may use different discovery mechanisms (e.g. Jini [33], Chord [34], UpnP [35]), and different underlying communication infrastructures (e.g. IP). It is assumed that each network has a gateway to access the other networks. The X-adhoc model creates an overlay network made up of the different gateways (Figure III.1).

After the X-adhoc is created, a Resource Registry (RR) is added to each network, and an X-adhoc Registry (XR) is added to the X-adhoc network. Each node in each involved network will publish a description of the resources it is willing to share with others in its own RR (e.g. resource id, type, technical aspects). It also publishes the policies that regulate the sharing and usage for each shared resource. The RR includes also the preferences of the resource owner. The XR maintains a record for each gateway. This record includes the gateway id, the gateway policies and the set of nodes seen by the gateway in its own network.

When a node wants to communicate with another node, it starts by querying the RR of its network to locate the destination node. If the node is in another network, the source node will start by locating a gateway in its own network. Then, this gateway will communicate with the XR to locate a gateway in the destination network. After that, the two nodes will communicate through the gateways of their respective networks (Figure III.2).



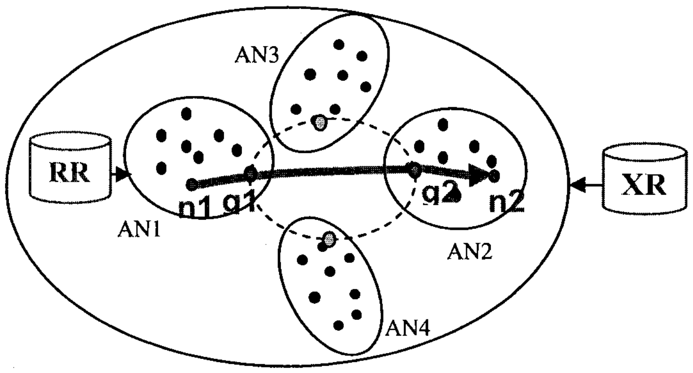


Figure III.1: An example of X-ad hoc network

Figure III.2: An example of communication Between two nodes

This approach does not address the composition of registries that already exist in the involved ad hoc networks. It actually creates new registries and somehow composes them in order to enable nodes in one network to access resources in another network. This approach only supports the interworking degree of composition, and is therefore not independent of the degree of network composition. Furthermore, it does not deal with both interface interworking and data interworking, because it only provides interface interworking. In addition, in order to achieve connectivity between different nodes, the involved networks must host the appropriate gateways that need to be created and configured offline. Therefore, with this approach, interface and data interworking are not done on-the-fly.

B. Network interworking approaches

The approaches for network interworking as known today tackle the interface interworking issue. Network interworking is provided via protocol interworking. There are three approaches [36] to protocol interworking: protocol conversion [37], protocol overlapping [38] and protocol complementation [39]. In protocol conversion, a user-transparent converter is used between two heterogeneous networks. This converter receives messages from one protocol, interprets them and delivers appropriate messages to the other protocol. Protocol overlapping modifies one of the protocols to make it absorb the other. The first protocol will act as a base to support the functioning of the second. Protocol complementation builds a virtual layer on top of the original protocols to provide a uniform view to the users.

The three approaches require the usage of gateways between the networks to interconnect. These gateways implement the converter, the absorbing protocol or the uniform protocol of the virtual layer. The gateways are created and configured offline. Therefore, with the three approaches, interface and data interworking are not done on-the-fly. They also do not deal with both interface interworking and data interworking, because they only handle interface interworking. Furthermore, they are not independent of the degree of network composition, because they only support network interworking degree of composition.

C. Distributed hash tables composition

DHT composition solutions also tackle the interface interworking issue. We split the DHT related work into two categories: DHT merging and DHT bridging. In DHT merging, all of the nodes of the DHTs to compose are merged in a unique and uniform DHT. In DHT bridging, gateways are used to enable automatic communication between the different DHT systems. The trivial way to merge two DHT structures is to move the nodes of the smaller structure to the other structure, one by one. This implies redistributing the data of the discarded structure and re-establishing the neighbourhood connections, which generates a huge amount of network traffic. Reference [40] proposes an optimization of the merging scheme for DHTs based on the Chord protocol: when joining the larger DHT, each node maintains its ID, the key-value pairs it manages and its finger-table entries (Figure III.3). The finger-table is a list of references to some long distance nodes in the DHT structure, used to optimize the search process (i.e. the search request can be sent to the next node and to all of the nodes in the finger table simultaneously).

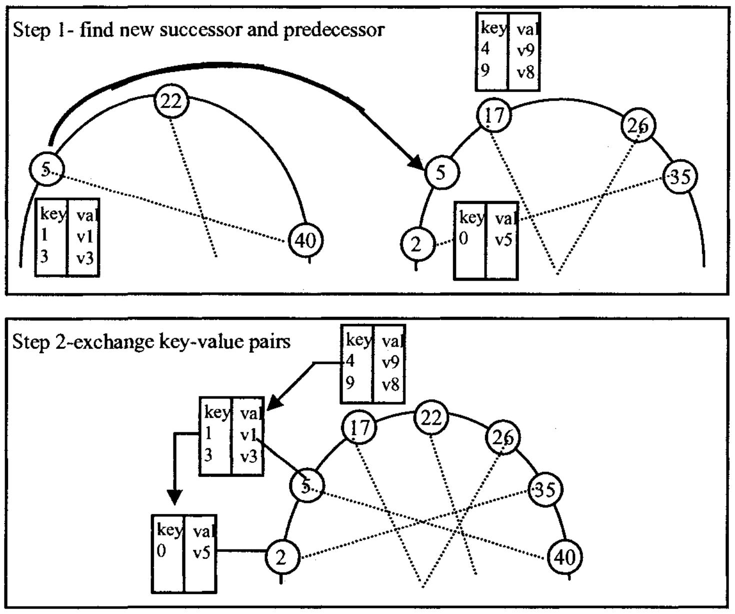
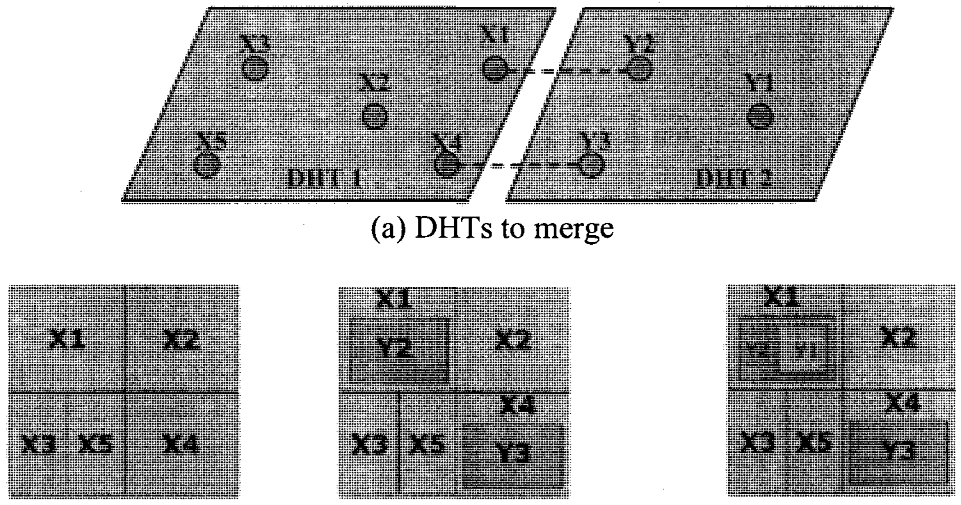


Figure III.3: Simple merging scheme for Chord

Reference [41] presents another optimization scheme, for merging heterogeneous CAN-based DHTs when their hosting wireless networks dynamically compose. The merging negotiation is conducted through the points of contact (i.e. the nodes with physical connections with other DHT-structures) of the DHTs to compose (Figure III.4.a). The points of contact of the absorbing DHT (i.e. x1 and x4) give up some of their key spaces to those of the absorbed DHT (i.e. y2 and y3). The given up key spaces are selected from within the owned key spaces, which minimizes the disturbance of the neighboring nodes (Figure III.4.c). Indeed, only the points of contact of the absorbing DHT would have to update their key spaces. The points of contact of the absorbed DHT will be responsible for distributing their obtained key space to the other members of their original DHT that wants to join the absorbing DHT (Figure III.4.d).



(a) DHTs to merge

(b) Before merging (c) After merging with Y2 (d) After merging with Y1

Figure III.4: A merging example for CAN-based DHTs

Examples of DHT bridging are presented in references [41] and [42]. In reference [41], the points of contact in the composing DHTs play the role of gateways between the two DHTs. Each node in a given DHT structure should maintain at least the address of one gateway in its network. If searched data is not found in the client's DHT, the request is forwarded to the other DHT structure through any of the existing gateways. This solution was designed for CAN- based DHTs. Reference [42] proposes a general bridging scheme for homogeneous, heterogeneous and assorted DHTs (Figure III.5). Homogeneous DHTs use the same DHT implementation and key-size (e.g. both DHTs are 160-bit Chord-based). Heterogeneous DHTs use the same implementation, but different key-size (e.g. a 160-bit Chord-DHT and a 256-bit Chord-DHT). Assorted DHTs use different implementations and/or key-size (e.g. a 160-bit Chord-DHT and a 256-bit CAN-DHT). The solution defines two types of gateways: nodes that are member of more than one DHT (e.g. node B in Figure III.5) and nodes that are physically connected to a node that is a member of a different DHT (node Z). To forward a request from DHT-2 to DHT-1, B performs the messages mapping between the two structures. Z should forward requests from DHT-2 to DHT-3. However, Z does not support DHT-3 implementation. Therefore, it sends a request to M, which will express the request in a format understandable by the DHT-3. Nevertheless, reference [42] does not provide much information on how the requests (and the answers) are reformulated before being passed to the disparate DHT.

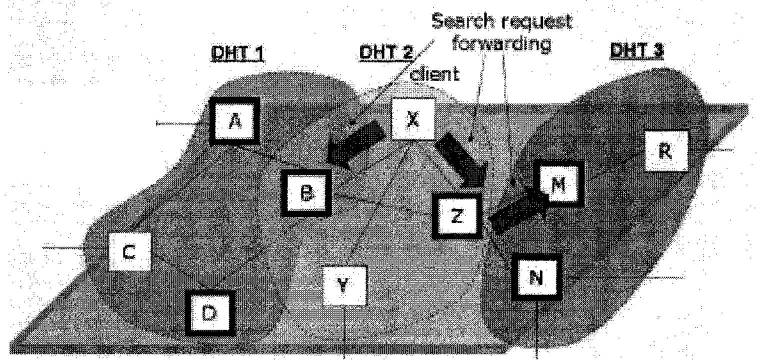


Figure III.5: A general bridging scheme

None of the two solutions --presented in [41] and [42]-- deal with both interface interworking and data interworking. Indeed, they only handle interface interworking.

Furthermore, none of them allow interface interworking to be done on-the-fly. Indeed, both solutions have the following limitations: in the case of merging, the nodes of the absorbed DHT must support the protocol of the absorbing DHT. With bridging, the gateways must support the protocols of both composing DHTs prior to composition. Furthermore, none of the two solutions is independent of the type of composing registries, because they both support only one type of registry: DHT-based registries.

D. Distributed and federated databases

Distributed and federated database systems tackle the data interworking issue. Distributed databases allow applications to operate on distributed data as if it was all managed by a single database management system (DBMS) running on a single machine, where distributed means that the data is spread across a variety of different databases, stored in multiple computers located in the same or diverse physical locations [43]. Distributed database content can also be distributed into separate partitions/fragments in the same or disparate machines. A federated system is a distributed system, usually heterogeneous, where the constituent databases are autonomous. Heterogeneity in Federated DataBase Systems (FDBS) arises due to several factors, such as the differences in data structures, semantics and the supported query languages.

FDBS systems provide some means for interface interworking. They are viewed by the clients as a single unit [44], and the location and the database platforms used for the implementation are transparent to the clients. Using their local database systems, clients can access information on other databases that make up the system. However, since the FDBS are created and configured off-line, the required request-translation mechanisms and gateways are defined and implemented off-line. Therefore, the FDBS presents the same limitations as the previous solutions, regarding interface interworking (i.e. access to the information is not fully automated). Furthermore, the composition process is not autonomous.

Data composition in the context of database systems is closely related to the data interworking problem. Much work has been done in this area since the emergence of distributed databases and FDBS [43][45][46][47]. To compose data from different sources, the FDBS field provides mechanisms for describing (or modeling) the content and structure of available data sources and for creating the related domain models. A domain model describes the domain about which information is stored in the data sources. Data source models can be created from the external schema of the sources (e.g. the database schema). The FDBS field also provides mechanisms for describing query capabilities of available data sources and mechanisms for describing the clients' queries and efficient algorithms for creating query planes, using the source descriptions. A query plan is the set of sub-queries and relevant data sources (and their execution sequence) that are required to answer a client request. The FDBS field also provides the algorithms needed to combine the results of the different sub-queries to get the final result.

Figure III.6 presents a fragment of a domain model representing military transportation planning domain (a) and a data source model embedded in the domain model (b) [48]. The modeled domain involves the movement of personnel and materiel from one location to another using various transportation means (e.g. aircraft, trucks). In the two models, the circles denote concepts and the arrows indicate relations between concepts. For example, the Port concept has two sub-concepts (i.e. Sea-Port and Air-Port) and an attribute named geocode. Shaded concepts (e.g. AFSC-Air-Port) represent those that can be retrieved directly from some database.

When a user query is received, the solution starts by identifying which data sources contain the data relevant to the concepts (e.g. Sea-Port) referenced in the received query (e.g. AFSC-Sea-Port). For those concepts which appear to have no matching data sources (e.g. Rail-Port), the solution determines if any knowledge encoded in the domain model (e.g. relationships to other concepts) permits reformulation in a way that will enable suitable data sources to be identified. A request to retrieve the names of existing rail-ports (i.e. select name from Rail-Port) for instance can be transformed to a request for the names of Sea-Ports with a railway capability (i.e. select name from Sea-Port where rail='Y'). Then, this can be reformulated to requesting the names of existing AFSC-Sea-Ports, that can be directly retrieved from a database.

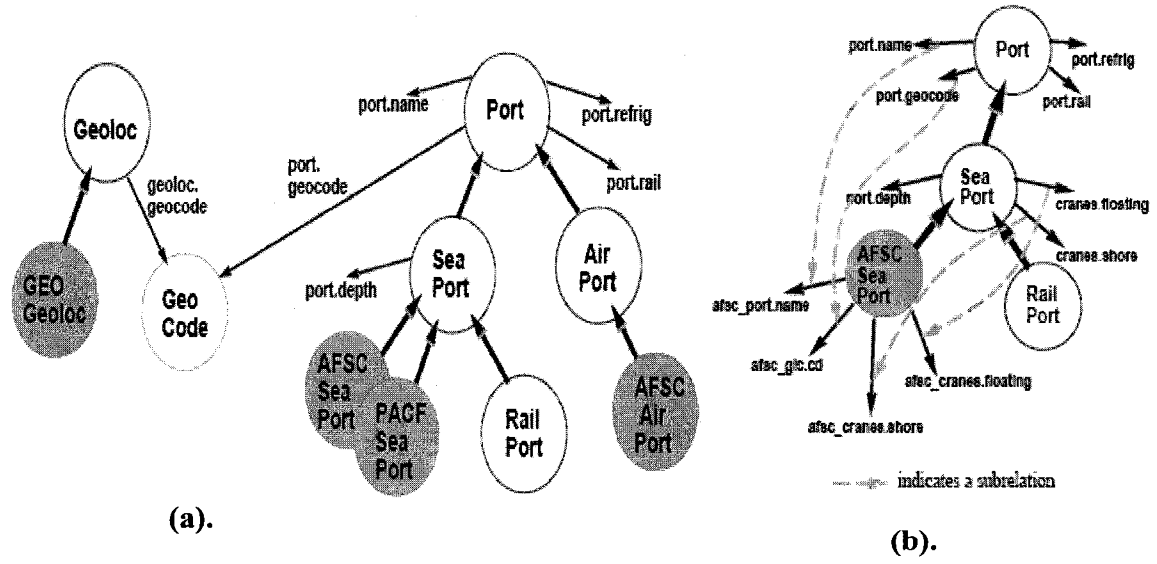


Figure III.6: Examples of a domain and data source models

After the data sources are identified, the solution creates the query plan that when executed will provide the requested information. To request –for instance- the names of all ports with rail facilities in Germany, three sub-queries are needed: One to each of the two databases that contain the related information (i.e. one containing information about ports and the other containing information about geographic locations) and one to combine the intermediate results obtained by the first two sub-queries.

To simplify the modeling tasks and the addition of new data sources, the domain model and the models of different data sources should be independent of each other. Therefore, to execute the query plan, the domain-level concepts (used in the sub-queries) should be transformed into concepts that can be retrieved directly from databases. Reference [49] provides a simple transformation mechanism: The domain model includes a mapping table that, for each concept in the data model, the table includes the corresponding concept in each data source. An example of a such table is presented in Figure III.7. The different columns present the domain model concepts and the concepts used by two different databases (i.e. C2 and S2) respectively.

The FDBS proposed solutions deal only with data interworking. They deal with the resolution of the different problems related to data heterogeneity (e.g. differences in data semantic, syntax, and granularity). However, they do not allow data interworking to be done on-the-fly, because federated database systems are created and configured offline in these works. Furthermore, they are not suitable for peer-to-peer registries, because they all assume a static environment.

UAttribute	**C2**	**S2**
Customer ID		Buyer.buyerNo
Customer Name		Buyer.name
Customer Phone		Buyer.phone
Order ID	Order.orderID	Order.orderNo
Order Date	Order.date	Order.date
Order Quantity	Order.quantity	Order.quantity
Order Status	Order.status	Order.status
Part ID	Parts.part#	Prod.prodNo
Part Hardness	Parts.hardness	Prod.hardness
Part Length	Part.length	
Part Price	Order.price	Order.price
Part Quantity	Parts.quantity	Stock.quantity

Figure III.7: Concepts mapping example

Table III-2 gives a summary of the review of the work related to information publication and discovery. If the necessary gateways are available, the X-adhoc and DHT composition approaches provide an automated registry composition for the network interworking degree of network composition (and for network integration in case of DHT merging). For network interworking approaches, the interworking process should be applied for each two heterogeneous interfaces. Therefore, the approaches do not satisfy the scalability requirement. The same is applied to the FDBS solutions, which provide a partial automation of the process allowing access to existing information. Indeed, the FDBS systems are created off-line, but when they are running, clients can have an automatic access to the information stored in any of the system constituents.

Requirement	X-ad hoc	DHT composition	FDBS	Network interworking approaches
R1	NO	NO	NO	NO
R2	YES	YES	NO	NO
R3	YES	YES	NO	YES
R4	YES	YES	NO	NO
R5	YES	YES	YES	YES
R6	NO	NO	NO	NO
R7	YES	YES	NO	NO
R8	NO	YES	NO	NO
R9	YES	YES	YES	YES
R10	YES	YES	YES	YES
R11	YES	YES	YES	YES

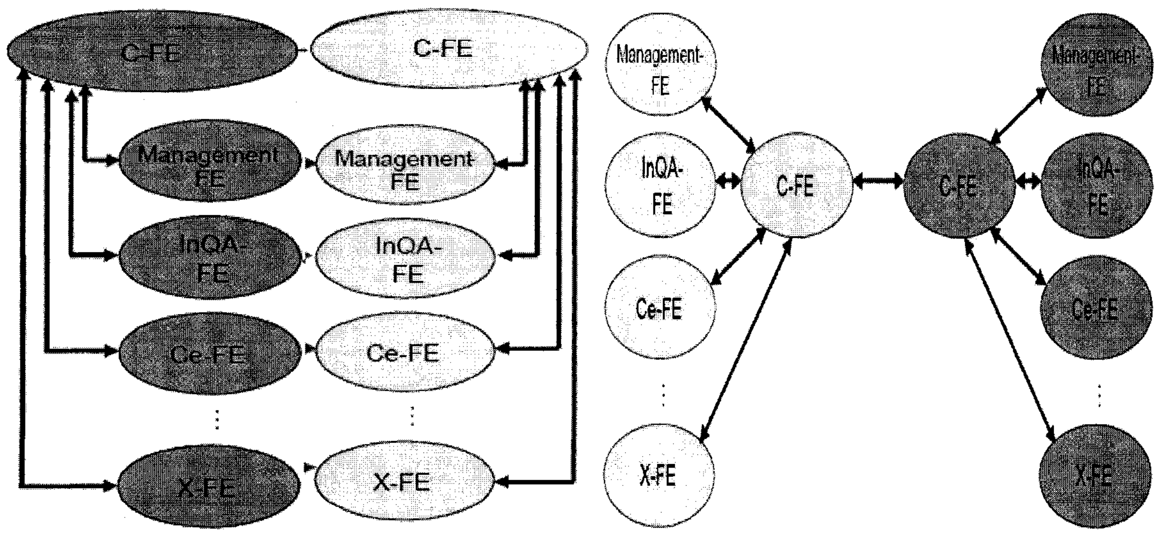
Table III-2: Summary of the review of the work related to the interface and data interworking.

III.2.2 Negotiation architecture

We split the negotiation related work into three categories: the negotiation architectures that rely on a permanently centralized entity, architectures that do not support the use of a third party, and architectures that do support a third party and do not rely on a permanently centralized entity. An example of the first category is presented in reference [50]. It is a framework for the negotiation of QoS, in wired and wireless networks, based on the central entity “Global QoS Server (GQS)” that is responsible for providing Service Level Agreements to mobile terminals.

A second example is presented in reference [4]. It presents two approaches for negotiating ANs composition: Centralized and distributed. In the centralized approach (Figure III.8.b), only the C-FEs of the composing networks negotiate with the peer entities. Each C-FE negotiates both its CA-related parameters and the parameters provided by its sub-ordinate X-FEs (i.e. the X-FEs in its network). X-FE refers to any FE in the network Ambient Control Space (ACS). When the negotiation process is triggered, each X-FE communicates its local information related to CA negotiation (e.g. recommended boundary conditions, preferences, capabilities) to the C-FE. The C-FE conducts the entire negotiation process using the received information. It controls the correlation and assessment of the recommendations received from the different X-FEs in its ACS, as well as the correlation and assessment of the proposals/counter-proposals received from the peer C-FE(s).

In the distributed approach (Figure III.8.a), both the C-FE as well as each X-FE negotiate with its respective peer FEs. Each C-FE orchestrates the negotiation of its sub-ordinate X-FEs. When triggered, the initiating C-FE signals its peer C-FEs. Then, each C-FE will signal its sub-ordinate X-FEs to start the negotiation with their respective peers. When all X-FEs (including peer C-FEs) have finished their negotiation, they signal their completion back to their local C-FE. Each C-FE will then compile the various results of its sub-ordinate X-FEs negotiations into a global and validated CA document.



(a): Distributed CA negotiation **(b):** Centralized CA negotiation

Figure III.8: CA negotiation in Ambient Networks

It is clear that the first category of solutions do not satisfy the requirement on reliance on a permanently centralized entity. The second category does not allow a third party to arbitrate the negotiation and create the CA-proposals.

The C-FE in the centralized approach (Figure III.8.a) plays the role of a mediator towards the X-FEs of its ACS, where it conducts the negotiation process on behalf of them. However, each C-FE is required to implement all the logic related to the negotiation, which fail to meet our requirement on allowing the third party to be either co-located with an RCE or be independent. In the distributed approach, each peer X-FEs negotiate directly and no mediator is supported. Therefore, the distributed approach does not allow a third party to arbitrate the negotiation and create the CA-proposals.

In the third category, to the best of our knowledge, there is no solution that allows a third party to create proposals. Furthermore, the solutions in this category are either designed to resolve a specific problem or they are general enough to be used in different circumstances. In the first case, these solutions do not address the particularities of registry composition and their related problems. In the second case, the solutions are too cumbersome, which can affect their scalability. Furthermore, we still have to specify the negotiation mechanisms and parameters concerning the registry composition. Reference [51] is a good example of a general solution.

Table III-3 gives a summary of the review of the negotiation related work. All the requirements related to composition (i.e. R1, R2 and R3) are not relevant for the existing solutions, because none of them deal with registry composition specificities.

Requirement	**Category 1**	**Category 2**	**Category 3**
R1, R2, R3	-	-	-
R12	NO	Could be if not centralized	NO (they are too cumbersome)
R13	YES	YES	YES
R14	NO	NO	NO
R15	-	-	-
R16	NO	Could be	YES

Table III-3: Summary of the review of the negotiation related work.

III.2.3 Signaling framework

We split the signaling related work into two categories: signalization for specific applications (e.g. QoS, call control) and general signaling frameworks that can be used by various applications. The first category includes Resource ReSerVation Protocol (RSVP), Session Initiation Protocol (SIP) and H.323. We review these three protocols. RSVP is a resource reservation protocol, for simplex (i.e. in only one direction.) multicast and unicast data flows. In RSVP [52], the signaling sessions are defined by the IP addresses of the source and the destination, which prevents RSVP from supporting session mobility. Furthermore, RSVP does not support symbolic names and the signaling is flow dependent. It also presents a tight coupling between the signaling semantic (i.e. resources reservation) and the delivery of the signaling messages.

SIP and H.323 are designated to call control. SIP is an IETF standard and H.323 a set of specifications from ITU-T. SIP [16] is a point-to-point protocol. It does not separate the semantic of the signaling application from the message delivery. It is not designed for negotiation and it does not support session mobility. Indeed, if the destination address changes during the same session, there is no way to deal with this change and an error message 'destination unreachable' is sent to the entity trying to contact the entity whose address has been changed. H.323 also does not separate between transport and signaling functionalities, and it does support neither session mobility nor symbolic names.

Examples of the second category are Cross Application Signaling Protocol (CASP), NSIS and GANS. CASP is a general-purpose signaling protocol suite [53][54], which is employed to establish a control state about data flow along its path in the network. Figure III.9 presents CASP architecture. It consists of a generic messaging layer and a client layer. The messaging layer transports the signaling messages between the signaling peers, where as the client layer consists of a next-hop discovery client and any number of signaling client protocols (e.g. QoS client for QoS resource reservation). CASP addresses the session mobility problem by introducing the concept of a location-independent session identifier. It also reuses the existing transport and security protocols and decouples message transport from the next signaling hop discovery.

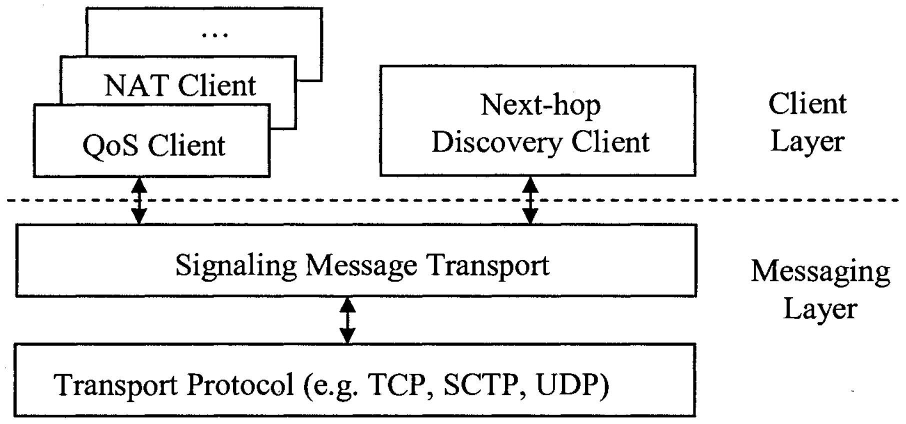

...
NAT Client
QoS Client
Next-hop Discovery Client
Client Layer
Signaling Message Transport
Messaging Layer
Transport Protocol (e.g. TCP, SCTP, UDP)

Figure III.9: CASP signaling architecture

The INSIS framework re-uses many CASP concepts. It is modular and flexible and it supports different applications. Furthermore, it enables signaling across different network environments. It can be used in different parts of the Internet (e.g. at the edge, in the core) and it supports mobility by allowing efficient service re-establishment after handover [55]. Its modular architecture enables lightweight implementations and framework extensibility. This feature allows it to work over different kinds of networks for various types of applications. Examples of NSIS-based signaling protocols are the extended RSVP QoS signaling protocol [56] and the middlebox configuration protocol [57].

The framework architecture is composed of two layers (Figure III.10): the NSIS Transport Layer (NTLP) that provides the application independent signaling functionalities (e.g. message transport), and the NSIS Signaling Layer (NSLP) that consists of a set of signaling applications providing application specific functionalities (e.g. resource reservation).

General Internet Signaling Transport (GIST) provides a concrete solution for the NTLP [58]. Its architecture is composed of a common messaging layer, running over a set of existing transport and security protocols (e.g. UDP, TCP, TLS). It has two modes of operation: Datagram mode (D-mode) and Connection mode (C-mode). The transport protocols used by each mode are UDP and TCP, respectively.

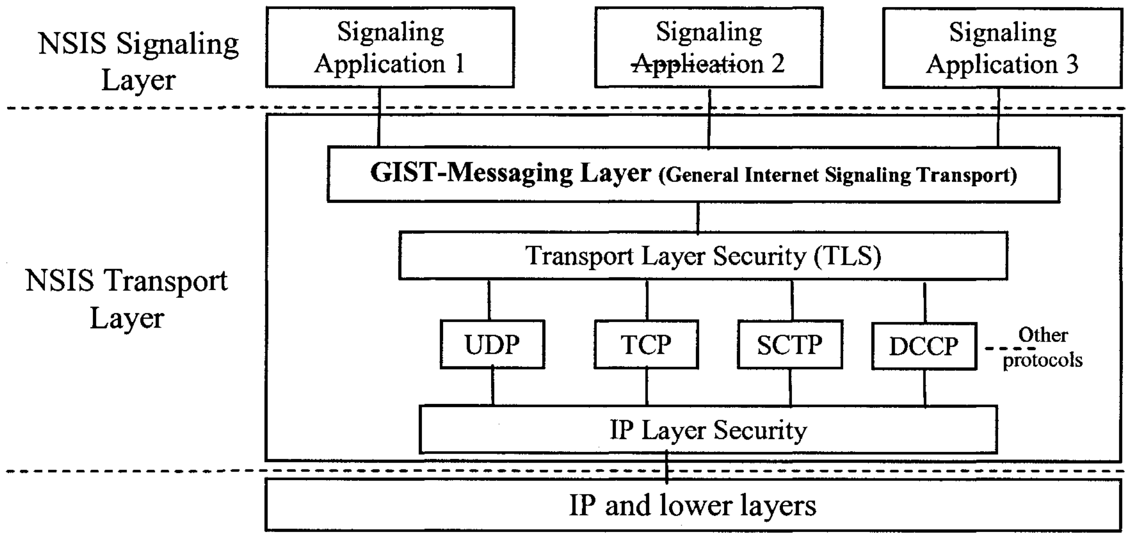


Figure III.10: NSIS protocol stack

GANS is a backward compatible generalization of NSIS. Its main extensions are the support of symbolic names, session mobility, and flow independent signaling applications [59]. Signaling applications can address destinations using symbolic names, which are translated by GANS' transport layer into corresponding IP addresses. A mechanism is provided to allow dynamic update of the IP-Symbolic name binding.

CASP and NSIS do not support both flow-dependent and flow-independent signaling applications, because they only define flow dependent signaling. They do not support symbolic names, and along with GANS, they support only one-to-one communication. Therefore, none of CASP, NSIS and GANS is independent of the negotiation model.

Table III-4 gives a summary of the review of signaling related work. All the requirements related to composition (i.e. R1, R2, R3, R4, R17 and R20) are not relevant for the specialized solutions, because their main objective is not to exchange the negotiation agreements and proposals. The general frameworks meet the majority of our requirements, but they do not provide any signaling application that can be used for registry composition (i.e. none of the designed signaling applications deal with the registry composition specificities).

Requirements	Signalization for specific applications			General frameworks		
	RSVP	**SIP**	**H.323**	**CASP**	**NSIS**	**GANS**
R1, R2, R3, R4	-	-	-	YES	YES	YES
R5	NO	YES	NO	YES	YES	YES
R17	-	-	-	YES	YES	YES
R18	NO	NO	NO	NO	NO	YES
R19	NO	NO	NO	NO	NO	NO
R20	-	-	-	YES	YES	YES
R21	NO	YES	NO	YES	YES	YES
R22	NO	NO	NO	YES	YES	YES
R23	NO	YES	NO	NO	NO	YES

Table III-4: Summary of the review of signaling related work

III.3 Conclusions

There is no existing overall architecture related to the overall architecture for registry composition. However there are existing architectural components that are related to the three components of the registry composition architecture (i.e. information publication and discovery, negotiation and signaling).

No existing solution meets all of our requirements for information publication and discovery after composition. DHT composition and decomposition solutions meet most of our requirement, but they are limited to a specific type of registries (i.e. DHT based registries) and they rely on gateways that should be created and configured offline. No solution exists for autonomous interface interworking. Only database systems deal with data interworking. Data composition in the context of FDBS resolves most of the problems related to data interworking, but FDBS are created offline. Therefore, we propose a new architecture for interface interworking and reuse the FDBS mechanisms and algorithms to propose a new data interworking architecture. The data interworking architecture provides a new procedure for solving autonomy.

For negotiation, the negotiation architectures that support a third party and do not rely on a permanently centralized entity are more appropriate for registry composition. However, the existing architectures do not meet any of the requirements related to registry composition. They are either designed to resolve a specific problem or general enough to be used in different circumstances. The specific architectures do not tackle the registry composition problem. The general solutions can be extended to address registry composition specificities, but they are too cumbersome. Furthermore, no existing architecture allows a third party to create and validate agreements. Therefore, we need to design a framework that is specific to registry composition, allows a third party to create and validate agreements, and do not rely on a permanently centralized entity.

For signaling, NSIS provides a promising signaling framework for registry composition. It is modular, easily extensible and has already been successfully used as the basis for signaling in different areas (e.g. QoS, signaling through mailboxes such as firewalls). GANS extends the NSIS messaging layer by resolving three more issues related to registry composition: use of symbolic names and support of session mobility and flow independent signaling applications. Thus, we use NSIS as basis for the design of our signaling framework, we reuse the GANS extensions, and we add support for group management (to allow point-to-multipoint message delivery) and a new signaling application to deal with the registry composition specificities.

CHAPTER IV: General architecture for registry composition

To orchestrate the registry composition process, we propose a new functional entity called the Registry Composition Entity (RCE). RCE is a sub-functional entity of the network composition-FE (C-FE), the functional entity that orchestrates network composition. This chapter starts by presenting RCE architectural components and the overall composition procedure. After that, it discusses the potential approaches to registry composition and to intercommunication between heterogeneous registries. This is followed by two illustrative scenarios. The conclusion is presented after that.

IV.1 Architectural components and overall composition procedure

This section starts by presenting the architectural components of the RCE and the role of each one of them. After that, it discusses the overall composition procedure.

IV.1.1 Architectural components

RCE is made up of three components (Figure IV.1): Composition Agreement negotiator (CA-negotiator), composition manager and co-ordination component. The CA-negotiator creates the composition agreement, after negotiating with the RCEs of each of the composing networks. An example of issues to negotiate is which protocol to use to enable intercommunication between heterogeneous registries. The interworking protocol agreed on may be supported by none of the composing registries. In a such case, the negotiating RCEs should also negotiate where and how to get the protocol (e.g. from a protocol server or to be created on-the-fly) and where it can be installed (e.g. based on resource availability and security policies). Examples of parameters used for CA negotiation are the protocol stack used -by each composing registry- for publication and discovery, the type of the discovery approach used (e.g. centralized, peer-to-peer) and the information publication and discovery interface (IPDI) used.

The composition manager is responsible for executing the composition agreement. This includes the configuration of the relevant network nodes and the execution of the necessary tasks, in order to reflect the composition agreement. An example is to install the negotiated intercommunication protocol, on the nodes agreed on. The co-ordination module enables intercommunication between different RCEs.

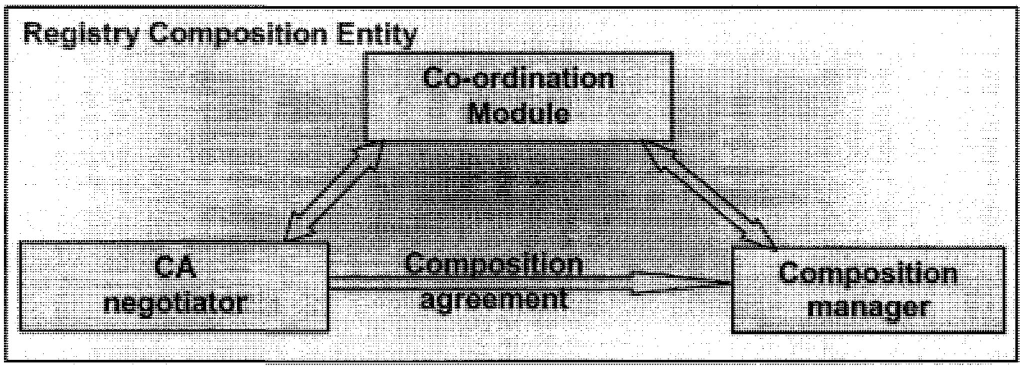


Figure IV.1: RCE architectural components

IV.1.2 Overall procedure for registry composition

Registry composition is executed as part of the network composition procedure, and is done during the last two phases of network composition (i.e. composition agreement negotiation and composition agreement execution). The registry-related procedures for the negotiation and the execution of the composition agreement are executed as sub-procedures of the network-related procedures that have the same name. Indeed, negotiation of network composition can be seen as a set of sub-procedures for composition negotiation between peer functional entities in the composing networks (e.g. the peer QoS FEs negotiate the QoS related parameters) and a global procedure executed by the C-FE, which monitors and orchestrates these different sub-procedures and creates the global agreement. Each FE can also be responsible for executing its related part of the global composition agreement.

Composition of the registries is initiated by the C-FEs of the composing networks. Indeed, when the network composition reaches the stage where the registries must compose (i.e. network composition negotiation phase), each C-FE informs the RCE of its own network. Then, the different RCEs communicate to negotiate the composition agreement.

IV.2 Potential approaches to composition

This sub-section identifies and analyses potential approaches to the creation of the post-composition registries, and to intercommunication between heterogeneous post-composition registries (i.e. registries that are using heterogeneous information publication and discovery interfaces). It starts by reviewing potential approaches related to the creation of post-composition registries. Then, it discusses the post-composition intercommunication approaches.

IV.2.1 Potential approaches to the creation of post-composition registries

As we have seen in the background chapter, in network interworking (i.e. the first degree of network composition), the original networks keep control over their individual resources (including registries) after composition. In control sharing, a new network is created and some of the resources of the original networks become part of the shared space. In network integration, all of the original resources belong and are controlled by the new network.

Three approaches are therefore possible for the creation of the post-composition registries: keeping each of the pre-composition registries, creating a new registry to store the shared resources and keeping the original pre-composition registries for resources that are under the control of the individual composing networks, and using a single new registry for the composed network.

In the first approach, existing pre-composition registries are kept as they are and updated by deleting the resources that are no more available after composition. The new resources that may be created after composition are added to any of the pre-composition registries. The pre-composition registries become therefore the post-composition registries, in this case.

In the second approach, pre-composition registries are also kept as they are, and a new registry is created to store the shared resources. Newly created resources are either added to the shared registry (if it is a shared resource) or to any of the pre-composition registries (if not). In the third approach, a new registry is created or one of the existing ones is selected, and the whole content of the pre-composition registries is copied to this registry.

It is clear that the first approach is the best choice for network interworking, because the composing networks remain separate and keep separate control over their registries. In case of network integration, the registry of the composed network can be seen as the collection of all of the individual registries that belong to that network. Therefore, the first approach can also support network integration by configuring the pre-composition registries to appear to the clients as a single registry. This means that when a client sends a discovery request to one of the pre-composition registries, the requested information should be searched for in all of the pre-composition registries. To support control sharing, the shared resources can be designated using policies (e.g. if Net1 resources are shared with Net2 users, new policies are added –to Net1 registry for instance- to reflect that).

The second approach can support network interworking and network integration in a similar way as the first approach. In case of control sharing, a new registry needs to be created on-the-fly and the shared resources should be copied from the pre-composition registries to the newly created one. Furthermore the components of the created network should be configured to access the new registry.

The third approach can be efficient if the network composition is permanent and the content of the composing registries is similar and relatively small. If the content is large or heterogeneous, the processing overhead (e.g. to copy the whole content) -- in term of time delay, network overhead and resources used-- can be too significant, and probably inacceptable. However, even in case of permanent composition, the approach remains too cumbersome for network interworking and control sharing, and generally not needed. Indeed, in the two cases, each composing network still benefit from some autonomy and independence from the other networks. Therefore, some resources still need to be separated (e.g. resources that are under the control of each network).

Table IV-1 presents a summary of the analysis of the three approaches, according to the degree of network composition and according to their support for registry decomposition. The first approach seems to be the best choice for temporary composition. However, it can compete with the third approach in case of permanent composition. Indeed, in permanent composition, the third approach can be costly --in term of processing overhead--, but it is efficient in responding to requests (all the content is in the same place). The first approach is easy to implement and it provides a good solution for load balancing among the post-composition registries, but it can be less efficient than the third approach when it comes to answering queries.

In this thesis, we use the first approach, because it supports all three degrees of network composition and supports network decomposition. Furthermore, it is the best solution for network interworking that is the most common degree of network composition in real life.

	Network interworking	**Control sharing**	**Network integration**	**Registry decomposition**	**Comments**
Approach 1	Supported easily	Supported easily	Supported easily	Supported easily	*Seems to be the best choice.*
Approach 2	Supported easily	A new registry should be created on-the-fly	Supported easily	Supported easily, except in case of decomposition after control sharing composition.	
Approach 3	Too cumbersome	Too cumbersome	Supported with a certain overhead (all the content should be copied to a single registry)	Too costly: the composed content should be distributed again.	Can be efficient in case of permanent network integration

Table IV-1: Analysis of the approaches to the creation of post-composition registries.

IV.2.2 Potential approaches to registry intercommunication

Considering that after composition, the composed network may host several registries; these registries must communicate in order to respond to clients' requests. Intercommunication between heterogeneous registries can be provided using two approaches. The first is to create gateway(s) between the concerned registries on-the-fly. The other option is to deploy a common protocol on-the-fly to these registries. This can be either a standard protocol specified off-line, or one of the protocols supported by the registries, chosen during the negotiation. The usage of a standard protocol will limit the number of protocols to deploy, and hence provide more scalability.

The first approach requires the ability to create the necessary interworking protocols on-the-fly, or their existence in the network before composition. The second approach requires reconfiguring the registries to use the old protocol to maintain communication with the clients that use that protocol (clients must not be changed), and use the newly installed protocol to communicate with the other registries.

The two approaches require on-the-fly deployment of protocols (e.g. deploy the common protocol to registries or interworking protocols to gateways).

Table IV-2 presents a summary of the analysis of the two approaches, according to the number of protocols to deal with, the simplicity, the network storage and processing overhead, and the time needed for the solution to take place. A significant difference between the two approaches is that the gateway approach is less scalable, because a different interworking protocol is needed for each two different protocols. On the other hand, the protocol deployment requires more configurations (to use the two protocols and to translate from one protocol to another). An interesting approach can be an approach that somehow combines the two approaches. An example is to use a standard protocol for intercommunication between registries, a common configuration procedure, and varies the translation solution depending on the protocol used by each registry (to translate between the standard protocol and the protocol used by the registry).

In this thesis, we use this last combined approach. This will lower the number of protocols to deal with and the network storage, and facilitate the nodes configuration.

	Number of protocols to deal with	**Network storage and processing overhead**	**Time needed**	**Simplicity**
Protocol deployment	1: the same protocol is deployed each time (i.e. the standard protocol)	Only one protocol should be stored	- Time to deploy the protocol and configure the node	The node should be configured to support the two protocols, and mapping procedures between the two protocols should be added
Gateway	Many: for each two different protocols, we have a different interworking protocol)	Different protocols should be stored or created on-the-fly	- Time to deploy the protocol and configure the node + - Create the interworking protocol (if it does not exist)	Mapping between protocols is already part of the interworking protocol

Table IV-2: Analysis of the intercommunication approaches

IV.3 Illustrative scenarios

This section presents two scenarios, illustrating both the approaches to the creation of the post-composition registries and the approaches to intercommunication between heterogeneous registries. In these two scenarios, the pre-composition registries are kept as they are. The first scenario uses the protocol deployment approach for registry interworking. The second one uses the gateway approach.

IV.3.1 First scenario

John is a very busy businessman, who decides to take a vacation. He is visiting Paris for the first time on a guided tour in a bus. He also wants to keep up-to-date on the status of his business. So, he is often connected to the Internet using his laptop. Today, he received an important document that he has to review as soon as possible. To do this, he needs to get the document printed. Unfortunately, the moving network available in the bus does not provide such a service. However, his preferences and requirements for printing quality and format are added to his profile and stored in the moving network. During one of the bus stops, a wireless static network with a printer that provides a service that fulfils John's printing requirements (Figure IV.2) is available.

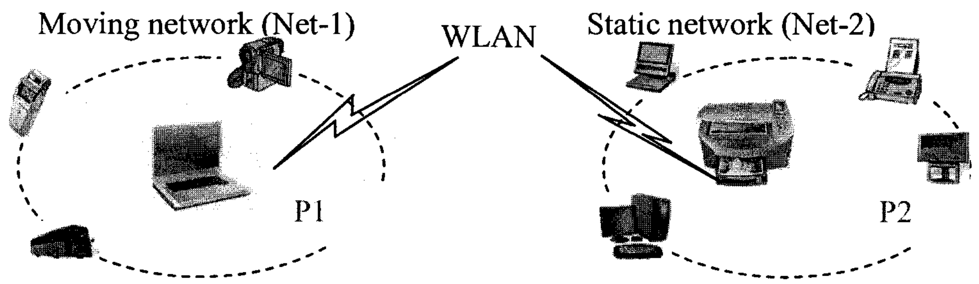


Figure IV.2: Registries' composition scenario

The two networks use distributed registries for storing information about the services provided and they use different discovery protocols; let us say P1 and P2. The registries' composition is activated by the C-FEs when the networks' composition is automatically initiated, once the two networks get close enough. Using the interchanged network characteristics and some predefined policies, the RCEs of the two networks realize that in order to enable the inter-network service discovery, P2 must be deployed in the Net-1 registry.

Furthermore, this deployment is deemed to be possible through some verification carried out by the RCEs. Next, the composition agreement is executed and the Net-1 registry prepares to communicate with the registry of Net-2.

According to their policies, the discovery and use of Net-2 services by Net-1 does not violate the discovery policies of Net-2 (nor those of Net-1). So, Net-1 automatically discovers the printing service, creates a connection between John's laptop and the printer using the WLAN interface, the document is formatted using John's preferences and added to the printer spool. John is informed that his document will be ready in 2 minutes and that he can pick it up before his bus leaves. He is also provided with detailed instructions so that he will find the printer.

IV.3.2 Second scenario

In this scenario, a static network (Net-1) hosts a conferencing application that creates a conference between a given numbers of users, who are in their respective offices (Figure IV.3). Each user's location is calculated and stored in a relational database – R1-- using the format "The user U1 is in room R1". Alice and Bob are visiting Net-1, where they have temporary offices. When they move, their coordinates (x,y) are stored in registries R2 and R3 of their respective Personal Area Networks. R2 is an object-oriented database, and R3 is a distributed registry that uses the Pervasive Discovery Protocol (PDP [60]) for information publication and discovery. Each of the three networks uses a different technology for user localization.

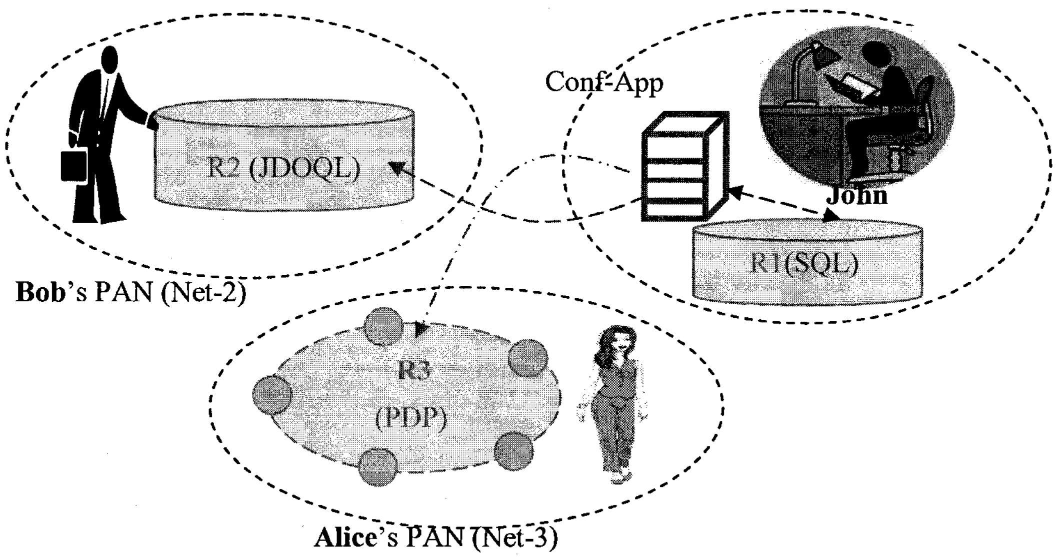


Figure IV.3: A composition scenario

We assume that John is already in his office and that Bob and Alice are still on their way. When they arrive at their offices, the conferencing application should create a conference between the three users. However, this will not happen because the application is unaware of Bob and Alice's location. This is because the interface (i.e. SQL) and the data granularity supported by the conference application are different from that provided by Net-2 and Net-3. Furthermore, the localization technologies used are different, which prevents Net-1 from directly getting Bob and Alice's location. Therefore, to enable the application to get the needed information, the three registries have to compose.

After the RCEs of the composing networks negotiate a composition agreement, they agree that the composing registries should be kept as they are and that gateways should be created in order to enable registries' intercommunication. They also agree that the protocol to be used by the gateways is PDP. We assume that R1 and R2 also support

PDP. Therefore, each RCE configures the registry of its network as a gateway between the local clients (i.e. the clients inside the network) and the other two registries.

After the gateways are configured, when the conferencing application issues a request to get the current locations of John, Bob and Alice, the request will get to R1. R1 will respond about John's location, and asks the other two registries about Bob and Alice's locations. It can for instance start by asking R2. R2 will give Bob's location. Since the location of Alice is not yet resolved, R1 asks R3.

IV.4 Conclusions

In this chapter, we have proposed a general architecture for registry composition. We have proposed a new functional entity (i.e. RCE) and a new procedure to orchestrate the composition. The RCE is made up of: CA-negotiator that negotiates with the other RCEs and creates the composition agreement, the composition manager that executes the composition agreement and the co-ordination component that enables intercommunication between different RCEs. Registry composition is initiated by the C-FEs of the composing networks, and is executed as part of the network composition procedure.

We have also identified and analyzed the potential approaches to the creation of post-composition registries and to intercommunication between heterogeneous post-composition registries. For the creation of post-composition registries, we selected to keep the pre-composition registries as they are. This is because this approach provides an easy support for all three degrees of network composition and for network decomposition. For registry intercommunication, we selected to use a standard protocol to enable intercommunication, and vary the inter-protocol translation solution according to the protocol used by each registry. This approach provides more scalability and is less costly (in term of network storage) and less time consuming.

The proposed general architecture can also be used for registry decomposition, where the RCEs of the decomposing networks negotiate and execute a decomposition agreement. The architecture is independent of the type of composing networks, network composition, and composing registries.

CHAPTER V: Information Publication and Discovery after Composition

The architecture for information publication and discovery after composition deals with two issues: interface interworking and data interworking. In this thesis, we propose an architecture for interface interworking and another one for data interworking. The data interworking architecture is an extended version of the interface interworking architecture. This chapter presents the two architectures. It starts by interface interworking. Then, it describes how the proposed architecture is extended to support data interworking. The conclusion is presented after that.

V.1 Interface Interworking architecture

We based our architecture on Peer-to-Peer (P2P) overlay network mechanism. We selected P2P overlay networks because they enable scalability, full decentralization and self-organizing. It also allows information publication in a distributed manner, which suites most ANs (entities can leave and join at any time).

This section starts by background information on P2P overlay networks. Then, it presents the architectural principles of the proposed overlay network and an illustrative scenario.

This is followed by a description of the related procedures. The section concludes with a discussion of the overlay protocol and messages.

V.1.1 Background on P2P overlay networks

Peer-to-Peer networks are distributed networks in which all nodes are equivalent in functionality and perform similar tasks [61] [62]. The different peers are autonomous and operate without centralized organization or control. They are able to organize themselves into some network topology and are capable of preserving connectivity when nodes join or leave the network.

A peer-to-peer network usually consists of a large number of equal peer-nodes. Each node acts both as a client and as a server, towards the other nodes in the network. Every peer stores local content and makes some/all of it available to other peers. The nodes of the network are connected in order to share resources such as files, computing power and network bandwidth.

Overlay networks are networks that run on top of an existing infrastructure, and provide additional functionality [63] [64]. They create a virtual topology on top of an existing physical one. P2P overlay networks are P2P networks, where the connected peers construct a set of logical connections with their neighbors. The overlay network is not necessarily the same as the physical one.

P2P overlay networks can be structured or unstructured [6]. In structured overlay networks, each data item is assigned a key, and the peers in the network are organized into a graph that maps each data key to a peer. The mapping of data items to numeric keys is done using a hash function. Each data item is stored at a particular peer. To find where a given data is stored, the peers use a hash table. The hash table is a data structure that maps keys onto values that help locating the node that possesses the data item. This technique enables an efficient discovery of data items, using given keys. However, it does not support complex queries and it is necessary to store a copy or a pointer to each item at the peer responsible for the data item's key.

In unstructured overlay networks, data items are randomly distributed over the peers. To look for a certain content, the peers use different techniques such as flooding or random walks. Each peer visited evaluates the query locally on its own content. Unstructured overlay networks enables the usage of complex queries, but theirs search techniques are inefficient in some circumstances, because queries for content that are not widely replicated must be sent to a large fraction of peers.

V.1.2 Architectural principles and scenario

One type of node makes up the overlay network that we propose: the Virtual Registry (VR). A virtual registry communicates with the other virtual registries using the "Overlay Interface" and with post-composition registries using the "Registry Interface". The overlay network is called the Registry Overlay Network (RON).

V.1.2.1 Architectural principles

For each different Information Publication and Discovery Interface (IPDIs) (i.e. protocol or programmatic interface) used by a post-composition registry, we have one and only one corresponding node in the RON network, and that corresponding node supports this interface (Figure V.1). Each overlay node supports only one IPDI. Each client communicates with the pre-composition registry which, before composition, belonged to the same network as the client. Each post-composition registry communicates only with the virtual registry that supports the same IPDI.

A description is associated with each post-composition registry; it includes the type of the registry and the description of the information it contains. The main parts of this description are: the registry address, the registry type (e.g. UDDI), the type of information maintained by the registry (e.g. web services descriptions) and a brief description that presents the purpose of this information (e.g. printing, user location). Each post-composition registry maintains its own description.

The RON has a P2P overlay architecture, with a fully interconnected topology. It uses a P2P protocol for information discovery and publication. This protocol fulfills a set of requirements that are presented later in this section. The network architecture and the related principles are illustrated in the following scenario.

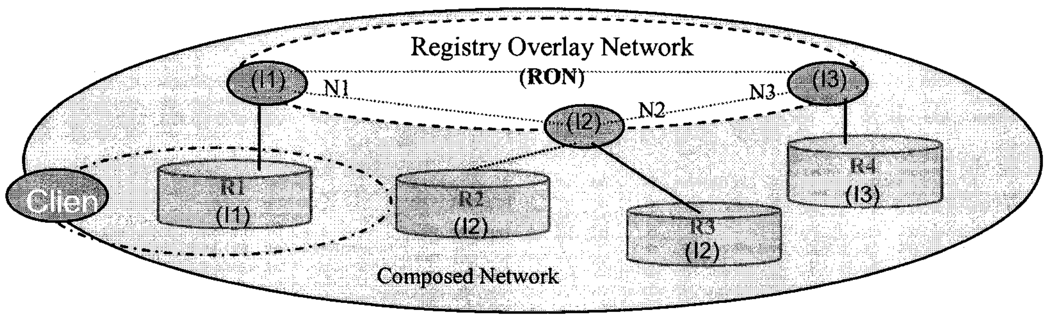


Figure V.1: General architecture

V.1.2.2 Scenario

John has a laptop in which a printing application is installed. To print documents, the application must know the address of a suitable printer. The information about printers (e.g. addresses, printing characteristics) is stored in a relational database (R1). Information about other resources in the network to which the laptop belongs (e.g.

scanners, faxes) is stored in an object-oriented database (R2) (Figure V.2). When a printing is requested, the application retrieves the printer address from the database, connects to the printer and then prints the document. The printer to use is chosen according to the printing characteristics it provides (and its availability).

When John is in motion, the usual printer becomes out of reach. Meanwhile, John approaches another moving network (Net-2) that hosts a printer (P2) with the same characteristics required by the printing application. P2's information is stored in a distributed registry (R3). The two networks get close to a static network (Net-3) that hosts two registries: a UDDI registry (R4) and an object-oriented database registry (R5). When the three networks get close enough, the moving networks -- Net-1 made up of John's laptop and Net-2-- compose with Net-3. The RCEs of the three networks compose the five registries and create the RON. The RON will be made up of four nodes: N1 uses SQL as IPDI, N2 uses Java Data Object Query Language (JDOQL), N3 uses Pervasive Discovery Protocol (PDP) and N4 uses UDDI APIs. PDP is a fully distributed protocol for services discovery in ad-hoc networks [60]. JDOQL is an implementation of the Object Query Language, a standard query language for object-oriented databases [65].

After the RON is created, if John orders a document to be printed, the registry overlay network is used and the printing application automatically gets the address of P2 and prints the document.

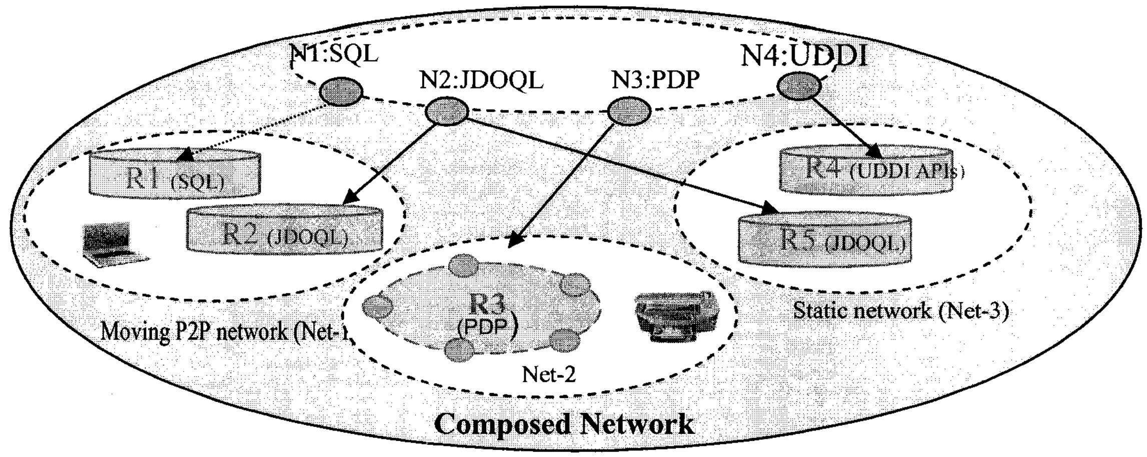


Figure V.2: Illustrative scenario

V.1.3 Procedures

This section presents the procedures related to the creation and the churn of the overlay network, and procedures related to information publication and discovery after RON creation.

Creation of the overlay network: The RON is created during the registries' composition process. In this thesis, we assume that the RCE that orchestrates the composition of the registries also orchestrates the creation of the overlay network. We further assume that it has the following information: the types, the addresses and the IPDIs used by each post-composition registry (e.g. it gets this information during the first steps of the negotiation). In the case of a P2P registry, the address of the registry is in fact the address of the super-node of the P2P network representing the registry. The super-node concept is used in order to take advantage of the heterogeneous character of P2P systems, improving the systems' performance. A super-node is generally chosen according to its capabilities (e.g. bandwidth, processing power) to play a special role and/or to serve other nodes [66]. If the network representing the registry does not use the super-node concept, one of the existing solutions for electing a super-node can be used (e.g. [67]).

To create the overlay network, the RCE starts by building the multicast groups, based on the list of IPDIs used by the post-composition registries. For each different IPDI, a multicast group is created, which includes all the post-composition registries that support that IPDI. Next, the RCE specifies a virtual registry for each group, and then it chooses, for each overlay registry VRi, the real node (i.e. post-composition registry) that will support the functionality of a particular VRi. Each VRi is mapped to a post-composition registry that supports the same IPDI. If more than one registry supports the same IPDI, the registry to which VRi is mapped is chosen randomly. With P2P registries, mapping is done in the same way, except that the virtual registry is mapped to the super-node of the chosen post-composition registry. At the end, the RCE activates the chosen nodes to act as virtual registries. Each VRi retrieves the descriptions from each of the post-composition registries that are part of its related multicast group, and publishes them to the overlay network, using the overlay protocol.

Information publication and discovery: When a client wants to publish new information, it sends a publication request to the same pre-composition registry that it was in communication with before composition. This will result in the publication of the new information into that registry.

To discover some information, the client sends a discovery request to the same registry. If this registry has the requested information, it sends it to the client. If not, it redirects the request to the virtual registry, which discovers the target post-composition registry that contains the information the client is looking for. It then retrieves the requested information from that registry and responds to the discovery request. If the registry that receives the discovery request from the client is P2P, the request is redirected to the virtual registry through the super-node of the former registry. The discovery of the target registry is based on the registry description.

RON churn: After the creation of the RON, a new registry may want to join the composed network (e.g. the composed network wants to compose with a new network that hosts a registry, or a new registry is added to the composed network). A registry may also need to leave (e.g. due to network decomposition). Two types of departure are possible: voluntary departure, where the departing node decides to leave the network, and forced departure, where a node is forced to disconnect from the network (e.g. node failure). In this thesis, we consider both voluntary departure and forced departure. This section presents the joining and departure procedures.

Join: Figure V.3 presents the procedure for joining the network, after the RON is created. MGi is the multicast group represented by VRi.

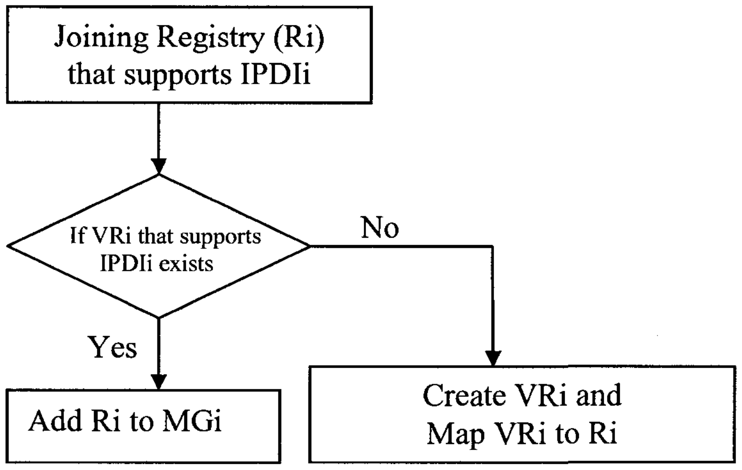

Joining Registry (Ri) that supports IPDIi
If VRi that supports IPDIi exists
No
Yes
Add Ri to MGi
Create VRi and Map VRi to Ri

Figure V.3: Registry joining procedure

Voluntary departure: given a post-composition registry Ri, the multicast group MGi to which Ri belongs, and the virtual registry VRi representing MGi, Figure V.4.a presents the procedure for a registry quitting the network (a P2P registry quits the network when its last element quits the network). Figure V.4.b represents the procedure when the Ri is P2P, the super-node (Si) of Ri wants to quit the network and Si is not the last element of Ri.

(**). VRi replaces the super-node:

VRi is automatically aware of the new super-node of Ri. Indeed, VRi uses the same P2P protocol (IPDI) as Ri. Therefore, VRi is part of Ri. Furthermore, in P2P networks, whenever a new super-node is created, all the nodes of the network are informed.

(***). RCE replaces the super-node:

If MGi is null, the VRi also sends the address of an arbitrary node Nj of Rj to the RCE, along with the quit message. The RCE activates Nj as a new temporary VRi. When the new super-node –Sj- of Ri is elected, Nj informs the RCE, which deactivates Nj and activates Sj as the new VRi.

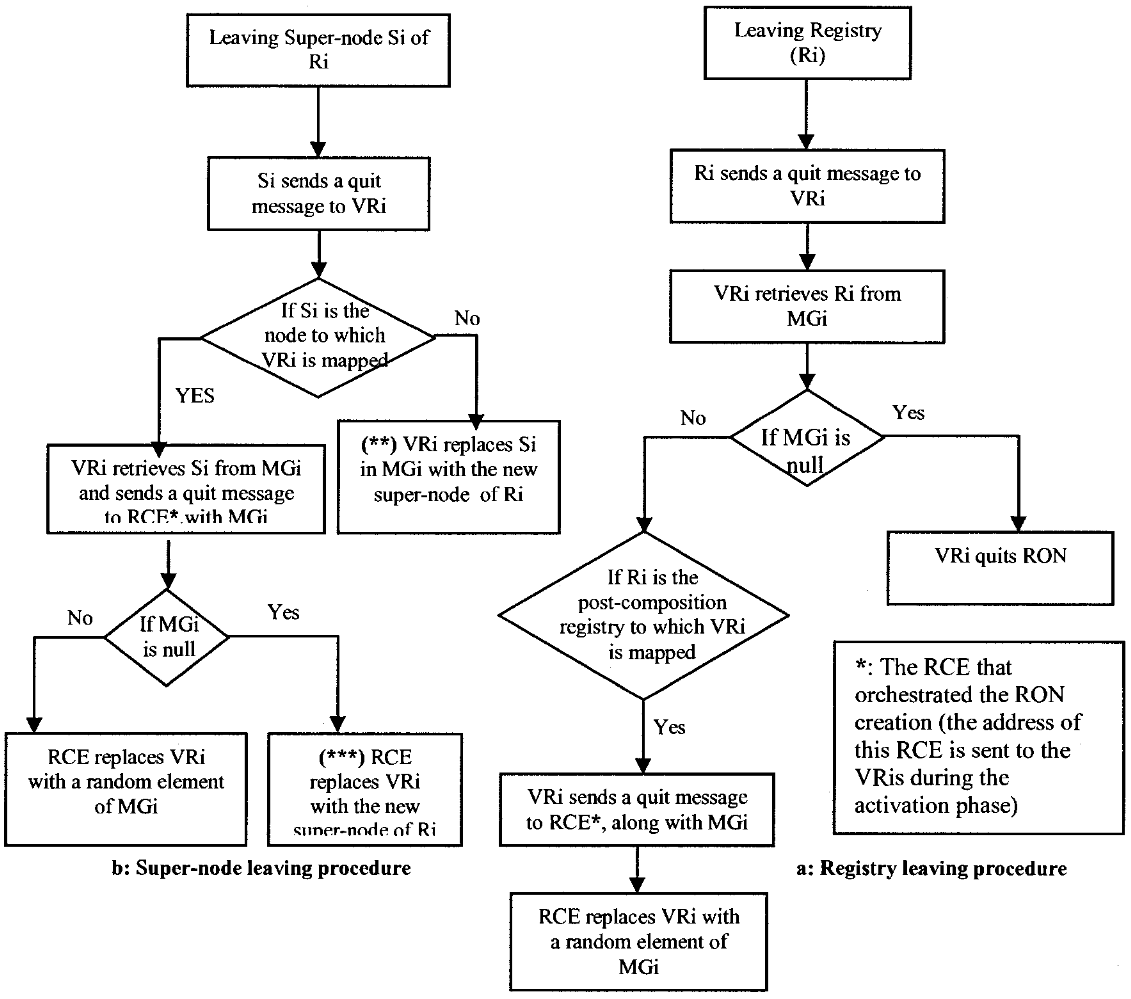

Leaving Super-node Si of Ri
Si sends a quit message to VRi
If Si is the node to which VRi is mapped
No
YES
VRi retrieves Si from MGi and sends a quit message to RCE* with MGi
(**) VRi replaces Si in MGi with the new super-node of Ri
If MGi is null
No
Yes
RCE replaces VRi with a random element of MGi
(***) RCE replaces VRi with the new super-node of Ri
b: Super-node leaving procedure
Leaving Registry (Ri)
Ri sends a quit message to VRi
VRi retrieves Ri from MGi
If MGi is null
No
Yes
VRi quits RON
If Ri is the post-composition registry to which VRi is mapped
Yes
VRi sends a quit message to RCE*, along with MGi
RCE replaces VRi with a random element of MGi
*: The RCE that orchestrated the RON creation (the address of this RCE is sent to the VRis during the activation phase)
a: Registry leaving procedure

Figure V.4: Voluntary departure procedure

Forced departure: To detect the eventual forced departure of the registries, we used the heartbeat scheme proposed in reference [68]. The authors in [68] propose a session recovery mechanism, for cluster-based signaling architecture for conferencing in MANET. The mechanism is based on the concept of heartbeat, and the session is defined as a signaling link between two entities participating in the conference. In the signaling architecture for conferencing, the conference participants are organized in different clusters. Each cluster has a super-member that maintains information about its cluster members and the other super-member. This information should be updated according to members' departure. The heartbeat scheme proposed to detect forced departure uses a request/reply protocol. It defines two entities: sender and responder. The sender is the entity that sends the heartbeat request. The responder is the entity that receives the heartbeat request and responds to it. Each session in the conference maintains a heartbeat. Heartbeat is defined as a periodic exchange of a request and a reply. If the session is created between a super-member and a member, the super-member becomes the sender and the member becomes the receiver. If the session is created between two super-members, one of them (e.g. the one with more capabilities) becomes the sender and the other one becomes the receiver. The sender periodically sends a heartbeat request to responder and starts a timer. If the timer fires and no reply is received from the responder, the sender re-sends the request and restarts the timer. If there is no reply upon a number of requests, the sender considers that the responder has unintentionally departed.

The departing registry (Ri) can be either a virtual registry or a normal registry (i.e. no VR is mapped to Ri). We consider the two cases.

(a). Forced departure of a normal registry: Each VRi is responsible for keeping track of the normal registries belonging to its MGi. To detect the forced departure of normal registries, we equate the super-member in the heartbeat scheme presented before to the VR, and the members to the normal registries. The session is a link between a VR and a normal registry.

If a normal registry disappears, its VRi detects its forced departure using the heartbeat scheme, and removes it from the MGi.

(b) Forced departure of a virtual registry: Forced departure of virtual registries is detected by the RCE that orchestrated the creation of the RON. In this case, we equate the super-member to the RCE, and the members to the VRis. The session is a link between the RCE and a VRi.

The RCE maintains a heartbeat session with each virtual registry. If a VRi leaves the network, the RCE will detect its forced departure and replace it with a random element of MGi. To keep the RCE up-to-date on the MGi elements, each VRi sends an update message to the RCE each time a normal registry quits the network (via voluntary departure or forced departure).

V.1.4 Overlay protocol and messages

The overlay protocol should fulfill a set of requirements that are refinements of the requirements for the global information publication and discovery solution as presented in the chapter on related work. First, it should be suitable for P2P, and therefore distributed and not rely on a central entity. Furthermore, it should allow for self-reorganization – enabling nodes to join and leave "easily". Second, it should enable the publication of the registries' descriptions and the discovery of the registry that contains given information using the registries' descriptions. Third, it should use time-efficient mechanisms for publication and discovery. Fourth, it should be as simple as possible, to allow its usage with small devices that require a small footprint. It also should scale in terms of the number of nodes that make up the overlay network.

Many existing P2P protocols, such as Tapestry [69] and Chord [34], can be used as the overlay protocol of our network architecture.

Messages:

Table V-1 and Table V-2 below present the messages exchanged between two different virtual registries, and between a virtual registry and a post-composition registry, respectively.

Publish-Description	*Description*: Publishes a description to the overlay network. Sent by a virtual registry to the overlay network, after retrieving a description from a post-composition registry. **Address*: Broadcast. *Parameters*: The description to publish. * Depending on the publication protocol used, it can also be Unicast or Multicast.
Find-Registry	*Description*: Finds the post-composition registry that stores a given type of information. Sent by a virtual registry to the overlay network when it receives a discovery request from a post-composition registry, or when a retrieval request is received from another virtual registry. *Address*: Unicast and Anycast *Parameters*: The description of the information to retrieve.
Retrieve-Information	*Description*: Retrieves information from a post-composition registry. Sent by a virtual registry (VR1) to a virtual registry (VR2), when VR1 receives a discovery request from a post-composition registry and discovers that the information to retrieve is stored in a registry that belongs to the VR2 group.

	Address: Unicast *Parameters*: The target registry from which to retrieve the information. The description of the information to retrieve.
Response	*Description*: Sends a response to a post-composition registry, via a virtual registry. Sent by a virtual registry (VR1) to a virtual registry (VR2) when VR1 receives a request from the post-composition registry via VR2. *Address*: Unicast *Parameters*: The target registry where the response is to be sent. The response.

Table V-1: Messages between virtual registries

Get-Description	*Description*: Gets the description of post-composition registries. Sent by a virtual registry (VR1) to the post-composition registries belonging to VR1 multicast group: • At the creation time of the overlay network • When a new registry joins *Address*: Multicast and Unicast
Retrieve-information	*Description*: Retrieves information from a post-composition registry. Sent by a post-composition registry (R1) to a virtual registry when R1 receives from a client a discovery request for information that it does not have. Sent by a virtual registry (VR1) to

	a post-composition registry (R1), when VR1: (a) receives the same message from a post-composition registry (VR2) and discovers that the requested information is stored by R1 (R1 and R2 belongs to the same multicast group), (b) receives a retrieval request bound to R1 from another virtual registry. *Address*: Unicast. *Parameters*: The target registry from which to retrieve the information. The description of the information to retrieve.
Response	*Description*: Sends a direct response to a post-composition registry. Sent by a post-composition registry (R1) to a virtual registry (VR1) when R1 receives a request from VR1. Sent by VR1 to R1, when VR1: (a) receives a request from R1, (b) receives a response bound to R1 from another virtual registry. *Address*: Unicast *Parameters*: The message target. The response

Table V-2: Messages between a virtual registry and a post-composition registry

V.2 Data Interworking architecture

We propose to extend the RON architecture to handle data interworking. This will require extending the internal behavior of the virtual registries, in order to take into account data heterogeneity. Indeed, when a virtual registry receives a discovery request, it has to know where the related information is stored and especially, how to get it. This may require –

for instance- translation between the concept names used by the client and those used by the target registry.

This section starts by describing the data interworking problem in more details. After that, it presents our new data interworking architecture.

V.2.1 Problem statement

We divide the data interworking problem into four sub-problems: content update, content mismatch resolution, content composition and content discovery after composition. Content update deals with the consistency of the registry content after network composition. Indeed, when ANs compose, some data may become obsolete or may need to be updated. For instance, take the case of a registry maintaining the list of printers in its network, where each printer is described by its name, the IP address and the port number to use in order to communicate with it. After network composition, the IP addresses may change (e.g. one of the networks is absorbed by another and thereby requires changes to its addresses range and network mask). Therefore, the first step in composing the registries' content in ANs is to dynamically update that content in order to make it consistent.

Content mismatch resolution deals with content heterogeneity in terms of naming mismatch (where different names are given to the same concept by different providers), representation and structure mismatch, semantic and syntax mismatch, and granularity mismatch (e.g. get the office where Alice is, using her coordinates).

Content composition oversees how the content from the different registries is composed. Two different approaches may be used to solve this sub-problem: content integration and content federation. In content integration, the content of the different registries is totally merged and stored in a single registry (e.g. the one with more capabilities such as processing power and bandwidth). In content federation, the different registries are kept as they are, and higher-layer processing logic is provided to seamlessly answer requests using the entire content. Content integration may be especially needed for network integration. The content to integrate can either be of the same type or be heterogeneous. If the content is heterogeneous (e.g. it has heterogeneous granularities), the integrated content to be stored in the single registry is obtained by executing the appropriate content mismatch resolution algorithms (e.g. aggregation). In content federation, clients are given a uniform and transparent access to the content, which is spread over the different registries. This is similar to federated database systems [70], where various autonomous database systems are perceived as a unique system by end clients.

It is clear that content integration may be too costly, in terms of processing time and power (e.g. to copy a huge amount of data from one registry to another). Furthermore, some information may be lost when the content to compose is processed, which may introduce an extra processing overhead when a request is received. If, for instance, all the content is brought to the higher level of aggregation, we loose the lower granularity information that may still be needed by some applications.

Therefore, we choose to use content federation for content composition in registries when ANs compose. This will speed up the composition process, facilitate the decomposition, if any, (i.e. due to the networks' decomposition) and enhance the content availability after composition (i.e. if one registry fails; only its content becomes inaccessible).

Content discovery after composition deals with how clients will access the composed content.

In this work, we reuse the FDBS mechanisms and algorithms to model contents and queries, to resolve content mismatch, and to create and execute query-plans. We provide new procedures for solving the autonomy and the content update issues.

V.2.2 A new architecture for data interworking

In order to answer clients requests, virtual registries execute some internal logic (e.g. information publication and discovery procedure). We will call the module responsible for executing this logic Overlay Application.

This section presents the functional components forming the overlay application and the data interworking related procedures. It also presents an illustrative example that shows how data interworking can be provided using the proposed architecture.

V.2.2.1 Architectural components

The architecture of the overlay application is presented in Figure V.5. It consists of the Startup Module (SM), the Query Manager (QM), the Data Composition Manager (DCM), the Query Execution Manager (QEM) and the Registry Interrogator (RI). The SM is responsible for node bootstrapping. The QM is the module that receives the incoming requests, before they are processed. The DCM is the module responsible for creating the appropriate query plans to answer the received queries, by communicating with the other DCMs. The algorithm used by the DCM should fulfill the following requirements:

- Distributed (i.e. does not require all registry descriptions to be stored in a single node).
- Scalable in terms of the number of registries to compose.

- Allows registries to leave and to be added without disturbing the algorithm's functioning and efficiency.

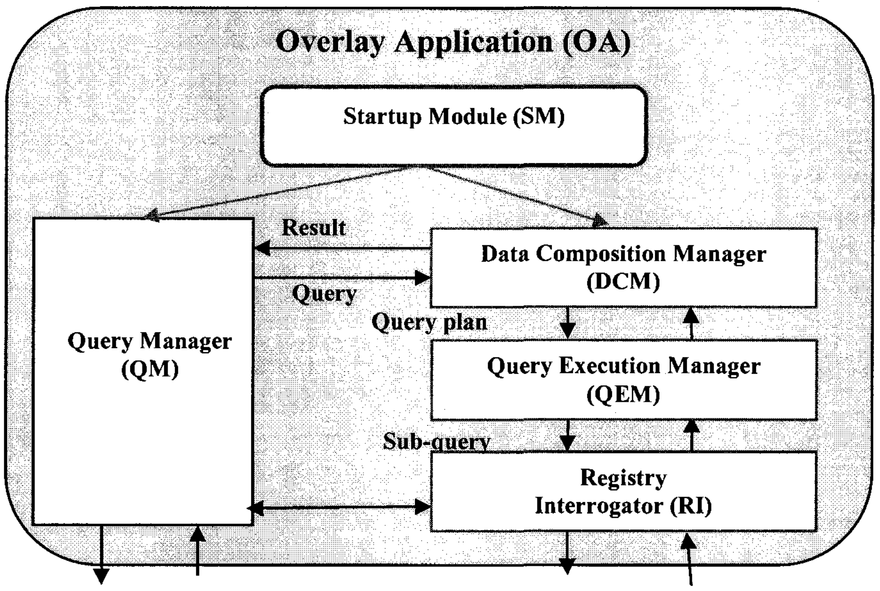


Figure V.5: Overlay application architecture

Some data composition algorithms from the database systems field almost meet these requirements ([71][72][73]). The QEM is responsible for executing the query plans generated by the DCM and creating the final answer to the received request. The RI is the module used to communicate with the registries targeted by the sub-queries. To communicate with a registry using a different IPDI than that used by the current overlay node, the RI starts by identifying the overlay node using the same IPDI as the target registry, and then it transfers the message to it.

Each overlay node maintains the data model of the registries that it represents (i.e. registries that support the same IPDI as the overlay node). A data model describes the data content and serves as the registry description used by the DCM to create query plans. The data models are intended for machine-to-machine communication between heterogeneous nodes. Therefore, we have chosen XML as the underlying representation language.

V.2.2.2 Procedures

This section presents a discussion of the content update and composition procedure and the procedure for content discovery after composition.

Content update and content composition procedure: When RON creation is initiated, each SM module starts up an instance of the QM, DCM and RI modules. Then, it executes the necessary content updates on all of the registries supporting the same interface as the overlay node to which the SM belongs. To communicate with the registries involved, the SM uses the RI module.

The different registries are kept as they are, except the consistency updates. To create the federated content, the data models of the different registries need to be integrated (e.g. the relationships between the different concepts in the different models are created). Therefore, after the consistency updates, the different DCMs communicate in order to create the integrated model. This model is reused to answer received queries, and is updated each time a registry quits or joins the composed network.

Content discovery procedure: After composition, each time a request is received by the overlay application, the procedure in Figure V.6 is executed. If the request source is another overlay node, this means that the current overlay node is the one using the same IPDI as the target registry. To communicate with a registry targeted by the sub-queries in the query plan, the QEM uses the RI.

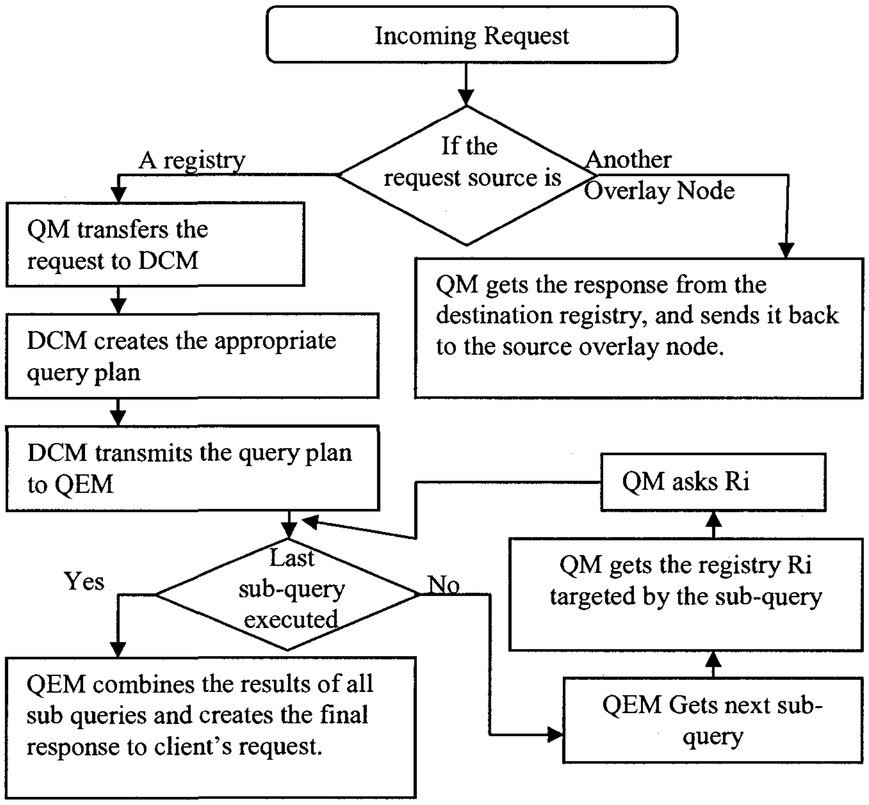

Incoming Request
If the request source is
A registry
Another Overlay Node
QM transfers the request to DCM
DCM creates the appropriate query plan
DCM transmits the query plan to QEM
QM gets the response from the destination registry, and sends it back to the source overlay node.
QM asks Ri
QM gets the registry Ri targeted by the sub-query
Last sub-query executed
Yes
No
QEM combines the results of all sub queries and creates the final response to client's request.
QEM Gets next sub-query

Figure V.6: Procedure for answering requests

V.2.2.3 Illustrative example

To illustrate the proposed architecture, we discuss how it can be used to answer the conferencing application requests in the second scenario presented in the previous chapter. In this scenario, a conferencing application needs to get the current location of John, Alice and Bob. The three users belong to three distinct networks (Net-1, Net-2 and Net-3 respectively). The location of each user is stored in the local registry of its network (R1, R2 and R3 respectively).

The three registries in the scenario are using three distinct IPDIs (i.e. SQL, JDOQL and PDP). Therefore, the registry overlay network will be composed of three nodes: N1, N2, and N3, representing the registries R1, R2, and R3, respectively.

We assume that the registry R1 is using the schema R1(#userID, locationRoom) for information storage and that R2 and R3 are using the schema R2(#userID, locationX, locationY). We also assume that we have a fourth registry R4 that is using the schema R4(#roomID ,minX, maxX, minY, maxY) to describe the different offices using their delimiting coordinates.

The conferencing application is configured to create a conference between John, Alice and Bob. Therefore, it will issue a request for the location of each of them. The request will go to R1, the only registry known to the application. R1 will then reply regarding John's location, but it will transfer the other two requests to N1 (R1 does not have the requested information), where the request will get to the overlay application OAN1. Within OAN1, the QM receives the request, determines that it is coming from a registry, and it sends it to the local DCM. The DCM creates the necessary query plans and asks the QEM to execute them. Figure V.7 presents the created query plan to get Bob's location. A similar plan is used for Alice.

Query: get locationRoom where userID = BobID.

Query plan:

- View1: get(locationX, locationY) from R2 where userID = BobID.
- View2: for each(x,y) in view1, get roomID from R4 where: minX<x<maxX and minY<y<MaxY.

Figure V.7: Query and query plan example.

V.3 Conclusions

In this chapter, we have proposed an overlay architecture for information publication and discovery after composition. The proposed overlay architecture deals with both interface interworking and data interworking. The registry of the composed network (i.e. the composed registry) is seen as the collection of the individual registries in the composed network. An overlay network is created on-the-fly, to allow autonomous access to the whole content of the composed registry. Clients can seamlessly access the composed registry through the pre-composition registries of their networks.

If we analyze the overlay architecture with respect to the information publication and discovery requirements, we can find that the architecture is very promising. Indeed, the architecture supports all three degrees of network composition and it is suitable for both P2P and centralized registries. Furthermore, clients in the composed network are able to discover and publish information after composition, they have seamless and automated access to the whole content, and the publishing and discovery policies of the composed registries are not violated (i.e. the policies of each registry are still enforced by the same registry after composition). The architecture also supports registry decomposition (i.e. through nodes departure procedures) and is transparent to the clients. It reuses FDBS mechanisms for data composition and P2P protocols for information publication and discovery.

In Chapter 8, we will further discuss the implementation and performance measurement of the overlay architecture.

CHAPTER VI: Negotiation architecture

As discussed in the chapter on general architecture, registry composition is based on a composition agreement that is negotiated between the different parties involved in the composition. This chapter presents an architecture for the dynamic negotiation of an agreement for the composition of registries. It starts by introducing the negotiation architecture and its components. Next, it presents the negotiation protocol for the case when no entity leaves the negotiation when it is started. The section after that describes how the negotiation protocol is extended in order to support entities departure during the negotiation. This is because entities in ANs can join and leave the network any time. The last section draws the conclusion.

VI.1 Negotiation architecture

The negotiation is triggered by the C-FE of one of the composing networks (e.g. the one that orchestrates network composition), and it is done among the RCEs of all of the composing networks. The triggering C-FE gives as well the IDs and addresses of the other RCEs to the triggered RCE (the triggering C-FE gets this information from the C-FEs of the other composing networks).

Our negotiation architecture is made up of negotiating entities, a proposals' template, a description of the main negotiation steps, and a negotiation protocol. This section starts by background information on negotiation. Then, it presents the general principles of our architecture. After that, it presents a template for the composition agreement proposals, and describes the main steps of the negotiation. The negotiation protocol is presented in the sections after that.

VI.1.1 Background on negotiation

During a negotiation process, the negotiating parties can either communicate directly or via a mediator (i.e. a third party that arbitrates the negotiation). They can negotiate a single issue (e.g. price) or multiple issues (e.g. price, quality). With multiple issues, they can present all their demands at once or present them one by one [74]. The first approach is called parallel negotiation and the second one is called sequential negotiation [31]. They can also negotiate a group of issues first (e.g. tightly coupled issues that can be solved together) and then move on to another group. We call this last approach the hybrid approach.

The main existing negotiation decision models are accept-it-or-leave-it., offer/answer, offer/counteroffer and contract ranking. In the first model, one of the negotiating parties makes an offer to the other(s) party(s), which has only the possibility to accept or reject the offer. In the second model, the receiving party can also give feedback on its decision, such as specifying why the offer is rejected. The third model allows the receiving party to make a counter-offer if the first offer is rejected. In the contract-ranking model, the offer-originating party creates a set of proposals that are sent to the interested party. The receiving party ranks the received offers according to its own criteria and chooses the most appropriate [75].

VI.1.2 General principles of our architecture

To meet our requirement on allowing a third party to orchestrate the negotiation and create proposals, we have identified two negotiation entities: participant and mediator. The participant is any entity that is participating in the negotiation. A participant can be either an initiator or a responder. The initiator is the entity that initiates the negotiation process. The responder is the entity that receives a CA-proposal and decides the acceptance or rejection. A participant can alternate between being an initiator or a responder, but it can play only one of the two roles at any given time. The mediator is the (third party) entity that orchestrates the negotiation process, and is responsible for creating CA-proposals and arbitrating the negotiation. It can be either co-located with the participant or be an independent entity.

The negotiating entities (i.e. RCEs) communicate via a mediator. They can negotiate multiple issues using the hybrid approach, where a group of issues are negotiated at the same time. This can optimize the general composition process. In case the set of negotiation parameters include mandatory and optional ones, we can for instance start by negotiating the mandatory ones. While negotiating the optional parameters, we can execute these parts of the agreement that are only related to the mandatory parameters (i.e. those independent of the optional parameters).

The negotiating parties use the offer/answer decision model. The offers are created by the mediator and sent to the negotiating parties, which have to decide the acceptance or rejection. Sending feedback to the mediator about the reason why the proposal is rejected can help the mediator in creating a more suitable proposal for the next round of negotiation.

VI.1.3 Proposals' template

The CA-proposals' template includes two main parts: a network specific part and a composition related part. The network specific part includes the information that is specific to each network and that is necessary for the composition negotiation. Such information can be the type of the local registry(ies) (e.g. centralized, distributed), and the local protocols used within the network and that influence the composition (e.g. the interface -protocol or API- used for information publication and discovery). This information is used by the mediator to make proposals.

The composition related part includes the information that is more related to the composition and the negotiation. Such information is the maximum negotiation lifetime, which is the time after which the negotiation is aborted if no agreement is reached. If an agreement is reached, the filled template includes the agreement validity time that specifies the time after which the agreement is no longer valid. It may also include, later on, the output of the negotiation (i.e. the Composition Agreement).

Figure VI.1 presents an example of a proposal template, used for the composition of the two networks presented in Figure VI.2. Each RCE fills the first part of the template (i.e. Network Related) with the local information. The template in Figure VI.1 is filled with the information related to net-1. The template can include as many <Registry> objects as the number of the registries in the network. The RCE also specifies its preference for the maxNegotiationLifeTime, if any. The filled template is sent to the mediator in the initiation phase: with the first negotiation request (in case of an initiator) or with the negotiation acceptance message (in case of other participants).

```
<NetworkRelated>
      <Network>
            <NetID>   Net_1  ID </NetID>
            < Registry>
                          <Type> Centralized </Type>
                          <IPDI> SQL </IPDIl>
                          <Version> 1.1 </Version>
            </ Registry>
            < Registry>
                   <Type> Centralized </Type>
                  <IPDI> JDOQL </IPDIl>
                  <Version> 1.2 </Version>
           </ Registry>
           <StartingConditions>  </StartingConditions>
      </Network>
</NetworkRelated>
<CompositionRelated>
         <maxNegotiationLifeTime> 50 </maxNegotiationLifeTime>
         <Agreement>
                  < ValidityTime> 300 </ ValidityTime >
                  <AgreementBody> the actual content of the reached
                          agreement or of the CA-proposal
                  </<AgreementBody>
         </Agreement>
</CompositionRelated>
```

Figure VI.1: Example of a proposal template

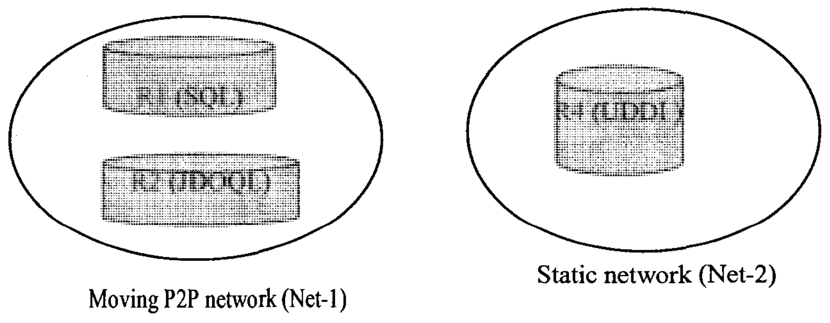


Figure VI.2: Example of two composing networks

At the end of the negotiation, if an agreement is reached, the mediator fills the <Agreement> object, before it sends the filled template to all of the participants. The mediator can also fill the <NetworkRelated> object with the list of the registries that accepted the negotiation.

VI.1.4 Main negotiation steps

To initiate a negotiation, the initiator starts by locating the mediator to use. We assume that it uses the co-located one, if any, or uses one of the existing solutions for discovery (e.g. PDP [60]) to find the address of an existing mediator. We also assume that the mediator has published its existence beforehand.

The main steps of the negotiation after the initiator locates a mediator are as follows (Figure VI.3):

Initiation of the negotiation: The initiating RCE creates a negotiation request that it sends to the mediator. The negotiation request includes the ID of the initiator, the IDs of the entities with which it wants to negotiate (i.e. destination RCEs that we will now call the destination participants) and the local parameters that can be used to create a proposal (e.g. local IDPI, local registry type). The initiator can also include specific requirements for the negotiation such as the IDs or the minimum number of participants that have to accept before the negotiation takes place. We will call these conditions "starting conditions". If the mediator accepts the request, it sends another negotiation request to the destination participants. If they accept, they send their network related information to the mediator. A mediator can reject to orchestrate a new negotiation because it does not –

for instance- have enough resource because it is already involved in other(s) negotiation(s).

Negotiation: If the mediator accepts to orchestrate the negotiation and the starting conditions are met, the mediator creates a first CA-proposal that it then sends to all of the participants. These can either accept or reject the proposal. If a participant rejects the proposal, it specifies the reason. If the proposal is rejected, the different participants begin to negotiate a new one. This is repeated until an agreement is reached, or the maximum negotiation lifetime expires.

Termination of the negotiation: At the end of the negotiation, if an agreement is reached, the mediator creates the final agreement. This agreement is sent to all of the participants. If the negotiation has stopped because of an error (e.g. time-out), the mediator sends an error message to the participants to inform them that the negotiation has failed, along with the error description.

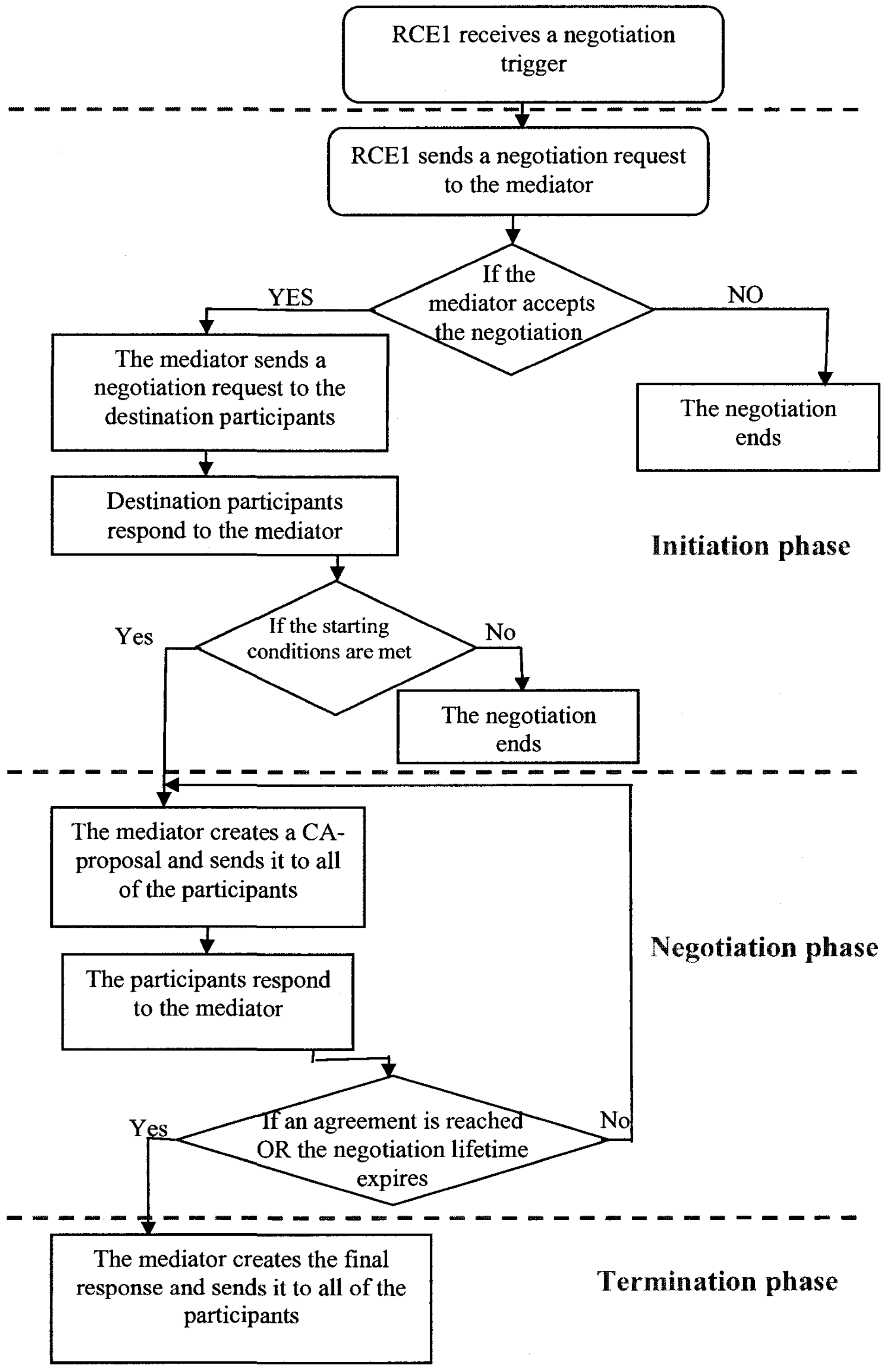

RCE1 receives a negotiation trigger
RCE1 sends a negotiation request to the mediator
If the mediator accepts the negotiation
YES
NO
The mediator sends a negotiation request to the destination participants
The negotiation ends
Destination participants respond to the mediator
Initiation phase
If the starting conditions are met
Yes
No
The negotiation ends
The mediator creates a CA-proposal and sends it to all of the participants
Negotiation phase
The participants respond to the mediator
If an agreement is reached OR the negotiation lifetime expires
Yes
No
The mediator creates the final response and sends it to all of the participants
Termination phase

Figure VI.3: Negotiation steps

VI.2 Negotiation protocol

The negotiation protocol entities are: participant and mediator (already described in section VI.1.2). This section describes the negotiation messages and the state diagrams of the protocol entities.

VI.2.1 Negotiation messages

To support the offer/answer negotiation-decision-model, we designed our negotiation protocol as a request-response protocol. Therefore, we have two messages: Request and Response. We have defined three types of requests: Initiate, Offer and Ack. Each request (except Ack) has a response message that is an Ok message. Each Ok message includes a response code that specifies the type of the response. We have defined four response types: Accept, Reject, Agreement and Error. A request can be point-to-point (e.g. Initiate from the Initiator to the Mediator) or point-to-multipoint (e.g. Offer). All the response types are point-to-point. Figure VI.4 illustrates the flow of messages in the case of a successful negotiation.

Table VI-1 below describes the negotiation messages and message codes.

Initiate	*Description*: Initiates a new negotiation. Sent by the initiator to the mediator. Sent by the mediator to the destination participants, when it receives an *Initiate* message and accepts to orchestrate the negotiation. *Address*: Unicast and Multicast. *Parameters*: ID of the initiator, IDs of the destination participants, local parameter (e.g. local IPDI) and the starting conditions.

Offer	*Description*: Sends a CA-proposal to the participants. Sent by the mediator to all of the participants. *Address*: Multicast. *Parameters*: The CA-proposal.
Ack	*Description*: Informs the participants about the outcome of the initiation phase of the negotiation (i.e. the negotiation will take place or not). Acknowledges the reception of the final response to the negotiation. Sent by the initiator to the mediator or bye the mediator to the destination participants (in the initiation phase). Sent by all of the participants to the mediator, in the termination phase. *Address*: Unicast and Multicast. *Parameters*: A response code (i.e. *Accept* or *Reject*) to specify if the negotiation is accepted or rejected (when sent by the mediator to the destination participants in the initiation phase).
Accept	*Description*: Accepts new negotiation or a CA-proposal. Sent by a destination participant to the mediator after it receives an *Initiate* request and it accepts to participate in the negotiation. Sent by the mediator to a participant to inform it that the negotiation has been accepted (after it accepts to orchestrate the negotiation and verifies that the conditions for the negotiation to take place are met). Sent by a participant to the mediator to accept the received CA-proposal. *Address*: Unicast and Multicast. *Parameters*: The list of the participants that accepted the negotiation.

Reject	*Description*: Reject a new negotiation or a CA-proposal. Sent by the mediator to the initiator or by a destination participant to the mediator after it receives an *Initiate* request and it rejects to participate in the negotiation (Figure VI.5). Sent by the mediator to all of the participants to inform them that the negotiation has been rejected (i.e. the conditions for the negotiation to take place are not met) (Figure VI.6). Sent by a participant to the mediator to reject the received CA-proposal. *Address*: Unicast and Multicast. *Parameters*: Reason of rejection
Agreement	*Description:* Sends the final agreement to the participants. Sent by the mediator to all of the participants. *Address:* Multicast *Parameters:* The reached agreement.
Error	*Description*: Sends an error message. Sent by a request receiver to the source of the request when the last received request has generated an error. Sent by the mediator to all of the participants if the negotiation ends because of an error (e.g. timeout). *Address*: Unicast and Multicast *Parameters*: Error description.

Table VI-1: Negotiation messages and message codes

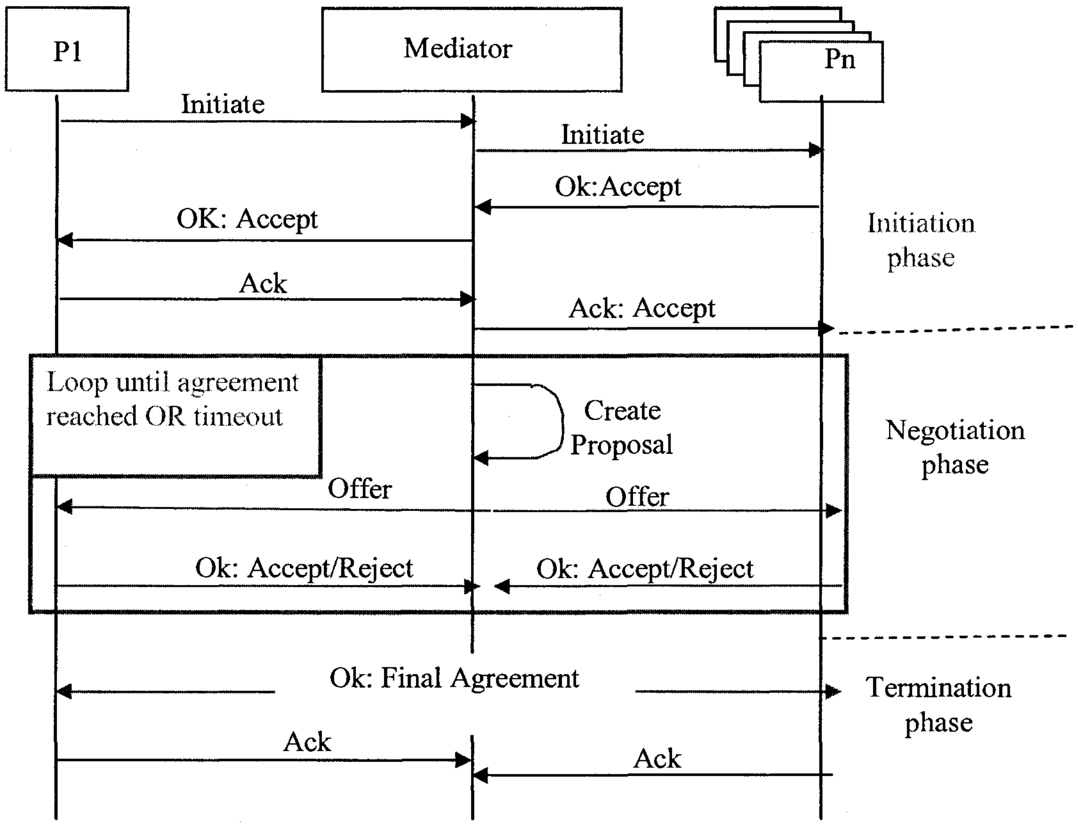


Figure VI.4: Sequence diagram for successful negotiation

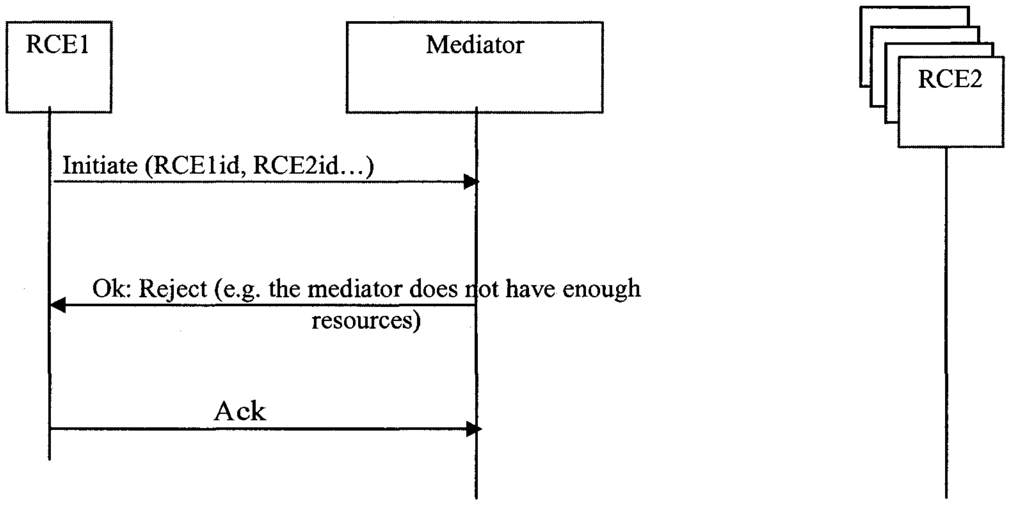

RCE1
Mediator
RCE2
Initiate (RCE1id, RCE2id…)
Ok: Reject (e.g. the mediator does not have enough resources)
Ack

Figure VI.5: Negotiation rejected by the mediator

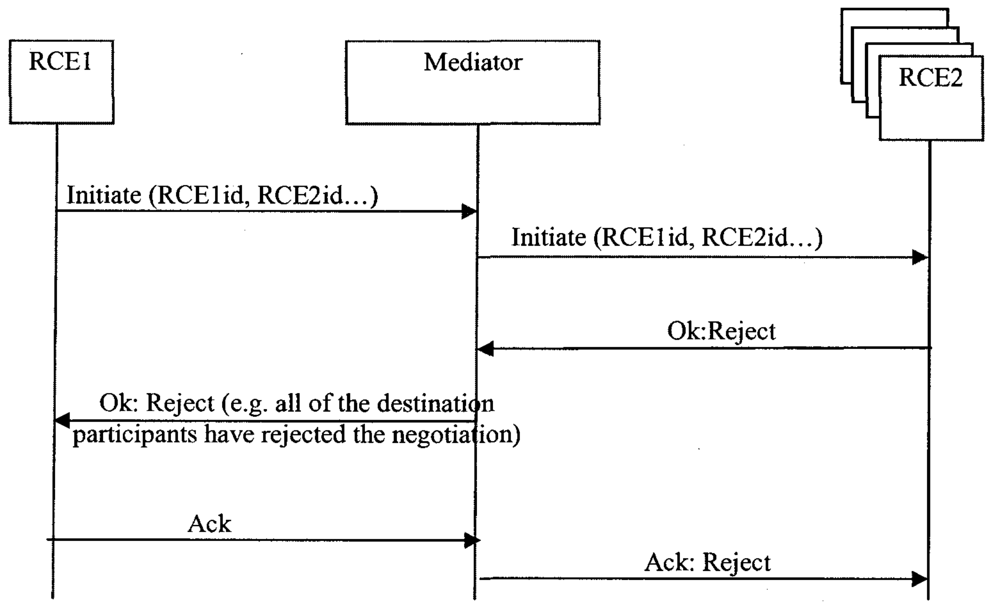

RCE1
Mediator
RCE2
Initiate (RCE1id, RCE2id...)
Initiate (RCE1id, RCE2id...)
Ok:Reject
Ok: Reject (e.g. all of the destination participants have rejected the negotiation)
Ack
Ack: Reject

Figure VI.6: Negotiation rejected by destination participants

VI.2.2 State diagrams

For the state diagrams, we focus on the core process of the negotiation (i.e. the initiation, negotiation and termination phases). We assume that the composition agreement is created only when all participants reach an agreement, and that no entity leaves the negotiation after it is started. The negotiation process is triggered by a C-FE sending a *tg-Initiate* message to the initiator.

Figure VI.7 presents the state diagram of the participant entity. Incoming messages are prefixed with a question mark, the outgoing messages with an exclamation point and the conditions are presented in brackets.

The initial state of the participant process is idle. If it receives a *tg_Initiate* message (from the RCE that initiates the negotiation), the participant becomes an Initiator. Therefore, it requests a new negotiation process by sending an *Initiate* message to the mediator. It then moves to the Waiting_Acceptance state, where it waits for the acceptance of the negotiation that he just requested. If the negotiation is rejected, it acknowledges the response reception and goes back to the idle state. If the negotiation is accepted, it goes to the Waiting_Offer state (where it waits for a proposal) after acknowledging the response reception.

If while in the idle state the participant receives an *Initiate* message (from the mediator), it takes the role of a Responder. Then, it first verifies if it can participate in a new negotiation. If no, it sends a *Reject* message and goes back to the idle state after receiving an acknowledgment of its response. If it decides to participate in the negotiation, it responds by an *Accept* message and waits for confirmation that the negotiation will take place. If it receives a negative confirmation (i.e. *Ack:Reject*), it goes to the idle state. If it receives a positive confirmation, it then waits for a CA-proposal. Each time a proposal is received; the participant evaluates it, sends its response and waits for the mediator response. If a final response is received, it acknowledges the response and returns to idle.

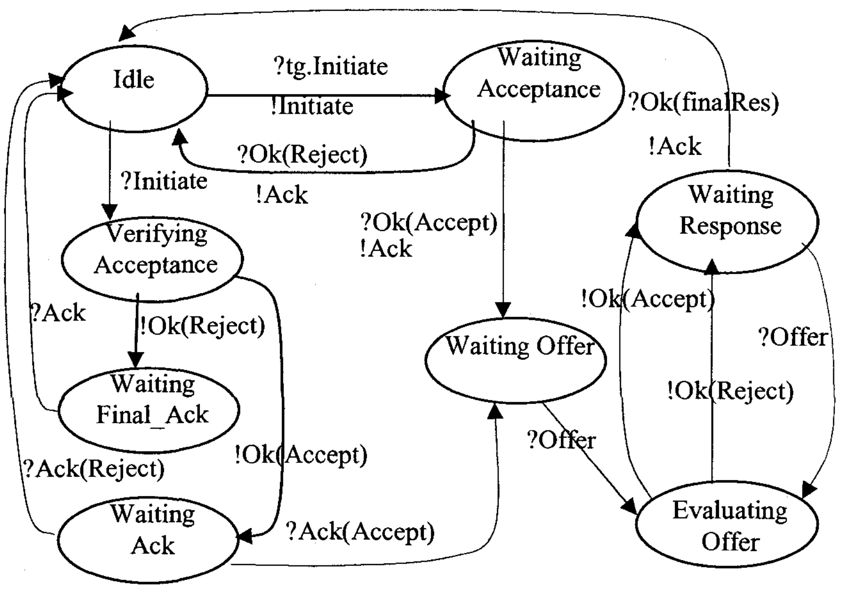


Figure VI.7: Participant state diagram

The behavior of the mediator is presented in the state diagram of Figure VI.8. When in the idle state, it can only accept an *Initiate* message. Then, it verifies if it can accept a new negotiation. If this is not possible, it sends a *Reject* message and waits for the request source to acknowledge the reception of its response. When this is done, it returns to idle. If it accepts the negotiation, it sends an *Initiate* message to the destination participants and waits for their acceptances (Waiting_Acceptances). It stays in this state until it has received all the responses. If the minimum acceptances required (q) is not reached, the mediator sends a *Reject* to the initiator and sends an *Ack(Reject)* to the other participants. If the minimum is reached, an *Accept* message is sent to the initiator, an *Ack(Reject)* is sent to the participants that rejected the negotiation, and an *Ack(Accept)* is sent to those that accepted. When an *Ack* is received from the initiator, the negotiation phase is started, where the mediator creates CA-proposals, sends them to the participants and waits for their responses. When the negotiation is terminated, the mediator sends the final response to the participants using an *Ok* message, and waits for their acknowledgments before returning to idle.

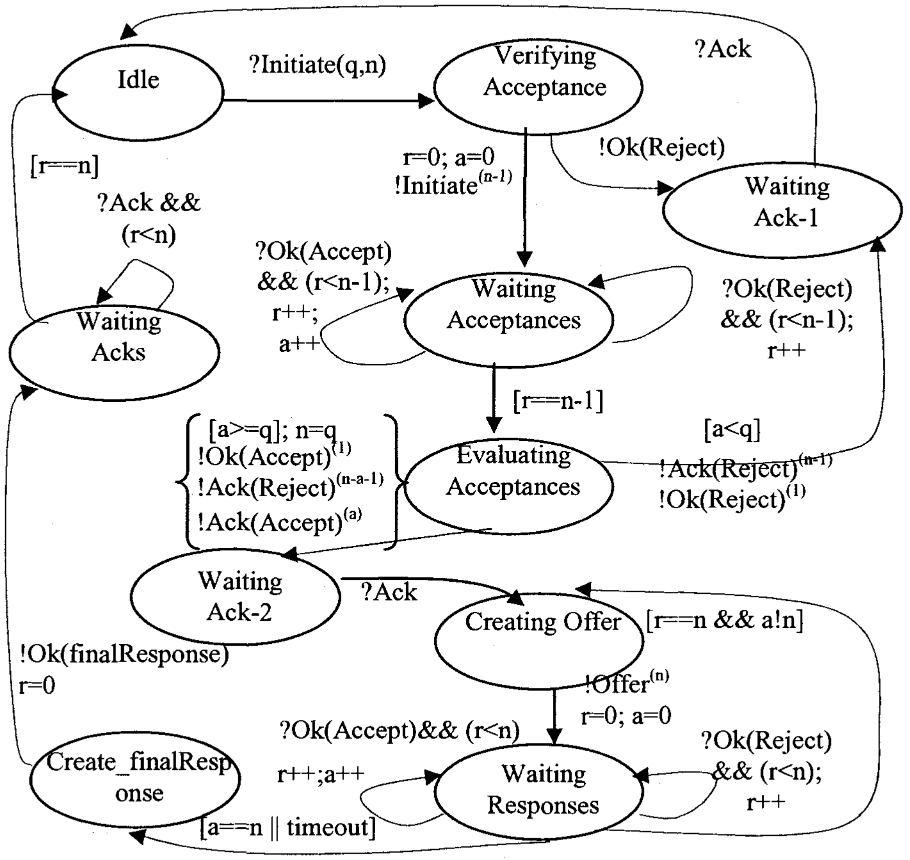


Figure VI.8: Mediator state diagram

VI.3 Support of nodes departure

After the negotiation is started, the mediator or a participant may leave the negotiation (e.g. due to network decomposition). Two types of departure are possible: voluntary departure and forced departure. Voluntary departure is when the departing entity decides to leave the negotiation (e.g. it is no more interested in the composition because the network to which it belongs moved away from the other networks). Forced departure is when the entity is forced to disconnect from the network (e.g. node failure, connectivity problems). In this thesis, we consider the two types. We also consider participant departure and mediator departure. We assume that when a participant (i.e. initiator or responder) quits, the remaining participants are interested in continuing the negotiation (e.g. if one network moves away, the other networks are sill close to each other).

VI.3.1 Voluntary departure

A. Participant voluntary departure

When a participant decides to quit the negotiation, it sends a *Bye* message to the mediator. An initiator cannot send a *Bye* message before receiving a response to its *Initiate* message. Indeed, if this is allowed, the mediator may receive the *Bye* message before the *Initiate* one. The same requirement applies to the destination participant that sent an *Ok* message to accept a negotiation.

Figure VI.9 presents the sequence diagram for a participant P1 quitting during the negotiation phase. When the mediator receives the *Bye* message, it sends a response message and terminates the negotiation session with P1, by sending the final negotiation response and accepting the final acknowledgment. The mediator also re-verifies the starting conditions (e.g. the number of participants still two or more). If the conditions are still met, the mediator continues the negotiation phase with the remaining participants. If the conditions are violated, the mediator sends a final response message to these participants, with an error message. If the mediator has already sent an offer message when it receives the *Bye* message, it waits until it gets all the responses (except from the quitting participant) before sending the next message. This will help in determining the message to send (e.g. an agreement or a new proposal). If an agreement has been reached, the final message includes the agreement and the list of participants that have accepted it.

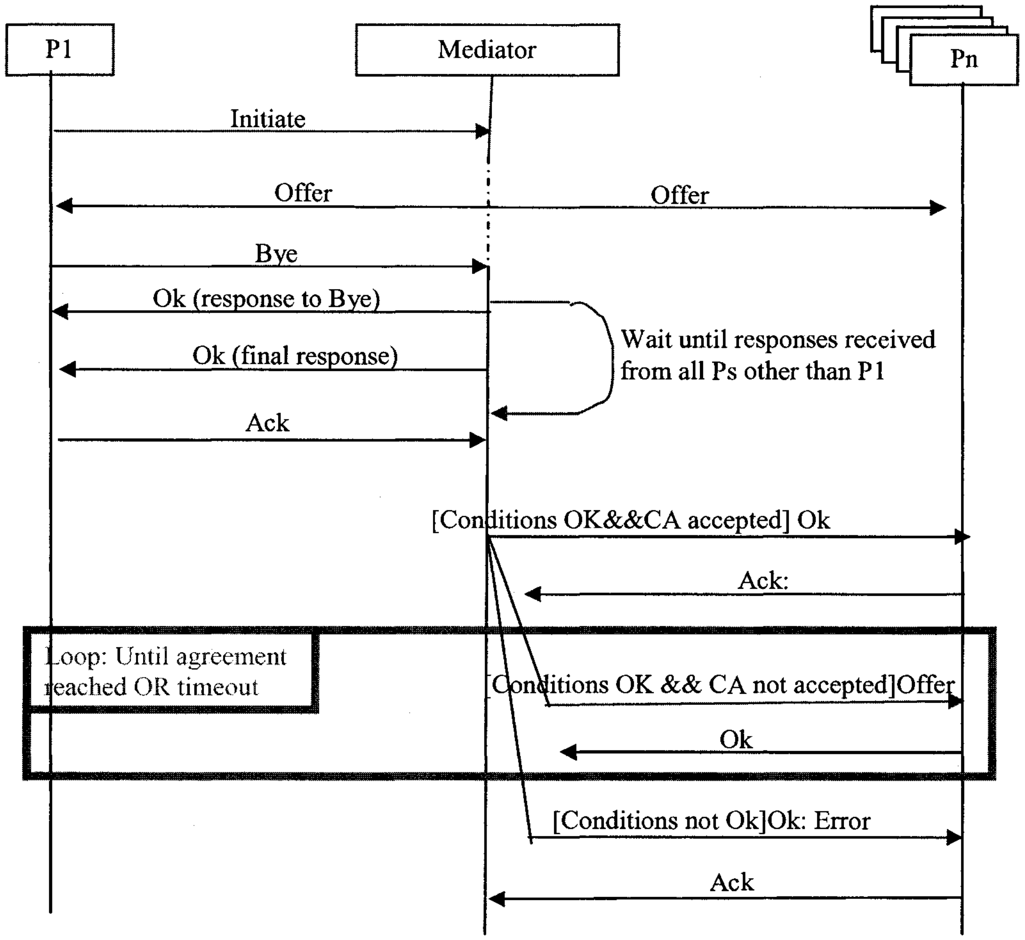

P1
Mediator
Pn
Initiate
Offer
Offer
Bye
Ok (response to Bye)
Ok (final response)
Wait until responses received from all Ps other than P1
Ack
[Conditions OK&&CA accepted] Ok
Ack:
Loop: Until agreement reached OR timeout
[Conditions OK && CA not accepted]Offer
Ok
[Conditions not Ok]Ok: Error
Ack

Figure VI.9: The initiator quits the negotiation

B. Mediator voluntary departure

If an active mediator (i.e. the mediator that is orchestrating an ongoing negotiation) decides to quit, it should insure that the negotiation process will continue among the remaining participants. Therefore, it should find another mediator that can replace it. This is done as follows:

Each mediator is responsible for keeping track of the other mediators in the network. When a mediator joins the network, it publishes itself to the network members. When a mediator receives a publication message from another mediator, it establishes a connection with it. This results in the creation of an overlay network between all of the mediators in the network (Figure VI.10).

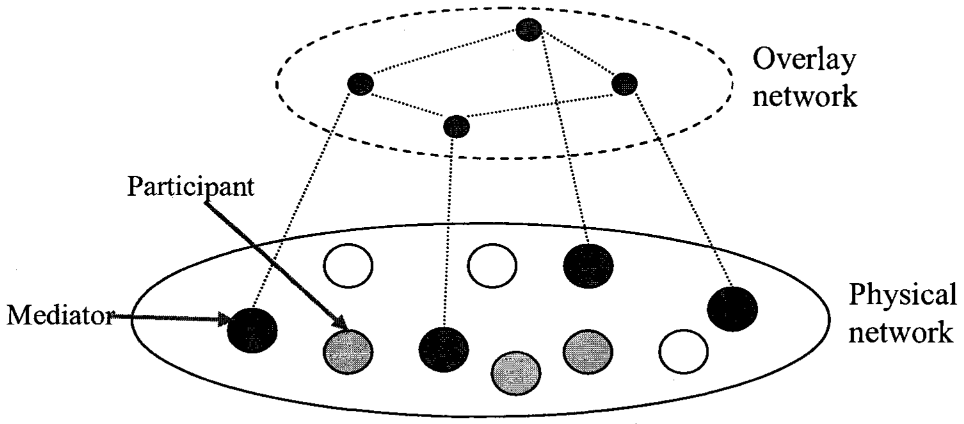


Figure VI.10: Mediators' overlay network

When an active mediator decides to quit, it sends a *Bye* message to one of its neighboring mediators, along with the current status of the negotiation and its related information. If no other mediator is part of the network, the mediator sends a *Bye* message to all of the participants, which will terminate the negotiation process. The mediator cannot quit the negotiation if it is waiting for a message from one or more participants. Indeed, if this is allowed, the expected message will be lost, and the status transferred to the new mediator will be corrupted. Therefore, the mediator should quit only when it is in a stable state (i.e. no expected incoming message is missing). Moreover, it cannot quit in the termination phase. The participants will get the address of the new mediator in the next message they receive.

VI.3.2 Forced departure

A. Participant forced departure

A participant forced departure is handled by using a timer. If the mediator fails to receive an expected message from a participant within the duration of a configurable timer, the mediator considers that the participant to whom the timer is associated has been forced to quit the negotiation.

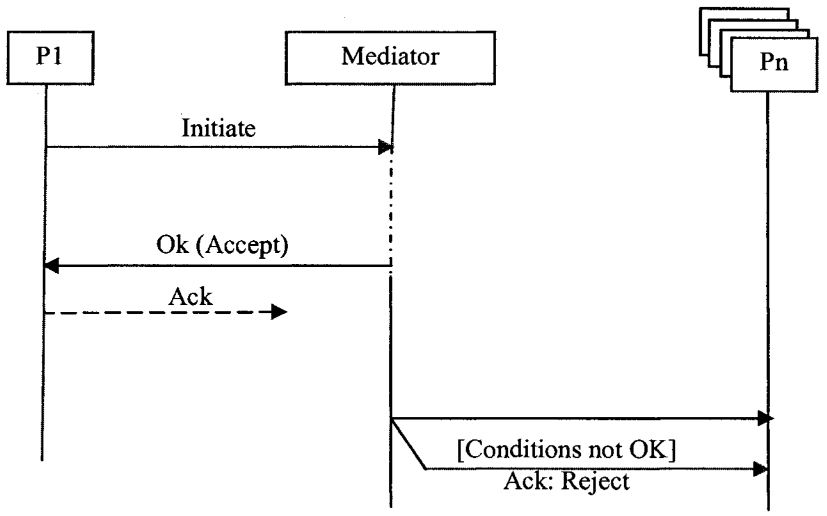


Figure VI.11: Forced departure of the initiator during the initiation phase

Figure VI.11 presents the case of the mediator failing to receive the *Ack* message that terminates the initiation phase with the initiator. The mediator considers that this is a forced departure of the initiator. Therefore, it re-verifies the negotiation starting conditions. If these conditions are still met, the mediator sends an *Ack* message -- specifying that the negotiation is accepted-- to all of the participants that accepted the negotiation, and the negotiation process continues normally. The *Ack* message includes the list of the participants that accepted to participate in the negotiation. If the conditions are violated, the mediator sends an *Ack* message to the participants with the information that the negotiation is rejected. If the mediator fails to receive a response message to an *Invite* message, it interprets this as a *Reject.* Forced departure of the other participants (i.e. not an initiator) is processed the same way as the initiator forced departure.

B. Mediator forced departure

Each active mediator (randomly) chooses one of its neighboring mediators as its backup. We assume that the probability that both an active mediator and its backup leave the network at the same time is very low. Each mediator detects the eventual forced departure of the other mediators by sending periodic heartbeat messages. For the detection of the mediators' forced departure, we used the scheme proposed in reference [68] (already discussed in the previous chapter).

Authors in [68] propose a session recovery mechanism, for cluster-based signaling architecture for conferencing in MANET. The conference participants are organized in different clusters, and each cluster has a super-member that is responsible for detecting the forced departure of the members of its cluster and the other super-members. To reach this goal, each super-member maintains a heartbeat session with each member of its cluster and with each of the other super-members. A session is defined as a signaling link between two nodes and heartbeat is defined as a periodic exchange of a request and a reply. The authors in reference [68] also propose an election algorithm to select a new super-member among several candidates, using the candidates' capabilities.

In our case, we equate the super-member to the mediator, and the members to the participants. The session is a link between a mediator and a participant or between two mediators.

If a backup mediator is no longer reachable, the active mediator to which it is assigned chooses a new backup. If the active mediator disappears, its backup will detect its forced departure using the heartbeat scheme, and will continue its ongoing negotiation sessions.

VI.4 Conclusions

In this chapter, we have proposed a new architecture for negotiating an agreement for registry composition. We have presented the architectural principles and a template for the composition agreement proposals, and described the main steps of the negotiation. We have also described the negotiation protocol (i.e. entities, messages and state diagrams) and described how to support nodes departure during the negotiation. The architecture handles both voluntary and forced departure, of both participants and mediator.

The proposed architecture is very promising in meeting our negotiation requirements. It allows a third party to arbitrate the negotiation and create the CA-proposals, does not rely on a permanently centralized entity and it is independent of the types of composing ANs and registries and of the degree of composition. It enables autonomous negotiation and can be used to negotiate registry decomposition.

Chapter 8 discusses the validation of the negotiation protocol.

CHAPTER VII: Signaling framework

This chapter presents a general signaling framework for registry composition. The framework is a backward compatible extension of the IETF-NSIS framework, and it defines a new signaling application to support both the negotiation and execution of the registry composition agreement. This chapter starts by presenting our extensions to NSIS. Then, it describes the new signaling application. The conclusion is presented after that.

VII.1 Extensions to NSIS

NSIS is a suite of protocols for signaling about a data flow along its path. We selected NSIS as the basis for our work because it is modular and easily extensible. Furthermore, it has already been successfully used as the basis for signaling in different areas (e.g. QoS, signaling through mailboxes such as firewalls). NSIS was discussed in depth in the chapter on related work.

To meet the requirements identified in chapter 3, for the signaling framework, we added two types of extensions to the NSIS framework: a messaging layer extension and the definition of a new Signaling Application for Registry Composition (SARC). The messaging layer extension is to support flow independent applications, support symbolic names and provide a group management solution to allow point-to-multipoint message delivery. SARC application is designed to support different negotiation models, and enable both the negotiation and execution of the composition agreements.

The first two messaging layer extension functions (i.e. support of flow-independent applications and symbolic names) are already offered by the GANS framework, via the Extended-GIST (EGIST) messaging layer. Thus, we used the EGIST as the basis for the design of our messaging layer in order to provide support for group management. GANS is a set of protocols that enables signaling among ANs (e.g. to negotiate the composition agreement).

This section is organized as follows: The first subsection presents the general architecture of our framework. The second subsection discusses the messaging layer extensions in more detail, through the APIs that it provides. The third subsection presents how signaling messages are routed towards the destination. SARC is presented in the section after that.

VII.1.1 General architecture

As in NSIS, our architecture has two layers: a signaling layer and a common layer (Figure VII.1). The signaling layer consists of the SARC application, but it can include any other GANS, NSIS or new signaling application. The common layer provides the functionalities that are common to all of the signaling applications (e.g. message transportation from one node to another). It is composed of two layers: transport layer and messaging layer. The transport layer is responsible for transporting negotiation messages. The messaging layer (ML) executes the necessary common functions before sending the message to its destination. The messaging layer uses existing standard transport protocols (e.g. TCP, UDP), provided by the transport layer, to transmit signaling messages.

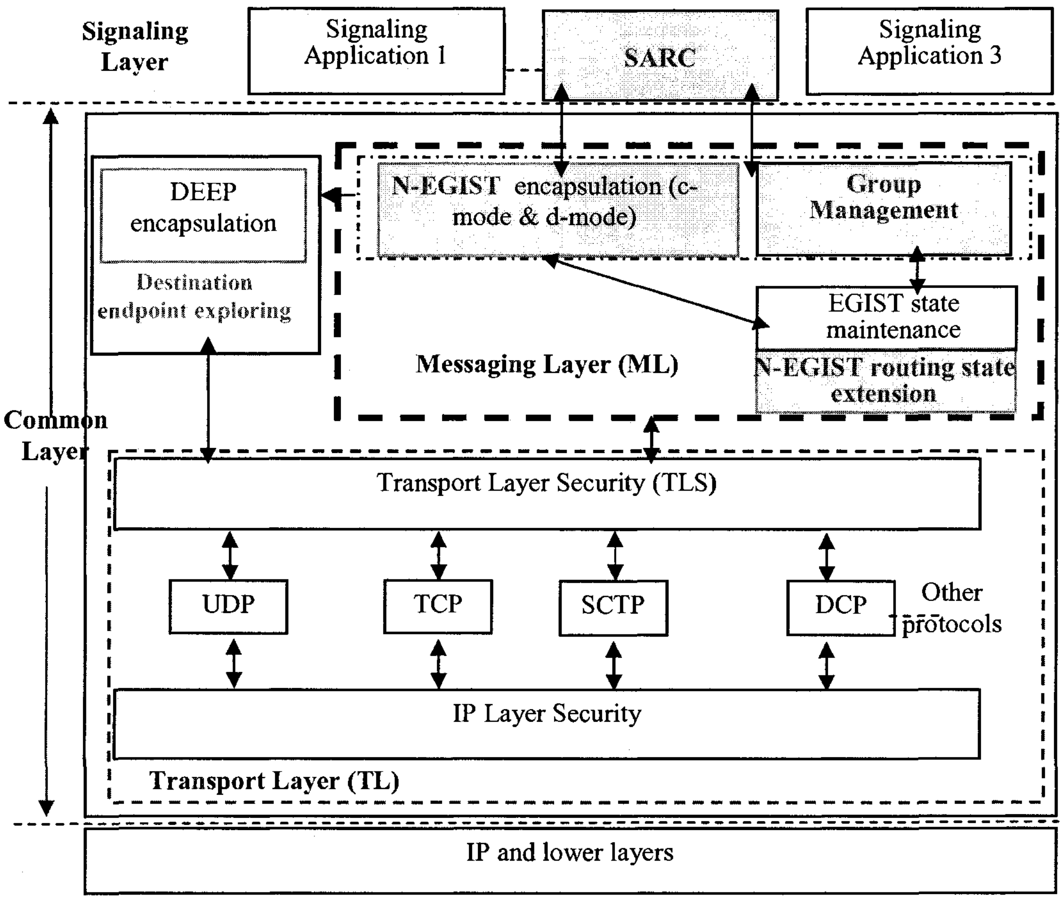


Figure VII.1. Framework architecture

Our general architecture components are related to the NSIS framework as follows: the signaling layer, the common layer and the messaging layer correspond to the NSLP, NTLP and (extended) GIST messaging layer, respectively. The messaging layer comprises two main building blocks: Negotiation- EGIST (N-EGIST) and Group Management (GM). N-EGIST is an extension of the GANS' EGIST. The main new features added by our ML to those provided by EGIST are:

- Support of point-to-multipoint
- Group management: group members can be identified by their IP addresses and/or symbolic names.
- Interaction between signaling applications and GM.
- Storage and maintenance of the name binding state, without modifying the routing state information used by EGIST for routing messages towards the group members.
- Extension of EGIST APIs to handle point-to-multipoint message delivery.

Destination Endpoint Exploration Protocol (DEEP) was added by GANS and it is used to get the IP address corresponding to a given symbolic name [25]. To translate symbolic names into IP addresses, DEEP relies on existing name resolution systems (e.g. DNS). Figure VII.2 presents a scenario where Node1 in Network1 wants to get the IP address associated to the symbolic name: ServiceY@Network2. Node1, Node2, Node3 and Node4 are DEEP nodes. Node1 issues a name resolution request (i.e. DEEP EXPLORE message), that it sends to the next DEEP node (i.e. Node2). Node2 uses a local name resolution system (e.g. DNS) to resolve the "Network2" part of the symbolic name into the IP address of a Network2 gateway (i.e. Node3). Node2 then sends the EXPLORE request to Node3. Node3 also uses a local name resolution system to resolve the "ServiceY" part of the symbolic name into the IP address of the node that provides ServiceY (i.e. Node4). The request is then forwarded to Node4, which will send a RESPONSE message with its IP address, directly to Node1. Node1 address is included in the EXPLORE message.

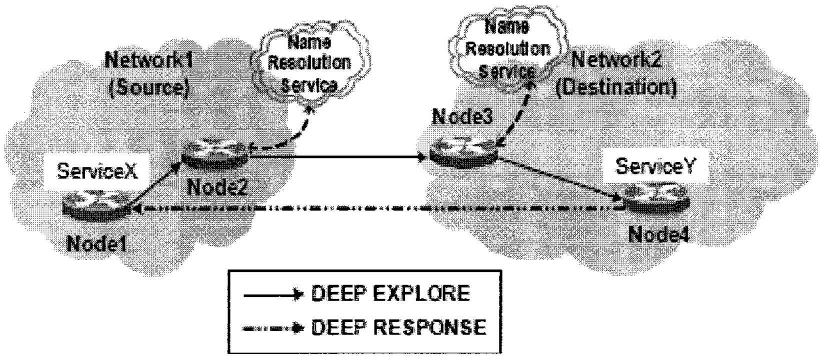


Figure VII.2: Name resolution using DEEP.

VII.1.2 Messaging layer APIs

The messaging layer Application Programming Interfaces (APIs) are the collection of group management and N-EGIST APIs. These are discussed below.

A. Group Management:

The group management module provides the signaling applications with four APIs:

- Create_group: creates a communication group, to enable communication with more than one destination (e.g. in case of one-to-many negotiation). This API takes as a parameter the list of symbolic names and/or IP addresses of the destination entities. Each group has a unique identifier.
- Add_member(group_id, name, ip): Adds a new member -identified by its name or IP address- to a created group.
- Remove_member(group_id, name, ip): removes the member identified by its name or IP address from a given group.
- Change_member_ip(name, ip): If a group member changes its IP address and the application somehow becomes aware of the new address, it uses this API to make necessary changes to the stored routing state.

B. N-EGIST

N-EGIST APIs are backward compatible with EGIST APIs. In fact, N-EGIST provides the same primitives as EGIST, but it adds new parameters and slightly changes the semantics of some parameters. This section presents only the changed primitives, with a focus on the new and modified parameters.

- SendMessage: Is used by the signaling applications to send a message to one or more destinations. It has two new parameters: group_id and min_resp. Group_id identifies the group to which the message should be sent. Min_resp is the minimum number of different responses that N-EGIST must receive before responding to the application. The Timeout parameter, already defined by GIST, is used as the length of time the N-EGIST layer can wait for min_resp responses.
- RecvMessage: Is used by N-EGIST to transmit received messages to signaling applications. In the case of a response, N-EGIST verifies if this belongs to an application that requires min_resp responses. If this is not the case, the response is directly transmitted to the application. If the response belongs to an application that requires min_resp responses, N-EGIST waits until it gets the minimum required responses, or the waiting timeout expires. Then, it creates a list containing the number of the responding parties along with their names and responses and passes it to the signaling application.

VII.1.3 Routing information

NSIS -- and thus GANS -- framework uses the Message Routing Method (MRM) to specify how signaling messages are routed towards the destination. MRM is provided by GIST, whose design supports multiple MRMs. Signaling applications indicate to GIST the MRM to be used for message forwarding. The default MRM used by NSIS framework is path-coupled, where the signaling messages follow the data-path. GANS has added an MRM to enable the use of symbolic names and allow the exchange of non data flow-related signaling messages. We add a new MRM to N-EGIST, in order to allow message delivery to all the members of a given group. N-EGIST gets the IP addresses of the group members and sends the message to each of them. The addresses of the group members are stored in the routing state table when the group is created. Updating of the name binding state is done in a similar manner as in GANS.

VII.2 An NSIS based Signaling Application for Registry Composition (SARC)

The primary function of SARC is to enable the exchange of messages related to registry composition --encapsulated in SARC messages-- between communicating peers. SARC architecture includes two entities: Requestor and Responder. The Requestor is the entity that sends a request and the Responder is the entity that responds to the request.

Signaling for negotiation and for agreement execution is end-to-end. Therefore, SARC provides an end-to-end message delivery. In other words, the communication between the Requestor and the Responder may go through a number of intermediate nodes, but the signaling messages are terminated only at the destination node (Figure VII.3). The forwarding of SARC messages is performed at the transport layer and their content is not visible to the intermediate nodes.

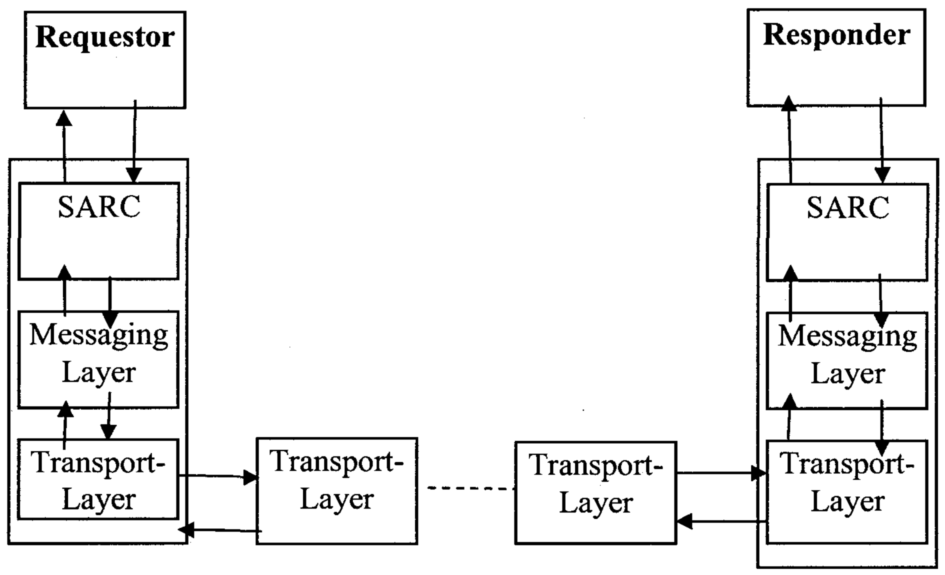


Figure VII.3: Signaling entities and topology

The following sub-sections present the SARC APIs, describe SARC message types and formats, and discuss the SARC end-to-end behavior.

VII.2.1 APIs

SARC provides the group management primitives described earlier, plus the following two interfaces:

- SendMessage: Used by negotiating entities to send a message to peer entities. Its main parameters are the type and the payload of the message to send, min_resp, the ID of the destination group (for sending requests), the destination IP address and name (for sending responses), the decision model to use for the negotiation and the negotiation approach.

- RcvMessage: Used by the SARC to pass the content of a received message to the local negotiating entity.

VII.2.2 Message types and format

SARC messages consist of a common header, which indicates the message type, followed by a body made up of a variable number of Type-Length-Value (TLV) objects. This structure makes them flexible and easily expendable.

SARC messages are of two types: CANegotiation and CAExecution. CANegotiation messages are used for CA negotiation, whereas CAExecution messages are used for CA execution. Each of the two message types has sub-types. These sub-types are as follows:

- CANegotiation sub-types: Initiate, Ok, Ack, Offer, and Bye.
- CAExecution sub-types: ActivateNode, ConfigOvNode, Join, and Quit.

The Initiate message has three main TLV objects: Local Information, Conditions and Negotiation Model. Local Information includes the information that is local for each network and which is necessary for the creation of the agreement proposals. It includes a list of Registry Information objects, where each Registry Information object describes a registry in the composing network. It includes the registry type, IPDI, and address. The Conditions object includes the Initiator conditions (if any) concerning the negotiation (e.g. an agreement is reached only if it is accepted by all of the participants). The Negotiation Model object includes information about the negotiation model to use. Examples are the negotiation approach and the decision model.

Offer includes the definition of two objects: Offer Identifier and Offer Data. Offer Identifier is a cryptographically random identifier chosen by the entity that created the offer. Offer Data includes the offer content, and it may be itself a set of TLV objects.

Each Ok and Ack message carries a TLV INFO object, which contains a response code and the corresponding object. The defined codes are as follows:

- 0x1: Accept
- 0x2: Reject
- 0x3: Agreement
- 0x4: Error

To each of the Agreement and Error codes corresponds a TLV object of the same name. The Agreement object includes three main objects: Agreement Identifier, Agreement Data and Agreement Validity Time. The first two objects are similar to Offer Identifier and Offer Data. Agreement Validity Time specifies the time after which the agreement is no longer valid.

Error has as object Error Data, which includes the error description. A message carrying a Reject code may transport a Reason object, which describes the reason of the rejection. A message with an Accept code may carry a Local Information object (i.e. when the message is a response to an Initiate message, sent from a destination participant to the mediator to accept the negotiation).

The ActivateNode message carries the list of the registries supporting the same IPDI as the message destination. Each registry is described using a Registry Information object. ConfigOvNode also uses a Registry Information object to carry information about the overlay node (e.g. IP address). The Join and Quit messages carry a Registry Information object, describing the registry that want to join or quit. If the Quit message is sent by an overlay node, it should also include the list of the registries that are served by the quitting node (i.e. the registries belonging to the multicast group maintained by the quitting overlay node).

VII.2.3 End-to-end behavior

Figure VII.4 presents the procedure for a requestor sending a one-to-many message. SARCi and MLi are the signaling application and the messaging layer on the requestor side. At the destination side, the messaging layer gets the message, records the state information, processes the message (e.g. verifies if min_res is required), and passes it to the Responder.

When creating the destination group, if any of the destination entities is described only by its symbolic name, the messaging layer uses DEEP to get the corresponding IP address (as described in GANS [25]).

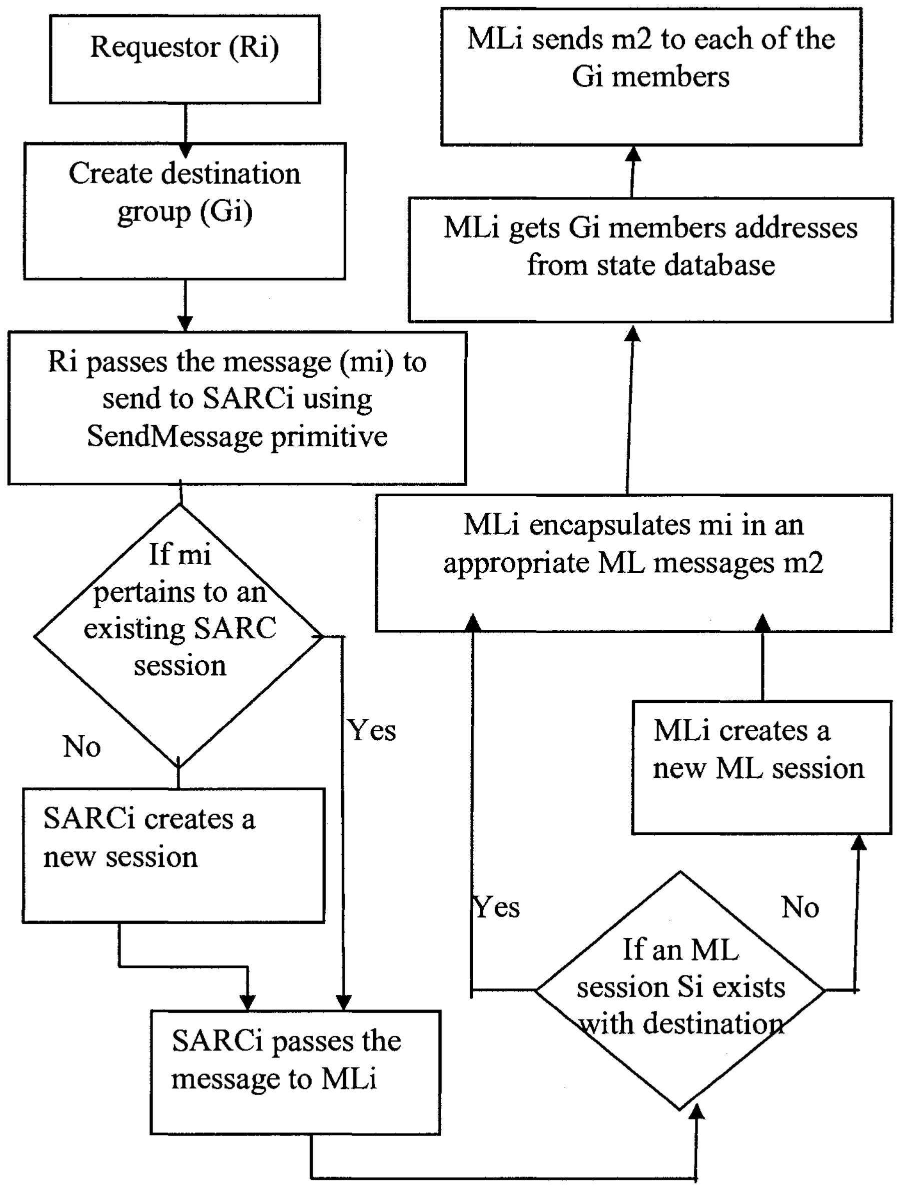

Requestor (Ri)
Create destination group (Gi)
Ri passes the message (mi) to send to SARCi using SendMessage primitive
If mi pertains to an existing SARC session
No
Yes
SARCi creates a new session
SARCi passes the message to MLi
If an ML session Si exists with destination
Yes
No
MLi creates a new ML session
MLi encapsulates mi in an appropriate ML messages m2
MLi gets Gi members addresses from state database
MLi sends m2 to each of the Gi members

Figure VII.4: Sending a message to multiple destinations

VII.3 Conclusions

In this chapter, we have proposed a general signaling framework for registry composition, based on the NSIS framework. The main extensions we have made are the support of point-to-multipoint message delivery and the definition of a new signaling application for registry composition.

The proposed framework benefits from the GANS and NSIS framework advantages and fulfills all of the signaling specific requirements. It supports both the negotiation of registry composition and the execution of the agreement reached. It is lightweight (NSIS is a lightweight framework) and it supports symbolic names and session mobility. It is modular, extensible and independent of the negotiation model. It allows the exchange of the negotiation agreements and proposals. It separates the semantic of the signaling application from the message delivery. It enables the usage of existing and standard transport protocols (e.g. TCP, UDP). It supports flow-dependent and flow-independent signaling applications.

CHAPTER VIII: Proof of concepts and evaluations

In the previous chapters, we presented a general architecture for registry composition, an architecture for information publication and discovery after composition, a negotiation architecture to negotiate registry composition, and a signaling framework for agreement negotiation and execution. A part of this thesis, we validated the general architecture and the information publication and discovery architecture through proof-of-concepts prototypes, in order to show the feasibility of the main concepts (i.e. protocol deployment on-the-fly and registry overlay network, respectively). For the negotiation architecture, we formally validated the negotiation protocol. The signaling framework scalability was validated via simulations, in order to be able to capture the behavior in large scale networks (compared to a prototype) and under different circumstances (e.g. different scenarios).

This chapter focuses on the proof-of-concept prototypes and evaluations. It starts by the prototype related to the overall architecture. Then, it presents the prototype related to information publication and discovery. It concludes after that.

VIII.1 General architecture

For the validation of the general architecture, we focused on the on-the-fly protocol deployment, since it is required by the two registry intercommunication approaches (i.e. gateway and protocol deployment).

Programmable networks can enable on-the-fly protocol deployment. For this reason, we used them as the foundation of our architecture. This section starts by a short overview of network programmability, as background information. After that, it discusses how to deploy a new protocol on-the-fly, using network programmability. Then, it presents a software architecture that will enable this deployment. After that, it presents a prototype implemented using this architecture.

VIII.1.1 Background on network programmability

Network programmability refers to the ability to inject executable mobile code into the network elements (e.g. router, switch), to create new functionalities at run time [76]. It enables the realization of application-specific service logic, or the performing of dynamic service provision on demand. Active networks are programmable networks, extensible at runtime, and they can accommodate the rapid evolution of protocols and services required by applications [77] [78] [79].

Many active network architectures use mobile code technologies. There are three approaches for active networking realization: active packets, active nodes and active packets and nodes. In the first approach, transmitted packets carry the code to be executed in the intermediate nodes. In the second approach, the packets carry the reference to predefined functions that reside in the active nodes. In the third approach, predefined and more complex code resides in the active nodes, where as specific and less complex code is carried by the active packets.

To deploy a new protocol in active networks, two approaches are possible: in-bound and out-bound [77]. In the in-bound approach, the protocol deployment is done in the same flow as the data flow, using active packets. Active packets carry the data to be transmitted and the protocol to execute in the crossed active nodes. In the out-bound approach, the protocol deployment is done in a separate flow. This approach uses active nodes and can achieve protocol deployment in two ways: The protocol can either be injected into the first node and gradually propagated from one node to another on the first packet path using this protocol, or downloaded from a protocol server.

There are many programmable network platforms such as DINA [80], ANTS [81], CANES [82] and PLANet/SwitchWare Management [83]. A review of these platforms and others is given in reference [76]. Our implementation architecture is based on DINA. We have chosen DINA because it is freely available and it allows the usage of JAVA, as opposed to proprietary languages or technologies used by some of the other platforms. It is also more flexible in term of the active functionalities provided, as opposed to the other platforms where the protocols to deploy are limited to those that can be created using the primitive "functions" or "services" provided by the active nodes. Furthermore, DINA allows the usage of "active packets" and "active packets and nodes" approaches. This gives more flexibility compared to the other platforms that support only one approach.

DINA is a programmable network platform that enables the deployment and management of programmable services [80] [84]. It can be attached to different types of network nodes (e.g. routers, media gateways) and makes them active nodes. The main components

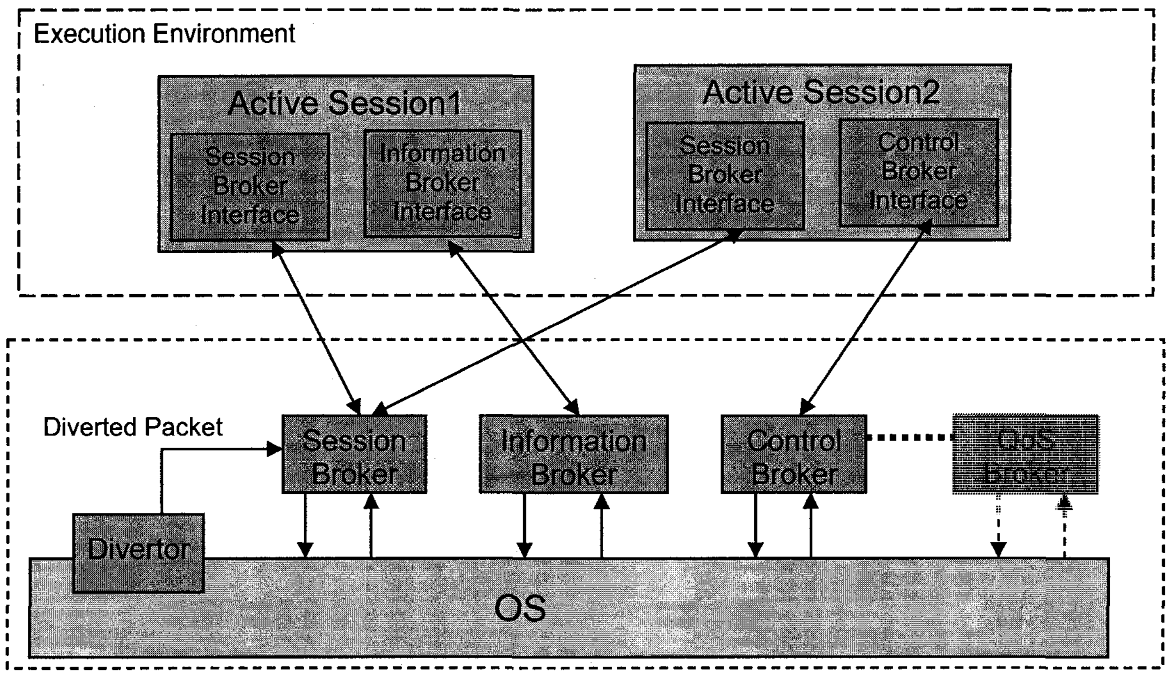


Figure VIII.1 : DINA architecture

of DINA are the active sessions, that present the active code to be executed, and a set of brokers that enable active sessions to get information from and control the managed nodes (Figure VIII.1). When an active packet reaches an active node, it is diverted to the session broker. The session broker will then create an active session that will execute the code of the active packet.

VIII.1.2 Protocol deployment on-the-fly

To enable the automatic deployment of protocols, as part of enabling registry intercommunication, we propose the following solution: for each network, we use a protocol server where we store the IPDI of the local registry, the standard protocol, and/or the interworking protocol(s) needed for gateway creation. We assume that the gateway solution is chosen only if the required interworking protocols are available in the network (i.e. thy are not created on the fly, due to the significant overhead that would be generated).

When protocol deployment is needed, RCEs negotiate the protocol to deploy and use one of the protocol deployment approaches provided by active networks to make the protocol available. The protocol agreed upon is downloaded to all appropriate nodes (registries and/or gateways). This deployment is enabled by the software architecture presented in the next section.

VIII.1.3 Software architecture for protocol deployment on-the-fly

The software architecture that we propose is based on DINA. The main service components of our architecture are the policy server, the protocol installer and the installation broker (Figure VIII.2). The policy server and the protocol installer components are added to the RCEs. The policy server includes and manages the policies that regulate the registries' composition. The protocol installer is part of the composition manager entity and is responsible for the initiation of the protocol installation. The installation broker is added to the DINA platform on the gateway/registry side to enable and control the actual protocol installation and activation.

Figure VIII.2 presents a scenario for deploying a protocol that resides in a protocol server. After the composition agreement is created, the protocol installer creates the active packet that is sent to the node where the protocol must be installed (steps 1 and 2). When this packet is received by the session broker on the gateway/registry side, an installation active session is created to execute the active code (step 3). It will start by downloading the required protocol, and then use the installation broker to install and activate the new protocol in the current node (steps 4,5,6 and 7, respectively).

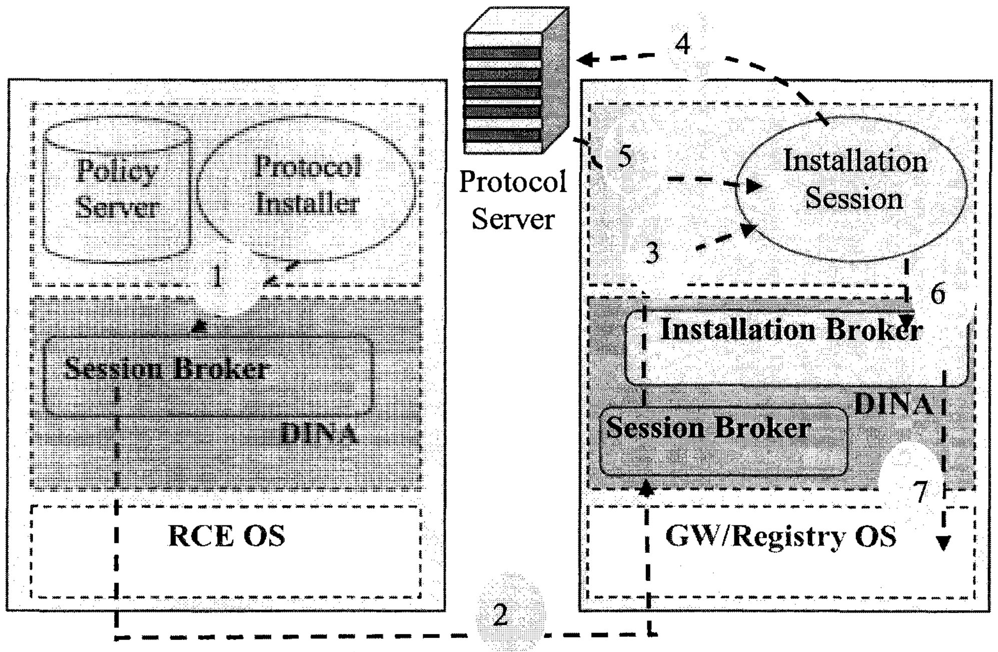


Figure VIII.2 : Protocol deployment using DINA

VIII.2 Prototype

VIII.2.1 What is implemented

As a prototype, we implemented the scenario in Figure VIII.3, where a printing application installed in the laptop needs to use the printer in Net-2. To this end, the application has to get the printer address, which is stored in the Net-2 local registry. The registry (R1) of Net-1 is a distributed registry and uses Chord [34], a P2P protocol, for information discovery and publication. Registry (R2) of Net-2 is implemented as a UDDI registry and the printing service is implemented as a web service that is published to the UDDI registry.

The UDDI (Universal Description, Discovery and Integration) registry provides standard specifications for a web service registry [7]. A web service is a "software system designed to support interoperable machine-to-machine interaction over a network' [85]. The web service architecture is based on the interaction between three roles: the service provider, the service registry and the service requestor. The service provider creates a web service and publishes its description to the service registry. The service requestor discovers the web service by consulting the service registry, binds to the service implementation and starts using the service. Communication between the three roles is carried out using Simple Object Access Protocol (SOAP) messages [86]. The most widely used transport protocol for SOAP is the HTTP protocol.

In the implemented prototype, the service requestor is the laptop, the service provider is Net-2, the service registry is R2 and the web service that we are interested in is the printing service. To access the R2 content, an implementation of UDDI APIs is used. These APIs are used for publishing, discovering, and managing information about web services. The R2 protocol stack is SOAP 1.1/HTTP 1.1. This represents the stack of protocols (and their respective versions) used by the registry, in order to enable communication with the service provider and requestor.

To use the printing web service, the client (i.e. the laptop) has to start by discovering the service, through R1. Since R1 does not include a UDDI APIs implementation, the laptop is unable to discover the existence of the printing web service. So, at composition time, Net-1 and Net-2 decide to make the laptop UDDI client compliant (i.e. the laptop becomes the gateway between R1 and R2). Then, using the implementation architecture presented earlier, the client UDDI APIs are installed in the laptop, as is the protocol

SOAP 1.1 because the laptop does not initially support this protocol. We assume that the laptop supports HTTP 1.1. The client UDDI APIs are installed to enable the laptop to communicate with the UDDI registry. HTTP and SOAP are required for service discovery and execution.

For this prototype, we assume that the composition agreement has already been created and that it consists of automatically deploying the UDDI APIs and the SOAP protocol. We also assume that the RCE that initiates the protocol deployment knows the address of the protocol server, and knows which port number on this server to use to download the protocol to deploy. At the end of the composition, the laptop will automatically discover the existence of the printing web service, and the document is automatically printed using this service.

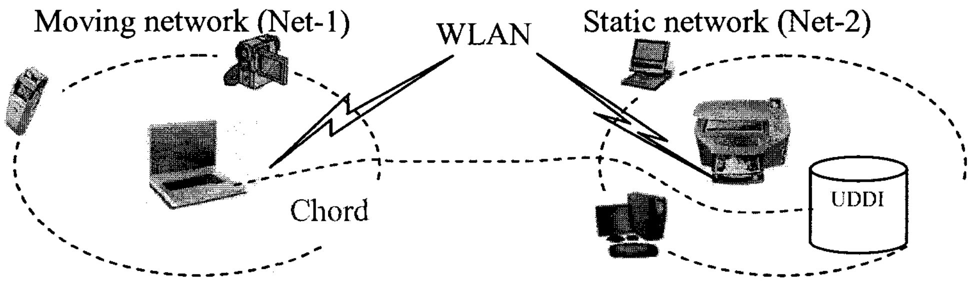


Figure VIII.3: Implemented scenario for protocol deployment on-the-fly

VIII.2.2 How it is implemented

The protocol installer component of the protocol deployment architecture, is implemented via the class **ProtocolInstallerInterface.** This class has one main public method: ***createInstallationPacket***. This method is responsible for creating the active packet to send to the registry, in order to ask it to install a new protocol. The active packet includes the protocol server address and port number and the name of the protocol to deploy.

The installation broker is implemented by the class **InstallationBrokerInterface.** This class provides the functionalities required to download and install the new protocol. Its main methods are:

- ***downloadProtocol(String protocolName, InetAddress protocolServerAddress, int portNumber):*** This method downloads the protocol identified by the *protocolName* parameter, from the protocol server located at the IP address and port number specified in the parameters *protocolServerAddress* and *portNumber,* respectively. It also downloads the active code that will generate the installation instructions.
- ***createSetupFile(String setupActiveCodeFileName):*** This method will search for the file named by the *setupActiveCodeFileName* parameter in the downloaded protocol directory. This file is used to generate a setup file, named *setup.exe*, which will be used to install the new protocol. The setup.exe file must be created on-the-fly because it is dependent on some local parameters that can be only determined at run time (e.g. the directory the protocol is downloaded into, the directory where the DINA platform is running).
- ***installProtocol (String protocolExecutablePath):*** This method will look for and execute the generated *setup.exe* file, located on the path specified by the *protocolExecutblePath* parameter. Installation and activation of the new protocol is the immediate result.
- ***downloadAndInstallProtocol(String protocolName, InetAddress protocolServerAddress, int portNumber):*** This method downloads the protocol

identified by the *protocolName* parameter, using the method ***downloadProtocol***. Once the protocol is downloaded, it is installed by calling the ***createSetupFile*** method, then the method ***installProtocol***.

After the new protocol is downloaded, it is saved to the local file system before being installed. This is done via the **FileBrokerInterface** class. The **FileBrokerInterface** is used by the active session to access the local file system. It is introduced to maintain the separation between the local system and the active session. This will facilitate session monitoring and control of the access to the system. It provides two methods:

- ***createDir(String newDirName, String ParentDir):*** This method creates a new directory named *newDirName* if the directory does not already exist. If the directory cannot be created, this method generates an exception to explain the failure reason.
- ***saveFile(String fileName, File parentDir, String info, Boolean append):*** This method creates a new file named *filename,* if one does not already exist. Then, it either overwrites the existing file or appends the *info* content based on the *append* parameter.

Figure VIII.4 presents the sequence diagram for protocol deployment. It presents the main steps and classes used for deployment. The class FileBrokerInterface is not presented in this diagram for the sake of clarity.

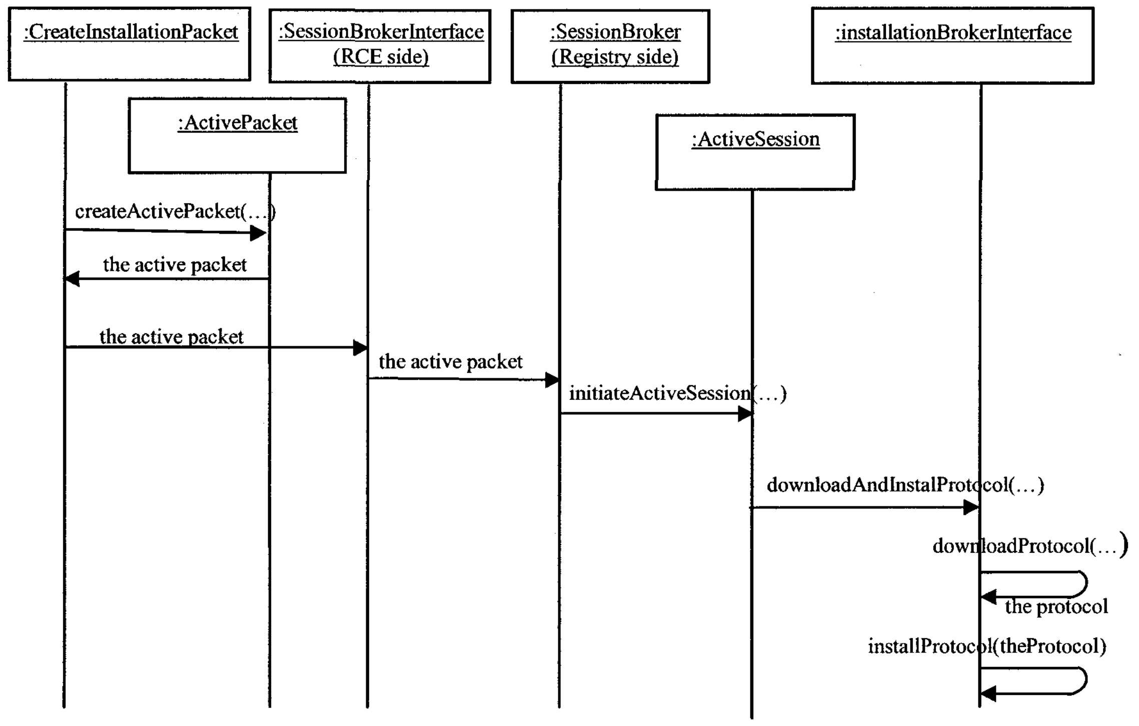

:CreateInstallationPacket
:SessionBrokerInterface (RCE side)
:SessionBroker (Registry side)
:installationBrokerInterface
:ActivePacket
:ActiveSession
createActivePacket(...)
the active packet
the active packet
the active packet
initiateActiveSession(...)
downloadAndInstalProtocol(...)
downloadProtocol(...)
the protocol
installProtocol(theProtocol)

Figure VIII.4 : Sequence diagram for new protocol deployment

VIII.3 Information publication and discovery after composition architecture

We focused on the interface interworking architecture, since it is also used as basis for data interworking. In particular, we describe the implementation and creation of the overlay network, and the information publication and discovery procedure. This section presents the software architecture of the virtual registries and describes the implemented prototype.

VIII.3.1 Architecture of the virtual registries

The architecture of a virtual registry is presented in Figure VIII.5. The Overlay-Application module includes the intelligence and the logic required for information discovery and publication. The IPD-Service module with the IPDI provides the "Registry Interface" of the virtual registry. The Overlay-Service module with the overlay protocol provide the "Overlay Interface". The "Registry Interface" is used to communicate with a registry that supports the same IPDI as the overlay node to which the application belongs. To communicate with a registry that supports a different IPDI, the application identifies the overlay node that supports the same IPDI as the target registry and sends the message to it. This node will then transmit the message to the target registry and send the response, if any, back to the initiating node.

The re-director module is added to each registry, to enable registries to redirect the requests received from clients to the RON when needed.

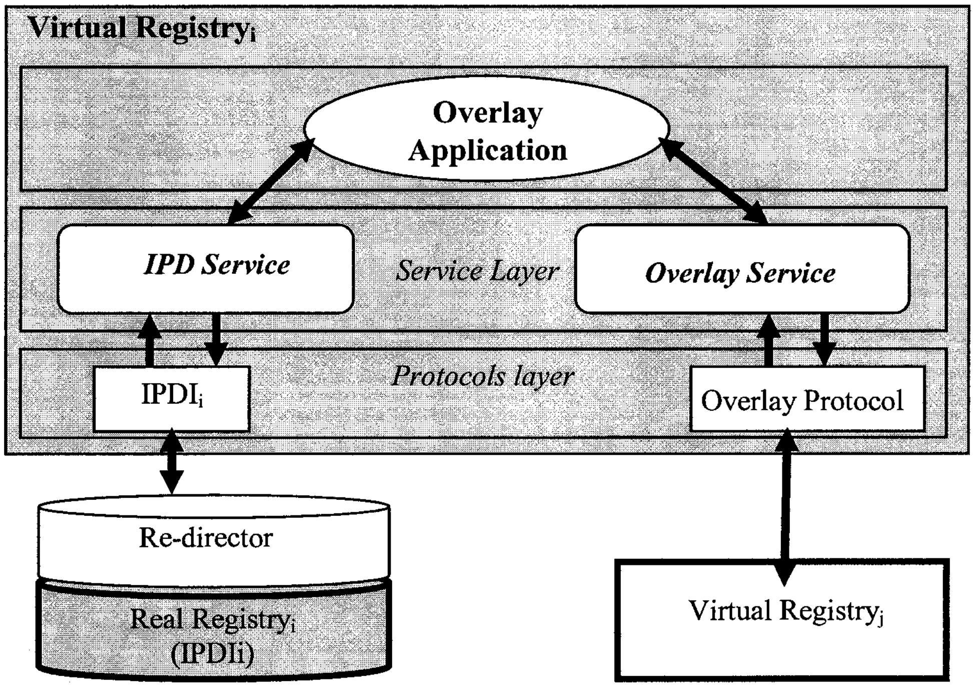


Figure VIII.5: Architecture of an overlay node

Next sub-section presents the application programming interfaces provided by the service layer of the architecture

VIII.3.2 Application Programming Interfaces (APIs)

Two types of APIs are provided by the service layer: APIs provided by the IPD-Service, and APIs provided by the Overlay-Service.

IPD-Service APIs: Used by the overlay-application to communicate with a registry that supports the same IPDI, they are:

- **Get_description_request**: Gets the description of the registries given as parameters.
- **Publish_info_request**: Publishes information to a given registry.
- **Retrieve_info_request**: Retrieves information from a given registry.
- **Send_response_request:** Sends a given response to a given registry. The response may be created by the local overlay application or received from another post-composition or virtual registry. It can be of any type, such as: the requested information (e.g. in the case of information discovery), a success response (e.g. the information was correctly published) or an error response.

Overlay-Service APIs: Used by the overlay-application to communicate with another overlay-application (the last two methods), or with a post-composition registry that supports a different IPDI (the first three methods), via another virtual registry.

The first three primitives "**Publish_info_request**", "**Retrieve_info_request**", and "**Send_response_request**" are similar to the primitives of the same name presented above. The only difference is that overlay-service APIs are used to send a message to a registry with a different IPDI. The other two methods are:

- **Publish_description_request**: Publishes a registry description to the overlay network.
- **Find_registry_request:** Finds the registry that stores given information, by interrogating the overlay network.

VIII.3.3 Prototype

As proof of concept, we implemented the scenario presented in Figure VIII.6 (the scenario was already described in chapter 5).We consider that the registries R1, R2, R4 and R5 are centralized. R3 is a P2P registry. This section discusses how the virtual registries and the registry redirection modules are implemented, describes the end-to-end behavior that summarizes how the RON is created and how the printing application gets the requested information, and presents the performance evaluation.

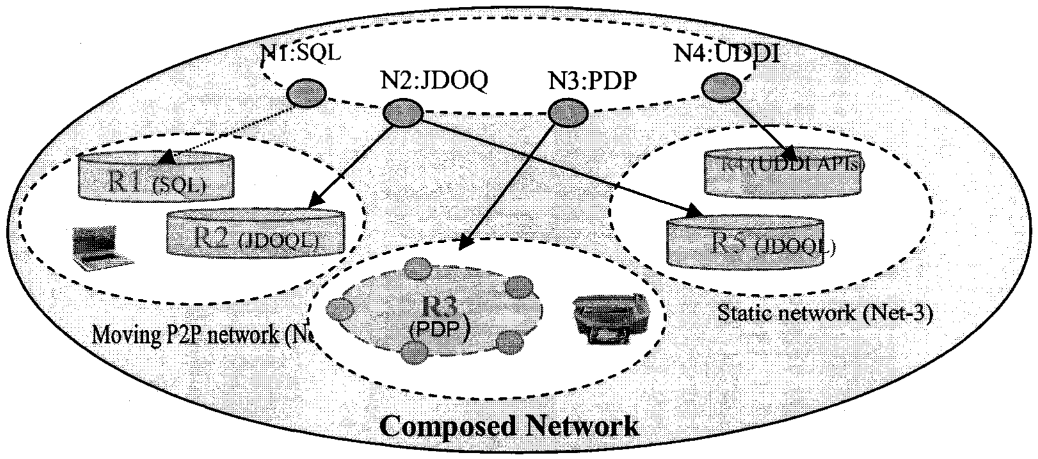


Figure VIII.6: Implemented scenario for registry overlay

VIII.3.3.1 Modules implementation

Virtual registries: The implementation of the virtual registries is based on JXTA middleware [87]. It is a set of open protocols that allow devices on the network to communicate and collaborate in a P2P manner. We have chosen JXTA because it is platform-independent, it allows an extensible and expressive description of the different registries (i.e. the descriptions are not limited in the type and amount of information to include) and it supports all types of network devices (e.g. PDAs, computers).

The Peer Discovery Protocol (PDP [87]) of JXTA is used as the overlay protocol of our architecture. In JXTA, PDP is used to publish and discover resource advertisements. A resource can be a peer, a peer group, or any resource or service that has an advertisement. An advertisement is a meta-data document used to describe resources. JXTA advertisements are presented in XML. In our case, the resources to advertise are the different registries. Figure VIII.7 shows the advertisement template we used to describe the different registries.

The implementation of the virtual registries includes the implementation of the related modules and APIs, and the related procedures, except those for RON churn and super-node selection.

```
<RegistryAdv>
    <Id> registry unique id </Id>
    <Name> registry name (optional) </Name>
    <registryAddress>
        <address>the registry address (e.g. ip address)</address>
        <port> registry port number </port> </registryAddress>
   <registryType/>e.g. UDDI registry        V3 </registryType>
   <infoType>e.g. web services descriptions</infoType>
    <infoDescription/>        e.g. printing web service. A registry advertisement can
   include more than one inoDescription elements.    </infoDescription>
   <repOvNode> provides information about the overlay node to use, in   order to
   communicate with the advertised registry</repOvNode>
</ RegistryAdv >
```

Figure VIII.7: registry advertisement template

Registry re-director module: The re-director module is implemented with the related functionality. It includes two sub-modules: the “redirectorLogic” that implements the necessary logic to enable real registries to redirect the requests received from clients to the RON, and the “redirectorInterface” that enables communication between the first module and the traditional registry or an overlay node. The “redirectorLogic” provides two main functions:

- **setRepOvNode**: configures the address of the virtual registry to which to redirect the received requests, when necessary. It is called when the RON is created: when a virtual registry VRi receives an activation message from the RCE, it sends two messages to the members of MGi: "Get_descrition" to retrieve the registry's description and "ConfigOvNode" to initiate the execution of the *setRepOvNode* function.
- **processRequest**: processes the received requests (i.e. publication or discovery requests), by executing the corresponding procedures. It is called when a new request is received.

VIII.3.3.2 End-to-end behavior

The RCE creates four multicast groups (Figure VIII.8). Then, it chooses R1, R2, R3 and R4 as overlay nodes, which it will activate in order to act as VR1, VR2, VR3, and VR4, respectively (i.e. this activates the virtual registry modules already installed in these nodes). In the activation phase, each VRi gets the description(s) of the registry(ies) it represents and publishes it(them) to the overlay network using the JXTA platform. It also sends an activation message to the appropriate re-director module, in order to configure the address of the representing overlay node.

When the printing application sends a request to R1, the request is received by the re-director module of R1 (D1). D1 interrogates R1 for the requested information and gets a null answer (R1 does not have the information). Then, D1 redirects the request to VR1, and VR1 uses the JXTA capabilities to discover the registry that maintains the requested information (i.e. R3) and to send a Retrieve_information message to it. The response is sent back to the application through VR1 and D1.

```
Group: 1
   [
    R1: [IPDI=SQL, ipAddress=192.168.0.100, port=5200]
   ]
Group: 2
   [
    R3: [IPDI=PDP, ipAddress=192.168.0.101, port=4221]
Group: 3
   [
    R2: [IPDI=JDOQL, ipAddress=192.168.0.102, port=5202]
    R5: [IPDI=JDOQL, ipAddress=192.168.0.103, port=5202]
   ]
Group: 4
   [
    R4: [IPDI=UDDI, ipAddress=192.168.0.105, port=5204]
   ]
```

Figure VIII.8: Multicast groups created

VIII.3.3.3 Performances evaluation

We measured the total time needed for RON creation ($T_{ron\text{-}creation}$) and the time overhead for information discovery via the RON ($T_{overhead}$). $T_{overhead}$ is the difference between the time needed to discover information in R3 via the RON and the time needed if the client (i.e. the printing application) has a direct access to R3.

We used the following configuration for running the prototype: Each registry (i.e. the actual registry and the re-director module) is running in a different machine, whose characteristics vary between machine1 (Pentium 4 CPU 2.66 GHz, 512 MB of RAM) and machine2 (Pentium M CPU 1.73 GHz, 504 MB of RAM). The RCE is running on machine2.

$T_{ron\text{-}creation}$ is calculated using formula (1). T_{mg} and T_{nd} are the time needed to create the multicast group and the longest time to activate an overlay node, respectively. The messages' transmission and propagation times are not considered. $T_{ron\text{-}creation}$ does not depend on the number of the overlay nodes to activate, because these are activated in parallel (i.e. via a multicast activation message). The measured T_{mg} is almost 0ms. The T_{nd} average measured is 4ms.

$T_{ron\text{-}creation} = T_{mg} + T_{nd}$ (1)

$T_{overhead}$ average is 1.3s. Most of this overhead is introduced by the execution of the find_registry procedure. The average of $T_{find\text{-}registry}$ is 1.1s, which is the time used by the JXTA platform to discover a given information. This is somewhat high, due to a problem we encountered when using JXTA: we were unable to discover a registry description based on its attributes (e.g. infoDescription). Only a discovery based on the description Unique ID is possible. Therefore, to implement the registry_discovery procedure, we started by discovering the list of all of the existing descriptions, and then we selected the appropriate one by going through the list and comparing the attributes of each component to the attribute values provided.

VIII.4 Conclusions

In this chapter, we proposed a software architecture for protocol deployment on the fly and described the application programming interfaces provided by the service layer of the architecture of the virtual registries that make up the overly network for information publication and discovery after registry composition. We also described implemented prototypes for protocol deployment on the fly and for the information publication and discovery architecture.

Some important lessons were learned. First, DINA seems to provide a flexible platform for the implementation of active nodes. The new designed deployment architecture seems to be very promising for software deployment on the fly. It can basically be used to install and activate any software or protocol, provided that the installation don't require a re-boot of the machine and all its required actions can be executed using a combination of the functionalities provided by the Java Virtual Machine and the Operating System. Second, JXTA seems to provide a suitable platform for implementing the overlay nodes, in the case of P2P and centralized networks. However, it may not be suitable for Mobile Ad-hoc NETworks (MANETs). Indeed, JXTA does not efficiently support highly dynamic environments, where nodes frequently leave and join the network.

CHAPTER IX: Formal validation, simulations and evaluations

This chapter starts by presenting the formal validation of the negotiation protocol. After that, it presents the scalability validation of the signaling framework, using simulations. The conclusion is presented after that.

IX.1 Negotiation architecture

For the validation of the negotiation architecture, we focus on the negotiation protocol, because it is the core component of the architecture. We start by presenting the validation environment used. Then, we present the validation models, define the correctness requirements to validate and describe the validation process and results.

IX.1.1 The validation environment

The validation is conducted using SPIN –Simple PROMELA INterpreter- [88]. It is a software tool for simulating and validating programs written in PROMELA. PROMELA is a language for modeling the interactions of processes in a distributed system. It is defined at a high level of abstraction, which allows designers to focus on the system design than in its implementation. PROMELA programs are called validation models.

PROMELA allows also the description of the correctness criteria about the behavior of the validation models. These criteria can be expressed using two types of claims: they can either be formalized as inevitable or impossible behaviors. In PROMELA, the correctness criteria are expressed as behaviors that are claimed to be impossible.

SPIN includes two modules: simulator and validator. The simulator simulates the system behavior by interpreting the PROMELA program on-the-fly. The validator validates the model through an automatic application of the correctness requirements. There are two main methods of validation: exhaustive search and controlled partial search. In the exhaustive search, all reachable states of the interacting finite state machines of the system to validate are explored. In controlled partial search, only a partial set of these sates are analyzed. The states to analyze are selected in such a way that all major protocol functions are tested. This technique is used when the system size does not allow the usage of the exhaustive search.

IX.1.2 Validation models

PROMELA models: We developed PROMELA validation models for the mediator, the participant and the C-FE that initiates the negotiation. The first two models are based on the behavior of the protocol entities (i.e. mediator and participant). The third model is used only to send the triggering message *-tg_Initiate-* to the initiator.

We assume that the mediator and the participants are connected via a reliable link (when they are reachable), so that no messages are reordered or duplicated. To release this assumption, we can use a layered architecture where our protocol layer uses an under layer transmission module that takes care of putting the received messages in the right order and of removing the duplicated ones. An example of a such module is given in [88]. The usage of the transmission layer will not affect the behavior of our models.

We also assume that the composition agreement is created only when all of the participants reach a common agreement. The only starting condition completely modeled is the fact that the number of the accepting participants (i.e. participants that accepted the negotiation) must be at least equal to a configurable variable min_participants (min_participants' default value is 2). Starting conditions are the conditions for the negotiation to take place. The details of the other conditions are part of the internal processing of the mediator and they cannot affect the external behavior of the model used. Nevertheless, to model the general case, we used a non-deterministic choice between successful and fail, when verifying if the starting conditions are met.

The use of non-deterministic choice may result in choosing the successful case, even if only one participant is in the session. To avoid this, we used the non-deterministic choice only in case the number of participants in the negotiation is equal to min_participants or more. This is modeled as follows:

```
if
        ::( nbr_participants < min_participants ) ->
                        goto Fail;
        ::else ->
                if
                        ::True -> goto Fail;
                        ::True -> goto Success;
                fi
fi
Fail:  /*processing for the negotiation rejection (if in the initiation phase) or termination
          (if in the negotiation phase)*/
Success: /*continue the negotiation process*/
```

To model the case of entities quitting the negotiation, we used a non-deterministic choice between sending the actual message (if the entity is interested in continuing the negotiation), or the Bye message.

Communication channels: The models include the definition of three main channels of communication: cfe_to_initiator, to_mediator and to_participant. cfe_to_initiator is used to send the triggering message *tg_initiate* to the initiator. to_mediator is used by the participants to send messages to the mediator. For the mediator to figure out which participant sent him a given message, each participant is given a unique id within a negotiation session, and each message includes the id of its sender.

To send a message to all of the participants, the mediator needs a point-to-multipoint channel. However, PROMELA channels are only point-to-point. To model the point-to-multipoint, we used a different channel for each participant. We then declared the to_participant as an array of channels as follows:

chan to_participant[max_participants_number] = [QSZ] of {byte,byte}. QSZ is the maximum size of each channel. Each message sent by the mediator includes the message type (e.g. *Ok*) and the information conveyed by the message (e.g. *Reject*). The channel associated to each participant is indexed by the participant id. For instance, the messages intended to the participant whose id is 5 are written/read to/from to_participant[5].

Timeout simulation: our system includes two types of timeouts: negotiation timeout and message timeout. The negotiation timeout is the timeout after which the negotiation is aborted if no agreement is reached. The message timeout is the timeout that an entity waits for a message, before it declares the message source unreachable. The first timeout is modeled by the maximum number of times that the negotiation phase can be repeated. The timeout is formalized as an integer that is incremented each time the mediator creates an offer. The timeout fires when its value reaches a configurable upper-limit. The second timeout is modeled by combining the default timeout of PROMELA and the simulation of an entity never sending a given message. The default PROMELA timeout gives the possibility to make an entity stop waiting for a message that can never be received. To simulate an entity never sending a message, we used a "thief" process that steals the messages sent by this entity. This process is modeled as follows:

```
proctype Thief(int p_id) /*p_id is the id of the participant from which to steal messages*/
{
do
        ::toParticipant[p_id]?msg_type(process_id,msg_sub_type)
        ::toMediator?msg_type(msg_sub_type)
        ::skip
 od
}
```

The Thief process randomly chooses to steal the current message sent to the mediator, to the participant identified by p_id, or do nothing.

IX.1.3 Correctness requirements

The negotiation process can either terminate with a successful result (i.e. agreement reached), or with an error. The first requirement (R1) that we want to validate is that the protocol is deadlock-free, there is no wrong unreachable statement (i.e. a statement that must be reachable but it is not), and that the mediator and the participants go back to the initial state (i.e. idle), in the two cases of the termination (i.e. success and error). To this end, we mark the idle state as the only valid end-state for the two entities.

As a second requirement (R2), we are interested in validating that each started negotiation phase is correctly terminated and that the negotiation phases are always executed in a correct order. The initiation phase is started when the *Initiate* message is sent (for the initiator) or received (for the mediator and destination participants), and terminated by sending necessary *Ack*(s) (for the initiator and the mediator) or by receiving an *Ack* (for the destination participants). An initiation phase is said to be successful if it results on the acceptation of the negotiation (i.e. an *Ack*(Accept) is sent to the destination participants and received from the initiator). The negotiation phase starts at the end of a successful initiation phase. It terminates when an agreement is reached or an error occurred. The termination phase starts at the termination of the negotiation phase and terminates by sending (receiving) the final *Ack*(s) by the participants (the mediator).

To validate for instance that the negotiation phase always starts after the initiation phase has been successfully completed, we instrumented the validation models by adding a variable "successfulInitiation" to them. This variable is initialized to false and is turned to true only when the initiation phase is successfully completed. To validate our requirement, we added the statement Assert(successfulInitiation) to the mediator and participant models, just before they start the negotiation phase. The added statements (e.g. variable definition and assertions) are used only for validation and they do not affect the protocol correctness.

Our third requirement (R3) is about the correctness of the different negotiation phases, when all the negotiating entities are reachable during the whole negotiation process. For the initiation phase, we want to validate that the mediator and the participants always respond to an Initiate message, with a *Reject* or *Accept* message within a finite amount of time. Similarly, in the negotiation phase, the responder should always respond with *Accept* or *Reject* to the received *Offer* messages, within a finite amount of time. These requirements are formalized using temporal claims. The claim in Figure IX.1 formalizes the requirements on the responder. It specifies that the following responder behavior is absent from the model: the participant never receives an *Initiate* or an *Offer* message, or it receives any of them and never responds to it, neither by *Accept* nor by *Reject.* Seen that all of the participants have a symmetric behavior, the requirement was applied only to the participant with the id 1. The requirement for the mediator can be formalized in a similar way.

A similar procedure is followed to validate that each started negotiation session is correctly terminated (R4). The session is created after the initiation phase is successfully terminated, and is finished at the end of the termination phase. This can be formalized by specifying that the following behavior is impossible: the session is never created or there is a case where it is created and never/wrongly terminated (i.e. because the negotiation or the termination phases never end).

Our last requirement (R5) is that the case of entities quitting the negotiation (i.e. by getting unreachable or by sending a Bye message) is handled correctly. This means that whenever an entity quits the negotiation, the system executes a correct sequence of statements and goes to a correctly stable state in a limited number of steps. We also have to re-verify the previous properties in this case. This requirement is also verified using temporal assertions.

```
never { do
        ::!to_participant[1]?[Initiate]&&
          !to_participant[1]?[Offer]
        :: to_participant[1]?[Initiate]-> goto accept0
        :: to_participant[1]?[Offer] -> goto accept1
      od;
  accept0:
      do
        ::!to_mediator?[Accept,1,Initiate] &&
           !to_mediator?[Reject,1,Initiate]
     od;
  accept1:
     do
        :: !to_mediator?[Accept,1,Offer] &&
           !to_mediator?[Reject,1,Offer]
    od;}
```

Figure IX.1: An example of temporal claims used

IX.1.4 Validation results and discussion

Validation environment:

Hardware: Pentium(R)4 2.20GHz, with 512 MB of RAM.

Software: Windows XP, SPIN 4.2.7, XSPIN 4.2.7

The search depth bound was 1,000,000 and the memory limit was 512 MB.

Validation process: the validation models were created and validated incrementally, in three phases. In phase 1, we concentrated on the system behavior in case of the basic negotiation approach. In phase 2, we extended the basic models with voluntary departure. In phase 3, we validated the entire extended system (with voluntary and forced departure). We started by processing the case of voluntary departure separately because it is easier to handle a single set of problems at once, but also to make sure that the system behaves correctly in this case. Indeed, forced departure is based on PROMELA timeout, and PROMELA timeouts are executable whenever no other instruction is executable. Therefore, their usage may hide some protocol design problems that may cause deadlock; if the PROMELA timeout instruction is not used (e.g. the expected message is erroneously not sent).

In each phase, we started by simulating the protocol behavior in different scenarios. First, we simulated the negotiation process with three participants, in case of successful and error termination. Then, we repeated the simulation with different values for min_participants, different numbers of participants (from 2 to 10), and experienced the cases where the number of participants that accepted the negotiation is less/equal or more than min_participants. We then used the validator to confirm the simulation results in a formal way.

As it was impractical to use the exhaustive search for the validation, we used the supertrace mode. This is a controlled partial search technique [88], which requires much less memory than exhaustive search, but still retains excellent coverage. An indicator of that coverage is given by the factor “hash factor”. This is calculated by the validator at the end of the verification run and it indicates a very good coverage when it is greater than 100.

Models creation and validation in phase 1 was relatively simple, because the sequence of messages is very clear. However, we faced different types of problems when modeling the extended approach. Soon after we started phase 2, SPIN detected different deadlock situations. The majority of these deadlocks come from unspecified receptions and messages’ interleaving. For instance, when a participant Pi sends a *Bye* message in the negotiation phase, it waits for *Ok(Bye)* message. If the mediator has sent the final response (e.g. *Ok(Agreement)*) before receiving the Bye message, it will block waiting for the final *Ack* message from Pi, that blocks waiting for *Ok.* To resolve this problem, we modified the participant and the mediator models to take into account this scenario. When the participant sends a Bye message, it can either receive an *Ok(Bye)* and then goes for the termination of the negotiation, or directly receive a final response which closes the negotiation. The mediator ignores any *Bye* message received after the final response is issued.

Validation results and discussion: In phase 1, the validation process took less than one minute and it required around 43MB of memory for each correctness requirement. In phases 2 and 3, the average time and memory used was 07 hours and 73 MB. An exhaustive search would have required approximately 7145MB (number of stored states * memory required for each state) of memory.

The validation process concluded, with acceptable probability (hash factor>100), that the mediator and the participant models do not include any deadlock, that all theirs states are reachable, and that at the end of each negotiation, the two entities are in a correct end-state. It also concluded that the correctness assertions and the temporal claims are never violated, which means that the correctness requirements associated are met.

IX.2 Signaling framework

This section presents the simulation environment and set-up, describes the simulation models and scenarios used, and presents the simulation measurements along with their analysis.

IX.2.1 Simulation environment and set-up

The validation was done using OPNET V.12.0, a software tool for modeling and simulating communication networks and distributed systems [89]. OPNET provides a comprehensive and modular environment for user development, based on Finite State Machine (FSM). The supported programming language is Proto-C, a combination of C, C++ and OPNET Event Simulation APIs.

For network composition to occur, at least two networks that were distant should get close to each other. In the general case, one of these networks is mobile, and therein wireless. Therefore, to capture this feature and the nature of the majority of ANs, our network nodes are modeled as wireless nodes. This will also facilitate the modification of the number of nodes in simulated networks.

Figure IX.2 shows an overall view of the simulation set-up and the modular architecture of a wireless node. The OPNET environment provides the lower layer modules of the architecture, such as wireless LAN transmitter and receiver, IP, TCP and UDP. We built the modules associated to our signaling framework layers (i.e. Messaging Layer, Signaling Layer and Application Layer) directly on top of the TPAL (Transport Adaptation Layer) module. TPAL provides a common and uniform interface to the transport layer. It enables the modules running on top of it to use any transport protocol (e.g. TCP, UDP, AAL5, X.25 transport protocol, Frame Relay transport protocol) and to be easily modified to use different transport protocols. It also allows entities to address other entities using symbolic names (i.e. TPAL address). Interactions with remote modules through TPAL are organized into *sessions*. A session is a single conversation between two peer modules through a transport protocol.

Figure IX.4 and Figure IX.3 present the state diagrams of our Messaging Layer (ML) and Signaling Layer (SL) respectively. When SL receives a message from the upper layer, it encapsulates it in an SL message and transmits it to the ML. For each new TPAL session, the MessagingLayer module spawns a new ConnectionManager (Figure IX.5) process, which will handle the connection. When a node receives a request to open a session (i.e. via and OPEN_IND interruption), it also spawns a new ConnectionManager process that will take care of the connection.

ML can receive a request to send a message to the TPAL session end-point before the session is really established. Indeed, the application can issue a send request just after a create_group request. Therefore, ML creates a new sending queue for each session to create. The messages to send are added to the corresponding queue. When the session is established, the ConnectionManager process starts by sending the pending messages. Messages received from the upper layer, in order to be sent to the peer module, after session establishment are directly transmitted to the ConnectionManager to handle them.

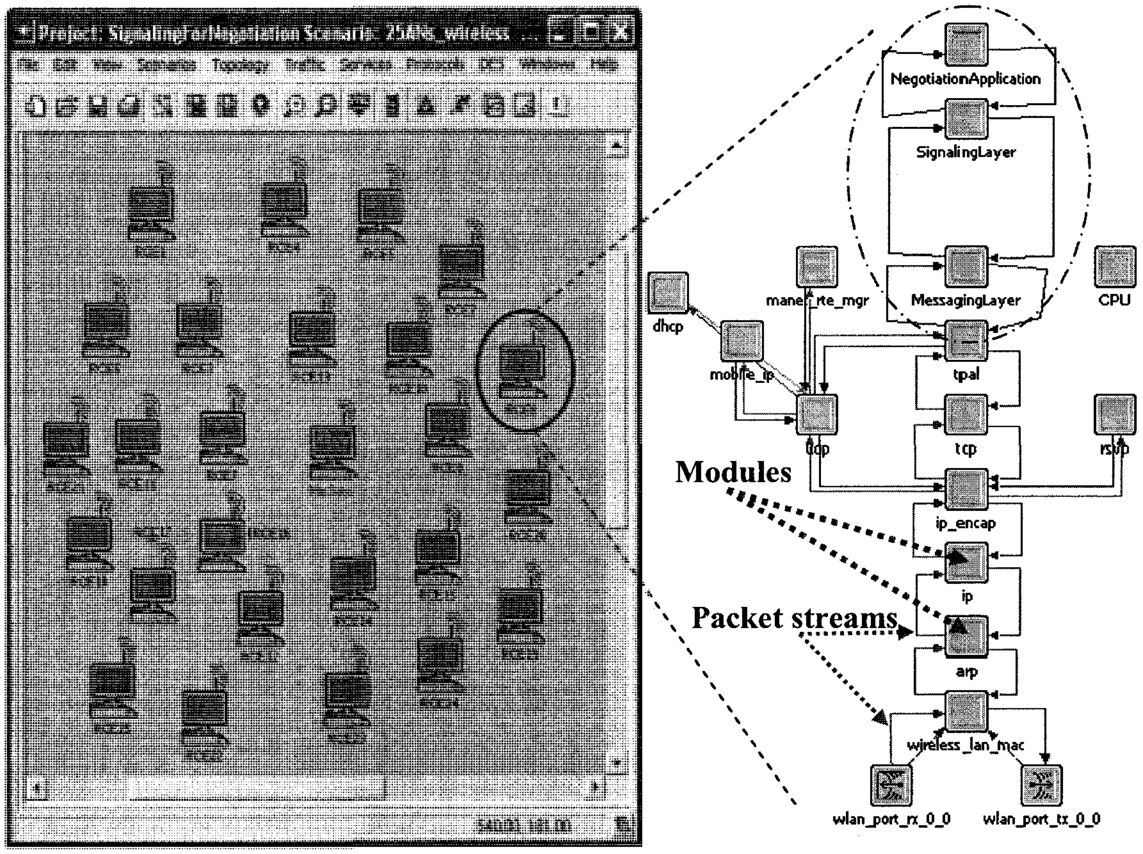


Figure IX.2: Simulation set-up and node architecture

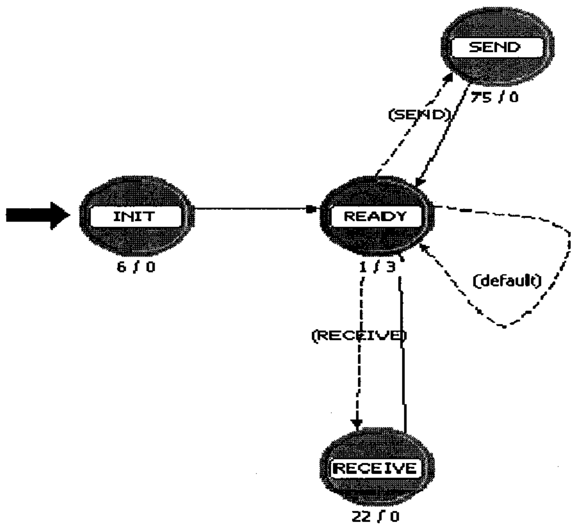


Figure IX.3: Signaling Layer state diagram

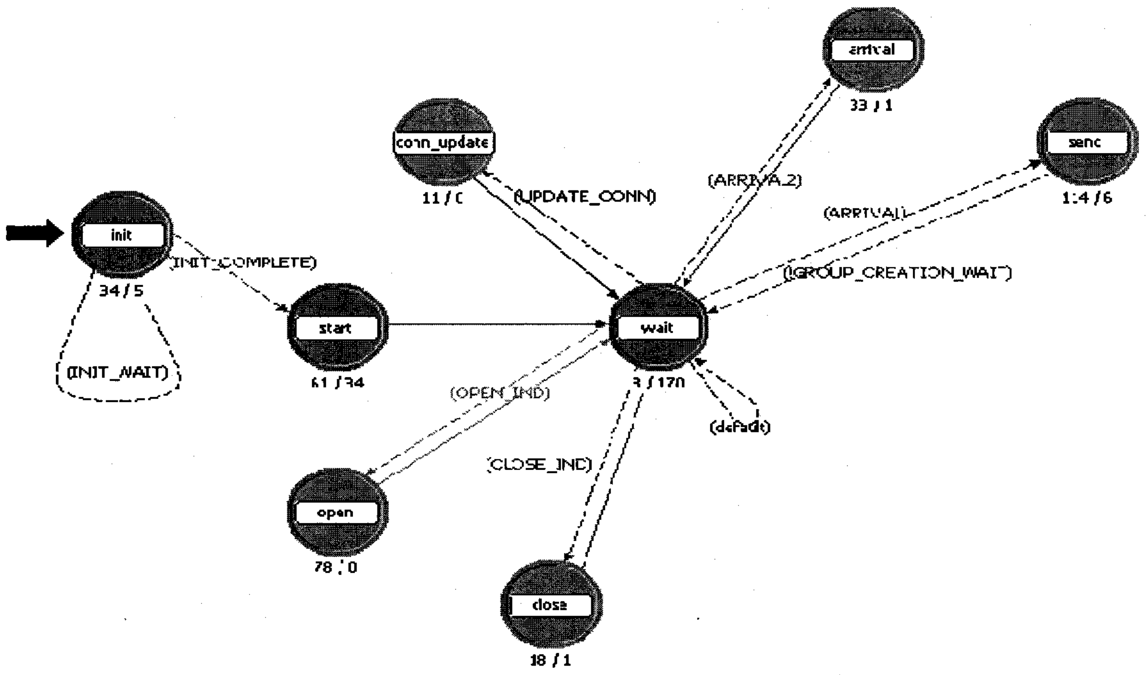


Figure IX.4: Messaging Layer state diagram

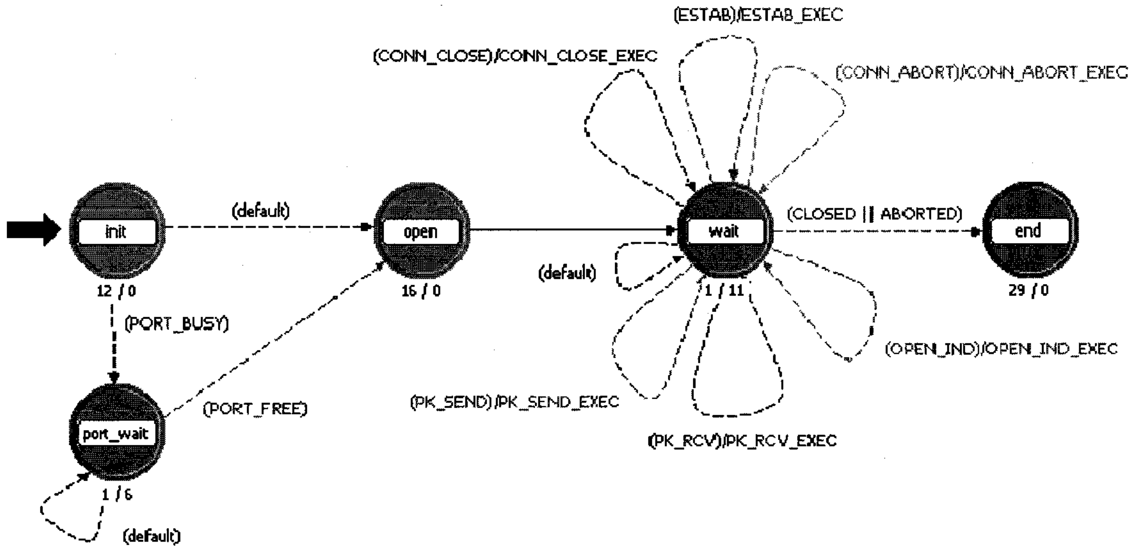


Figure IX.5: Connection Manager state diagram

IX.2.2 Simulation models and scenarios

To separate the validation aspects (and measurements) of agreement negotiation from those of agreement execution, we implemented the two concepts separately. This section presents the simulation models and scenarios used for agreement negotiation and execution.

A. Agreement negotiation

Our simulation model for negotiation is composed of a number of sub-networks, each containing an RCE, the functional entity responsible for registry composition. The mediator is modeled as a separate entity and is part of one of the composing networks (Figure IX.6).

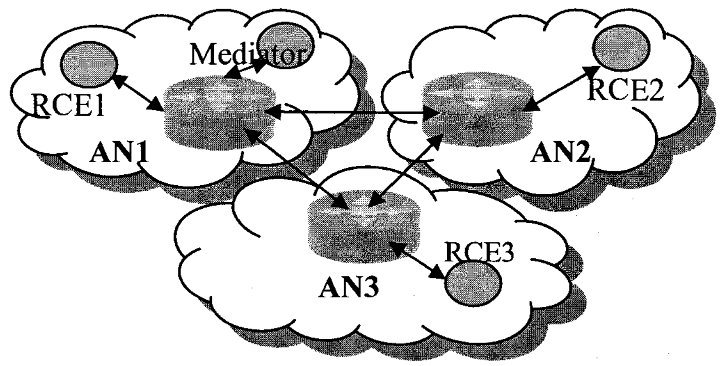


Figure IX.6: A simulation scenario presenting the case where three networks are trying to compose.

In order to evaluate different aspects of agreement negotiation, we implemented two types of scenarios: successful negotiation and negotiation quitting. The case of an entity joining a negotiation is not modeled, because no entity can join a negotiation when it is started. The implementation of the successful negotiation scenario includes the definition of two nodes: Mediator and Participant. They represent the negotiation protocol entities (i.e. mediator and participant respectively). The list of the destination participants is given to the initiator as parameter. In the negotiation quitting scenario, a participant quits after it receives an offer from the mediator.

B. Agreement execution

Figure IX.7 presents the model used for the validation of the agreement execution part of the signaling framework. The composed network hosts a given number of heterogeneous registries, which should be organized into different multicast groups. The only modeled RCE is the one that orchestrates the creation of the registry overlay network.

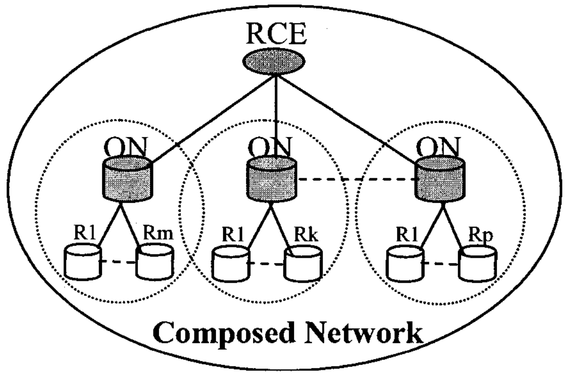


Figure IX.7: The simulation model for agreement execution

We implemented three types of scenarios for agreement execution: RON creation, RON joining and RON quitting. In the RON creation scenario, the RCE creates an RON network between a predefined numbers of registries. This scenario includes the definition of two different nodes: RCE and Registry. RCE is given the list of the multicast groups (i.e. the negotiation output) to create as parameter. RON joining is modeled by a normal registry (i.e. not a virtual registry) joining an already created RON. For RON quitting, we only modeled the case of a virtual registry quitting the network. The case of a normal registry quitting is simple (only one message is needed) and it is transparent to the other entities in the network, except the virtual registry that receives the *Quit* message.

Figure IX.8 and Figure IX.9 present the state diagrams of Registry and RCE respectively. We used the same state diagram to implement Registry nodes, but we used two configuration parameters (i.e. join, quit) to specify which scenario to execute. To execute the RON creation scenario, the two parameters are unset for all of the nodes in the network. To execute the RON join/quit scenario, the join/quit parameter is set for the registry that has to join/quit the network and for the RCE.

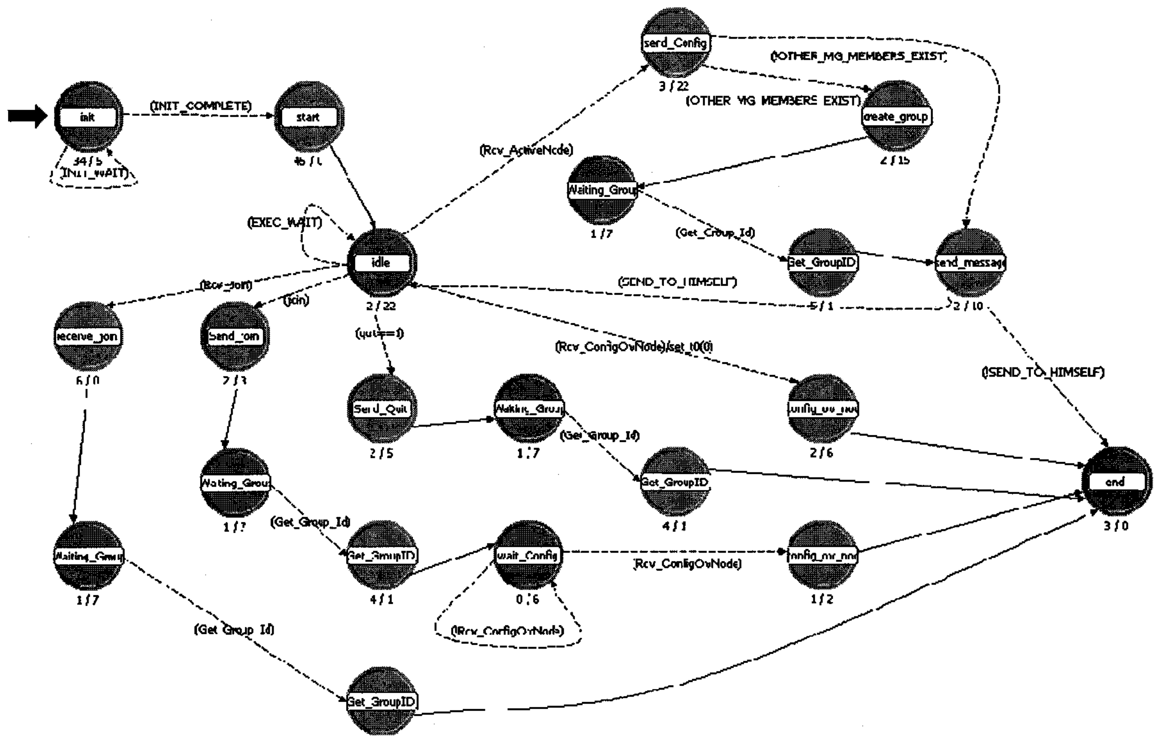


Figure IX.8: Registry state diagram

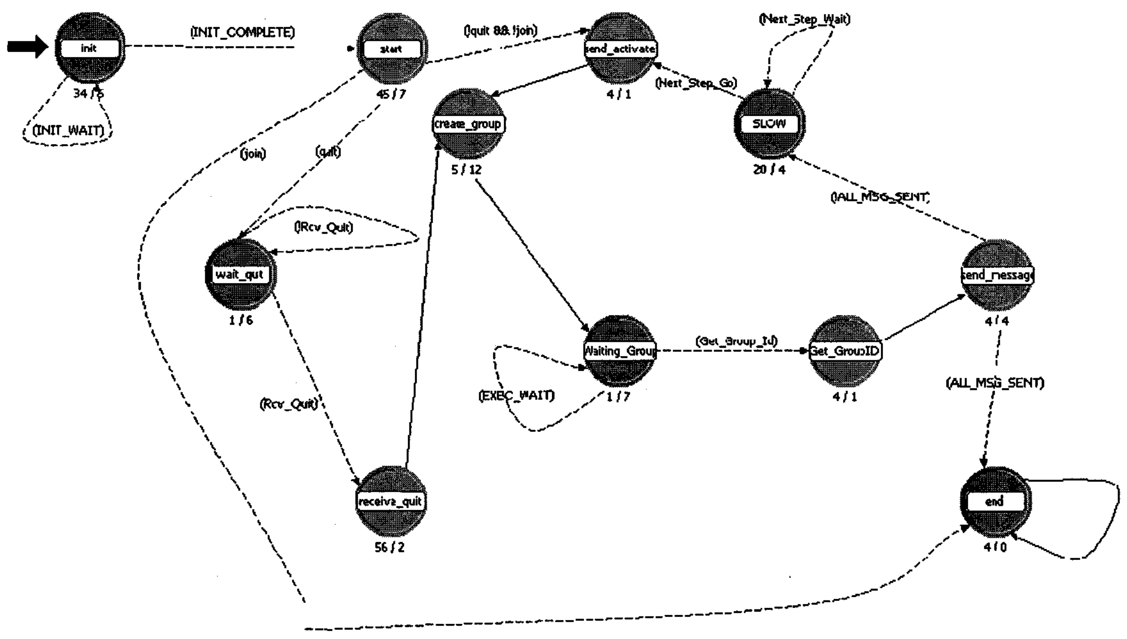


Figure IX.9: RCE state diagram

IX.2.3 Measurements and analysis

This section presents the performance metrics, followed by a set of performance results collected from the simulation. It also includes an analysis of these results.

A. Metrics

To validate our signaling framework, we focused on scalability, regarding the number of negotiating parties (for negotiation) and the number of registries in the composed network (for agreement execution), the negotiation and execution time delay and the network load in terms of number of exchanged messages.

We used the following metrics to measure and evaluate our signaling framework:

- **Network load**: The number of messages transmitted. We measured and/or calculated network load for negotiation, a participant and a mediator to quit the negotiation, RON creation and for a registry to join and quit the RON.
- **Delay**: The delays are calculated in seconds. We measured the following types of delays: total time delay for successful negotiation and for RON creation, average propagation time delay for successful negotiation, RON creation and for a virtual registry to quit the RON. Total time delay for successful negotiation is the total time delay required for the complete negotiation. This is the difference between the time when the initiator creates the first *Initiate* message, and the time when the mediator receives the last *Ack* for the agreement. These measurements do not include the delays for internal processing related to the negotiation (e.g. agreement creation).

  Total time delay for RON creation is the total time delay required for the complete RON creation. This is the difference between the time when the RCE creates the first *ActivateNode* message and the time when the last registry receives a *ConfigOvNode* message. A different *ActivateNode* message –containing different information such as the multicast group members- is created for each virtual registry. Average propagation time delay is the average time delay for a single message to get from the source to the destination.

B. Agreement negotiation

During the simulation, we varied the number of composing networks and calculated the scalability parameters in each case. In general, only a small number (e.g. 2 or 3) of networks will compose at any given time. The case where a high number of networks compose can be seen as a number of successive composition sessions between a limited numbers of networks. This is because not all of the networks will get near each other at the exact same time. Therefore, we have simulated the negotiation process for 2, 3, 5, 10, 15 and 20 entities.

As we can see in Figure IX.10, to achieve a successful negotiation between two entities, 14 messages are needed (7 in case the mediator is co-located with one of the entities). For each extra entity, 7 extra messages are needed. Therefore, the number (nbr) of exchanged messages between n entities in a successful negotiation is nbr = 7n (7n-7 if the mediator is co-located with one of the entities). This is because the messaging layer -at the mediator side- sends a copy of each outgoing message to each of the group members. The formula presented above was validated by the simulation results (Figure IX.11). To reduce the number of exchanged messages (nbr), a multicast solution can be utilized, which will duplicate messages only when needed.

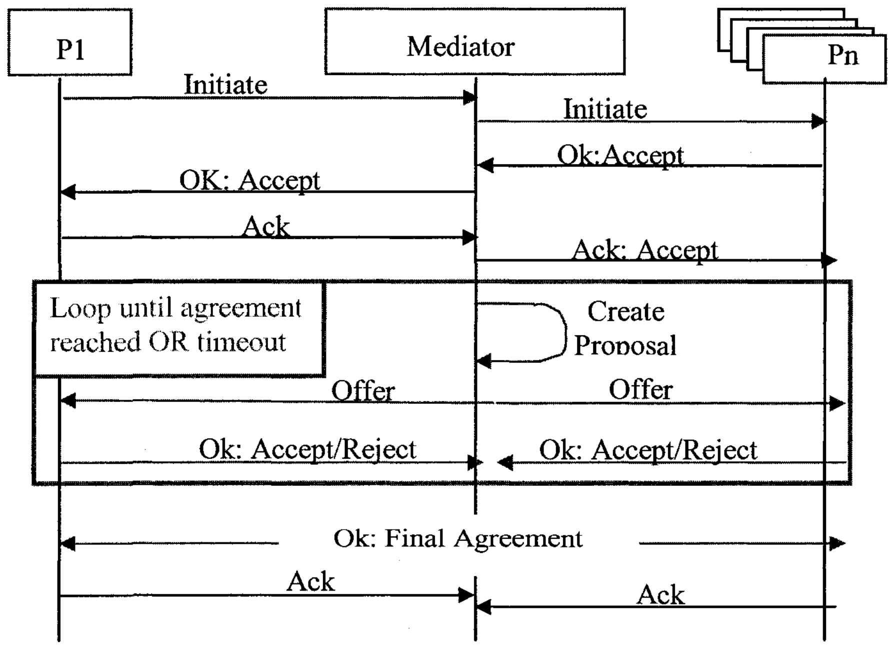


Figure IX.10: Number of messages for a successful negotiation

Figure IX.12 presents the average message propagation time delay in each of the six experiments. It also presents the total time delay for successful negotiation.

The time delay is linearly proportional to the number of composing networks, and it remains barely noticeable by the client. The average time delay varies by less than 0.01 s (0.05 s for total time delay) for each additional five networks. The total time to complete negotiation between 20 ANs is less than 0.16 s.

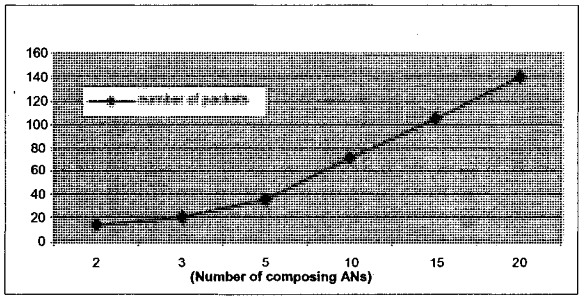


Figure IX.11: Total number of exchanged packets.

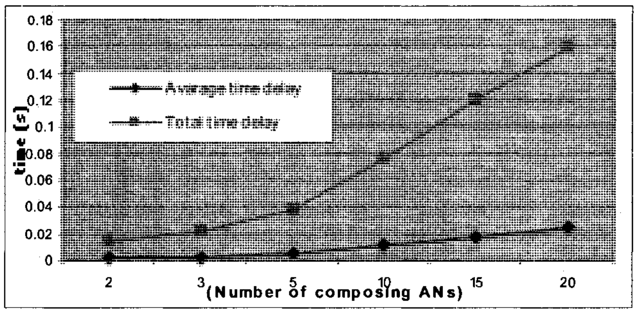


Figure IX.12: Negotiation time delay

The number of messages exchanged to allow an entity to quit the negotiation is always equal to two (i.e. *Bye* followed by *Ok*). The total time delay needed for a participant to quit the negotiation is the time needed to exchange those two messages (about 0.04s in case of 20 entities are negotiating). This time remains negligible compared to the total negotiation time delay and the quitting process does not disturb too much the negotiation process among the remaining parties. However, in case of a mediator quitting the negotiation, the quitting time delay depends on the amount of information to be exchanged between the new and the quitting mediators.

C. Agreement execution

We started with two heterogeneous registries (R1 and R2), where two overlay nodes (OV1 and OV2) are needed. Then, we varied the number of registries from 2 to 35 and the number of used overlay nodes from 2 to 7, and calculated the scalability parameters.
The average message propagation time delay (Figure IX.13) and the total time delay required for the complete RON creation (Figure IX.14) varied depending on the number of registries in the network and the number of overlay nodes used. They are linearly proportional to the number of registries. However, for each number of registries, the average propagation time delay is minimal for a different number of overlay nodes (e.g. for five registries, the minimum average delay is obtained for 3 overlay nodes). The same is applied to the total time delay.
This information can be used to optimally choose the number of registries to be served by the same overlay node, instead of using a single overlay node for all of the registries using the same IPDI as the overlay node, independently of their number.

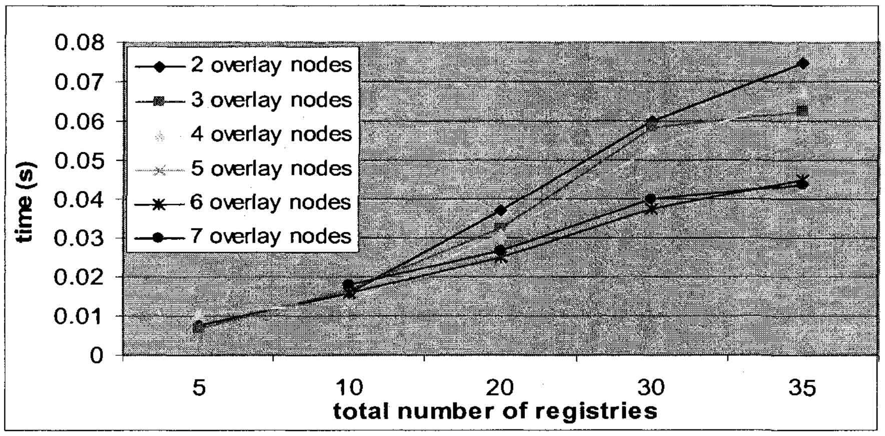


Figure IX.13: Message propagation time delay, for RON creation.

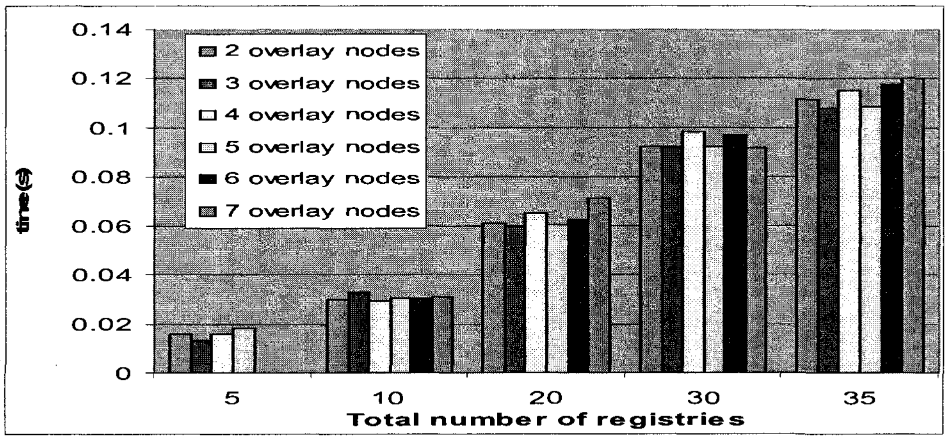


Figure IX.14: RON creation time delay

Figure IX.15 shows the number (nbr_{msg}) of messages exchanged for agreement execution. For each number of registries (nbr_{reg}), nbr_{msg} is the same independently of the number of overlay nodes ($nbr_{msg} = nbr_{reg}$). Indeed, in each case, the RCE sends a single ActivateNode message to each overlay node ($nbr_{ovnodes}$), and each overlay node sends nbr_{same_regs} ConfigOvNode message, where nbr_{same_regs} is the number of registries that have the same IPDI as the overlay node. The message sent by an overlay node to itself (an overlay node is also a registry) is not counted, because it is not sent to the network.

Therefore, $nbr_{msg} = nbr_{ovnodes} + \sum_{i=1}^{ovnodes} (nbr_{same\_regs,i} - 1) = nbr_{reg}$. $nbr_{same\_regs,i}$ can also be reduced by using a multicast solution.

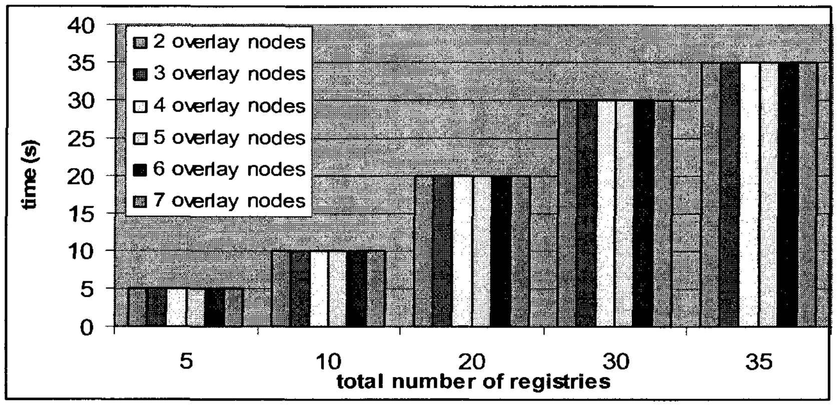


Figure IX.15: Total number of exchanged messages, for RON creation

For a registry to quit RON, only one message is needed (i.e. *Quit* sent to the overlay node). The quitting process is transparent to the other entities in the network (except the overlay node). When an overlay node decides to quit, more processing, messages and time are required. When an overlay node quits, the RCE should replace it, which is equivalent to creating a RON with one overlay node and ($nbr_{same\_regs,i}$-1) registries. $nbr_{same\_regs,i}$ is the number of registries served by the quitting node, including itself. Figure IX.16 shows the average propagation and total time delay for an overlay node to quit, where $nbr_{same\_regs,i}$ varies from 2 to 20, along with the average propagation and total time delay for the creation of the related RON.

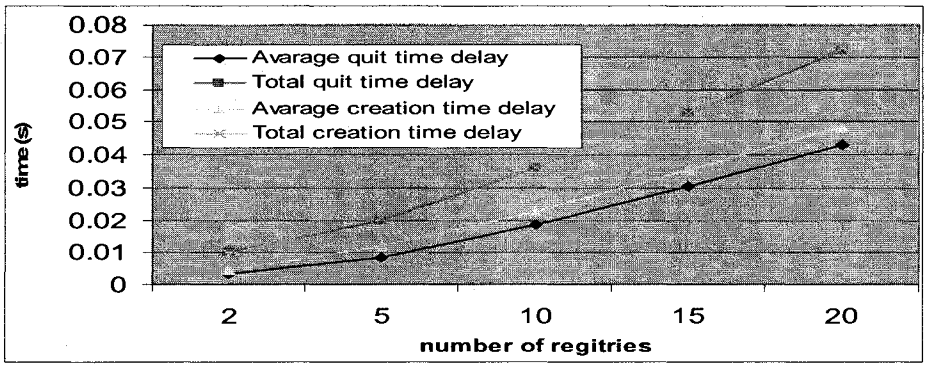


Figure IX.16: Quit time delay

For a registry to join RON, two messages are needed: a *Join* message, followed by a *ConfigOvNode* message sent by the overlay node. The joint time delay (i.e. the difference between the time when the registry decides to join and the time when it receives the *ConfigOvNode* message) is about 0.006 s.

IX.3 Conclusions

In this chapter, we have formally validated the negotiation protocol for registry composition and validated the scalability of the signaling framework via simulations. For formal validation, we described the validation models used and the correctness requirement validated, and presented the validation process and results. For simulations, we presented the simulation models and scenarios, described the performance metrics used and analyzed the simulations' results.

SPIN and PROMELA were found very easy to use for modeling and validating the basic behavior of the negotiation protocol (i.e. no entity leaves the negotiation when it is started). However, some extra effort was needed to model the behavior of the extended protocol (i.e. support nodes departure), and also to model the correctness properties. We had especially to model the timers and the multicasting, two main properties needed by the negotiation protocol models, but not directly supported by PROMELA.

During the validation process, we encountered some problematic situations that we did not think about at design time. These problems were mainly related to unspecified receptions, where an entity sends a request and waits for a specific response to that request, but the responding entity sends a different message before having received that request. The resolution of these problems helped us in improving the proposed negotiation protocol.

OPNET was found very challenging to understand and use, but it provides a rich environment for the simulation of all of the aspects of our framework that we were interested in validating. Through the experiments on simulation scenarios, we have found that the signaling framework is very promising in term of scalability. The generated overhead in term of exchanged messages and in term of time delay remain acceptable, for the number of networks and registries simulated. The simulation results also gave us a hint about how to optimize our architecture for information publication and discovery, depending on the number and characteristics (i.e. the IPDI used) of registries hosted by the composed network.

CHAPTER X: Conclusions and future work

This chapter summarizes the contributions of this thesis. It also discusses the remaining issues and directions for future work.

X.1 Summary of contributions

Ambient Networks are a new networking concept for beyond 3G. Network composition is a core concept of ANs. It enables a level of network cooperation which goes far beyond the static cooperation of today. Network composition provides a uniform, dynamic and scalable cooperation solution between heterogeneous networks, where a network can range from a single node to a full-fledged operator network. ANs can host several registries, such as management information bases and context information bases. When ANs compose, the hosted registries have also to compose. Registry composition is very challenging, because the registries to compose may be heterogeneous. They may be of different types (e.g. centralized, distributed), and be accessible via different types of interfaces (i.e. protocols or APIs). They may include various types of data, represented using different formats (e.g. relational databases vs. object oriented databases). Their content may be of varying granularity levels and it may also be described using different syntaxes and semantics.

In this thesis, we proposed an overall architecture for registry composition in ANs. The major contributions of this thesis are as follows:

- **Identified the issues related to registry composition**: Four types of issues are related to registry composition: interface interworking, data interworking, negotiation and signaling. Interface interworking deals with dynamic intercommunication between registries that are using heterogeneous interfaces. Data interworking deals with how to dynamically overcome data heterogeneity (e.g. data representation, semantics and granularity). Negotiation allows the negotiation of a registry composition agreement. Signaling is required to allow the exchange of the messages related to the negotiation and execution of the composition agreement.
- **Derived requirements for registry composition and reviewed related work**: We derived requirements for the overall composition architecture and for each of the architectural components, and reviewed related work. The overall architecture is made up of three components: an architecture for information publication and discovery after composition, a negotiation architecture and a signaling framework. The architecture for information publication and discovery after composition deals with both interface interworking and data interworking.

  We found that there is no existing overall architecture related to the overall architecture for registry composition. However, there are existing architectural components related to the three registry composition architectural components. Nevertheless, none of them meets all of the related requirements. Existing interface interworking and data interworking related architectures are all based on static configuration and/or gateways. None of the existing negotiation architectures allow a third party to create agreement proposals and none of them deals with registry composition. For signaling, NSIS provides a most promising framework for signaling about registry composition. However it misses some important requirements related to registry composition.

- **Proposed a general architecture for registry composition**: We proposed a new functional entity and a new procedure to orchestrate registry composition. We identified and analyzed the different potential approaches to registry composition (i.e. keep the composing registries as they are, keep the composing registries and create a new one to host shared resources, copy all of the content of the composing registries to a single registry). We also identified and analyzed the potential approaches to enable intercommunication between heterogeneous registries (i.e. protocol deployment on the fly, use gateways).

  For our composition architecture, we choose to keep the composing registries as they are. And to handle intercommunication between heterogeneous registries, we selected to use a standard protocol to enable intercommunication, and vary the inter-protocol translation solution according to the protocol used by each registry (to translate between the standard protocol and the protocol used by the registry).

- **Proposed an architecture for interface interworking**: The architecture is based on overlay P2P networks. It consists of an overlay network that includes for each different interface used by a registry in the composed network, a single overlay node that supports this interface. The overlay nodes also support a common overlay protocol. To the contrary to the existing solutions, our overlay network is created on-the-fly and it handles nodes joining and quitting the network.

- **Proposed an architecture for data interworking**: The data interworking architecture is an extended version of the interface interworking architecture. It reuses the FDBS algorithms and mechanisms to provide data interworking, and provides new procedures to support autonomy and content update. Content update is a sub-issue of data interworking when registries compose in ANs. It deals with the consistency of the registry content after network composition. Indeed, when Ambient Networks compose, some of the registries' content may become obsolete or may need to be updated (e.g. a new service is proposed by the composed network ,by combining two elementary services provided by the composing networks).
- **Proposed a negotiation architecture**: The architecture is made up of four components: negotiating entities, negotiation protocol, template for the composition agreement proposals and description of main negotiation steps. To the contrary to existing negotiation solutions, our architecture allows a third party to create and validate the composition agreement-proposals. The negotiation protocol is not relied on a permanently centralized entity. It handles both voluntary and forced departures of the entities involved in the negotiation (i.e. participants and mediator).
- **Proposed a signaling framework**: The proposed signaling framework is a backward compatible extension of the IETF NSIS framework. The main extensions we have made are the support of point-to-multipoint message delivery and the definition of new signaling application for registry composition. The new signaling application supports both the negotiation of registry composition and the execution of the composition agreement.

- **Proposed implementation architectures and built proof-of-concept prototypes**: We proposed software architectures for protocol deployment on-the-fly and for overlay nodes. We also implemented two proof-of-concept prototypes to show the feasibility of: protocol deployment on-the-fly and the interface interworking architecture.
- **Formally validated the negotiation protocol using SPIN**: We derived correctness requirements for the negotiation protocol and used them to validate the protocol using SPIN. The validation process helped us improve our protocol, by calling our attention to some problematic situations that we did not think about at the design time (e.g. unspecified message receptions).
- **Evaluated performance of the signaling framework using OPNET**: We simulated the signaling framework using OPNET. We built different scenarios and measured different scalability parameters for the framework. We found that the signaling framework is very promising in term of scalability. The generated overhead in term of time delay is barely noticed by the clients. The simulation results also gave some hint about how to optimize our architecture for information publication and discovery.

X.2 Future work

Network composition is a new and broad area that still under-researched. In this thesis, we focused on registry composition and explored many of the related issues. However, much work remains to be done to investigate other aspects of registry composition and network composition. We organize this work into two categories: registry composition related and network composition related.

X.2.1 Registry composition related

We classify the future work related to registry composition in four categories: overall architecture, data interworking architecture, negotiation architecture and security.

- **Overall Architecture**: The characteristics of the composing networks may affect the performance and the efficiency of the composition architecture. In MANET -for instance- nodes can join and leave the network frequently and have heterogeneous capabilities. It would be interesting to make the composition architecture take advantage of those characteristics, in order to enhance its performance. An example is to map the overlay nodes of the interface interworking architecture to the registries with more capabilities, instead of randomly mapping them to any registry that supports the same interface.
- **Negotiation architecture**: Beyond the negotiation architectural components proposed by this thesis, the definition of a negotiation architecture includes the definition of negotiation objects, negotiation strategies and negotiation mechanism [90][91]. The negotiation objects are the different issues over which an agreement should be reached (e.g. price, quantity). The negotiation strategies are the decision functions that each negotiating entity will use for the evaluation of the received proposals. The negotiation mechanism defines the rules of negotiation (e.g. obliging a participant to improve on a previous offer). The study of the negotiation objects, strategies and mechanisms is an interesting item for future work.
- **Security:** Security is an interesting open issue that we have not tackled in this thesis, because it is not critical to the basic registry composition functionality. However, it is

important for the real deployment of the composition architecture. Indeed, the composition should be initialized by an authorized entity and communications between the entities involved in the composition (e.g. negotiating entities) should be secured. The execution of the composition agreement should also be secured (e.g. only the authorized protocols are deployed on-the-fly).

X.2.2 Network composition related

Many issues related to network composition still need to be investigated. An example is how does network composition affects service provisioning and service continuity. Let's consider –for instance- a scenario where a conference is created among a given number of users that belong to separate networks. When these networks compose, the users' connectivity information and the network(s) configuration may have been changed. The challenge is how to insure the continuity of the conference session during and after composition, in a transparent way.

Another example is how are the different functional entities in the ACS composed, when their hosting networks compose. In case of QoS-FEs for instance, an interesting topic for future work is how to provide end-to-end QoS after composition. Reference [92] discusses other items for future work, related to network composition.

References

[1] N. Niebert et al, "Ambient networks: An architecture for communication networks beyond 3G", IEEE Wireless Communications, Volume 11, Issue 2, pages: 14 – 22, April 2004.

[2] "D18-A.4: Annex F AN System Description", FP6-CALL4-027662-AN P2/D18-A.4, Ambient Networks Phase 2 public deliverable, December 2007, http://www.ambient-networks.org/deliverables.html, June 2008.

[3] C. Kappler et al, "Dynamic Network Composition for Beyond 3G Networks: A 3GPP Viewpoint", IEEE Network, Volume: 21, Issue: 1, pages: 47-52, January/February 2007.

[4] "D3-G.1 Design of Composition Framework", FP6-CALL4-027662-AN P2/D3-G.1, Ambient Networks Phase 2 public deliverable, November 2006. http://www.ambient-networks.org/deliverables.html, June 2008.

[5] "D1.4 AN Concepts", IST-2002-507134-AN/WP1-D04, Ambient Networks Phase 1 public deliverable, August 2005, http://www.ambient-networks.org/phase1web/main/deliverables.html, June 2008.

[6] E.K. Lua et al, "A survey and comparison of peer-to-peer overlay network schemes", IEEE Communications Surveys & Tutorials, volume 7, Issue 2, pages: 72-93, Second Quarter 2005.

[7] "UDDI, Universal description, discovery and integration of web services". http://uddi.org/pubs/uddi-v3.0.2-20041019.htm, April 2008.

[8] F. Belqasmi, J. Mattam, R. Glitho, R. Dssouli, F. Khendek, "An Architecture for Composing Registries when Ambient Networks Compose", IEEE Consumer Communications and Networking Conference (CCNC 07), pages: 503-507, January 2007.

[9] F. Belqasmi, R. Glitho, R. Dssouli, "Registry Composition in Ambient Networks", submitted to IEEE Transactions on Computers on June 12, Special issue on autonomic network computing.

[10] F. Belqasmi, R. Glitho, R. Dssouli, "An Overlay Architecture for Information Publication and Discovery after the Composition of Registries in Ambient Networks", IEEE Consumer Communications and Networking Conference (CCNC 07), pages:105 – 109, January 2007.

[11] F. Belqasmi, R. Glitho, R. Dssouli, "Data Inter-working Aspect of Network Cooperation in 4G: the Case of Registry Composition in Ambient Networks", IEEE Broadband Convergence Networks (BCN), pages:1 – 6, April 2008.

[12] F. Belqasmi, R. Dssouli, R. Glitho, "A Negotiation Framework for the Composition of Registries in Ambient Network", IEEE Symposium on Computers and Communications (ISCC2007), pages: 981-987, July 2007.

[13] F. Belqasmi, R. Glitho, R. Dssouli, "Validation of a Negotiation Protocol for Registry Composition in Ambient", Global Information Infrastructure Symposium (GIIS 2007), pages: 106-111, July 2007.

[14] F. Belqasmi, R. Glitho, R. Dssouli, "An IETF NSIS-Based Signaling Framework for Negotiating Registry Composition in Ambient Networks", IEEE International Conference on Networks (ICON 2007), pages: 272-277, November 2007.

[15] G. Camarillo, M.A. Garcia-martin, "The 3G IP Multimedia Subsystem: Merging the Internet and the Cellular Worlds" (book), ISBN 0-470-87156-3, 2004.

[16] J. Rosenberg et al, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[17] "Connecting Ambient Networks-Requirements and Concepts", IST-2002-507134-AN/WP3/D31, Ambient Networks public deliverable, June, 2004, http://www.ambient-networks.org/phase1web/main/deliverables.html, June 2008.

[18] "D2-C.1 Multi-Access & ARI design and Initial Specification", FP6-CALL4-027662-AN P2/D02-C.1, Ambient Networks Phase 2 public deliverable, December 2006. http://www.ambient-networks.org/deliverables.html, June 2008.

[19] "D12-F.1 System Design of SATO & ASI", FP6-CALL4-027662-AN P2/D12-F.1, Ambient Networks Phase 2 deliverable, December 2006, http://www.ambient-networks.org/deliverables.html, June 2008.

[20] D. Zhou et al, "Ambient Network Interfaces and Network Composition", Global Mobile Congress (CIC/IEEE GMC05), October 2005.

[21] F. Belqasmi, R. Glitho, R. Dssouli, “Ambient Network Composition”, Accepted for publication in IEEE Network Magazine, special issue on Composable Context Aware Services, July/August 2008.

[22] R. Campos et al, “Dynamic and Automatic Interworking between Personal Area Networks using Composition”, IEEE 16th International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC 05), Volume 2, pages: 947- 951, September 2005.

[23] M. Johnsson et al, “Network composition”, WWRF#15 (Wireless World Research Forum Meeting 15), Paris, France, December 2005.

[24] “D5-H.1 Final Application Scenarios and Prototype Design”, FP6-CALL4-027662-AN P2/D5-H.1, Ambient Networks Phase 2 public deliverable, December 2006, http://www.ambient-networks.org/deliverables.html, June 2008.

[25] N. Akhtar et al, “GANS: A Signalling Framework for Dynamic Interworking between Heterogeneous Networks”, IEEE 64th Vehicular Technology Conference (VTC 2006), pages: 1-5, September 2006.

[26] R. Hancock et al., “Next Steps in Signaling (NSIS): Framework”, RFC 4080, June 2005.

[27] X. Fu et al, “NSIS: a new extensible IP signaling protocol suite”, IEEE Communications Magazine, Volume 43, Issue 10, Pages:133 – 141, October 2005.

[28] S. Cheshire, and M. Krochmal, "Multicast DNS", draft-cheshire-dnsext-multicastdns-06.txt, Internet Draft, August 2006.

[29] B. Aboba, and D. Thaler, and L. Esibov, ”Linklocal multicast name resolution (LLMNR)”, IETF RFC 4795, January 2007.

[30] I. Roussaki, M. Louta and L. Pechlivanos; “An Efficient Negotiation Model for the Next Generation Electronic Marketplace”, 12th IEEE Mediterranean Electrotechnical Conference (MELECON 2004), Volume 2, Page(s):615 – 618, May 2004

[31] H. Li, J.J. Jeng, and J.Y. Chung; “Commitment-based approach to categorizing, organizing and executing negotiation processes”, IEEE International Conference on E-Commerce (CEC 2003), page(s):12 – 15, June 2003.

[32] M.Patini et al, "A Middleware Architecture for Inter ad-hoc networks Communication", Fourth IEEE International Conference on Web Information Systems Engineering Workshops, pages:201 - 208, December 2003.

[33] S. Microsystem, "Jini community resources: Jini technology architectural overview", January 1999, http://wwws.sun.com/software/jini/whitepapers/architecture.html, April 2008.

[34] I. Stoica et al. "Chord: A scalable peer-to-peer lookup service for internet applications", Conference on applications, technologies, architectures, and protocols for computer communications, pages 149–160, August 2001.

[35] "UpnP: Universal plug and play", http://www.upnp.org/, April 2008.

[36] J. Chang and M.T. Liu, "An Approach to Protocol Complementation for Internetworking", First IEEE International Conference on Systems Integration, Page(s):205 – 211, April 1990

[37] Z.P. Tao, G.V. Bochmann, and R. Dssouli, "An Efficient Method for Protocol Conversion", 4th IEEE International Conference on Computer Communications and Networks (IC3N 95), Pages:40 – 47, September 1995

[38] F. J. Lin and M. T. Liu, "A Formal Model for Protocol Interworking in ISDN". IEEE International Conference on Communications, Volume 1, pages:107 – 113, June 1988

[39] P. Dhar et al, "Network interconnection and protocol conversion-A protocol complementation approach", IEEE International Conference on Computers, Communications and Automation towards the 21st Century, Volume 1, pages:116 - 120 November 1992.

[40] T. Heer et al, "Adapting Distributed Hash Tables for Mobile Ad Hoc Networks", 4th IEEE International Conference on Pervasive Computing and Communications Workshop (PERCOM 2006), pages: 173-178, March 2006.

[41] L. Cheng et al, "Towards Distributed Hash Tables (De)Composition in Ambient Networks", Proceedings of the 17th IFIP/IEEE Distributed Systems: Operations and Management (DSOM), October 2006.

[42] L. Cheng, “Bridging Distributed Hash Tables in Wireless Ad-Hoc Networks”, IEEE Global Telecommunications Conference (GLOBECOM 2007), pages: 5159-5163, November 2007.

[43] C.J. Date, “An Introduction to Database Systems”, eighth edition, ISBN 0-321-19784-4, 2004.

[44] C.W. Chung: “DATAPLEX: An Access to Heterogeneous Distributed Databases”, Communications of the ACM, Volume 33, Issue 1, Pages: 70-80, January 1990.

[45] M.W.Bright, A.R.Hurson and S.Pakzad, “Automated Resolution of Semantic Heterogeneity in Multidatabases”, ACM Transactions on Database Systems, Volume 19, Issue 2, Pages: 212 - 253, June 1994.

[46] N. H.Cohen et al, “iQueue: A Pervasive Data Composition Framework”, IEEE Third International Conference on Mobile DataManagement (MDM’02), pages: 146-153, June 2002.

[47] G. Wiederhold, “Mediators in the Architecture of Future Information Systems”. IEEE Computer, Volume 25, Issue 3, pages: 38-49, Mars 1992.

[48] Y. Arens et al, “Retrieving and Integrating Data from Multiple Information Sources”, International Journal on Intelligent and Cooperative Information Systems, Volume. 2, No. 2, , pages:127–158, June, 1993

[49] J.L. Zhao, A. Segev, A. Chatterjee, "A universal relation approach to federated database management", 11th International Conference on Data Engineering (ICDE'95), pages: 261-270, Mars 1995.

[50] J.C. Chen et al, “Dynamic Service Negotiation Protocol (DSNP) And Wireless Diffserv”, IEEE International Conference on Communications, Volume 2, pages: 1033 –1038, 2002.

[51] C. Bartolini et al, “A Software Framework for Automated Negotiation”, Book "Software Engineering for Multi-Agent Systems III", LNCS 3390, Pages 213-235, Springer-Verkag, 2005

[52] R. Braden et al, “Resource ReSerVation Protocol (RSVP)", RFC 2205, Septembre 1997.

[53] X. Fu, D. Hogrefe and S. Willert, "Implementation and evaluation of the cross-application signaling protocol (CASP)", Proceedings of the 12th IEEE International Conference on Network Protocols (ICNP 2004), Page(s):61 – 71, October 2004.

[54] H.Schulzrinne et al, "Design of CASP- a Technology Independent Lightweight Signaling Protocol", In IPS'03, Salzburg, Austria, February 2003

[55] M. Brunner, "Requirements for Signaling Protocols", RFC 3726, April 2004.

[56] J. Manner, G. Karagiannis, and A. McDonald, "NSLP for Quality-of-Service Signaling", IETF Internet draft, February 2008.

[57] M. Stiemerling et al, "NAT/Firewall NSIS Signaling Layer Protocol (NSLP)", IETF Internet draft, February 2008.

[58] H. Schulzrinne and R. Hancock, "GIST: General Internet Signaling Transport", draft-ietf-nsis-ntlp-13, Internet Draft (work in progress), February 2008.

[59] R. Campos et al, "On the Evaluation of the Extended Generic Internet Signalling Transport Protocol", Proc. of the 15 IST Mobile and Wireless Communications Summit, Myconos, June 2006.

[60] C. Campo et al, "PDP and GSDL: a new service discovery middleware to support spontaneous interactions in pervasive systems", Third IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom 2005 Workshops), Pages:178 – 182, March 2005.

[61] X.L. Lin et al, "Active Peer to Peer", IEEE Sixth International Conference on Networking (ICN '07), pages: 31 – 31, April 2007.

[62] J. Risson and T. Moors, "Survey of Research towards Robust Peer-to-Peer Networks: Search Methods", RFC 4981, September 2007.

[63] Y. Zhu and B. Li, "Overlay Networks with Linear Capacity Constraints", IEEE Transactions on Parallel and Distributed Systems, Volume 19, Issue 2, Page(s):159 – 173, February 2008

[64] J. Han, D. Watson, and F. Jahanian, "Topology aware overlay networks", Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2005). Volume 4, Pages:2554 – 2565, March 2005.

[65] http://www.objectdb.com/database/jdo/manual/chapter7/#7.1, April 2008.

[66] Z. Xu and Y. Hu; “SBARC: A supernode based peer-to-peer file sharing system”, Eighth IEEE International Symposium on Computers and Communication (ISCC 2003). Volume 2, Page(s):1053 – 1058, June/July 2003

[67] V. Lo et al, “Scalable Supernode Selection in Peer-to-Peer Overlay Networks”, Second IEEE International Workshop on Hot Topics in Peer-to-Peer Systems (HotP2P’05), Pages: 18 – 25, July 2005.

[68] C. Fu, R. Glitho, and F. Khendek, “A Novel Session Recovery Mechanism for Cluster-based Signaling Architecture for Conferencing in MANETs”, In the Workshop on Wireless Ad hoc and Sensor Networks (WWASN 2007), June 2007.

[69] B.Y. Zhao, J.D. Kubiatowicz, and A.D. Joseph, “Tapestry: An infrastructure for fault-tolerant wide-area location and routing” Technical Report: CSD-01-1141, University of California and Berkeley, 2001.

[70] Amit P. Sheth and James A. Larson, "Federated Database Systems for Managing Distributed, Heterogeneous and Autonomous Databases", ACM Computing Surveys, Volume 22, Issue 3, pages: 183-236, September 1990.

[71] J.L. Zhao, A. Segev, and A. Chatterjee, “A universal relation approach to federated database management”, Proceedings of the Eleventh International Conference on Data Engineering (ICDE 1995), Pages: 261 – 270, 1995.

[72] N.H.Cohen et al, “Composing pervasive data using iQL”, Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications, pages: 94-104 , August 2002.

[73] Y. Arens et al, "Retrieving and Integrating Data from Multiple Information Sources," International Journal of Intelligent & Cooperative Information Systems, Volume 2, pages: 127-158, 1993

[74] M. Bennicke and P. Langendorfer, “Towards Automatic Negotiation of Privacy Contracts for Internet Services”, 11th IEEE International Conference on Networks (ICON2003), Pages:319 – 324, September/October. 2003.

[75] I. Roussaki, M. Louta, and L. Pechlivanos; “An Efficient Negotiation Model for the Next Generation Electronic Marketplace”, Proceedings of the 12th IEEE Mediterranean Electrotechnical Conference (MELECON 2004), Volume 2, Pages:615 – 618, May 2004

[76] "Specification, design and implementation of the necessary components for the enhancement of an active platform for the validation of the project approach", CONTEXT-WP4-UPC-D4.2-101204Del, IST – 2001 – 38142 – CONTEXT, December 2004, http://context.upc.es/deliverables.htm, June 2008.

[77] K.Psounis, "Active networks: applications, security, safety, and architectures", IEEE Communications Survey, Volume 2, Issue 1, pages: 2-16, First Quarter 1999.

[78] D.L. Tennenhouse et al, "A survey of active network research", IEEE Communications Magazine, Volume 35, Issue 1, Pages:80 – 86, January 1997

[79] J.M. Smith and S.M. Nettles, "Active Networking: One view of the past, present and future", IEEE Transactions on Systems, Man, and Cybernetics, Part C, Volume 34, Issue 1 Pages:4 -18, February 2004.

[80] S.Vrontis et al, "Enabling distributed QoS management utilizing active network technology", IFIP-IEEE International Conference on Network Control and Engineering (Net-Con 2003), pages: 139-151, October 2003.

[81] "Active Networks at UW", University of Washington, http://www.cs.washington.edu/research/networking/ants/, June 2008.

[82] "Composable Active Network Elements Project (CANES)", http://www.cc.gatech.edu/projects/canes/, April 2008.

[83] D.S. Alexander et al., "The SwitchWare Active Network Architecture," IEEE Network Special Issue on Active and Controllable Networks, Volume 12, Issue 3, pages: 29-36, May/June 1998.

[84] K.Jean, N.Vardalach, and A.Galis, "Towards Programmable Context-aware Voice Services", INTELLCOMM 2005, October 2005.

[85] W3C Working Group, "Web Services Architecture", Note 11, February 2004. http://www.w3.org/TR/ws-arch/, April 2008.

[86] W3C web site for SOAP at http://www.w3c.org/TR/SOAP/, April 2008.

[87] "JXTA v2.0 Protocol Specification", http://www.jxta.org/, June 2008.

[88] Gerard J. Holzmann, "DESIGN AND VALIDATION OF COMPUTER PROTOCOLS" (book), Bell Laboratories Murray Hill, New Jersey 07974.

[89] http://www.opnet.com, April 2008.

[90] I. Roussaki, M. Louta and L. Pechlivanos; "An Efficient Negotiation Model for the Next Generation Electronic Marketplace", Proceedings of the 12th IEEE Mediterranean Electrotechnical Conference, (MELECON 2004), Volume 2, Pages:615 – 618, May 2004

[91] M. Beer et al, "Negotiation in Multi-Agent Systems", The Knowledge Engineering Review 14(3): pages: 285-289, 1999.

[92] "D26-G.2: Validated Composition and Compensation Architecture", FP6-CALL4-027662-AN P2/ D26-G.2, Ambient Networks Phase 2 public deliverable, December 2007. http://www.ambient-networks.org/deliverables.html, April 2008.