



THE HONG KONG  
POLYTECHNIC UNIVERSITY

香港理工大學

Pao Yue-kong Library

包玉剛圖書館

---

## Copyright Undertaking

This thesis is protected by copyright, with all rights reserved.

**By reading and using the thesis, the reader understands and agrees to the following terms:**

1. The reader will abide by the rules and legal ordinances governing copyright regarding the use of the thesis.
2. The reader will use the thesis for the purpose of research or private study only and not for distribution or further reproduction or any other purpose.
3. The reader agrees to indemnify and hold the University harmless from and against any loss, damage, cost, liability or expenses arising from copyright infringement or unauthorized usage.

### IMPORTANT

If you have reasons to believe that any materials in this thesis are deemed not suitable to be distributed in this form, or a copyright owner having difficulty with the material being included in our database, please contact [lbsys@polyu.edu.hk](mailto:lbsys@polyu.edu.hk) providing details. The Library will look into your claim and consider taking remedial action upon receipt of the written requests.

**The Hong Kong Polytechnic University**

Department of Computing

**Scalable Service Composition and  
Reconfiguration in  
Pervasive Computing Environments**

By

**JOANNA IZABELA SIEBERT**

A Thesis Submitted in Partial Fulfillment of

the Requirements for the Degree of

Doctor of Philosophy

**June 2012**

## **Certificate of Originality**

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

\_\_\_\_\_  
(Signature)

Joanna Izabela SIEBERT

(Name of Student)

This thesis is dedicated to the memory of my grandmother, Helena Siebert.

## Abstract

Service composition is one of the fundamental mechanisms in pervasive computing. Different services are provided by physical objects embedded with computing devices which are interconnected in an ad hoc manner. However ensuring scalable service composition and composed service re-configuration is a desirable as well as challenging task in large scale and highly dynamic pervasive environments.

In pervasive computing, addressing the issues of scalability and dynamicity of environments are important and challenging. In service composition, service requestors require services instantaneously to accomplish their goals, but they may have no prior knowledge about the available services, as service providers can join and leave environments at runtime. Also, new devices and services can be added, so the scale of a pervasive computing system will grow.

Existing service composition protocols did not adequately address the issues in providing users with scalable service composition at all the times in a localized manner. In this research we address the challenging issues and make the following original contributions in this field.

Firstly, we propose a bottom-up community framework which works as the basis of our research in scalable service composition and reconfiguration in large scale and dynamic environments. The bottom-up community framework consists of a set of service providers along with a suite of protocols and algorithms to collaboratively provide scalable service composition, reconfiguration and heterogeneity supports for mobile users in large scale and dynamic environments.

Secondly, using afore-mentioned bottom-up community, we propose a localized scalable service composition protocol. The bottom-up community nodes exchange messages only with their local neighbors and collaborate on composing only a part of the request. To avoid problem of nodes collaboration with partners that may fail to compose remaining parts of the request, we introduce a technique called Alien-information-based Acknowledging (A-Ack). We also design two additional techniques to optimize the performance with respect to time and message cost. This

---

approach guarantees arriving to the solution by collaboration of nodes without global information and maximizes quality of the solution by setting optimal locality area size for a given deadline.

Third, based on the bottom-up community, we develop a composed service reconfiguration mechanism, named LASER. The proposed approach is suitable for applications deployed in environments which are large scale and highly dynamic. In LASER, no coordinator collects global information, nodes and links between them are mapped in subsequent stages and nodes construct only local part of the solution, while together achieving global goal. To avoid problem of hidden abort, we develop a technique called gradual reveal. We also design two additional techniques to optimize the performance with respect to time and message cost. Comparing with existing decentralized and pull-based centralized approach, the proposed localized algorithm is delay-constrained and message-efficient.

Finally, in order to cope with the problem of heterogeneity in service composition we study how to provide service composition across different environments supported by different service management systems. We propose Universal Adaptor, a novel approach towards supporting multi-protocol service discovery in pervasive computing. UA consists of Universal Adaptor Primitives and Universal Adaptor Mapping. It provides mapping from UAP to protocol specific primitives. The client makes use of UAP to discover and access services. UAM performs mapping from UAP to service specific SDP primitives. From the point of view of the service side, Universal Adaptor is a tailored component that uses native SDP and performs service discovery on behalf of the client. Our contribution to the interoperability of service composition is providing solution that in a lightweight manner bridges not only all existing but future service discovery systems, including standard ones, as well as service discovery mechanisms that support multiple protocols within the domain. Our implementation has shown that Universal Adaptor is a simple and flexible solution. It is easy in the sense of writing the code, implementing in diverse infrastructures and using by client.

# Publications

## Book Chapters

1. **Joanna Izabela Siebert**, Jiannong Cao, "*Scalable Service Composition in Pervasive Computing Environments*," In Scalable Computing and Communications: Theory and Practice, S.U. Khan, L.Wang, A.Y. Zomaya (Eds.), John Wiley & Sons, Ltd., 2013.
2. Jiannong Cao, **Joanna Izabela Siebert**, "*Service Management in Pervasive Computing Environments*," In Pervasive Computing and Networking, M.S. Obaidat, M.Denko, I.Woungang (Eds.), John Wiley & Sons, Ltd., 2011.

## Journal Papers

1. **Joanna Izabela Siebert**, Jiannong Cao, Steven Lai, Peng Guo, and Weiping Zhu, "*LASEC: A Localized Approach to Service Composition in Pervasive Computing Environments*," submitted to IEEE Transactions on Parallel and Distributed Systems.

## Conference Papers

1. **Joanna Izabela Siebert**, Jiannong Cao, Long Cheng, Edwin Wei, Canfeng Chen, and Jian Ma, "*Decentralized Service Composition in Pervasive Computing Environments*," In Proceedings of the 6th International Wireless Communications and Mobile Computing Conference (IWCMC '10), pp. 1258-1262, June 28-July 02, 2010. Caen, France.
2. Long Cheng, Jiannong Cao, Canfeng Chen, Hongyang Chen, Jian Ma, **Joanna Izabela Siebert**, "*Cooperative Contention-Based Forwarding for Wireless Sensor Networks*," In Proceedings of the 6th International Wireless Communications and Mobile Computing Conference (IWCMC '10), pp. 1136-1140, June 28-July 02, 2010. Caen, France.

- 
3. **Joanna Izabela Siebert**, Jiannong Cao, "Service Composition in Pervasive Computing Environments: a Survey," In Proceedings of 3rd International Interdisciplinary Technical Conference of Young Scientists, May 19-21, 2010. Poznan, Poland.
  4. **Joanna Izabela Siebert**, Jiannong Cao, Yu Zhou, Miaomiao Wang, and Vaskar Raychoudhury, "Universal Adaptor: A Novel Approach to Supporting Multi-protocol Service Discovery in Pervasive Computing," In Proceedings of International Conference on Embedded and Ubiquitous Computing (EUC07), pp. 683-693, December, 2007, Taipei, Taiwan.
  5. Miaomiao Wang, Jiannong Cao, **Joanna Izabela Siebert**, Vaskar Raychoudhury, and Jing Li, "Ubiquitous Intelligent Object: Modeling and Applications," In Proceedings of 3rd International Conference on Semantics, Knowledge and Grid (SKG'07), pp. 236-241, October 29-31, 2007. Xian, China.
  6. Yu Zhou, Jiannong Cao, Vaskar Raychoudhury, **Joanna Izabela Siebert**, and Jian Lu, "A Middleware Support for Agent-Based Application Mobility in Pervasive Environments," In Proceedings of the 27th International Conference on Distributed Computing Systems Workshops (ICDCSW07), June 25-29, 2007, Toronto, Ontario, Canada.



## Acknowledgements

First, I would like to thank my supervisor, Professor Jiannong Cao, for his rigorous supervision of my research. I thank him for his guidance, patience and encouragement during my PhD study. He unremittingly trained me to be a good researcher. He taught me how to find research issues and how to solve problems. Time and again, he showed me how to express my ideas and write research papers. His knowledge, vision, passion, professional spirit and attitude towards the research deeply inspired me. Without his help and support, this body of work would not have been possible. What I have learned and experienced during the time I spent in his research group will always help and encourage me in the future.

Second, I thank Dr. Peng Guo and Dr. Zhu Weiping for their great help and wise advices during my PhD study. They made a lot of contributions to this research work. Also, I would like to convey my thanks to Dr. Lai Yi, Dr. Ma Chao, Dr. Kong Junjun, Dr. Cheng Long, and Dr. Feng Yuhong, for their kind help in my research and project works. Furthermore, I would like to thank all my teachers from whom I have learned so much during my long journey in academia. They are Prof. Henry Chan, Prof. Alvin Chan and Prof. Hong-va Leong at the Hong Kong Polytechnic University, and many others. In addition, I also wish to thank Yao Gang, Dr. Fan Xiaopeng, Dr. Wu Weigang and Dr. Wang Miaomiao who have rendered their help to my research work in one way or another and shared with me the pleasure of study at the Hong Kong Polytechnic University.

Last but not least, I thank my family for their continuous love, support, trust, and encouragement for me through the entire period of my study. Without them, none of this would have been possible.



# Table of Contents

<b>Abstract</b>	<b>i</b>
<b>Publications</b>	<b>iii</b>
<b>Acknowledgements</b>	<b>v</b>
<b>Table of Contents</b>	<b>vii</b>
<b>List of Figures</b>	<b>xi</b>
<b>List of Tables</b>	<b>xv</b>
<b>1 Introduction</b>	<b>1</b>
1.1 UIO-based Pervasive Computing Environments . . . . .	1
1.2 Service Composition in UIO Framework . . . . .	4
1.3 Contributions of the Thesis . . . . .	7
1.4 Outline of the Thesis . . . . .	10
<b>2 Background and Literature Review</b>	<b>13</b>
2.1 General Components of a Service Composition System . . . . .	13
2.2 Scalability in Pervasive Computing Environments . . . . .	17
2.3 Approaches And Techniques For Scalable Service Composition in PvCE . . . . .	20
2.3.1 Approaches to Achieve Scalability . . . . .	20
2.3.2 Techniques for Achieving Scalability . . . . .	25

2.4	Virtual Network Embedding . . . . .	28
2.5	Localized Algorithms . . . . .	29
2.6	Composed Service Reconfiguration Approaches . . . . .	30
2.6.1	System Consistency in Dynamic Reconfiguration of Distributed Systems	30
2.6.2	Recovery in Discovery Systems . . . . .	30
2.6.3	Workflow Adaptation Approaches . . . . .	31
<b>3</b>	<b>Bottom-up Community: A Framework for Scalable Service Composition and Reconfiguration</b>	<b>33</b>
3.1	Generic Framework Structure . . . . .	33
3.2	Bottom-up Community Creation . . . . .	36
3.2.1	General Idea . . . . .	36
3.2.2	Challenging Issues . . . . .	38
3.2.3	Informed Acknowledge Technique . . . . .	40
3.3	Scalable Service Composition and Reconfiguration using Bottom-up Community	40
3.4	Summary . . . . .	46
<b>4</b>	<b>LASEC: Localized Approach to Service Composition</b>	<b>47</b>
4.1	System Model . . . . .	48
4.2	Problem Specification . . . . .	51
4.2.1	Problem Illustration . . . . .	51
4.2.2	Problem Description . . . . .	54
4.2.3	Problem Formulation . . . . .	56
4.2.4	Complexity Analysis . . . . .	56
4.3	Localized Service Composition Algorithm . . . . .	57
4.3.1	Description of the Algorithm . . . . .	58
4.3.2	Discussion of the Algorithm . . . . .	63

4.3.3	Analysis of the Algorithm . . . . .	64
4.4	Performance Evaluation . . . . .	68
4.4.1	Simulation . . . . .	68
4.4.2	Performance Analysis . . . . .	69
4.4.3	Prototype Implementation . . . . .	71
4.5	Summary . . . . .	75
<b>5</b>	<b>LASER: Localized Approach to Composed Service Reconfiguration</b>	<b>77</b>
5.1	System Model and Preliminaries . . . . .	78
5.1.1	Service Monitors . . . . .	78
5.1.2	Recovery and Reconfiguration . . . . .	80
5.1.3	Consistency Requirements . . . . .	80
5.1.4	Protocol Preliminaries . . . . .	82
5.2	The Proposed LASER Algorithm . . . . .	86
5.2.1	Service Monitors . . . . .	86
5.2.2	Localized Commit Protocol . . . . .	88
5.3	Performance Evaluation . . . . .	90
5.4	Summary . . . . .	94
<b>6</b>	<b>LASEH: Localized Approach to Service Heterogeneity</b>	<b>97</b>
6.1	Background . . . . .	99
6.2	System Model . . . . .	102
6.3	Universal Adaptor Approach . . . . .	103
6.3.1	Universal Adaptor Primitives (UAP) . . . . .	103
6.3.2	Universal Adaptor Mapping (UAM) . . . . .	105
6.4	Performance Evaluation . . . . .	107
6.4.1	Prototype Implementation and Experience . . . . .	107

Table of Contents

---

6.4.2	Simulation . . . . .	109
6.5	Summary . . . . .	111
<b>7</b>	<b>Conclusions and Future Works</b>	<b>113</b>
7.1	Conclusions . . . . .	113
7.2	Future Works . . . . .	115
	<b>Bibliography</b>	<b>119</b>

# List of Figures

1.1	Service composition application. For specified required functions multiple devices in the environment are available. . . . .	5
1.2	Block Diagram for Research Contributions. . . . .	8
2.1	Service Composition Framework . . . . .	14
2.2	Centralized approach. . . . .	21
2.3	Decentralized approach. . . . .	21
2.4	Hybrid approach. . . . .	21
3.1	Bottom-up Community Framework. . . . .	35
3.2	Composed service request as a jigsaw puzzle. . . . .	36
3.3	Forming service sections. . . . .	37
3.4	Available services jigsaw puzzle pieces. . . . .	37
3.5	Formed sections: A-C, B-D, F-G. . . . .	37
3.6	Local messages sent between parents and children. . . . .	38
3.7	False edge concept illustration. . . . .	39
3.8	Uncertain edges concept illustration. . . . .	40
3.9	Informed acknowledge messages dependency. . . . .	41
3.10	Service providers in the environment. . . . .	42
3.11	An example execution of the localized algorithm with informed acknowledge. . .	44
3.12	Local operation dependencies. . . . .	45

3.13	Decreasing local response time with Accelerated Acknowledge. . . . .	46
4.1	Localized service composition system. Devices in the communication layer explore localized neighbourhood to provide functions specified in the request layer. . . . .	48
4.2	Service description. . . . .	49
4.3	Requested composite service. . . . .	49
4.4	Parent/child relationship in the request. Parents unfold local request and children fold received local request. . . . .	50
4.5	Definition of neighborhood. Parameter $k$ corresponds to the maximum distance of neighbours from the vertex. . . . .	52
4.6	Service Composition Area. Each service provider searches within $k$ -hops for service which type matches the type on the other end of the edge specified in the request. . . . .	53
4.7	Example service request. ID of nodes specifies requested functionalities of atomic services. Links specify relationships. . . . .	54
4.8	Service environment with possible 1-neighborhood service graphs. . . . .	54
4.9	Service environment with possible $k$ -neighborhood service graphs. . . . .	55
4.10	The unconverged case in localized composition. . . . .	58
4.11	The basic idea of LASEC. . . . .	60
4.13	Impact of network size on message overhead. . . . .	72
4.14	Impact of network size on response delay. . . . .	72
4.15	Impact of request complexity on message overhead. . . . .	72
4.16	Impact of request complexity on response delay. . . . .	72
4.17	Impact of service density on message overhead. . . . .	72
4.18	Impact of service density on response delay. . . . .	72



4.18	Composition locality. Atomic services in the composed service should be close to each other. . . . .	73
4.19	Prototype implementation. . . . .	73
4.20	System architecture. . . . .	74
4.21	Experimental results for message cost. . . . .	75
4.22	Experimental results for message delay. . . . .	75
5.1	Localized service recovery and reconfiguration system. . . . .	79
5.2	Monitoring relationship in the request. . . . .	83
5.3	Consistent service reconfiguration. . . . .	84
5.4	Example of hidden abort. . . . .	85
5.5	Forwarding Node Selection Algorithm. . . . .	87
5.6	Localized service reconfiguration algorithm LASER. . . . .	89
5.7	Dynamicity - one of the services becomes unavailable and it must be replaced. . .	93
5.8	Coping with request dynamicity. . . . .	93
5.9	Simulation results for composed service recovery. . . . .	93
6.1	Universal Adaptor system model. . . . .	104
6.2	Universal Adaptor Primitives. . . . .	105
6.3	Universal Adaptor Mapping. . . . .	106
6.4	Simulation results for success rate and quality. . . . .	110



# List of Tables

3.1	Achieving Global Behavior by Local Operations . . . . .	38
4.1	Definitions of some notations. . . . .	59
4.2	Definitions of messages. . . . .	59
4.3	Simulation Parameters . . . . .	69
5.1	Message Types in Forwarding Node Selection . . . . .	86
5.2	Data Structures in Forwarding Node Selection . . . . .	88
5.3	Message Types in LASER . . . . .	89
5.4	Data Structures in LASER . . . . .	90
5.5	Simulation Parameters . . . . .	91
6.1	Function of UAM components . . . . .	107
6.2	Discovery overhead of UA . . . . .	108
6.3	Access overhead of UA . . . . .	109



# Chapter 1

## Introduction

Pervasive computing is a new computing paradigm following distributed and mobile computing paradigms. It aims to change the way in which users use computing and related technologies. It sees personal computers as limitation and envisions computation and communication capabilities being immersed around users, being always available wherever and whenever they may need them. Pervasive computing enables things that surround users to serve them and fulfill their needs in invisible manner. It studies how embedding computation capabilities into physical environments can help people in different tasks they perform in their lives. Since Mark Weiser envisioned the computer for the twenty-first century[120], we have witnessed the rapid integration of the cyber and physical worlds. Pervasive computing emerged from distributed and mobile computing and shares many research themes in common with these fields [102]. However, new problems are encountered and the solutions of many previously-encountered problems become more complex.

### 1.1 UIO-based Pervasive Computing Environments

The main objective of pervasive computing research is to build pervasive computing environment (PvCE). Examples of PvCEs are unlimited: airports, hospitals, large enterprises, intelligent homes and offices, shopping malls, conference venues, battlefields, streets, movie theaters, restaurants, and other smart spaces. What makes them pervasive is embedding computers in these physical environments in the form of smart, connected and collaborating devices. Therefore, PvCE is the physical space surrounding users and transformed into a pervasive computing platform by aug-

menting in it objects with computation and communication capabilities. Advances in the technology in nano-scale semiconductor, wireless communications and microelectro-mechanical systems (MEMS) are making physical devices smaller in size and more and more intelligent in terms of sensing, computing and communication capabilities. In the foreseeable future, physical objects augmented with such capabilities will be deployed everywhere into surrounding environments. They can closely integrate the cyber-physical worlds and autonomously interact with each other to achieve desirable system behaviors. Devices with sensing, memory, computing and communication capabilities are immersed into our living environments. For example, objects at home can serve as interfaces for controlling robots, mirrors can help with shopping and wearable computing enables trend of computing on the body. The widespread deployment of pervasive computing devices is transforming the physical world into computing medium.

Pervasive computing environments bring new challenges to the application development process. PvCEs are highly flexible and dynamic in nature. New devices can leave and join environments in any moment. Moreover, abundant number of services is provided in environments saturated with pervasive computing devices.

Scale of pervasive computing environments varies, from small personal networks, through smart spaces and smart cities to all Earth connected into Internet of Things. Examples of smart spaces include, smart home [25], [26], smart meeting room, smart office, smart classroom, or smart museums. Smart cities [105] are the system of many pervasive applications working together for the benefit of citizens. Example of such applications may be browsing for physical objects in a real world similar to browsing the internet [116], [124]. Development of Internet of Things technology contributes development of Smart cities. Endless applications are being developed that utilize large number or resource constrained devices, providing services such as temperature, pressure, vibrations, and energy measurements.

As the range of devices embedded in PvCE widens, heterogeneity in computing systems in-

creases as well. This is due to the high heterogeneity of integrated technologies in terms of networks, devices and software infrastructures. However, for the benefit of users in PvCE, heterogeneous devices will be required to interact seamlessly, hiding differences in hardware and software capabilities.

Many PvCEs are very dynamic in nature. Due to the mobility of users as well as devices the view of PvCE changes constantly. Failure of devices increases dynamicity of the environments.

It is overwhelming for the user to manage growing number of devices, configure diverse applications, and dynamically find the available computing services in such pervasive computing environments. Programmers require efficient middleware support to bridge the gap between complexity of the underlying computing environment and the high level requirement.

Most existing middleware systems for pervasive computing are based on a top-down, centralized model where a central controlling entity facilitates and coordinates the functional operations of component devices deployed in the system. They need to be improved because they are designed with the view that the component devices are passive objects and need to be centrally managed. Given the increasing number of embedded devices, centralized middleware architecture compromises the scalability of the middleware. More importantly, the increasing capability of the component devices can be used to alleviate the role and load of the central controlling entity and facilitate the design of distributed and localized algorithms for achieving middleware functions.

We propose an alternative bottom-up, decentralized approach to designing and programming pervasive computing middleware functions based on ubiquitous interacting objects (UIOs), which are smart devices augmented with various processing capabilities. Many middleware functions, such as service discovery, service composition, context derivation and inconsistency checking, can be performed through decentralized interaction and collaboration of the UIOs. Building these functions into individual UIOs will facilitate the deployment of pervasive computing applications because the UIOs can autonomously discover and coordinate with each other to carry out the mid-

deware and application-specific functions, with the minimum support from the central controller. Also, without relying heavily on central controllers, the proposed approach makes the middleware system more scalable and flexible.

Designing UIO-based middleware functions using a bottom-up, decentralized approach faces many challenging problems. We focus on service composition support in UIO based middleware and we identify and address the fundamental issues, and seek for effective and efficient solutions.

## **1.2 Service Composition in UIO Framework**

In this section, we first describe the characteristics of service composition in UIO based PvCEs. Then, we identify and discuss the requirements of solutions.

We define the terms - service and service composition in the context of pervasive systems.

*Service:* A functionality of a computational entity whose execution satisfies the requestor's requirement.

*Service Composition:* A process of identifying and combining component functionalities to compose a higher level functionality and provide means to perform the requested functionality.

At the conceptual level, a UIO model is defined to represent the essential features of UIO, which are extracted from the physical object itself with its inherent abilities and additional intelligent abilities provided by the associated pervasive computing devices. These capabilities are implemented as services and exposed to the potential requestors. Through services users can interact with PvCE. UIOs can provide hardware or software services. Examples of software services can be weather forecast, stock quotes, and language translation. Services provided by the hardware UIOs can be for example controlling a temperature in the room or printing a document. UIOs that expose their functionalities in form of services are called service providers, and UIOs that need some functionality are called service requestors. Requestors of services can be users, other objects or middleware that facilitates interaction in PvCE.

Service composition mechanisms are necessary when no single service provider can provide



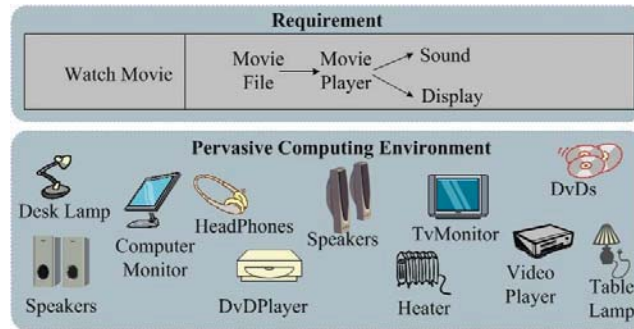


Figure 1.1: Service composition application. For specified required functions multiple devices in the environment are available.

a requested service and several service providers have to collaborate. Fig. 1.1 shows service composition application in pervasive computing environment. Consider that user specifies the requirement of watching the movie, for which following services must be satisfied: file with the movie, movie player, device to output sound of the movie and device to output its display. During runtime usually multiple instances of requested functionalities can be found. And apart from the services specified in the requirement, other services exist in the environment as well. According to requirements, we need to select one instance for each service type.

The motivation behind the research on service composition in PvCEs derives from the huge gap between the high-level requirements from pervasive computing applications and the complexity of PvCEs. To handle this issue, middleware systems with service composition functions have been proposed. Most existing middleware systems for smart spaces are based on a top-down, centralized model where a central controlling entity facilitates and coordinates the functional operations of component devices deployed in the system. They depend on the existence of central directories or registries that maintain the global service information. For example, in Aura [111], the Environment Manager (EM) plays a central role in maintaining and intermediating services provided by various suppliers. In order to compose a complex service, the user requirement will be propagated down to the Environment Management level, and the EM interprets the requirements and issues messages to the service suppliers to be interconnected.

However UIO-based PvCEs are much more heterogeneous, large scale and dynamic than tra-

ditional environments. Service composition for large scale PvCE can be designed and performed through the localized interaction and coordination of UIOs. However, how to facilitate the UIO interaction and coordination in order to achieve global system behaviors is challenging and gives rise to many important requirements that we will address in this thesis:

*Bottom-up UIO collaboration support* Most of the existing PvC middleware follow a centralized coordination. However, in UIO based PvCE, there is very large number of resource constrained UIOs and no guarantee of having a central entity capable of managing all other devices. To achieve desirable system behaviors, autonomous interaction of UIOs is required. This can enable to coordinate the UIOs and make them serve people in a less obtrusive manner, by supporting discovery of services matching composed service request. What is required is an efficient decentralized and bottom-up approach of system design which will work through spontaneous interoperation and decoupled coordination among participating UIOs. Without requiring a centralized control structure dictating how individual UIOs should find and talk to each other, we need to design the rules that can be used by the UIOs to interact with each other to achieve basic coordination objectives such as discovering neighbors, forming composed services and reconfiguring them.

*Handling changes in environments* - PvC applications aim to assist people with computation support everywhere and all the time. This often requires support for user and application migration across PvCEs. Also, the PvCE is extremely dynamic in nature and UIOs frequently join and leave the environment. There is a need for design of mechanisms to allow a UIO decide which other UIOs they need to interact with, and discover them in real-time. Each UIO must keep track of other available and nearby UIOs with which it can interact. In a dynamic ad hoc environment and with resource-constrained UIOs, this task is a non-trivial one.

*Heterogeneity support* - Heterogeneity in computing systems will not disappear in the future, but instead will increase as the range of computing devices widens. Especially in UIO based PvCE heterogeneity is particularly high. Services may be hosted on range of platforms, from resource-

rich fixed machines to wireless resource-constrained devices, to any physical object turned into UIO. Because of the platform heterogeneity, various service composition mechanisms, including service discovery, are available. Heterogeneous UIOs will be required to interact seamlessly, despite wide differences in hardware and software capabilities. This will require new mechanisms to enable heterogeneous UIOs to freely discover and collaborate with each other. However, developing such a mechanism is difficult, given lack of centralized control in PvCE.

In this work, we propose an approach that can address the above requirements and can support scalable service composition and reconfiguration in UIO based PvCEs.

### 1.3 Contributions of the Thesis

The objective of this research is to study the scalability related problems in service composition in large scale dynamic pervasive computing environments and to provide innovative and cost-efficient solutions to address those problems. The dissertation makes several research contributions to achieve the afore-mentioned objective, which consists of the development of a framework for scalable service composition. Based on that framework we have developed two mechanisms to support localized service composition and to provide service composition and reconfiguration for pervasive environments. Figure 1.2 shows how all of our works seamlessly fit into a scalable service composition and reconfiguration framework and present a coherent picture of our complete research. Here, we briefly describe our research contributions. A more detailed description of the framework will follow in Chapter 3.

We study the problem of a localized approach to service composition. In this part of our work we focus on how to select the proper services for the given service composition. Given a set of nodes each providing certain services, the nodes are interconnected and can communicate with each other via wireless network, assume a user specified requested composite service with specified types of requested services and relationships between them. Our problem is to design a localized protocol for the nodes to collaborate in the composition process such that the communication cost

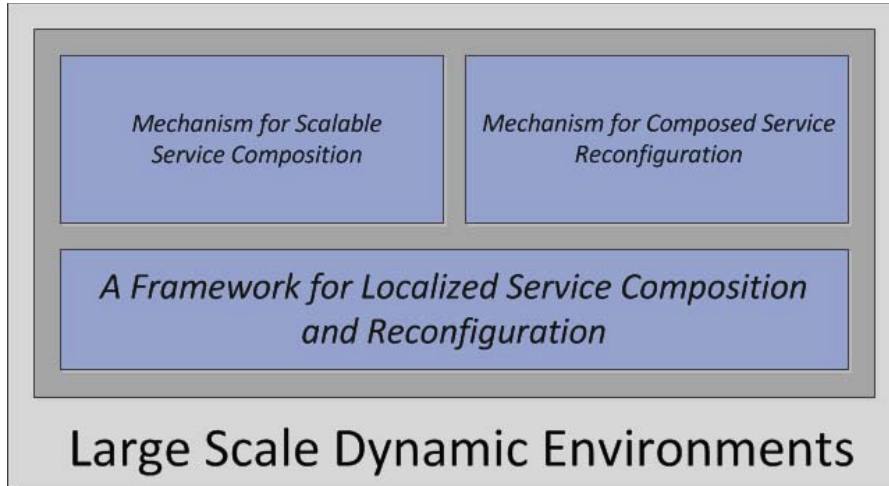


Figure 1.2: Block Diagram for Research Contributions.

is minimised. In this research we significantly decrease the communication cost during service composition in highly dynamic large-scale environments, while satisfying the composition locality. We achieve this by avoiding collecting global information in service composition process, which is common for existing works. Moreover, in our approach, nodes and links are mapped in subsequent stage. We propose a novel localized service composition algorithm (LASEC), in which the nodes collaborate only with their local neighbors to compose the service specified by a user. A technique called "Alien-information-based-Acknowledging" is proposed to help nodes avoid making premature decisions due to lack of global knowledge. We demonstrate feasibility of our approach in prototype implementation and conduct extensive simulations to study the performance of the proposed LASEC compared with existing decentralized and pull-based centralized techniques. Simulation results show that our technique decreases message cost and response delay. Based on collected results we discuss performance of our approach in terms of scalability, composition locality and coping with dynamic environments.

We also study the problem of a localized approach to service reconfiguration. This work relates to maintaining the already composed service. Given a composed service executed in the network, assume dynamic changes in the network during service execution. Our problem is to design a localized protocol for the nodes to collaborate in the reconfiguration process such that

the consistency of the composed service is preserved. Reconfiguration of composed service refers to changing the composed solution after the initial service composition has been identified and service provisioning established. This is very challenging in the large scale and highly dynamic pervasive computing scenarios. Existing works on composed service reconfiguration require coordinator to collect global information and make decisions, therefore cannot be directly used in these scenarios. We propose a localized approach to improve scalability by avoiding collecting global information. Nodes communicate only with their local neighbors and a technique called "Informed Acknowledge" is proposed to help nodes avoid hidden abort. We conduct extensive simulations to study the performance of the proposed approach compared with existing techniques. Simulation results show that our technique performs better than existing approaches in terms of communication cost and response delay, particularly in large scale and dynamic scenarios where collecting global information is impractical.

We investigate the problem of heterogeneity in service composition. In particular, we focus on the lower levels of the composition process, namely service discovery to aid higher levels of the service composition. We study how to provide service composition across different environments supported by different service management systems, which may use standard protocols as well as tailor-made mechanisms. We propose the Universal Adaptor approach, which consists of two major components: the Universal Adaptor Primitives and the Universal Adaptor Mapping. UAP is the universal set of primitives used by the user to discover the services across different environments, while UAM provides the mapping between the Universal Adaptor Primitives to the primitives used in various service composition systems. We develop a prototype of the proposed approach and we describe the implementation issues and the experiment results. Our solution in a lightweight manner bridges not only all existing but future service composition systems, including standard ones, as well as service composition mechanisms that support multiple protocols within the domain.

## **1.4 Outline of the Thesis**

With a view to provide users with scalable service composition in large scale dynamic pervasive computing environments, we initially propose a localized service composition algorithm. Next, we develop a localized composed service recovery mechanism and support for heterogeneity of the service composition environment . Below we describe the organization of the dissertation -

Chapter 2 (Background and Literature Review): This chapter provides a basic introduction to the service composition characteristics and issues in pervasive computing environments. We also study the existing research works in service composition and analyse their advantages and disadvantages.

Chapter 3 (A Framework for Scalable Service Composition): In this chapter we describe our framework which is developed to provide scalable service composition for users in large scale dynamic pervasive computing environments.

Chapter 4 (Localized Approach to Service Composition): In this chapter we describe our localized algorithm for service composition which is developed to provide scalable service composition for users in large scale and dynamic pervasive computing environments.

Chapter 5 (Localized Approach to Composed Service Reconfiguration): This chapter discusses our approach for composed service recovery in pervasive systems. We propose a localized approach to improve scalability by avoiding collecting global information in the reconfiguration process.

Chapter 6 (Localized Approach to Service Heterogeneity Support): In this chapter we study how to provide service composition across different environments supported by different service management systems. We propose the Universal Adaptor approach to bridge in a lightweight manner different service composition systems by abstracting the common characteristics of existing approaches and providing a universal set of primitives that enable service composition across diverse environments.

Chapter 7 (Conclusion and Future Directions): This chapter concludes the dissertation with summary of our research works and discussions on future research directions.





## Chapter 2

# Background and Literature Review

This chapter discusses the background of service composition research in general with special attention to the service composition systems and solutions developed especially for pervasive computing systems. In Section 2.1 we discuss the basic building blocks of a service composition system along with some essential system support components. We then, in Section 2.2 briefly discuss the scalability issues and in Section 2.3 currently available design choices to setup the background for classifying the existing service composition protocols. In sections 2.4 and 2.5 we discuss virtual network embedding and localized algorithms. Section 2.6 describes the service composition reconfiguration techniques.

### 2.1 General Components of a Service Composition System

In this section we shall discuss the framework and the major components of a service composition system. As described in Fig. 2.1, the service composition framework has two operational modules - the functional module and the execution module. Basic functions include means to specify the requirement by users and service descriptions by service providers, specifying composition plan, and selecting the best suited service providers. The execution functionalities include service monitoring and adaptation policies to suit the user need under dynamic environment changes.

The lowest layer of a service composition framework consists of the pervasive computing environment. Selecting of service providers is facilitated by the use of various protocols designed for different composition models. Users, human or devices alike, can specify composition plan

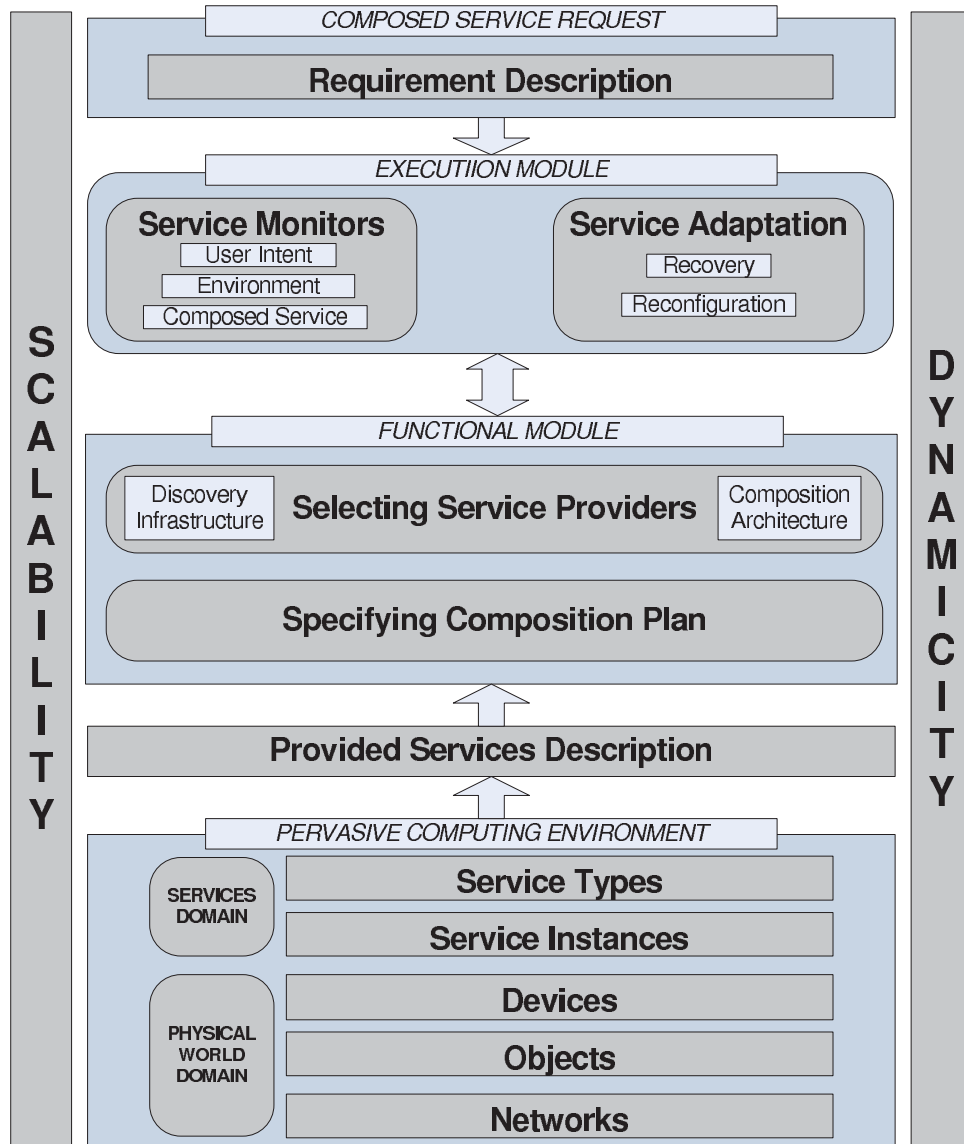


Figure 2.1: Service Composition Framework

and access services using different primitives. Service composition systems require scalability supports as well as measures to deal with dynamicity of the environments. We consider all the horizontal boxes as core components of a generic service composition system and the vertical boxes as essential system support service components. Different protocols may choose to implement the system support modules depending on the application and user requirements. Below we describe the issues associated with different parts of the framework.

**Pervasive computing environment** We define PvCE as a medium that provides user with all the functionality he needs to satisfy his requirements. It is built on objects in physical world with embedded computing devices interconnected according to underlying communication network. Functionalities provided by devices embedded in PvCE are exposed as services. A service instance is software deployed on the device. It exists to be invoked or to be interacted with. Abstract descriptions of service instances are classified into service types. In PvCE many service providers can provide the same functionality. Also, there may be no atomic service provider that can meet the requirement. Moreover, service providers can enter and leave environment dynamically.

**Provided services description** Smart objects may deliver functionalities for other objects and become service providers. Therefore, the service providers will provide description of their atomic services. Addelee et al. [3], also specify composition plans in which they can take part. Existing service composition approaches are characterized by different expressiveness of the language used to describe the provided services.

**Requirement description** In the meantime, the service requester can also express the requirement in a service specification language. Regarding the way that requested services are defined, we observe two approaches: low-level description and high-level description. In low-level description [21- 24], [3,5,26], [28-33] requested service is specified as a workflow, given the set of atomic services to be composed. In high-level description, requested service is specified as a goal to be achieved [21], [32-33].

**Specifying composition plan** Next service composition system tries to provide the needed functionalities by composing the available functionalities which are exposed by service providers. It tries to generate one or several composition plans with the service providers available in the environment. Since pervasive computing environments are saturated with service providers, often there can be several ways to satisfy the same requirement. In order to create the composition plan

diverse techniques can be used. In the most common approach [21-23], [3], [5-7], [29-31], [33] a composite service is broken down into sequence of interactions between atomic services. The system generates a customized composition plan that describes how various services should interact with one another as well as with the requestor. Facet [128-130] is a new component paradigm which is functionality centric, independent of data or user interface. The advantage of this paradigm is that it allows clients to download service codes for execution, without worrying that the execution cannot be completed in the resource-constrained devices. Also, it is much easier to build applications. Facet programmers need to provide different versions of facets and application programmers need to specify functionalities which make up an application.

**Selecting service providers** While many similar service providers are available to a requestor, it is important to determine which service provider should be used. It is challenging to find service providers for requestors efficiently and accurately. Service provider selection is a process of choosing service provider among discovered candidates. Many service providers can expose the same or similar functionalities. In that case, the services can be ranked based on the information provided from the non-functional attributes. In pervasive environments, this evaluation depends strongly on many criteria like the application context, the ability to interact with other service providers chosen to satisfy the requirement, the quality of the network, the non functional service QoS properties, and so on. In majority of the solutions selection is done by the service requestor [16-17], [18], [20]. After receiving a full or partial list of available services, a user chooses a service based on service information and additional information, such as context. However, too much user involvement causes inconvenience. It may be distracting for a user to examine many candidates for service providers and compare them. Second group of approaches is protocol based selection, in which protocols may select services for a user [19]. The advantage of protocol based selection is that it decreases user distraction. However, it is difficult to design a protocol selection which will reflect the actual user's will, since predefined selection criteria may not apply to all

cases. A balance between protocol selection and user selection is needed. One way to achieve this is to enhance protocol based service selection with context information [3], [5-7], [13], [19], [21], [23-33]. In addition to that, more desirable service providers can be identified and proposed to the user by considering QoS of service providers [3], [22], [29-31].

**Service adaptation** Service providers may leave environments due to mobility, unexpected power off or failures. In a pervasive computing environment, even when once the initial composition is identified and a service provisioning is established, the dynamicity of the environment may require change of the composed solution, which is called service recovery. In addition to dynamicity of the environment, the context in which composed services are deployed changes dynamically. For example user intent may change during execution of the service. Moreover, the requirement may be changed according to new environment conditions. Composed service must adapt to such changes. Reconfiguration of composed service refers to the capability of a service to adapt to changing user needs and environment conditions.

**Service monitors** In order to successfully perform recovery or reconfiguration of the composed service, first the system needs to decide when it is necessary to invoke such functions. In general, recovery and reconfiguration should be invoked upon detected changes regarding user intent (new needs), environment (new opportunities) or currently executed composed service (failure handling). This function is provided by service monitors.

## 2.2 Scalability in Pervasive Computing Environments

When designing mechanisms for service composition in PvCE, issues of scalability and dynamicity must be taken into consideration. Both environment and users are impacted by the scale and dynamicity of PvCE. Due to increased interactions, users experience problem of distraction. Also, in saturated PvCE, response time becomes an issue. Increased interactions are also problem to be tackled from the point of view of environments, where service providers must handle mes-

sages generated by different composed service requests. Moreover, with the increase of service providers, heterogeneity is more likely. Because computing resources are embedded in a physical environment, the distance between service providers and users becomes an issue.

This implies that algorithms and techniques should be different for pervasive computing environments than they are for small scale and static computing systems. There is a need for efficient mechanisms to perform service composition that will scale well and manage well dynamic and large-scale PvCE.

Although cloud computing can be a solution to service composition, UIO-based environments face challenges that require new approaches. Service composition in UIO-based pervasive computing environments refers to everyday objects turned into service providers and locally collaborating with each other to achieve user specified goals. Cloud computing refers to hosting services and running applications on the internet rather than locally. Although some of the service types can be hosted on the internet, there are some scenarios that require localized bottom-down approach. The main benefit of localized algorithms is their scalability in dynamic environments. Therefore, applications based on localized approaches are adjustable to the changes in the environment. In particular, applications deployed over infrastructureless networks may take full advantages of localized approaches. One of specific exemplary application is emergency response, such as natural disasters, accidents, military crisis. In these scenarios, there is no infrastructure support in which service composition could be managed in top-down manner. UIOs must act through localized coordination in order to compose services with other UIOs. Connections between them are created and broken, there is need for minimal configuration and quick deployment, and on top of that, local service providers are often preferred.

**Dynamicity support** Dynamicity of the environments is one of the major challenges in pervasive computing and to handle it efficiently, service recovery mechanisms are necessary. For example, the user with his/her mobile phone may walk out of a room while user's mobile phone

was providing a service in collaboration with other devices in this room. Or, user's mobile phone may be no longer available due to power limitations. In scenarios of one of atomic services in a composed service becoming suddenly unavailable, the challenge is to recompose the service as quickly as possible considering new state of the environment. Majority of proposed solution to service composition problem assume static service composition. In this approach, if one of the services fails, service composition needs to start over again. Dynamic service composition approaches [33], [35], [36] support replanning of the composed service during the execution of the composition. Services can be replaced, added or removed without starting over the service composition processes. Dynamic service composition is more difficult to implement than static since every service provider of the composed service is being monitored and should be replaced immediately in case of failure.

**Scalability support** Scalable service composition is main consideration when designing PvCE. We measure scale of the environment along several dimensions - scale of services and service providers, scale of service requests, scale of interactions in the environment. First, we are concerned with number of service providers as well as number and granularity of service types. Scale of environments varies, from small personal networks, through smart spaces and smart cities to all Earth connected into Internet of Things. Within each environment the number of embedded services grows. Moreover, the scale of the request, such as size of the request, number of requests and number of users is important as well. Additional consideration is distance between collaborating services, as well as distance between users and services. There are more and more interactions between users and computing environments.

In fact, all the mechanisms in service composition frameworks should be designed with a goal to achieve scalability. However, it is most critical in case of mechanisms that depend on the knowledge of the environment state, such as service selection. Since in large scale, dynamic PvCE, we assume that service requestors do not know what services are available in the environment

and service providers may join and leave environment at run-time, collecting knowledge about environment state is very expensive. Therefore, in next section we will mostly focus on selecting service providers in service composition.

## **2.3 Approaches And Techniques For Scalable Service Composition in PvCE**

In previous section we have presented mechanisms in service composition framework followed by issues introduced by scalability considerations. In this section we focus on techniques to achieve scalability to address these issues. We will show how to build scalable service composition mechanisms for PvCE.

### **2.3.1 Approaches to Achieve Scalability**

An important part of service composition framework presented in previous section is identifying appropriate service providers to contribute to the composed service. Depending on the environment characteristics different approaches can be chosen when designing techniques for selecting service providers. Service composition mechanisms in this respect adopt either a centralized, a decentralized or hybrid architecture. They are presented in Figures 2-4. These approaches detail how the knowledge about the environment (such as knowledge about service providers) is managed. Specifically, they differ from each other with respect to if global knowledge is necessary and how many entities collect any knowledge about environment. In addition to three approaches presented in Fig. 2.2 - 2.4, we will also discuss a localized version of the decentralized approach to service composition.

**Centralized approach** Many works [2, 3, 5, 6, 7] for PvCE propose centralized solutions to service composition. This category of works depends on the existence of a centralized directory of services. Namely, they rely on one or more central entities to maintain the global service information in order to make composition decisions. Service requestors submit requirements to



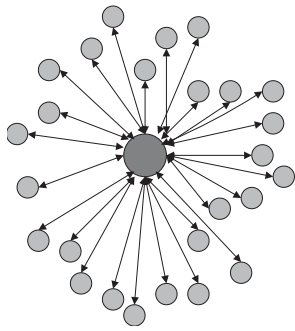


Figure 2.2: Centralized approach.

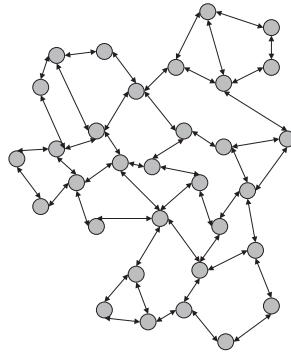


Figure 2.3: Decentralized approach.

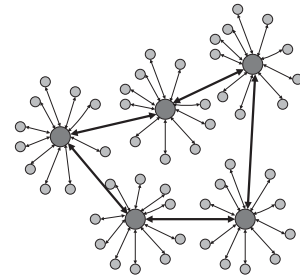


Figure 2.4: Hybrid approach.

the directory and the directory makes a decision on the list of services that should be returned to the requestors. This approach works well for environment that are well managed and relatively stable. However, the assumption on centralized control becomes impractical in scenarios with dynamic arrivals and departures of service providers, which requires frequent updates of the central entities, resulting in large system overhead. Moreover, relying on central entities to maintain global knowledge leads to inability of the system to serve the request when the central manager is compromised.

**Decentralized approach** Next category of distributed service composition approaches removes the need for a service directory and provides a fully distributed search for needed services. In [10] a hierarchical task-graph based approach to do service composition in ad hoc environments has been proposed. A composite service is represented as a task-graph and sub trees of the graph are computed in a distributed manner. This approach assumes that service requestor is one of the services and relies on this node to coordinate service composition. The coordinator uses global search across the whole network to do composition. The same domain of the problem was studied in [4]. It is different from [10] in terms of the way of electing coordinator. For each composite request, a coordinator is selected from within a set of nodes. The service requestor delegates the responsibility of composition to the elected coordinator. The elected coordinator utilizes distributed service discovery architecture, and subsequently integrates and executes services needed for

a composite request. However, many decentralized approaches still rely on global information. Even if there is no central directory, there is still need for a coordinator to collect global information. One category of decentralized approaches which do not rely on global information are localized algorithms.

Localized scalability is important issue in service composition. It is often better to identify service providers that are closer to the user and there is no need to know the environment status in other parts of the PvCE. Therefore, localized algorithms can benefit the scalable service composition in PvCE. They can significantly decrease the communication cost as well as response delay.

In general, a localized algorithm is such distributed algorithm where each node decides on its own behavior based on the information received from only nodes in its immediate neighborhood.

We have proposed a localized approach to service composition process in [8]. Our work is based on the analysis of the requirements of the fully decentralized service composition: (I) there should be no special entity to manage service composition process; (II) service providers can communicate only with their local neighbors, not all other service providers, (III) no service provider knows the full global information or gathers it.

Our approach starts at lower levels of hierarchy and decisions arise from joint involvement of entities. Service providers distributed in physical environment utilize localized interactions to satisfy given requested composite service. In this approach it is necessary for service providers to collaborate with their neighbors. We assume that all service providers in the environment can access to a whiteboard showing the user-specified service description, which is a set of component functionalities together with composition relationships. Our problem is to construct the composite service in a distributed manner, through localized interactions between service providers. We transform this problem to subgraph isomorphism problem, which is proven to be NP-complete.

Unlike previous algorithms that rely on global knowledge, in our algorithm each device only

maintains local state information about its physical neighbors. As a result, service providers will build an overlay network graph which satisfies the requirement with the quality of service comparable to centralized approach. We propose an algorithm for the devices to cooperatively construct the requested services through localized interactions. Device candidates decide with whom to cooperate only based on the information of the devices within its physical neighborhood. Our idea is to grow sections of composed service from available services. First, service provider identifies what service type it needs to interlock with through his output and then searches for appropriate service provider with matching input type. It is possible that service provider forms a section with another section. By merging pieces with each other, eventually global solution emerges.

**Hybrid approach** In order to address deficiencies of centralized approaches, some works exploiting hybrid approach have already been proposed. As the first step towards decentralization of service composition they introduce distributed directory of services. For example, in [9] a hierarchical directory has been proposed. The resource-poor devices depend on resource-rich devices to support service discovery and composition.

In the rest of this section, we compare the approaches discussed above in terms of their scalability. We further characterize them by the knowledge that algorithms rely on. In centralized approach, algorithm relies on global information to make composition decisions, while decentralized and hybrid approaches can be designed either based on global or local information. Localized approach is the example of decentralized algorithm with local information. In this section we only compare centralized approach as an example of algorithm requiring global information against localized approach that requires local information only.

Our comparison is in two dimensions, network size and request complexity. They describe ability of the service composition mechanism to manage increasing number of service providers and of atomic services in the request, with minimal effort given in number of messages and response delay [8].

**Network size** Network size in the service composition in PvCE is bounded by the number of service providers as well as their distribution in the physical environment. To discuss the impact of the network size on the scalability of service composition approaches, we consider the number of messages necessary to find first response and the time necessary to find the first answer.

Localized service composition mechanisms require fewer messages than centralized approaches for finding first result. This is because centralized directory needs to pull the request from the network, which includes generating and forwarding messages for sending the request and receiving the response, while localized approach takes advantage of localized interactions: does not collect global information and considers only some limited neighborhood. Moreover, localized approach scales very well with the growing size of the network while centralized mechanisms impose larger cost when the network grows.

Similar observations can be made for the time necessary to find first answer, according to the size of the network. Localized mechanism performs better than centralized. Since in centralized approach messages are flooded in the network, the collision effect causes increase in delay. Localized approach experiences lower rate of collisions, due to forwarding node selection. Similar as with message cost, localized approach scales better than centralized due to considering only local neighborhood.

**Request complexity** Size of the request influences scalability of the service composition mechanisms as well. We discuss influence of the request complexity on the number of messages generated and time necessary to find first answer by service composition protocol.

Although, localized mechanism avoids flooding the network with messages, numbers of messages generated increases with the complexity of the request. In this respect, centralized approach is more scalable. Since centralized mechanism collects global information, it collects the same amount of information for different request complexity, and incurs only a little additional cost for computing the result, while the cost of localized mechanism depends on the complexity result.

With respect to the time necessary to find first answer, according to the size of the request, localized approach performs better than centralized for finding first result. This is possible due to localized nature of discovery of services to be composed. If service providers are close to each other they can compose service faster that it takes for the central entity to collect information from the environment. With respect to scalability, the performance of localized and decentralized approaches is similar. Centralized approach requires longer time for more complex requests. Although the time for collecting the information from the environment is similar for different requests, however centralized mechanism will need additional time for evaluating the information and making decision.

### 2.3.2 Techniques for Achieving Scalability

We discuss further three main techniques for achieving scalability in service composition. We show not only how to select service providers for the composed service, but also how to recover from the failed composed service. Service replication objective is to provide several copies of services that are kept consistent. In case of service failure, copies can be used. In accordance with service locality principle, a service provider that is nearer should be used. Environment partitioning is a principle in which PvCE is split into parts that can operate independently to a large extent, such as smart houses, smart classrooms, etc.

**Service replication** Service composition mechanisms must provide a consistent view of service providers available in the network. They must be scalable and resilient to changes in the network's status. Service redundancy allows fast recovery of service composition. Service redundancy can apply to redundant service providers or redundant composition plan. Redundant service providers act as duplicates to failed services. Redundant composition plans allow choosing a completely different solution to the user request.

There are two main categories of methods to redundancy based service recovery. Persistence based method [11] assumes that the failure causes no further problems in the composed service.

The failing service provider is ignored and has to provide its own recovery actions if it wants to join the composition again. Blough et al. [12] proposed a relocation based method which migrates service from a failing service provider to a different one.

Repair and reconfiguration are also objectives of composition plan adaptation works which focus on action that modifies composition plan in a way that changes composition plan behavior in response to the change of context considered relevant to that composition plan. Reconfiguration is an adaptation that is triggered by the change of context related to user's need such as the need for new services or the re-placement of an original service. Repair - during execution a service that is bound to a task might become unavailable and the composition plan needs to adapt to compensate the failure.

The composition plan adaptations can be done in many methods. One of them is specifying all the possible adaptations in the composition plan itself [13]. However, this method is not flexible since the composition plan must be changed every time a new adaptation is needed. It is only suitable for a composition plan that does not change often.

Another method is utilizing abstract composition plan concept [14]. Abstract composition plan allows for adaptation in the abstract composition plan by determining the actual implementation of services in the abstract composition plan at execution time based on the context. In this method, the abstract composition plan contains one or more abstract activities which are replaced by one of the concrete implementation, which are predefined, to customize the service for the user. Variability point defines a part of the composition plan to which the service is selected and bound during execution. Flexibility of utilizing abstract composition plan is limited but it ensures that the adapted composition plan gives the expected result and is suitable for rigid composition plan such as the hospital workflow presented as their scenario. Taking into account the resource context, this method may require the executing task to restart if the resource becomes unavailable.

Adapting the composition plan instance [15] is yet another method. Each service in the func-

tional model is attached with a set of service versions which can be added at run-time. During execution, either the original service is executed or is replaced by one of the available versions based on various types of sensed context and the predefined adaptation rules. Therefore, the adaptation in this method takes place at the instance level and the service is bound during execution. This is more flexible method, but requires runtime service discovery and that the list of available services must be maintained

These adaptation levels imply different composition plan management. Composition plan instance adaptation can be handled easier while the adaptation in the concrete definition requires the handling of the running composition plan instances. Late binding in which the services in the composition plan are bound to the services as the execution proceeds to each service in the composition plan instance, provides more flexibility but requires runtime service discovery and that the list of available services must be maintained.

**Service locality** In service composition for PvCE, service locality refers to the distance between the services used in composition and it is required that it should be as small as possible. Composition locality is an important factor when the tasks include services which need to interact with each other and if they are embedded on different service providers in the network. To minimize such cost of composition all atomic services in the composed service need to be located as close to each other as possible. Localized approaches naturally tend to select compositions with high degree of composition locality. Service locality can also be achieved by caching of service descriptions.

**Environment partitioning** Works in traditional areas of identifying service providers as well as service recovery share common feature, which is a globalized nature of algorithms. As mentioned earlier, in a globalized algorithm, at least one node needs to maintain global network information. This is unsuitable for highly dynamic and large scale environments. Gathering such global information creates huge communication overhead, especially with frequent changes of topolo-

gy. Moreover, global knowledge is not always necessary. Splitting service environment into parts that operate independently increases scalability of the service composition mechanisms. This can be achieved by deployment of cooperating directories within the network. A distributed set of directories is deployed over base stations and the directories are responsible for a spatial region. The traffic generated by the service discovery process is kept to a minimum, and consumption of resources, in particular energy, is minimized.

## **2.4 Virtual Network Embedding**

Our problem may be regarded as similar to embedding problem, which is concerned with mapping a new virtual network (VN), with constraints on the virtual nodes and links, on to specific physical nodes and links in the substrate network [16], [17], [18]. Composed service graph can be seen as virtual network, where virtual nodes are required functionalities and virtual links are required relationships. Communication network in pervasive computing can be seen as substrate network with substrate nodes being service providers and substrate links ? communication links between service providers. Therefore virtual network embedding can be seen as service composition overlay (mapping a service composition graph, with constraints on the nodes and links, on to specific physical nodes and links in the communication network).

There are two categories of algorithms solving this problem. In two stage approach [16], [17], there are separate stages for node and link assignment. In single stage [18], nodes and links are mapped in subsequent stage. Single stage approach achieves faster recovery from a bad mapping decision ? this is because algorithm does not need to re-map all nodes and links from the beginning, it can backtrack to the last valid decision.

In all of surveyed works the VN mapping algorithm is carried out in a centralized manner and a central entity is responsible for receiving VN requests from users and for selecting and assigning a set of virtual nodes to a set of substrate nodes. We have already pointed out the limitations of centralized approach presented on scalability, efficiency and resiliency, especially in a highly



dynamic and changing environment (e.g. node/link failures, node mobility, etc). Although none of these works could be used directly for our approach, we were inspired by the idea of single stage mapping.

## 2.5 Localized Algorithms

Works in both areas of research, service composition and virtual embedding share common feature, which is collecting global information. This is unsuitable for highly dynamic and large scale environments. To remedy this problem, we have opted to adopt a localized approach.

Localized algorithms have been applied in wireless and mobile ad hoc networks to enable such tasks as target monitoring [19], service discovery[20], and topology management [21]. We have learned that localized approach can significantly decrease the communication cost as well as response delay. However, we are unaware of any localized algorithms for service composition in pervasive computing environments.

Localized approach to service composition requires nodes to broadcast their information in order to find collaborating nodes. There have been localized algorithms proposed for broadcasting in ad hoc networks, in which a small set of forward nodes is selected to reduce broadcast redundancy problem. The forward node set forms a connected dominating set (CDS). Summary of such methods is presented in [22]. In neighbor-designating methods the forwarding status of each node is determined by its neighbors. The source node selects a subset of its 1-hop neighbors as forward nodes to cover its 2-hop neighbors. Each forward node in turn designates its own forward node list. These methods consider broadcasting from one source to whole network. Requirements on localized service composition are different. Information needs to reach only selected nodes, and there may be many sources of the information. Moreover, additional, application oriented requirements have to be considered.

## **2.6 Composed Service Reconfiguration Approaches**

### **2.6.1 System Consistency in Dynamic Reconfiguration of Distributed Systems**

Preservation of system consistency is a major recon-figuration requirement. A system can become useless in case the preservation of consistency is ignored. The system under reconfiguration must be left in a 'correct' state after reconfiguration. A system is said to be correct if: (i) the system satisfies its structural integrity requirements, (ii) the entities in the system are in mutually consistent states and (iii) the application state invariants hold. Existing approaches that work with a driven safe state fall into two major categories. First category considers models in which during reconfiguration interactions are aborted and that rely on entities to recover from abortions [4].

Mechanisms based on interaction abortion require the application developer to provide roll-back mechanisms to recover from abortions without proceeding to errors. Therefore, the range of applications to which these mechanisms can be used is quite limited.

Another group of approaches avoids interactions to be aborted [5]. Mechanisms that do not abort interactions are designed to assure that interactions in progress are eventually completed, either before reconfiguration has started or after reconfiguration has finished.

### **2.6.2 Recovery in Discovery Systems**

Discovery systems must provide a consistent view of distributed components. They are designed to support the discovery of different resources with a variety of distributed applications on nodes with distinct requirements specified by individual semantics. Such systems must be highly available, scale, and be up to date on the network's status. Failure sources include the unreliable behavior of the resources that suddenly disappear, crash, recover, and then possibly later reappear.

Existing approaches to redundancy based service discovery fall into two major categories. Persistence based approaches [6] assume that an occurring defect causes no further degradation and is carried by all parts. This can be seen as a 'passive' recovery method. The failing part is ignored and has to provide its own recovery actions if it wants to join the system again.

Relocation based approaches [7] move an application or task from a failing to a different host re-directing also the possibly affected communication and data flow.

An appropriate amount of redundancy affecting both hardware and software allows a combination of different types of recovery. While hardware redundancy assists recovery only with spare parts as duplicates or additional resources, software and application redundancy can also include implementation diversity or relocation of services.

### 2.6.3 Workflow Adaptation Approaches

Workflow adaptation works focus on action that modifies workflow in a way that changes workflow behaviour in response to the change of context considered relevant to that workflow. Objectives of context-aware workflow adaptation are correction and customization. Customisation is an adaptation that is triggered by the change of context related to user's need such as the need for new services or the replacement of an original service. Correction - during execution a service that is bound to a task might become unavailable and the workflow needs to adapt to compensate the failure.

The workflow adaptations can be done in many approaches. One of them is specifying all the possible adaptations in the workflow definition itself [1]. However, this method is not flexible since the workflow definition must be changed every time a new adaptation is needed. It is only suitable for a workflow that does not change often.

Another approach is utilising abstract workflow concept [2]. Abstract workflow allows for adaptation in the workflow by determining the actual implementation of tasks in the abstract workflow at execution time based on the context. In this approach, the abstract workflow contains one or more abstract activities which are replaced by one of the concrete implementation, which are predefined, to customise the service for the user. Variability point defines a part of the workflow to which the web service is selected and bound during execution. Flexibility of utilizing abstract workflow is limited but it ensures that the adapted workflow gives the expected result and is

suitable for rigid workflow such as the hospital workflow presented as their scenario. Taking into account the resource context, this approach may require the executing task to restart if the resource becomes unavailable.

Adapting the workflow instance [3] is yet another approach. Each service in the functional model is attached with a set of service versions which can be added at run-time. During execution, either the original service is executed or is replaced by one of the available versions based on various types of sensed context and the predefined adaptation rules. Therefore, the adaptation in this approach takes place at the instance level and the service is bound during execution. This is more flexible approach, but requires run-time service discovery and that the list of available services must be maintained

These adaptation levels imply different workflow management. Workflow instance adaptation can be handled easier while the adaptation in the concrete definition requires the handling of the running workflow instances. Late binding in which the tasks in the workflow are bound to the services as the execution proceeds to each task in the workflow instance, provides more flexibility but requires run-time service discovery and that the list of available services must be maintained.

## Chapter 3

# Bottom-up Community: A Framework for Scalable Service Composition and Reconfiguration

This chapter describes the bottom-up community framework which refers to a group of candidate service providers, called bottom-up community, along with a suite of protocols and algorithms for scalable service composition and reconfiguration in large scale and dynamic network environments. Section 3.1 presents a generic description of our bottom-up community framework. In Section 3.2, the bottom-up community creation has been discussed with necessary design principles, challenging issues and our proposed solution technique. Section 3.3 discusses the design principles and our proposed solution approaches to provide scalable service composition and reconfiguration operations using the bottom-up community framework. Finally, Section 3.5 concludes this chapter by summarizing our contributions.

### 3.1 Generic Framework Structure

As already mentioned in Chapter 1, the existing support for service composition operations in large and scale dynamic pervasive environments is inadequate. The primary objective of our research is to ensure scalable service composition and reconfiguration for mobile users in heterogeneous pervasive computing environments. A large number of UIOs in these environments work collaboratively in a purely ad hoc manner and without any central control. This necessitates a localized

and bottom-up interaction model which will achieve scalable service composition minimizing both message costs and time response. To achieve this goal, we prefer a coordinator-less based service composition solution to a coordinator based one as the former appears to be more scalable than the latter. Moreover, the dynamic resource constrained and error-prone nature of pervasive environment prohibits the use of single, centralized and dedicated coordinator node. Even using dynamic coordinators (new coordinator for each request) leads to expensive message broadcasts which is not practical for pervasive computing environments with large number of dynamically changing devices.

To cope with this limitation we propose a new solution namely, the bottom-up community framework, in which no node knows the full information necessary to find a solution.

The bottom-up community is composed of a set of service provider nodes in an ad hoc network environment and a suite of protocols and algorithms for these service provider nodes to collaboratively provide scalable service composition and reconfiguration as well as heterogeneity support for users in UIO-based PvCE. The bottom-up community is used as a basic infrastructure over which many different service composition related protocols have been developed. To make it more clear, the bottom-up community provides application developers with a sense of scalability in a heavily dynamic and large scale UIO-based PvCE. The bottom-up community is scalable enough to carry on with service composition operations not depending on the scale of the environments. However, the quality of the solution depends on the range of locality considered.

In our work we are concerned with three layers of the service composition process. First, we focus on how to select the service for the requested composed service, then how to maintain the already composed service, and finally, how to aid the above two levels by discovering candidates for selection regardless of their implementation.

The bottom-up community framework shown in Figure 3.1 has three different layers. The lowest layer concerns about the heterogeneity support. We have also developed two higher layers

intended to provide scalable service composition and reconfiguration supports for users. However, application developers can use the bottom-up community to develop many kinds of scalable service management protocols, including service discovery, access, composition, reconfiguration etc.

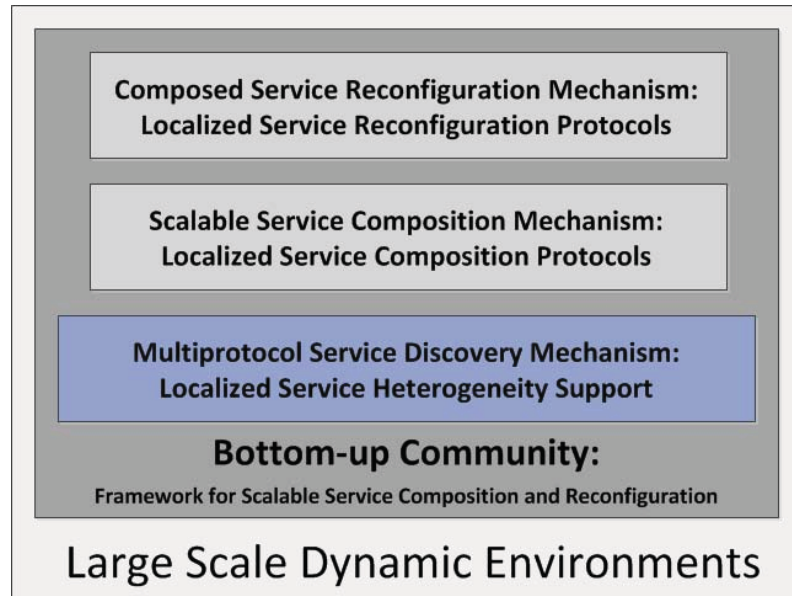


Figure 3.1: Bottom-up Community Framework.

For the bottom-up community we have adopted a common system model. All the protocols and algorithms developed using the bottom-up community share the same system model for their functioning. The bottom-up community framework is set up over a network that consists of a set of UIO nodes, each of which has a unique ID. The nodes communicate by sending and receiving messages through wireless channels. Whether two nodes are neighbors, i.e. they are directly connected, is determined by the signal coverage range and the distance between the nodes. Each node is a router and the communication between two nodes can be multiple hops. We assume variable message delay, so the message delivery is non FIFO. A node only knows the IDs of its neighboring nodes. Each node has a weight associated with it. The weight of a node indicates its quality and can be any functional attribute or non-functional attribute, such as the nodes battery power, memory size, computational capabilities etc. Two nodes may have the same weight value. In the next

few sections, we will discuss the different layers of the bottom-up community framework along with the general design principles, challenging issues and our proposed solution approaches.

## 3.2 Bottom-up Community Creation

### 3.2.1 General Idea

First, we describe the general idea of the localized approach to service composition.

Constructing composite service has many similarities to the problem of solving a jigsaw puzzle. A jigsaw puzzle consists of a number of oddly shaped pieces with parts of picture on them. By fitting together interlocking pieces a complete picture can be assembled. Similarly, composed service is assembled from services that are interlocked through their inputs and outputs. Composed service as a jigsaw puzzle is presented in Fig. 3.2.

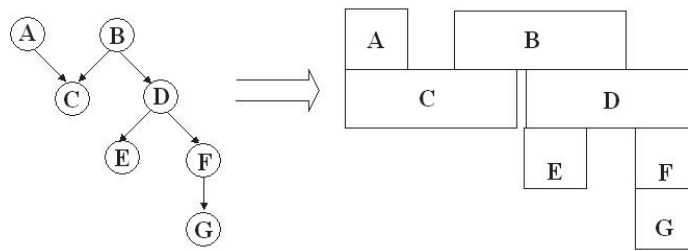


Figure 3.2: Composed service request as a jigsaw puzzle.

Our idea is to grow sections of composed service from available services. In the initial stages of search it is only some pieces that are used to construct service sections. At later stages these sections are joined together with other sections to gradually construct the completed solution. The basic operation in our jigsaw puzzle strategy is process of forming sections.

*Forming Service Sections.* First, service provider identifies what service type it needs to interlock with through his output and then searches for appropriate service provider with matching input type. Example of forming service section is presented in Figure 3.3. Service of type A interlocks with service of type C to form section A-C. However, there are two providers of service A. They may differ in their QoS or other non-functional properties. Only service provider A whose non-functional properties match properties of service C can form service section with C.



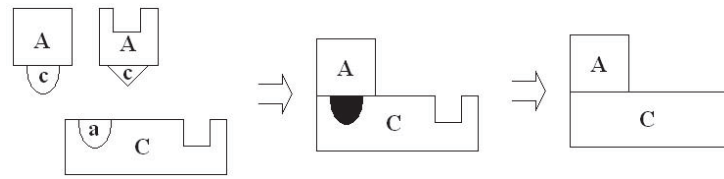


Figure 3.3: Forming service sections.

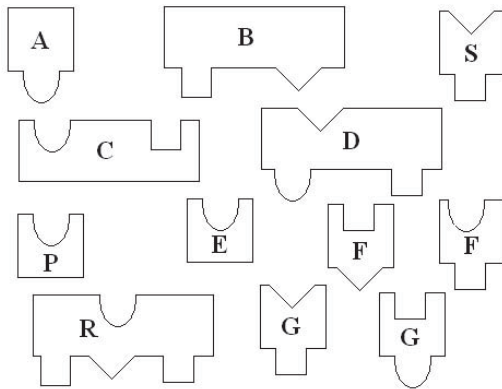


Figure 3.4: Available services jigsaw puzzle pieces.

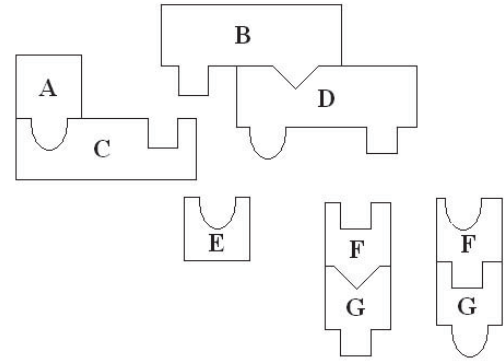


Figure 3.5: Formed sections: A-C, B-D, F-G.

*Growing Service Sections.* Each service provider tries to form a service section with a service that matches the requirement and is able to interlock with. It is possible that service providers forms a section with another section. The process of growing service sections is presented in Figures 3.4 and 3.5.

First, we show the available services in the environment. Some service types are not necessary to build a requested composite service, some service types are duplicated, with possible difference in input and output types. For example, in environment presented in Fig. 3.4, service types P, R, S do not contribute to the requested service presented in Fig. 3.5. Moreover, there are two candidates for service of type F and two candidates for service of type G. In the growing process, providers of requested service types try to form sections with other service providers. Fig. 3.5 shows formed sections A-C, B-D and F-G.

At later stages, using the same forming service sections process, service sections are joined together to construct the completed global solution. Therefore, focus is on solving locally one part of the puzzle. Each jigsaw piece tries to find appropriate neighbor with whom it can interlock. By

merging pieces with each other, eventually global solution emerges.

### 3.2.2 Challenging Issues

#### 1) Local Operations vs Global Behaviour

In our localized service reconfiguration, global behavior is achieved by strictly local operations.

Fig. 3.6 illustrates local messages sent between two service providers that have parent-child relationship. First, parent sends the Initialize message, child responds with Acknowledge message and parents confirms with Overlay message. These messages only carry local information; there is no propagation of the information in the network. However, using only these local messages, we can still achieve de-sired global behavior, which is summarized in Table 3.1.

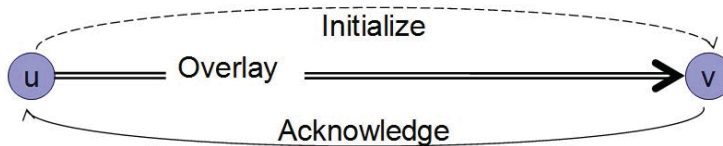


Figure 3.6: Local messages sent between parents and children.

Table 3.1: Achieving Global Behavior by Local Operations

	Initialize	Acknowledge	Overlay
Local operation	Send request for edge maintenance.	Accept request for edge maintenance.	Select edge for maintenance.
Global behaviour	Identify paths in the environment between nodes that have parent-child relationship in request graph.	Identify paths between leaves nodes and root node of the request graph.	From acknowledged paths choose those that form overlay graph.

#### 2) False and Uncertain Edges

In the process of forming and growing sections described above service providers create edges between themselves and their parents and children. However, if during forming a service section service provider takes into consideration only the information about the node with whom the edge is about to be created, hidden abort problem may occur as mentioned earlier. Therefore, such an edge can turn out to be false. This means that the node with whom the edge is created cannot compose remaining part of the request.

Example of this problem is shown in Fig. 3.7. Based on the request, in the localized algorithm, service providers with functionality A try to form sections with service providers with functionality B, and service providers with functionality B form sections with service providers with functionality C. In the environment with service providers shown in Fig. 3.7 service provider a1 has two candidates for forming a service section: b1 and b2. However, edge created between a1 and b1 will be false, since b1 cannot find in its neighbourhood service provider with functionality C.

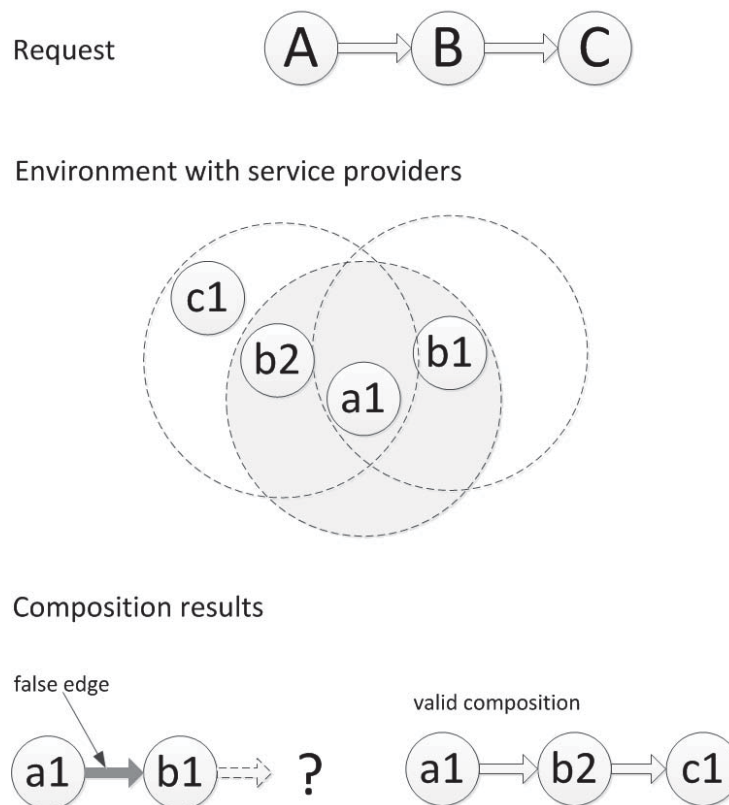


Figure 3.7: False edge concept illustration.

When there is a false edge existing in the created service section, then remaining edges are uncertain. Example is shown in Fig. 3.8. In this example a group of service providers a1, b1, c1, d1, e1, g1, h1, i1, j1, k1 created edges specified in the request. However, service provider e1 cannot find in its neighbourhood service provider with functionality F. Therefore, all the edges formed so far are uncertain.

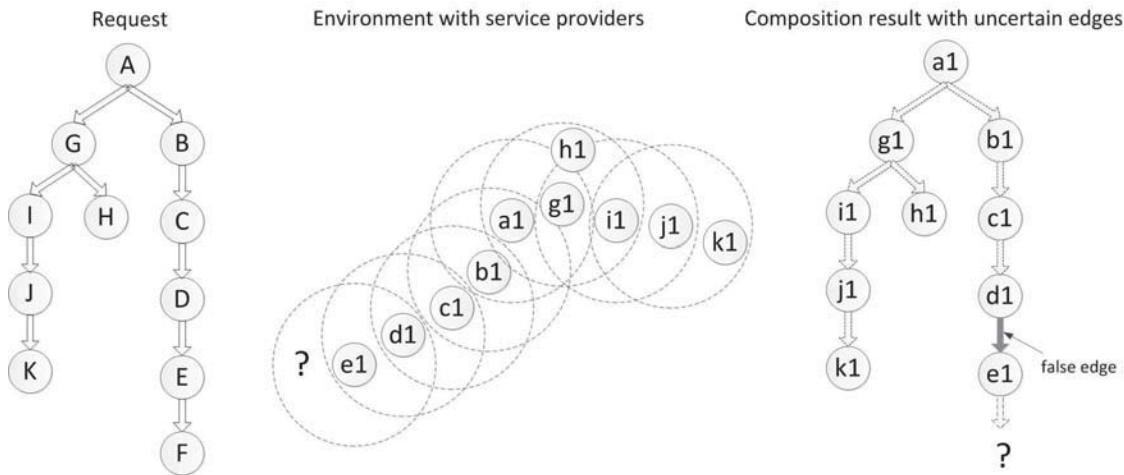


Figure 3.8: Uncertain edges concept illustration.

### 3.2.3 Informed Acknowledge Technique

To deal with the problem of false and uncertain edges, while avoiding collection information outside of local neighbourhood, we developed a technique called Informed Acknowledge. Using this technique ensures that the candidate service provider who has accepted the request for edge maintenance can successfully take part in reconfiguration. This is achieved by introducing dependency between acknowledge messages to nodes parents and acknowledge messages from its children. Candidate node delays sending acknowledge message to its parent(s) until it receives acknowledge message to all initialize messages it sent to its children. The illustration of the Informed Acknowledge technique is shown in Fig. 3.9. According to the request, service provider with functionality B should accept edge creation from service provider with functionality A, and request edge creation with service providers with functionality C and D. Upon receiving Initialize message from node A and before sending Acknowledge message in response, node B needs to wait for service providers with functionality C and D to acknowledge its local request.

## 3.3 Scalable Service Composition and Reconfiguration using Bottom-up Community

Using our bottom-up community, devices cooperatively construct the requested service. They can achieve that by utilizing the following operations: (i) local request creation, and (ii) local request

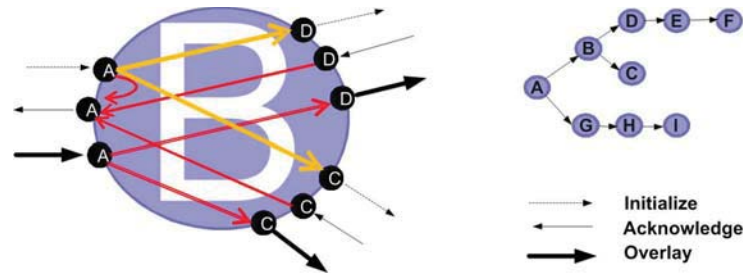


Figure 3.9: Informed acknowledge messages dependency.

unfolding/folding. Local request creation facilitates which services are required to be obtained from which node. Local request unfolding/folding is based on rules indicating which nodes are best to contact next and which nodes should not be considered further, since information that they have is irrelevant for the final solution.

*Component I: Local request creation.* Each node receives the user request and identifies if it can provide any service specified in the request. If the answer is positive, the node becomes candidate node and identifies what local requests it should create. Each node creates local requests only for its children.

*Component II: Local request unfolding/folding.* Nodes who have no parent unfold the local request to their 1 hop neighbors. At the same time broadcasting nodes use our Forwarding Node Selection algorithm to inform which 1-hop neighbors should re-broadcast the local request. This broadcast is repeated in k-neighborhood of initiating node. If local request arrives to the node which is intended receiver, the receiver determines if it should unfold its own local request or fold the received request.

*Example.* To understand the concept of local request un-folding/folding lets consider request A-D-E-F and environment in Fig. 3.10. For example, node A80 will unfold the local request for D in three directions: D50, D30 and D07. However D50 will never fold this local request, because it cannot satisfy finding E in its neighborhood. D07 will also not fold this local request, due to absence of I in its neighborhood. As for D30 it will unfold its local request for I and E. E11 and E05 in the neighborhood of D30 will unfold local request for F. Next, F30 will fold to E11 and

E05, upon which E11 and E05 will fold to D30 and eventually D30 will fold to A80. In each step the unfolding/folding node will perform local computation to determine when and in which direction to unfold/fold.

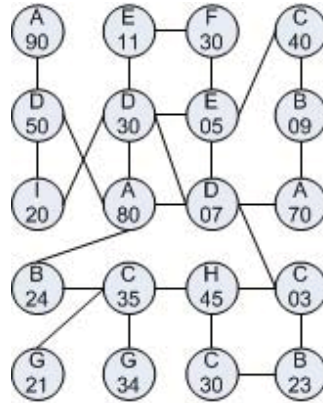


Figure 3.10: Service providers in the environment.

Local request unfolding/folding involves following steps:

*Step 1: wait for local requests from all parents.* If a node has multiple parents (such as node providing service F for the example request shown in Fig. 4.4) it needs to wait until it receives at least one local request from each of its parents. It is possible that the node will receive more than one local request from the same parent type.

*Step 2: unfold own local requests to all children.* When a node received local request from all parents, it puts them away and sends out its own local request created in phase I. If a node has no children (such as node F and E) it goes directly to step 4.

*Step 3: wait for response from all children.* After sending local requests a node waits for response to each local request. While waiting for the response from children, new local requests from parents can arrive, which node accepts and put away with other local requests.

*Step 4: fold to all parents.* When response to local re-quest was received from all children (or node has no children), node retrieves local requests that were put away, merges them with responses and sends to its all parents.

At least one response should be received from each child, and at least one response should be

sent to each parent. A node can use candidate ranking mechanism to limit the number of responses.

### Execution Example

The following example execution shown in Fig 3.11. illustrates the usage of informed acknowledge clearly.

The example execution shows the scenario with request in which services that need to be composed are A, B, C and D, and there are connected by edges AB, BC and BD.

At the beginning all service providers have status EMPTY and their sets of `in_edges` and `out_edges` are set to EMPTY (Fig. 3.11a). When the request is submitted to the network, all service providers are aware of it, including service providers shown in the example: a1, b1, c1, d1.

Based on the received request service providers set the values of `in_edges` and `out_edges`. Service provider a1 sets `in_edges` to empty, since in the request there is no incoming edge to the node A, and it sets `out_edges` to B. Service provider b1 sets `in_edges` to A and `out_edges` to C,D. Service providers c1 and d1 set their `in_edges` to B and `out_edges` to empty. In the same step service providers which set `in_edges` to any value other than empty, set their status to CANDIDATE, while service provider which `in_edges` is empty sets its status to INITIATOR (Fig. 3.11b).

In the next step, CANDIDATE nodes do nothing, and INITIATOR node a1 sends the request to its k-neighborhood to find node providing service type B. Service provider b1 receives this request and removes A from its `in_edges` (Fig. 3.11c).

Because `out_edges` of b1 is not empty, it cannot send an acknowledgement to A, it first needs to send requests to its neighborhood to find service providers providing services that are in its `out_edges`. Upon receiving this request, service providers c1 and d1 remove B from their `in_edges` (Fig. 3.11d).

For service provider d1, its `out_edges` and `in_edges` are empty so it can send acknowledgment to b1. Node b1 removes D from its `out_edges`, however, it still cannot send acknowledgment to a1, because there is still C in `out_edges` (Fig. 3.11e).





Figure 3.11: An example execution of the localized algorithm with informed acknowledge.

Only upon receiving acknowledgement from c1, node b1 can remove C from out\_edges making it empty and it then can send acknowledgment to node a1 (Fig. 3.11f).

Node a1 removes B from its out\_edges, and since both out\_edges and in\_edges are now empty and node a1 is initiator it can change its status to DONE and inform b1 about change in status



(Fig. 3.11g).

Node b1 responds by changing status of c1 and d1 (Fig. 3.11h).

### Optimizing

We have designed several techniques to optimize our algorithm along several dimensions. For decreasing number of messages we use method called informed local request and to save response time we propose accelerated acknowledge. Similar to our basic technique informed acknowledge, these techniques are based on introducing dependencies between messages. They are presented in Fig. 3.12.

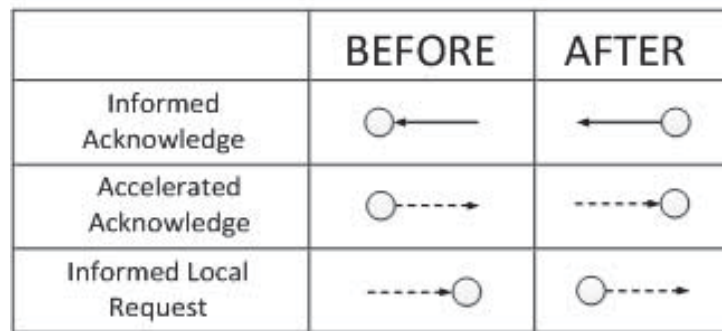


Figure 3.12: Local operation dependencies.

*Informed local request:* Before a node creates local request for its children, it waits to receive local request from its parent(s). This ensures node that the part of reconfiguration on its parents side is available. Therefore, service providers with false links on their parents side will never start sending messages, which can save global message cost.

*Accelerated acknowledge:* Allows node to send local request to its children before it receives any request from its parents. In this design, once the request arrives, node may be ready to send acknowledge and the local response time is decreased. Example of using accelerated acknowledge to decrease time is shown in Fig. 3.13.

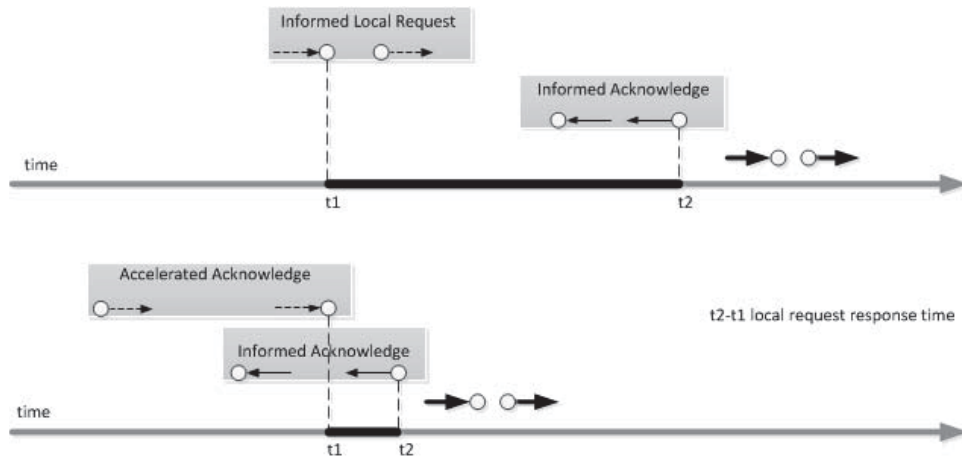


Figure 3.13: Decreasing local response time with Accelerated Acknowledge.

### 3.4 Summary

In this chapter we have introduced our bottom-up community framework. The framework has a collection of service provider nodes, chosen from all the nodes in an large scale and dynamic environment considering their functionalities, along with a suite of protocols that aim to provide scalable service composition and reconfiguration supports to mobile users in these environments. We have initially introduced the layered structure of the bottom-up community framework followed by the description of individual layers.

## Chapter 4

# LASEC: Localized Approach to Service Composition

In this research we significantly decrease the communication cost during service composition in highly dynamic large-scale environments, while satisfying the composition quality. We achieve this by removing coordinator from the composition process, and making service providers responsible to partially compose the request. Therefore, there is no node which knows full information about the composition process; however the global solution successfully emerges. We propose a novel localized service composition algorithm (LASEC), in which the nodes collaborate only with their local neighbors to compose the service specified by a user. To avoid problem of nodes collaboration with partners that may fail to compose remaining parts of the request, we introduce a technique called Alien-information-based Acknowledging (A-Ack). We also designed two additional techniques to optimize the performance with respect to time and message cost. We demonstrate feasibility of our approach in prototype implementation and conduct extensive simulations to study the performance of the proposed LASEC compared with existing decentralized and pull-based centralized techniques. Simulation results show that our technique decreases message cost and response delay. Based on collected results we discuss performance of our approach in terms of scalability, composition locality and coping with dynamic environments.

The rest of the chapter is organized as follows. In Section 4.1, we describe our system model and in Section 4.2, we formulate the problem. Next, in Section 4.3, we describe the design

of the proposed algorithms. We have evaluated the performance of the proposed approach using theoretical analysis in Section 4.3.3, prototype implementation in Section 4.4.3 and the simulation in Section 4.4.1.

## 4.1 System Model

The system model of the proposed approach is illustrated in Fig. 4.1. Communication layer consists of a finite collection of service providers, which can communicate with each other to satisfy user specified requests.

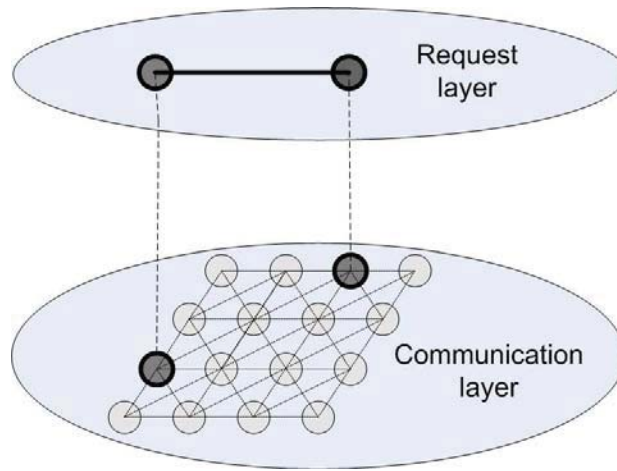


Figure 4.1: Localized service composition system. Devices in the communication layer explore localized neighbourhood to provide functions specified in the request layer.

Request layer shows the user-specified service description, which can be accessed by the service providers in the communication layer. We assume that all service providers in the environment can access to a whiteboard showing the user-specified service description, which is a set of component functionalities together with composition relationships.

*Definition 1 [Service Provider]:* A service provider is a directed attributed graph  $\vec{G}_s = \{V_s, \vec{E}_s\}$ , where  $V_s$  is a single element set that represents the service and  $\vec{E}_s$  is a set of directed edges that represent the inputs and outputs of the service [12]. The vertex attributes include description of the provided service type. The description of the service implemented by the service provider consists of elements presented in Fig. 4.2.

```

<serviceDescription> ::= <service> <provider> <input>
                        <output>
<service>             ::= <name> <functionality> | <name>
                        <functionality> <attributes>
<attributes>         ::= <attribute> | <attribute> <attributes>
<provider>           ::= <ID>
<attribute>          ::= <name> | <name> <values>
<values>             ::= <value> | <value> <values>
<functionality>      ::= <'light control'> |
                        <'temperature control'> | <'music'>

```

Figure 4.2: Service description.

*Definition 2 [Communication Network]:* The underlying communication network is described as a graph  $G_E = \{V_E, E_E\}$ , where the vertices correspond to service providers, and  $(x, y) \in E_E$  if and only if  $y$  can transmit and receive message directly from  $x$ . The set of service providers to which  $x$  can transmit and receive a message directly is called the physical neighbors of  $x$  and denoted by  $N(x) \subseteq P$ .

*Definition 3 [Composite Service]:* A composite service is modeled as a graph  $\vec{G}_R = \{V_R, \vec{E}_R\}$ , where  $V_R$  corresponds to required services and the set of edges  $E_R \subseteq V_R \times V_R$  represents required composition relationships, such as output of one service is accepted by output of another service. The description of the requested composite service consists of elements presented in Fig. 4.3.

```

<serviceRequirement> ::= <functionalRequirements>
<functionalRequirements> ::= <functionalRequirement> |
                            <functionalRequirement>
                            <functionalRequirements>
<functionalRequirement> ::= <functionality> | <functionality>
                            <attributes>
<functionality>         ::= <'light control'> |
                            <'temperature control'> |
                            <'music'>
<attributes>           ::= <attribute> | <attribute>
                            <attributes>
<attribute>            ::= <name> | <name> <values>
<values>               ::= <value> | <value> <values>

```

Figure 4.3: Requested composite service.

*Definition 4 [Parent and Child]:* For each service provider, if it can provide a service type specified in the composite service graph  $\vec{G}_R$ , the head endpoints adjacent to a node in  $\vec{G}_R$  are

called its parents, and tail endpoints are called its children. Each directed edge  $e_R = (x, y)$  has one parent and one child. Example is shown in Fig. 4.4. For such a requirement, each service provider which provides service A has one child (B). In the same example each service provider that provides service B has two children, D and E. Service providers which provide services E and F have no children for this request. Similarly, we can specify for each service providers how many parents they have with respect to particular request. The parent-child relationship relates to relationships specified in the composed service requests.

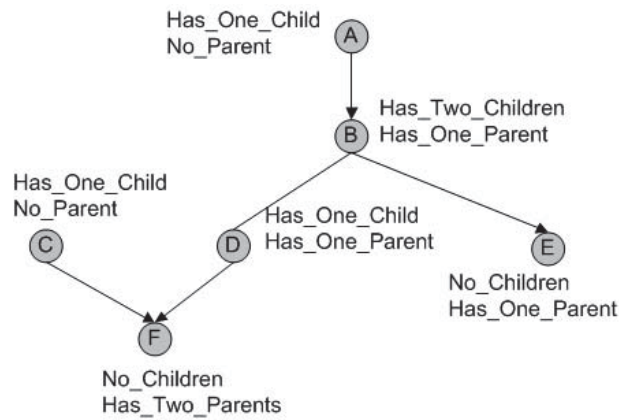


Figure 4.4: Parent/child relationship in the request. Parents unfold local request and children fold received local request.

*Definition 6 [Outgoing and Incoming Edges]:* For each service provider if it can provide a service type specified in the composite service graph  $\vec{G}_R$ , the edges in  $\vec{G}_R$  that come from its parents are called outgoing edges, and edges that come from it towards its children are called incoming edges.

Our problem is to construct the composite service in a localized manner, through collaboration between service providers. We transform this problem to subgraph isomorphism problem and prove that this problem is NP-complete. We also define a localized version of the subgraph isomorphism problem.

Unlike previous algorithms that rely on global knowledge, in our algorithm each device only maintains local state information about its physical neighbors. By enabling localized interactions

we achieve the solution which is highly scalable. We reduce redundant broadcast by employing forwarding node selection techniques. Moreover, in our approach, nodes and links are mapped in subsequent stage, which allows us to cope with dynamicity of the environments. Another important feature of our approach is high degree of composition locality. We validate our approach both through prototype implementation and simulation results.

## 4.2 Problem Specification

### 4.2.1 Problem Illustration

Our system consists of a finite collection  $P$  of nodes communicating by means of messages. An entity corresponds to a service provider in pervasive computing environment. Nodes communicate with other nodes to achieve a common goal, which is to satisfy given request. Node communicates by transmitting messages to and receiving messages from other nodes. Communication links are bidirectional.

#### 1) k-neighbourhood

In our system model node can communicate directly only with a subset of the other nodes. We denote by  $N(x) \subseteq P$  the set of entities to which node  $x$  can transmit and receive a message directly; we shall call such set of nodes the physical neighbors of node  $x$ . The physical neighborhood relationship defines a graph  $G_E = \{V_E, E_E\}$ , where  $V_E$  is the set of vertices and  $E_E \subseteq V_E \times V_E$  is the set of edges; the vertices correspond to entities, and  $(x, y) \in E_E$  if and only if the node  $y$  is a physical neighbor of the node  $x$ . The graph  $G_E = \{V_E, E_E\}$  describes the communication topology of the underlying network. In summary, node can only receive messages from and send messages to its neighbors and it can distinguish between its neighbors.

*Definition 6 [k-neighbourhood]:* The concept of k-neighbourhood is shown in Fig. 4.5. The k-neighborhood relationship defines a graph  $N_k(v_i) = (V_{ki}, E_{ki})$ , where  $V_{ki}$  is the set of vertices at distance  $k$  or less from  $v_i$ .

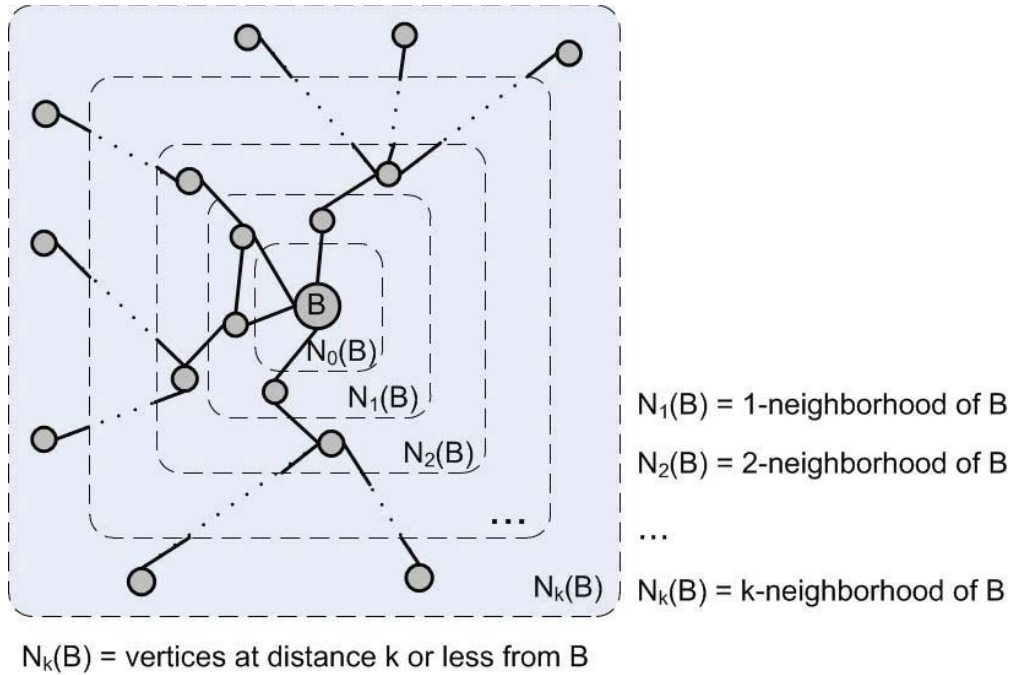


Figure 4.5: Definition of neighborhood. Parameter k corresponds to the maximum distance of neighbours from the vertex.

## 2) Service Composition Area

In localized approach to service composition, candidate service providers search for other candidates within k-hops. In our system model there is no coordinator who is responsible for collecting necessary information. Therefore each node explores its k-neighbourhood to find service providers which can collaborate in service composition. Since the edges in the request graph represent composition relationships that are necessary to be maintained in the composed service, we assume that service providers search in their k-neighbourhood only for their parents and children. Therefore, although each node in the environment searches only within k hops, the total area occupied by the resulting service composition is larger than k-hops. Size of service composition area depends on area of one node search (k-hops) and complexity of the request.

*Example.* In the request shown in Fig. 4.6, capital letters specify requested functionalities. Service providers in the environment are shown as nodes, small letters denote the function provided by the service providers, and numbers are unique ID to differentiate between different service providers of the service type. If we consider service provider a1, it only communicates with nodes



available in its k-neighbourhood and is not aware about the service providers available outside of it. For example, service provider a1 does not know that service provider f1 or d1 are available close enough to collaborate in the service composition process.

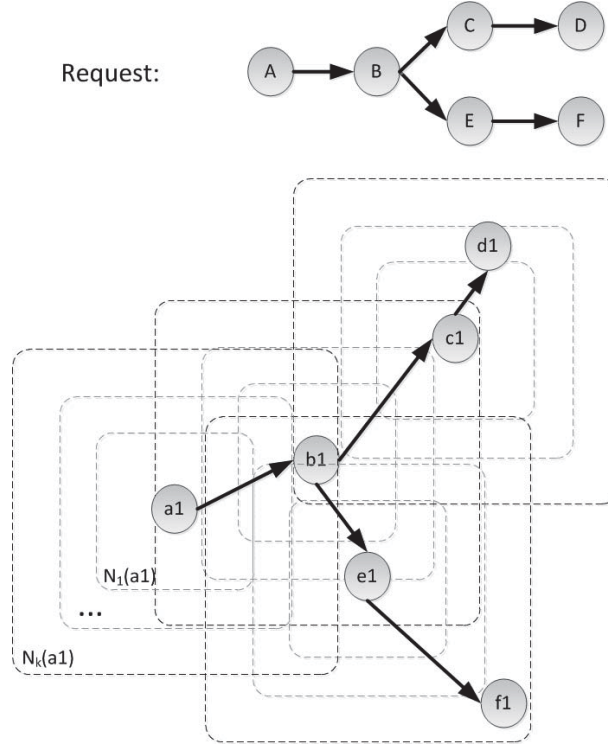


Figure 4.6: Service Composition Area. Each service provider searches within k-hops for service which type matches the type on the other end of the edge specified in the request.

### 3) Localized Service Composition in k-neighbourhood

*Example.* Lets consider example service request shown in Fig. 4.7. It can be represented by a graph:  $\vec{G}_R = (\{A, B, C, D, E, F, G, H, I\}, \{AB, BC, CG, CH, AD, DI, DE, EF\})$ . Furthermore, lets consider environment to which such request was submitted.

*1-neighbourhood case.* The example environment for 1-neighbourhood is shown in Fig. 4.8. Nodes are connected by communication links and for simplicity each node is described by service type, denoted with a letter of alphabet. We notice that in this environment, request from Fig. 4.7 can be satisfied by four different compositions. For example, the bottom left service is very dense, all nodes are one hop away from node providing service i. In contrast, in next composed service, two F and H nodes are 6 hops away from each other. However both of these composed services

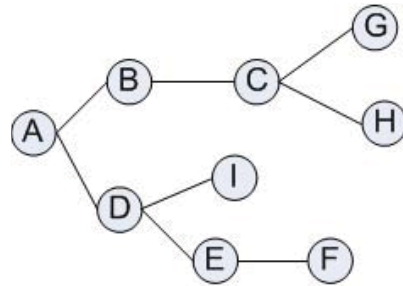


Figure 4.7: Example service request. ID of nodes specifies requested functionalities of atomic services. Links specify relationships.

are valid solutions to the request from Fig. 4.7.

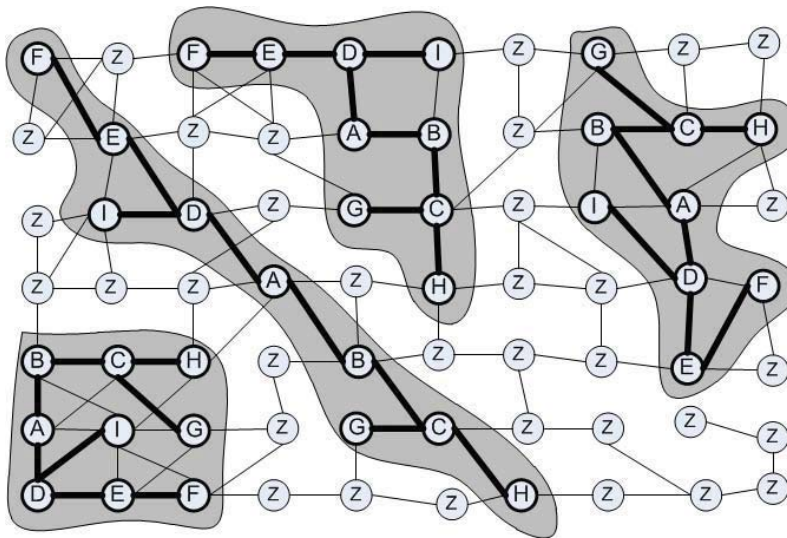


Figure 4.8: Service environment with possible 1-neighborhood service graphs.

*k-neighbourhood case.* In this case, request can be satisfied by nodes that are in at most  $k$ -hop distance from each other. The example environment for 3-neighbourhood is shown in Fig. 4.9. Request from Fig. 4.7 can be satisfied partially by nodes G and C, which are in 3-hop distance from each other.

#### 4.2.2 Problem Description

We study the problem of a localized approach to service composition. Given a set of nodes each providing certain services, the nodes are interconnected and can communicate with each other via wireless network, assume a user specified requested composite service with specified types of requested services and relationships between them. Our problem is to design a localized proto-

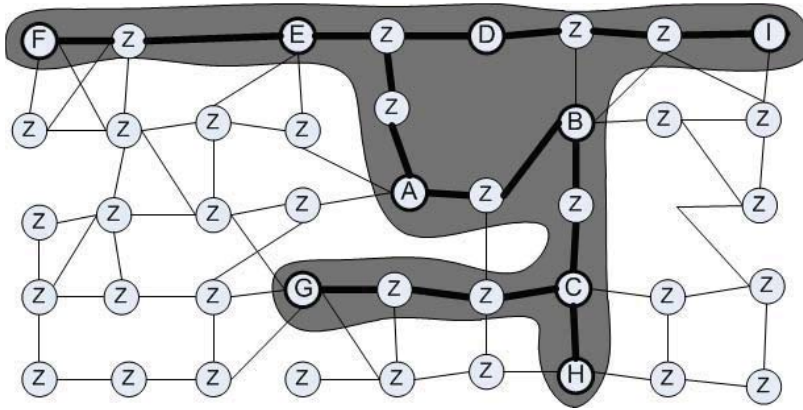


Figure 4.9: Service environment with possible  $k$ -neighborhood service graphs.

col for the nodes to collaborate in the composition process such that the communication cost is minimised.

We model this problem as follows. First we model the communication topology as a directed attributed communication graph (CG) with nodes representing service providers and edges representing communication links between them, and attributes represent service type. Service providers know only  $k$ -neighbourhood of the directed attributed graph. Requested composite service is modelled as a directed attributed service composition graph (SCG), where nodes represent requested services, edges represent relationships between services and attributes represent functionality of the service. We can see that in order to satisfy the request, we need to map each vertex and edge in service composition graph to exactly one vertex and edge in communication graph. Such a problem is known as a sub graph isomorphism problem.

Since our problem is localized, we define *a localized version of sub graph isomorphism problem*:

For a node  $i$  in the network, given  $k$ -neighbourhood of the node  $i$  in communication graph CG and service composition graph SCG our problem is to design a localized protocol for each node  $i \in CG$  to determine a set of outgoing and incoming edges from/to  $i$  to be maintained such that:

- there is a one to one mapping between edges maintained by  $i$  and edges from/to the node  $j$  in SCG, where  $j$  has the same functionality as  $i$ ,

- the number of edges maintained by  $i$  is the same as the number of edges from/to the node  $j$  in SCG, where  $j$  has the same functionality as  $i$ .

In our work, communication cost is cost incurred by the service composition process. Since in our localized approach the nodes need to collaborate with each other in bottom-up manner in a composition process, communication between them during composition may be costly, and therefore localized service composition algorithm should be designed with minimizing this cost in mind.

### 4.2.3 Problem Formulation

Let  $\overrightarrow{G}_R\{V_r, \overrightarrow{E}_r, \mu_r\}$  be a directed attributed graph with the vertex set  $V_r$  and the edge set  $E_r$  and let  $G_C\{V_c, E_c, \mu_c\}$ , be a directed attributed graph with the vertex set  $V_c$  and the edge set  $E_c$ . Let  $\mu_c$  be an attribute set. Let  $\forall v_i \in V_r \exists v_k \in V_c : f_i = f_k$ .

For each node  $v_a$  determine a set of edges  $E_a$  to be maintained such that  $E_a \subseteq E_c : \forall e_i(v, u) \in E_a \exists ! e_i(v, u) \in E_r : f_v = f_v \wedge f_u = f_u$  and  $|e(v_i, u_i1), e(v_i, u_i2) \dots e(v_i, u_i p)| = |e(v_a, u_a1), e(v_a, u_a2) \dots e(v_a, u_a p)| \forall v_i \in V_r : f_i = f_a$

### 4.2.4 Complexity Analysis

Given a service composition problem (SCP) instance I, find an overlay O of the communication topology graph GP, compatible with the service request graph GR. This problem is NP-complete. We prove its NP-completeness by showing that it is a generalization of the sub-graph isomorphism problem (SGI), a NP-complete problem [23], which is defined as follows:

*Instance:* Graphs  $G=(V1, E1)$ ,  $H=(V2, E2)$

*Question:* Does G contain a sub-graph isomorphic to H?

By setting  $GP=G$ ,  $GR=H$ , SGI can be reduced to SCP, while there still exist differences between SGI and SCP:

- G and H are undirected, unlabeled and non-attributed graphs, while GP and GR are directed,

labeled, and attributed graphs;

- SGI aims to find a sub-graph of  $G$ , while SCP aims to find an overlay of  $GP$ ;

For the first different point, according to [24][25], the variant of SCI for directed graphs is still NP-complete. [26] has also proved that labeled SGI problem contains unlabeled SGI problem by setting the label set size to be one, and consequently, the labeled SGI problem is still a NP-complete problem. Similarly, we can prove that non-attributed graph is an extreme case of attributed graph, where the attribute size is zero. That is to say, a directed, labeled, and attributed SGI problem is still NP-complete. Furthermore, we can show that a sub-graph is a special case of an overlay by restricting that the edges of the overlay  $E_o \subseteq E_r$ . In summary, the SCP problem is a generalization of the SGI problem, and thereby it is NP-complete.

### 4.3 Localized Service Composition Algorithm

Due to the NP-hardness of the problem SCP, a heuristic approach is needed to resolve it. In this section, we propose a localized service composition (LASEC) algorithm. The design of LASEC algorithm is inspired by the classic game of jigsaw puzzle. A jigsaw puzzle consists of a number of oddly shaped pieces with parts of picture on them. By fitting together interlocking pieces, a complete picture can be assembled. We expect to compose *sections* of *overlay graph* from available nodes and edges first and then join the *sections* into a completed *overlay graph*.

However, in jigsaw puzzle, the player usually has overall knowledge about the pieces, which helps to get to know which two pieces should be interlocked with each other. While, in localized composition, nodes composing the current *section* do not have the knowledge about other *sections* (or even about the current *section*). Hence, there are some challenges in localization composition:

- **Awareness.** How does a node know the current members of the *section* that it has participated in? How does a node know the *section* that it has participated in has already failed? Here, the failed *section* means the *section* that cannot further find other node in  $k$ -hop distance to

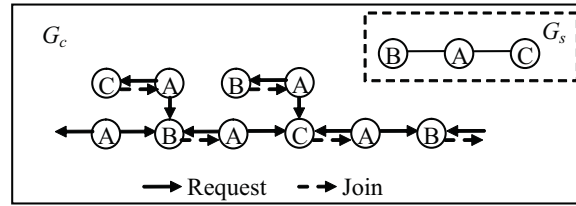


Figure 4.10: The unconverged case in localized composition.

compose with.

- **Judgement.** How does a node know which *section* is more possible to succeed the composition when the node is required to join? How to reduce or stop the useless composition which deems not to be completed?
- **Assembly.** Since there is no coordinator in the network, how to guarantee the composition to converge into one integrated *overlay graph*. For instance, in Fig. 4.10, three nodes with different types in  $G_c$  are required to be composed according to  $G_s$ . When the nodes of type  $A$  request to compose with two nodes of type  $B$  and  $C$ , respectively, no composition is achieved although there are many possible *overlay graphs* in the network.

Considering these challenges, we design our proposed LASEC algorithm as follows.

### 4.3.1 Description of the Algorithm

For convenience, some notations are defined in Table 4.1. Meanwhile, the root in  $G_s$  in LASEC is the node to which the maximum hops in  $G_s$  is minimum. We apply the *Floyd-Warshall* algorithm [24] to find the root in our algorithm. The parent of a node in  $G_s$  is the neighboring node with smaller hop count to the root, and the child of a node in  $G_s$  is the neighboring node with larger hop count to the root. The leaf in  $G_s$  is the node without a child.

We suppose all nodes in  $G_c$  have obtained the information of  $G_s$  (i.e., the service request) before the service composition. This can be achieved simply by broadcasting it to the service providers. In addition, as described in [15], there is a literature proposing different methods for making request known to the service providers. For example request can be already embedded in

Table 4.1: Definitions of some notations.

Notation	Definition
<i>Active node</i>	The node in $G_c$ with a corresponding node in $G_s$
<i>Non-active node</i>	The node in $G_c$ without a corresponding node in $G_s$
<i>Root node</i>	The <i>active</i> node in $G_c$ whose corresponding node in $G_s$ is the root
<i>Leaf node</i>	The <i>active</i> node in $G_c$ whose corresponding node in $G_s$ is a leaf
$G_s$ -neighbor	The <i>active</i> node in $G_c$ with a neighboring corresponding node in $G_s$
$G_s$ -parent	The <i>active</i> node in $G_c$ with the parent corresponding node in $G_s$
$G_s$ -child	The <i>active</i> node in $G_c$ with the child corresponding node in $G_s$
$G_s$ -descendant	The <i>active</i> node in $G_c$ with the descendant corresponding node in $G_s$
$T(k)$	The upper bound of the time for message flooding in $k$ -hop distance in $G_c$ .
$L$	The maximum hops of nodes in $G_s$ to the root.
$m$	The hops of the corresponding node in $G_s$ to the root.
$n$	The maximum hops of the corresponding node in $G_s$ to its descendant nodes.

Table 4.2: Definitions of messages.

Message type	Purpose of message	Information contained
<i>Local_request</i> message	Request <i>active</i> node to join in the composition	ID of the <i>active</i> node sending the message and the quality of current composition
<i>Acknowledge</i> message	Acknowledge the composition request	ID of the <i>active</i> node sending the message and the quality of current composition

the applications by the designers. Application can also have the ability to generate the requests based on the current context.

We define two types of messages as shown in Table 4.2. To guarantee that *active* nodes deliver messages within its  $k$ -hop distance, a time-to-live (TTL) variable is used in each message. In addition, we define two states for the nodes in  $G_c$ : *FREE* (the default state) and *SELECTED* (the state for nodes selected in an *overlay graph*).

We assume the quality of a composition is the sum of the qualities of all *active* nodes therein. The quality of each node may stand for the computation resource, bandwidth, etc. We define two

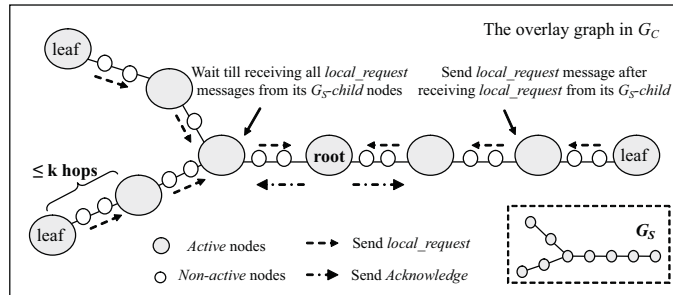


Figure 4.11: The basic idea of LASEC.

basic composition operations in the localized composition.

- *Operation 1*: When an *active* node receives a *local\_request* message (or an *acknowledge* message) sent from one of its  $G_s$ -neighbor nodes and decides to join in the composition, the *active* node will add its quality to the quality recorded in the message.
- *Operation 2*: when an *active* node receives each type of its  $G_s$ -child nodes' one *local\_request* message, the *active* node will add all the qualities recorded in the messages as well as the quality of itself.

The basic idea of LASEC algorithm is illustrated in Fig. 4.11, where a novel mechanism called *Alien-information-based Acknowledging (A-Ack)* is designed. Different from the traditional acknowledgement mechanism where the receiver will acknowledge the sender if receiving a message, in *A-Ack* mechanism, the receiver will not acknowledge the sender until the receiver obtains some additional information (we call it alien information) from other nodes. This additional information relates to whether the receiver is able to collaborate with other necessary nodes or not. This mechanism allows us to guarantee the composition to converge into one integrated *overlay graph* without nodes having any global information. We introduce the *A-Ack* mechanism as follows.

- At the beginning, all *active* nodes identify whether they are *root* nodes, *leaf* nodes or others.
- Then, each *leaf* node in  $G_C$  initiates a *local\_request* message and sends it to its  $G_s$ -parent nodes.



- When an *active* node receives the *local\_request* message from its  $G_s$ -child, it will execute *operation 1*, and then sends a new *local\_request* message to its  $G_s$ -child nodes. Specially, if the *active* node's corresponding node in  $G_s$  has multiple child nodes, the *active* node will wait until it has received each type of its  $G_s$ -child nodes' at least one *local\_request* message. Then, the *active* node executes *operation 2* and sends a new *local\_request* to its  $G_s$ -parent.
- When a *root* node has received each type of its  $G_s$ -child nodes' at least one *local\_request* message, the *overlay graph* achieves. Then, the *root* node broadcasts an *acknowledge* message to all *active* nodes involved in the *overlay graph*.

Note that, an *active* node may receive multiple *local\_request* messages sent by different  $G_s$ -child nodes. For this case, the *active* node selects one message which is the message recording the highest quality of the current composition.

Based on the *A-Ack* mechanism, we design the LASEC algorithm for each node in  $G_c$  with

**Algorithm 1, 2, 3 and 4.**

---

**Algorithm 1** The proposed LASEC algorithm

---

```

1: node_state = FREE
2: Input:  $G_s$ 
3: Apply Floyd-Warshall algorithm
4: Output: the root in  $G_s$ 
5: if the node's corresponding node  $\in G_s$  then
6:   if the node's corresponding node in  $G_s$  is the root then
7:     Execute Algorithm 2
8:   else if the node's corresponding node in  $G_s$  has no child then
9:     Execute Algorithm 3
10:  else
11:    Execute Algorithm 4
12:  end if
13: else
14:   for timer  $\leftarrow$  0 to  $2L * T(k)$  do
15:     Keep receiving messages and forward the messages
16:   end for
17: end if

```

---

From the algorithm, we can see: 1) *active* nodes are composed orderly in  $G_c$  with the rule in the algorithm that *active* nodes send *local\_request* messages only if they receive *local\_request* from their  $G_s$ -child nodes, thus reducing many redundant and meaningless compositions in  $G_c$ ;

---

**Algorithm 2** Algorithm for *root* nodes

---

```

1: for timer  $\leftarrow$  0 to  $2L * T(k)$  do
2:   Keep receiving messages and help to forward messages whose destinations are not itself
3: end for
4: if receive each type of  $G_s$ -child's at least one local_request message then
5:   For each type of  $G_s$ -child, select one message recording the highest quality
6:   Broadcast an acknowledge message to all the  $G_s$ -child nodes recorded in the selected local_request
   messages
7:   node_state=SELECTED
8:   OUTPUT: IDs of the  $G_s$ -child nodes recorded
9: else
10:  node_state=FREE
11: end if

```

---



---

**Algorithm 3** Algorithm for *leaf* nodes

---

```

1: Send a local_request message to all its  $G_s$ -parent nodes
2: for timer  $\leftarrow$  0 to  $(L + m) * T(k)$  do
3:   Keep receiving messages and help to forward messages whose destinations are not itself
4: end for
5: if receive acknowledge messages from its  $G_s$ -parent nodes then
6:   record IDs of the  $G_s$ -parent nodes sending the acknowledge messages
7:   node_state=SELECTED
8:   OUTPUT: IDs of the  $G_s$ -parent nodes recorded
9: else
10:  node_state=FREE
11: end if

```

---



---

**Algorithm 4** Algorithm for other *active* nodes

---

```

1: for timer  $\leftarrow$  0 to  $n * T(k)$  do
2:   Keep receiving messages and help to forward messages whose destinations are not itself
3: end for
4: if receive each type of  $G_s$ -child's at least one local_request message then
5:   For each type of  $G_s$ -child, select one message recording the highest quality
6:   Execute operation 2
7:   Record the IDs of the  $G_s$ -child nodes in the selected local_request messages
8:   Send a new local_request message with the updated quality to all its  $G_s$ -parent nodes
9: end if
10: for timer  $\leftarrow$  0 to  $(L - n + m) * T(k)$  do
11:   Keep receiving messages and help to forward messages whose destinations are not itself
12: end for
13: if receive acknowledge messages from its  $G_s$ -parent nodes then
14:   record IDs of the  $G_s$ -parent nodes sending the acknowledge messages
15:   node_state=SELECTED
16:   OUTPUT: IDs of the  $G_s$ -parent nodes recorded and IDs of the  $G_s$ -child nodes recorded
17: else
18:  node_state=FREE
19: end if

```

---

2) each node records only the IDs of its selected  $G_s$ -neighbor nodes and the quality of current composition. The node does not need to forward all the IDS of nodes involved in the composition, hence making the message cost of the algorithm low.

### 4.3.2 Discussion of the Algorithm

The algorithm proposed in the previous section can guarantee the optimal service composition under the constraint that the communication between one node and its neighbours are within a number of  $k$  hops. However, if we relax this constraint the distributed way to achieve the global optimal may become quite costly. Here we discuss how to guide the service composition in such case. We suppose there is the knowledge about the range of benefit when a user requests a tree-based composite service. It means that each node and hence each tree rooted at a node in the service tree can have a benefit range estimate. Based on this knowledge and the processing condition in the real environment, when we face multiple combinations of nodes that all may be possible to be final service composition, we can select the ones whose sub-tree dominate the benefit of the whole service-tree and eliminate the ones whose benefit is definitely less than the others.

Centralized approaches, due to global information available to them, are naturally able to find the composition with highest QoS. Since in localized approaches the global information is never revealed to the nodes, achieving optimal service composition is challenging. On the other hand, finding the optimal subgraph in a large complex graph is an NP-hard problem. However, our proposed LASEC algorithm can obtain the optimal subgraph when given the constraint that active nodes deliver messages within local  $k$ -hop area, as shown in the proof for Theorem 6.

The other thing we need to emphasize is the position we initiate the service composition. A trade-off is necessary about the latency and the energy combustion. In our LASEC algorithm, the leaf node starts this composition processing and the upper-stream nodes are waiting for the results from leaf nodes. It is suitable for the applications which do not have strict real-time requirement but only concern about energy consumption or traffic cost. However, for the application which has

strict real-time requirements, we need to set a proper location for the start of service composition. A node between the root and leaf node can serve this purpose. It explores several levels downstream (but not deep enough to leaf nodes) to collect the real service benefit information, estimate the possible benefit, and then at one side directly report to the root and on the other side continue the exploring. In this way, the latency requirement can be met and a probabilistic optimal result can be achieved.

### 4.3.3 Analysis of the Algorithm

In this section, we first prove the correctness of the proposed algorithm, including *safety*, *liveness* and *uniqueness*. Then, we analyze its performance in terms of message cost, running time and quality.

#### Correctness of the Algorithm

First, we give some lemmas and the proofs as follows. For convenience, we call the process that an *active* node receives messages, selects the messages and records the IDs of nodes sending the selected messages, as making composition.

**Lemma 1.** *An active node makes composition only with its  $G_s$ -neighbor nodes in  $G_c$ .*

*Proof.* According to LASEC algorithm, only two kinds of messages are delivered between *active* nodes in  $G_c$  to implement the composition. Meanwhile, the *active* nodes send *local\_request* messages only to their  $G_s$ -parent nodes to request composition, and they send *acknowledge* messages only to their  $G_s$ -child nodes to acknowledge the composition. Therefore, the *active* nodes make the composition only with their  $G_s$ -neighbor nodes.  $\square$

**Lemma 2.** *An active node makes composition with only one of its  $G_s$ -child nodes in each type.*

*Proof.* According to LASEC algorithm, when an *active* node receives *local\_request* messages from its  $G_s$ -child nodes, it always selects one message recording the highest quality in each type and records the IDs of the  $G_s$ -child nodes who send the selected messages. When the *active* node

delivers *acknowledge* messages, the node sends the messages only to the recorded  $G_s$ -child nodes to acknowledge the composition. Therefore, the active node makes composition with only one of its  $G_s$ -child nodes in each type.  $\square$

**Lemma 3.** *An active node makes composition with its  $G_s$ -child nodes only if they have already made a composition, except for the case that  $G_s$ -child nodes are leaf nodes.*

*Proof.* According to LASEC algorithm, an *active* node will not send *local\_request* messages unless it has received each type of its  $G_s$ -child's at least one *local\_request* message, which means the  $G_s$ -child sending the message has already received *local\_request* messages and made a composition unless the  $G_s$ -child is *leaf* node. Therefore, the *active* node can make composition only if its  $G_s$ -child nodes have already made a composition except for the case of *leaf* nodes.  $\square$

With the Lemmas above, we prove the correctness of LASEC algorithm including *safety*, *liveness* and *uniqueness* as follows.

**Theorem 1:** *Safety: non-active nodes in  $G_c$  will not change its state to SELECTED.*

*Proof.* According to LASEC algorithm, a *non-active* node surely cannot change its state to *SELECTED* during the composition, no matter it receives what kinds of messages. Therefore, LASEC guarantees the *safety* property.  $\square$

**Theorem 2:** *Liveness: the proposed LASEC algorithm can implement an overlay graph if there is one in  $G_c$ .*

*Proof.* According to LASEC algorithm, only *leaf* nodes initiate composition request. The *local\_request* messages of *leaf* nodes are sent to only the  $G_s$ -parent nodes during the composition. According to **lemma 1** and **3**, each *active* node makes composition only with its  $G_s$ -neighbor nodes, and there is a sequential rule for the composition that the *active* node makes composition only if its  $G_s$ -child nodes have made one. Since  $G_s$  is a tree graph, each node in  $G_s$  has only one path to the root and one parent in  $G_s$ . Hence, when an *active* node receives *local\_request* message

from one of its  $G_s$ -child node, it can ensure the composition of an overlay graph's branch that is formed by each type of the descendant *active* nodes of the *active* node. With the constraint of **lemma 1** and **3**, *active* nodes will be composed gradually and orderly from the *leaf* nodes till the *root* nodes without any local endless loop. And the *root* nodes will not broadcast the *acknowledge* message until it affirms the completeness of the composition of all its descendant *active* nodes, i.e., the *overlay graph*. Hence, the proposed LASEC algorithm can implement completed *overlay graph*.  $\square$

**Theorem 3:** *Uniqueness: When the algorithm is finished, each active node with the SELECTED state records only one  $G_s$ -child node in each type of service.*

*Proof.* According to **lemma 2**, each *active* node makes composition with only one of its  $G_s$ -child nodes in each type. When the *root* node receives all the *local\_request* messages, it also selects each type of its  $G_s$ -child nodes' one *local\_request* messages and form the *overlay graph*. Then, the *root* node broadcasts the *acknowledge* message to its descendant *active* nodes uniquely recorded by their  $G_s$ -parent nodes, and each *active* node receiving *acknowledge* messages will change its state to be *SELECTED*. Therefore, each *active* node with the *SELECTED* state records only one  $G_s$ -child node in each type of service.  $\square$

### Performance of the Algorithm

We give the performance analysis of the LASEC algorithm in terms of message cost, running time and the quality of the *overlay graphs* achieved by LASEC as follows.

**Theorem 4.** *Each active node in  $G_c$  sends no more than two messages during the execution of the LASEC algorithm.*

*Proof.* According to LASEC algorithm, an *active* node will send a *local\_request* message if it receives *local\_request* messages from its  $G_s$ -child nodes during a certain time  $n * T(k)$ . And, the *active* node will send an *acknowledge* message if it receives *acknowledge* messages from its

$G_s$ -parent nodes during a certain time  $(L - n + m) * T(k)$ . Therefore, each active node in  $G_c$  will send no more than two messages during the execution of the LASEC algorithm.  $\square$

**Theorem 5.** *The upper bound of the running time of the LASEC algorithm is  $2L * T(k)$ .*

*Proof.* We discuss the upper bound of the running time for each kind of *active* nodes first. From **Algorithm 2**, it can be easily seen, the upper bound of *root* nodes' running time is  $2L * T(k)$ . From **Algorithm 3**, the upper bound of *root* nodes' running time is  $(L + m) * T(k) < 2L * T(k)$ . While, from **Algorithm 4**, the upper bound of other *active* nodes' running time is  $n * T(k) + (L - n + m) * T(k) = (L + m) * T(k) < 2L * T(k)$ . Since all nodes are assumed to execute the LASEC algorithm upon obtaining the information of  $G_s$  simultaneously, the upper bound of the running time of the LASEC algorithm is  $2L * T(k)$ .  $\square$

**Theorem 6.** *The root nodes in the LASEC algorithm can obtain the overlay graph with the highest quality, when given the constraint that active nodes send messages within local  $k$ -hop area.*

*Proof.* According to the LASEC algorithm, an *active* node always selects the `local_request` message recording the highest quality among the messages that it has received from its  $G_s$ -child nodes. Then, the *active* node adds its own quality into the quality recorded in the selected message, and sends a `local_request` message to its  $G_s$ -parent nodes. In this way, an *active* node always selects the composition with the highest quality within its local  $k$ -hop area. Hence, when a *root* node gets each type of its  $G_s$ -child nodes' `local_request` messages, it obtains the highest-quality *overlay graph* that it can to achieve in global view with the constraint that *active* nodes send messages within local  $k$ -hop area.  $\square$

Obviously, the larger  $k$  is, the higher quality of *overlay graph* can be obtained. When  $k$  equals to the maximum hop counts in  $G_c$ , the quality of *overlay graph* achieved by LASEC is equivalent

to that of *overlay graph* achieved by a centralized algorithm. However, larger  $k$  leads to larger  $T(k)$ , thus resulting in higher running time of the algorithm.

Note that, we define the quality of the *overlay graph* as the sum of all individual quality of *active* nodes therein in this chapter. When various weights of the nodes' quality need to be considered in the *overlay graph*'s quality, the proposed LASEC can just combine with the decentralized approach in [29] which focuses on computing the services' weight in the QoS of composition.

## 4.4 Performance Evaluation

### 4.4.1 Simulation

We have carried out extensive simulations to evaluate the performance of our algorithm. Moreover, to show the advantage of our algorithm (LASEC), we have also implemented a distributed approach (DEC) [13] as well as a pull-based model of the centralized algorithm for service composition (CEN) [11], and have compared their results with ours. Below, we give a brief description of the implemented algorithms.

*LASEC. Localized service composition algorithm.* Our localized service composition algorithm has been applied to the network with implemented forwarding node selection protocol.

*DEC. Distributed embedding algorithm.* In this algorithm for each service composition coordinator is assigned, which broadcasts queries for each service type requested, available service providers respond, coordinator sends ACK message to them which is followed by confirmation from service providers.

*CEN. A centralized service composition algorithm.* In this algorithm, we provide a dedicated service directory. Initially there are no services registered with the directory. When a user specifies a composite service, it is submitted to the directory in the form of a graph. Upon receiving the request, the directory pulls from the environment the information related to service types specified in composition request. The pull operation is supported by broadcast.

In our simulations, we consider a grid network of  $N$  nodes which are uniformly deployed.



The total number of nodes in the network is varied to examine the effect of system scale on the performance. Also we varied the complexity of the request. A service type is randomly assigned to each node. In each network we set the total number of service type so that the density of service types ( $ds=N/ns$ ) is comparable. The simulation parameters are listed in Table 5.5.

Table 4.3: Simulation Parameters

Parameters	Values					
Number of nodes, (N)	64	81	100	121	144	169
Number of service types ( $n_s$ )	10	13	17	20	24	28
Service density	6, 7.5, 9, 11, 12.5, 14					
Request complexity	3, 4, 5, 6, 7, 8					
Territory scale ( $m^2$ )	200					
Transmission radio range (m)	40					

We evaluated the performance of the algorithms using the following metrics:

*Message overhead:* the number of messages generated by the composition process. We do not consider injecting request as the part of composition process. We start measuring messages that are generated after the request was specified.

*Response delay:* the interval between the time a request is received by the system and the time corresponding reply is returned by the system.

Below we present our simulation results with analysis. We have simulated our LASEC algorithm as well as DEC and CEN approaches. Each point is obtained by averaging 100 runs.

## 4.4.2 Performance Analysis

### 1) Scalability

#### *Scalability with Regards to the Network Size*

The number of messages generated by service composition protocol according to the size of the network is presented in Fig. 4.13. It considers the number of messages necessary to find first response. Our LASEC service composition mechanisms require less messages than CEN for finding first result. This is because CEN needs to pull the request from the network, which includes

generating and forwarding messages for sending the request and receiving the response, while LASEC takes advantage of localized interactions. Moreover, we observe that LASEC scales very well with the growing size of the network while CEN imposes larger cost when the network grows. DEC also scales very well, since it does not collect global information and considers only some limited neighborhood. However, for each interaction between service coordinator and prospective service provider it sends more messages (query, reply, ack, confirmation). Similar results can be observed for the time necessary to find first answer, according to the size of the network, which is presented in Fig. 4.17. As can be seen from the results, our LASEC algorithm performs better than DEC and CEN. Since in CEN messages are flooded in the network, the collision effect causes increase in delay. LASEC experiences lower rate of collisions, due to forwarding node selection. Similar as with message cost, LASEC and DEC scale better than CEN.

#### *Scalability with Regards to the Request Complexity*

Fig. 4.15 gives the number of messages generated by our service composition protocol according to the size of the request. It shows that number of messages increases with the complexity of the request. LASEC algorithm takes advantage of the localized approach and due to forwarding the messages only on the SCB. LASEC avoids flooding the network with messages. However, we observe, that along with the growth of request complexity, LASEC and DEC generate more messages, while CEN is more scalable. Since CEN collects global information, it collects the same amount of information for different request complexity, and incurs only a little additional cost for computing the result, while LASEC and DEC cost depend on the complexity result. Fig. 4.16 gives the time necessary to find first answer, according to the size of the request. Our LASEC service composition mechanism performs better than DEC and CEN for finding first result. This is possible due to localized nature of discovery of services to be composed. If service providers are close to each other they can compose service faster than it takes for the central entity to collect information from the environment. Moreover, DEC, CEN and LASEC perform similar in terms of

scalability. CEN requires longer time for more complex requests. Although the time for collecting the information from the environment is similar for different requests, however CEN will need additional time for evaluating the information and making decision.

Fig.s 4.17 and 4.18 show the time and message overhead necessary to find optimum answer, according to the service density in the environment. In these simulations we have explored localized characteristics of our approach and allowed nodes to forward message to 2 hop neighbours only. We observe that response delay as well as number of messages increase with service density. Our LASEC service composition mechanism has smaller over-head than DEC and CEN for finding the result.

## 2) Composition Locality

Fig. 4.18 shows the result of our simulation. For the request of 6 types of services two possible compositions are shown.

In pervasive computing environments composition locality refers to the distance between the services used in composition and it is required that it should be as small as possible. Composition locality is an important factor when the tasks include services which need to interact with each other and if they are embedded on different nodes in the network. To minimize such cost of composition all atomic services in the composed service need to be located as close to each other as possible. Through simulation we have observed that LASEC tends to select compositions with high degree of composition locality.

### 4.4.3 Prototype Implementation

#### System Architecture

We developed a prototype for our service composition approach to study its performance in the real environments. The system architecture for the prototype is shown in Fig. 4.20. In order to support flexible interaction between the application programmer and our service composition middleware, our prototype mainly consists of two layers. On the top layer, the application user will provide

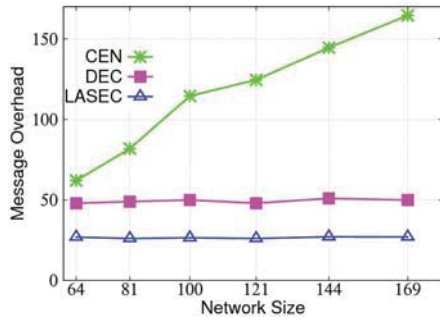


Figure 4.13: Impact of network size on message overhead.

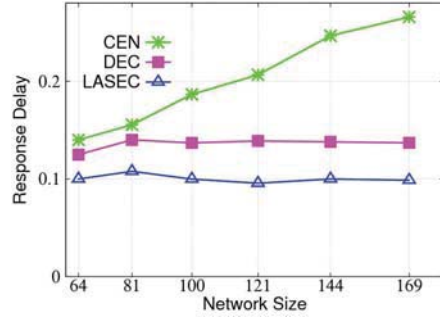


Figure 4.14: Impact of network size on response delay.

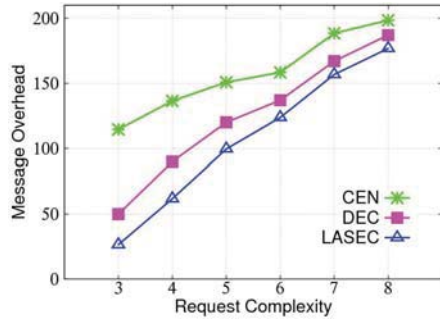


Figure 4.15: Impact of request complexity on message overhead.

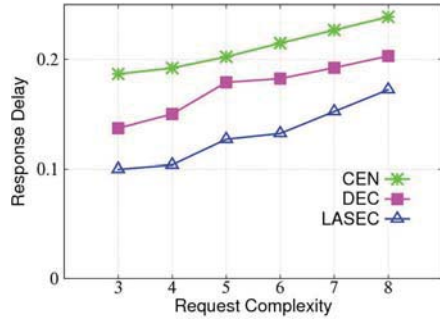


Figure 4.16: Impact of request complexity on response delay.

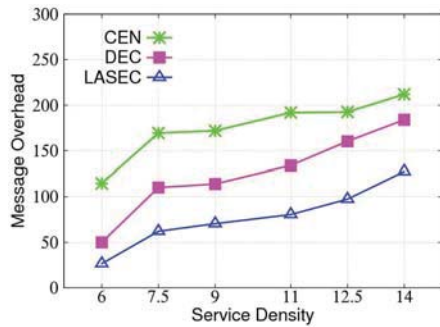


Figure 4.17: Impact of service density on message overhead.

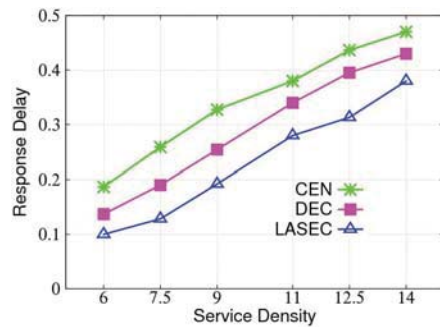


Figure 4.18: Impact of service density on response delay.

()

an XML file which defines service composition. Then the file will be parsed and packetized for dissemination into the pervasive computing environment. On the bottom layer, after the devices receive the service specification they will start executing the service composition algorithm and notify the user when the composition process is complete. As shown in Fig. 4.19(a)- 4.19(c), we used 40 MicaZ nodes with a MIB600 gateway for the hardware and deployed them in an indoor environment.

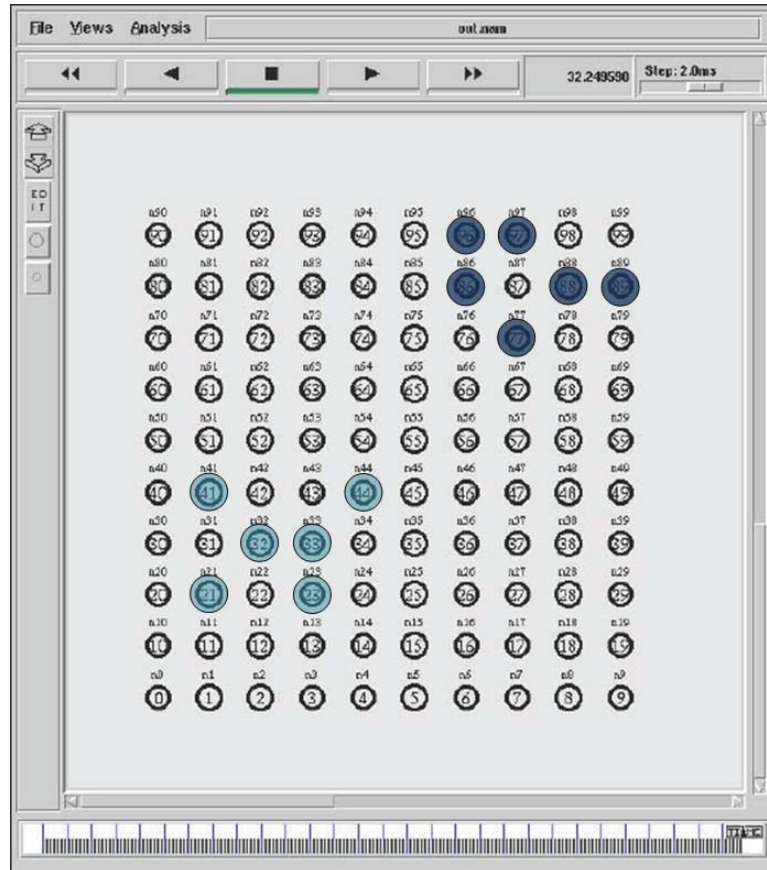


Figure 4.18: Composition locality. Atomic services in the composed service should be close to each other.

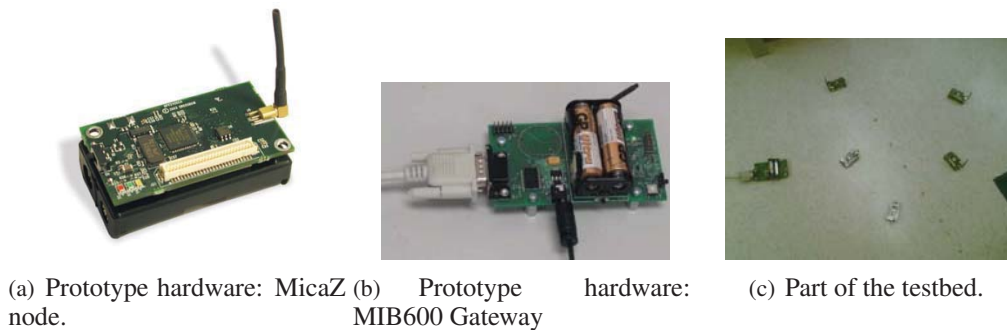


Figure 4.19: Prototype implementation.

## Experiments

We conducted experiments based on our prototype. In order to enlarge the number of services in the prototype, we implemented several services on one node and pre-defined a number of services for each sensor node in our testbed. Similar to our simulation, we use CEN as a reference for performance comparison. We implemented CEN based on existing multihop routing protocols

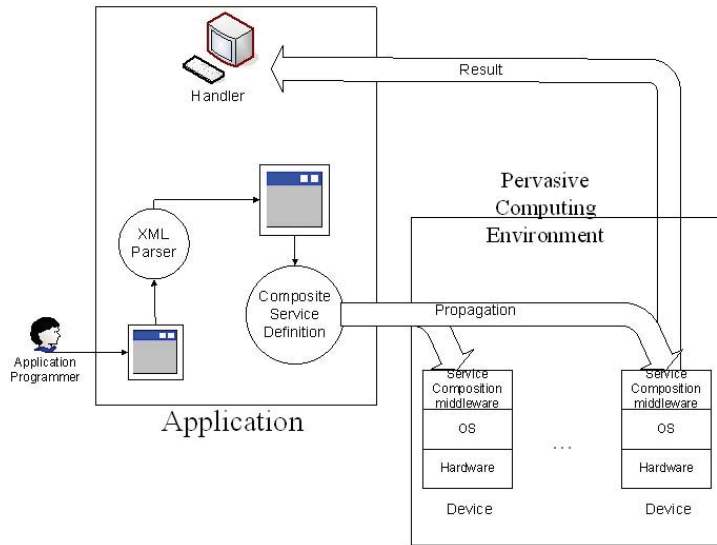


Figure 4.20: System architecture.

provided by TinyOS [35] where the nodes will simply send their available services to sink for composition. We use message cost and response delay as metrics for the experiments and study the performance according to the size of the request.

The experimental results for message cost are shown in the Fig. 4.21(a) and in the Fig. 4.21(b). Data in the Fig. 4.21(a) are obtained from experiments on 10 pre-defined services while the data in the Fig. 4.21(b) are obtained from experiments on 20 pre-defined services. Compared to CEN, LASEC can reduce the message cost by over 50 percent. This is primarily because of LASEC's decentralized service composition. Unlike CEN where all the messages must be transmitted to the sink, a lot of messages in LASEC are processed in-network. In addition, LASEC also scale better when the number of services increases. This is because when the number of services increases, the number of messages to be sent to the sink in CEN will increase accordingly. This inevitably causes a lot of packet collisions and message re-transmission. We believe such cost saving is of particular significance when considering the fact that many devices in pervasive computing environment are battery-powered.

The experimental results for message delay are illustrated in the Fig. 4.22(a) and in the Fig. 4.22(b). Similarly, the data in the Fig. 4.22(a) are obtained from experiments on 10 pre-defined

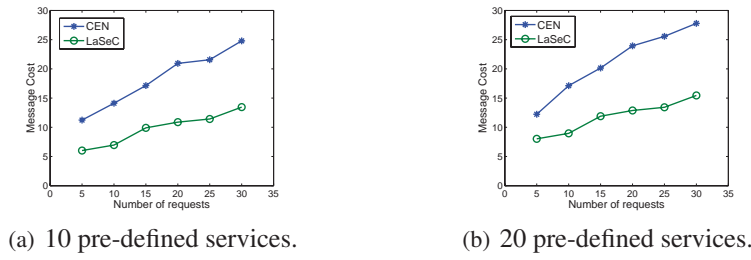


Figure 4.21: Experimental results for message cost.

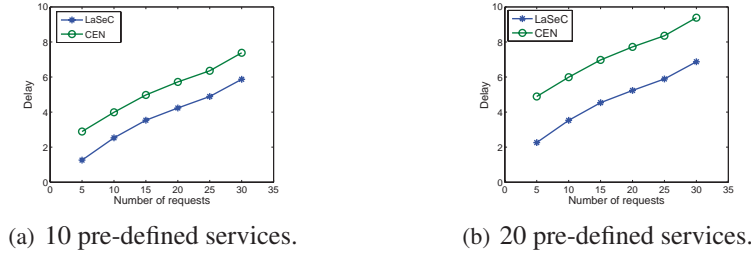


Figure 4.22: Experimental results for message delay.

services while the data in the Fig. 4.22(b) are obtained from experiments on 20 pre-defined services. We can see that on average, LASEC achieves smaller delay, especially when there are more services in the environment. This is because in CEN, the delay is primarily introduced by message collision and packet retransmission when the devices try to deliver messages to the sink. On the other hand, the decentralized service composition in LASEC allows more data to be processed in-network and hence, reduces the chance of packet collision.

## 4.5 Summary

In this chapter, we have proposed a localized approach to service composition, named LASEC. The proposed approach is suitable for service composition applications deployed in environments which are large scale and highly dynamic. Comparing with existing distributed and pull-based centralized techniques LASEC is more scalable, achieves higher degree of composition locality and copes better with dynamic environments. This is achieved by utilizing localized interactions between service providers. In LASEC, no coordinator collects global information, neither about environment nor about full solution. Nodes and links between them are mapped in subsequent stages and nodes construct only local part of the solution, while together achieving global goal.





## Chapter 5

# LASER: Localized Approach to Composed Service Reconfiguration

Dynamicity of the environments is one of the major challenges in pervasive computing. Service providers may leave environments due to mobility, unexpected power off or failures. In a pervasive computing environment, even when once the initial composition is identified and a service provisioning is established, the dynamicity of the environment may require change of the composed solution, which is called service recovery. In addition to dynamicity of the environment, the context in which composed services are deployed changes dynamically. For example user intent may change during execution of the service. Moreover, the requirement may be changed according to new environment conditions. Composed service must adapt to such changes. Reconfiguration of composed service refers to the capability of a service to adapt to changing user needs and environment conditions. Such an adaptation requires the detection of changes, but also the selection of new service configuration.

The decision about a recomposition is difficult. An adaptation of one composite service may influence another composite service. Frequent recomposition is unfavourable for the performance and also for the user experience. Majority of existing service composition frameworks consider design-time adaptation, which ends, once a service is composed. Due to characteristics of pervasive computing environments this approach is not sufficient. In PvCEs, availability of services changes frequently due to failures of devices, as well as mobility of users and devices. Therefore,

we need run-time mechanisms for service recovery and reconfiguration. The goal is to always deliver the best possible composite service. We propose a localized approach to supporting recovery and reconfiguration of composed service. We conduct extensive simulations to study the performance of the proposed LASER compared with existing techniques.

## **5.1 System Model and Preliminaries**

In this section, we describe our system that enables the devices to maintain provisioning of requested service by monitoring their local neighbourhoods as well as user requests. When considering recovery or reconfiguration, devices decide only based on the information of the devices within its k-neighborhood.

The system model of the proposed approach is illustrated in Fig. 5.1. The bottom layer consists of a finite collection of service providers, which are k-hop neighbours of the given node. The top layer shows the user-specified service description, which can be accessed by the service node. We assume that all service providers in the environment can access to a whiteboard showing the user-specified service description, which is a set of component functionalities together with composition relationships.

Middle layer is implemented on each node and its task is to enable service recovery and reconfiguration. There are two components in this layer. Service Monitors are responsible for detecting changes in either user intent or environment. Changes in environment can refer to detection of service failures as well as discovery of better suited services. Recovery and Reconfiguration component performs actions related to finding new collaborating services, migrating service to better nodes or adjusting service according to the new requirements.

### **5.1.1 Service Monitors**

In order to successfully perform recovery or reconfiguration of the composed service, first the system needs to decide when it is necessary to invoke such functions. In general, recovery and

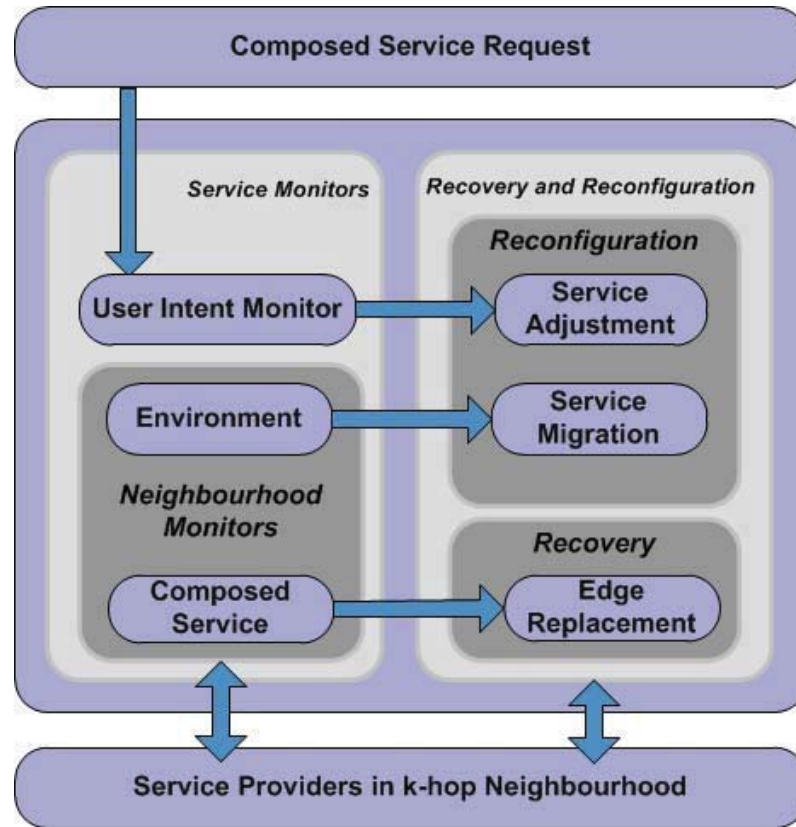


Figure 5.1: Localized service recovery and reconfiguration system.

reconfiguration should be invoked upon detected changes regarding user intent (new needs), or changes related to service providers, such as environment (new opportunities) or currently executed composed service (failure handling). Therefore, there are two kinds of service monitors: *Neighbourhood Monitors* and *User Intent Monitor*.

First we describe *Neighbourhood Monitors*. The task of these components is to monitor the changes in the services available in the local environment of the node. Two types of information are monitored.

*Environment Monitors*. The new availability of a useful service is also monitored. This may happen if the node moves and comes into reach of a new service. If a new service is available this may lead to a reconfiguration of an existing composite service. First, decision is necessary if a new service is better suitable. In such case, service migration to the new service provider is initiated.

*Composed Service Monitors.* Each node which participates in the composed service maintains edges with its direct neighbours from the user request. Therefore it is responsible for monitoring the status of these neighbours. If any of them fail the node starts Edge Replacement process.

The second type of service monitors is *User Intent Monitor*. This component monitors the changes in the user intent. If user intent changes two cases are possible, either currently composed service is sufficient to satisfy the new user intent, or the service composition needs to be adjusted.

### 5.1.2 Recovery and Reconfiguration

The task of these components is to perform actions in response to the changes detected by the Service Monitors.

Recovery refers to actions that need to be taken upon discovery of failures in the execution of the composed service due to the sudden unavailability of the already discovered individual services. *Edge Replacement* is a process triggered in such case.

Reconfiguration refers to the voluntarily replacement of the part or the whole of the composed service. *Service Migration* refers to the improvement of the composed service upon discovery of better suited services in the environment, while *Service Adjustment* is triggered in response to detecting changes in the composed service requirement.

In the following sections, we propose a set of protocols for the nodes to collaborate in service recovery and reconfiguration process.

### 5.1.3 Consistency Requirements

A major reconfiguration requirement is preservation of composed service consistency. If this is ignored, composed service can become useless. The composed service after reconfiguration must be left in a correct state. Correctness of the composed service is defined by following aspects of consistency preservation requirements:

1. The composed service satisfies its composition reliability requirements,

2. The service providers in the composed service are in mutually consistent states, and

### **Compositional reliability**

Compositional reliability requirements constrain the configuration of a composed service in terms of the relationships between service providers and the ways in which these they might be put together. Reconfiguration may affect the compositional reliability of the whole composed service, so that corrective measures must be taken.

For example, let us consider the replacement of one service provider by new provider with better QoS. Other services in the composed service which collaborate in the composed service should be capable of invoking the functions of new service provider during reconfiguration and after reconfiguration has taken place.

This implies that two conditions on the compositional reliability of the system must hold: (i) the re-replacing service provider must satisfy the interface definition of the original service provider, and (ii) the collaborating services should be able to access the service provided by the new service provider through the interface, i.e., they should know how to refer to the new service provider.

In this Chapter we do not focus on this issue. We will address this requirement in Chapter 6.

### **Mutually consistent states**

Service providers in a composed service need to be in mutually consistent states if they are to interact successfully with each other. Service providers are said to be in mutually consistent states, if each interaction between them, on completion, results in a transition between well defined and consistent states for the parts involved. Interactions are the only means by which service providers can affect each others state.

In order to provide an example, we can consider that service provider A invokes a function on B. Service providers A and B are said to be in mutually consistent states if A and B have the same assumptions on the result of the interactions between them. To be more specific, either both of them perceive that an invocation has occurred successfully, or both of them perceive that the

invocation has failed. Suppose it was decided in the reconfiguration process to replace B by B\_new after A initiated a function on B. For the resulting composed service to be in a consistent state, either (i) the invocation has to be aborted, A is informed and synchronization is maintained; or (ii) B receives the request, finishes processing it and sends the response, and then is replaced by B\_new; or, (iii) B is replaced by B\_new, and B\_new has to honour the invocation, by processing the request and sending a response to A. In case none of these alternatives occur, A might be kept waiting for a response forever.

#### **5.1.4 Protocol Preliminaries**

The result of a successful composition is a set of service providers S, where each service provider is responsible to provide atomic functionality requested by user in the request graph GR. Therefore, each node in the graph GR corresponds to one service provider in S. If one or more service providers from S becomes unavailable, then corresponding part of the GR must be recomposed. Based on our localized approach, services responsible for new composition, are only immediate neighbours of the unavailable service provider: that means service providers that corresponds to the nodes in GR which share links with the node corresponding to unavailable service provider.

For example, in original composition following set of service providers has been identified: F1, E1, D1, I1, A1, B1, C1, H1. Due to sudden unavailability of service provider B1, the service cannot be fully composed. However, only path between service providers A1 and C1 has been affected and only this path requires repair. Service provider A1 will initiate our localized algorithm, including all nodes apart from that of service of type D. Recomposition completes after finding such service provider of type B, which is both neighbour of service provider A1 and C1. This is because remaining part of composition does not need to be recomposed.

Each node maintains a set of edges, selected according to method presented section 5.2.1. Each maintained edge corresponds to the edge adjacent to this node in the request graph. Children of a given node are such nodes which can provide service type that in the request consumes the

output of the service type provided by the given node. Example is shown in Fig. 5.2. For such a requirement, a node which provides service A maintains one edge and needs to monitor one node on the other end of the edge. In the same example, node that provides service B maintains two edges, towards nodes D and E, and consequently needs monitor these two nodes. Nodes providing services E and F have no edges to be maintained; therefore they will monitor no nodes.

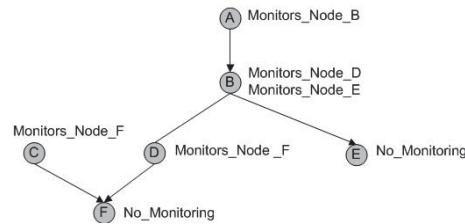


Figure 5.2: Monitoring relationship in the request.

Based on such implemented service monitoring, nodes then can make decision on necessary actions for service recovery and reconfiguration as shown earlier. Our LASEC algorithm presented in previous chapter can be applied for this purpose. After new service provider has been selected and wants to join the execution of the composed service, the consistency must be maintained.

Our reconfiguration platform provides mechanisms to transform composed service with service providers in mutually consistent states into new composed service that maintains this mutual consistency. The concept of consistent service reconfiguration is presented in Fig. 5.3. We provide capabilities for coping with the temporary interruption of interactions and for continuing these interactions in new composition.

Reliable service composition is achieved by service recovery. Service recovery is a set of operations that reconfigures composed service from one consistent state to another. Service recovery provides the ACID properties:

1. *Atomicity*. The changes during service recovery are atomic: either all operations that are part of the recovery occur or none occurs.
2. *Consistency*. Service recovery reconfigures composed service between consistent states.

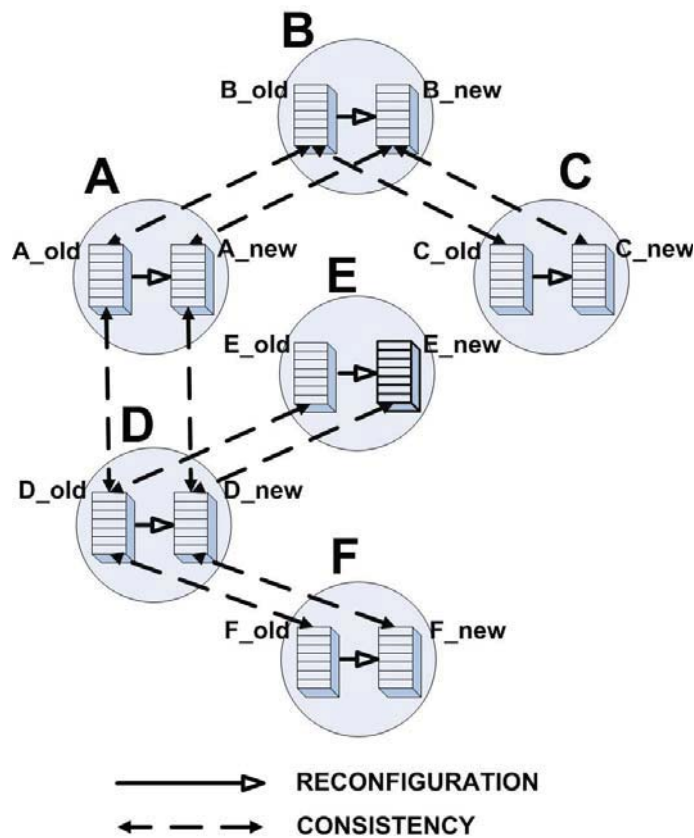


Figure 5.3: Consistent service reconfiguration.

3. *Isolation*. Even though service recoveries can be executed concurrently, no service recovery sees another service recovery work in progress. The service recoveries seem to run serially.
4. *Durability*. After a service recovery completes successfully, its changes survive subsequent failures.

In order to guarantee that mutual consistency is preserved after reconfiguration, we restrict the reconfiguration to be allowed to start only when the composed service is in the reconfiguration-safe state. Reconfiguration-safe state means that failed service providers should not leave unfinished interactions. When the service provider failed, the interactions have been aborted; therefore the composed service should recover from abortions. There is a need for rollback mechanisms to recover from abortion without entering error states. Localized service recovery system maintains the ACID properties by using two mechanisms:



1. Recoverable services. Recoverable services log their actions and therefore can restore earlier states if a failure occurs.
2. A commit protocol. A commit protocol allows multiple service providers to coordinate the committing or aborting of a partial executions of the composed service.

This is very challenging in the large scale and highly dynamic environments. Existing commit protocols for distributed transactions require coordinator to collect global information and make abort/commit decision, therefore cannot be directly used in service reconfiguration in pervasive computing scenarios. We propose a localized approach to improve scalability of commit protocol by avoiding collecting global information. In our approach nodes communicate only with their local neighbours and none of them knows or gathers global information. Because of this lack of global knowledge, problem of hidden abort arises, which we define in this section. Problem of hidden abort in localized commit protocol is shown in Fig. 5.4. We develop a technique called gradual reveal to help nodes avoid making premature decisions on commit. We achieve that by coupling of Vote and Completion phases of commit protocol. The abort in the distant node gradually reveals itself to the interested nodes. This solution is described in section 5.2.2

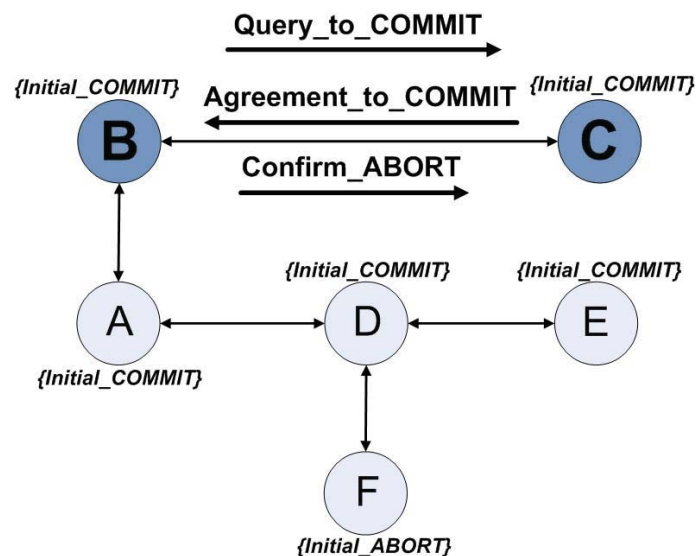


Figure 5.4: Example of hidden abort.

## 5.2 The Proposed LASER Algorithm

In this section, we describe our LASER algorithm that enables the devices to cooperatively construct the requested service. When constructing the desired overlay graph, device candidates decide with whom to cooperate only based on the information of the devices within its k-neighborhood. We propose a protocol for the nodes to collaborate in service reconfiguration process. The main operation the protocol is broadcast when some nodes try to find other nodes to cooperate in service reconfiguration process. To reduce redundant broadcast, we modify MPR based technique [27] in two respects. First, we propose a new forwarding node selection rule to prioritize nodes based on their possible contribution to the final solution. Second, we allow re-use of selected forwarding nodes by other nodes broadcasting for the collaboration in the same user request.

### 5.2.1 Service Monitors

Localized algorithm for Forwarding Node Selection is summarized in Fig. 5.5. While executing our algorithm, each node  $i$  exchanges messages listed in 5.1 and maintains necessary data structures listed in Table 5.2.

Table 5.1: Message Types in Forwarding Node Selection

Message	Purpose
$Req(id, service\ type, if\ FNS)$	Node $i$ requests from 1-hop neighbors their id and service type and if they were chosen as forwarding node by other broadcasting nodes
$Req(1-hop\_n)$	Node $i$ requests from 1-hop neighbors their set of all 1-hop neighbors

At the beginning node  $i$  collects id and neighborhood information from its 1-hop neighbors. Upon receiving information about 1-hop neighbors of its 1-hop neighbors, node  $i$  builds information about its 2-hop neighborhood. The information includes service type of neighbors and if they already have been nominated by their neighbors to be in their Forwarding Node Set (FNS). Node

```

1  INPUT: 1-hopn, idi,
2  OUTPUT: FNS
3
4  // The code executed by broadcasting node i
5  {
6  Send Req(id, service type, if FNS) to 1-hopn;
7  Send Req(1-hopn) to 1-hopn;
8  if (node k is the 1-hop neighbor that provides service
9     type specified in SCG) {
10     FNS ← node k;
11  }
12  Not_Covered ← 2-hop neighbors that are not
13     connected with node i through node from FNS
14  N ← nodes from Not_Covered that are connected
15     with node i only through one 1-hop neighbor;
16  if (node k is the only 1-hop neighbor that connects
17     node i with the 2-hop neighbor from set N) {
18     FNS ← node k;
19  }
20  remove N from Not_Covered
21  repeat if (node k connects largest number of nodes
22     from Not_Covered) {
23     FNS ← node k; until (Not_Covered = 0)
24  }
25 }

```

Figure 5.5: Forwarding Node Selection Algorithm.

*i* selects a subset *N* of 1-hop neighbors that can provide any service type in requested SCG and puts this nodes to FNS and marks which 2-hop neighbors are connected. Next, from remaining unconnected 2-hop neighbors, node *i* selects a subset *N* of 2-hop neighbors that are connected with node *i* only through one 1-hop neighbor. If node *k* is the only 1-hop neighbor that connects node *i* with the 2-hop neighbor from set *N*, node *i* puts node *k* in the FNS. Next, node *i* examines if all 2-hop neighbors are connected. Node *i* selects the 1-hop neighbor that connects the largest number of two-hop neighbors that are not yet connected. If necessary, node *i* repeats this step until all 2-hop neighbors are connected. When node *i* broadcast messages to its 1-hop neighbors, it also informs those that are in FNS.

Table 5.2: Data Structures in Forwarding Node Selection

Variable	Meaning
$id_i$	Unique id of node $i$
$id_i(1-hop_n)$	Set of id's of 1-hop neighbors of node $i$
$1-hop_n$	Set of all 1-hop neighbors of node $i$
$2-hop_n$	Set of all 2-hop neighbors of node $i$
$N$	Set of 2-hop neighbors that are connected with node $i$ only through one 1-hop neighbor.
$Not\_Covered$	Set of 2-hop neighbors that are not connected with node $i$ through node from FNS
$FNS$	Set of 1-hop neighbors that connect node $i$ with its all 2-hop neighbors

### 5.2.2 Localized Commit Protocol

In this section, we describe our algorithm that enables the devices executing the composed service to cooperatively commit partial executions of the service. It is a fully decentralized algorithm, based on local information of the devices. That means when voting on the commit, device candidates communicate only with their local neighbours, yet avoiding hidden abort problem.

#### Description of the Algorithm

Unlike previous algorithms that rely on coordinator having global knowledge, in our algorithm each device only maintains local state information about its physical neighbors. In particular, devices in our algorithm exchange a set of local requests locally to vote on commit. It is summarized in Fig. 5.6. While executing our algorithm, each node  $i$  exchanges messages listed in 5.3 and maintains necessary data structures listed in Table 5.4.

*Phase I.* Each node has a policy in when it needs to commit, therefore each node in the system can become a commit initiator. Each initiator generates a message called `local_request_unfold`. Each `local_request_unfold` is set with its designated service type. Designated service type is set according to the node that succeeds the type of node  $i$  in the executed service  $\vec{G}_R$ .

*Phase II.* In the second phase, the `local_request_unfold` will be processed by physical neigh-

```

1  INPUT:  $\bar{G}_R = (V_R, \bar{E}_R)$ ,  $N(x) \subseteq \mathcal{P}$ ,
2  OUTPUT:  $\bar{G}_A = (V_A, \bar{E}_A)$ 
3
4  Find Candidate Solutions()
5
6  //The code executed by each node  $i$ 
7  {
8  status  $\leftarrow$  WAITING;
9  Wait until received  $G_R$ ,
10 if ( $in\_edges$  of  $i$  in  $G_R$  is empty) {
11     status  $\leftarrow$  INITIATOR;
12 }
13 if ( $in\_edges$  of  $i$  in  $G_R$  is not empty) {
14     status  $\leftarrow$  CANDIDATE;
15 }
16 }
17 //The code executed by each INITIATOR node
18 {
19 Send ( $local\_request\_unfold$ ) to  $out\_edges$ 
20 }
21 //The code executed by each CANDIDATE node
22 {
23 Wait until received ( $local\_request\_unfold$ );
24      $in\_edges = in\_edges - (sender)$ ;
25 if ( $in\_edges = 0$ ) {
26     Generate ( $local\_request\_unfold$ );
27     send ( $local\_request\_unfold$ ) to  $out\_edges$ ;
28     else store ( $local\_request\_unfold$ );
29 }
30 Wait until received ( $local\_request\_fold$ );
31 if (merged  $local\_request\_fold = G_R$ ) {
32     become DONE;
33     else send ( $local\_request\_fold$ ) to  $in\_edges$ ;
34 }
35 }

```

Figure 5.6: Localized service reconfiguration algorithm LASER.

Table 5.3: Message Types in LASER

Message	Purpose
$local\_request\_unfold$	Candidate node $i$ announces local request to nodes which service type succeeds the type of node $i$ in the requested service
$local\_request\_fold$	Response to the local request

bors. At each neighbor,  $local\_request\_unfold$  collects local state information from the visited device. When a  $local\_request\_unfold$  arrives at a device that is a candidate for the designated service type of that  $local\_request\_unfold$ , the  $local\_request\_unfold$  is stopped and merged with  $local\_request\_unfold$  from other incoming edges. When there was at least one  $local\_request\_unfold$

Table 5.4: Data Structures in LASER

Variable	Meaning
<i>status</i>	Status of node $i = \{WAITING, INITIATOR, CANDIDATE, ROUTING, DONE\}$
<i>out_edges</i>	Service types that succeed the type of node $i$ in the requested service $G_R$
<i>in_edges</i>	Service types that precede the type of node $i$ in the requested service $G_R$
<i>sender</i>	Node ID from which $i$ received a message

coming from all incoming *\_edges*, the device generates new *local\_request\_unfold*, with new designated service. When *local\_request\_unfold* arrives at the service type that has no outgoing edge in the requested graph, it starts phase III.

*Phase III.* In the third phase, the device without out-going edges returns *local\_request\_fold* to the received *local\_request\_unfold* to its initiators. When any device receives the *local\_request\_fold* from different devices, it merges them. When merging results with a complete execution graph phase III is completed.

### 5.3 Performance Evaluation

We have carried out extensive simulations to evaluate the performance of our algorithm. Moreover, to show the advantage of our algorithm (LASER), we have also implemented a distributed approach (DEC) [13] as well as a pull-based model of the centralized algorithm for service reconfiguration (CEN) [11], and have compared their results with ours. Below, we give a brief description of the implemented algorithms.

*LASER. Localized service reconfiguration algorithm.* Our localized service reconfiguration algorithm has been applied to the network.

*DEC. Distributed embedding algorithm.* In this algorithm for each service reconfiguration coordinator is assigned, which broadcasts queries for each service type requested, available service providers respond, coordinator sends ACK message to them which is followed by confirmation



Table 5.5: Simulation Parameters

Parameters	Values					
Number of nodes, (N)	64	81	100	121	144	169
Number of service types (n <sub>s</sub> )	10	13	17	20	24	28
Service density	6, 7.5, 9, 11, 12.5, 14					
Request complexity	3, 4, 5, 6, 7, 8					
Territory scale (m <sup>2</sup> )	200					
Transmission radio range (m)	40					

from service providers.

*CEN. A centralized service reconfiguration algorithm.* In this algorithm, we provide a dedicated service directory. Initially there are no services registered with the directory. When a user specifies a composite service, it is submitted to the directory in the form of a graph. Upon receiving the request, the directory pulls from the environment the information related to service types specified in reconfiguration request. The pull operation is supported by broadcast.

In our simulations, we consider a grid network of N nodes uniformly deployed. The total number of nodes in the network is varied to examine the effect of system scale on the performance. Also we studied influence of request complexity, which in this chapter is defined as the number of services specified in the request graph. A service type is randomly assigned to each node. In each network we set the total number of service types so that the density of service types ( $ds=N/n_s$ ) is comparable. The simulation parameters are listed in Table 5.5.

We evaluated the performance of the algorithms using the following metrics:

*Message overhead:* the number of messages generated by the reconfiguration process.

*Response delay:* the interval between the time a recovery request is initiated by the system and the time recovery is completed.

*Success Rate:* The ratio of the total number of successful recovery operations to the total number of recovery operations triggered.

Below we present our simulation results. Each point is obtained by averaging 100 runs.

With our localized approach, it is possible to ensure that the request can be recomposed with

minimal overhead. Although in extreme cases of dynamic (mostly theoretical) the composed service may need to be completely recomposed, typically, it is sufficient to recompose the service partially, maintaining those services identified previously which are still available and replacing only the service that left the environment. In our simulations, we considered the scenario shown in Fig. 5.7.

The result of a successful composition is a set of service providers  $S$ , where each service provider is responsible to provide atomic functionality requested by user in the request graph. Each node in the request graph corresponds to one service provider in  $S$ . If one of the service providers from  $S$  becomes unavailable, then corresponding node in the request graph must be recomposed. Based on our localized approach, only immediate neighbors of the unavailable service provider will be responsible for a new composition. That means service providers that corresponds to the nodes in the request graph which share links with the node corresponding to unavailable service provider.

In original composition following set of service providers has been identified  $F1, E1, D1, I1, A1, B1, C1, H1$ . Due to sudden unavailability of service provider  $B1$ , the service cannot be fully composed. However we see, that only path between service providers  $A1$  and  $C1$  has been affected therefore only this path requires repair. Service provider  $A1$  will initiate our localized algorithm, including all nodes apart from that of service of type  $D$ . Recomposition completes after finding such service provider of type  $B$ , which is both neighbor of service provider  $A1$  and  $C1$ . This is because remaining part of composition does not need to be recomposed. Service provider  $C1$  maintains links leading to successful composition.

Through simulations, pictured in Fig. 5.8(a) and Fig. 5.8(b), we have observed, that re-composition cost is stable and similar to that of composing simple requests.

The results under mobility conditions are presented in Fig. 5.9(a), Fig. 5.9(b) and Fig. 5.9(c).

We show the results of message overhead on Fig. 5.9(a). Message overhead depends on the



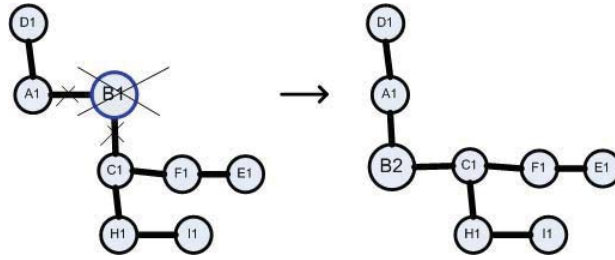


Figure 5.7: Dynamicity - one of the services becomes unavailable and it must be replaced.

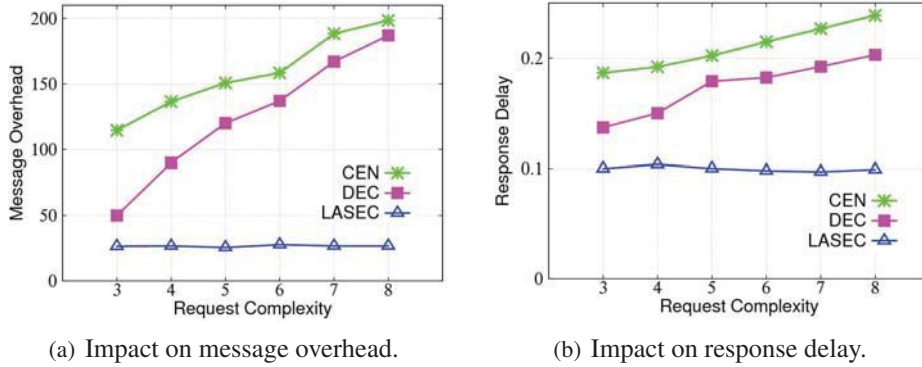


Figure 5.8: Coping with request dynamicity.

number of nodes that response to recovery request. Since in the CEN approach, global knowledge is necessary, therefore, all the nodes take part in responding to recovery. The message overhead is therefore very high and increases with increased network density. However, for the LASER, node initiating recovery, considers only local environment, and only nodes from its local neighborhood take part in the recovery process. Therefore the message overhead is not very high. It also increases with the increase in network density, but not as much as CEN.

Fig. 5.9(b) shows the change of Response Delay with increasing Network Density. We

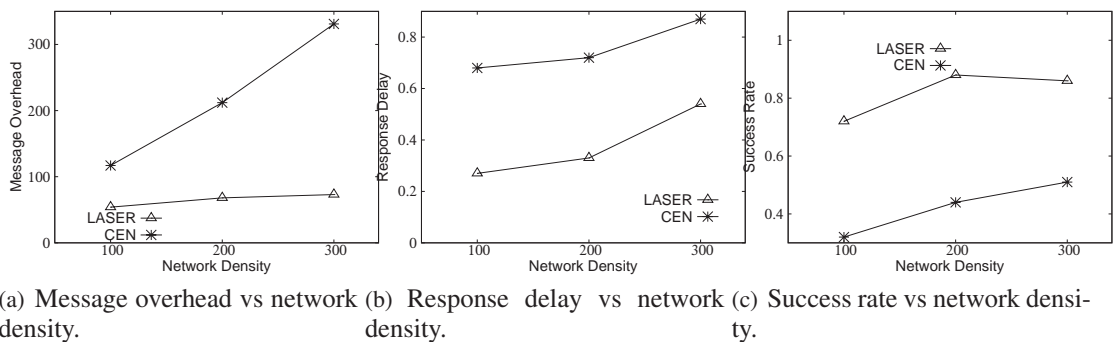


Figure 5.9: Simulation results for composed service recovery.

observe that Response Delay for CEN increases with the increase in Network Density. This is because CEN pulls information from the network upon each reconfiguration and when there are more nodes it needs to process more information. Similar in LASER, the sparse network means that each node does not have many neighbors, so it does not have much choice, while increased network density means it must process more messages and the time will increase. In any case, LASER is better than CEN because it explores smaller amount of nodes.

Fig. 5.9(c) shows the variation of Success Rate with increasing Network Density for the composed service recovery. For our LASER, Success Rate is high and initially increases with Network Density and then decreases with further increase in Network Density. This is because at lower node density the recovery is often unsuccessful due to network disconnection. The success rate increases as the network grows denser. However, at very high network density the high recovery delay may render the recovery unsuccessful. On the other hand, CEN has quite low values of Success Rate and increases with node density. This is because CEN has global value of the current environment, but upon moving to another environment it either has no knowledge, or else has to start the discovery process from the beginning. The success rate upon moving is not zero, because we defined movement to the new environment as partially overlapping with the previous one. Therefore, with increase Network Density, CEN has more service available in overlapping region, therefore its Success Rate improves. However, our LASER approach achieves better success for scenarios with mobility.

## **5.4 Summary**

In this chapter, we have proposed a localized approach to composed service reconfiguration, named LASER. The proposed approach is suitable for applications deployed in environments which are large scale and highly dynamic. To avoid hidden abort problem and creating false edges, we have proposed a new technique called Informed Acknowledge Technique. In LASER, no coordinator collects global information, nodes and links between them are mapped in subse-

quent stages and nodes construct only local part of the solution, while together achieving global goal. Comparing with existing decentralized and pull-based centralized approach, the proposed localized algorithm is delay-constrained and message-efficient.



## Chapter 6

# LASEH: Localized Approach to Service Heterogeneity

Although services itself aim to hide the heterogeneity of the underlying hardware by abstracting its functions, however, they are implemented by different developers, without any collaboration, which results in different technologies being utilised and lack of interaction capabilities between them. Service composition mechanism should be designed to be loosely coupled and any service should be able to collaborate with other services. However, in practice, collaboration of services is difficult to be achieved due to their different implementation. In such dynamic and large-scale environment as PvCE, heterogeneity is particularly high. Services may be hosted on range of platforms, from resource-rich fixed machines to wireless resource-constrained devices, to any physical object enhanced with some networking capacity thus turned into a UIO. Because of the platform heterogeneity, various protocols are available. For example, most common implementation in the internet are Web Services. However, other implementations exist as well, suited to different environments, for example OSGi, JINI, and UPnP. Service composition for PvCE should be able to provide collaboration for different services, not depending of their implementation. There is a need for mechanisms to provide the interoperability in order to support the composition of the services from different domains. In addition to improving collaboration between services this will have direct impact on quality of composed services. First, more composed services will be available in the environment, and they will be more diverse. The mechanisms presented in previous chapters

(LASEC and LASER), will be able to utilize more services for the particular services composition, therefore with increased choice, the quality of the composed service will be potentially improved as well. Because the choice of services to participate in the composed service will not be limited by the particular implementation, the performance of the composed service will be improved as well. Reconfiguration will be aided by having bigger choice of different implementation of the same service.

In order to cope with the problem of heterogeneity in service composition we studied how to provide service composition across different environments supported by different service management systems, which may use standard protocols as well as tailor-made mechanisms. As mentioned earlier, an important part of any service composition framework is identifying appropriate service providers to contribute to the composed service, which is called service discovery. The previous chapters focused on the mechanisms for higher levels of the composition process, namely selection of the services for the composed service (LASEC) and maintaining composed service in dynamic environments (LASER). In this chapter we focus on the multiprotocol service discovery to aid the higher levels of the composition process.

Service discovery is an important and challenging issue in pervasive computing. To date, many service discovery protocols have been proposed and new ones are under development. However, pervasive computing involves applications running in heterogeneous environments and application developers must cope with diversity of network infrastructures, middleware platforms, and hardware devices. There is a need for integrating or bridging these service discovery schemes in order to support the discovery of the services available in different environments without compromising their functionality. In particular, the support should benefit the localized approach to service composition. It should be lightweight and scalable. One approach is to provide all the possible translations between protocols, but doing so in the localized system would require to provide all the translations on each service, and therefore would be too heavy and not flexible. It is not feasible

that a single service will have a full knowledge about all the possible service protocols with which it may need to interact, including the ones developed in the future. Therefore, we first abstract the common functionalities of the service discovery process and provide on each service only one translation, between its native discovery protocol and our abstractions. We propose the Universal Adaptor (UA) approach, which consists of two major components: the Universal Adaptor Primitives (UAP) and the Universal Adaptor Mapping (UAM). UAP is the universal set of primitives used by the user to discover the services across different environments, while UAM provides the mapping between the Universal Adaptor Primitives to the primitives used in various service discovery systems. We have developed a prototype of the proposed approach and we describe the implementation issues and the experiment results.

The rest of the chapter is organized as follows. Section 6.1 presents a brief overview of our motivation and the issues that are required to be addressed for interoperability system for pervasive computing as well as gives some background information about the previous endeavors to realize it. In Section 6.2, we describe our system model. Next, in Section 6.3, we describe the design of the proposed Universal Adaptor approach. We have evaluated the performance of the proposed approach using prototype in Section 6.4.1.

## 6.1 Background

Along with the advances in pervasive computing technologies, users are no longer constrained to only a single computing environment but can work across different environments, meeting with a diversity of software, hardware devices, and network infrastructures. Heterogeneity of the environments brings significant problems that application developers must cope with. For instance, devices must be able to discover and share services among each other. However, manual configuration may require special skills together with long time to set up. Therefore, service discovery is an important, challenging task in pervasive computing.

Many service discovery protocols have been proposed. Examples include UPnP, SLP, and Jini

[1, 2, 3]. They allow automatic detection of hardware devices, software, and other resources in a computing environment along with services offered by the various kinds of entities. Existing service discovery protocols differ in the way service discovery is performed, and no single protocol is suitable for all the environments. Due to the differences in the service discovery approaches implemented, a user working in one environment may not be able to search for the services available in another environment that suits the users need. To support service discovery across different environments, new techniques are needed to integrate or bridge the service discovery protocols used in a diversity of environments.

Service discovery protocols have been proposed for different applications in various environments [5, 3, 1, 4, 2]. Survey and comparisons between the existing service discovery protocols can be found in [6, 7, 8, 9, 10]. New service discovery protocols targeting at pervasive environments are also emerging [16, 18, 19]. In this section, we focus on the works that support interoperability of the service discovery protocols. Several approaches on supporting multi-protocol service discovery have been proposed [11, 12, 13, 14, 15, 16].

All the existing service discovery mechanisms realize the concept of client/server application. Clients are entities that need some functionality (service) and servers are entities existing in the environment and offering this functionality. Service discovery is the framework for connecting clients with services. Existing works towards achieving multi-protocol service discovery use a middleware approach but differ in where the interoperability support is provided. The middleware can be on the service side, client side, or intermediate entity.

An example of providing the support on the service side is INDISS [14] designed for home networks. INDISS provides parsers and composers, which decompose a request from the source service discovery protocol into a set of events and then compose them into message understood by target service discovery protocol.

INDISS can also be deployed on the client side. Another example of middleware on the client



side is ReMMoC [13]. In ReMMoC, the client side framework provides the mappings between all the supported service discovery mechanisms. Abstraction of discovery protocols to the generic service discovery is achieved by using a generic API or doing discovery transparently to the client. In this approach, all possible mappings need to be provided at the client side.

Service side support can also be based on service proxies [16]. When a new service appears or disappears in one of the environments, the framework detects it and creates or removes its proxies from other environments. However, this approach requires dynamic service proxies to be implemented for each environment, which increases developers workload.

Another way of providing interoperability support is to provide an intermediate entity [12, 11]. In Open Service Gateway Initiative OSGi [12] all the connections and communications between the devices are brokered by a Java-based platform. An internal service registry exists in the framework. Interoperability is achieved by providing an API to map the given service discovery protocol to OSGi and vice versa. Supporting new service discovery protocols requires defining new APIs for them. Gateway functionality is also utilized by protocol adapters in the FuegoCore Service broker [11] designed for mobile computing environments. The service broker registers the mappings between its internal template and the external templates used by different service discovery protocols. Extending the FuegoCore service broker to new service discovery protocols requires creating and deploying additional service discovery protocol adapters. INDISS [14], mentioned above, can be deployed as an intermediate entity as well. The intermediate entity approach requires broker to integrate all the adapters into one system. In a network with a large number of service discovery protocols the framework may not be scalable.

Another approach is providing service that discovers services across different environments [15]. In MUSDAC, it registers itself in all the environments, so clients can use whatever protocol to discover it. However, clients must have the knowledge about MUSDAC and the process of discovering the service has high processing requirements.

The centralized approach has the scalability problem. In the future pervasive environment, an unlimited number of service discovery mechanisms will emerge.

Providing all possible translations in one middleware would result in a very heavy system. Our work reported in this chapter is based on the analysis of the following requirements on the localized interoperability system for pervasive computing:

- no change should be imposed on the existing service discovery mechanisms,
- no change should be imposed on the services registered in domains - no functionality of the environment should be compromised,
- the system should be lightweight, scalable, and extendable,
- support both standard and tailor made service discovery mechanisms,

Our approach addresses all of these requirements. In this chapter, we study how to provide service discovery across different environments supported by different service discovery systems, which may use standard protocols such as SLP and Jini, as well as tailor-made mechanisms that support multiple protocols within an environment, such as ReMMoC [13]. We propose the approach, which can be implemented in any environment, independent of the service discovery system used in that environment. This approach enables users to discover available services with no knowledge of the service discovery system adopted in an environment. Comparing with existing approaches, Universal Adaptor provides a simple and flexible solution. It provides only a single set of APIs and supports not only all existing but also future service discovery systems. It is lightweight and easy to implement in diverse infrastructures and to use by users. In the following sections we describe design of the system.

## **6.2 System Model**

The aim of our study is to support interoperability between existing as well as yet unknown service discovery systems.

In this section, we describe the system model of the proposed UA approach. We define an *Environment* as a Service Discovery Domain, where a native service discovery system is able to find a specified resources if they are available in the domain. A native service discovery system can take the form of an existing service discovery protocol, e.g SLP, Jini. Moreover, it can be a tailor-made mechanism, e.g. ReMMoC which supports the interoperability of service discovery within an environment.

Services in an environment can be provided by hardware devices, software, and other entities. Examples of software services can be weather forecast, stock quotes, and language translation. Hardware devices can provide services directly or via another device. For example, the printing service can be provided by a Jini enabled printer, or by a printer that is attached to computer running Jini software.

Fig. 6.1 shows the major entities in the UA system. There are two major components: The Universal Adaptor Primitives (UAP) and the Universal Adaptor Mapping (UAM). UAP acts as uniform interface to the client. The client makes use of UAP to discover and access services. The main function of UAM, on the other hand, is to provide the mapping from UAP to the specific primitives of service discovery protocols (SDP).

The Universal Adaptor (UA) is installed in each environment. On the client side, the UA provides Universal Adaptor Primitives (UAP) for the communication between the client and UA. On the service side, UA is a component tailored to the environment, mapping the UA primitives to those of the native service discovery system and performing service discovery on behalf of the client.

## 6.3 Universal Adaptor Approach

### 6.3.1 Universal Adaptor Primitives (UAP)

We studied existing service discovery protocols and observed that all the service discovery systems provide the same functionality for end users and differ only in the underlying models and

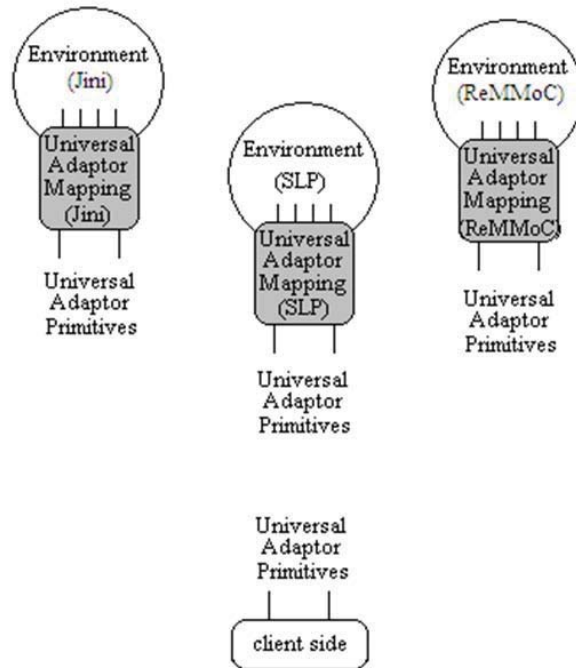


Figure 6.1: Universal Adaptor system model.

operations. We can abstract the common characteristics of existing approaches and to provide a universal set of primitives that enable service discovery across diverse environments.

The format of the proposed primitives is the following:

*[Return\_Values] Primitive\_Name [Parameters]*

*Primitive\_Name* represents the functionality requested by the client; *Parameters* is a set of attributes required in the discovery process; *Return\_Values* is set of values returned to the client.

The discovery process starts when the client expresses his interest in a particular service. Next, the discovery system searches for the service and returns result to the client. The common feature of all the existing service discovery mechanisms is that they support the process of looking for service and enable the access to the service. Therefore, we propose two universal primitives: *Discovery* and *Access*. Specific UA Primitives with parameters are shown in Fig. 6.2.

The *Discovery* primitive is of the following format:

*[RV\_ST, RV\_F, RV\_S] Discovery (Service Type, Filter, Security).*

A client requests the specific service by providing its type in the *ServiceType* parameter. The

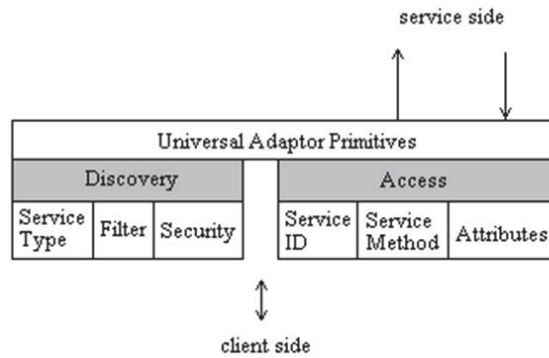


Figure 6.2: Universal Adaptor Primitives.

attributes of the service are specified by the client in the *Filter* parameter. If the authentication of the client for using the service is required, the client provides the required information in the *Security parameter*. The *Discovery* primitive provides three return values. *RV<sub>ST</sub>* is the return value containing list of the discovered services of the requested type. Each of the discovered services has a unique ID assigned by the UA. *RV<sub>F</sub>* provides a list of attributes for each of the discovered services. For example, for a printing service, an attribute can be specified to indicate whether the printer supports colour printing or only black&white. *RV<sub>S</sub>* is a return value indicating the authentication status.

After the successful discovery of the requested service, the client selects one of the returned services and access the service by using the *Access primitive*, which is of the following format:

*[RV<sub>status</sub>] Access (ServiceID, Service Method, Attributes.)*

The client provides the parameter *ServiceID*, the ID of the selected service. The *ServiceMethod* parameter specifies the specific method provided by the service. The *Attributes* parameter is a set of attributes specific to the particular service. *RV<sub>status</sub>* is the return value indicating the status of accessing the service, e.g., success or fail.

### 6.3.2 Universal Adaptor Mapping (UAM)

In this section, we describe UAM, which performs the mapping from UAP to the SDP specific primitives used in the target environment. As we mentioned before, there is a number of currently

existing service discover mechanisms, including protocols and interoperability systems. Moreover, new mechanisms are under development. Therefore, the many-to-many approach providing a mapping between the primitives of each pair of these mechanisms would result in a heavy system with difficulties in adapting to the future service discovery mechanisms. Our approach is a one-to-many approach, providing the tailored mapping between the UAP to each native protocol used in the target environment.

For the structure of components of the UAM, please refer to Fig. 6.3. More specifically, the functions of UAM components are described in Table 6.1.

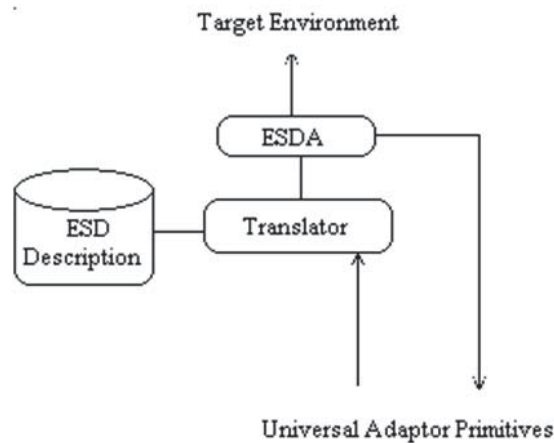


Figure 6.3: Universal Adaptor Mapping.

Recall that to discover or access a service, the client uses the UA primitives described in previous section. *Translator* is a component responsible of mapping the UA primitives to the primitives of the native protocol used in the Environment. The translation process takes place each time when a client sends a query.

*Translator* performs mapping based on the *ESD Description* (Environment Service Discovery Description), which is set of rules specifying the characteristics of the service discovery protocol used in the particular environment. Because different environments adopted diverse models at the low level, the rules used by *ESD Description* are tailored to each environment and are described in the form of the primitives extracted from the native service discovery protocol. *Translator* outputs

Table 6.1: Function of UAM components

Component	Function
<i>ESD Description</i>	Rules specifying the characteristics of the service discovery protocol used in the particular environment, in the form of the primitives extracted from the native protocol
<i>Translator</i>	Module that translates UA primitives into native primitives, and outputs an algorithm for service discovery using the native primitives
<i>ESDA</i>	Agent that uses the algorithm generated by Translator to perform service discovery in the target environment. Depending on the native service discovery protocol, ESDA can register for service advertisements or performs active service discovery

an algorithm for the service discovery using the native primitives.

*ESDA* (Environment Service Discovery Agent) is representing the client in the process of discovery in the target environment. It uses the algorithm generated by *Translator*. From the servers point of view, *ESDA* is the same as any other client in the environment using the native protocol for service discovery. All the communications take place between the service discovery protocol and *ESDA*. Another function of *ESDA* is to provide the client with the return values from environment, e.g. information about the attributes.

## 6.4 Performance Evaluation

### 6.4.1 Prototype Implementation and Experience

We have developed a prototype of our proposed UA approach. The system has two service discovery environments. One uses Jini and the other uses SLP as the native service discovery protocols. For Jini, we use Jini2\_1 [20], which is an implementation of the Jini technology from Sun Microsystems. For SLP, we use a pure Java implementation of SLP jSLP [21].

For each environment, a tailored UA is implemented. The implementation of UA for each environment consists of two parts client side UA and service (environment) side. UA, both are

written in Java. Client side UA is the same for both Jini specific implementation and SLP specific implementation. On the service side, the Java UA listens on a port for the incoming requests. It parses the request, and uses reflection mechanism to invoke the corresponding service in the local environment, then passes the result back to the client.

As an example, we implemented a weather information service using Jini and SLP separately in two environments. We measured the performance of our UA system, in terms of the overhead in time comparing with the native service discovery protocols. Experiments are conducted with a desktop computer with CPU Pentium D 2.8GHz, 1G Memory and a laptop computer with CPU Pentium Mobile 1.6GHz, 512M Memory connecting to the LAN with 100Mbps Ethernet and 11Mbps 802.11b protocol respectively.

We first measured the time for service discovery in the two environments using native service discovery clients (both for Jini and SLP). Next, we compared that with the discovery time using JiniUA and SLP UA respectively. Results are presented in Table 6.2.

Table 6.2: Discovery overhead of UA

	<b>Jini Environment</b>	<b>SLP Environment</b>
<b>Service discovery time without UA [ms]</b>	638	152
<b>Service discovery time with UA [ms]</b>	743	197
<b>Overhead [%]</b>	16	30

In the Jini environment, the average service discovery time is 638 milliseconds, and in the case of using UA, the average time is 743 milliseconds. In the SLP environment, the average service discovery time is 152 milliseconds and the average time in the case with UA support is 197 milliseconds. The overhead that UA imposes on service discovery time is mainly introduced by the socket setup and request parse.

We also measured the time for service access in the Jini and SLP environments and compared them with the access time using UA. Results are presented in Table 6.3.

In the Jini environment, the average service access time is 832 milliseconds, and in UA client,



Table 6.3: Access overhead of UA

	<b>Jini Environment</b>	<b>SLP Environment</b>
<b>Service access time without UA [ms]</b>	832	223
<b>Service access time with UA [ms]</b>	926	268
<b>Overhead [%]</b>	11	20

the average time is 926 milliseconds. In the SLP environment, the average service invoking time is 223 milliseconds, and the average time in the case with UA is 268 milliseconds. Similar to the discovery time experiments, the overhead that UA imposes service access time is related to the socket setup and request parse.

In summary, the experiment results show that the discovery and access time overhead imposed by UA is very small.

### 6.4.2 Simulation

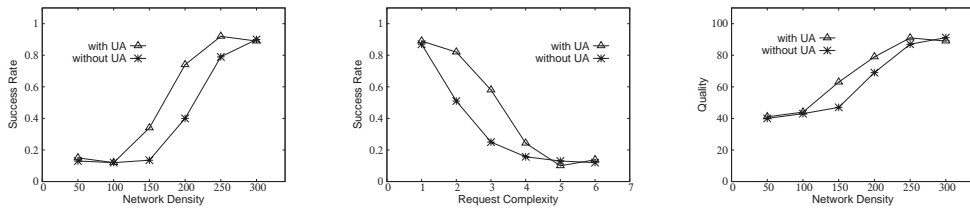
We have carried out simulations to evaluate the benefits of using our proposed mechanism (LASE-H). We have implemented two simulation environments and compared the results collected in each of them. Below, we give their brief description:

*without UA.* The first one using only native discovery protocols, that is without using our proposed UA; we have implemented two different service discovery protocols and randomly assign them to the nodes.

*withUA.* In the second environment we also implemented the same two service discovery protocols, but on top of them the UA was implemented.

In our simulations, we consider grid of  $N$  nodes. The density of the nodes in the network is varied to examine the effect of the sparsity of the network on the performance. Also, we varied the complexity of the request, to show how supporting service heterogeneity may benefit the service composition process. We evaluated the performance of the approaches using the following metrics:

*Success Rate:* The ratio of the total number of successful discovery operations to the total



(a) Success rate vs network density. (b) Success rate vs request complexity. (c) Quality vs network density.

Figure 6.4: Simulation results for success rate and quality.

number of discovery operations triggered.

*Quality of service:* The total quality of the service.

The results are presented in Fig. 6.4(a), Fig. 6.4(b) and Fig. 6.4(c).

First, we observe that for the too sparse network, neither approach with UA not without UA can achieve good results. We treat this case as a disconnected network, where there are no neighbors in the communication range of the node. Then, with initial increased density of nodes in the network, both the success rate as well as quality is improved. This proves that the more neighbors the node has the bigger success rate of finding the preferred collaboration partner. Moreover, with bigger choice comes higher quality as well. However, when the network is very dense, the improvement is not visible any more. Probably, because network became too dense and additional neighbors available for the node to choose from are treated as duplication.

We next observe that approach with UA achieves better success rate and quality than without UA. This again is related to number of neighbors visible to the node. Since a node can only discover and collaborate with neighbors who use the same discovery implementation, it becomes blind to those implemented in different discovery domains. Therefore, approach with UA always results with providing bigger choice to the node, which in turn gives better success rate and higher quality.

We also see that with UA and without UA results are not very symmetric. The network for the approach with UA becomes too dense earlier than for the approach without UA.

## 6.5 Summary

In this chapter, we have proposed Universal Adaptor, a novel approach towards supporting multi-protocol service discovery in pervasive computing. UA consists of Universal Adaptor Primitives and Universal Adaptor Mapping. It provides mapping from UAP to protocol specific primitives. The client makes use of UAP to discover and access services. UAM performs mapping from UAP to service specific SDP primitives. From the point of view of the service side, Universal Adaptor is a tailored component that uses native SDP and performs service discovery on behalf of the client.

Our contribution to the interoperability of service discovery is providing solution that in a lightweight manner bridges not only all existing but future service discovery systems, including standard ones, as well as service discovery mechanisms that support multiple protocols within the domain. Our implementation has shown that Universal Adaptor is a simple and flexible solution. It is easy in sense of writing the code, implementing in diverse infrastructures and using by client.



## Chapter 7

# Conclusions and Future Works

This chapter concludes this dissertation by summarizing our original contributions and suggestions on the ways in which the current research can be advanced.

### 7.1 Conclusions

As discussed in this thesis, service composition concerns several functionalities for bridging the gap between users and services embedded in PvCE. In particular, scalable service composition is concerned with ability of the service composition mechanism to manage increasing scale of service providers and service requests with minimal effort given in number of messages and response delay. Growing scale of the environments as well as their dynamicity make it more difficult to manage information about available service providers. Traditional centralized approaches which rely on special node collecting global information become not suitable for such environments. Techniques that adopt decentralized and more specifically localized architectures are proposed to address scalability issues. Further, service replication, service locality and environment partitioning are important principles to achieve scalability of the service composition mechanism.

In this thesis, we first investigated the characteristics of UIO based large scale and dynamic pervasive computing environments, and identified the new challenges caused by these characteristics in the design of service composition support. Based on the investigations, we study how to design localized service composition protocols in large scale dynamic environments.

Our research in scalable service composition is based on a bottom-up community framework

which consists of a collection of service provider nodes, chosen from all the nodes in an large scale and dynamic environment considering their functionalities, along with a suite of protocols that aim to provide scalable service composition and reconfiguration supports to mobile users in these environments. The bottom-up community is used as a basic infrastructure over which many different service composition related protocols have been developed. To make it more clear, the bottom-up community provides application developers with a sense of scalability in a heavily dynamic and large scale network environment. The bottom-up community is scalable enough to carry on with service composition operations not depending on the scale of the environments.

Our work consists of three parts. Based on the bottom-up community framework, we have developed three different mechanisms: one for scalable service composition, one for composed service reconfiguration and the other for heterogeneity support.

We have proposed a localized approach to service composition, named LASEC. The proposed approach is suitable for service composition applications deployed in environments which are large scale and highly dynamic. In this research we significantly decrease the communication cost during service composition in highly dynamic large-scale environments, while satisfying the composition quality. We achieve this by removing coordinator from the composition process, and making service providers responsible to partially compose the request. Therefore, there is no node which knows full information about the composition process; however the global solution successfully emerges. We propose a novel localized service composition algorithm (LASEC), in which the nodes collaborate only with their local neighbors to compose the service specified by a user. To avoid problem of nodes collaboration with partners that may fail to compose remaining parts of the request, we introduce a technique called "Alien-based Acknowledging". We demonstrate feasibility of our approach in prototype implementation and conduct extensive simulations to study the performance of the proposed LASEC compared with existing decentralized and pull-based centralized techniques. Simulation results show that our technique decreases message cost

and response delay. Based on collected results we discuss performance of our approach in terms of scalability and composition locality.

On the top on our bottom-up community we have also developed a composed service reconfiguration mechanism. The proposed approach is suitable for service composition applications deployed in environments which are large scale and highly dynamic. In particular, we have addressed the problem of hidden abort in commit protocol that is implemented on the bottom-up community and proposed techniques to solve it. We have conducted extensive simulations to show how our approach copes with dynamic environments.

The last part of our research on service composition is about coping with the problem of heterogeneity and providing service composition across different environments supported by different service management systems, which may use standard protocols as well as tailor-made mechanisms. We proposed the Universal Adaptor approach, which consists of two major components: the Universal Adaptor Primitives and the Universal Adaptor Mapping . UAP is the universal set of primitives used by the user to discover the services across different environments, while UAM provides the mapping between the Universal Adaptor Primitives to the primitives used in various service discovery systems.

## 7.2 Future Works

Several issues are still worth investigating. One core challenge that remains more or less unexplored is how we can further utilize localized algorithms for service composition in PvCEs. Developing localized algorithms can be the key to address the challenges posed by the dynamic and ad hoc nature of pervasive systems. Pervasive devices must coordinate locally with peer devices to achieve some global objective with respect to service composition.

For the future, we plan to test the proposed solution with larger networks and use real devices to compare results with the simulation. So far, our mechanisms have been implemented and tested using a wireless sensor network with all the nodes being MicaZ sensor motes. We want to carry

out further experimentation using multitude of UIOs . In addition, to assist developers with design and testing of various service composition mechanisms for large scale dynamic environments, we plan to build a simulator for PvC environment. System development for PvCE involves a multitude of embedded and hand-held hardware devices varying widely in characteristics and performance. Developing protocols using these heterogeneous resources is non-trivial due to the following two reasons. Firstly, acquisition of all such devices to test a PvC application is often impossible for every developer. Even if all the resources are available, managing them properly for carrying out experiments is unwieldy and time-consuming. Secondly, composed services are specific to certain environments, like smart home or office, smart museum, hospital or airport, etc. While some environments are possible to simulate using a laboratory setting, it is nearly impossible to simulate a hospital or an airport locally. We will develop a framework and a software environment for simulating UIO based PvCEs. It will allow different PvC environments to be modeled and protocols for service composition to be tested dynamically under various realistic conditions. The potential merit of such a system is that it would allow us to create any PvC environment, with any number and type of devices, and enable testing and performance evaluation of diverse service composition mechanisms deployed over the large scale and dynamic UIO based PvCE.

Currently in our work, we evaluated the performance of the algorithms using message overhead defined as the number of messages generated by the composition process. At the moment message size used in our simulations did not impact the algorithm in a noticeable way. However, we can envision that in real applications message size may grow, since additional information may be necessary to be included about different functional and nonfunctional properties of the service providers, for example when providing QoS support. Processing larger messages may cause drop in performance and therefore impact of message size on the performance of the localized algorithms is an interesting issue to further study as well.

QoS support is another issue that we plan to further investigate. Currently our localized mech-



anism considers maximizing QoS locally. If we allow for longer running time, the local environment explored is larger, and the quality of the final solution may increase as well. However, for centralized approach, due to maintaining global information is easier to come up with optimal QoS. We plan to study different approaches to increasing QoS utilizing localized interactions. It would be also interesting to consider diverse parameters of quality and incorporate them into the service composition mechanism. One of the problems is difficulty to estimate parameter  $k$ , which influences both quality of the solution as well as performance of the composition mechanism. The main problem however has its origin in lack of global information although we can still successfully compose the service, however the system cannot achieve best possible quality. Extensions to the proposed mechanisms can improve the quality of the solution, although it can never be as good as when using approaches with global knowledge.

Also the fault tolerance issue has grossly been overlooked by the researchers. Pervasive computing is mainly application-specific and many of the applications are safety-critical, e.g. health-care or elderly-care applications. Service composition in this type of systems certainly needs highest reliability support that can be provided. Fault tolerance for localized approaches is more difficult than for environments with some entities knowing global knowledge. The dynamic nature of the environment, resource-constrain of the participating devices, and the limited knowledge make it hard to design robust fault tolerant mechanisms for them compared to the traditional distributed environment.

For the heterogeneity support, we will conduct more experiments to investigate the performance of the UA approach. In our future work, there are many issues to investigate. First, we plan to use mobile agents to find services in environments on behalf of users. Second, so far, we have assumed that the client will use UA primitives for service discovery and access, and not considered how to support clients using existing protocols. We will develop the mechanisms to provide transparent UA multi-protocol service invocation to these clients. Finally, we will inves-

investigate the possibility to let the client take the Universal Adaptor and dynamically install it to the target environment.

# Bibliography

1. G.D. Abowd, and E.D. Mynatt, "Charting Past, Present, and Future Research in Ubiquitous Computing", *ACM Transactions on Human-Computer Interaction*, 7(1), pp. 29-58, March 2000.
2. C. Adjih, P. Jacquet, and L. Viennot, "Computing Connected Dominated Sets with Multipoint Relays," Technical Report 4597, INRIA-Rapport de Recherche, Oct. 2002.
3. M.D. Addlesee, R. Curwen, S. Hodges, J. Newman, P. Steggles, A. Ward, and A. Hopper, "Implementing a Sentient Computing System", *IEEE Computer Magazine*, Vol. 34, No. 8, August 2001, pp. 50-56.
4. S. Adhikari, A. Paul, U. Ramachandran, "D-Stampede: Distributed Programming System for Ubiquitous Computing", *Proc. of International Conference on Distributed Computing Systems (ICDCS)*, July 2002.
5. W. Adjie-Winoto, E. Schwartz, H. Balakrishnan, and J. Lilley, "The Design and Implementation of an Intentional Naming System", *Proc. of the 17th ACM Symposium on Operating Systems Principles (SOSP)*, 1999.
6. G. Banavar, J. Beck, E. Gluzberg, J. Munson, J. Sussman, and D. Zukowski, "Challenges: An Application Model for Pervasive Computing," *Proceedings of the 6th ACM MobiCom*, Boston, MA, USA, August 2000
7. J. Barton and T. Kindberg, "The Challenges and Opportunities of Integrating the Physical World and Networked Systems," *HP Labs Technical Report*, Jan 31, 2001.

8. M. Beek, A. Bucchiarone, and S. Gnesi, "A survey on service composition approaches: From industrial standards to formal methods", Technical Report 2006-TR-15, 2006.
9. U. Bellur, N. C. Narendra, I. I. T. KReSIT, and I. Mumbai. "Towards service orientation in pervasive computing systems". International Conference on Information Technology: Coding and Computing, 2:289-295, 2005.
10. C. Bettstetter and C. Renner, "A Comparison of Service Discovery Protocols and Implementation of the Service Location Protocol," In Proceedings of the EUNICE Open European Summer School, Twente, Netherlands, Sept. 2000.
11. C. Bisdikian, I. Boamah, P. Castro, A. Misra, J. Rubas, N. Villoutreix, D. Yeh, "Intelligent pervasive middleware for context-based and localized telematics services", Proceedings of the second international workshop on Mobile commerce, ACM Press, pp 15-24, 2002.
12. D. Blough, M. Leoncini, G. Resta, and P. Santi, "The K-Neigh Protocol for Symmetric Topology Control in Ad Hoc Networks," Proc. MobiHoc, pp. 141-152, June 2003.
13. G. Borriello, M. Chalmers, A. LaMarca, and P. Nixon, "Delivering REAL-WORLD Ubiquitous Location Systems", Communications of the ACM, pp. 36-41, March 2005.
14. G. Borriello, R. Want, "Embedded Computation Meets the World Wide Web," Communications of the ACM, pp. 59 - 66, May 2000.
15. Y. D. Bromberg, and V. Issarny, "INDISS: Interoperable Discovery System for Networked Services," In Proceedings of Middleware 2005. Grenoble, France November 2005.
16. M. Burkhard, P. Rickenbach, R. Wattenhofer, and A. Zollinger, "Does Topology Control Reduce Interference?" Proc. MobiHoc, 2004.
17. J. Cartigny, D. Simplot, and I. Stojmenovic, "Localized Minimum-Energy Broadcasting in Ad-Hoc Networks," Proc. Infocom, pp. 2210-2217, 2003.

18. D. Chakraborty, A. Joshi, T. Finin, and Y. Yesha. Service composition for mobile environments. *Journal on Mobile Networking and Applications, Special Issue on Mobile Services*, 10(4):435-451, January 2005.
19. D. Chakraborty, F. Perich, A. Joshi et al., "A reactive service composition architecture for pervasive computing environments," *Mobile and Wireless Communications*, pp. 53, 2003.
20. Y. P. Chen, A. L. Liestman, and J. Liu, "Clustering algorithms for ad hoc wireless networks," in *Ad Hoc and Sensor Networks*, Y. Xiao and Y. Pan, Eds. Nova Science Publisher, Date 2004.
21. B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "SPAN: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks," *ACM Wireless Networks J.*, vol. 8, no. 5, pp. 481-494, 2002.
22. T. Cottenier and T. Elrad. Adaptive embedded services for pervasive computing. In *Workshop on Building Software for Pervasive Computing – ACM SIGPLAN conf. on Object-Oriented Programming, Systems, Languages, and Applications*, 2005.
23. F. Dai and J. Wu, "Distributed Dominant Pruning in Ad Hoc Wireless Networks," *Proc. IEEE Int'l Conf. Comm.*, vol. 1, pp. 353-357, May 2003.
24. N. Davies, and H. Gellersen, "Beyond Prototypes: Challenges in Deploying Ubiquitous Systems", *IEEE Pervasive computing*, pp. 26-35, January/March 2002.
25. P. Dobrev, D. Famolari, C. Kurzke and B. A. Miller, "Device and service discovery in home networks with OSGi," *IEEE Communications Magazine* 40, 8 (Aug 2002): 86-92.
26. K. Edwards, R. Grinter. "At Home with Ubiquitous Computing: Seven Challenges", *Ubiquitous Computing 2001*, Atlanta, GA, September 2001.
27. B. Ensink, J. Stanley, V. Adve, "Program Control Language: A Programming Language for Adaptive Distributed Applications." Submitted for publication to *Journal of Parallel and Distributed Computing*, special issue on Computational Grids

28. D. Estrin, D. Culler, K. Pister, and G. Sukhatme, "Connecting the Physical World with Pervasive Networks", *IEEE Pervasive Computing*, pp. 59-69, January/March 2002.
29. K.F. Eustice, T.J. Lehman, A. Morales, M.C. Munson, S. Edlund, M. Guillen, "A Universal Information Appliance," *IBM Systems Journal*, Vol 38, No. 4. 1999.
30. C. A. Flores-Cortés, G.S. Blair, and P. Grace, "A multi-protocol framework for ad-hoc service discovery," In *Proceedings of the 4th international Workshop on Middleware For Pervasive and Ad-Hoc Computing (MPAC 2006)* (Melbourne, Australia, November 27 - December 01, 2006). vol. 182. ACM Press, New York, NY.
31. K. Fujii, and T. Suda, "Semantics-based dynamic service composition," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 12, pp. 2361-2372, 2005.
32. J. Gao, L.J. Guibas, J. Hershberger, L. Zhang, and A. Zhu, "Discrete Mobile Centers," *Proc. Symp. Computational Geometry*, pp. 188-196, 2001.
33. D. Garlan, D.P. Siewiorek, A. Smailagic, and P. Steenkiste, "Project Aura: Toward Distratcion-Free Pervasive Computing", *IEEE Pervasive Computing*, pp. 22-31, April/June 2002.
34. Y. Goland, T. Cai, P. Leach, Y. Gu, and S. Albright, "Simple Service Discovery Protocol 1.0," *IETF, Internet-Draft Version 03*, Oct. 1999.
35. P. Grace, G.S. Blair, and S. Samuel, "A reflective framework for discovery and interaction in heterogeneous mobile environments," *SIGMOBILE Mob. Comput. Commun. Rev.* 9, 1 (Jan. 2005).
36. R. Grimm, "One.world: Experiences with a pervasive computing architecture", *IEEE Pervasive Computing*, 3(3), pp. 22-30, July/September 2004.
37. R. Grimm, J. Davis, B. Hendrickson, E. Lemar, A. MacBeth, S. Swanson, T. Anderson, B. Bershad, G. Borriello, S. Gribble, and D. Wetherall, "Systems directions for pervasive computing", *Proceedings of the 8th Workshop on Hot*

- Topics in Operating Systems (HotOS-VIII), pages 147-151, Elmau, Germany, May 2001.
38. R. Grimm, J. Davis, E. Lemar, A. MacBeth, S. Swanson, T. Anderson, B. Bershad, G. Borriello, S. Gribble, and D. Wetherall, "System Support for Pervasive Computing", *ACM Transactions on Computer Systems*, 22(4), pp. 421-486, November 2004.
  39. R. Grimm, J. Davis, E. Lemar, A. MacBeth, S. Swanson, S. Gribble, T. Anderson, B. Bershad, G. Borriello, and D. Wetherall, "Programming for Pervasive Computing Environments", Technical Report, UW-CSE-01-06-01, University of Washington, Department of Computer Science and Engineering, June 2001.
  40. R. Gupta, S. Talwar, and D.P. Agrawal, "Jini Home Networking: A Step Toward Pervasive Computing", *IEEE Computer*, pp. 34-40, August 2002.
  41. E. Guttman, C. Perkins, J. Veizades, and M. Day, "Service Location Protocol, Version 2," IETF, RFC 2608, June 1999.
  42. J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, D. Ganesan, "Building Efficient Wireless Sensor Networks with Low-Level Naming", *Proc. of the 18th ACM Symposium on Operating Systems Principles*, October, 2001.
  43. S. Helal, "Standards for Service Discovery and Delivery", *IEEE Pervasive Computing* 1, 3 (Jul. 2002), 95-100.
  44. S. Helal, N. Desai, V. Verma et al., "Konark-a service discovery and delivery protocol for ad-hoc networks," 2003 *IEEE Wireless Communications and Networking*, 2003. WCNC 2003, vol. 3, 2003.
  45. C. Hesselman, A. Tokmako\_, P. Pawar, and S. Iacob. Discovery and composition of services for context-aware systems. volume 4272, 2006.
  46. R. Hermann, D. Husemann, M. Moser, M. Nidd, C. Rohner, and A. Schade, "DEAPspace: Transient Ad-hoc Networking of Pervasive Devices", *Proc. of MobiHoc*, 2000.

47. T. D. Hodes, S. E. Czerwinski, B. Y. Zhao, A. D. Joseph, R. H. Katz, "An Architecture for Secure Wide-Area Service Discovery", *ACM Wireless Networks Journal*, special issue. Volume 8, Issue 2/3, March/May 2002, pp. 213-230.
48. R. Hou and H. Shi, A localized algorithm for finding disjoint paths in wireless sensor networks, *IEEE Communications Letters* 10 (2006) (12), pp. 807–809.
49. M. Imai, Y. Hirota, S. Satake, H. Kawashima, "Semantic Connection between Everyday Objects and a Sensor Network" *Semantic Sensor Networks Workshop: A workshop of International Semantic Web Conference*, 2006.11
50. C. Intanagonwiwat, R. Govindan and D. Estrin, Directed diffusion: A scalable and robust communication paradigm for sensor networks, in: *Proc. ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2000)*, Boston, MA, USA, (Aug 2000) pp. 56–67
51. K. Isla, S.G. Akl, "A Localized Algorithm for Target Monitoring in Wireless Sensor Networks," In *Proceedings of the 8th international Conference on Ad-Hoc, Mobile and Wireless Networks* (Murcia, Spain, September 22 - 25, 2009).
52. L. Jia, R. Rajaraman, and T. Suel, "An Efficient Distributed Algorithm for Constructing Small Dominating Sets," *Proc. ACM Symp. Principles of Distributed Computing*, pp. 33-42, Aug. 2001.
53. G. Kaefer, R. Schmid, G. Prochart, and R. Weiss. Framework for dynamic resource-constrained service composition for mobile ad hoc networks. *UBICOMP, Workshop on System Support for Ubiquitous Computing*, 2006.
54. J. M. Kahn, R. H. Katz and K. S. J. Pister, "Mobile Networking for Smart Dust", *Proc. ACM/IEEE Intl. Conf. on Mobile Computing and Networking (MobiCom 99)*, Seattle, WA, August 17-19, 1999
55. S. Kalasapur, M. Kumar, B. Z. Shirazi. Dynamic Service Composition in Pervasive Computing. *IEEE Transaction on Parallel and Distributed Systems*, 2007.
56. S. Kalasapur; K. Senthivel; M. Kumar, "Service oriented pervasive computing for emergency response systems," *Pervasive Computing and Communications*



- Workshops, 2006. M.J.Kim, M.Kumar, B.A. Shirazi, , "Service discovery using volunteer nodes for pervasive environments," *Pervasive Services*, 2005. ICPS '05. Proceedings. International Conference on , vol., no., pp. 188-197, 11-14 July 2005
57. S. H. Kang, S. Ryu, N. Kim, Y. Lee, D. Lee, and K. D. Moon, "An Architecture for Interoperability of Service Discovery Protocols Using Dynamic Service Proxies," *ICOIN 2005* : 786-795.
  58. T. Kindberg and A. Fox, "System Software for Ubiquitous Computing", *IEEE Pervasive Computing*, pp. 70-81, Jan-March 2002.
  59. T. Koponen, and T. Virtanen, "A Service Discovery: A Service Broker Approach," In *Proceedings of the 37th Annual Hawaii international Conference on System Sciences (Hicss'04) - Track 9 - Volume 9 (January 05 - 08, 2004)*. HICSS. IEEE Computer Society, Washington, DC, 90284.2.
  60. F. Kuhn, T. Moscibroda, and R. Wattenhofer, "Initializing Newly Deployed Ad Hoc and Sensor Networks," *Proc. MobiCom*, pp. 260-274, Sept./Oct. 2004.
  61. F. Kuhn, T. Moscibroda, and R. Wattenhofer, "What Cannot Be Computed Locally!" *Proc. ACM Symp. Principles of Distributed Computing*, July 2004.
  62. F. Kuhn and R. Wattenhofer, "Constant-Time Distributed Dominating Set Approximation," *Proc. ACM Symp. Principles of Distributed Computing*, 2003.
  63. J. Kulik, R.B. Heinzelman and H. Balakrishnan, *Negotiation-based protocols for disseminating information in wireless sensor networks (2000)*, *ACM Wireless Networks*.
  64. S. Y. Lee, J. Y. Lee, and B. I. Lee. *Service composition techniques using data mining for ubiquitous computing environments*. *International Journal of Computer Science and Network Security*, 6(9):110-117, 2006
  65. W. L. C. Lee. S. Ko, S. Lee and A. Helal. *Context-aware service composition for mobile network environments*. In *4th International Conference on Ubiquitous Intelligence and Computing (UIC2007)*, 2007.

66. L. Li, J.Y. Halpern, V. Bahl, Y.M. Wang, and R. Wattenhofer, "Analysis of a Cone-Based Distributed Topology Control Algorithm for Wireless Multi-Hop Networks," Proc. ACM Symp. Principles of Distributed Computing, pp. 264-273, Aug. 2001.
67. N. Li, J.C. Hou, and L. Sha, "Design and Analysis of an MST-Based Topology Control Algorithm," Proc. Infocom, vol. 3, pp. 1702-1712, Mar./Apr. 2003.
68. X.Y. Li, Y. Wang, P.J. Wang, W.Z. Song, and O. Frieder, "Localized Low-Weight Graph and Its Application in Wireless Ad Hoc Networks," Proc. Infocom, 2004.
69. H. Lim and C. Kim, "Flooding in Wireless Ad Hoc Networks," Computer Comm. J., vol. 24, nos. 3-4, pp. 353-363, 2001.
70. T. D.C. Little, B. Prithwish, K. Wang. A novel approach for execution of distributed tasks on mobile ad hoc networks. In IEEE WCNC. Orlando. Florida, 2002.
71. Y. Liu, H.P. Schwefel, "Localized Algorithms for Virtual Backbone Formation in Wireless Multi-hop Networks with unidirectional links". I: 16th IST Mobile and Wireless Communications Summit, 2007. Electrical Engineering/Electronics, Computer, Communications and Information Technology Association, 2007.
72. W. Lou and J. Wu, "On Reducing Broadcast Redundancy in Ad Hoc Wireless Networks," IEEE Trans. Mobile Computing, vol. 1, no. 2, pp. 111-122, Apr.-June 2002.
73. R. Marin-Perianu, P. Hartel, and H. Scholten, "A Classification of Service Discovery Protocols". Technical Report TR-CTIT-05-25, Centre for Telematics and Information Technology, University of Twente, Enschede. ISSN 1381-3625, June 2005.
74. W. Mark, "Turning Pervasive Computing into Mediated Spaces," IBM Systems Journal, Vol 38, No. 4. 1999.
75. J. Ma, L. T. Yang, B. O. Apduhan, R. Huang, L. Barolli, M. Takizawa, T. K. Shih, "A Walkthrough from Smart Spaces to Smart Hyperspaces towards a Smart World

- with Ubiquitous Intelligence.” In Proceedings of the 11th international Conference on Parallel and Distributed Systems 2005.
76. C. Mascolo, L. Capra, and W. Emmerich, "Principles of Mobile Computing Middleware". In Q. Mahmoud (ed), *Middleware for Communications*. pp. 261-280. John Wiley, 2004.
  77. P. J. McCann, G.C. Roman, "Compositional Programming Abstractions for Mobile Computing." *IEEE Transactions on Software Engineering*, Vol. 24, No. 2, Feb 1998.
  78. D. Mennie and B. Pagurek, "An Architecture to Support Dynamic Composition of Service Components," 5th International Workshop on Component-Oriented Programming, WCOP 2000, Cannes, France, 2000.
  79. B. Miller, T. Nixon, C. Tai et al., "Home networking with universal plug and play," *IEEE communications magazine*, vol. 39, no. 12, pp. 104-109, 2001.
  80. N. Minar, M. Gray, O. Roup, R. Krikorian, P. Maes, "Hive: Distributed Agents for Networking Things," *Proc. of ASA/MA '99*.
  81. S. Mokhtar, D. Fournier, N. Georgantas et al., "Context-aware service composition in pervasive computing environments," *Lecture Notes in Computer Science*, vol. 3943, pp. 129, 2006.
  82. S. Mokhtar, N. Georgantas, and V. Issarny, "Ad hoc composition of user tasks in pervasive computing environments," *Lecture Notes in Computer Science*, vol. 3628, pp. 31, 2005.
  83. S. B. Mokhtar, N. Georgantas, and V. Issarny. *Cocoa: Conversation-based service composition in pervasive computing environments*. Proceedings of the IEEE International Conference on Pervasive Services, 2006.
  84. J. Nakazawa, H. Tokuda, W.K. Edwards, and U. Ramachandran, "A Bridging Framework for Universal Interoperability in Pervasive Systems," In Proceedings of the 26th IEEE International Conference on Distributed Computing Systems (July 04 - 07, 2006). ICDCS. IEEE Computer Society, Washington, DC.

85. Q. Ni. Service composition in ontology enabled service oriented architecture for pervasive computing. In Workshop on Ubiquitous Computing and e-Research, 2005.
86. B. Noble, "System Support for Mobile, Adaptive Applications", IEEE Personal Communications 7(1), pp. 44-49, 2000.
87. B. Noble, M. Price, M. Satyanarayanan, "A Programming Interface for Application-Aware Adaptation in Mobile Computing." In Proceedings of 2nd USENIX Symposium on Mobile & Location-Independent Computing, Ann Arbor, Apr. 1995.
88. S. Osman, D. Subhraveti, G. Su, and J. Nieh, "The Design and Implementation of Zap: A System for Migrating Computing Environments", Proc. of the 5th Symposium on Operating Systems Design and Implementation (OSDI 2002), Dec. 9-11, 2002.
89. W. Peng and X. Lu, "AHBP: An Efficient Broadcast Protocol for Mobile Ad Hoc Networks," J. Science and Technology, 2002
90. W. Peng and X. Lu. Efficient broadcast in mobile ad hoc networks using connected dominating sets. Journal of Software, 1999.
91. W. Peng and X. Lu, "On the Reduction of Broadcast Redundancy in Mobile Ad Hoc Networks," Proc. MobiHoc, pp. 129-130, June 2000.
92. S.R. Ponnekanti, B. Lee, A. Fox, P. Hanrahan, and T. Winograd, "ICrafter: A Service Framework for Ubiquitous Computing Environments", Proc. of Ubiquitous Computing, Sep. 2001.
93. H. Pourreza and P. Graham. On the fly service composition for local interaction environments. In IEEE International Conference on Pervasive Computing and Communications Workshops, page 393. IEEE Computer Society, 2006.
94. P. G. Raverdy, V. Issarny, R. Chibout, and A. de La Chapelle, "A Multi-Protocol Approach to Service Discovery and Access in Pervasive Environments," In Proceedings of MOBIQUITOUS – The 3rd Annual International Conference on Mobile and Ubiquitous Systems: Networks and Services. July 2006, San Jose, CA, USA.

95. J. Robinson, , Wakeman, I., and Chalmers, D. 2008. Composing software services in the pervasive computing environment: Languages or APIs?. *Pervasive Mob. Comput.* 4, 4 (Aug. 2008),
96. V. Rodoplu and T.H. Meng, "Minimum Energy Mobile Wireless Networks," *IEEE J. Selected Areas in Comm.*, vol. 17, no. 8, pp. 1333-1344, Aug. 1999.
97. M. Roman, C. Hess, R.Cerqueira, A. Ranganathan, R.H. Campbell, K. Nahrstedt, "A Middleware Infrastructure for Active Spaces," *IEEE Pervasive Computing*, Oct.-Dec. 2003, pp. 74 -- 83.
98. M. Roman, C. Hess, R. Cerqueira, A. Ranganat, R.H. Campell, and K. Nahrstedt, "Gaia: A Middleware Infrastructure to Enable Active Space", *IEEE Pervasive Computing*, pp. 74-83, October/December 2002.
99. K. Romer, F. Mattern, T. Dubendorfer, J. Senn, "Infrastructure for Virtual Counterparts of Real World Objects."
100. P.E. Ross, "Managing Care Through the Air", *IEEE Spectrum*, December 2004.
101. D. Saha and A. Mukherjee, "Pervasive Computing" A Paradigm for the 21st Century", *IEEE Computer*, pp. 25-31, March 2003.
102. M. Satyanarayanan, "Pervasive Computing: Vision and Challenges", *IEEE Personal Communications*, pp. 10-17, August 2001.
103. N. Schilit, N. Adams, and R. Want. "Context-aware computing applications", In *Proceedings Workshop on Mobile Computing Systems and Applications*. IEEE, December 1994.
104. M. Sheshagiri, N. M. Sadeh, and F. Gandon. Using semantic web services for context-aware mobile applications. *Second International Conference on Mobile Systems (MobiSys 2004), Applications, and Services - Workshop on Context Awareness*, 2004.
105. H. Shimizu, O. Hanzawa, K. Kanehana, H. Saito, N. Thepvilojanapong, K. Sezaki, and Y. Tobe, "Association Management between Everyday Objects and Personal Devices for Passengers in Urban Areas," *Pervasive 2005*,

106. J. Siebert, J.N. Cao, L. Cheng, E. Wei, C. Chen, J. Ma. Decentralized Service Composition in Pervasive Computing Environments. In International Wireless Communications and Mobile Computing Conference (IWCMC 2010).
107. P. Sinha, R. Sivakumar, and V. Bharghavan, "Enhancing Ad Hoc Routing with Dynamic Virtual Infrastructures," Proc. Infocom, pp. 1763-1772, Apr. 2001.
108. J. Soldatos, I. Pandis, K. Stamatis, L. Polymenakos, J. Crowley, "Agent Based Middleware Infrastructure for Autonomous Context-Aware Ubiquitous Computing Services.", Computer Communications, 2007.
109. Z. Song, Y. Labrou, and R. Masuoka. Dynamic service discovery and management in task computing. *Mobiquitous*, 00:310-318, 2004.
110. W.Z. Song, Y. Wang, X.Y. Li, and O. Frieder, "Localized Algorithms for Energy Efficient Topology Control in Wireless Ad Hoc Networks," Proc. MobiHoc, 2004.
111. J.P. Sousa, D. Garlan, "Aura: An Architecture Framework for User Mobility in Ubiquitous Computing Environments," Proc. 3rd Working IEEE/IFIP Conf. on Software Architecture, 2002.
112. J. Sucec and I. Marsic, "An Efficient Distributed Network-Wide Broadcast Algorithm for Mobile Ad Hoc Networks," CAIP Technical Report 248, Rutgers Univ., Sept. 2000.
113. P. Tandler, "Software Infrastructure for Ubiquitous Computing Environments: Supporting Synchronous Collaboration with Heterogeneous Devices", Proc. of UbiComp, 2001.
114. M. Vallee, F. Ramparany, and L. Vercouter. Flexible composition of smart device services. In International Conference on Pervasive Systems and Computing (PSC05), pages 165-171. CSREA Press, 2005.
115. J. Waldo, "The Jini Architecture for Network-Centric Computing", Communications of the ACM, Vol. 42 No. 7, July, 1999.

116. H. Wang; C.C. Tan,.; Q. Li, "Snoogle: A Search Engine for the Physical World," INFOCOM 2008. The 27th Conference on Computer Communications. IEEE , vol., no., pp.1382-1390, 13-18 April 2008
117. R. Want, G. Borriello, T. Pering, K.I. Farkas, "Disappearing Hardware," IEEE Pervasive Computing, vol. 1, no. 1, Jan.-Mar. 2002, pp. 36-47.
118. M. Weiser, "Hot Topics: Ubiquitous Computing", IEEE Computer, October 1993.
119. M. Weiser, "Some Computer Science Issues in Ubiquitous Computing", Communications of the ACM, pp.75-84, July 1993.
120. M. Weiser, "The Computer for the Twenty-First Century", Scientific American, pp. 94-10, September 1991.
121. M. Weiser, "The world is not a desktop", Interactions, pp. 7-8, January 1994.
122. J. Wu and F. Dai, "A Generic Distributed Broadcast Scheme in Ad Hoc Wireless Networks," Proc. Int'l Conf. Distributed Computing Systems, pp. 460-468, May 2003.
123. J. Wu and H. Li, "On Calculating Connected Dominating Set for Efficient Routing in Ad Hoc Wireless Networks," Proc. Int'l Workshop Discrete Algorithms and Methods for Mobile Computing and Comm., pp. 7-14, 1999.
124. K. Yap, V. Srinivasan, and M. Motani, "MAX: human-centric search of the physical world," in Sensys 2005.
125. S.S. Yau, F. Karim, Y. Wang, B. Wang, and S.K.S. Gupta, "Reconfigurable Context-Sensitive Middleware for Pervasive Computing", IEEE Pervasive Computing, pp. 33-40, July-September 2002.
126. M. Yu, A. Taleb-Bendiab, D. Reilly, and Wail Omar, "Multi-Standard Service Interoperation Protocol Through Polyarchical Middleware", In Proceedings of 4th Annual Postgraduate Symposium on The Convergence of Telecommunications, Networking & Broadcasting (PGNet2003), Liverpool, U.K., June, 2003.
127. F. Zhu, M. W. Mutka, and L. M. Ni, "Service Discovery in Pervasive Computing Environments," IEEE Pervasive Computing 4, 4 (Oct. 2005), 81-90

128. N. Belaramani, C.L. Wang and F.C.M. Lau, "Dynamic Component Composition for Functionality Adaptation in Pervasive Environments," In Proceedings 9th International Workshop on Future Trends of Distributed Computing Systems (FTDCS2003), San Juan, Puerto Rico, May 28 to 30, 2003.
129. V.W.M. Kwan, F.C.M. Lau, and C.L. Wang, "Functionality Adaptation: A Context-Aware Service Code Adaptation for Pervasive Computing Environments", The 2003 IEEE/WIC International Conference on Web Intelligence, pp. 358-364, Halifax, Canada, October 13-17, 2003.
130. N.M. Belaramani, Y. Chow, V.W.M. Kwan, C.L. Wang, and F.C.M. Lau, "A Component-based Software Architecture for Pervasive Computing," Chap. 10, pp.201-222, Intelligent Virtual World: Technologies and Applications in Distributed Virtual Environments, chapter 10, pp. 191-212, World Scientific Publishing Co.