UNIVERSIDAD POLITÉCNICA DE VALENCIA

DEPARTMENT OF COMPUTER ENGINEERING

Integrated Architecture for Configuration and Service Management in MANET Environments

Thesis submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science

José Cano Reyes

Ph.D. advisors:
Dr. Juan Carlos Cano Escribá
Dr. Carlos Tavares Calafate

Valencia, January 2012

To my parents, brothers and sister.

Two things are infinite: the universe and human stupidity; and I'm not sure about the universe.

-Albert Einstein-

Acknowledgments

This six-year-long journey was not made alone, being the result of a cooperative learning process where my advisors, my colleagues and several researchers worldwide also made an important contribution.

This thesis would not have been possible without the kind support of my two advisors: Dr. Juan Carlos Cano and Dr. Carlos Tavares Calafate. Thanks for your never-ending flow of ideas, energy and efforts. Your contributions and clear-mindedness were only surpassed by your patience towards me. I could never forget your help and support, your trust and friendship.

I would like to express my gratitude to my research advisor at Parallel Architectures Group (GAP), Professor José Flich, whose support has also been very important for the completion of this thesis, as he allowed me to devote a lot of time to it when it was necessary.

Special thanks to Professor Chai-Keong Toh for his close collaboration in Easy-MANET, it was an honor for me to work with him.

I would also like to thank the remaining members of the Computer Networks Groups (GRC), led by Professor Pietro Manzoni, which helped me in some critical programming issues and experiments: Álvaro, Johan, Miguel, Jorge, Carlos, Marga, and especially to Sergio for his support with the OMNeT simulator.

I am grateful to my laboratory colleagues at Parallel Architectures Group (GAP): Alex, Mario, Jose María, Toni, Ferran, Carlos, Santi, Yana, and Paulus. The joy and enthusiasm they had for their work was motivational for me.

I thank all the members of my family and my girlfriend for all their love and encouragement. Thanks for always being so supportive, encouraging, patient and for accompanying me in my growth as an individual, teaching me ethical and moral principles that will prevail for all my life.

Finally, I offer my regards to all those who supported me during the completion of this thesis. Thanks to you all...

José Cano Reyes
Valencia, January 2012

Abstract

This thesis has enabled us to translate some theoretical concepts of ubiquitous computing to real scenarios, while identifying specific needs in different types of applications. In order to achieve such objective, we propose two research prototypes that provide context-aware services in different environments, such as conference meetings or hospitals' recovery wards. Those experimental prototypes exploit Bluetooth technology to offer information based on the user preferences. In both cases, we run some experiments in order to evaluate the systems' behaviour and its performance.

The autoconfiguration problem of 802.11-based MANETs is also addressed in this thesis through two novel solutions. The first one is a centralized solution based on Bluetooth technology, while the second one is a distributed solution which does not need resorting to any additional technology because it relies on SSID parameter embedding. Both methods are designed for allowing non-expert users to transparently join an existing MANET, providing a fast and reliable autoconfiguration of the terminals. Experimental testbed results using real implementations have allowed us assessing the performance of the proposed solutions and show that nearby stations can be configured in few seconds. Furthermore, we have compared the solutions between them to evidence the different performance trade-offs.

The main contribution of this thesis is EasyMANET, an extensible and configurable platform whose goal is to automate as much as possible all the tasks that affect the configuration and startup of MANETs to make their use simpler and more widely available. EasyMANET makes use, as mandatory components, of an address autoconfiguration system (such as the ones named above), and a name resolution service known as Visual DNS. Visual DNS offers a graphical view of the MANET participants and gives users the possibility to access the services made available by other users. Through a large-scale validation using the *OMNeT++* simulator, we show that EasyMANET is a highly scalable solution.

This set of proposals will be presented in detail along the document, evidencing how wireless technologies like Bluetooth or IEEE 802.11 can significantly contribute to the development of ubiquitous applications.

Resumen

Esta tesis nos ha permitido trasladar algunos conceptos teóricos de la computación ubicua a escenarios reales, identificando las necesidades específicas de diferentes tipos de aplicaciones. Con el fin de alcanzar este objetivo, proponemos dos prototipos que proporcionan servicios sensibles al contexto en diferentes entornos, tales como conferencias o salas de recuperación en hospitales. Estos prototipos experimentales explotan la tecnología Bluetooth para ofrecer información basada en las preferencias del usuario. En ambos casos, hemos llevado a cabo algunos experimentos con el fin de evaluar el comportamiento de los sistemas y su rendimento.

También abordamos en esta tesis el problema de la autoconfiguración de redes MANET basadas en el estándar 802.11 a través de dos soluciones novedosas. La primera es una solución centralizada que se basa en la tecnología Bluetooth, mientras la segunda es una solución distribuida que no necesita recurrir a ninguna tecnología adicional, ya que se basa en el uso del parámetro SSID. Ambos métodos se han diseñado para permitir que usuarios no expertos puedan unirse a una red MANET de forma transparente, proporcionando una configuración automática, rápida, y fiable de los terminales. Los resultados experimentales en implementaciones reales nos han permitido evaluar el rendimiento de las soluciones propuestas y demostrar que las estaciones cercanas se pueden configurar en pocos segundos. Además, hemos comparado ambas soluciones entre sí para poner de manifiesto las diferentes ventajas y desventajas en cuanto a rendimento.

La principal contribución de esta tesis es EasyMANET, una plataforma ampliable y configurable cuyo objetivo es automatizar lo máximo posible las tareas que afectan a la configuración y puesta en marcha de redes MANET, de modo que su uso sea más simple y accesible. EasyMANET utiliza, como elementos obligatorios, un sistema de configuración automática de direcciones (como los mencionados anteriormente), y un servicio de resolución de nombres, denominado Visual DNS. Visual DNS muestra gráficamente los participantes de la red MANET y ofrece a los usuarios la posibilidad de acceder a los servicios puestos a disposición por otros usuarios. A través de una validación a gran escala utilizando el simulador *OMNeT++*, mostramos que EasyMANET es una solución altamente escalable.

Este conjunto de propuestas se presentarán con detalle a lo largo del documento, poniendo de manifiesto que tecnologías inalámbricas como Bluetooth o IEEE 802.11 pueden contribuir significativamente al desarrollo de aplicaciones ubicuas.

Resum

Aquesta tesi ens ha permès traslladar alguns conceptes teòrics de la computació ubiqua a escenaris reals, identificant les necessitats específiques de diferents tipus d'aplicacions. Per tal d'alcançar aquest objectiu, proposem dos prototips que proporcionen serveis sensibles al context en diferents entorns, com ara conferències o sales de recuperació en hospitals. Aquests prototips experimentals exploten la tecnologia Bluetooth per oferir informació basada en les preferències de l'usuari. En els dos casos, hem dut a terme alguns experiments per tal d'evaluar el comportament dels sistemes i el seu rendiment.

També abordem en aquesta tesi el problema de l'autoconfiguració de xarxes MANET basades en l'estàndard 802.11 a través de dos solucions novedoses. La primera és una solució centralitzada que es basa en la tecnologia Bluetooth, mentre la segona és una solució distribuïda que no necessita recórrer a cap tecnologia addicional, ja que es basa en l'ús del paràmetre SSID. Els dos mètodes s'han dissenyat per a permetre que usuaris no experts puguen unir-se a una xarxa MANET de forma transparent, proporcionant una configuració automàtica, ràpida, i fiable dels terminals. Els resultats experimentals en implementacions reals ens han permès evaluar el rendiment de les solucions propostes i demostrar que les estacions properes es poden configurar en pocs segons. A més, hem comparat les dos solucions entre si per posar de manifest els diferents avantatges i desavantatges pel que fa a rendiment.

La principal contribució d'aquesta tesi és EasyMANET, una plataforma ampliable i configurable l'objectiu de la qual és automatitzar el màxim possible les tasques que afecten a la configuració i posada en marxa de xarxes MANET, de manera que el seu ús sigua més simple i accessible. EasyMANET utilitza, com a elements obligatoris, un sistema de configuració automàtica d'adreces (com els mencionats anteriorment), i un servei de resolució de noms, denominat Visual DNS. Visual DNS mostra gràficament els participants de la xarxa MANET i ofereix als usuaris la possibilitat d'accedir als serveis posats a disposició per altres usuaris. A través d'una validació a gran escala utilitzant el simulador *OMNeT++*, mostrem que EasyMANET és una solució altament escalable.

Aquest conjunt de propostes es presentaran amb detall al llarg del document, posant de manifest que tecnologies sense fils com Bluetooth o IEEE 802.11 poden contribuir significativament al desenvolupament d'aplicacions ubiqües.

Contents

1 Motivation, Objectives and Organization of the Thesis **1**
1.1 Motivation . . . 1
1.2 Objectives of the Thesis . . . 2
1.3 Organization of the Thesis . . . 2

2 Ubiquitous Computing **5**
2.1 Overview . . . 5
2.2 Technical foundations . . . 7
2.3 Pervasive applications . . . 8
2.3.1 Future trends . . . 11
2.4 Summary . . . 11

3 Wireless Technologies **13**
3.1 Overview . . . 13
3.2 IEEE 802.11 . . . 15
3.2.1 Physical level . . . 16
3.2.2 IEEE 802.11 frame format . . . 18
3.2.3 Distributed Coordination Function (DCF): CSMA/CA . . . 20
3.2.4 Point Coordination Function (PCF) . . . 20
3.2.5 MAC-level retransmissions . . . 21
3.2.6 RTS/CTS . . . 21
3.2.7 IEEE 802.11i: Security . . . 22
3.2.7.1 Wired equivalent privacy (WEP) . . . 22
3.2.7.2 Wi-Fi Protected Access (WPA, WPA2) . . . 23
3.2.7.3 Wi-Fi Protected Setup . . . 25
3.2.8 Network architecture . . . 25
3.3 Bluetooth . . . 26
3.3.1 Specification . . . 27
3.3.2 Architecture . . . 28
3.3.3 Establishment of network connections . . . 28
3.3.4 Service Discovery Protocol (SDP) . . . 29
3.3.5 Basic Bluetooth Profiles . . . 29
3.3.6 Final considerations . . . 31
3.4 Conclusions . . . 32

4 Short Radio Range Wireless Networks **33**
4.1 Personal Area Networks . . . 33
4.1.1 Task Groups . . . 35
4.2 Mobile Ad Hoc Networks . . . 36
4.2.1 Classification of routing protocols . . . 37
4.2.1.1 Basic routing techniques . . . 38
4.2.2 Routing in ad hoc networks . . . 38
4.2.2.1 Routing protocol families for ad hoc networks . . . 39
4.2.2.2 The Optimized Link-State Routing (OLSR) . . . 41
4.2.2.3 Ad hoc On-Demand Distance Vector (AODV) . . . 44
4.2.2.4 Dynamic Source Routing (DSR) . . . 45
4.2.2.5 Dynamic MANET On-demand (DYMO) . . . 47
4.2.2.6 Multipath routing protocols . . . 48
4.2.3 Autoconfiguration in MANETs . . . 48
4.2.3.1 Interface configuration . . . 49
4.2.3.2 Name-to-Address Translation . . . 51
4.2.3.3 Service Discovery . . . 52
4.2.3.4 Automatic Multicast Addresses Allocation . . . 52
4.3 Spontaneous Networks . . . 53
4.4 Conclusions . . . 54

5 First Experiences Developing Pervasive Applications **55**
5.1 Introduction . . . 55
5.2 First proposal: BluePeer . . . 56
5.2.1 The prototype application . . . 57
5.2.1.1 Server functionality . . . 58
5.2.1.2 Client functionality . . . 58
5.2.2 Experimental results . . . 59
5.2.2.1 Testbed performance evaluation . . . 60
5.2.2.2 Simulation results . . . 65
5.3 Second proposal: BlueHospital . . . 68
5.3.1 System Architecture . . . 68
5.3.2 System Development . . . 70
5.3.2.1 Patient Device (BH_Patient) . . . 70
5.3.2.2 Room Manager (BH_Room_Manager) . . . 71
5.3.2.3 Doctor Application (BH_Doctor) . . . 72
5.3.2.4 Central Database . . . 72
5.3.3 Performance Evaluation . . . 73
5.3.3.1 Application Transfer Time . . . 74
5.3.3.2 Inquiry Delay Evaluation . . . 74
5.4 Summary . . . 76

6 Autoconfiguration of IEEE 802.11-based MANETs. **77**
6.1 Introduction . . . 77
6.2 Autoconfiguration through Bluetooth (BlueWi) . . . 79
6.2.1 System architecture . . . 79
6.2.2 Design issues . . . 82

6.2.3 Experimental results 84
6.3 Autoconfiguration through 802.11 beaconing 87
6.3.1 Implementation details 89
6.3.1.1 Proposed SSID partitioning strategy 89
6.3.1.2 Deriving the session key 90
6.3.1.3 MANET setup 91
6.3.2 Validation and performance analysis 92
6.3.2.1 Assessing the overhead introduced per task 92
6.3.2.2 Autoconfiguration times in a multi-hop environment 93
6.4 Comparison between both solutions 96
6.5 Summary 97

7 EasyMANET: An Extensible and Configurable Platform for Service Provisioning in MANET Environments 99
7.1 The EasyMANET Platform 99
7.1.1 Configuration of Basic Network Parameters 101
7.1.2 The Visual DNS Service 101
7.1.2.1 Data dissemination strategies 101
7.1.2.2 Discovery protocol 102
7.2 Performance Evaluation 105
7.2.1 Real testbed 106
7.2.2 Simulation results 109
7.2.2.1 OMNeT++ overview 109
7.2.2.2 Simulator setup 110
7.2.2.3 Validation: testbed vs simulation 111
7.2.2.4 Baseline scenario. 114
7.2.2.5 Scalability analysis of the total setup time. 114
7.2.2.6 Scalability analysis of the reconfiguration time . . 118
7.3 Summary 120

8 Conclusions, Publications and Future Work 121
8.1 Publications Related to the Thesis 122
8.1.1 Book Chapters 122
8.1.2 Journals 123
8.1.3 International Conferences 124
8.1.4 National Conferences 125
8.2 Future work 127

List of Algorithms

1 Client/Server code for every peer. 58
2 Visual DNS neighbor discovery protocol. 105

List of Figures

2.1 Examples of pervasive applications over a ubiquitous network. . . . 9

3.1 Bandwidth variation with range for different wireless technologies. 14
3.2 Different layers integrating the IEEE 802.11 architecture. 16
3.3 PLCP framing. 18
3.4 MAC layer frame format for IEEE 802.11. 19
3.5 Information contained in the frame control field. 19
3.6 The CSMA/CA mechanism. 20
3.7 Example of a distribution ESS aggregating two BSSs. 26
3.8 Bluetooth protocol stack. 27
3.9 Mutual impact of the Bluetooth and IEEE 802.11 technologies in terms of throughput. 31

4.1 Possible devices in a Personal Area Network. 34
4.2 Example of multi-hop routing in MANETs. 37
4.3 Illustration of the multipoint relay concept for node N. 42

5.1 A pictorial representation of the peer-to-peer application. 57
5.2 A client screen capture of the application for desktop/laptop. . . . 59
5.3 A client screen capture of the application for pocket PC. 60
5.4 Inquiry delay value for 100 tests. Distance between peers is $\simeq$ 1m. 61
5.5 The cumulative inquiry delay distribution for 100 tests. Distance between peers is $\simeq$ 1m. 62
5.6 Inquiry delay value for 100 tests. Distance between peers is $\simeq$ 10m. 62
5.7 Application transfer time for different shared resources with respect to distance. Peers are static. 64
5.8 Application transfer time for different shared resources with respect to peers speed. Spatial distance of about 5m. 64
5.9 The cumulative inquiry delay distribution for 100 simulated tests. Distance between peers is $\simeq$ 1m. 65
5.10 Inquiry delay value for 100 simulation tests. Distance between peers is $\simeq$ 10m. 66
5.11 Simulated transfer time for different file size with respect to distance. Peers are static. 67

5.12 Simulated transfer time for different file size with respect to peers speed. Spatial distance of about 5m. 67
5.13 The BlueHospital System Architecture. 69
5.14 Schematic BH_Patient system overview. 70
5.15 A screen capture of the application related to the heart rate of a monitored patient offered to a doctor or nurse. 73
5.16 Obtained application transfer time comparison of an ACL link using different DH_x packets as a function of distance. 75
5.17 Obtained application transfer time comparison of an ACL link using different DM_x packets as a function of distance. 75
5.18 The cumulative inquiry delay distribution for the 100 tests. Distance between the Bluehospital application and the BH_Room_Manager is 5m. 76

6.1 System architecture. 80
6.2 XML configuration file for a client. 81
6.3 Flow diagram between client and server. 82
6.4 Server application capture. 83
6.5 Application window for parameters configuration. 84
6.6 Client application capture. 85
6.7 Configuration time while varying distance between Bluetooth devices. 86
6.8 Configuration time (inquiry time excluded) when varying the number of close-by Bluetooth devices for different distance values. . . . 86
6.9 Details of an IEEE 802.11 beacon frame. 87
6.10 Cumulative distribution function for the SSID set analysed. 88
6.11 Generic SSID partitioning strategy. 88
6.12 Detailed SSID partitioning strategy. 89
6.13 Session key generation process. 91
6.14 Chain topology used to measure the propagation time for autoconfiguration beacons. 93
6.15 Autoconfiguration times for nodes at multiple hop distances from the initiating node. 94
6.16 OLSR topology updating time when joining the network at different hop counts from Station 1. 95
6.17 Autoconfiguration time when varying the number of terminals being configured at one hop. 96

7.1 EasyMANET protocol architecture. 100
7.2 Screen-shot of the Visual DNS interface. 103
7.3 User profiles exchange with Visual DNS. 104
7.4 Multi-service support in the EasyMANET platform. 106
7.5 Data transmission overhead when varying the amount of new information. 107
7.6 Visual DNS profile initialization time when varying the inter-HELLO interval (nodes at 1-hop). 107
7.7 Visual DNS profile reconfiguration time - due to a new node joining in the existing MANET for different network diameters. 108

7.8 Example of star topology with 5 nodes. 111
7.9 Initialization time varying the inter-HELLO interval (nodes at 1-hop). 112
7.10 Example of chain topology formed by 5 nodes. 113
7.11 Reconfiguration time due to a new node joining in the existing MANET for different network diameters. 113
7.12 Screenshot of a Visual DNS running example in OMNeT with an UDP broadcast. 115
7.13 Baseline scenario. 116
7.14 Initialization time when varying the inter-HELLO interval and the number of nodes maintaining density. 117
7.15 Initialization time when varying the inter-HELLO interval and the number of nodes maintaining the scenario size. 117
7.16 Initialization time when varying the inter-HELLO interval and the nodes speed. 118
7.17 Reconfiguration time due to a new node joining in the existing MANET for different number of nodes maintaining density. 119
7.18 Reconfiguration time due to a new node joining in the existing MANET for different number of nodes maintaining the scenario size. 119

List of Tables

3.1 Four address fields based on the value of the *To DS* and *From DS* bits. 19

6.1 Average time overhead associated with autoconfiguration tasks. . . 92
6.2 Comparison between BlueWi and the SSID-based autoconfiguration techniques. 97

7.1 Taxonomy for data dissemination in MANETs. 102
7.2 Initial configuration parameters. 110
7.3 Parameters of the baseline scenario. 114
7.4 Scenario sizes for different number of nodes. 115

Chapter 1

Motivation, Objectives and Organization of the Thesis

1.1 Motivation

Nowadays our lives increasingly depend on information and communication technologies. People want to be connected anytime, anywhere, and so the use of the Internet through the wide range of mobile devices available in the market is becoming more and more common in our societies, being even indispensable in some cases. Although wireless technology has existed for more than two decades [JJ87] and became commercially available over fifteen years ago, two main factors have contributed to its recent popularity:

- On the one hand, technological advances have allowed the massive proliferation of wireless technologies (such as Bluetooth, Wi-Fi, WiMax, GPRS, UMTS, GPS) and wireless devices (such as smartphones, personal digital assistants (PDAs), tablets PC, laptops).

- On the other hand, and strongly linked to these advances, the liberalization of the ISM (industry, science and medicine) frequency bands, that do not require any license, has encouraged the continued development of higher performance devices with competitive market costs.

Also notice that electronic devices are becoming much smaller, and can be integrated into nearly all everyday objects. Objects and appliances can thus react and operate in a context-sensitive manner and appear to be smart (without actually being intelligent), leading to the conclusion that pervasive computing and ubiquitous scenarios are becoming a reality. For all the above, it seems very interesting to develop prototype applications able to promote the use of all these new technologies and devices available.

1.2 Objectives of the Thesis

In this thesis we propose showing how some wireless technologies, beyond its traditional use, can contribute significantly to the development of pervasive applications based on ubiquitous computing. Along the document, we have tried to describe every proposal giving a useful and simple approach from the user's point of view.

The first objective of this thesis is having a first contact with short radio range wireless technologies and try to apply them in possible ubiquitous scenarios derived from environments such as academic, business or clinician.

The second objective is to propose systems for autoconfiguring quick and transparently the required parameters that allow a wireless device be integrated into a mobile ad-hoc network (MANET). More generally, the goal is to be able to configure an entire group of wireless devices attempting to join a MANET for different purposes. We can imagine for example, a rescue team, a group of persons in a meeting, or a group of students.

Since MANETs are infrastructureless, the third objective is to study mechanisms to solve the name-to-address translation and service discovery problems in MANETs. Solving these two problems efficiently in this type of networks is very important because it will determine the scope of the derived application. In other words, depending on the number and type of services offered by the network members, solutions for different kinds of environments will be made available.

The main objective of this thesis is to integrate the highest number of concepts and issues described above in a prototype platform (EasyMANET) able to provide solutions in different ubiquitous computing environments. This platform is open to future modifications and extensions. Furthermore, we want to evaluate EasyMANET through both real and simulated environments in order to validate the prototype and having the lowest possible cost.

After describing our distinct objectives in detail, we proceed by making a joint evaluation of all the previous proposals, obtaining a clear picture of the overall improvements achieved.

1.3 Organization of the Thesis

This thesis is organized as follows: in the next three chapters we make a background introduction to the technological framework and the different technologies used in this thesis, including related works in each of the referred research fields. So, in chapter 2 we make an introduction to ubiquitous computing and pervasive applications, which form the theoretical basis that underlies all the contributions of the thesis. Chapter 3 is dedicated to wireless technologies. Our focus is given to the Bluetooth and IEEE 802.11 standards since they provide the technology we use to create wireless links between the stations of short radio range wireless networks. The last introductory chapter, chapter 4, analyzes the state-of-the-art of several short radio range wireless networks (such as personal area networks, mobile ad-hoc networks, and spontaneous networks), focusing on issues such as routing protocols and autoconfiguration.

Chapter 5 discusses two initial proposals able to easily establish spontaneous networks without any fixed infrastructure. These proposals (named BluePeer and BlueHospital) rely on Bluetooth technology and the concept of peer-to-peer (P2P), and were a first contact with all the technologies described above that served us as proof-of-concept in order to make new proposals. The resulting experimental applications enable the network members to exchange both resources and monitoring information.

In Chapter 6 we focus our attention on solving the problem of configuring automatically (autoconfiguration) some parameters of the stations that attempt to join a mobile ad-hoc network (MANET) based on IEEE 802.11 technology. We propose two solutions to this problem: the first one, based on using Bluetooth technology, and the second one, based on the 802.11 technology itself. Since most mobile devices currently available offer both technologies, we tried to get, in each case, the most of their capabilities.

Chapter 7 introduces EasyMANET, an extensible platform whose main objective is to encourage the widespread adoption and use of MANETs by non-expert users. EasyMANET offers solutions to the first three problems defined by the IETF Zeroconf Working Group (i.e., *Interface Configuration*, *Name-to-address Translation* and *Service Discovery*), and provides two essential elements: an address autoconfiguration system, and a name resolution service known as Visual Domain Name Service (DNS). The autoconfiguration system allows users to join an 802.11-based MANET transparently, while Visual DNS offers a graphical view of MANET participants, giving users the possibility of accessing the services made available by other users.

Finally, in Chapter 8, we present a summary of the main results of this thesis, along with some concluding remarks. We also include a list of the publications related to the thesis, and we comment on future research works that can be derived from the work here presented.

Chapter 2

Ubiquitous Computing

Over the last 30 years, and due to the continuing advances in the fields of computer science, microelectronics, communication technology, and materials science, we have seen the power of microprocessors double about every 18 months. An equally rapid increase applies to some other technological parameters such as storage capacity and communications bandwidth. This continuing trend means that computers are becoming considerably smaller, cheaper, and more abundant: they are becoming ubiquitous, and are even finding their way into everyday objects. This is resulting in the creation of smart things that can access the Internet and its multiple resources, and even cooperate with each other. Today, Internet connects almost all the computers around the world. From a technological point of view, the *ubiquitous computing* could be described as the chance of connecting anything to the Internet, in order to provide information on anything, anytime, anywhere.

This continuing trend will lead to a paradigm shift in computing applications [RG00]. In this chapter, we illustrate the concept of ubiquitous computing and describe its technical foundation and current trends. In addition to reviewing the research accomplishments in these application research themes, we also outline some of the developed applications and remaining research challenges.

2.1 Overview

The term ubiquitous computing was coined twenty years ago by Mark Weiser, a researcher at XEROX's Palo Alto Research Center [Wei91]. Weiser saw technology only as a means to an end, which should take a back seat in order to allow the user to fully concentrate on the task itself. In this respect, the Personal Computer as an universal information technology tool would be the wrong approach, since its complexity would take up too much of the user's attention. According to Weiser, the computer as a dedicated device should disappear, while at the same time making its information processing capabilities available throughout our surroundings. While Weiser saw the term ubiquitous computing in a more academic and idealistic sense as an unobtrusive, humancentric technology vision, industry has since

coined the term pervasive computing with a slightly different slant [ULMT01]. Even though its vision is still to integrate information processing almost invisibly into everyday objects, its primary goal is to use such objects in the near future in the fields of electronic commerce and Web-based business processes.

The concept of ubiquitous computing deals with a world where computational technology and services embrace almost everything around us, and many computing devices are so deeply integrated within the environment that people do not even realize that they are using computers. The central concept is to harmonize with users through a digital environment that is aware of their presence and context, being able to provide personalized services according to their needs, besides being capable of anticipating their behaviour and responding to their movements. Applications may change or adapt their functions, information and user interface depending on the context and the client's profile [Wei93].

For this ubiquitous vision to become a reality, we must rely on small, hand-held, wireless computing devices that enable the interaction between users and their environments. These devices should offer functionalities that can be described, advertised and discovered by others. Moreover, they are able to inter-operate even though they have not been specifically designed to perform joint tasks. Therefore, thanks to the advanced technology in devices and low power wireless communication systems, consolidating the ubiquitous computing paradigm in a near future is a major source of motivation in the research and development of new systems.

In particular, context-awareness is intimately related to effective interaction with ubiquitous computing systems. Interaction is via objects, and thus situated within the context of a particular activity. Detailed knowledge of the parameters of this situation is necessary to infer an appropriate behaviour. Context incorporates several aspects:

- The physical environment: for example, the user location, the presence of other persons or objects in the same location, and the environmental conditions observed. For example, knowledge of user location can assist at inferring their current activity.
- Time can also reveal considerable information about the current user activity, for example, whether a particular person is occupied by professional or personal concerns.
- Device and network characteristics, especially in the case where an information service must be delivered to the end user. In this case, service provision parameters must be optimized, and the service adapted to meet the characteristics of its operating context.
- Information context is the semantic knowledge regarding the domain being investigated; an example of such knowledge is the short-term information needs of the user as they might be expressed in a query. Information context also includes the user profiles that reveal long-term interests, for example via demographic details.
- Social context is perhaps the most challenging aspect of context-awareness as it relates to information relevant to the characterization of a situation

that influences the interactions of one user with other users. Automated interpretation of a social situation is challenging and requires higher level abstractions than those provided by sensing, even more so in those cases where the user is part of a group.

Currently, there are many companies and research centers actively working on the issue of context-awareness or, more generally, on ubiquitous computing [BDR07]. In particular, several proposals focus on smart spaces and intelligent environments ([SKB+98]; [HHS+99]; [KBM+02]; [FCK+05]; [SFD+06]; [Sma07]), where it is expected that smart devices all around us maintain updated information about their locations, the contexts in which they are being used, along with relevant data about the users. There is no doubt that pervasive computing systems will provide flexible services and unsuspected benefits. However, there are still only a few examples of pervasive computing environments moving out from academic laboratories into our everyday lives. This occurs because their design is still a difficult task, requiring much theoretical and practical work. Moreover, it is complex to define what a real pervasive system should be like. The OneWorld project [GDL+04] builds an architecture that provides an integrated and comprehensive framework for developing pervasive applications. It includes a set of services that help to structure applications and directly simplify the task of coping with constant change.

2.2 Technical foundations

The driving force behind these dynamic technological developments is the field of microelectronics, where progress has truly fulfilled Moore's Law during recent decades. Recent achievements in the fields of microsystems and nanotechnology are also of increasing importance. One example are tiny embedded sensors suitable for detecting a multitude of environmental parameters. Another interesting development is that of radio sensors that can transmit pressure or temperature changes across several meters without any explicit source of power. The energy required for transmitting the measured data, and also for adding an individual identification code to the data, is obtained through the measurement process itself using piezoelectric or pyroelectric materials [IEE88].

The so-called smart labels or radio tags also operate without a built-in source of power. Such labels contain transponders that receive a high frequency signal from a distance of up to two meters. The energy of the signal is used by the transponder to decode the message, power its internal information processing capabilities, and send back its reply. This allows up to several hundred bytes to be transmitted and received over the air within the space of a few milliseconds. The transponders have form factors of a few square millimeters, are as thin as a sheet of paper, and are available as flexible address labels for less than one Euro a piece.

Complete computer systems-on-chip (SoCs) measuring only a few square millimeters and including several kilobytes of memory (enough for a simple operating system) can already be manufactured for a few Euros. This technology is mainly used for smart cards, but can also be found in embedded systems, where proces-

sors are integrated into all sorts of appliances to carry out control tasks. These processors, together with suitable sensors, I/O-interfaces and communication capabilities, are the primary components that could make real world objects smart. For objects that communicate with each other, research is underway to create processors that also contain appropriate communication components on the chip itself [BDM06, FB10].

Significant advances have also been made in the field of wireless communications. For ubiquitous computing, short-range communication technologies that use very little energy are extremely relevant. One such example is the already well-established WLAN technology, but even more relevant are wireless room networks. For the latter, the Bluetooth standard is currently becoming a de facto standard. Bluetooth modules are presently available at form factors of about half the size of a match box, but by integrating memory, high-frequency, and digital components on a single chip, much smaller sizes are expected in the near future. Improved methods for determining the position of mobile objects (for example, using satellite-based systems such as GPS [EC96] or the radio-based methods used by mobile phones) are also being developed.

Finally, recent developments in the field of materials science will give computers of the future a completely different shape, or even mean that computers are no longer recognizable as such because they blend into their surroundings. One important example in this context are light-emitting polymers, which enable displays consisting of highly flexible, thin and bendable plastic foils to be created. Research is also taking place into electronic ink and smart paper, which will enable pen and paper to become truly mobile input/output media. However, this technology is still some years away from becoming usable in practice, for example in the form of a computer as a foldable road map. Another significant option currently under development is laser projection from within eyeglasses directly onto the retina, as a replacement for traditional output media.

2.3 Pervasive applications

The potential of applications using smart everyday objects seems immense, especially if we assume that objects could, theoretically, use spontaneous networking technology to cooperate with each other, access information stored in online databases, or use any suitable Internet-based service available. Figure 2.1 shows an example of an ubiquitous network and a set of possible pervasive applications over it.

The ultimate vision of ubiquitous computing, however, extends well beyond such applications, towards scenarios that verge on science fiction. We are talking here about ordinary things such as pencils that digitize (and process) everything that is written with them, or suitcases that remember the places they travelled to and the objects that were carried in them (or even recall conversations they overheard?). Apart from being an extraordinary technical and organizational challenge in itself, extending the Internet into everyday objects also raises the question of how we would communicate with our smart objects, and how we could put this new technology to good use for society as a whole.

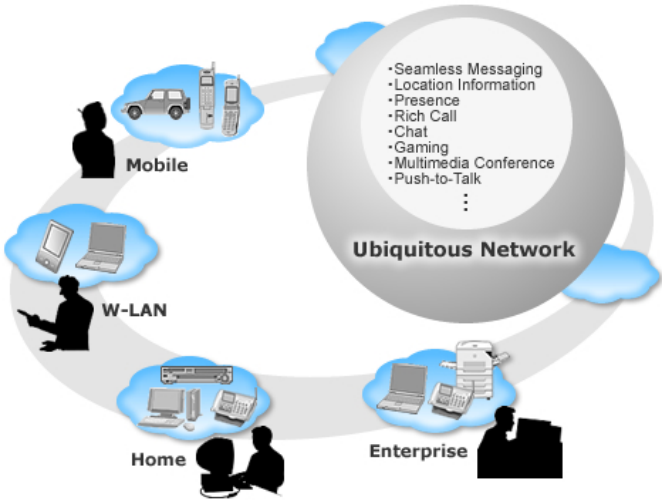

Figure 2.1: Examples of pervasive applications over a ubiquitous network.

From the point of view of the research community, pervasive computing has been receiving attention for more than fifteen years. Although new technologies are emerging, and the target scenarios are heterogeneous (museums, schools, malls, official departments, hospitals, military), a great number of leading technological organizations are exploring pervasive computing. The most crucial objective is not necessarily to develop new technologies, but finding ways to integrate existing technologies with a wireless infrastructure in real settings.

Tourism was one of the pioneer areas in the adoption of ubiquitous applications, mainly due to the high number of potential users, and also because, in case of a system failure, the consequences were not as dangerous as in other environments (e.g. hospitals). Other development areas began to emerge gradually, and so we can now find a multitude of applications designed for a wide variety of sectors and fields. Next, we will review some of the most important contributions in recent years.

The Cyberguide project [AAH+97] consists of a series of prototypes of a mobile, hand-held context-aware tour guide that, based on the user's current location and history of past locations, is able to provide services concerning location and information. The project was developed for indoor and outdoor applications. In both cases, Cyberguide uses as a positioning solution based on infra-red (indoor) and GPS technologies (outdoor). Cyberguide was influenced by earlier location-aware works such as the PARCTab at Xerox PARC [WSA+95], the InfoPad project at Berkeley [LKAA96], the Olivetti Active Badge system [WHFG92] and the Personal Shopping Assistant at AT&T [ACK94]. Similar systems to Cyberguide have also been proposed by other researchers, including the GUIDE [DMC+98] project proposed at Lancaster University, the HIPS [BM97] project from the University of Siena, and the ILEX [COO99] project developed at the University of Sussex that talks about intelligent labels.

The CoolTown project [KBM+02] developed at HP Laboratories focuses on building ubiquitous computing systems by integrating web services to enhance communication with mobile users, to provide location-specific services in visited areas, and to provide interaction with the objects that they encounter. The goal is to extend the Web technology together with wireless networks and mobile devices to create a virtual link between the physical entities, including users, and services provided by the environment. The Websign project [SCP+04] is a component of the CoolTown research program which allows users to visualize services related to physical objects of interest. Although Websign can be adapted to provide tourist guide services, its use generally addresses the interaction between users and services associated with physical objects. The Rememberer system [FFF+02], also part of the Cooltown project, is a tool for capturing personal experiences during a museum visit that helps users to build a record of their experiences, which they can use during or after their visit. The record consists of web pages on the statement of the visit, including notes and photos. It is aimed primarily for visitors overwhelmed by the enormous amount of information presented in the museum. Location is identified using infra-red technology and RFID sensors. Other works related to the Cooltown project include [SSKK01] and [SS02].

At the Tokyo University, the digital museum project [Sak98] uses smart cards to detect the proximity of visitors and then provide information about the exhibited objects. The information provided can be based on a static profile stored previously in a smart card. Similar work has also been done in [Ing99], where an infra-red infrastructure and wireless LAN connections were used for connectivity and location awareness, respectively.

Another environment that has been studied lately is the ubiquitous hospital, where clinical parameters can be monitored automatically. This idea is very important since it can improve the reliability and speed of some hospital procedures, which are extremely important when human lives are at stake. In iHospital [HBS06] the authors create an interactive and pervasive system to be used in hospitals, showing that issues that seem trivial in the laboratory become major obstacles in the real world. The system aims to efficiently coordinate the operations of a large hospital. The system can also monitor the position of users through some mobile devices.

In [CFBM05], authors used a network simulator to evaluate the feasibility and performance of using Bluetooth as the underlying networking technology to establish context-aware services. They compare results obtained from simulation with those obtained from a real test-bed. Authors observed that simulation results show a much smoother behaviour than those obtained in real experiments.

Finally, the Massachusetts Institute of Technology (MIT) has a project called Oxygen [Rud01] that envisions a future where the goal is to have a very large number of ubiquitous computing devices in the environment, enabling people to interact with them in a very simple way, i.e. through natural perceptual interfaces of speech and vision, to perform the tasks they want. In other words, devices will be freely available and easily accessible as oxygen is today.

2.3.1 Future trends

We are moving gradually towards the final vision of ubiquitous computing, where almost all everyday objects may communicate and cooperate with each other. However, this vision requires a close cooperation between three converging areas of Information and Communications Technology: computing devices, communications, and user interfaces. Future smart devices will have various shapes and sizes and will be designed for different task-specific purposes. These devices will rely on some kind of wireless communication technology (such as 802.11, Bluetooth, or ZigBee) to act intelligently, being able to create the most effective form of connectivity in a given scenario. Finally, the user interface represents the union point between technology and human users. Natural interfaces, based on both speech and gesture recognition, will facilitate a richer variety of communication capabilities.

Envisioning concrete applications is not easy. However, the appropriate combination of these three areas yield up to a wide range of applications not only in traditional areas (e.g., healthcare, education, transport, monitoring, tourism information, and smart networking), but also in those where the Internet already plays an important role, such as mobile commerce, telematics, and entertainment. Emerging technologies supporting those applications include wearable computing, smart homes, smart toys, intelligent environments, and augmented reality, among the many other imaginable. The 21st century will be characterized by the applicability of near-invisible technology that is easy to spread throughout the environment.

2.4 Summary

The key idea behind pervasive computing is to deploy a wide variety of computing devices throughout our living and working spaces. These devices cooperate with each other to offer network services, with the ultimate goal of seamlessly assisting people in completing their tasks. Pervasive computing marks a major shift in computing, moving away from the current computing paradigm to become more human-centric.

Research about pervasive computing has been around for more than ten years. People are fascinated by the possibilities of pervasive computing due to the proliferation of inexpensive mobile computing devices, wireless communication and sensing technologies, setting up a new kind of intelligent environment where applications can transparently search and use services without the users' intervention. Despite few real-world applications have been developed and deployed up to now, the constant pace of innovation will definitely contribute to achieving Weiser's vision of an ubiquitous computing world.

The following chapters will describe the research work we have done in the area of pervasive computing to define ubiquitous systems that fit into different types of environments. These systems are prototypes from which more sophisticated versions can be developed when targeting real ubiquitous scenarios. Selected environments include spontaneous networks in academic and business scenarios that

allow P2P connections to exchange any type of resources, MANET (Mobile Ad Hoc Network) environments that allow an organized team to communicate independently in different situations, and also intelligent hospitals where it is possible to monitor patients automatically. For each prototype, we provide a detailed description, including the system architecture, the application, and the implementation details, along with evaluations results from both testbed and simulation environments.

Chapter 3

Wireless Technologies

The development of communication networks was a significant step for mankind, undoubtedly agilizing everyday's tasks and improving the quality of life. Telecommunication and computer networks began with a strong emphasis on wires, both for the communication infrastructure and for the last hop where the actual connection towards the users' terminals takes place. In recent years this trend has shifted towards wireless networks, especially at the user side. This shift comes from the demand of improved mobility support and greater flexibility, so as to face the challenges of our fast-changing society.

In this chapter we analyze the state-of-the-art of the wireless technologies that are being widely adopted by both consumers and industry. The main focus is given to IEEE 802.11 and Bluetooth technologies, since they are usually the main technologies of choice for Personal Area Networks (PANs) and Mobile Ad-hoc Networks (MANETs), which are currently some of the most common physical support for ubiquitous and pervasive applications (chapter 4).

Therefore, section 3.2 will be dedicated to the IEEE 802.11 standard, as well as to some of its annexes that are relevant to this thesis. Section 3.3 offers an overview of Bluetooth, which is also receiving much interest from both consumers and industry. In this section we also present a study on the mutual interference between Bluetooth and IEEE 802.11 technologies. Finally, section 3.4 concludes the chapter.

3.1 Overview

In the last few years wireless networks have become ubiquitous. This is due to reasons such as the current life style, where there is the need to stay constantly connected to local area networks or to the Internet, and also due to other needs such as supporting mobility and offering greater flexibility. Depending on their purpose, we can separate wireless networks into four groups: Personal Area Networks (e.g.: Bluetooth, UWB), Local Area Networks (e.g.: ZigBee, IEEE 802.11a/b/g/n), Metropolitan Area Networks (e.g.: WiMAX), and Wide Area Networks or Cellular (e.g.: GSM/2G, GPRS/2.5G, UMTS/3G, HSDPA/3.5G, LTE/4G).

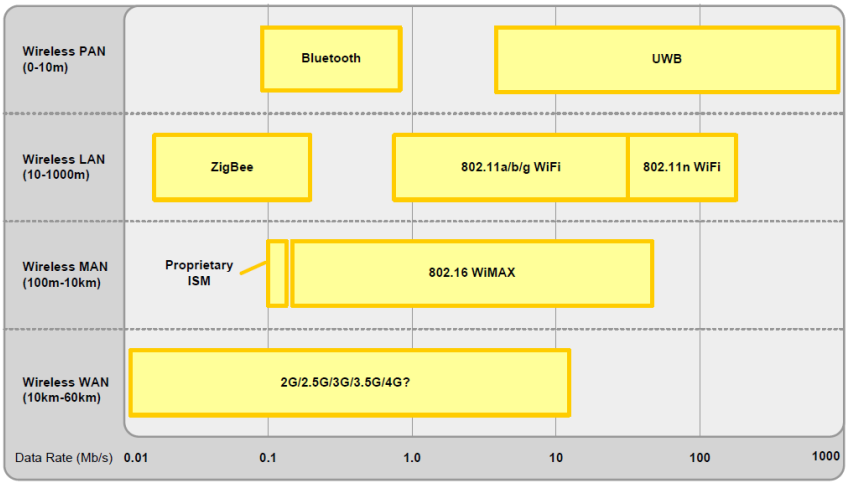

Figure 3.1: Bandwidth variation with range for different wireless technologies.

Figure 3.1 shows the differences between some wireless technologies available in terms of communication range and bandwidth. That figure evidences the inverse relationship between bandwidth and range; in the wired networks field, a similar situation happened some years ago.

The appearance of both wireless local area networks (WLAN) and wireless personal area networks (WPAN) offers several advantages, besides those referred before for wireless networks in general. Among these we have compatibility with the already available wired networks, ease of installation, cost reduction, ease of management, scalability, passing through physical barriers, etc. From a commercial point of view, we can also appreciate the advantages that this sort of networks offer. The installation of the so-called *hot-spots* allows cafes, pubs and restaurants to attract more clients; it also allows airports, hotels and trains to receive an extra income by offering Internet access, and it can also improve the productivity of companies by allowing workers to access the internal network while moving through zones where wired connections are not possible.

The main purpose of wireless networks is to support computational and communication services while moving. Depending on the type of network used, though, the techniques employed vary. For instance, cellular networks are typically characterized by a single wireless hop before reaching the wired portion of the network. Also, the space is divided into cells, where each user is assigned to a cell's base station. As we will see in the rest of this chapter, IEEE 802.11-based wireless LANs differ from cellular networks in several ways. Concerning Bluetooth-based networks (see section 3.3) and mobile ad hoc networks (topic of the next chapter), the differences towards cellular networks are even more evident, as we will show.

3.2 IEEE 802.11

The IEEE 802.11 standard [IEE99] is a technology whose purpose is to provide wireless access to local area networks (WLANs). Stations using this technology access the wireless medium using either the Point Coordination Function (PCF) or the Distributed Coordination Function (DCF).

The Point Coordination Function is a centralized access mode optionally used in a Basic Service Set (BSS) when a point coordinator (PC) is available. The PC is typically an *access point* (AP), and so the stations are said to operate in infrastructure mode. When relying on the PCF, contention-free periods (CFP) and contention periods (CP) alternate over time. The regular generation of beacons allows stations to associate and synchronize with the PC. Typically a CFP is started after a beacon management frame, followed by a CP; together they form a superframe. During the CFP there is no contention, and so stations are simply polled by the PC. The CF-End control frame is transmitted by the PC to indicate the end of the CF period, and the beginning of the CP. During the CP stations access the medium using the DCF.

The Distributed Coordination Function (DCF) uses a listen-before-talk scheme named *carrier sense multiple access* (CSMA) with *collision avoidance* (CA). It is used by stations in a BSS during the CP, and also by stations in an IBSS (Independent Basic Service Set) operating in ad hoc mode. The CSMA technology distributes the medium access task among all stations, making every station responsible for assuring the delivery of *MAC service data units* (MSDUs) and reacting to collisions. The collision avoidance (CA) scheme is used to reduce the probability of collisions between different stations. To achieve this, it applies a backoff procedure before initiating a transmission if the medium was not previously idle. Stations select a random number of slots to wait before transmission in an interval between 0 and the current *contention window* (CW) value. The CW is set initially to the minimum value defined for the radio technology being used (CWmin), being increased when consecutive collisions occur up to a maximum value (CWmax).

The CSMA/CA mechanism shows good adaptation to different numbers of transmitting stations, and probabilistically shares the channel equally among all transmitting stations. However, it offers no mechanisms to perform traffic differentiation, making it very difficult to offer QoS support. The IEEE 802.11e working group was created to focus on this issue, and a new international standard was completed at the end of 2005.

Products using the IEEE 802.11 standard are experiencing a growing interest by companies all over the world. This is due to a good balance between cost, range, bandwidth and flexibility. The bandwidths defined by the standard currently range from 1 to 54 Mbps, but other standards being developed in the 802.11 family shall offer greater bandwidth, though maintaining the same frequency bands. The IEEE 802.11 standard offers the two operation modes referred before: *Point Coordination Function* (PCF) and *Distributed Coordination Function* (DCF). PCF can only be used in the infrastructure mode where access points are responsible for the coordination among nodes; DCF is a fully distributed mechanism which allows stations to compete for the medium without requiring an access point for support.

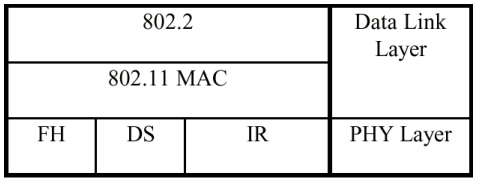

Figure 3.2: Different layers integrating the IEEE 802.11 architecture.

As shown in figure 3.2, IEEE 802.11's architecture allows, at the physical level, to use *Frequency Hopping* or *Direct Sequence* modulation techniques, apart from the modulation techniques defined on the IrDA standard [IrD01]. Above IEEE 802.11's MAC level an 802.2 layer is available, being responsible for the logical control of the channel (LLC). In the next section we offer more details relatively to IEEE 802.11's physical layers.

3.2.1 Physical level

The IEEE 802.11 standard specifies three physical layers techniques that were standardized in 1997. Two of them were designed for operation at the ISM (Industry, Scientific and Medical) frequency band (2.4 GHz); these are the Frequency-hopping (FH) and Direct-sequence (DS) spread-spectrum techniques. A physical layer using infrared light (IR) was also defined.

In the ISM bands, the initial IEEE 802.11 standard defined operation with data rates of 1 or 2 Mbps using either *Direct Sequence Spread Spectrum* (DSSS) or *Frequency Hopping Spread Spectrum* (FHSS) modulation techniques. We will now offer more details for the DSSS technology since the two remaining physical layer technologies are currently not being used.

The DSSS technology works by modulating data with a second pattern (chipping sequence). In IEEE 802.11, that DSSS sequence is known as Barker's code, which is simply a sequence of 11 bits (10110111000) that has some mathematical properties that makes it ideal to modulate radio waves. The basic data flow goes through a block that does the XOR operation with the Barker code to generate a series of objects known as chips. Each bit is coded by the 11 bits of the Barker code, and each 11 chips group codes a data bit.

To transmit at 1 Mbps it makes use of BPSK (*Binary Phase Shift Keying*) modulation with a phase change per bit. To achieve transmission at 2 Mbps it uses QPSK (*Quadrature Phase Shift Keying*) modulation. QPSK uses four rotations (0, 90, 180 and 270 degrees) to code two information bits on the same space where BPSK codes only one. We therefore have a trade-off between power and range. Notice that it is not allowed for mobile radios to transmit wirelessly with more that 1 Watt EIRP (*Equivalent Isotropic Radiated Power*) in the USA and with more than 100 mW in Europe. Therefore, as nodes separate, the radio adapts itself by using a slower and less complex mechanism to send data.

IEEE 802.11a The IEEE 802.11a technology is a physical layer annex to IEEE 802.11 for operating on the 5 GHz radio frequency. It supports several different data rates between 6 and 54 Mbit/s. Since the 5 GHz band is used, it suffers from less RF interferences that those physical layers that operate on the 2.4 GHz band (e.g. IEEE 802.11b and 802.11g) since the penetration capability of radio waves at the 5 GHz frequency is lower. So, by offering high data rates and low interference, the IEEE 802.11a technology allows achieving good results supporting multimedia applications in environments with several users. The only drawback is that more access points are required to cover a similar area than for IEEE 802.11b or 802.11g.

Concerning modulation, it uses Orthogonal Frequency Division Multiplexing (OFDM) with 52 sub-carriers.

IEEE 802.11b The IEEE 802.11b specification enhances the IEEE 802.11 physical layer to achieve higher data rates on the 2.4 GHz band. It uses another modulation technique known as *Complementary Code Keying* (CCK) to achieve 5.5 and 11 Mbps. Instead of using the Barker code, CCK uses a series of codes known as Complementary Sequences. The CCK technique uses 64 unique codewords that can be used to code the signal, being that a particular codeword can encode 4 or 8 bits (instead of a single bit as represented through a Barker symbol).

The final solution consists of combining the DSSS (*Direct Sequence Spread Spectrum*) techniques based on CCK with DQPSK modulation, which is the key for achieving data rates of 5.5 and 11 Mbit/s.

IEEE 802.11g The IEEE 802.11g is a most recent specification available for IEEE 802.11's physical layer. It results of merging IEEE 802.11a with IEEE 802.11b, which allows the IEEE 802.11g technology to achieve data rates similar to IEEE 802.11a (54 Mbit/s). The main advantage of IEEE 802.11g is that it maintains compatibility with more than 11 million Wi-Fi products (IEEE 802.11b) already sold.

The IEEE 802.11g standard defines several modulation types:

- ERP-DSSS: refers to physical layers using Direct Sequence Spread Spectrum (DSSS) modulation, and exists to ensure compatibility with the original IEEE 802.11 standard.
- ERP-CCK: refers to physical layers using Complementary Code Keying (CCK) modulation as defined for IEEE 802.11b, and exists also to ensure compatibility with that standard.
- ERP-PBCC: refers to physical layers using extended rate Packet Binary Convolutional Coding (PBCC) modulation. PBCC was added as an option in the IEEE 802.11b. In 802.11g, the ERP-PBCC option also supports data rates of 22 and 33 Mbps.
- ERP-OFDM: refers to physical layers using Orthogonal Frequency Division Multiplexing (OFDM) modulation. OFDM was defined in the IEEE 802.11a supplement. In IEEE 802.11g, ERP-OFDM supports data rates of 6, 9, 12, 18, 24, 36, 48, and 54 Mbps.

Figure 3.3: PLCP framing.

- DSSS-OFDM: refers to physical layers using hybrid DSSS-OFDM modulation. DSSS-OFDM was added in the IEEE 802.11g standard and is an optional mode that does not use the Extended Rate PHY (ERP) protection mechanism. Instead, DSSS-OFDM combines the DSSS preamble and header with the OFDM payload, supporting rates similar to ERP-OFDM.

IEEE 802.11n In January 2004, the IEEE announced that it had formed a new 802.11 Task Group (TGn) to develop a new amendment to the IEEE 802.11 standard for local-area wireless networks, it was published in October 2009. The real data throughput ranges from 54 Mbit/s to 600 Mbit/s (with the use of four spatial streams at a channel width of 40 MHz), and theoretically is up to 10 times faster than IEEE 802.11a or 802.11g, and near 40 times faster than IEEE 802.11b. IEEE 802.11n also offer a better operating distance than 802.11b/g networks.

IEEE 802.11n builds upon previous 802.11 standards by adding MIMO (multiple-input multiple-output) and orthogonal frequency-division multiplexing (OFDM). MIMO uses multiple transmitter and receiver antennas to allow for increased data throughput through spatial multiplexing and increased range.

3.2.2 IEEE 802.11 frame format

Although designed for total compatibility with existing IEEE 802.3 based networks, IEEE 802.11 frames differ from the former due to the requirements of the wireless medium itself.

The physical level for IEEE 802.11 is divided into two sub-layers: the Physical Layer Convergence Procedure (PLCP) and the Physical Medium Dependent (PMD) sub-layer. While the PMD sub-layer is the one responsible for actually transmitting the bits on the channel, the PLCP sub-layer works as the interface between the MAC layer and the radio transmission itself. The PLCP sub-layer also adds its own headers to transmitted frames. Figure 3.3 shows the generic PLCP framing structure.

The preamble depends on the physical level and includes a synchronization sequence of 80 bits, apart from a frame delimiter of 16 bits whose sequence is: 0000 1100 1011 1101. Concerning the PLCP field, it is always transmitted at 1 Mbps for IEEE 802.11 and contains information that allows decoding the frame at the physical level, consisting of:

- PLCP_PDU length word, which contains the size of the packet in bytes.
- PLCP signaling field, which contains information relative to the transmission's bandwidth.
- Header Error Check Field, that is a 16 bits CRC field used to detect errors on the frame's header.

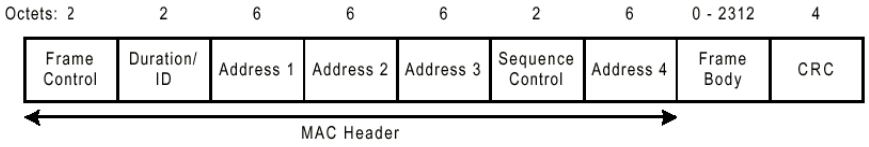

Figure 3.4: MAC layer frame format for IEEE 802.11.

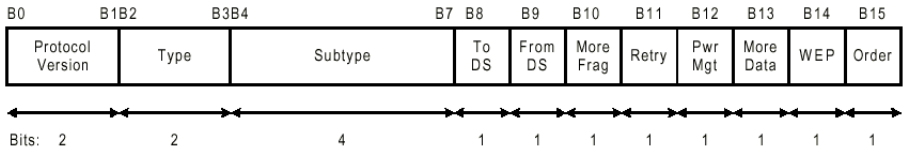

Figure 3.5: Information contained in the frame control field.

We now proceed to analyze MAC layer frames in detail. An IEEE 802.11 MAC frame has the format shown in figure 3.4. IEEE 802.11's MAC layer adds both a header and a trailer - the final 32 bits CRC field designed to detect errors on data.

The frame control field contains a large amount of data, as shown in figure 3.5, relative to the protocol version, the frame's type and subtype, as well as other flags with different purposes.

The *Duration/ID* field's content depends on the operation mode, being either the destination station identifier on power save mode or a duration used on the calculation of NAV (*Network Allocation Vector*).

The next four addresses are used to increase the flexibility, so that address 1 refers to the destination and address 2 to the source; addresses 3 and 4 identify source and destination on those situations where one or more access points are used for communication among two wireless nodes. Table 3.1 resumes the contents of each of the fields depending on *To DS* and *From DS* bits. Notice that SA and DA refer to the actual source and destination addresses, while BSSID is the address of the access point involved in the frame relaying process. When two access points must communicate wirelessly, their addresses are referred as RA and TA (receiver, transmitter), being SA and DA the actual source and destination of the data.

As for the sequence control field, it is used for both defragmentation and to discard duplicates frames. It consists of two fields - fragment number and sequence number - that define the frame and the fragment within the frame, respectively.

Table 3.1: Four address fields based on the value of the *To DS* and *From DS* bits.

Bit To DS	Bit From DS	Direction 1	Direction 2	Direction 3	Direction 4
0	0	DA	SA	BSSID	N/A
0	1	DA	BSSID	SA	N/A
1	0	BSSID	SA	DA	N/A
1	1	RA	TA	DA	SA

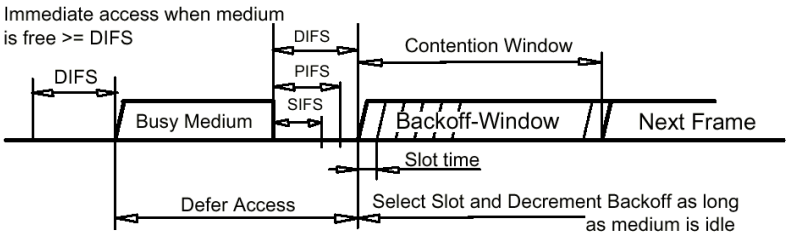

Figure 3.6: The CSMA/CA mechanism.

3.2.3 Distributed Coordination Function (DCF): CSMA/CA

The CSMA/CA access mechanism (*Carrier Sense Multiple Access/Collision Avoidance*) used by most Wireless LANs is the core of the protocol, specifying when a station must listen and when it should transmit. The basic principles of CSMA/CA are therefore: to listen before transmitting, and sending messages asynchronously (without connection). There are neither bandwidth nor latency guarantees.

The CSMA/CA mechanism derives from the CSMA/CD (*Collision Detection*) mechanism used by Ethernet. The only difference is that it avoids collisions (see figure 3.6). The CSMA/CA protocol starts by listening to the channel (*carrier sensing*) and, if it is free, the first packet on the transmission queue is sent. If it is occupied (due to the transmission of other nodes or interference) the node waits until the current transmission finishes, and it enters a contention period (waits for a random amount of time). When the contention timer expires the node sends the packet if the channel is still free. The node choosing the lowest contention time wins and transmits the packet. The remaining nodes merely wait for the following contention period to take place, which occurs when the packet transmission ends. Since contention depends on a random number and is done per packet, all the nodes have the same chances of accessing the channel (on average). Moreover, collisions can not be detected once transmission begins; since the radio requires some time to switch from transmission to reception mode, the contention mechanism makes use of time slots. So, transmission can only take place at the beginning of a slot, whose duration is 50 ms for 802.11 FH (Frequency Hopping) and 20 ms for 802.11 DS (Direct Sequence). This causes the contention time to increase, but reduces collisions significatively (it is impossible to eliminate them completely).

3.2.4 Point Coordination Function (PCF)

Apart from DCF, there is another operation mode known as PCF that can be used to support services with temporal restrictions, such as video and voice. This is possible since PCF operation is related to a smaller interval between frames (PIFS), so that access to the medium is obtained before any regular node using DCF. By making use of this higher priority, an access point can poll the different stations associated with it, thereby controlling access to the medium.

Though an access point has priority when accessing the medium, it must allow, between polling processes, nodes to send frames using distributed channel access techniques. Otherwise, nodes using such techniques would never have chances of transmitting, suffering from starvation.

3.2.5 MAC-level retransmissions

As referred before, the main problem of the CSMA/CA protocol is that the transmitter can not detect collisions on the wireless medium. Moreover, there is a greater error rate on the air than on a cable, so the chances for a packet to become corrupt are higher. TCP reacts poorly towards MAC level packet losses; so, most MAC layer protocols implement acknowledgment and retransmission functionality to avoid this problem. The principle is quite simple: each time a node receives a packet it immediately sends a short acknowledgment message (ACK) back to the source to indicate that the packet has been received without errors. If, after sending a packet, the source does not receive and ACK, it assumes that the packet has been lost and retransmits after a new contention period. Most MAC layer protocols use a *Stop & Wait* mechanism through which they only transmit the next packet on the queue after the arrival of the current packet is confirmed (it retries up to a certain number of times). This technique simplifies the protocol and avoids that packets arrive unordered.

ACK messages are part of the MAC protocol, so it respects their delivery to avoid collisions - contention starts after the ACK packet is transmitted. These acknowledgments are totally different from those used in TCP since they operate at a different protocol layer. *Broadcast* and *multicast* packets do not benefit from ACK frames since there is no specific destination, so they will be lost frequently.

3.2.6 RTS/CTS

The main effect to take into consideration when transmitting a radio wave is signal attenuation. It is due to attenuation that the hidden terminal problem is likely to occur. This problem takes place in those situations where not all the nodes can listen to the rest since the signal attenuation is too strong. Taking into account that transmissions are based on a carrier sensing mechanism, nodes ignore those that are far away, and so can transmit at the same time. Generally, this is good since it allows frequency reuse. However, for a node located between two other nodes, simultaneous transmissions have a similar power, and so they collide. From the point of view of that intermediate node, all the information transmitted is lost.

The main problem with the carrier sensing mechanism is that the transmitter tries to estimate if the channel is available on the receiver based solely on local information, which results in errors. A simple and elegant solution (proposed by Phil Karn on his MACA protocol [Kar90] for AX.25) is using RTS/CTS (*Request To Send/Clear To Send*). RTS/CTS is a handshake mechanism: before transmitting a packet the transmitter sends an RTS and waits for a CTS from the destination node. The reception of a CTS indicates that the receptor has received an RTS, which means that the channel is available on that area. At the same time, each node on the receiver's transmission range listens to the CTS (despite possibly not

listening to the RTS), deducing that a transmission is going to take place. The nodes listening to the CTS potentially could provoke collisions on the receiver, supposing that the channel is symmetric. Since these nodes can not listen to the transmission, both RTS and CTS packets contain the expected duration of such transmission. This characteristic of RTS/CTS is the one that allows avoiding collisions: all the nodes avoid accessing the channel after listening to the CTS even if the carrier sensing mechanism indicates that the medium is idle.

The RTS/CTS mechanism has another advantage: it reduces a collision's overhead on the medium (collisions are shorter). If two nodes try to transmit on a same slot of their contention windows, their RTS will collide and no CTS is received; that way all that is lost is a simple RTS. Without this mechanism, an entire packet would be lost. Since the RTS/CTS mechanism adds a significant overhead, it is not used with small packets or in lightly loaded networks.

3.2.7 IEEE 802.11i: Security

One of the concerns when standardizing a new wireless access method is security. Usually, a wireless transmission that relies on a spread-spectrum technology cannot be interpreted without knowing the spreading code or hopping sequence. By standardizing this spreading method, anyone can easily eavesdrop on any transmissions using the standard. To fix this, a method of encrypting the payloads of the MPDUs was introduced with the 802.11 standard.

When the initial 802.11 standard was released, it included two methods of authentication and one method of encryption. The two authentication methods are *Open System Authentication* and *Shared Key Authentication*. The only encryption mechanism available was *Wired Equivalence Privacy*, or WEP.

Due to flaws with shared key authentication, the open system authentication with encryption provides better security than the shared key authentication with encryption mode. By requiring encryption, clients will have to know the encryption key to join the network, inherently authenticating them.

3.2.7.1 Wired equivalent privacy (WEP)

When the IEEE released the initial 802.11 standard, all of the security relied on the Wired Equivalent Privacy (WEP) algorithm. This algorithm was responsible for both authentication and encryption. The name came from the idea that the algorithm would provide the same security attributes inherent to a wired medium. The original standard only included the option of using a 40-bit encryption key. This key size was chosen because of the US restrictions on exporting cryptography technology at that time. By choosing a small key size, the IEEE hoped to make the algorithm exportable to other countries without modification.

With only a 40-bit key, brute-force attacks were practical for many organizations with access to fairly modest computing power at the time of the release (1999). For this reason, many manufacturers offered an extended 104-bit key (128-bit total with the 24-bit IV) to eliminate brute-force out as a method of attack [BGW01]. All of the text covering the WEP operation covers the 40-bit key; however, the process is identical for 104-bit keys.

The encryption mechanism that WEP employs is the RC4 algorithm [Riv94, HM02] from RSA Data Security, Inc. RC4 is a stream cipher, which means that it encrypts each message one bit at a time, as apposed to block ciphers that encrypt groups of bits at a time. The design of RC4 is to simulate a one-time pad.

The one-time pad, or Vernam's Cipher, is an unbreakable encryption method when provided with a truly random key [Ran95]. The key must be at least as long as the data that needs to be encrypted, and both parties must have identical copies of the key. The key must be unpredictable and can never be reused.

Due to these strict requirements, key distribution becomes unfeasible for widespread implementations or the encryption of large amounts of data. Since each message bit requires an individual key bit, encrypting a 4.7GB DVD would require a 4.7GB key. The idea of RC4 is to simulate a one-time pad by using a pseudo random number generator (PRNG) to produce the key-stream used to encrypt the data. The key-stream produced is controlled by an encryption key. Therefore, the same encryption key always produces the same key-stream. The PRNG is a one-way function in that the key-stream cannot be given to the PRNG to produce the original encryption key. It accomplishes this by creating an ordered 256-byte array and then randomizing its contents based on the encryption key's length and value. It then produces a key-stream from the array while continually randomizing the array to prevent the key-stream from repeating.

The encryption operation itself is relatively simple. Each bit of plaintext is XORed with a bit of the key-stream to produce the ciphertext. The other party then XORs the ciphertext with the the same key-stream to reproduce the plaintext. The only difference between this operation and the one-time pad is the source of random numbers.

3.2.7.2 Wi-Fi Protected Access (WPA, WPA2)

The Wi-Fi Protected Access (WPA) certification was first developed when the 802.11i revisions were still being made to the standard. It was released as soon as possible because of the apparent security flaws with the WEP algorithm. It implements most, but not all of the standard. Most noticeably, the Counter-mode/CBC-MAC Protocol (CCMP) was not yet included. One of the extra things that WPA offered outside of the 802.11i standard was special handling of the pre-shared key (PSK). One of the concerns with the Temporal Key Integrity Protocol (TKIP) and CCMP is the users poor choice of a PSK.

Normally, the PSK would be sent directly to the PRF-384 or PRF-512 function to generate the PTK. In WPA compliant devices, the key is first expanded using the PBKDF2 key derivation function in the following manner:

- The SSID of the network is appended to the PSK.
- The PBKDF2 function calculates the HMAC-SHA1 hash of the input.
- The hash of the output hash is calculated again.
- The process is repeated 4096 times.

This makes it very difficult to derive the PSK that was entered by the user from the PSK actually used in the PRF function to derive the PTK. Mixing in the SSID also adds another layer of security against tables of known keys.

On the other hand, WPA2 was introduced in 2004 and requires all of the revisions introduced in 802.11i document. As of 2006, all new devices have to be WPA2 compliant to bear the Wi-Fi logo.

The WPA and WPA2 standard also define the EAP methods that are allowed for 802.1X authentication. The 802.11i revisions only specified that EAP methods had to mutually authenticate the user and the server. The following EAP methods are allowed by the WPA2 standard:

- EAP-TLS- This was the only protocol allowed by WPA. It was very difficult to implement because it required every client to use a certificate in order to be authenticated. It is often used with the client certificate integrated into a smart card.
- EAP-TTLS- Uses TLS (Transport Layer Security) and certificates to deliver establish communications between the client and the server. Lack of support on most operating systems.
- EAP-TTLS/MSCHAPv2- Extension of EAP-TTLS to use the MSCHAPv2 username and password authentication protocol.
- PEAPv0/EAP-MSCHAPv2- The second most popular EAP method behind EAP-TLS. Native support in most operating systems. Also uses TLS tunnel but does not require a client certificate. Known as the PEAP standard.
- PEAPv1/EAP-GTC- Developed by Cisco as an alternative to PEAPv0. No native operating system support. Rarely used because Cisco promotes it's own nonstandard LEAP protocol.
- EAP-SIM- Created for the GSM mobile industry. Uses a SIM card for authentication. Rarely implemented. No native OS support.
- EAP-AKA- Same as EAP-SIM but with a Universal Mobile Telecommunications Systems Subscriber Identity Module (USIM) instead of a SIM (Subscriber Identity Module).

Overall, WPA2 includes everything in the 802.11i standard plus a few extra added requirements and enhancements. If a client were completely 802.11i compliant, but not WPA2 compliant, it would not be able to connect to a WPA2 network even though it uses all of the underlying protocols. WPA2 has become the *de-facto* standard for wireless equipment. Any wireless hardware that does not meet the standard should be avoided at all costs. It will not likely work with nearly all of the deployed wireless equipment.

3.2.7.3 Wi-Fi Protected Setup

Wi-Fi Protected Setup was introduced by the Wi-Fi alliance in 2007 to simplify the configuration of WPA2 in small networks. It allows users to configure a secure network without knowing about SSIDs, pre-shared keys, or encryption types. The three configuration methods are *Personal Identification Number (PIN)*, *Push Button Configuration*, and *Near Field Communication Token* [RFIP06, GHP01].

The PIN method requires a user to read a PIN number of the wireless adapter and enter it on the interface of the wireless access point. The access point then generates EAPOL (Extensible Authentication Protocol over LAN) keys based on the PIN. The client will generate the same keys and can then establish secure EAPOL messaging with the AP. The AP then deliver the correct keying information to the client and it will re-associate with the network. The push button configuration allows the user to push a button on the access point and then push a real or virtual button for his/her wireless adapter at the same time. The AP and the client then recognize each other and generate pre-shared keys. During this generation time, the network is vulnerable to eavesdropping.

Finally, the near field communications token relies on an out-of-band transmission medium such as an RFID tag. The user simply places their wireless adapter near the access point, and the AP will read the keying information from the RFID tag. It can then use this to establish a secure EAPOL connection and deliver the network keying information to the client. Overall, Wi-Fi protected setup has not been widely deployed. The Wi-Fi alliance shows that 700 devices have the certification. It is good for inexperienced users; however, it does not scale well in larger networks. Mixing devices that do not support it could become a management hassle. There is also no standard built into the operating system network managers for it yet. Thus, each vendor usually has a custom interface for the activation.

3.2.8 Network architecture

In this section we will describe the three possible network configurations available within the IEEE 802.11 framework. The three configurations are known as IBSS, BSS and ESS. We now proceed to detail the basic architecture for each of them.

Independent Basic Service Set (IBSS) An IBSS, also known as an ad hoc network, is a short duration IEEE 802.11 network established from a mesh of mobile stations without any sort of infrastructure for support. On IBSS networks each user has to be within the destination's range for communication to be feasible, unless other nodes are acting as routers.

Basic Service Set (BSS) BSS-based networks, also known as infrastructure networks, are created around an access point that typically has a wired connection with a network of greater dimensions. With this configuration, each mobile node communicates directly with the access point, being the latter responsible for managing the communication of all nodes that registered with it. On this mode of operation, and independently of the distance between mobile stations, all the communications must pass through the access point.

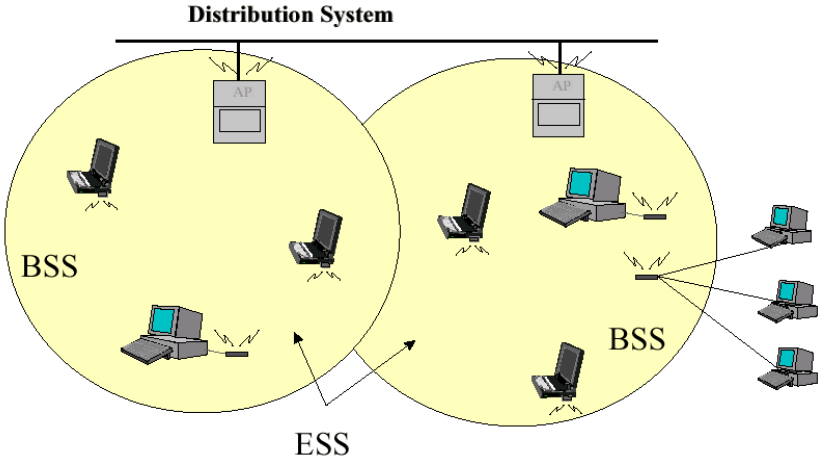

Figure 3.7: Example of a distribution ESS aggregating two BSSs.

Extended Service Set (ESS) ESS networks are characterized by the existence of multiple access points whose coverage area partially overlaps. The distribution services for the access points integrating an ESS include cooperation techniques for the interchange of frames among them. This way we achieve communication among mobile terminals associated with different access points on an ESS in an entirely transparent manner, as if it was all a single large subnet. Access points also allow dynamic association of stations, so that terminals can roam between access points belonging to a single ESS.

Figure 3.7 shows an example where we can distinguish the BSS and ESS concepts, evidencing the coverage of each of the access points and the required overlapping in terms of coverage to make roaming possible.

3.3 Bluetooth

Bluetooth [Blu02] is a standard created by the industry designed to allow simple interconnection between laptops, PDAs, mobile phones and other devices at short distances (10/100 meters maximum, depending on the version of the standard considered). Bluetooth uses fast frequency hops at 1600 hops per second on the 2.4 GHz band at a rate of 1 Mbit/s on version 1.1, 10 Mbit/s on version 1.2, and up to 24 Mbit/s on version 3.0 via 802.11. The transmission power is limited to 1 mW, and access to the medium is done using a *Frequency Hopping Spread Spectrum* (FHSS) technique.

In the sections that follow we offer more information about the Bluetooth technology at different levels.

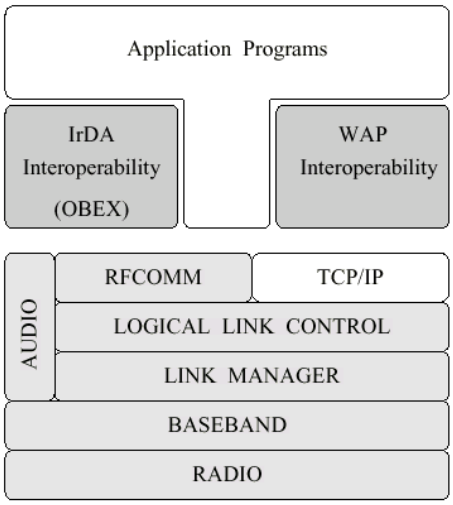

Figure 3.8: Bluetooth protocol stack.

3.3.1 Specification

The Bluetooth specification defines all the necessary requirements to assure the interoperability among devices of this family. The specification is divided in two main parts: radio and protocol definitions, and requirements for interoperability. Figure 3.8 presents the Bluetooth protocol stack defined in the specification.

The Radio layer handles all the issues related to the transmission and reception of modulated signals. The Baseband protocol defines temporization, frames and flow control on the channel. The Link Manager is responsible for managing the connection states, fair sharing among slaves, power management, as well as other management tasks. The logical link control layer multiplexes higher level protocols, being also responsible for the fragmentation and de-fragmentation of large packets, and for device discovery tasks. Audio data are sent directly to the baseband, though audio control is done over the logical link control layer also. Above the LLC layer, both the RFCOMM layer and network level protocols offer different abstractions to communication. The RFCOMM layer offers the possibility of emulating a serial cable by using part of the *ETSI GSM 07.10* standard [GSM97]. Other parts of the Bluetooth specification handle the interoperability towards other protocols or protocol stacks. The use of TCP/IP over Bluetooth requires solving the problems associated with bridging, address resolution, MTU definition, multicast and broadcast.

The second part of the specification defines those aspects related to interoperability. Due to the large variety of possible Bluetooth devices, different sets of requirements are necessary. For instance, the minimum requirements for headphones shall be different from those of a laptop. The purpose of the interoperability section is assuring that any device showing the Bluetooth logo is sure to offer a minimum set of services to the final user.

3.3.2 Architecture

Bluetooth has been developed and designed with the purpose of achieving a low-cost and robust communications system. Its implementation is based on a high performance radio transceiver, though cheap. Bluetooth aims at mobile users that need to establish a connection, or small network, using a computer, a mobile phone or other devices. The required and nominal range for the Bluetooth radio is of 10 meters (with an output power of 0 dBm). For other sort of applications (e.g. a home environment) Bluetooth devices can be enhanced with an external signal amplifier that allows increasing its range (up to 100 meters with an output power of 20 dBm). It is also possible to add hardware to support, for example, four extra voice channels or more. These extensions are totally compatible with the specification, and so their adequateness depends on the target application itself.

When two Bluetooth devices are within range they can start an ad hoc connection called *piconet*. Each *piconet* is formed by up to eight different units where there must always be a device operating as Master; the rest of the *piconet* members act as slaves. The unit establishing the *piconet* is the one responsible for acting as a Master, though this may change so that there is only a single device acting as Master instead of several.

When two or more piconets coexist in a same area we say that a *scatternet* is formed. In a *scatternet* all the units share the same frequency range, but each *piconet* uses different hop sequences and so transmits on different 1 MHz channels. Since all the piconets share a 80 MHz band, there will not be significant channel interference as long as different hop sequences are used.

Bluetooth devices operate on the international band of 2.4 GHz with a theoretical bandwidth of 1 Mbit/s and with a low power consumption, so that it can be used on battery operated devices. With the *scatternet* technology it is possible to achieve an aggregated bandwidth value of 10 Mbit/s or 20 voice channels instead. The structure also enables a range extension at the radio level simply by adding Bluetooth units working as bridges at strategic positions.

A single unit can support a data rate up to 721 kbit/s, or a maximum of 3 voice channels. A mix of data and voice is also possible to support multimedia applications. Concerning voice coding, this process uses a quite robust scheme whose bandwidth is of 64 kbit/s. We should point out that Bluetooth offers a mechanism known as *graceful degradation* when operating on congested radio environments.

3.3.3 Establishment of network connections

When a connection is established for the first time, or when you must add components to a piconet, devices must be identified. These can connect and disconnect from the piconet at any time. The two available options for this process lead to connection times of 0.64 and 1.28 seconds on average. A device does not need to be always connected for a transaction to take place, on average, within less than a second. That way, when the unit is not being used, it can hibernate for most of the time (STANDBY), being a low power oscillator the only active element. This technique offers great power savings. Before connections are established all the

units are on a *standby* state. In this mode, a disconnected unit shall only listen to messages every 1.28 or 2.56 seconds, depending on the chosen option. Each time a unit becomes active, it will listen to one of the 32 hop frequencies defined for that unit. The connection process is started by one of the units, the master. A connection is established with a PAGE message if the destination address is already known, or by an INQUIRY followed by a PAGE message if the address is unknown. On the initial PAGE state the unit performing the PAGE function (master) will send a sequence of 16 identical messages on 16 different hop frequencies defined for the destination unit (slave). This sequence covers half of the sequence of frequencies used by the slave when awakening. It is repeated 128 or 256 times (1.28 or 2.56 seconds) depending on the requirements of the slave unit. If no reply is received after this period, the master transmits an identical sequence of messages on the 16 remaining frequencies. The maximum delay achieved is therefore of 2.56 or 5.12 seconds. This technology offers a clear trade-off between power consumption and latency.

The INQUIRY message is typically used to find public printers, fax machines or similar equipment with an unknown address. This message is quite similar to the PAGE one, but it may require an additional search period to recollect all the answers. If no data must be transmitted, units can be configure to wait (HOLD), situation where only the internal timer is active. When the units come out of this mode the data transmission may start immediately. The HOLD state is normally used to connect several piconets, being also used by units that require sporadic sending of data, and where power savings are an important factor.

3.3.4 Service Discovery Protocol (SDP)

SDP defines how a client application using Bluetooth must act to find the services made available by Bluetooth servers, as well as their characteristics.

The protocol defines how a client can search for a service based on specific attributes, without the client knowing anything about the available services. The SDP provides means for the discovery of new services that become available when the client enters an area where a Bluetooth server is operating.

To activate a new service, a server application must register it with SDP; such information is added to the service record, which consists entirely of a list of service attributes. Based on that information, the client can then proceed to establish a connection with the server, thereby making use of the service advertised. The SDP also provides functionality for detecting when a service is no longer available.

3.3.5 Basic Bluetooth Profiles

To avoid different interpretations of the Bluetooth standard relatively to the interface between application and Bluetooth, the *Bluetooth Special Interest Group (SIG)* defined certain user models and protocol profiles. A profile is merely a selection of messages and procedures obtained from Bluetooth specifications, offering a clear description of the interface used for the specified services. A profile can be described as a certain subset of the protocols defined in the Bluetooth stack,

defining for each protocol, the options that are mandatory for the selected profile, as well as the parameter ranges that can be used with those protocols.

Four generic profiles have been defined, and the main user models are based on them. These four models are the *Generic Access Profile* (GAP), the *Service Discovery Application Profile* (SDAP), the *Serial Port Profile* (SPP) and the *Generic Object Exchange Profile* (GOEP).

Generic Access Profile (GAP) This profile defines how Bluetooth devices must find and establish a connection among them. GAP also assures that Bluetooth devices, independently of the manufacturer and the application they work with, can exchange information to find what sort of applications are supported by both. Conformity to this profile is essential to guarantee interoperability and coexistence.

Service Discovery Application Profile (SDAP) Is is expected that the number of services offered through Bluetooth channels increases at an undetermined and possibly uncontrolled manner. That way the required procedures for users to select one of the available services must be defined. Though most of the services to be created are unknown at the moment, a standard procedure can be created to locate and identify them. The Bluetooth protocol stack offers the *Service Discovery Protocol* (SDP), which allows locating those services available on the vicinity.

Serial Port Profile (SPP) The Serial Port profile defines how to configure virtual serial ports on two devices, and how to connect them through Bluetooth. This profile allows Bluetooth devices to emulate a serial cable with RS232 control signaling, assuring a bandwidth up to 128 kbit/s. This profile depends on the GAP profile.

Generic Object Exchange Profile (GOEP) This profile defines the protocols and procedures that will be used by applications, indicating also some models of use. These models can be, for example, *Synchronization*, *File Transfer* or *Object Push*. This profile depends on the SPP profile.

Other profiles Besides the profiles referred before, there are others that, despite not being so important, are also relevant. Among these we have: the *Cordless Telephony Profile*, that offers communication between phone terminals so that one terminal can use services of another one; the *Intercom Profile*, that allows phone devices to work as "walkie-talkies"; the *Headset Profile*, that allows the remote transmission of sound; the *Dial-up Networking Profile*, that allows a device to use another to connect to a network (e.g. Internet); the *Fax Profile*, that allows a device to offer fax functionality; the *LAN Access Profile*, that allows accessing a LAN using PPP (Point-to-Point Protocol); the *Synchronization Profile*, which defines the interoperability requirements between protocols; the *Exchange Profile*, which allows performing the interchange of objects and doing management tasks; and also the *File Transfer Profile*, that supports the transference of files.

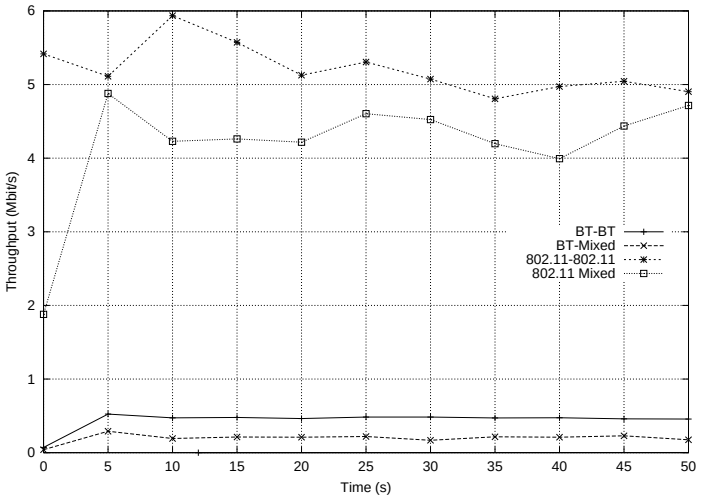

Figure 3.9: Mutual impact of the Bluetooth and IEEE 802.11 technologies in terms of throughput.

3.3.6 Final considerations

Bluetooth was designed as an enabling technology for Wireless Personal Area Networks (chapter 4). Therefore, the main focus was on how to design this technology so as to achieve inter-operation with a number of devices as large as possible. Obviously, this technology allows connecting two or more devices as if a cable was connecting them, which enables networking. Despite such fact, this technology was not designed to support mobile ad hoc networks with fast changing topologies. Two of the main impediments are the relatively high device discovery and device connection times. Another impediment is related to the organization of *scatternets* when there is a large number of nodes involved. As yet another drawback, we can also refer to the relatively low range achieved by most of the devices using this technology. These issues cause Bluetooth to be, currently, not the best technologic option to support mobile ad hoc networks.

Calafate et al. [CRP03] studied the mutual impact of the Bluetooth and IEEE 802.11 technologies in terms of measured TCP throughput. These results are shown on Figure 3.9. That figure shows that, when two IEEE 802.11-enabled devices are operating close-by, without interference from Bluetooth devices, the throughput achieved is of about 5 Mbit/s. In a similar fashion, when two Bluetooth-enabled devices are operating close-by, without interference from IEEE 802.11 devices, the throughput achieved is of about 0.5 Mbit/s. However, when both pairs of devices are close-by and transmitting at the same time, there is a considerable mutual impact between them, being that the throughput between the two IEEE 802.11-enabled devices drops to about 4.2 Mbit/s, and that the throughput between the two Bluetooth-enabled devices drops to about 0.25 Mbit/s. These substantial performance drops experienced, especially in the Blue-

tooth case, lead to the conclusion that the segregation of both technologies is perhaps the most appropriate decision to take when performance is at premium.

3.4 Conclusions

In this chapter we have offered an overview of the most widely deployed wireless technologies currently available for free personal use: IEEE 802.11 and Bluetooth. We presented some architectural details of these technologies, including information about their physical layers and their basic architecture. By analyzing their characteristics and performance we reach the conclusion that, currently, the IEEE 802.11 technology is the most adequate choice for wireless mobile ad hoc networks, which is the main area we focus on this thesis. On the other hand, Bluetooth is more adequate for personal area networks. Such networks will be the topic of the next chapter.

Chapter 4

Short Radio Range Wireless Networks

In this chapter we will review the state-of-the-art of Short Radio Range Wireless Networks, including Personal Area Networks (PANs), Mobile Ad Hoc Networks (MANETs), and Spontaneous Networks. This kind of networks along with the wireless technologies we have seen in chapter 3, are a strong basis for implementing the concepts of ubiquitous and pervasive applications.

First, we address the main characteristics of PANs, showing the existing working groups within the IEEE 802.15 standard to attend different requirements and scopes. Regarding MANETs, we start by explaining the ad hoc network concept, referring the main characteristics of these networks and their fields of application. We then focus on some of the most important characteristics that should be solved to successfully deploy an integrated architecture for MANET environments. In particular, we review the most important routing protocols developed for MANETs, evidencing their characteristics in terms of responsiveness to mobility and of route maintenance. Furthermore, since configuring the different MANET parameters is a complex task, we review some available autonconfiguration solutions. Finally, we will briefly present the concept of spontaneous networks, a special kind of network that combines characteristics of both PANs and MANETs.

4.1 Personal Area Networks

The concept of Personal Area Network was established by Thomas Zimmerman and other researchers at MIT's Media Lab. Zimmerman explained in a research paper [Zim96] why this concept can be useful :

"As electronic devices become smaller, lower in power requirements, and less expensive, we have begun to adorn our bodies with personal information and communication appliances. Such devices include cellular phones, personal digital assistants (PDAs), pocket video games, and pagers. Currently there is no method for these devices to share data. Networking these devices can reduce functional I/O redundancies and allow new conveniences and services".

Figure 4.1: Possible devices in a Personal Area Network.

A personal area network (PAN) consists on interconnecting several information technology devices within the range of an individual person, typically 10 meters. As an example, if we consider a person traveling with a laptop, a smartphone, and a portable printer, all these devices could be wirelessly interconnected. In general, a PAN can interconnect at any time the ordinary mobile devices that people carry with them. Moreover, a personal area network could also be connected to other PANs, or even to the Internet. Figure 4.1 illustrates the concept of PAN, and shows some of the different devices that this kind of network can support.

A personal area network can also be defined as the technology which makes wearable computer possible. In this paradigm, electronic devices are considered as an extension of the human body, thus being able to provide intelligent assistant, augmented reality, etc.

The term wireless personal area network (WPAN) can be considered a synonym of PAN since almost any personal area network needs a wireless technology to operate (e.g. Bluetooth, Wi-Fi, or ZigBee). On the other hand, the conceptual difference between PANs and wireless mobile ad hoc networks (MANETs) is that, as we said, PANs are usually centered around a person, while MANETs, as we will see in chapter 4.2, can be established between multiple users.

Technologies for WPANs are constantly evolving. As an example, the initially proposed operating frequencies were around the 2.4 GHz band, but with the emergence of new technologies, such as ultra-wide band (UWB), new frequencies are being added. Anyway, each device in the WPAN should be able to connect with any other device in the same WPAN (whenever these devices are within the same radio coverage). However, an important property for WPANs is that each device has the ability of locking out other devices selectively, thus preventing needless interference or unauthorized access to private information.

The IEEE 802.15 Working Group is the organization which defines the Wireless Personal Area Network (WPAN) standards. It is divided into a set of task groups.

4.1.1 Task Groups

There are seven main task groups for WPAN technology, each one with specific characteristics and interests that generate standards which meet specific communication needs. Next, a brief description of each one is shown:

802.15.1 This task group has conducted a standard based on the specifications of the Bluetooth consortium. This working group published the standard IEEE 802.15.1 on June 14, 2002. This version was updated in 2005.

802.15.2 This task group studies the possible problems of coexistence between WLANs and WPAN, as well as the involved devices. The group developed: (i) a coexistence model to quantify the mutual interference; and (ii) a set of coexistence mechanisms to facilitate coexistence. This standard was approved in 2003 and the group went into "hibernation".

802.15.3 This task group developed a standard that defines the physical and MAC levels for high speed WPANs (11-55 Mbps). In addition to offering high transmission speed, the standard was designed to address low-cost solutions for multimedia applications, and low power consumption. The standard was initially issued in 2003, and extended in 2005, 2006 and 2009.

802.15.4 This task group researches and develops solutions that require low speed data transmission, thus extending battery life to months or even years, as well as a relatively low complexity. Potential applications are sensors, interactive toys, smart badges, remote controls, and home automation. ZigBee protocols [zig05] are based on the specification produced by this working group. The first revision was approved in May 2003 and updated in 2006.

802.15.5 This task group focuses on determining the needed mechanisms that must be present in the PHY and MAC layers of WPANs to enable mesh networking. A mesh network employs one of the two following connection arrangements: (i) full mesh topology, where each node is connected directly to all others; or (ii) partial mesh topology, where only some nodes are connected to all others. The standard was approved on March 2009.

802.15.6 This task group focuses on BAN (Body Area Network) technologies. Its main contribution is a low-power and low-frequency short-range wireless standard optimized for devices and operation on, in, or around the human body (but not limited to humans). Applications may include medical, consumer electronics, and personal entertainment. A draft was approved in December 2011.

802.15.7 This task group was chartered to write a PHY and MAC standard for Visible Light Communications (VLC). VLC is a new communication technology which uses "Visible Light" (band between *400-790 THz*). The working group inaugural meeting was held in January 2009.

4.2 Mobile Ad Hoc Networks

The history of wireless networks dates from the late 70s, and interest has been growing ever since. Towards the end of the 90s decade, interest reached a peak mainly due to the fast growth of the Internet. More recently, developments have been centered around infrastructure-less wireless networks, more commonly known as ad hoc networks. The term ad hoc, despite sometimes having negative overtones and being equated with improvised or not organized, is used in this context to express a higher level of flexibility. All nodes within an ad hoc network provide a peer-level multi-hopping routing service to allow out-of-range nodes to be connected. Unlike a wired network, nodes in an ad hoc network can move, thus giving rise to frequent topology changes.

Such a network may operate in a stand-alone fashion, or be connected to the larger Internet. An ad hoc architecture has many benefits, such as self-reconfiguration and adaptability to highly variable characteristics including power and transmission conditions, traffic distribution variations, and load balancing. However, such benefits come with many challenges. New algorithms, protocols, and middleware have to be designed and developed to create a truly flexible and decentralized network. Protocols should be adaptable, that is, they should learn and anticipate the behaviour of the network using parameters such as level of congestion, error rate, and topology change rate. Resources and services have to be located and used automatically, without the need for manual configuration. Access and authentication issues should also be considered to ensure security and user privacy.

Therefore, we can see mobile ad-hoc networks (MANETs) as autonomous systems composed of independent mobile terminals which communicate among themselves using any sort of wireless technology. The terminals are free to move around and organize themselves to conform an IP-based network. Each node operates not only as an end-system, but also cooperates on routing and packet forwarding tasks. If we see it from the IP layer perspective, a MANET is a Layer-3 multi-hop network implemented over a collection of links. Therefore, each node pertaining to the network is, in principle, acting as a Layer-3 router in order to provide connectivity to other nodes. Each node maintains host routes to the rest of nodes within the network, in addition to network routes to destinations outside the MANET, if any. Figure 4.2 illustrates the multi-hop routing concept in MANETs.

One of the main advantages of MANETs is that they do not require any fixed infrastructure or a centralized administration, being an attractive networking option for connecting mobile devices quickly and spontaneously. Ad-hoc networking can be applied anywhere where there is little or no communication infrastructure, or the existing infrastructure is expensive or inconvenient to use. Ad-hoc networking also allows devices to maintain dynamic connections to the network, as well as easily adding and removing devices to and from the network.

MANETs applications are diverse, ranging from ad hoc video/audio conferencing through wireless sensor networks, campus ad hoc learning tools, networked robots, emergency rescue communication networks, defense tactical communications, context-aware environments, and so on. In general, mobile ad hoc networks can be used on all those situations characterized by lack of fixed infrastructure,

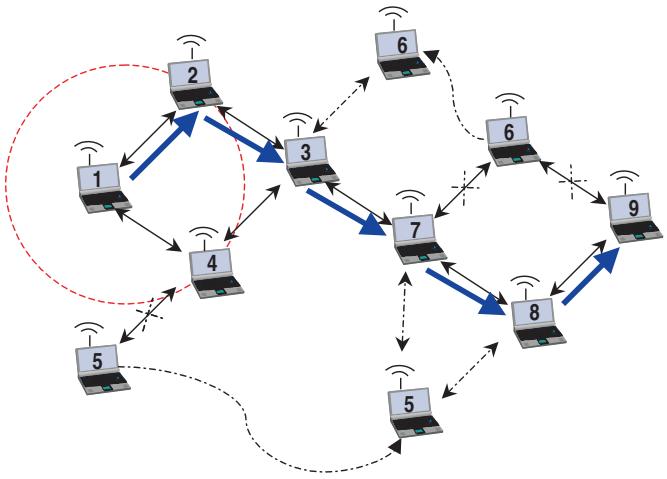

Figure 4.2: Example of multi-hop routing in MANETs.

peer-to-peer communication and mobility support. However, due to their flexibility and adaptation capabilities, MANETs are also characterized by being complex to deploy and maintain in an automatic manner. One of the main problems when deploying an ad-hoc network is the configuration of nodes. In the next chapters, we are going to discuss the most important issues of MANETs, including routing and autoconfiguration tasks.

4.2.1 Classification of routing protocols

A routing protocol is required when a packet must go through several hops to reach its destination. It is responsible for finding a route for the packet and making sure it is forwarded through the appropiate path. Routing protocols based on algorithms such as *distance vector* (e.g. RIP [Mal98]) or *link-state* (e.g. OSPF [Moy98]) were available before solutions were sought in the field of wireless ad hoc networks. These routing protocols generate periodic control messages, a procedure that is not adequate for a large network with long routes since it would result in a large number of control messages. Moreover, the period between messages would have to be reduced in the presence of mobility. This effect is critical for mobile nodes where CPU use, as well as radio transmissions and receptions, would cause batteries to be quickly depleted. Also, all the conventional routing protocols assume bidirectional routes with a similar quality, something that is not always true on some kinds of networks (e.g. wireless ad hoc networks). Routing protocols can be classified according to three different criteria:

- Centralized or distributed: when a routing protocol is centralized, all the decisions take place at a central node. However, with a distributed routing protocol, all the nodes share the routing decisions.
- Adaptive or static: an adaptive routing protocol can change its behaviour

according to the network state, which can be the congestion on a certain connection or other possible factors, contrarily to a static one.

- Reactive, proactive or hybrid: a reactive routing protocol must act to find routes when necessary, while a proactive routing protocol finds routes before these are required. Reactive routing protocols are also known as *on-demand* routing protocols. Since these are executed on-demand, the control packets' overhead is considerably reduced. Proactive methods maintain routing tables, being these periodically updated. Concerning hybrid methods, these use a combination of both reactive and proactive techniques to achieve a more balanced solution.

4.2.1.1 Basic routing techniques

Independently of how a routing protocol is classified according to those criteria, the routing techniques used can be divided into three families: *distance vector*, *link state* and *source routing*. We now detail the basic principles of each one.

Distance Vector This technique maintains a table for the communication taking place and employs diffusion (not flooding) for information exchange between neighbors. All the nodes must calculate the shortest path towards the destination using the routing information of their neighbors.

Link State The protocols based on this technique maintain a routing table with the full topology. The topology is built by finding the shortest path in terms of link cost, cost that is periodically exchanged among all the nodes through a flooding technique. Each node updates its routing table by using information gathered about link costs. This technique is prone to cause loops on networks with a fast changing topology.

Source Routing Technique where all the data packets have the routing information on their headers. The route decision is made on the source node. This technique avoids loops entirely, though the protocol overhead is quite significant. This technique can be inefficient for fast moving topologies due to route invalidation along the path of a packet.

4.2.2 Routing in ad hoc networks

An ideal routing protocol for ad hoc networks must have certain properties that make it different from the rest. To begin with, it must be distributed to increase reliability: when all the nodes are mobile, it makes no sense to have a centralized routing protocol. Each node must have enough capabilities to make routing-related decisions with the aid of the rest of the nodes. Also, a routing protocol should assume that the links detected are unidirectional connections. On a wireless channel, an unidirectional connection may be created due to physical factors, so that bidirectional communication may result impossible. Therefore, a routing protocol should be designed to take into account this possibility.

It is also important that an ad hoc routing protocol takes into account issues such as power consumption and security. Obviously, mobile nodes depend on batteries. This means that a protocol that minimizes the total power consumption of network nodes would be ideal. Concerning security, you must take into account that the wireless medium is very vulnerable. At the physical level, DoS attacks can be avoided by using frequency hopping or code-based *Spread Spectrum* techniques. At the routing level, though, both the authentication of neighbors and the encryption of data are required.

4.2.2.1 Routing protocol families for ad hoc networks

In section 4.2 we introduced the most important characteristics of an ad hoc network. Concerning the routing protocols used on these networks they should be, according to the classification of section 4.2.1, both distributed and adaptive.

Relatively to the third category (reactive/proactive/hybrid), there is no consensus over which is the most adequate strategy. Below we present different proposals for each of these protocol families, and we also include other non-cataloged proposals.

Proactive routing protocols

The concept of proactive routing means that all the nodes (routers) periodically interchange routing information (or upon detecting topology changes) with the aim of maintaining a consistent, updated and complete view of the network. Each node uses the exchanged information to calculate the costs towards all possible destinations. That way, if a destination is found, there will always be a route available; this avoids delays associated with finding routes on-demand.

Proactive techniques typically use algorithms such as *distance vector* or *link-state*. Both techniques require routers to periodically broadcast information and, based on that information, to calculate the shortest path towards the rest of the nodes.

The main advantage of proactive routing schemes is that there is no initial delay when a route is required. On the other hand, these are usually related to a greater overhead and a larger convergence time than for reactive routing techniques, especially when mobility is high. To increase the performance in ad hoc networks both *link-state* and *distance vector* algorithms were modified. Examples of routing protocols using *distance vector* techniques are the *Destination-Sequenced Distance Vector* (DSDV) [CP94] and the *Wireless Routing Protocol* (WRP) [MGLA96]. Examples of *link-state* based protocols are the *Open Shortest Path First* (OSPF) [Moy98], the *Optimized Link State Routing* (OLSR) [TP03], the *Topology Broadcast Reverse Path Forwarding* (TBRPF) [BR99], the *Source Tree Adaptive Routing* (STAR) [GLAS99], the *Global State Routing* (GSR) [CQS98], the *Fisheye State Routing* (FSR) [PGC00] and the *Landmark Routing Protocol* (LANMAR) [PGH00].

Reactive routing protocols

Reactive routing does not depend, in general, of periodic exchange of routing information or route calculation. Therefore, when a route is required, the node must start a route discovery process. This means that it must disseminate the route request throughout the network and wait for an answer before it can proceed to send packets to the destination. The route is maintained until the destination is unreachable or until the route is no longer necessary. By following this strategy, reactive routing protocols keep to a minimum the resource consumption by avoiding the maintenance of unused routes. On the other hand, the route discovery process causes a significant startup delay and causes a considerable waste of resources. If the network is wide enough, the overhead will be similar or superior to that achieved with proactive routing protocols.

The most common routing algorithms found among reactive routing protocols are *distance vector* and *source routing*. Example of reactive routing protocols are the *Ad-hoc On-demand Distance Vector* (AODV) [CES03], the *Dynamic Source Routing* (DSR) [DDY04], the *Dynamic MANET On-demand* (DYMO) [IC08], the *Associativity Based Routing* (ABR) [Toh97], the *Signal Stability based Adaptive routing* (SSA) [DRWT96], the *Temporally Ordered Routing Algorithm* (TORA) [VS00] and the *Relative Distance Micro-discovery Ad-hoc Routing* (RDMAR) [AT99].

Other strategies

There are other strategies proposed for the design of routing protocols. There are, for instance, hybrid solutions such as the *Zone Routing Protocol* (ZRP) [ZM99] which uses both reactive and proactive concepts: each node maintains a zone (on a radius of 2 hops) where it employs proactive routing; to access nodes outsize that zone it uses reactive routing. The overhead is limited since the maintenance of proactive routes is only performed with neighbor nodes, and the reactive search for routes is limited to the communication within the selected subset of nodes.

It is common that the routing protocol has a flat architecture, though there are some protocols based on *clustering* and hierarchical architectures, such as the *Clusterhead Gateway Switch Routing* (CGSR) [LCWG97], the *Distributed Mobility-Adaptive Clustering* (DMAC) [Bas99] and the *Cluster-based Energy Saving Algorithm* (CERA) [JDP03]. The advantage of these solutions is mainly the discovery of more robust routes with fewer control messages, though periodic messages may be required for the maintenance of clusters. A clear disadvantage is the centralization of routes through cluster leaders, which provokes congestion and also single points of failure which can cause large recovery periods.

The procedure of route discovery for reactive routing protocols, such as AODV and DSR, is based on a variant of the flooding technique, being typically quite resource consuming. The LAR protocol [KV98] tries to avoid this problem by using GPS information so that only those nodes on a certain geographic area between source and destination must retransmit route requests.

Reducing the control overhead achieves, in general, improved scalability and reduced power consumption. There are several techniques that intend to improve

the power consumption. PAR [SWR98] is a solution that takes into account the battery lifetime, selecting those routes that minimize the energy consumption of the system. Another solution is to reduce the transmission power, which also reduces interference and improves spacial reuse [San01]. Normally, this kind of protocols work in cooperation with the MAC layer protocol so that there is a per-packet energy assessment, as with the PARO [GCNB01] protocol. This technique also suffers from some problems, such as generating unidirectional routes and decreasing the transmission power, which increases the bit error rate and reduces the bit-rate under IEEE 802.11-based network environments.

4.2.2.2 The Optimized Link-State Routing (OLSR)

The Optimized Link State Routing protocol [TP03] is a proactive routing protocol specifically designed for MANETs. It is based on the definition and use of dedicated nodes, called multipoint relays (MPRs). MPRs are selected nodes which are responsible for forwarding broadcast packets during the flooding process. This technique allows to reduce the packet overhead compared to a pure flooding mechanism where every node retransmits the packet when it receives the first copy of it. Contrarily to the classic link-state algorithm, partial link-state information is distributed throughout the network. This information is then used by the OLSR protocol for route calculation. The protocol is particularly suitable for large and dense networks as the technique of MPRs works well in this context.

Basic principles

The OLSR protocol inherits its stability from link-state algorithms, offering the advantage that available routes can be used immediately.

Pure link-state algorithms declare and propagate the list of neighbors for each node throughout the network. OLSR tries to improve this solution by using different techniques. To start with, it reduces the size of control packets since it does not declare all of its neighbors, but only a subset of these referred as Multipoint Relay Selectors. A node's Multipoint Relay is in charge of retransmitting its broadcast messages. The use of MPRs serves the purpose of minimizing the amount of retransmissions upon a flooding or broadcast event.

Besides periodic control messages, the protocol does not generate additional control traffic in response to failures or association with new nodes. The protocol maintains routes towards all network destinations, being useful in those situations where a great number of MANET nodes is communicating, especially when source/destination pairs are changing frequently. This protocol is more adequate for large and dense networks, where the optimizations achieved by introducing Multipoint Relays offer important benefits.

The protocol is designed to operate in a distributed fashion, and so it does not depend on a central entity. Moreover, it does not require reliable transmission of its control messages: each node sends periodic control messages, being tolerant to sporadic losses of control packets. Packet reordering, a frequent phenomena in ad hoc networks, will not cause OLSR to misbehave since each message carries a different sequence number.

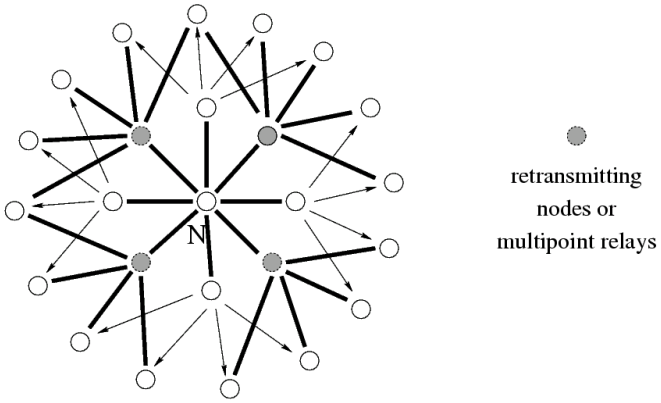

Figure 4.3: Illustration of the multipoint relay concept for node N.

The OLSR protocol uses per-node packet forwarding, which means that each node uses its most recent information to route a packet. That way, when a node is moving, its information is successfully transmitted as long as neighbor nodes can keep track of the mobile node. The ability to follow a node can be adjusted by setting the interval between consecutive control messages.

Multipoint Relays

The Multipoint Relay concept consists in trying to minimize the flooding caused by broadcast traffic by eliminating duplicated transmissions on a same region. Each network node selects a subset of those nodes in its vicinity to retransmit its packets. Nodes belonging to this subset are a node's Multipoint Relays (MPRs). The neighbors not part of the MPR subset of a certain node N will still receive packets from it, but will not re-transmit them again. That way, each node maintains a table with the nodes which have selected it as their MPR.

Each node selects its own set of MPRs among their neighbors with a criteria that consists of assuring that all those nodes two hops away from it can be reached with a minimal number of MPRs. Figure 4.3 illustrates this concept.

OLSR trusts on the MPR node selection to calculate routes towards all the destinations having these as intermediate stations. This solution requires each node to periodically broadcast the list of neighbor nodes chosen as its MPRs. When receiving this information, each neighbor node updates the routes towards all known stations.

MPR nodes are selected among those neighbor nodes where bidirectional communication is feasible, which avoids attempting to transmit packets through unidirectional links.

Neighbor detection

Each node must detect those neighbor nodes towards which bidirectional communication exists. To achieve this purpose, a node periodically broadcasts HELLO messages containing information about its neighbors and the state of the channel towards them. These messages are received by all neighbor nodes but are not retransmitted.

These HELLO messages allow a node to discover those other nodes that are two hops away. It is based on this information that a node selects its MPR node set. The MPR set is indicated on HELLO messages through the MPR identifier; such information is used by other nodes to construct their MPR Selector table.

For adequate operation each node will then maintain a table with a list of all the nodes it can see either directly or indirectly. Links to one hop neighbors are tagged as either unidirectional, bidirectional or MPR. Each table entry has a both a sequence number and a timeout value associated, so that old entries can be removed.

Multipoint Relay selection

Each network node independently chooses its MPR set. This set is calculated among the direct neighbors so as to reach all the nodes two hops away. Maintaining a list of the two-hop neighbors requires analyzing HELLO messages and filtering all the unidirectional links. The higher the degree of optimality of the MPR selection algorithm, the more benefits will it bring to all nodes.

The MPR set is only altered when a change is detected in terms of one-hop or two-hop neighbors (bidirectional connections only).

MPR information broadcasting

Each node must broadcast topology control messages (TC) in order for all nodes to maintain their database updated. These messages are broadcasted throughout the network using a technique similar to the one used for traditional link-state routing protocols, with the only difference that it employs MPRs to improve scalability.

A TC message is sent periodically to each network node to declare its MPR selector set. This means that the message must contain a list with those direct neighbors that have selected it as their MPR. This list has always a sequence number associated.

The list of addresses on each TC message can be partial, but it must be completed before each refresh period ends. These messages will allow each node to maintain its own table with the network topology. If a node has not been selected as any other node's MPR, it does not send TC messages, thereby saving power and bandwidth.

The interval between the transmission of two TC messages depends on whether there have been changes on a node's MPR selector set. If so, the next TC message can be transmitted before the scheduled time, though respecting the minimum inter-message time.

Calculation of the routing table

Each node maintains a routing table with information on how to access other network terminals. When nodes receive a TC message, they store sets of two addresses indicating the last hop before reaching a certain destination node, as well as the destination node itself. By combining the information in these address pairs the node is able to find what is the next hop towards a certain destination node. Minimum distance criteria should be followed to restrict the search options.

Routing table entries are composed of destination, next hop and estimated distance to destination. On this table we only register those entries for which the route towards the destination is known. This means that the routing table must be constantly updated according to the topology changes detected.

In a real implementation, the OLSR daemon must update the kernel's forwarding table according to the routing table it maintains, so that packets are sent through valid routes.

4.2.2.3 Ad hoc On-Demand Distance Vector (AODV)

The Ad-hoc On-demand Distance Vector (AODV) is a reactive routing protocol that, as the name indicates, uses distance-vector algorithms to create and maintain routes.

An important advantage of AODV is that it generates no extra traffic for communication along existing routes. Also, distance vector routing is simple and does not require much memory or calculation. However, AODV requires more time to establish a connection and, besides, the initial process required to establish a route introduces more routing overhead than proactive approaches.

Route discovery

AODV only finds routes to a destination on demand, which means that no routing packet flows through the network until a connection is needed. When a route is required the source node starts a route discovery process by broadcasting a route request packet (RREQ) that is re-broadcasted by intermediate nodes until the destination is reached. During this process, each intermediate node makes sure that it re-broadcasts a RREQ packet only once for the same route discovery action; this is possible because each request for a route has a sequence number associated. Intermediate nodes also update an internal table where they keep temporary route information about how to reach the source node. Once RREQ packets reach the destination, a route reply (RREP) message is sent back to the source, and it can then determine which of the available routes offers the least number of hops.

Management of the routing table

To avoid keeping invalid routes, the routes created during each route discovery process are assigned a timer. Routes that are not used are invalidated when this timer is triggered.

Routes towards neighbor nodes are also maintained. A neighbor node is considered active if it originates or forwards at least one packet within the most recent

active_timeout period. This information is used to notify neighbors when a link of a path being used breaks, thereby stopping traffic if the path to the destination becomes unavailable.

The contents of AODV's routing table are the following: destination, next hop, number of hops, sequence number for the destination, active neighbors and expiration time. Notice that it is very important for all the routes in the routing table to be tagged with a destination sequence number. This technique will allow assuring that no routing loops can be formed, even in the presence of out-of-order packet delivery or high degrees of node mobility.

Maintenance of routes

AODV's route maintenance is required due to node mobility. To detect that a link being used breaks, the source has two options. One of them consists in sending HELLO messages frequently. These messages allow detecting when a link is no longer available, as described below. A much better option is using information from the link layer to detect link failures. These link failures occur every time that an attempt to send a packet through that link fails.

When a link failure is detected, the node detecting it will send a RERR message back to the source(s) using that link. It must state the problem encountered and use an updated sequence number. This process ends when all active sources receive the message. Source nodes still requiring the connection can start a new route request process to find new routes to active destinations.

Neighbor management

Detecting the availability of new neighbors and, more important, detecting that a neighbor has become unavailable, is an important issue in mobile ad hoc networks. AODV allows nodes to detect their neighbors in two different ways. When a neighbor receives a broadcast packet from one of his neighbors it updates its internal tables to include that neighbor. Sometimes a node does not send any packets downstream, which could case downstream nodes to consider it unavailable. In those cases such nodes broadcast an unsolicited RREP packet within a *hello_interval* time so that downstream nodes become aware of their liveliness. This packet has a TTL of 1, so it is not re-broadcasted. When a node fails to listen to up to *allowed_hello_loss* consecutive packets from a node participating on an active path it considers the link with the upstream node to be lost and, therefore, generates a route error message to be sent to the source of that stream.

4.2.2.4 Dynamic Source Routing (DSR)

The Dynamic Source Routing (DSR) protocol [JM96, DDY04] is a high performance reactive routing protocol for MANETs. Its route discovery process is on-demand, which means that routes are only built when needed; route maintenance also depends on the existence of traffic. Therefore, when there is no data traffic on the network, the routing traffic is effectively reduced to zero. In some ways it is quite similar to the AODV protocol described before; the main difference is that

DSR uses source routing, which means that the source determines the entire route towards the destination. So, packets are sent to their destinations with the entire route in their IP header. This method, despite provoking a small increase in terms of overhead, has other advantages such as avoiding routing loops in a simple and efficient manner.

One of the main differences between DSR and other routing protocols for MANET is its intensive use of caching. Each node participating in the MANET maintains a route cache where it saves all the routes it has learned. So, when a packet must be sent to a particular destination, nodes first check if a route is available on their cache. When there is no route for a packet, a route discovery process is started. During that time DSR can be configured to hold on a buffer those packets waiting for a route, or it can choose to discard them, relying on higher-layer protocol software to recover from that loss. DSR typically allows a data packet to be queued while waiting for a route for up to 30 seconds.

Route discovery

When initiating a route discovery the source broadcasts a route request (RREQ) packet, which is then successively broadcasted by other nodes until the destination is reached. Each node forwards only the first packet it receives for a certain route request ID originated at that same source. This aims at reducing the broadcast storm generated as much as possible. When re-broadcasting RREQ packets, nodes add themselves to the DSR header as elements of the route. This allows other nodes (including the destination) to also learn about the path. Therefore, when the destination receives a RREQ packet, it constructs the entire route for the route reply packet merely by calculating the opposite path through route reversal.

The destination of a route request can send a reply only for the first RREQ packet arriving, or for all of them. Replying only to the first one allows reducing the routing overhead, but when that route is invalidated a new route request cycle has to be initiated. If it replies to all of the route requests, the source is able to cache the different routes found and, when the first route is lost, it can try using the remaining routes successively. If no route reply arrives to the source before the established timeout, a new route discovery procedure is launched.

Route maintenance

DSR's route maintenance procedure consists of acting when link breaks occur. DSR uses the information from lower layers in order to detect broken links. This method allows it to react very quickly to link failures, but only for unicast packets since only these are acknowledged. The node which detects the broken link sends a route error message (RERR) back to the source indicating the link that broke. The source, as well as the rest of the nodes through which the RERR packet passes by, removes from its cache all routes including that broken link.

Another option in the case of detecting a link break event is to save the route error packet locally in a buffer, perform a route discovery for the original sender, and then send the route error packet along with the new route as soon as it receives the respective route reply.

4.2.2.5 Dynamic MANET On-demand (DYMO)

The Dynamic MANET On-demand (DYMO) routing protocol enables reactive, multihop routing. It is adapted to network topology changes and determines unicast routes between nodes within the network. DYMO is the evolution of previous MANET reactive protocols such as AODV and DSR, sharing many of its benefits; however, it is slightly easier to implement.

The main operations of DYMO are route discovery and route maintenance. The protocol makes use of sequence numbers to guarantee loop freedom. These sequence numbers enable routers to determine the order of route discovery messages, thus avoiding to use outdated routing information.

Route discovery

When a source router needs to forward a packet, it disseminates a route request (RREQ) throughout the network to find a route to the destination router. During the dissemination process, each intermediate router records a route to the source router. When the destination router receives the RREQ, it responds with a route reply (RREP) message sent toward the source router. Each intermediate DYMO router that receives the RREP creates a route to the destination router, and then the RREP is forwarded towards the source router using unicast transmissions. When the source router receives the RREP, routes between both source and destination routers have been established in both directions.

If a route is not created within a RREQ_ *WAIT_ TIME*, the source router may again try to discover a route by sending a new RREQ. Notice that, in order to reduce congestion across the network, repeated attempts of route discovery for a particular destination node should adopt an exponential backoff. Additionally, if a route discovery is attempted RREQ_ *TRIES* times without receiving a route to the destination, all data packets to that destination are dropped from the buffer and an ICMP *Destination_ unreachable* message is generated.

Route maintenance

Route maintenance is basically composed of two operations: (i) To preserve the current routes, DYMO routers extend route lifetimes upon successfully forwarding a packet; (ii) To react to changes in the network topology, DYMO routers monitor those links over which traffic is flowing.

Notice that, if a data packet is received for forwarding and a route for the destination is not known or is broken, then the source router is notified. Moreover, a route error message (RERR) is sent toward the source router indicating that the current route to the destination is invalid or missing. When the source router receives the RERR, it deletes the route, but if the source router later receives a packet for forwarding to the same destination, it will perform a new route discovery for that destination.

4.2.2.6 Multipath routing protocols

In the past there have been several approaches in the literature related to the discovery and use of multiple routes in MANETs.

In the work of Wang et al. [LYM+01] a probing technique is used in order to assess the quality of available routes so that the traffic is forwarded based on the delay of each route. Their objective was to achieve load distribution as well as improved throughput, end-to-end delay and queue utilization.

In [MD01] the AODV protocol is extended in order to provide multi-path capabilities, though no new route discovery mechanism was proposed. Both node disjoint and link disjoint approaches are presented. In their work there is no traffic splitting. Also, neither of these two works referred before propose enhancements to the route discovery technique itself.

Nasipuri et al. [ARS01] propose a strategy for quick route recovery through packet re-direction on intermediate nodes in order to reduce the frequency of query floods. Their solution aims at reducing the number of *lost route* messages, as well as performing fewer route discoveries. However, the source is unaware of any extra routes, which means that their solution does not aid in the task of splitting traffic through disjoint routes.

Wu [Wu02] proposes a more selective route discovery procedure for DSR to increase the degree of disjointness of routes found without introducing much extra overhead. However, it allows the source to find a maximum of only two paths (node disjoint paths) per destination and required two consecutive route discovery processes to take place.

In the work of Lee and Gerla [LG01] the traffic is evenly split among the two first routes found in order to achieve load distribution; they analyze the options of starting a new route discovery process when one of the routes is lost or only when both are lost. The authors find that DSR's standard route discovery mechanism not only returns a few routes, but also that these routes are mainly overlapped (not disjoint). To solve this problem they enhance the route discovery mechanism of DSR to find more node disjoint paths.

4.2.3 Autoconfiguration in MANETs

MANETs can be useful in a wide variety of scenarios, but deploying and setting them up is a difficult task even for experienced users since some degree of knowledge and expertise in IP networks and wireless technologies is required. This fact seriously reduce the use of MANETs. Therefore, it would be very interesting to automate as much as possible all the tasks affecting the configuration and startup of MANETs to make their use simpler and more widely available.

The Zero Configuration Networking (zeroconf) Working Group [zer99] has been looking for solutions to enable networking (for wired networks) in the absence of any configuration and administration. Zero configuration networking ([SC05]) is required for such environments where administration is impractical or pointless (e.g. home, small offices, ad hoc meetings, or any kind of environment where a small group of devices need to spontaneously share or exchange information). On the contrary, is not appropriate for networks accessed by a wide number of users or

with a high degree of security and control. Zero configuration is expected to make seamless networking a reality. The Zeroconf working group is focused mainly on four areas:

- Interface Configuration (IP address, router, etc), without a DHCP server
- Name-to-Address Translation, without a DNS server
- Service Discovery, without a directory service
- Automatic Multicast Addresses Allocation, without a multicast server

An important challenge is to consider such needs for MANET environments. In the next subsections we present different solutions for these four areas defined by the Zeroconf working group. As we will see, most of the proposals are evaluated only via simulation, and no real implementation is available. These limitations motivate us to work on the EasyMANET platform, which will be elaborated in chapter 7. Finally, notice that security is an important issue in Zeroconf environments, especially if we consider wireless networks.

4.2.3.1 Interface configuration

In order to achieve unicast communication in MANET environments, each node must have a unique IP address. Traditionally, the solution to this problem requires all users to adjust the IP configuration of their wireless interfaces to a same subnet, avoiding addresses that have been picked up by other users. This task is tedious and time consuming since it must be repeated on every single terminal participating in the MANET, and hence it is better to perform it automatically. The Internet Engineering Task Force's (IETF's) Ad Hoc Network Autoconfiguration (autoconf) Working Group [IET03] is currently working on a standardized addressing model for ad hoc networks, and how nodes in these networks can configure their addresses automatically. In general, address autoconfiguration mechanisms can be classified into stateful, stateless, and hybrid approaches.

Stateful protocols

Stateful mechanisms assign *a priori* unique addresses to new nodes by maintaining a common set of addresses. Traditional IP-based networks use a central entity to maintain this set of addresses. The maintenance of a common set of addresses at all nodes in the network may be a complex and bandwidth consuming process.

Boleng [Bol02] uses an approach based on a variable-length network layer addressing scheme. This protocol significantly reduces the overhead produced by sending a lot of addressing information. However, it cannot be used by applications and protocols which use IP addresses without translating MANET addresses into IP addresses. In [MR02], authors propose a stateful approach for IP assignment in MANETs where nodes maintain a disjoint distributed set of addresses, but global synchronization is still required to guarantee disjointness. Moreover, issues like abrupt departure of nodes from the system are also taken into account.

Finally, Sheu et al. [JSL08] propose a scheme to assign IP addresses to newly-joined nodes. Some nodes are coordinators, which are organized in a tree topology by exchanging *hello* messages. New nodes are able to obtain an IP address by listening to the exchanged *hello* messages and contacting the closest coordinator.

Stateless protocols

Instead of assigning addresses from a common set, stateless autoconfiguration allows nodes to generate addresses by themselves, usually based on a hardware ID (e.g. the MAC address) or randomly generated numbers. In this approach, a Duplicate Address Detection (DAD) mechanism is required to ensure the uniqueness of the addresses (notice that DAD schemes usually introduce a large overhead to satisfy the uniqueness of IP addresses). Even IPv6 addresses, which are usually based on an hardware ID, are not necessarily unique (e.g. most network adapters allow users to change the MAC address to arbitrary values).

An example of a stateless protocol for IPv4 is discussed in [CES01], where each node floods the network with an address request message. If no reply is received before a timer expires, it is assumed that the address is not used. In [OAM09], authors proposed a strategy based on using a set of keys to reduce the imposed overhead of DADs mechanism. Another example of a stateless protocol is discussed in [CSX+08], where each node in the network is capable of assigning a unique IP address with low latency. The address allocation protocol is based on the concept of quadratic residue, and the method guarantees disjoint address spaces. In [KM02], a hierarchical approach for IPv6 is adopted, but a considerable amount of bandwidth is needed for the detection of duplicate addresses. Finally, in [YMK07], the authors proposed an autoconfiguration framework for MANETs. One element of the framework is the MANET local address acquisition (MLA). MLA operates in an analogous manner to IPv6 neighbor discovery mechanism, where an MLA node randomly selects an IP address from the reserved MLA range and sends out a broadcast query to confirm its uniqueness (i.e., no duplication). If a conflict exists, another address is picked.

Hybrid protocols

Hybrid protocols combine elements of both stateful and stateless approaches. This approach can lead to more robust protocols, but it may result in higher complexity and higher protocol overhead.

As a first example, the Hybrid Centralized Query-based Autoconfiguration (HCQA) protocol [YE03] utilizes a strong duplicate address detection (SDAD) mechanism along with a centrally maintained allocation table in order to improve address consistency. A node should select an address by itself and verify its uniqueness using the SDAD, thus increasing the autoconfiguration delay. On the other hand, the Passive Autoconfiguration for Mobile Ad Hoc Networks (PACMAN) protocol [Wen05] combines passive duplicate address detection (PDAD), where nodes passively collect information about already assigned addresses in the network from routing protocol traffic, with a distributed maintenance of a common set of addresses. In this protocol, no additional bandwidth is consumed.

Overall considerations

Notice that, currently, MANETs rely mostly on the IEEE 802.11 technology, which requires additional setup of the different MAC layer parameters before any IP address assignment can take place. In addition, to activate multi-hop relaying, a common routing protocol must be started by all the nodes, thereby achieving successful participation in the MANET. The main drawback of all the aforementioned proposals is that, for a fully functional 802.11-based MANET to be created, IP address assignment is not the only problem to solve. Thus, solutions offering full configuration of the different network protocols involved, both layer-2 and layer-3, are required.

Moreover, most of the earlier works consider MANETs with an undetermined period of operation life, where nodes can join or leave the network whenever they want. This means that the IP address assigned to a node each time it joins the network may be different, and so IP address management using any of the aforementioned procedures becomes critical. In addition, these solutions take into account the possibility of MANET partitioning/merging and connecting to the Internet. In chapter 6 we propose two methods offering full MANET configuration addressing both layer-2 and layer-3 parameters, while avoiding the limitations of other related works in this field.

4.2.3.2 Name-to-Address Translation

Concerning the name-to-address translation problem, many common user applications cannot run in MANETs if there is no method for name resolution. The hierarchical domain name system (DNS) structure works well on the fixed Internet, but it cannot be adopted for MANETs due to their dynamic network topologies. This dynamism can lead to frequent changes of nodes' IP addresses. Therefore, communications among network members are more likely to be using names.

The Name Directory Service (NDR) [JJH03] provides mobile users not only with the name service, such as name-to-address translation in multi-hop ad hoc networks, but also the directory service, allowing mobile users to know their neighbors and to exchange user information. In [XJRY05], authors present a Distributed Naming System for MANETs (ADNS) that supports unstructured name convention. ADNS includes: (i) a distributed server system that allows redundancy for fault tolerance, (ii) a lookup scheme that reduces query overhead and balances the query/response load, and (iii) a server maintenance scheme that enables the server overlay network to evolve with mobility. The Autonomous Name Resolution Scheme (ANARCH) [MMHH03] assigns to MANET members a unique user-oriented name and relieves users and their applications from the necessity of identifying the network addresses of destination nodes. ANARCH exchanges control messages only between nodes that are one hop away. However, it can also resolve names beyond the one-hop boundary. Finally, the Unmanaged Internet Architecture (UIA) [BJC+06] offers zero-configuration connectivity among mobile devices through personal names. Those personal names allow conveniently finding and expressing who they want to talk to, what devices they wish to access, and who can access their own devices.

Notice that some of the previous proposals are designed to be integrated with the Internet DNS, which restricts them to be applied on real mobile ad hoc network environments.

4.2.3.3 Service Discovery

Concerning the service discovery problem, it is important for MANET nodes to be able to advertise their own services to the rest of the network, as well as to locate network services along with the requested attributes. A standard solution to service discovery in MANETS is to broadcast a service discovery request throughout the network. This solution is inefficient in terms of bandwidth and resource usage since the request has to be processed by all nodes, which have limited processing capability and battery power.

In [GC01], the concept of service discovery for MANETs was presented. The authors use the concept of a service agent to introduce a scalable service discovery protocol for MANETs; they evaluate multi-hop service discovery queries and replies in MANETs. In [YW03], authors present a dynamic service discovery infrastructure that uses XML to describe and match services using the semantic content of service descriptions for MANET. Sailhan and Issarny [FV05] introduce a scalable service discovery protocol for MANETs, which is based on the homogeneous and dynamic deployment of cooperating directories within the network. Scalability of the protocol comes from the minimization of the generated traffic, and the use of a compact directory that enables to efficiently locate the directory that most likely caches the description of a given service. SpeakEasy [MSW+02] provides an infrastructure where users can control arbitrary devices and services on a network through custom user interfaces provided by the devices and services themselves. The Group-based Service Discovery (GSD) [DATY02] protocol proposes a distributed solution for MANETs based on the concept of peer-to-peer caching of service advertisements and group-based intelligent forwarding of service requests. Finally, the Candidate Node Pruning enhanced Group-based Service Discovery Protocol (CNPGSDP) [ZLMX06] proposes an efficient protocol that introduces two schemes to enhance GSD (Group-based Service Discovery): Broadcast Simulated Unicast (BSU) and Candidate Node Pruning (CNP).

Notice that most of the service discovery protocols presented tend to introduce a significant traffic overhead, making them primarily suitable for small networks (i.e., less than 30 nodes).

4.2.3.4 Automatic Multicast Addresses Allocation

Finally, concerning the automatic multicast addresses allocation problem, notice that some multicast-based applications need to obtain a unique multicast address in order to prevent conflicts with other applications, or even with sessions based on the current application. Multicast address conflicts can produce applications to fail in the same way that two hosts if they use the same IP address.

The Zeroconf Multicast Address Allocation Protocol (ZMAAP) [ODBE02] specifies a method for peer-to-peer allocation of multicast addresses without a

MADCAP (Multicast Address Dynamic Client Allocation Protocol). ZMAAP allows multicast-based applications to:

1. Allocate unique multicast addresses and maintain them over time
2. Prevent reallocation of assigned addresses
3. Notify allocation collision

These requirements are different from those required for unicast address autoconfiguration because multicast addresses are shared resources. Often, the allocation control needs to be shared by different processes present in the network. The last two requirements guarantee any process to prolong a session and be able to discover whether the session is still valid. Notice that although ZMAAP was initially designed for small wired networks, there exist implementations for MANETs. As an example, in [JJH04] authors present auto-networking technologies for IPv6 mobile ad hoc networks, offering solutions to the four areas proposed by Zeroconf. Therefore, they include automatic IPv6 multicast address allocation.

4.3 Spontaneous Networks

Somewhere between PANs and MANETs are Spontaneous Networks [LBA01]. A spontaneous network is a small infrastructureless network formed when a group of people come together to participate in some computer-based collaborative activity. The interaction between people in the context of an activity can be used for initializing the required network configuration. Spontaneous networks extend the concept of a PAN to a group of PANs where in addition to intra-personal devices communication, technology also allows inter-personal devices to communicate.

The main characteristics of spontaneous networks, which can differentiate them from other well-known wireless approaches, are:

- The network design is not planned in advance.
- Network boundaries are not defined, thus the network may experience arbitrary partitioning.
- Devices are not pre-configured. It is pointless configuring host names and addresses in advance.
- There are no servers.
- Users are not experts. Configuration steps should be as intuitive as possible.
- Routing issues are not a primary concern (unlike MANETs).

Therefore, in these networks, tasks such as identifying the other participants or configuring their own devices in advance are not required by users. Furthermore, users are not restricted by the availability of a pervasive wireless infrastructure which needs reliable access. Under these assumptions, activities such as interactive

presentations, resource sharing, collaborative tasks, or even games are the ones that better fit with the concept of spontaneous network.

By their nature, spontaneous networks are suitable for devices such as laptops, tablets or smartphones. These devices can be connected between them using a variety of wireless technologies (i.e. Bluetooth, IEEE 802.11, or Infrared), meaning that interactions between heterogeneous devices are possible. Moreover, despite the topology is unpredictable in such environments, it is expected that significant changes will be relatively infrequent. Finally, notice that seamless networking (e.g. Zeroconf) can help out making the formation of spontaneous networks easier.

4.4 Conclusions

In this chapter we offered an overview of the current state-of-the-art of several short radio range wireless networks. We presented personal area networks (PANs) and its main working groups. Then, we focused on two different main topics related to mobile ad hoc networks (MANETs): autoconfiguration of terminals, and routing protocols. Finally, we defined Spontaneous Networks as a type of networks located somewhere between PANs and MANETs.

Concerning routing protocols for MANETs, there is still no agreement among the scientific community members on which is the most adequate routing strategy for mobile ad hoc networks, and so a lot of research is still being done. We explored the main differences between the four most important routing protocols for MANETS (OLSR, AODV, DSR, and DYMO) showing how each routing protocol handles problems such as route discovery and route maintenance. We also offered an overview of the different multipath routing protocols available. Concerning autoconfiguration issues for MANETs, we saw how current solutions are not able to configure both Layer-2 and Layer-3 parameters. In chapter 6, we are going to propose two different solutions able to configure both layers.

In general, service discovery is an important step when we use short radio range wireless networks, mostly because services allow the user to complete some difficult tasks in an efficient an transparent manner. Moreover, many applications for such networks are service-based. In the following chapters we will see how different kinds of prototype applications make an intensive use of these services.

Chapter 5

First Experiences Developing Pervasive Applications

Research into the nature of pervasive computing has been around for more than a decade. Nowadays, pervasive applications exploit mobile wireless communication technologies to interconnect computing devices along with various sensing technologies, setting up a new kind of intelligent environment where applications can transparently search and use services without the users' intervention. However, there are few examples of real world deployment of pervasive computing moving out from academic laboratories into our everyday life. In this chapter we propose two research prototypes that provide context-aware services into different kind of environments, such as conference meetings or hospitals' recovery wards. The experimental research prototypes leverage Bluetooth technology to offer information based on the user preferences. In both cases, we describe the overall network architecture and discuss the implementation steps taken to create the context-awareness infrastructure. Furthermore, we run some experiments in order to evaluate the performance and the systems' behaviour.

5.1 Introduction

The advent of pervasive computing capabilities and their potential in wireless mobile services, are major motivations for developing context-aware information systems. Clearly, contextual services represent a milestone in nowadays mobile computing paradigm, providing timely information anywhere and at any time. Additionally, the miniaturization of devices and the rapid proliferation of handheld computers have paved the path towards a pervasive computing environment.

Therefore, there is no doubt that a pervasive computing system will provide flexible services and unsuspected benefits. However, there are still few examples of real world deployment of pervasive computing moving out from academic laboratories into our everyday life. This occurs because pervasive technologies are still premature, and also because it is hard to define what a real pervasive system should be like. Moreover, there is still no *killer application* in this area.

In this chapter we report our first experiences building pervasive prototype systems that provide context-aware services. The first proposal offers file exchange services between peers within one-hop wireless distance following a peer-to-peer (P2P) approach, which is useful, for example, between co-workers to synchronize their work. Each device or user could act as both provider and client of resource sharing services. The second proposal provides information and location based services to clinicians in hospitals' recovery wards. The system continuously keeps track of patients in emergency and recovery wards, offering precise information to clinical personnel. Moreover, it is also able to track clinicians and patients movements within the hospital. Both systems support many similar scenarios aimed at providing awareness and enhancing communication.

Our proposed systems employ Bluetooth, a versatile and flexible technology that has the ability to locate close-by devices and discover the type of services they offer. Thus, our proposals allow us not only to confirm the correct behaviour of the designed application, but also to acquire experimental data to evaluate how well Bluetooth is suited for context-aware and pervasive systems.

The rest of this chapter is organized as follows: Section 5.2 describes the BluePeer application framework, presenting details of the application itself and offering evaluation results from both testbed and simulation points of view. In section 5.3 we present the BlueHospital system, including the overall architecture, implementation details of the prototype application, and some evaluation results. Finally, in Section 5.4, we have some concluding remarks.

5.2 First proposal: BluePeer

One of our first experiences in developing a pervasive computing system was BluePeer [CCMD06], an application framework to deploy an easy, spontaneous, and infrastructureless network. We combined the Bluetooth technology with the concept of peer-to-peer (P2P) networking to develop an experimental application which enables peers to exchange their resources. A P2P approach for spontaneous networks depends on the existence of a mechanism that allows any node in the network to transparently locate close-by devices and discover the type of services they offer. Bluetooth is a versatile and flexible short-range wireless networking technology with low power consumption which has the aforementioned capabiliies.

The application provides context services to interchange any kind of resources between Bluetooth devices or peers within one-hop wireless distance (the small area covered by the Bluetooth RF range). In the case that two devices are not reachable in one wireless hop, BluePeer allows clients to obtain the information from in-between devices. It is an ideal application for co-workers to synchronize their work. It can also be used in conferences and meetings, allowing the organization to share information with the attendees, and also attendees to share their resources (e.g., snaps of the city, presentations, etc) with one another in a peer-to-peer manner. Therefore, Bluepeer is a simple but representative P2P application which makes the best use of Bluetooth's features for exchanging resources like files in pervasive network environments, and so it can be considered as a proof of concept for other future applications.

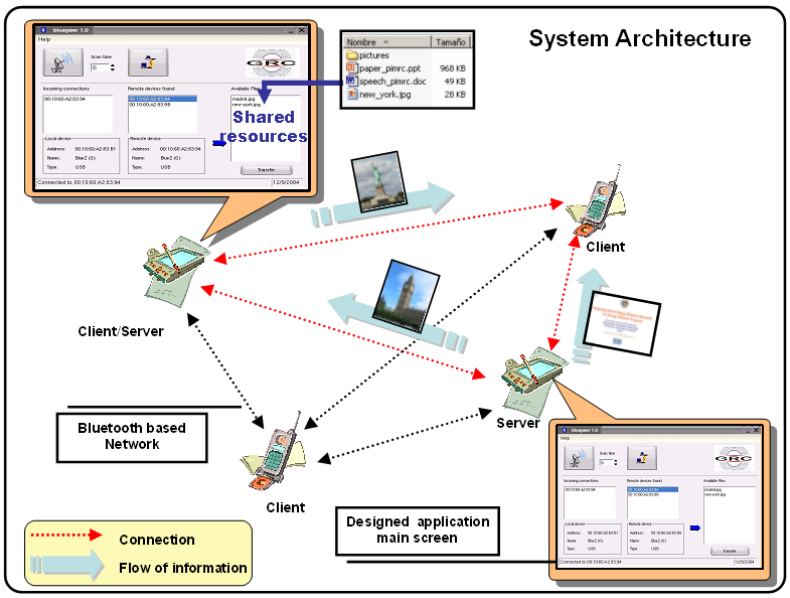

Figure 5.1: A pictorial representation of the peer-to-peer application.

5.2.1 The prototype application

Our proposed application was developed for both laptop and PDA devices using the OpenNETCF Bluetooth library [Foo02], which is a Bluetooth networking library for Windows Mobile, Windows CE and Windows XP devices running the Microsoft Bluetooth stack [BPS04].

The overall network architecture adopts a core peer-to-peer wireless network based on the Bluetooth technology. The basic philosophy behind pure peer-to-peer applications is serverless communication. Therefore, there must not be a single entity that coordinates, controls and informs about the existence of the services available in the network in a centralized manner. Note that a decentralized means of communication has a problem for organizing resources between clients because no centralized service provides a global knowledge of all the elements available in the network. Anyway, Bluetooth's inquiry, discovery and role-switching functionalities allow low-level search, finding, and communication of remote devices and resources preserving peer-to-peer communications. This approach makes coding easy because Bluetooth, as a networking support system, fits nicely into the functions needed for creating peer-to-peer software. Figure 5.1 illustrates the system architecture, in a setting where all nodes are able to communicate with each other.

Each participating node or device in the application can assume either server or client roles. This also means that a node can be in both states at the same time, retrieving information from a server and offering information to one or more clients. When the application starts up, each node starts its server side code and automatically listens for possible clients or peers that may want to interact with

Algorithm 1 Client/Server code for every peer.

```
for (every peer) do
    Server_Peer()
    Client_Peer()

Server_Peer()
    Register services and available resources
    Accept client connections

    while (connections)
        for (every new connection) do
            Update registry of connected clients
            Send resources list

        for (every resource request) do
            Send resource

        for (every disconnect request) do
            Disconnect client
            Update registry of connected clients
    exit()

Client_Peer()
    Register services and available resources
    responses = Inquiry()

    if (responses > 0) then
        Update servers list
        while (servers list size > 0)
            for (every disconnected server) do
                if (interesting server) then
                    Connect to a server

            for (every interesting resource) do
                resource request

            for (every no interesting server) do
                disconnect request
                Update servers list
        exit()
    else
        exit()
```

that particular node. Moreover, when a node wants to communicate or locate a piece of information or file, it starts looking for nearby nodes that offer that required service, inquiring for the information available.

5.2.1.1 Server functionality

The server side code basically deals with the administration of incoming connections for supplying clients with the files that have been chosen. The server keeps a registry of all connections made and the file traffic. Once a client has requested a certain file, the server searches for the file and sends it on the L2CAP connection previously created.

The server side works as follows. First, the server uses the *BluetoothListener()* class constructor to create a server object. Then, the *Start()* function starts the server socket to listen for incoming connections. This function just creates a L2CAP Bluetooth socket, and uses the traditional *bind()* and *listen()* functions to link to the server and attend client connections. Finally, for each client, the server uses the *AcceptSocket()* function to create a new L2CAP socket which will attend the client connection.

5.2.1.2 Client functionality

The client side code involves a little more work for setting up connections. This is mainly due to the necessity of using inquiry/discovery of remote devices, and then, starting a connection to the remote device. Inquiry will find available remote devices within RF range. Remote devices will be shown to the user (including its address and name for simplicity), as the user may want a certain file from any of them. We must state that inquiry's performance differs depending on the room, objects in the locality and distance from remote devices. We can limit to a

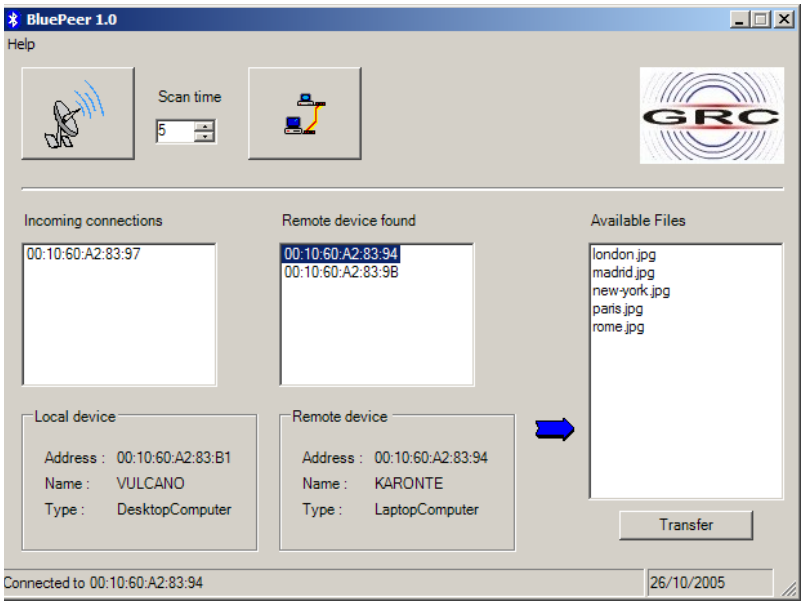

Figure 5.2: A client screen capture of the application for desktop/laptop.

certain degree the time that the user has to wait for results on the user interface, but reducing the inquiry time too much may result in many nodes remaining undiscovered. Once a remote device has been discovered, the client side software is able to connect to it, search for a file, or download the shared file list. Likewise, the client can connect to other devices and repeat the operation. The connection uses the same L2CAP connections to transfer data.

The client side uses the *btnConnect_ Click()* function to implement each of the clients' connections. This function basically works as follows. First, it creates a new *BluetoothSocket* that will be used to attend the connection. Next, the *connect()* function establishes a connection to the remote device, and finally data transfer can start from the remote device.

Algorithm 1 illustrates the client/server code for every peer. Figure 5.2 presents the application screen for a user equipped with a laptop device when finding some resources from nearby devices. Figure 5.3 shows the same screen, but now for a user equipped with a PDA device.

5.2.2 Experimental results

We evaluated the proposed application in a small testbed, and also through simulation. Through the experiments using the testbed, we attempt to acquire experimental data in order to check the applicability of Bluetooth to pervasive computing. We also compare the experimental results with those obtained using a simulation tool. The simulation tool could be a candidate evaluation tool to test our application in scenarios with a large number of peers.

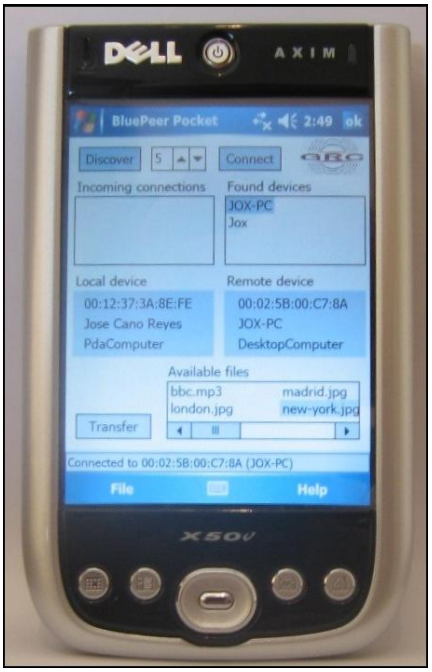

Figure 5.3: A client screen capture of the application for pocket PC.

Our experiments focused on evaluating inquiry delay and throughput performance of the Bluetooth channel. We present our findings in terms of inquiry delay and transfer time of resources with respect to spatial distance and peers speed. The final objective is to evaluate whether our application could work under acceptable conditions (in terms of delay of the inquiry procedure) without causing annoyance to the end user. These results can be extensible to other applications created using Bluetooth in a P2P manner.

5.2.2.1 Testbed performance evaluation

Our testbed consists of a set of peers, where each one can play the role of server, client, or both at any moment. Each peer runs on an Intel Pentium IV 2000 Mhz TOSHIBA Satellite 1900-303 laptop based on Suse Linux 9.1 with a 3Com CREB96 Bluetooth USB Dongle.

Inquiry delay evaluation

We measured the duration of the inquiry procedure through a sample of 100 independent runs. We evaluated two different cases: (i) considering a distance of 1 meter between client and server devices, and (ii) considering a distance of 10 meters. Peers move around randomly and at low speeds.

The inquiry cycle scans all available frequencies, and it is recommended that it should last for at most 10.24 seconds [Blu02]. To avoid consuming the whole

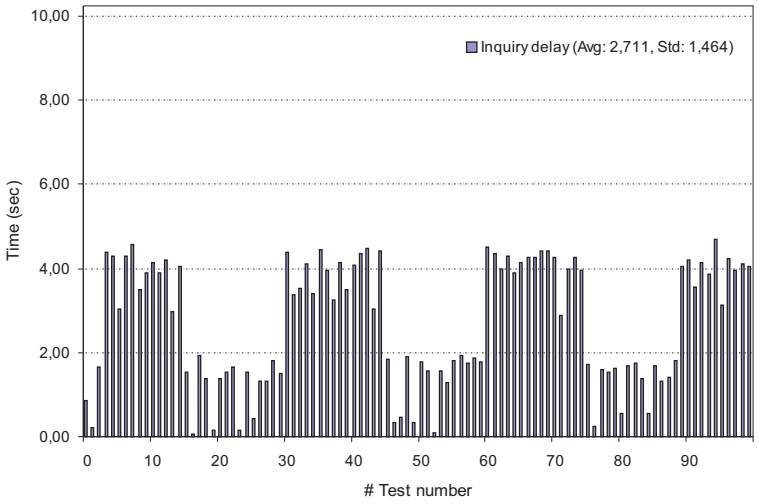

Figure 5.4: Inquiry delay value for 100 tests. Distance between peers is $\simeq$ 1m.

inquiry timeout period of 10.24 seconds, we fixed the maximum number of discovered devices equal to half the number of peers. In our testbed, the total number of peers ranges from 3 to 10.

Figure 5.4 shows the obtained inquiry delay by a client peer using a distance of 1 meter for a total of 100 tests. The obtained results have an average inquiry delay of 2.71 seconds and a standard deviation equal to 1.46 seconds, with a confidence interval of 95%.

Figure 5.5 shows the histogram distribution of the inquiry delay and the cumulative results as a function of time for the 100 tests. The results show that users can discover the desired information of nearby peers in less than 3 seconds in about 50% of the times. This waiting time of 3.00 seconds is normally acceptable to a user without causing annoyance. More important, we observe that this percentage increases to 95% after 4.50 seconds. Therefore, we can approximately assume 5 seconds as the highest waiting time for the inquiry process in a closed environment where peers are located close to each other.

We now consider a scenario where the distance among peers is of about 10 meters (Figure 5.6). We obtained an average inquiry delay of approximately 7.46 seconds and a standard deviation of 3.35 seconds with the same confidence interval. Therefore, it is observed that, as we increase the distance among peers to nearly 10 meters, the inquiry procedure delay increases. Moreover, for some tests, the inquiry timeout of 10.24 seconds expires with no device discovered. In those cases, we could detect some peers if we allowed more that just one inquiry cycle.

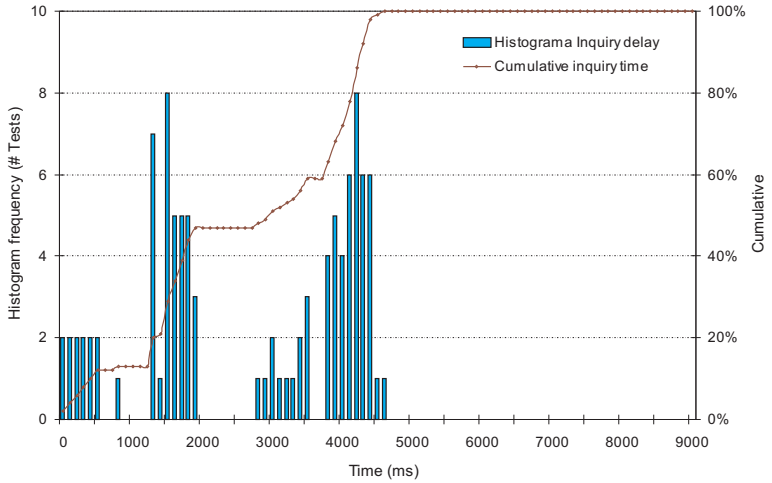

Figure 5.5: The cumulative inquiry delay distribution for 100 tests. Distance between peers is $\simeq$ 1m.

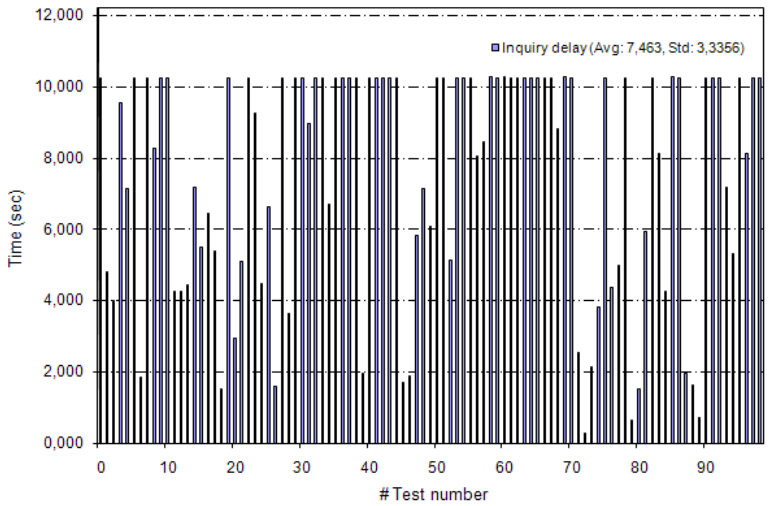

Figure 5.6: Inquiry delay value for 100 tests. Distance between peers is $\simeq$ 10m.

Throughput evaluation

We now measure the impact on throughput when varying the spatial distance between peers or the peers speed. We evaluate throughput performance in terms of transfer time of shared resources. We use the more efficient $DH5$ data packet, and select share resources with different sizes. $DH5$ packets use 5 data slots at the baseband layer, and Forward Error Correction (FEC) using Hamming codes is not used [Blu02].

Each of the results shown in the following figures represent an average (μ) of n different executions ($X_1, X_2, ..., X_n$). We varied the n value to obtain an estimation of (μ) with a 90 percent confidence interval using the following definition:

$$\overline{X}(n) \pm t_{n-1,0.95}\sqrt{\frac{S^2(n)}{n}} \tag{5.1}$$

where $t_{n-1,0.95}$ is the upper 0.95 critical point for Student's t distribution with $n-1$ degrees of freedom, $\overline{X}(n)$ is the sample mean and $S^2(n)$ the sample variance.

Figure 5.7 shows the obtained results when peers of the spontaneous network are static. As we increase the size of shared resources, transfer times obviously increase.

We observed that Bluetooth offers a decreasing throughput as the spatial distance among peers increases. However, for a given spatial distance, throughput remains stable regardless of the size of the shared resources. Up to 10 meters, the application still obtains an acceptable throughput: 69 kB/s for 1 meter, 45 kB/s for 3 meters, 40 kB/s for 5 meters, and 30 kB/s for 10 meters. Beyond 10 meters, the application suffers sharp performance degradation due to frequent packet losses. Throughput approaches 3 kB/s when the spatial distance is about 15 meters. Notice that all the obtained results are below the maximum throughput stated in the Bluetooth specification, which provides a reference value of 90.40 kB/s for DH5.

We also observed that the application offers a relatively stable transfer time with speed. As we increase the speed of peers from 0 to 2 m/s, the average throughput was approximately 40 kB/s, 36 kB/s, and 32 kB/s, respectively. Figure 5.8 shows the obtained transfer time using shared resources of $100kB$, $200kB$, and $700kB$ respectively.

Finally, we repeated all the previous experiments by increasing the number of concurrent clients. We observed that all the results obtained exhibit the same behaviour as the one presented in Figures 5.7 and 5.8. When using a higher number of concurrent clients, the capacity of the Bluetooth channel should be shared among several transmissions, thus increasing the transfer time for a given shared resource.

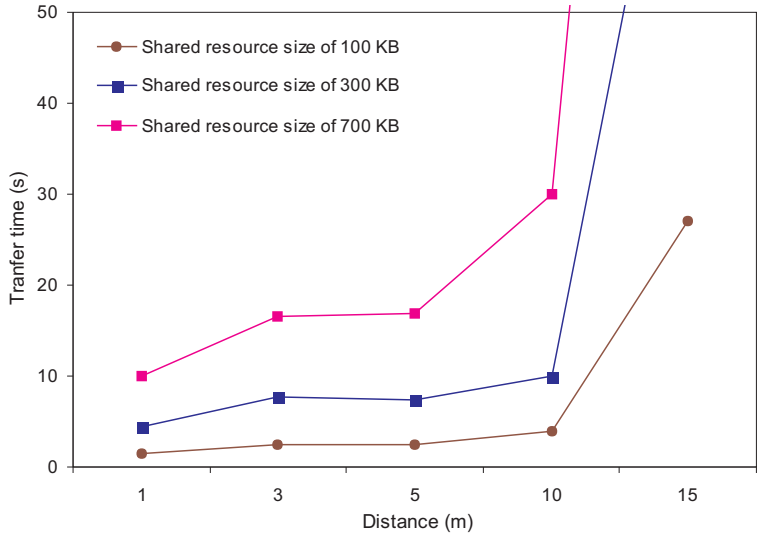

Figure 5.7: Application transfer time for different shared resources with respect to distance. Peers are static.

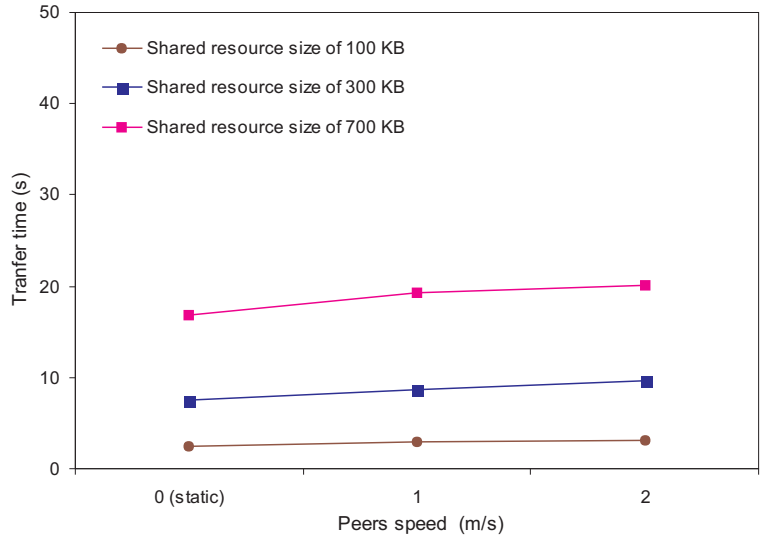

Figure 5.8: Application transfer time for different shared resources with respect to peers speed. Spatial distance of about 5m.

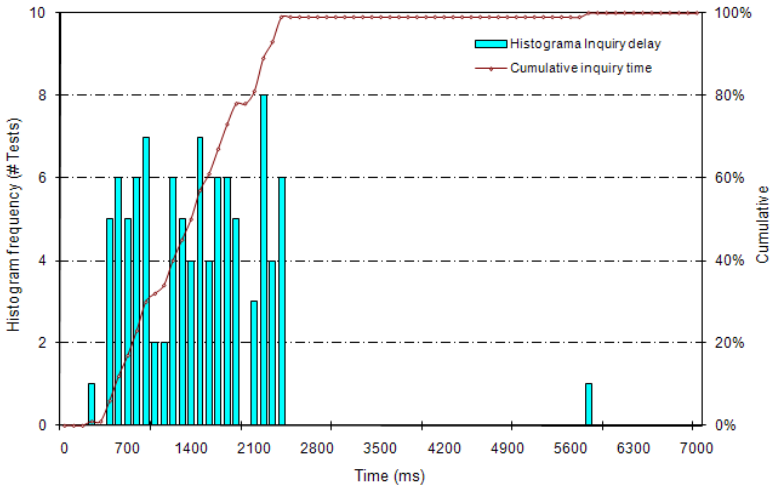

Figure 5.9: The cumulative inquiry delay distribution for 100 simulated tests. Distance between peers is $\simeq$ 1m.

5.2.2.2 Simulation results

Concerning the simulation tests, we used the IBM BlueHoc simulator [Lab01] to take advantage of the flexibility provided by simulation tools. BlueHoc is an open source code that extends the ns-2 simulator [KK00] to implement Bluetooth. We set up a BlueHoc scenario similar to our test-bed scenario. As in the test-bed experiments, we evaluated the duration of the inquiry procedure and the throughput performance.

Inquiry delay evaluation

In the following tests, we assume that peers are in both inquiry mode and inquiry scan mode. Moreover, the spatial distance among peers is incremented from 1 to 10 meters.

Considering the 1 meter scenario, we obtained an average inquiry delay of 1.516 seconds and a standard deviation value of 0.738 seconds with a confidence interval of 95% (not shown).

Figure 5.9 shows the histogram distribution of the inquiry delay and the cumulative results as a function of time for the 100 tests. The results show that users can discover the information of nearby peers in less than 2 seconds in about 50% of the times. Moreover, we see how this percentage increases to 95% after 2.3 seconds. Therefore, we can approximately assume 2.5 seconds as the highest waiting time for the simulation inquiry process in a closed environment where peers are located about 1 meter away to each other. Comparing with the testbed experiments, the simulated tests show a slightly reduced inquiry delay while obtaining a smoother behaviour.

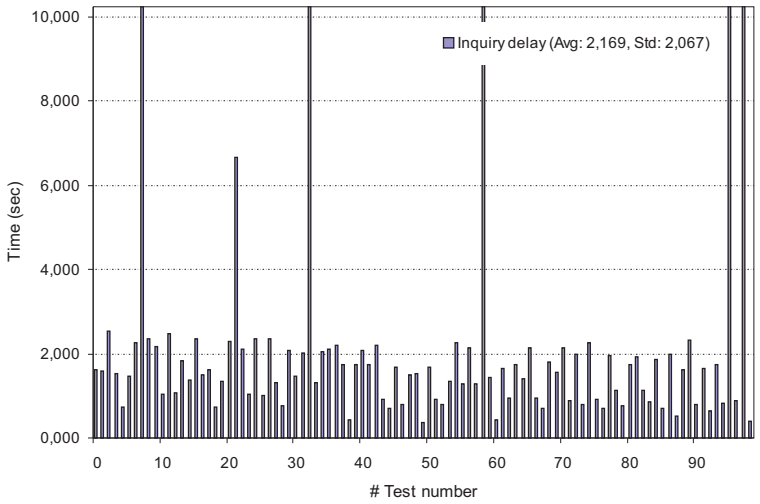

Figure 5.10: Inquiry delay value for 100 simulation tests. Distance between peers is $\simeq$ 10m.

Figure 5.10 shows the obtained results considering a distance between peers of 10 meters. As we observe, we obtained slightly higher values (average inquiry delay of 2.169 seconds and a standard deviation value of 2.067 seconds). Therefore, it is observed that, the simulated inquiry procedure does not seem to be sensible to distance as in the real case.

Therefore, we conclude that there are some differences between the experimental and the simulation cases. We think that those differences are basically due to the Bluetooth protocol stack used in each case.

Throughput evaluation

We now evaluate throughput using simulation with FTP traffic sources. Figure 5.11 shows the obtained results for a static network when using DH5 packets at the baseband layer. We observed that, as distance increases from 1 to 15 meters, Bluetooth offers a decreasing throughput from 69 kB/s to 12 kB/s. Compared to the testbed results, the average throughput starts to get worse exactly at 5 meters, while in the testbed it decreases more gradually. We also observed that, above 5 meters, the propagation model of the BlueHoc simulation tool imposes fewer performance penalties than those imposed in the real world.

Finally, we repeated the simulations in a dynamic network. Figure 5.12 shows the obtained transfer time when simulated peers keep moving. It is observed that peers speed does not significantly affect the performance. Therefore, we can conclude that, with respect to peers speed, in a spontaneous network the results are quite similar to those obtained in our testbed.

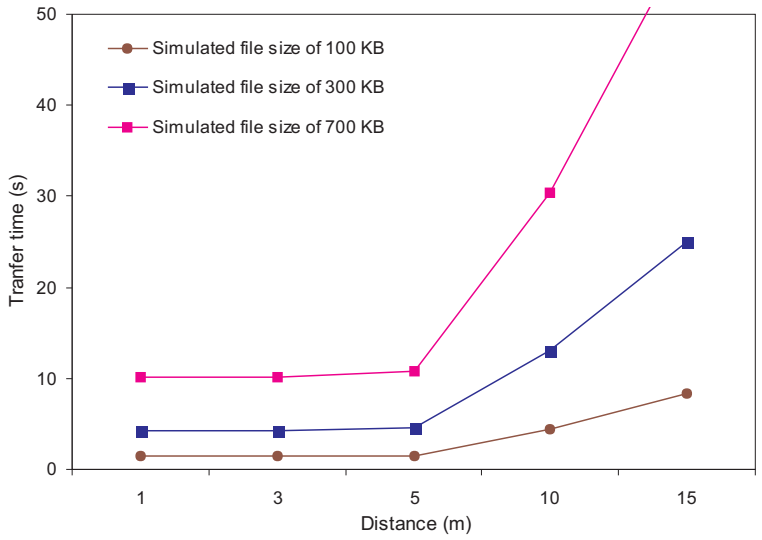

Figure 5.11: Simulated transfer time for different file size with respect to distance. Peers are static.

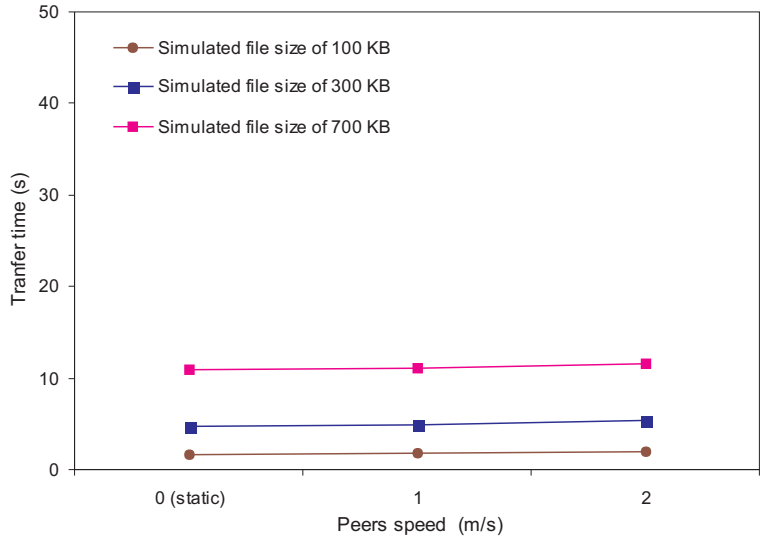

Figure 5.12: Simulated transfer time for different file size with respect to peers speed. Spatial distance of about 5m.

5.3 Second proposal: BlueHospital

In this section, we present the BlueHospital system ([CCCM10]), a pervasive prototype that provides context-aware information and location based services to clinicians in hospitals' recovery wards. We built the system to continuously keep track of patients in emergency and recovery wards, offering precise information to clinical personnel. The system is also able to track clinicians and patients in this location and context-aware infrastructure.

We developed both client and server applications, providing routines to handle doctor requests about their patients, find empty recovery wards, do profile logging, and also content filtering. For example, the system may use location tracking functionalities to find a doctor, or it can notify a patient's ward about his medical treatment. Our system supports many similar scenarios aimed at providing awareness and enhancing communication.

BlueHospital employs Bluetooth, a versatile and flexible short-range wireless networking technology which is also able to avoid interference with existing medical equipment which usually has proper shielding [JO01]. The proposed system allows us not only to confirm the correct behaviour of the designed application, but also to acquire experimental data to evaluate how well is Bluetooth suited for this kind of context-aware and pervasive systems.

5.3.1 System Architecture

The overall network architecture is based on the cooperation of an *edge* wireless network and a *core* wired network. The edge part is based on Bluetooth technology alone. The core network is based on a fixed 100 Mbps Ethernet local area network used to connect the BlueHospital *edge* infrastructure with the centralized database.

Figure 5.13 shows a pictorial representation of BlueHospital's system architecture. The system considers four types of entities: hospital patients (BH_Patient), room managers (BH_Room_Manager), clinical personnel (BH_Doctor), and the central database server (BH_Database_Server). A doctor provided with a Bluetooth enabled PDA is the basic example of a BH_Doctor entity. BH_Doctors are connected to the central database through the room manager to receive information about patients, including their case history as well as comparable cases. Doctors can make diagnoses faster using all the information available at the central database, choosing the best therapy or medicine for each particular patient. There is a BH_Patient associated to each patient who has been admitted and allocated in a recovery ward. The BH_Patient connects to the room manager to register with the central database. The data being monitored consists of frequent measurement of body temperature and heart rate. There is a BH_Room_Manager associated with every recovery ward that acts as a bridge between doctors and patients' sensors to the central database, and viceversa. Finally, the central database is able to generate precise patient profiles based on clinicians' requests and preferences.

Being BlueHospital a context aware system, it is able to provide valuable information to clinical personnel without any user interaction. When a clinician is visiting patients at recovery wards, his application will automatically search for the room manager device, which will offer any new information of interest about

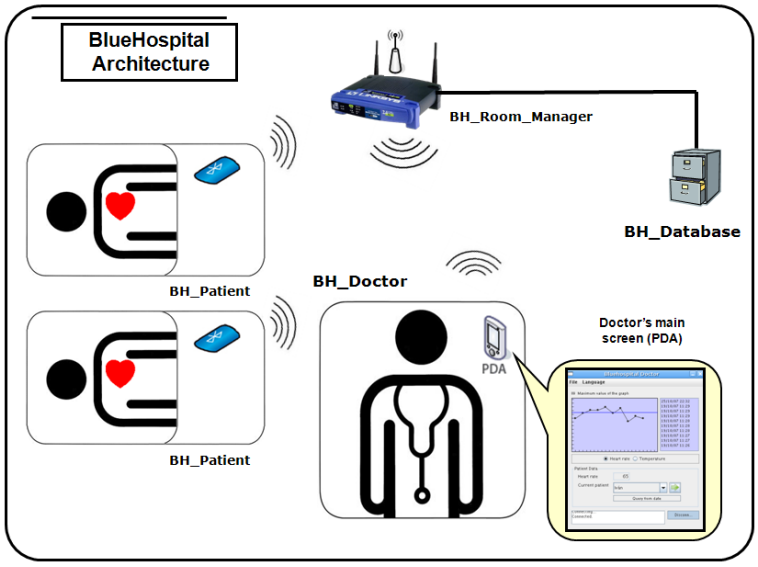

Figure 5.13: The BlueHospital System Architecture.

the patient. If a user wants to look up new information or introduce information in the system, he must send the request together with the user profile that was entered in the initial configuration process to the manager, i.e., the user is a doctor, a nurse, etc. Knowing the user profile, the room manager can process the request by combining the user profile with some additional information, and it will send it to the central data server. There the request is logged and processed, and a reply is returned to the room manager which relays it to the clinician. All the process takes place automatically, and clinicians can change their profile at any time to receive more details about the patients in their own language, and adapted to their device, i.e., mobile phone, PDA, or laptop.

The BlueHospital system is also able to track clinical personnel and patients in its location and context aware infrastructure. Instead of using some commercial system such as [Ubi03], a precise but too costly system, we delegate on Bluetooth the location functionally of our system. Since we only require location tracking of patient and clinicians with a room level granularity, we develop our own coarse-grained location system. All BH_Room_Manager devices in the recovery wards include a USB Bluetooth adapter with up to 10 meters range. When the BH_Room_Manager discovers a new authorized Bluetooth device enabled through Bluetooth's inquiry process, it will accordingly update the tracking information within the BH_Database_Server. The proposed solution also allows to use existing Bluetooth enabled mobile phones owned by clinical personnel and patients for location tracking.

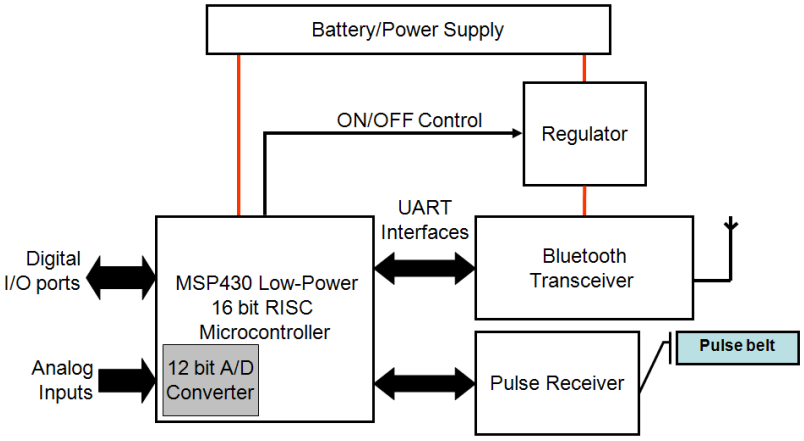

Figure 5.14: Schematic BH_Patient system overview.

5.3.2 System Development

When developing a suite for a pervasive computing system, two main lessons were confirmed from our previous experience. Firstly, we require a low-cost solution that offers computation and communication capabilities to an ordinary object. Secondly, minimizing power consumption and size are mandatory in order to make more apparent the realization of ubiquitous computing.

To this end, we collaborated with the Electronics Technology Department at the University of Malaga, in Spain, to develop an unexpensive platform prototype for ubiquitous computing, which has been implemented based on commercial off-the-shelf components. The BH_Patient device consists of a low power microcontroller for collecting data from a heart rate sensor and handling the Bluetooth wireless communication through a serial UART interface. Concerning the BH_Room_Manager, we select a low cost solution based on wifi routers which accommodate Bluetooth connectivity through an USB port, as well as a Fast Ethernet interface to connect to the central database. With respect to the BH_Doctor entity, we developed a PDA oriented application using the standard Java APIs for Bluetooth wireless technology (JABWT) [Kum02, TKK08]. The application was designed to be used by clinicians' PDAs. The central BH_Database_Server stores in an SQL database all the information related to the patients, as well as other information concerning the BlueHospital system.

5.3.2.1 Patient Device (BH_Patient)

Figure 5.14 shows the schematic system overview of the Bluetooth-based patient prototype ([CCM+07]). The system consists of a low power microcontroller connected to a Bluetooth transceiver and to a heart rate monitoring system through a serial UART interface, so that it can collect data from the measuring system and handle the wireless communication by sending the received data to the BH_Room_Manager device.

In order to optimize battery life, the Texas Instruments MSP430 [Tex05] ultra-low power family of microcontrollers has been selected because it is particularly well suited for power constrained applications. This family of microcontrollers features a wide range of special purpose peripherals (Real-Time Clock, A/D converter, PWM generators, generic Timers/Counters, UART interface, etc.) and combines a quite powerful 16-bit RISC 8 MIPS CPU with several low-power modes. In low power modes, the microcontroller CPU is halted so that the required current supply is as low as 3μA, but some peripherals such as the Real Time Clock or the External Interrupt Subsystem remain active and are able to wake up the microcontroller CPU upon certain events. A highly configurable clock system allows the CPU clock rate to be dynamically adapted to further adjust power consumption in the active mode. More specifically, the MSP430F1612 microcontroller has been selected for this application. This particular microcontroller features 55 kB of embedded Flash ROM and 5 kB of embedded RAM.

Concerning the wireless communication, a wide range of serial HCI Bluetooth transceivers are currently available from different vendors. In this system, a Mitsumi WML C11 [Mit05] class 1 Bluetooth module has been used. This Bluetooth module is based on the CSR BlueCore 2 chipset and is a fully qualified transceiver compliant with the Bluetooth 1.1 Specification [Blu02]. Some of the operational parameters of the Bluetooth module, such as the maximum transmission power, can be altered by using the CSR PSTOOL application [CSR02] to better fit the requirements of the end system. The Bluetooth transceiver implements the lower layers of the Bluetooth standard, i.e., the Radio Base-Band and the Link Manager. The layers supported by the microcontroller firmware are the HCI and the Logical Link Control and Adaptation Protocol (L2CAP). The Bluetooth transceiver is powered by a regulator and can be shut down by the microcontroller to reduce power consumption.

With respect to the heart rate monitoring system, we select a solution similar to the one presented in [TCJ03], which consists of a belt worn and a receiving unit. Another option, although expensive, consists on using a smart suit plugged into the shirt around the patient's chest and abdomen. The suit incorporates embedded sensors which collect relevant data coming from different sensor types, such as respiratory and cardiac functions. See [OCPP07] for more details.

5.3.2.2 Room Manager (BH_Room_Manager)

Each operating ward is equipped with a BH_Room_Manager, which consists of a low cost wireless router connected to the Bluetooth based *edge* wireless network and the Ethernet *core* wired network. The solution is based on the ASUS WL-500g Premium wireless router [Asu06], which contains almost all features required to deploy a small network. It includes the IEEE 802.11b/g wireless standard, and several Fast Ethernet ports to connect either to a LAN or to a WAN. You can even set up a network download center, a printer server or a public ftp-server within it. We select one of the 2 USB 2.0 ports available to configure our BH_Room_Manager Bluetooth connection using a Belkin class 2 Bluetooth USB adapter.

We set up the BH_Room_Manager by using OpenWRT [Ope04], which offers a complete Linux distribution with all the features required to tune the system in

a very flexible way. The ASUS WL-500g wireless router comes with the standard operating system, which offers only basic functionality. So, we have to update each router using the OpenWRT firmware. OpenWRT allows us to easily configure our USB Bluetooth interface providing all the functionality and flexibility of a Linux distribution for embedded systems.

The first action of a BH_Room_Manager is to connect to the hospital's central database to inform the server of the recovery ward the manager is at. After being assigned the proper ward, the room manager starts the Service Discovery Protocol (SDP) daemon, and will register its service along with the appropriate attributes. Each room manager will register a new service called BH_Room_Manager_X, where X represents the room number. Once all the SDP registrations have been done, the information point will create and bind the L2CAP layer sockets and wait for connections coming from the BH_Doctor applications.

When a connection comes in, the BH_Room_Manager will spawn a child process to deal with the mobile client's request. The child process will receive the client's profile, and it would be sent to the central Server. There, it will be logged and processed according to the client's profile. Eventually, the required information will be sent back and passed on to the client. The parent process will carry on waiting for further client connections.

5.3.2.3 Doctor Application (BH_Doctor)

The first step for each BH_Doctor application is initialization: the user has to state his preferences as a basic set of input parameters, that is: (a) the type of device, (b) the preferred language and (c) the user identification.

Once all the data is filled in, and the user is identified/authenticated, the application will search for information offered by the BH_Room_Manager through the Bluetooth inquiry process and SDP searches. Once connected to it, the application will send the stored profile to the room manager and it will eventually receive the information. Once all the information is received, the application will present it to the user. The user can then request new information, e.g., to check in detail a given value, to select the period of time to be analyzed, or to select a different patient. Figure 5.15 presents the screen that BlueHospital will show to a doctor with a PDA device while visiting their patient for diagnosis and medication.

5.3.2.4 Central Database

The central database server has two main functions: (a) to attend connections for BH_Room_Managers requests, and (b) to manage the SQL database, i.e., to handle all the information related to patients and clinicians, clinic schedules, and location tracking information.

The BH_Database_Server starts and waits for a connection on the default server port. When it receives a connection request, a new process is spawned to attend it. If the connection request comes from a valid BH_Room_Manager, the data server will receive the user profile. The server will acknowledge each profile option received to provide higher data consistency check. Once the whole profile

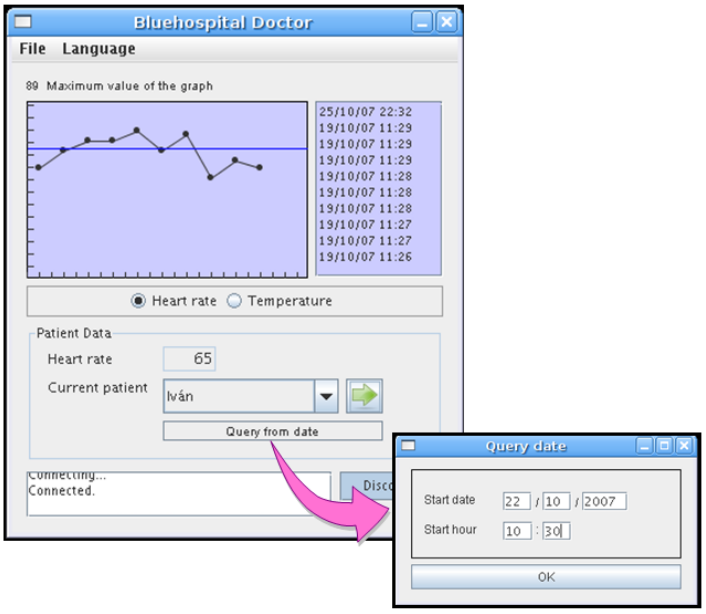

Figure 5.15: A screen capture of the application related to the heart rate of a monitored patient offered to a doctor or nurse.

is received, the server will log the request for security and for statistical data gathering. After the logging process completes, the data server will obtain (according to the received profile) the requested information from the SQL database. This information will be sent back to the room manager, which in turn will resend it to the BlueHospital application. Once the process of sending data to the room manager finishes, the connection will be dropped.

The central BH_Database_Server stores in an SQL database all the information related to the system. It is based on four different tables: (a) the patient table, (b) the patients measurement table (c) the clinical personnel table, and (d) the location table (See [Ben07] for more details). SQL provides a higher level of security and a more efficient storage support and maintenance.

5.3.3 Performance Evaluation

We evaluated our BlueHospital application in a small testbed to confirm its correct behavior. We focused on evaluating throughput performance and inquiry delay of the Bluetooth channel. We performed a sensitivity analysis to evaluate how spatial distance affects throughput and inquiry delay. The central BH_Database_Server runs on an Intel Pentium IV 2400 Mhz standard PC based on Suse Linux 8.2. We have one WL-500g Premium wireless router acting as the BH_Room_Manager with a Belkin class 2 Bluetooth USB adapter to connect to the Bluetooth-based network, i.e., patients and clinicians. Finally, the BlueHospital application runs on a QTEK S200 PDA based on Windows Mobile 6.

5.3.3.1 Application Transfer Time

We evaluate the impact on transfer time of both the ACL packet-type and the device distances between the BH_Room_Manager and the BH_Doctor application. We want to evaluate how much time the application should wait to receive the information. According to the Bluetooth specification, an Asynchronous Connection-Less (ACL) link is parameterized by packet size and data encoding. Each ACL link allows the use of 1, 3, and 5 slot data packets, where a slot is 625μs long. Additionally, it allows the optional use of Forward Error Correction (FEC). According to these parameters, at the baseband layer, Bluetooth offers 6 different data packets that can be classified in two main groups: Data Medium Rate packets, DM packets, which provide a 2/3 FEC Hamming code, and Data High Rate packets, DH, which provide no FEC coding at all.

After the inquiry process is completed, the room manager will collect all the parameters it needs from the client. After that it will send the actual data-block to the client. The application data-block size averages 150 kBytes, and includes images and text. Figures 5.16 and 5.17 show the obtained results when the room manager and the application are using DH packets and DM packets respectively.

We first observe that Bluetooth offers a quite steady throughput below 10 meters, independently of the data packet type selected. Above the 10 meters limit the application still works without a sharp performance reduction. When we select the more efficient DH data packets, we improve the observed transfer time with respect to the DM packets. The results confirm that, although the use of DM packets increases efficiency while reducing the probability of successful transmission, the DH packets can be a candidate for more efficient transmissions. Moreover, after 10 meters, and as distance increases, longer packets ($DH5$ and $DH3$) suffer a higher degradation.

5.3.3.2 Inquiry Delay Evaluation

We now measure the duration of the inquiry procedure. We evaluated two different cases: with the distance from the client application to the room manager device fixed at 5 meters, and with a distance of 10 meters.

Figure 5.18 shows the histogram distribution of the inquiry delay and the cumulative results as a function of time over the 100 tests (for a distance of 5 meters). The results reported that after 2.5 seconds the application discovers the room manager in 50% of the tests. More importantly, we observe that this percentage increases to 95% only after 5.264 seconds. Therefore, we can approximately assume 5 seconds as the highest waiting time for the inquiry process. This waiting time is completely acceptable to clinical personnel without causing annoyance.

As we increase distance among peers near to 10 meters, the inquiry procedure delay increases (not shown). Moreover, for some of the tests, the inquiry timeout of 10.24 seconds expires with no device discovered.

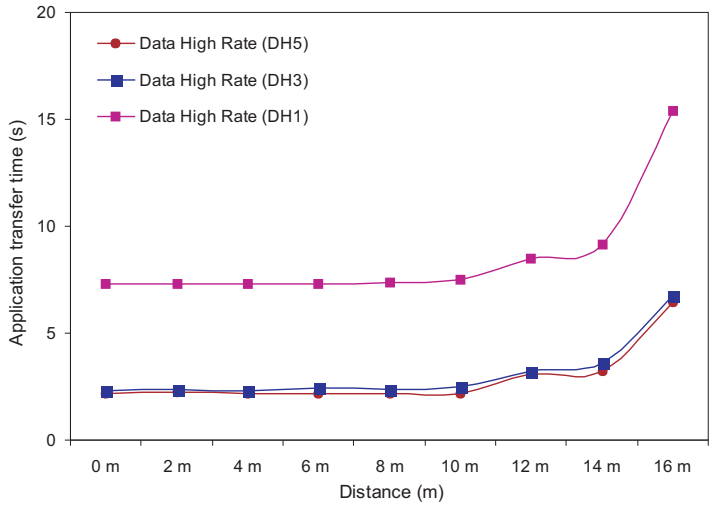

Figure 5.16: Obtained application transfer time comparison of an ACL link using different DH_x packets as a function of distance.

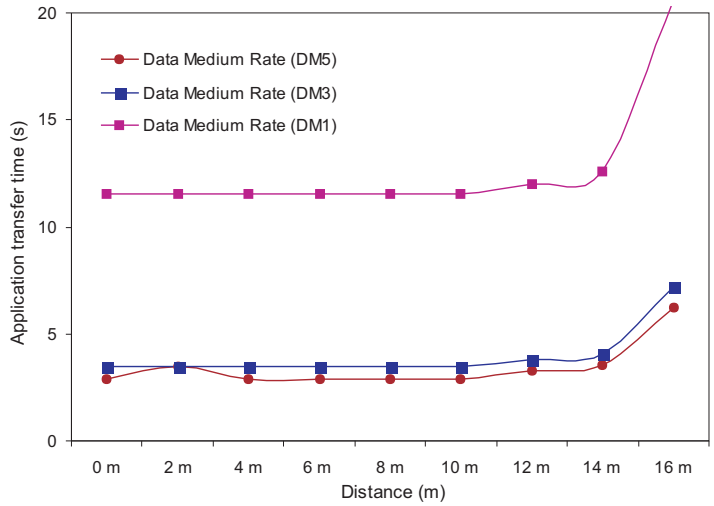

Figure 5.17: Obtained application transfer time comparison of an ACL link using different DM_x packets as a function of distance.

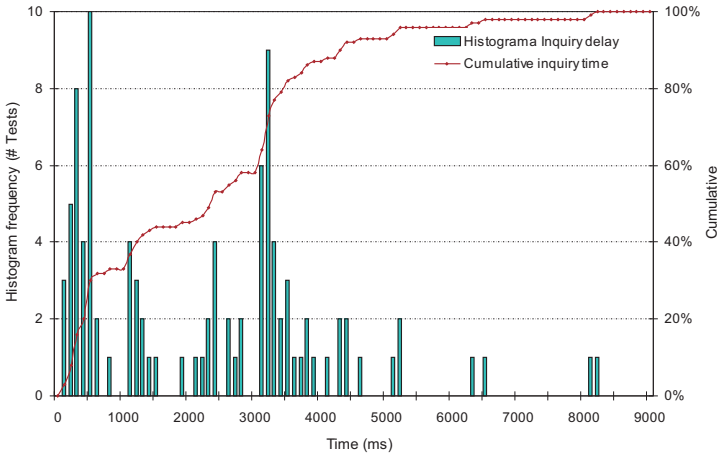

Figure 5.18: The cumulative inquiry delay distribution for the 100 tests. Distance between the Bluehospital application and the BH_Room_Manager is 5m.

5.4 Summary

Nowadays, pervasive applications exploit mobile wireless communication technologies to interconnect computing devices along with various sensing technologies, setting up a new kind of intelligent environment where applications can transparently search and use services without the users' intervention. However, there are still engineering problems to be solved before the envisaged applications become a reality, and up to now, few real-world applications have been developed and deployed.

In this chapter, we demonstrate the practicality of using the Bluetooth's low-level features to build application frameworks for pervasive network environments. We presented two experimental applications that provide: (i) transparent services to interchange resources between co-workers in order to synchronize their work, and (ii) context-aware information and location based services to clinicians at hospitals' recovery wards.

The prototypes allow us to acquire experimental results to evaluate how well the Bluetooth technology is suited for the development and deployment of pervasive applications. Performance evaluation was made with focus on inquiry delay and throughput. We find that Bluetooth offers a relatively steady throughput up to 10m. Overall, our experiments demonstrate the feasibility of Bluetooth technology as outstanding contenders among providers of network support systems for the deployment of applications in pervasive environments.

Therefore, these early experiences have shown that using services in pervasive environments is very important, because applications make heavy use of them. In the next chapters we are going to see how services can efficiently simplify tedious tasks from the user point of view.

Chapter 6

Autoconfiguration of IEEE 802.11-based MANETs.

The deployment of mobile ad-hoc networks involves several configuration steps, which complicate research efforts and hinder user interest. This problem prompts for new approaches offering full autoconfiguration of terminals at the different network layers involved. Continuing the trend of developing proposals for pervasive environments, in this chapter we propose two solutions for autoconfiguring 802.11-based MANETs. The first one is based on Bluetooth technology. The second one does not need resorting to any additional technology because it relies on SSID parameter embedding. Both methods allow non-expert users to transparently join an existing MANET, providing a fast and reliable autoconfiguration of the terminals, even in the presence of encrypted communications. This way, they provide an adequate system for the quick setup and deployment of any sort of organized team that, at least initially, are within a well-defined geographical area. An example could be a fireman rescue team departing from an initial meeting point where, while receiving instructions, terminals get auto configured. Once ready, they can enter the target scenario and communicate via Wi-Fi (in ad hoc mode) without the need for any infrastructure support. Experimental testbed results using real implementations allow assessing the performance of the proposed solutions and show that nearby nodes can be configured in few seconds. Additionally, we present a comparison between both solutions.

6.1 Introduction

As we saw in chapter 4, mobile ad-hoc networks are an attractive networking option for connecting mobile devices quickly and spontaneously. In fact, their characteristics in terms of flexibility and cost have made them an important candidate technology to be used in different kind of applications [Sun01, Toh07]. However, even after a decade of research efforts, their ease-of-use is still quite low, making it a technology only accessible for experts.

One of the main reasons hindering a large scale deployment of MANETs is the initial configuration phase [IET03]. For a MANET to be fully operational all nodes must be configured using compatible layer-2 and layer-3 parameters[1]. In particular, if the IEEE 802.11 technology [IEE99] is adopted, there are some basic parameters that all nodes must share. Concerning the layer-2, these parameters are: (i) the operation mode (ad-hoc or infrastructure), (ii) the service set identifier (SSID), (iii) the channel used, (iv) the security protocol used (WEP, WPA, WPA2), and (v) the shared key used for authentication and/or encryption. On the other hand, at layer-3 several parameters must also be set: (i) the IP version used (IPv4 or IPv6), (ii) the node's IP address, (iii) the network mask, (iv) the routing protocol used (e.g., AODV [CES03], OLSR [TP03], DYMO [IC08]), and (v) the gateway to the Internet. Concerning the latter two parameters, notice that routing protocols are essential in MANETs to make multi-hop communication possible; additionally, these protocols usually offer gateway information either automatically or upon user request.

The aforementioned list of parameters evidence the complexity in configuring MANET stations. Indeed, none of the autoconfiguration solutions presented in chapter 4 solve the configuration of both layer-2 and layer-3 parameters simultaneously. Also, since MANETs lack any sort of centralized server to handle configuration, all terminals involved must share this task in a distributed manner. Overall, we consider that there are mainly two barriers preventing distributed node configuration to be effective: (i) on one hand, a wireless link must be established to share all the configuration parameters required to configure the wireless link itself; (ii) on the other hand, the fact that wireless communications are easy to intercept typically requires encryption to be adopted, which further complicates the configuration process if the encryption key itself is one of the configuration parameters required. Notice that, in both cases, we have a variant of the bootstrapping problem, which is typically complex to solve. Up to now this problem has remained mostly untackled by the research community, and no real alternative to manual setup has been found.

In this chapter we propose two methods able to autoconfigure MANET terminals in a very efficient and straightforward manner, setting up all the different parameters associated with the network layers involved in the process (i.e., PHY, MAC and Network layers). The first solution makes the MANET initialization process fully automatic by relying on Bluetooth technology. Bluetooth properties (the concept of services, device/service discovery, secure connection and authentication, low battery consumption and reduced radio range) make it ideal for exchanging private and confidential configuration information in a very simple manner. The second solution assumes that IEEE 802.11 technology is used, and thus relies on the only unencrypted piece of information that a user can modify at layer-2, the SSID, to accomplish the goals set. Since the SSID is embedded into beacon frames, periodically broadcasted by all MANET participants, high efficiency is achieved with no cost in terms of additional network traffic.

[1]Notice that the common physical layer (PHY) parameters (the modulation type, the frequency, and the synchronization timestamp) are typically set automatically by the wireless interface without user intervention.

The rest of the chapter is organized as follows: Section 6.2 describes the first autoconfiguration method proposed, including the system architecture, the design issues and some initial experimental results. In section 6.3 we describe the second solution, offering details about an actual implementation on a GNU/Linux platform, and showing some performance results obtained in a real-life testbed. A comparison between both solutions is then performed in section 6.4. Finally, section 6.5, presents our conclusions along with some guidelines for future work.

6.2 Autoconfiguration through Bluetooth (BlueWi)

The availability of both Wi-Fi and Bluetooth technologies on currently available devices offers the possibility to combine both in order to make the most out of their capabilities. With this purpose, we propose using the characteristics of Bluetooth technology (version 2.0) for automating the configuration and integration process of users that wish to integrate an IEEE 802.11-based MANET by offering a Bluetooth service specifically designed for that purpose. In our work we denote this service as *MANET_Autoconf*. With the presence of this service, we can configure both Layer 2 and Layer 3 parameters automatically. Our setup is one practical way to demonstrate how MANETs can be easily and quickly configured using off-the-shelf devices.

6.2.1 System architecture

Our proposed autoconfiguration solution (BlueWi), requires a device acting as a Bluetooth server, which will be responsible for registering the *MANET_Autoconf* service and sending the appropriate configuration parameters to clients (making sure that all devices are configured with different IP addresses but with the same routing protocol/Wi-Fi parameters). Any node among the set of nodes that would like to build a MANET could act as the server node. The remaining devices will act as Bluetooth clients, which must search for the *MANET_Autoconf* service in the network. Therefore, we assume that all devices have both a Bluetooth and an IEEE 802.11 interface. The Bluetooth interface is merely used to obtain the configuration parameters required to join the MANET, while the Wi-Fi interface will allow the node to participate actively in the MANET for different purposes, such as sharing information with other nodes or participating in any sort of peer-to-peer communication.

Since BlueWi targets short-lived MANETs, the period of activity is limited to a maximum value, typically one day. When a client is configured, we assume that he will be a potential MANET member during that period. So, when this period expires, the server is in charge of configuring a completely new MANET by picking a new *MANET_ID* and, if necessary, applying any possible configuration change. In this way, we avoid possible IP address collisions between previous and new MANET members. Also note that our solution allows pre-defining the maximum number of nodes to be configured in order to check if anyone has failed autoconfiguring its parameters or is missing. In case that some parameters should be changed during the day (which is not common in our envisioned application),

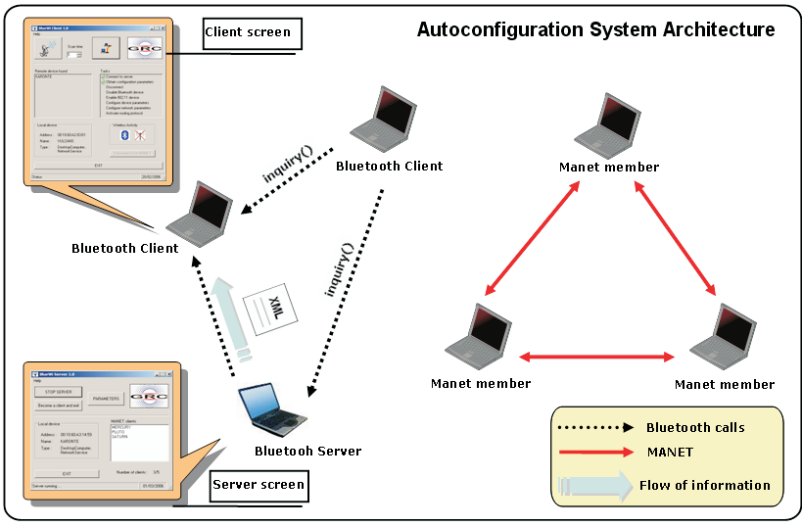

Figure 6.1: System architecture.

such fundamental changes would require all users to be warned, to perform configuration manually, or to restart the whole configuration process again.

Figure 6.1 depicts an example for the proposed system architecture where it is possible to notice, on one hand, three different kinds of scenarios: (a) Bluetooth clients waiting to be configured with suitable parameters, (b) the configuration server waiting for incoming Bluetooth connections, and (c) fully configured MANET nodes. On the other hand, it is possible to observe the coexistence of two networks operating with different wireless technologies: the Bluetooth network (composed by nodes that are linked by discontinuous lines) on the left part of the picture, and the Wi-fi network (composed by nodes that are linked by continuous lines) on the right side of the picture.

Therefore, every user wishing to join the MANET must first connect to the configuration server via Bluetooth (possibly competing with other users that are also waiting to be configured). To connect to the server, clients must perform an *inquiry* action to discover nearby Bluetooth devices. Thereafter, they must scan sequentially the different devices encountered until one that offers the MANET Autoconf service is found; for this endeavor Bluetooth's Service Discovery Protocol (SDP) is used. SDP provides the client with a mechanism for discovering services available on a certain device, along with the service attributes. Notice that SDP is designed to be decoupled from the service itself, and so it does not provide any mechanism or protocol to use these services. Finally, a personalized L2CAP or RFCOMM connection with the server is made to request the configuration parameters. The process of connection establishment is done using the appropriate L2CAP or RFCOMMM port, as advertised through SDP.

The configuration server must make sure it is visible by other devices, so that any device discovery attempt is successful. Besides, it must register the

```
<?xml version="1.0" encoding="UTF-8" ?>
- <BlueWi>
    <!-- MANET Autoconfiguration Service  -->
    <!--      Client-side parameters  -->
  - <Client_device>
      <mode>adhoc</mode>
      <SSID>BT-MANET</SSID>
      <channel>1</channel>
      <encryption>NONE</encryption>
      <encryption-key>NONE</encryption-key>
      <ip-address>192.168.1.2</ip-address>
      <ip-netmask>255.255.255.0</ip-netmask>
      <rt-protocol>OLSR</rt-protocol>
    </Client_device>
  </BlueWi>
```

Figure 6.2: XML configuration file for a client.

MANET_Autoconf service and advertise it so that clients can proceed to discover the service afterwards. The server will also be listening to the appropriate L2CAP or RFCOMM port for incoming connections. Taking into consideration the limitations of Bluetooth technology on the number of nodes in a same piconet, the server can acept up to 7 concurrent clients (note that Scatternet support is not necessary because the nodes' configuration is limited to one hop). If more clients are waiting to be configured, they must wait until one of the clients completes the configuration process and releases the resources occupied. As we will show in section 6.2.3, configuration times are low (typically below 8 seconds), being mainly due to the time for Bluetooth inquiry.

For each client that successfully establishes a connection, the server generates a customized XML file with all the appropriate parameters, i.e., (a) WiFi mode (ad hoc or infrastructure), (b) SSID, (c) channel, (d) type of encryption, (e) encryption key, (f) ip-subnet, (g) ip-netmask, (h) maximum number of MANET members, and (i) the selected routing protocol (note that a routing protocol is still required because communication between MANET members could be multi-hop). After that, it sends the XML configuration file back to the client to allow it to get configured. Note that initially, clients must be in the one-hop Bluetooth communication range of the server to get the XML file. Figure 6.2 shows an example of the XML configuration file. We have chosen XML because it is a standard for encoding the content, semantics and diagrams of a wide variety of applications.

By allowing the server to determine the IP address of each client we are able to solve the problem of IP address assignment referred earlier in a centralized way. It is important to point out that the server must assign IPs in increasing order and must maintain, at any moment, a registry of the connected clients (for example, in another XML file). After a client node receives its configuration data, it must switch to Wi-Fi mode (the Bluetooth interface is disconnected and the Wi-Fi interface is activated) to apply the new configuration parameters. With this technique, minimum interference between Bluetooth and Wi-Fi technologies (IEEE 802.11n and 802.11g) is achieved, a problem that affects devices operating in the 2.4 Ghz band [RRD01]. However, if 802.11a cards are used, this problem ceases

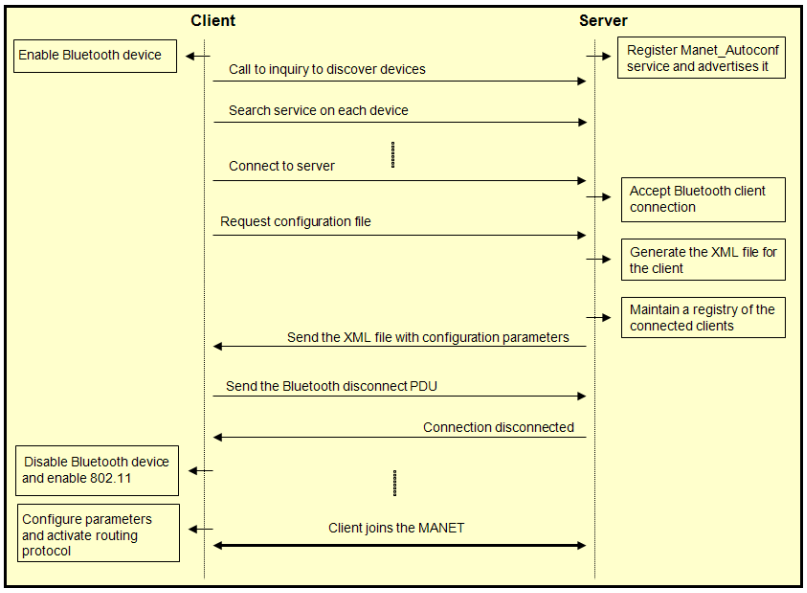

Figure 6.3: Flow diagram between client and server.

to exist (best case scenario). Finally, the client will start the selected routing protocol to make the MANET fully functional. Note that all the configuration process is activated without the need for a client to reboot. Figure 6.3 describes this process, showing the sequence of actions taken according for all the steps referred previously.

The server device may optionally join the MANET. In that case, it can join the MANET right from the beginning (possibly suffering from Bluetooth/Wi-Fi interference), or only after all the expected nodes are configured. In the latter case, it will perform a sequence of actions similar to that performed by clients, disconnecting the Bluetooth interface and enabling Wi-Fi. When this occurs, no more users can be automatically configured since the Bluetooth server no longer exists. For this reason the server always maintains data about the number of configured devices, as well as the maximum number of devices expected.

6.2.2 Design issues

We have implemented the BlueWi application (client and server) for both Windows and Linux operating systems. The Windows-based application uses the Microsoft Bluetooth protocol stack [BPS04]. We chose this Bluetooth stack because it is the one included by default on both standard and mobile Windows editions, and also because there are useful and free libraries that use it. To develop the application we used the .NET platform combined with the Visual Basic programming language, achieving an object oriented solution that is powerful and yet simple. Actual access to the Bluetooth protocol stack is done using the external library 32feet.NET [THL06], whose source code is maintained as a Microsoft open source project.

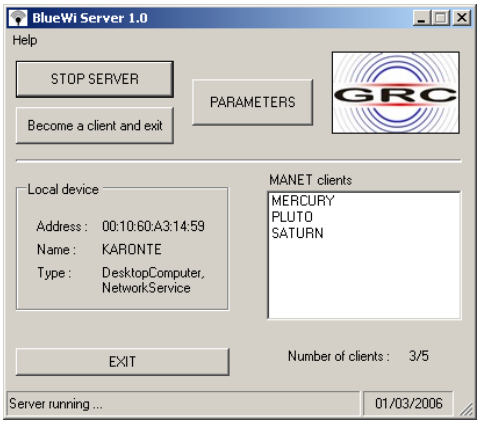

Figure 6.4: Server application capture.

Concerning the Linux-based solution, we have designed both a console and a graphical version. That way it can operate both as a service offered by the system, and as an interactive application. To control the Bluetooth interface we have used the BlueZ API [Kra03], which offers a high degree of functionality. The console solution was written in C, while the graphical application was designed in C++ using QT libraries [QDF09].

Both client and a server applications offer information about the local device, such as the MAC address, the public name and the type of device (laptop, PDA, etc.). Figure 6.4 shows the graphical server interface, depicting a situation where three clients have successfully connected to the server and joined the MANET. This screenshot represents the example depicted in Figure 6.1. We can start and stop the server just by clicking the corresponding button. If the server is stopped, the Bluetooh interface may be active but the *MANET_Autoconf* service is not registered, avoiding any connection attempt. Moreover, we can set the different configuration parameters (see Figure 6.5), which include the desired IP address and subnet, the maximum number of clients allowed to integrate the MANET, the routing protocol used, and the different Wi-Fi parameters. The server application also includes a manager to control the number of devices already configured (being identified either by the MAC address or by the name).

The clients, by using the interface shown in Figure 6.6, can initiate the discovery of Bluetooth devices within range (for version 2 of the standard can reach up to 100 meters); this is done by means of a "find" button, represented by a satellite dish. The discovery time is adjustable by the user as multiples of 1.28 seconds; this value is related to the time a device takes to scan sets of Bluetooth frequencies. The list of devices shown at the interface is restricted to those offering the *MANET_Autoconf* service to simplify the user's tasks. After a successful discovery action the client can proceed to connect with the selected device to retrieve the configuration parameters. A list of all the required tasks is also made visually available by the application; for each successfully completed task, the corresponding element of the list will be checked with a green icon. That way the

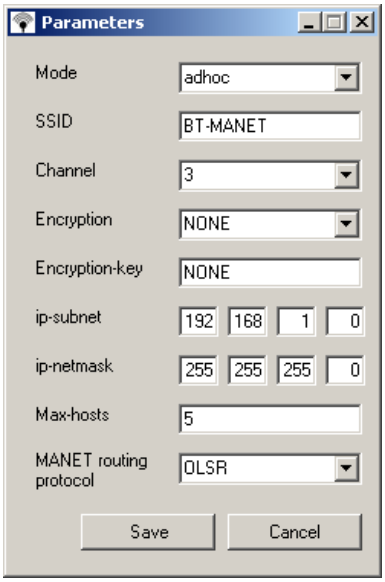

Figure 6.5: Application window for parameters configuration.

client can be aware of the current status and possibly identify errors/problems during the process. Below the task list there are two icons that indicate, at any moment, which wireless interface is active. Finally, two buttons that allow clients to disconnect from the MANET (restarting the configuration process) and to exit the application are also included.

6.2.3 Experimental results

In order to assess the impact of some of the parameters referred earlier, we have conducted a set of experiments to test the performance of our application under different conditions. The main purpose was to measure the times relative to inquiry, service discovery, and download of configuration data. These measurements are very important since they offer an estimate of the time a node takes to get configured and connected to the MANET.

Concerning the inquiry time, it can be adjusted at user's will as multiples of 1.28 seconds (as referred before), allowing to achieve a trade-off between latency and the probability of discovering a device. We measure the duration of that procedure through samples of 100 independent test runs. The results show that the inquiry time takes about 6.42 seconds on average, which means that the operating system introduces an overhead of about 20 ms.

We observe that the whole autoconfiguration process is completed within a matter of seconds (typically under 20 seconds). This time can be considered acceptable for our purposes, especially taking into account that the Bluetooth server is able to handle concurrency. Figure 6.7 shows the experimental results where we measure the time consumed by the Bluetooth inquiry process plus the

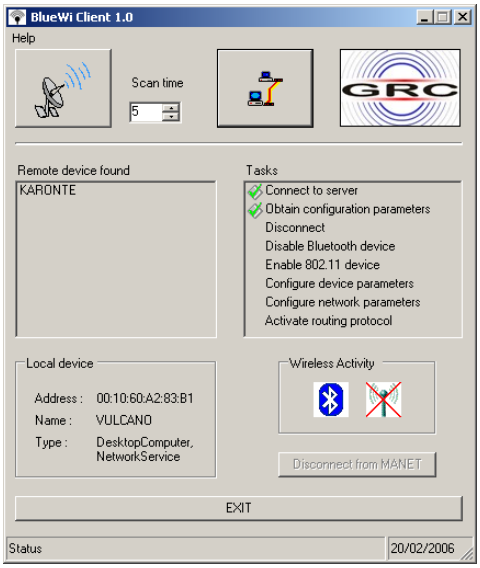

Figure 6.6: Client application capture.

service discovery and, depending on whether the service is found or not, the time for downloading the configuration file (denoted as *With Service* in the figure). Note that "No service" here means that configuration file download does not occur at all, so the measured time refers to the inquiry time plus the service discovery time. In this test only one client and one server are present. As shown, since we are dealing with Bluetooth for our autoconfiguration service, distance between nodes have a great impact on performance. We could describe the configuration wait time experienced by any user at a distance d (WTd) from the server by:

$$WT_d \cong INQ + SD_d + DT_d$$

where INQ refers to the inquiry time, SD to the service discovery time, and DT to the download time of the configuration file in case of the *MANET_ Autoconf* service is found.

Figure 6.8 shows the configuration time as the number of Bluetooth devices in the surrounding environment increases; in this test the Bluetooth inquiry time (standard-defined value of 5.12 seconds) is not included to allow a better visualization of the subsequent processes and the performance penalty of finding a large number of visible Bluetooth devices. Note that, if there is only one nearby Bluetooth device offering the *MANET_ Autoconf* service, then the process completes very quickly, typically in less than 2 seconds. However, if other Bluetooth-equipped nodes are also detected, the expected configuration time will increase linearly with the number of nodes. For this reason, we suggest that all devices attempting to configure themselves through the *MANET_ Autoconf* service should change their Bluetooth settings so as to make themselves invisible to other nodes in terms of inquiry.

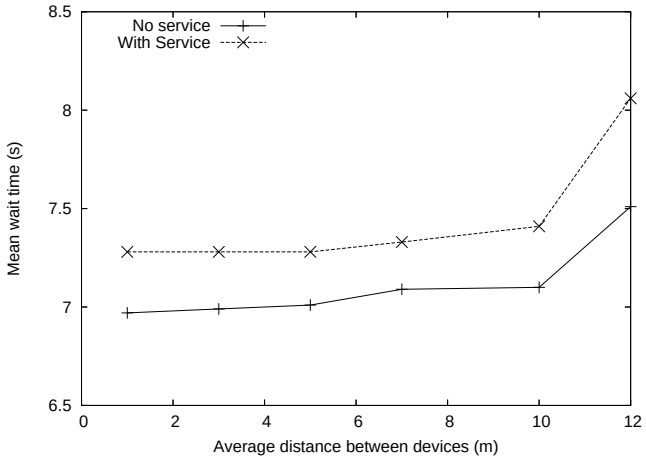

Figure 6.7: Configuration time while varying distance between Bluetooth devices.

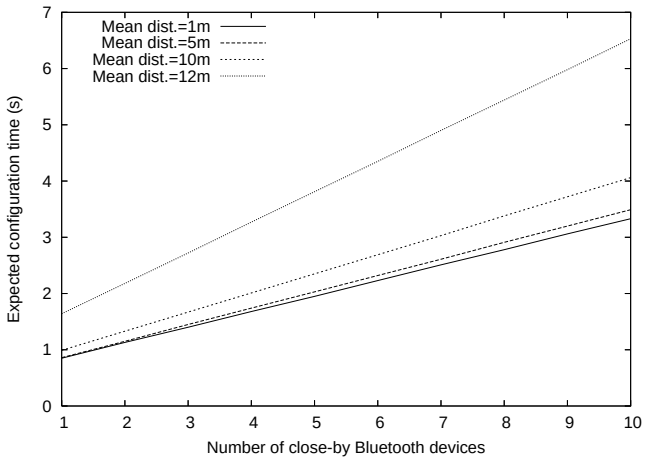

Figure 6.8: Configuration time (inquiry time excluded) when varying the number of close-by Bluetooth devices for different distance values.

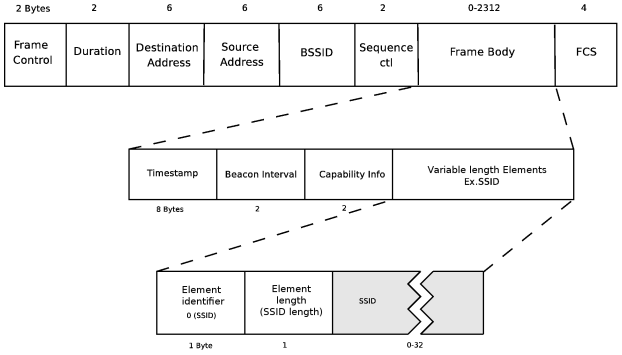

Figure 6.9: Details of an IEEE 802.11 beacon frame.

6.3 Autoconfiguration through 802.11 beaconing

The second envisioned solution for autoconfiguring 802.11-based MANETs is decentralized and takes into consideration that even when communications are encrypted, beacon frames are not. Thus, our second proposal relies on beacon frames as potential carriers of the vital information that allows a station to gain awareness of critical configuration parameters. By analysing the structure of an IEEE 802.11 beacon frame (see Figure 6.9) we may observe that all the frame fields are automatically set by the 802.11 MAC layer without user intervention, except for the SSID field. This field is set by the user and carries the network's name, having a maximum size of 32 bytes according to the 802.11 standard.

The SSID will be used not only to include the network's name, but also to inform stations about configuration details which will allow them to be transparently configured. Such duality is not expected to cause any drawback since most SSIDs in use are characterized by a low byte count. To justify this statement we took a large database including about 8 million samples corresponding to the top 1000 SSIDs used worldwide [WiG01], and then plotted the cumulative distribution function for this 1000 SSID sizes. The result of this analysis is presented in Figure 6.10. We can see that 92% of the SSIDs in use have a length between 4 and 9 characters, being very large sizes (>16) quite scarce and lacking any additional benefits. Thus, we consider that limiting the SSID to a smaller size would not represent any significant limitation, especially when targeting ad-hoc networks where the SSID must be defined every time a new network is created.

Taking the previous analysis into consideration, we now propose a strategy to partition the SSID into four different blocks, as shown in Figure 6.11. The name of the network, that is, the legacy SSID, will be in the first block. The second block includes basic network properties: whether it is an IPv4 or IPv6 network, which encryption mode is used, and also which MANET routing protocol should be running. The third block identifies the subnetwork prefix (also including the network mask when IPv4 is used). Finally, the fourth block holds a random value to be used as seed when attempting to derive the session key used for 802.11 MAC encryption.

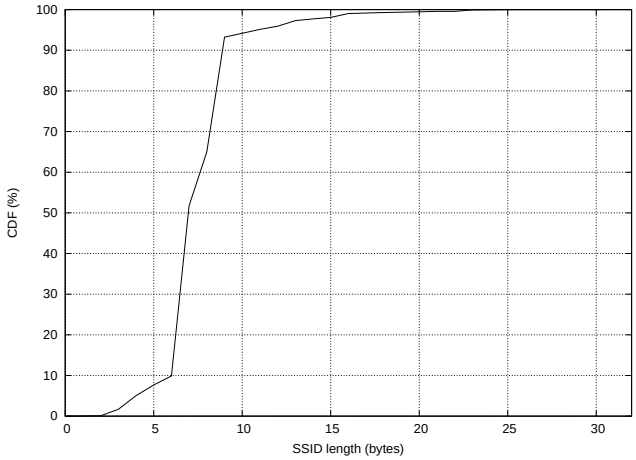

Figure 6.10: Cumulative distribution function for the SSID set analysed.

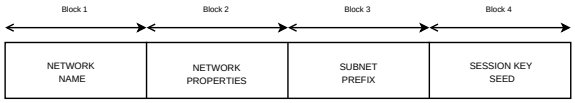

Figure 6.11: Generic SSID partitioning strategy.

The proposed solution requires that users attempting to connect to an autoconfigurable MANET parse the SSID to extract all the information required to be connected to the MANET. All data is retrieved automatically through client software, allowing the client to fully and transparently configure the network connection in terms of both layer 2 and layer 3 parameters. In terms of advantages offered, this configuration strategy is: (i) scalable, (ii) decentralized, (iii) robust, and (iv) efficient. The arguments that support this statement are the following:

- (i) Since beacons are periodically generated at a controlled rate by all MANET stations according to a randomization algorithm, the configuration information is made available to the whole MANET independently of its size, which makes the solution scalable.
- (ii) Any station attempting to join the MANET is able to obtain configuration data just by listening to the beacons from any nearby MANET member, avoiding the need for a central coordinator.
- (iii) As long as a single MANET member remains active, the proposed configuration strategy remains immune to the loss of participating stations, which makes the process robust to failures.
- (iv) The proposed strategy does not generate additional network traffic, and allows achieving full node configuration in a short period of time (see section 6.3.2, thus offering high efficiency.

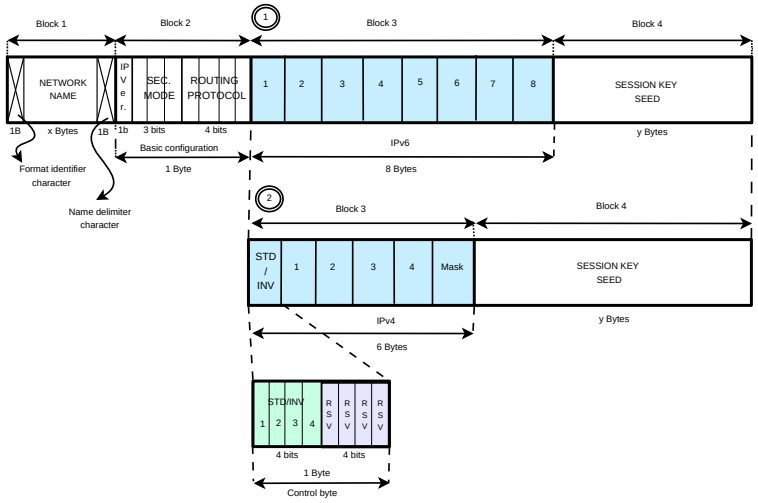

Figure 6.12: Detailed SSID partitioning strategy.

6.3.1 Implementation details

In this section we describe how the different parameters required for configuration are embedded into the SSID string and later parsed. We will also offer details of an actual implementation of our approach in a Linux-based testbed. The support for 802.11 wireless cards in the Linux operating system has been available since the late nineties through the Wireless Extensions API [Tou96] developed by Jean Tourrilhes, along with a set of wireless tools [Tou08] accessible through the command line that were developed by the same author.

For our endeavour, we also developed some command-line test applications that allow the user to join an existing MANET with autoconfiguration support, as well as starting such a MANET. Notice that the latter option implies the user to define the value of the different parameters required, as we stated in the introduction section, in order to fully configure a network interface card, which includes both layer-2 and layer-3 setup.

6.3.1.1 Proposed SSID partitioning strategy

Figure 6.12 illustrates the proposed SSID partitioning strategy, which has been implemented and validated in a real testbed. Notice that, in order to distinguish regular beacons from our formatted beacons, a special (non-printable) character has been inserted just at the beginning of the SSID block, thus allowing to quickly identify SSIDs formatted according to our proposal. As we see, the network name (or SSID) appears followed by another non-printable character that ends block 1. Block 2 is composed by a single byte where the first bit indicates which IP version is in use (IPv4 or IPv6), followed by 3 bits that indicate which 802.11 security mode is active (0=open access, 1=WEP-64, 2=WEP-128, 3=WPA-PSK, 4=WPA2-PSK, 5-7=reserved for future extensions); concerning the last 4 bits in block 2, they are

used to identify the MANET routing protocol used (0=forbidden value, 1=OLSR, 2=AODV, 3=DYMO, 4=DSR, 5-15=reserved for future extensions). Notice that value 0 is forbidden to avoid a situation where this single byte block is set to the NULL value, which would be considered as an *end of string* character by the operating system, thus causing and error.

Depending on which IP version was defined in block 2, block 3 will contain an IPv6 network address field (8 bytes) or an IPv4 network address field. In case IPv6 is used, these 8 bytes represent the first half of the address within the Unique Local Unicast [HH05] range of addresses (FC00::/7); the latter 8 bytes (Interface ID) are derived from the MAC address of the wireless network interface according to the strategy defined in RFC 4291 [HD06]. If IPv4 is used instead, we identify the network using 4 bytes plus an extra byte to set the network mask. To avoid those situations where one or more bytes are zero (NULL character), we use the first byte (STD/INV) to invert possible NULL values in any of the four bytes that define the IP address, thus converting any 0x00 value into 0xFF. Stations attempting to configure themselves must reverse the inverted bytes to recover the original values. Notice that this strategy was not required for IPv6 since we rely on the Unique Local Unicast range of addresses, which allows picking any value for the 7 bytes following the first one, meaning that we can easily discard any 0x00 values appearing and pick other values instead.

Concerning the last block, it includes the session key seed, which is used to derive the actual session key that will allow performing MAC layer encryption.

6.3.1.2 Deriving the session key

When a new MANET is generated, the value for the session key seed is picked randomly. This seed allows deriving the session key by supposing that all users are aware of a fixed pre-shared key (PSK). When relying on the 802.11 standard this shared key is used directly for MAC layer encryption; however, with our solution, this shared key is replaced by a variable session key. This strategy complicates the discovery of the MAC layer encryption key by a potential attacker by making it different every time a new ad hoc network is created.

The size of the seed itself is variable, and depends on the number of bytes used to identify the network (x). In our solution we will restrict the size of this network identifier to a maximum of 10 characters, which is not considered a restriction since 10 characters are enough to uniquely identify an ad-hoc network in any plausible scenario. Once the network identifier is defined, the size of the session key seed is picked so as to fill up the SSID size, thus reaching the maximum length for the SSID field. Although in theory there are 32 bytes available for the SSID, the fact that the operating systems handle it as a string ending with the NULL character reduces it to 31 bytes. This way the size of the session key seed will be either 20-x or 22-x bytes (10 bytes in the worst case), depending on whether IPv6 or IPv4 is used, respectively.

One of the limitations of having the seed embedded into the SSID has to do with the handling of NULL values, as mentioned above. This means that the number of possible combinations will be slightly reduced by this restriction. Thus, the original space of $256^{(20-x)}$ combinations ($256^{(22-x)}$ for IPv4) is reduced to

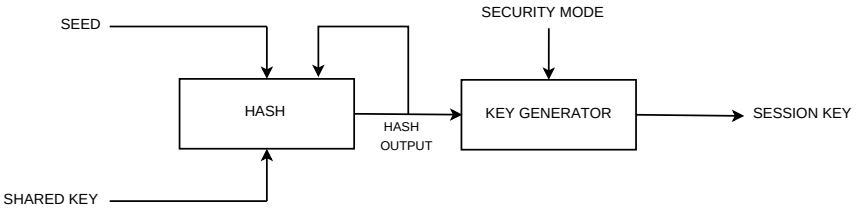

Figure 6.13: Session key generation process.

$255^{(20-x)}$ ($255^{(22-x)}$ for IPv4). In the worst case conditions (if the network name uses all 10 characters) there are still about 10^{24} possible seeds ($\sim 7 \times 10^{28}$ for IPv4); this means that the chances that the same seed repeats for a same group of users becomes negligible.

Figure 6.13 offers more details about the process of session key generation. Initially, the key shared by all MANET users is combined with the seed made available in the SSID (block 4) by using a hash mechanism such as MD5 [Riv92], SHA-1 [DP01], or RIPMED-160 [HAB96]. Since the number of bits in the hash may be shorter than the one required by the selected security mode, the hash output is fed back to generate a new hash until the key generator module gathers enough bits. Depending on the security mode selected, the key generator module may have to chop part of the input in order to obtain the correct number of bits for encryption. For example, if MD5 is used for hashing and WEP-64 is used for encrypting, a single hashing round suffices since the 128 bit output is enough to obtain the 40 bits required for a valid session key. On the contrary, if WPA-PSK is used, we need two MD5 hashing rounds to generate a session key of 256 bits, as required.

6.3.1.3 MANET setup

In our scheme, we suppose that there is a group of users that regularly creates and joins an ad-hoc network with a specific goal. Returning to the example of a firefighting unit, we consider that on every mission the same group of firemen creates a MANET for communicating among themselves. Furthermore, all users will use the same key for accessing the network, referred to as *shared key* in the previous section.

When these users intend to join the same ad-hoc network using our proposed solution, one of them (e.g., the head of the fire squad) creates the ad-hoc network, defining all the parameters required; among them we have the network name, the security mode, the routing protocol used, the IP addressing information and the *seed* used for session key generation. When the terminals used by the other users listen to the beacons generated, they will immediately parse the SSID field to retrieve the configuration details and successfully attach themselves to the MANET. This means that first there will be a layer-2 connection establishment in order to become a member of the Independent Basic Service Set (IBSS) created, followed by IP parameter definitions and the launching of the appropriate MANET routing daemon.

Table 6.1: Average time overhead associated with autoconfiguration tasks.

Autoconguration tasks	*No security*	*WEP (64 & 128 bits)*	*WPA (256 bits)*
Obtaining configuration	4 / 20 µs	4 / 20 µs	4 / 20 µs
Generating key	-	23 µs	46 µs
Establishing SSID	4,400 µs	4,400 µs	4,400 µs
Applying security key	-	200 µs	17,500 µs
Setting IP address	4,000 µs	4,000 µs	4,000 µs
Starting routing protocol	5,600 µs	5,600 µs	5,600 µs
Total time	*∼14 ms*	*∼14.2 ms*	*∼31.5 ms*

From that point on, any subsequent MANET generated would be quite similar, except that the seed used will always differ, and thus also the session key. This requires a potential attacker to find the session key used, and to launch the attack within the lifetime of a specific ad-hoc network (i.e., for a specific seed). Compared to the default solution, where the session key would be used over and over again, this strategy significantly reduces the effectiveness and interest at performing malicious activities.

6.3.2 Validation and performance analysis

In the previous section we described the implementation details of the proposed autoconfiguration system, which includes two components: one that allows creating the ad-hoc network, executed only by the first station, and another that allows joining an existing autoconfigurable MANET. In this section we present some performance results obtained when validating our solution in an ad-hoc network testbed.

Performance measurements were made using five middle-range laptops with similar hardware, running at 1.6 GHz with a single CPU and with 1 GB of RAM. The wireless cards used were Intel embedded devices supporting the IEEE 802.11g standard, and all terminals are within transmission range of the terminal that initially creates the ad-hoc network, unless stated otherwise.

6.3.2.1 Assessing the overhead introduced per task

Table 6.1 shows the average time overhead results obtained for the different security strategies. We do not include WPA2 encryption since it is not yet supported by the Linux OS in ad-hoc mode. With respect to the first task, the MANET creation component requires parsing the user's input and generating an SSID string with autoconfiguration information (according to the strategy shown in Figure 6.12), while the autoconfiguration component must merely parse the beacon received to extract configuration information. Thus, while the latter task is achieved in just 4 µs, the former (SSID generation) requires 20 µs.

The remaining tasks are similar for both components developed. In particular, the second task is related to key generation, which is achieved according to the strategy shown in Figure 6.13; obviously, this step is skipped if security is disabled. The third and fourth tasks consist in setting the layer-2 parameters, such

Figure 6.14: Chain topology used to measure the propagation time for autoconfiguration beacons.

as the SSID and the encryption key. In case WPA is used, a configuration file must be created before launching the *wpa_ supplicant* tool, which is responsible for WPA/WPA2 configuration tasks in Linux. This causes the time associated to that task to account for more than half of the total configuration time. The last two tasks (IP definition and launching the MANET routing daemon) are related to network layer configuration, being common in all cases.

Overall, the autoconfiguration times can be considered quite low, although we have to take into account that the measurements presented in this section refer to the tasks taking place at the application layer. Since the dissemination of configuration information requires beacons to be received, and autoconfiguration tasks to be completed, prior to start generating new beacons, the total autoconfiguration time is usually higher. In the next section we will focus on these issues.

6.3.2.2 Autoconfiguration times in a multi-hop environment

When attempting to autoconfigure different stations in a wireless multi-hop environment, two different issues must be taken into account: (i) there is a delay from the time the first station creates the MANET to the time nearby stations are able to receive the first beacon with autoconfiguration data embedded into the SSID; and (ii) when a new station wants to join the MANET, it must scan the different channels for beacons containing autoconfiguration data. Since, in the ad-hoc mode, the beacon generation process is distributed and follows a random algorithm, the actual time required to detect the beacons may vary.

To study the multi-hop propagation behaviour of autoconfiguration data, we devised a scenario (see Figure 6.14) where nodes are arranged according to a chain topology. Distances between nodes are high enough to ensure that radio communications are only possible with one-hop neighbours. In our setting, Station 1 is responsible for starting the MANET. So, at the beginning of our experiment, Station 1 uses the autoconfiguration application to create a new ad-hoc network, while Stations 2 to 6 attempt to connect to the existing network by starting the autoconfiguration application in the *join* mode.

Figure 6.15 shows how the autoconfiguration information propagates at multiple hops. We can see that Station 2 gets configured in about 2 seconds since it is very close to the station that initiates the ad-hoc network. In particular, most of this time is associated with detecting the beacon, being configuration parameters applied in just a few milliseconds, as shown earlier (see Table 6.1).

As we increase the number of hops, it would be desirable to experience a linear increase of this autoconfiguration time, being such linear increase represented as *best case* in Figure 6.15. However, experimental results show that the average

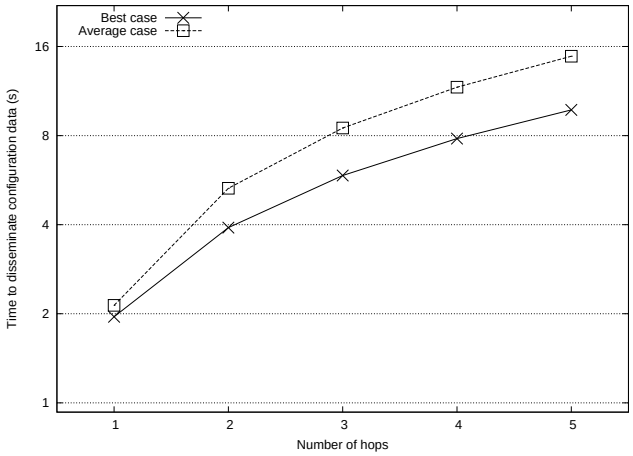

Figure 6.15: Autoconfiguration times for nodes at multiple hop distances from the initiating node.

propagation times are associated with a more than proportional increase, which is explained by the random beacon generation process. Remember that the IEEE 802.11 standard establishes that, in the ad-hoc network operation mode, beacons are generated by all stations involved in a distributed fashion by following a randomization algorithm. Thus, in our example, Station 2 would only generate a beacon about half the times, while Station 1 would generate beacons in the other half of the cases. As more stations get involved, the chances that a particular station generates a beacon become smaller, which slows down beacon dissemination. In our scenario, we find that Station 6 (at 5 hops from Station 1) must wait on average 14.8 seconds to detect the first autoconfiguration beacon. As a final remark, we should emphasize that such beacon propagation times are the typical times for multi-hop ad-hoc network environments, being that our solution does not impose a significant additional delay to the process.

In terms of scalability, we consider that our solution is scalable by design since the configuration information data propagates at the beacon propagation rate, which becomes highly effective even in large-sized and highly disperse MANET environments.

Concerning new nodes trying to join the MANET, they can initiate the configuration process as soon as the ad-hoc network is detected (after any periodic beacon is received), usually waiting for only a few seconds on average. In this context, we also measure the delay introduced by routing protocols, that is, the routing topology dissemination time. Notice that, once a station becomes configured and connected, there will be an additional delay introduced by the routing protocol to update the network topology. In our testbed, the routing protocol adopted was OLSR, using the standard parameter values defined in [TP03]. Thus, we performed a second group of tests where we measured the time elapsed from the instant when the autoconfiguration application completes its tasks, until a valid

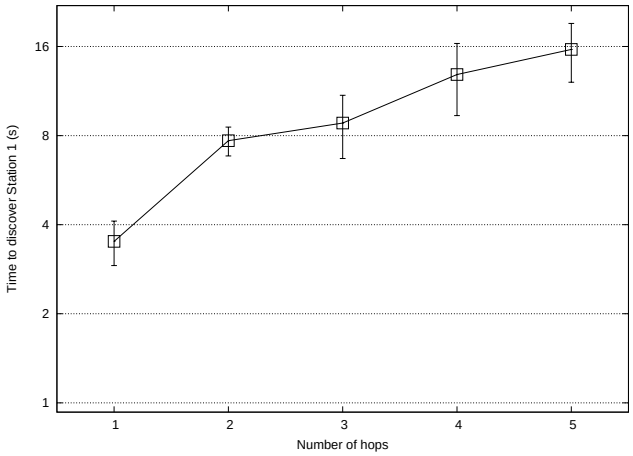

Figure 6.16: OLSR topology updating time when joining the network at different hop counts from Station 1.

route to the first node in the chain topology becomes available. Notice that, after the beacon is detected, each station will apply the autoconfiguration parameters, which also includes launching the OLSR protocol daemon.

In the following, all the previous nodes in the chain are configured and connected from a routing perspective when the new station arrives. Figure 6.16 presents our experimental results assuming that Stations 2 to 6 will gradually join the network, creating the topology shown in Figure 6.14. The values represented in Figure 6.16 show that routing information dissemination with OLSR imposes a significant time overhead, especially at more than one hop. This is expected since, in the scope of OLSR, communication with one-hop neighbours only requires neighbour detection procedures, while higher hop counts require topology updating procedures to be triggered. Thus, when Station 2 attempts to join the network and contact Station 1, OLSR takes between 3 and 4 seconds to provide a route to this station, while for Station 6 it will take OLSR between 12 and 19 seconds to provide a valid route.

By combining the results of Figures 6.15 and 6.16, we find that the time required for a MANET to be fully connected and operational can be reduced to less than one minute if the proposed autoconfiguration strategy is adopted, even with stations located several hops away from the station starting the MANET, and even when using a routing protocol with a relatively slow responsiveness (e.g. OLSR).

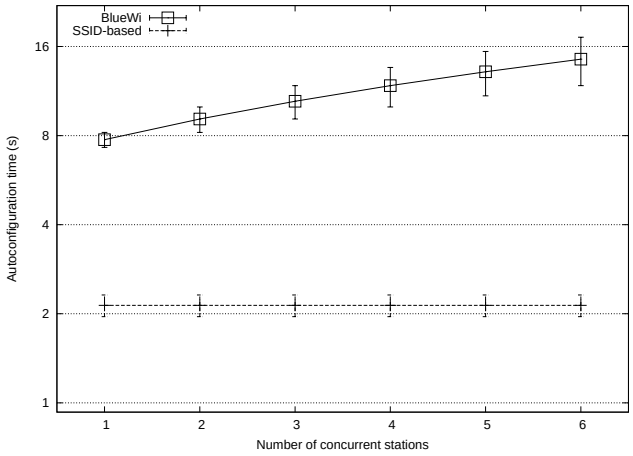

Figure 6.17: Autoconfiguration time when varying the number of terminals being configured at one hop.

6.4 Comparison between both solutions

In this section we perform a comparison between the two solutions presented. As referred before, BlueWi requires clients to establish a Bluetooth channel with a BlueWi server to retrieve all the configuration parameters required. The Wi-Fi interface is then configured according to that parameter set. The second proposal significantly differs from BlueWi since it does not assume any sort of server. In fact, any station can start the ad-hoc network, and, as long as a single station keeps that network alive, other stations can autoconfigure themselves and join the network.

Figure 6.17 shows the total autoconfiguration time when attempting to simultaneously configure different numbers of terminals. All terminals are assumed to be at one hop from either the Bluetooth server (BlueWi), or the station that starts the MANET (SSID-based proposal) for comparison. In terms of radio range, the BlueWi solution limits the maximum distance between the Bluetooth server and the stations being configured to 10 meters (default Bluetooth range) or 100 meters, depending on the Bluetooth device class. For the SSID-based solution, stations at one hop from the station starting the MANET are able to detect its beacons for distances up to 250 meters although, as shown in the previous section, multi-hop configuration is possible and does not suppose any impediment.

From Figure 6.17 we can see how the autoconfiguration time for the SSID-based solution is independent of the number of stations involved. This is expected since beacons are broadcasted, being received by all wireless devices within range. Concerning BlueWi, we find that autoconfiguration tasks require several seconds more. This additional time is mostly associated with Bluetooth device discovery procedures (*Inquiry*), which take about 5.12 seconds to complete, and that are a prerequisite before attempting to contact the Bluetooth server. Also, the number

Table 6.2: Comparison between BlueWi and the SSID-based autoconfiguration techniques.

Characteristic	***BlueWi***	***SSID-based***
Configuration strategy	Centralized	Distributed
Autoconfiguration server	Required	Not required
Multi-hop configuration dissemination	Not supported	Supported
Wireless technologies	Wi-Fi, Bluetooth	Wi-Fi
Number of simultaneous users serviced	7	No limit
IPv4 support	Yes (DHCP-like)	Yes (MAC address based)
IPv6 support	No	Yes (MAC address based)
WEP/WPA/WPA2 support	Yes	Yes
Rotating encryption keys	Yes (manually)	Yes (random seed)
Routing protocol	Any	Any
User control and logging	Yes	No
User access control	Bluetooth pin	Pre-shared Key
Best-case configuration time	7.31 s	1.95 s

of concurrent stations retrieving configuration parameters will reduce the channel capacity dedicated to each station, thereby increasing the total time involved.

To complete the comparison study, table 6.2 summarizes the main differences between both solutions. Overall, we find that the solution based on SSID parameter embedding offers significant improvements over BlueWi, although the later offers more control over the stations joining the network.

6.5 Summary

The proliferation of wireless computing devices is due to their low cost and high flexibility, reason why these count with an increasing number of supporters worldwide. However, the issue of MANET usability is still an important research topic since the complexity when attempting to configure MANET terminals remains high. Besides complexity itself, other issues such as the need to rely on encrypted communications further complicate the configuration problem.

In this chapter we propose two effective solutions to stimulate the use of MANETs by making the process of node configuration fully automatic, allowing clients to join a MANET in a simple and intuitive manner. The solutions configure both layer-2 and layer-3 parameters that are critical to join an 802.11-based MANET. The first solution relies on Bluetooth technology and makes use of Bluetooth services and device/service discovery. The second solution relies on the SSID field that is present on the periodic beacons generated by IEEE 802.11 compliant stations to announce basic configuration data. By listening to beacons and parsing the SSID field, new stations are able to determine all the information required to successfully join the MANET.

To validate our proposals we developed software components for creating and joining autoconfigurable MANETs. We evaluated both solutions and compared them in order to observe strong and weak points of each one, showing that the

SSID-based strategy offers significant benefits and improvements with respect to BlueWi. As future work we plan to develop (for both solutions) a similar set of tools to other devices and operating systems, thus embracing a greater number of potential users.

In the following chapter we propose a compact solution for the next two proposed problems in MANETs environments, that is:

- (i) The name-to-address translation: important for identifying MANET members.
- (ii) Service discovery: important for knowing the possible interactions with other MANET members.

Chapter 7

EasyMANET: An Extensible and Configurable Platform for Service Provisioning in MANET Environments

In chapter 6, different types of strategies for autoconfiguring MANETs efficiently were shown. Our goal now is to address solutions for the problems of name-to-address translation and service discovery in MANETs. In order to accomplish this goal, in this chapter we introduce EasyMANET, an extensible and configurable platform whose main objective is to encourage the widespread adoption and use of MANETs by non-expert users. EasyMANET makes use, as mandatory components, of an address autoconfiguration system (such as the ones seen in the previous chapter), and a name resolution service known as Visual DNS. Visual DNS offers a graphical view of the MANET participants and gives users the possibility to access the services made available by other users; therefore, Visual DNS solves the name-to-address translation and the service discovery challenges for MANETs. Examples of possible services are communication services (text-based chat, VoIP, videoconference), file sharing, and localization services. The rest of this chapter is organized as follows: Section 7.1 discusses the EasyMANET platform, focusing mainly on the name resolution service. In Section 7.2, we evaluate the EasyMANET platform analyzing the Visual DNS performance from both real testbed and simulation points of view. Finally, Section 7.3 concludes the chapter.

7.1 The EasyMANET Platform

This section describes the functionality of the EasyMANET platform, which was developed using Java 2 SE and tested on both Windows and Linux operating systems. As commented previously, EasyMANET has two basic components: (a) the address autoconfiguration manager, and (b) the Visual DNS service.

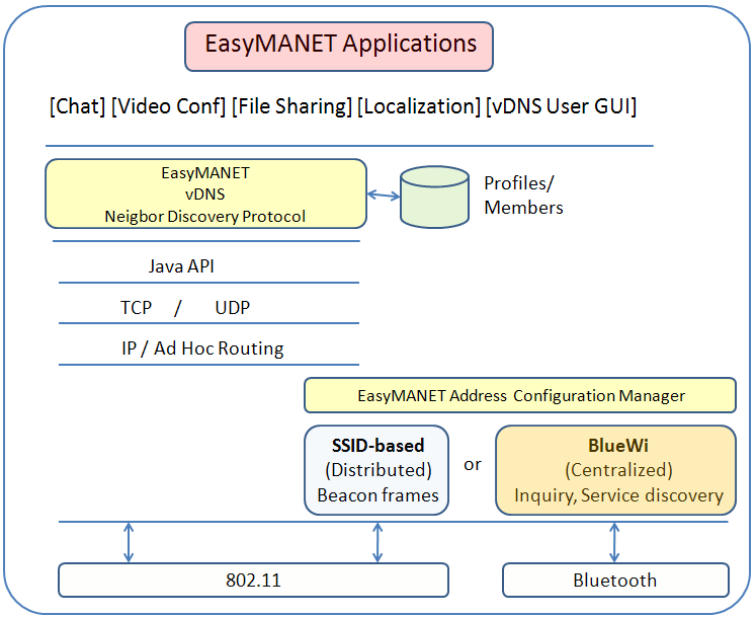

Figure 7.1: EasyMANET protocol architecture.

The address autoconfiguration manager sets up all the basic network parameters for each user who wants to be part of the MANET, allowing them to join the network in a transparent way. EasyMANET considers only isolated MANETs, where IP addresses are assigned to nodes for a short period of time, typically a day. Therefore, if a node leaves the network and joins it again subsequently, it is not necessary to assign a new IP address to the node because it can keep the previous one.

Once a user is a network member, it is useful to know the identities and characteristics of other members prior to communicating with them. In order to achieve this, the Visual DNS service relies on a neighbor discovery protocol, and offers to each MANET member a comprehensive overview of the network participants through a visual identification (relating users' photos and names to IP addresses). Moreover, for each discovered member, the list of available services is shown. Therefore, the problems of name-to-address translation and service discovery are solved by EasyMANET through the same component, Visual DNS.

Figure 7.1 shows our proposed EasyMANET protocol architecture. The EasyMANET address configuration manager can implement any of the two autoconfiguration systems presented in chapter 6 (the SSID-based scheme using 802.11 beacon frames or the BlueWi scheme using the *Bluetooth Inquiry* and *Service Discovery* primitives), but it is also possible to use any other autoconfiguration system. The Visual DNS neighbor discovery protocol, however, communicates over the WiFi interface and is implemented using Java, being located above the TCP/UDP layer. The following sections will elaborate on the components of EasyMANET.

7.1.1 Configuration of Basic Network Parameters

As stated previously, EasyMANET allows choosing between different autoconfiguration methods. Anyway, the autoconfiguration solution targets short-lived MANETs, where the activity period is limited to a maximum value, typically one day. When a client is configured, we assume that he will be a potential MANET member during that period. So, when this period expires, the server is in charge of configuring a completely new MANET by picking a new MANET ID and, if necessary, applying any possible configuration change. In this way, we avoid possible IP address collisions between previous and new MANET members. Also note that using BlueWi allows pre-defining the maximum number of nodes to be configured in order to check if anyone has failed the autoconfiguration process or is missing. In case that some parameters should be changed during the day (which is not common in our envisioned application), such fundamental changes would require all users to be warned about it, to perform configuration manually, or to restart the whole configuration process again.

7.1.2 The Visual DNS Service

The practical use of a name as a more user-friendly abstraction of a machine's numerical address on a network is quite old. In fact, the basic functionality of the DNS (Domain Name System) service consists of translating hostnames into IP addresses. The Visual DNS service is an extension to the DNS service that meets the requirements of MANET environments. Visual DNS allows any node in the network to discover any other node in a transparent manner, and independently of its location. Visual DNS makes use of a distributed neighbor discovery protocol which is based on reliable data dissemination to share information about users. Information shared includes the user name and photo, along with the host's IP address and the available services.

7.1.2.1 Data dissemination strategies

Depending on the scope and purpose, data dissemination in mobile ad-hoc networks is made through a variety of techniques. Finding the optimal dissemination strategy for each purpose is extremely important to improve performance in terms of overhead, delay, and efficiency. We propose a basic taxonomy to discriminate among dissemination techniques that takes into account the amount of information to be disseminated and the number of target nodes. Hence, four categories for data dissemination are devised (see table 7.1).

For category I, partial information is distributed to a subset of the nodes in the MANET. Examples of applications/protocols that fit into this category are peer-to-peer data sharing and MPR diffusion in OLSR [TP03]. Category II applications and protocols must send partial information to all nodes. This is usually achieved through MANETcasting techniques (broadcast propagation in MANETs), widely used by reactive routing protocols such as AODV [CES03], DSR [DDY04] and DYMO [IC08]. An example of a category III application/protocol is neighborhood information exchange in OLSR, where only nearby nodes are able to receive the

Table 7.1: Taxonomy for data dissemination in MANETs.

Data\Target	Subset of nodes	All nodes
Partial information	Category I	Category II
Complete information	Category III	Category IV

information, but the offered information about neighborhood is complete. Finally, for category IV, all nodes must receive all the information available in the scope of a service or protocol. Notice that this last category involves greater overhead than all the previous ones. In wired environments, routing protocols such as OSPF [Moy98] fit into this category since they disseminate individual routing information to all routers. In MANET environments all routing protocols avoid such technique to reduce the overhead involved. Hence, pieces of software with category IV requirements are scarce.

Visual DNS is an example of a category IV dissemination technique, since all the information is expected to be disseminated to all MANET nodes. Our application has no real-time constrains, meaning that information can be disseminated at a slow pace. Yet, a relatively large amount of information must be disseminated reliably. In order to avoid consuming excessive network resources, we propose a trade-off between delay and network overhead by avoiding communication with nodes more than one hop away. Therefore, we propose using a slow data dissemination technique based on the combined use of TCP and UDP functionality, where available information is announced through UDP, and then reliably exchanged through TCP only among neighbour nodes.

With respect to other available protocols offering similar services, such as Multicast DNS [SM06], our proposal has the advantage of including additional information besides host name to IP bindings. Notice that, in a MANET environment, users are usually not aware of the names of the machines participating on it, and so a visual identification of the user itself is perhaps the most effective option available. In the following subsections we offer more details about the Visual DNS discovery protocol.

7.1.2.2 Discovery protocol

Visual DNS must run over a pre-configured 802.11-based MANET (e.g. using one of the approaches mentioned in chapter refcha:Autoconf). Its main purpose is to offer MANET members a visual identification of other members. In particular, it offers their photos, names, and IP addresses along with a list of available services. In MANET environments, users are usually unaware of the names of other participating users, and so a visual identification of the user is useful. Figure 7.2 presents the graphical interface of Visual DNS, including information about the user and other MANET members. Note that anonymity issues are outside the scope of this work. When a user selects any of the discovered members (by clicking on his photo), communication with that member is started.

Figure 7.2: Screen-shot of the Visual DNS interface.

To make the service operative, users that start the EasyMANET application must fill-in their personal information, including name and photo. To discover other MANET members, Visual DNS relies on a discovery protocol. The information gathered by this protocol will be used to update the graphical interface of Visual DNS. Since EasyMANET has no real-time constraints, information can be disseminated at a moderate pace. Yet, information must be disseminated reliably to all nodes. Users first wait to obtain a global view of the participants involved, and later, users can proceed to check if connectivity with other users is possible.

The Visual DNS neighbor discovery protocol works as follows: each node will periodically broadcast an UDP packet (including a list with the IP addresses of all the known members in the MANET, along with a sequence number for each) to its one-hop neighbors. When nodes receive any of those UDP messages, they will compare the list of advertised members in the packet with their own list to find out if updated information or information about new members is available. If new members are detected, the node will open a TCP connection towards the neighbor that advertised it to request further information (user's name, photo, IP address, and available services). Note that, when a MANET member updates its list of services, it increases its own sequence number to notify neighbors that new information is available. Such information will then be disseminated throughout the MANET according to the procedure described.

Figure 7.3 shows a scenario which illustrates how the Visual DNS protocol behaves. In our example, although the node identified by IP address 192.168.1.3 does not receive the UDP packets broadcasted by nodes 192.168.1.18,22 because they are not one-hop neighbors, it will eventually receive an UDP packet from node 192.168.1.38 advertising information availability concerning these two nodes.

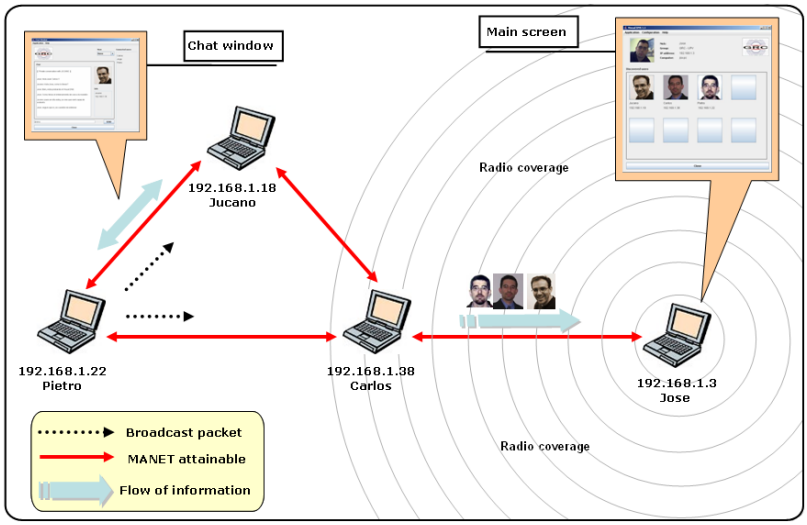

Figure 7.3: User profiles exchange with Visual DNS.

Upon receiving this UDP packet, node 192.168.1.3 will request service information about these new nodes to node 192.168.1.38 using a reliable TCP connection. This example shows that, although the variable topology of the network may impede a node from directly communicating with some nodes in the MANET, it will eventually receive information about unreachable ones, hence being able to keep track of all members.

The period of time between consecutive UDP broadcast messages is a configurable parameter which could impact the efficiency and effectiveness of the protocol. However, since our approach limits broadcast communication to direct neighbors, we avoid the broadcast storm problem [NTCS99], typical of MANETs, and caused by multiple re-propagations of broadcasted messages. Notice that although the Visual DNS discovery protocol limits their communications to one-hop neighbors, communications between any pair of MANET nodes can be multi-hop, as long as permitted by the underlying network topology and the routing protocol.

Algorithm 2 shows the pseudo code for the Visual DNS neighbor discovery protocol. From Algorithm 2, we see that the discovery protocol was implemented using three independent and simultaneous threads:

- A thread in charge of periodically advertising to one-hop neighbors the list of discovered nodes.
- A thread in charge of permanently listening and filtering information advertised by other neighbors, opening a TCP connection to download new information when available.
- A thread for the TCP server socket, which is responsible for handling all the incoming service information requests.

Algorithm 2 Visual DNS neighbor discovery protocol.

```
new thread SendUdpPacket
new thread OnReceivingaBroadcast

SendUdpPacket ()
    while true
        send known members list to 1-hop neighbors
        wait(period)

OnReceivingaBroadcast ()
    for (every received message) do
        Detect new/updated entries in the members list
        if (New information is available) then
                new thread ObtainInformation(member)
                Update my members list
                Update Visual DNS' GUI
        else
                Discard the packet

ObtainInformation (member)
    Create TCP connection
    Receive Information
```

Note that, since any participant can provide information about any other participant, the actual choice of which participant to download from is statistical, and dependent on the timing of HELLO messages. With respect to multicast, our proposal does not require creating and maintaining any topology or routing information. Additionally, since data is transferred using TCP, it is reliable and done at high speeds.

Eventually, all the nodes will update their Visual DNS interface as shown in Figure 7.3. By clicking on each of the nodes found, the user can access the list of available services, and make use of the desired one(s). Figure 7.4 shows an example of multi-service support in EasyMANET. Using Visual DNS, the EasyMANET platform can easily be extended to support: (a) videoconference, (b) chat, (c) file sharing and (d) localization services, in a way similar to other commercial applications, i.e., MSN or Yahoo messenger, but in the scope of a MANET.

7.2 Performance Evaluation

In this section, we first present some initial experimental results conducted in a real test-bed, which allowed us to validate the EasyMANET platform while checking and adjusting some of its parameters. Then, we will see the results of large-scale evaluation experiments on EasyMANET using the OMNeT++ network simulator. In the simulation part, an initial comparison between real and simulated results is included in first term (considering a very simple example scenario).

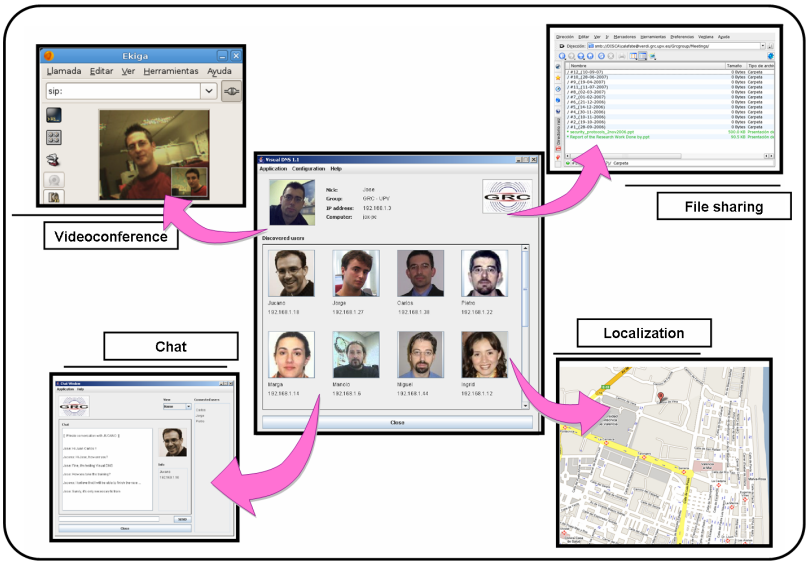

Figure 7.4: Multi-service support in the EasyMANET platform.

7.2.1 Real testbed

Concerning the real experiments, we present some performance results for Visual DNS. Our purpose was to assess the overhead imposed by the proposed data dissemination strategy, that is, the experiments focus on the process of exchanging user profiles conducted by the Visual DNS neighbor discovery protocol.

The first experiment focuses on the transference of user data. With Visual DNS, the set of descriptive information for each user usually occupies between 2.5 and 3.5 kilobytes. Figure 7.5 shows the estimated and worst case values for the data transference time when varying the amount of information about new users. The values shown apply to the IEEE 802.11g technology, and worst case results are calculated for a physical data rate of 1 Mbit/s. As shown in Figure 7.5, when transferring information about a single user the total time is typically less than 30 ms. When transferring information about many new users the total transference time is usually below 300ms, a very low value, approaching 2 seconds in worst case conditions. Notice, however, that massive data transfers are unfrequent, only applying to nodes that have recently joined the MANET.

To measure the performance, Visual DNS is assessed within the scope of an IEEE 802.11 based MANET. Experimental results show that the Visual DNS neighbor discovery protocol clearly achieves a trade-off between overhead and user discovery time. In our experiments we found that: (i) an inter-HELLO message interval (IHI) ranging from 2 to 8 seconds was appropriate for most cases, and (ii) the control traffic introduced did not affect the operation of active applications. To measure the time taken for completing the user profile initialization process to all nodes using different inter-HELLO intervals, we used a star topology to create a small ad hoc network formed by 5 static nodes. Figure 7.6 shows the

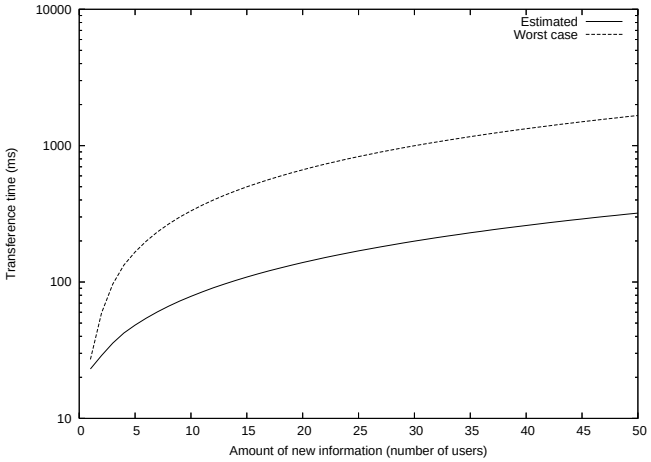

Figure 7.5: Data transmission overhead when varying the amount of new information.

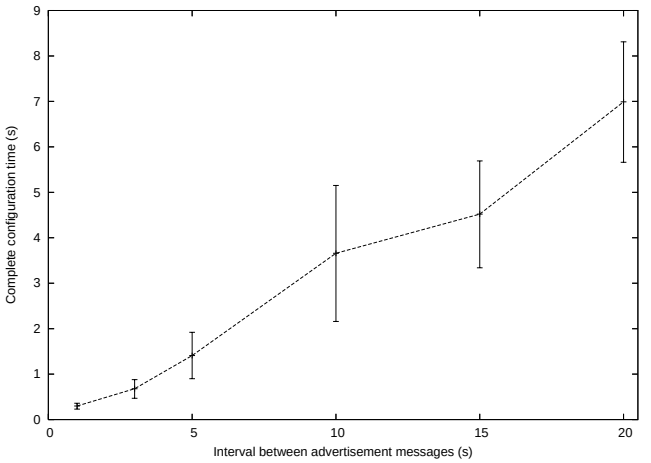

Figure 7.6: Visual DNS profile initialization time when varying the inter-HELLO interval (nodes at 1-hop).

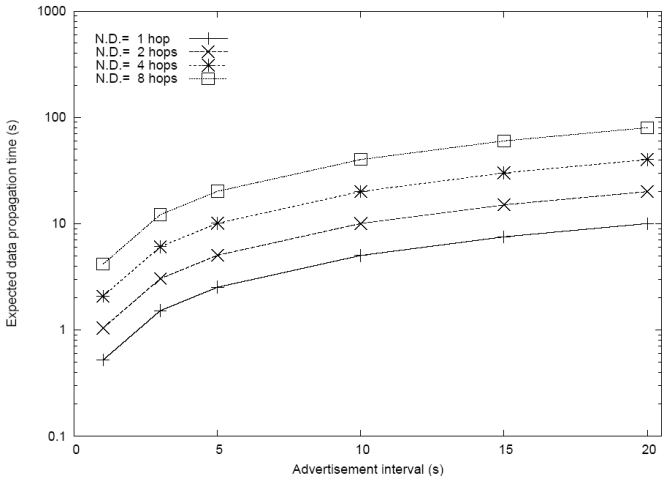

Figure 7.7: Visual DNS profile reconfiguration time - due to a new node joining in the existing MANET for different network diameters.

measured times, along with an interval whose range defines the standard deviation. From these results, we observe that user profile initialization time increases almost linearly with the advertisement interval.

To assess the impact of the inter-HELLO interval on Visual DNS profile reconfiguration time (PRT), we created a small testbed with a network diameter between 1 and 8 hops, forcing a chain configuration for nodes (worst case scenario). Figure 7.7 shows the expected Visual DNS profile reconfiguration time for a new user that has just joined the MANET at one of the chain's edges. This behavior can be broadly described by the following equation:

$$PRT \cong \sum_{i=1}^{N} (\frac{IHI}{2} + \Omega_i)$$

where Ω_i represents the time required to transfer the new user profile using TCP for hop i, and for a total of N hops.

We observe that an increase in the network diameter will cause the Visual DNS profile reconfiguration time to increase linearly, as expected. So, in order to avoid long profile reconfiguration times, it is important to keep the inter-HELLO message intervals low. However, we must also take into account that the protocol overhead is directly proportional to that interval, and so a trade-off must be sought. Based on the results, we consider that an interval of 3 seconds offers a reasonable trade-off between control overhead and configuration time. We chose that scenario because it was a worst case scenario in terms of user profiles exchange time, which is mostly proportional to the number of hops.

However, simulation-based analysis is required in order to make a more in-depth analysis. This is the objective of the next subsection.

7.2.2 Simulation results

We now evaluate the Visual DNS discovery protocol by using the OMNeT++ simulator. The use of simulation tools is very important since they allow to extrapolate results to large scenarios, making possible to detect errors that are impossible to detect in real scenarios; notice that some errors only become evident with an increasing complexity. Therefore, our objectives were twofold: firstly, to validate the results obtained from real scenarios in the simulator, and secondly, to evaluate the protocol's scalability as we increase the number of nodes.

Notice that in the results shown in the following sections, each value represented is the result of running ten tests randomly selected, and getting the average of these ten values. In each test, randomness can be applied to some of the parameters considered (e.g. the initial position of each node in the scenario).

7.2.2.1 OMNeT++ overview

OMNeT++ [Var01] is an extensible, modular, object-oriented and discrete event simulation framework which is currently becoming widely adopted in the scientific community. Its main application area focuses on simulating communication networks, but due to its generic architecture it can also be used for modeling and simulation of any other system that fits the discrete event approach. Domain-specific functionality, such as support for wireless mobile ad-hoc networks, is provided by complementary frameworks developed as independent projects (e.g. INET [INE04], INETMANET [INE09]).

OMNeT++ itself is not a specific simulator for a given scenario, but instead it provides the necessary infrastructure and tools for creating simulation scenarios. A key element of this infrastructure is that simulation models are built through a component-based architecture. To this end, models are assembled from reusable components termed modules, which can be connected between them via *gates*, allowing to create combinations to form compound modules. Note that there is no limit to the depth of module nesting.

Communication between modules is done through message passing along: (i) predefined paths via gates and connections, or (ii) directly to their destination. Notice that the latter case is useful for wireless simulations, which is the case at hand. In the lowest level of the module hierarchy, simple modules encapsulate the model behavior. These modules are programmed in C++, thus making use of the simulation library.

Concerning the simulation IDE, OMNeT++ is based on the Eclipse platform, which is extended with new properties and functionalities such as editors, views, and wizards. Therefore, as a result of this combination:

- OMNeT++ adds the required functionality for creating and configuring simulation models, performing batch executions and finally analyzing the obtained simulation results.
- Eclipse provides C++ programming (also supports Java and C#), version control integration, and other optional features such as UML modeling, bug-tracking support, database access or real-time simulation.

Simulations in OMNeT++ can be run under graphical and command-line user interfaces. The first one is highly useful for demonstration and debugging purposes, while the second is a better option for batch execution. All the tools available in the simulator environment are highly portable, as they have been tested on the most widely extended operating systems (such as Linux, Mac OS/X, and Windows) and compiled on most Unix-like operating systems. Finally, it is important to remark that OMNeT++ is free for academic and non-profit use, but for commercial purposes it is needed to obtain OMNEST licenses from Simulcraft Inc.

7.2.2.2 Simulator setup

As commented in the previous section, providing support for specific functionalities requires complementary frameworks over the basic OMNeT++ model. In our case, we needed support for wireless mobile ad-hoc networks. To this end, the INET framework offers general support for TCP/IP networks, including wired and wireless networking protocols such as: IPv4, IPv6, TCP, UDP, SCTP, Ethernet, 802.11, PPP, MPLS, OSPF, etc. Although the INET framework supports wireless and mobile simulations, it does not offer coverage for simulating mobile ad-hoc networks or using other wireless link layer protocols not included on it. That is, the INET framework is not accurate enough for modeling our system.

In order to overcome this limitation, however, the INETMANET framework arises as an extension derived from the INET framework adding specific support for mobile ad-hoc networks. That is, INETMANET provides the same support as the INET framework, but it also includes additional protocols and components especially used in MANET environments. Some of them are listed below:

- Link layer protocols: 802.11a/g/e, 802.15.4 (ZigBee), 802.16e (Wimax), rstp.
- Propagation models: Free Space, Log normal shadowing, Nakagami, Rayleigh, Rice, Two-Ray Ground.
- Mobility models: Chiang, Gauss-Markov, Linear, Ns2, Random waypoint.
- Mobile ad hoc routing protocols: OLSR, DSR, DSDV, DYMO, AODV.
- Other modules: Multi-radio interfaces, Battery model, IP broadcast, etc.

To achieve our purpose, it was enough with only adding the INETMANET framework to OMNeT++ and then implementing our protocols over them. Since it is not possible performing simulations without setting up some of these parameters, the next step was configuring the essential ones (Table 7.2). Note that the routing protocol is required for communicating with nodes located at distances greater than 1-hop, but in the following tests we only evaluated the dissemination protocol.

Table 7.2: Initial configuration parameters.

Link layer	Propagation	Mobility	Routing
802.11g (54Mbps)	Two-Ray Ground	Random waypoint	None

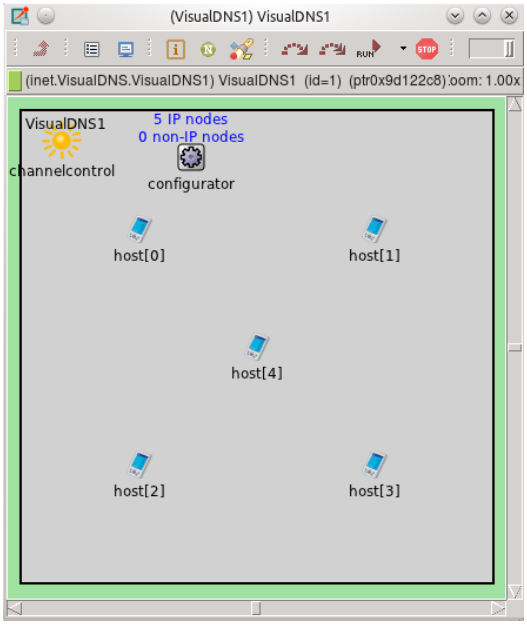

Figure 7.8: Example of star topology with 5 nodes.

7.2.2.3 Validation: testbed vs simulation

After setting up the required simulator parameters, the objective was to compare the results obtained in the previous real scenarios with the results obtained through the representation of these scenarios in the simulator. This comparison is very important because it gives us an idea of the simulator accuracy, which is essential to validate the results in more complex scenarios.

Therefore, in the first test we measured the total setup time of 5 nodes that form a star topology, while considering different inter-HELLO intervals. Figure 7.8 shows the topology representation in the simulator's graphical tool. Note that all nodes are at 1-hop distance.

Figure 7.9 shows the obtained results. As can be seen, if we compare this graph with Figure 7.6, which represents the real case, we observe how the trend of the curves are the same, although the simulation values are slightly higher than those obtained in the real scenario. These differences are mainly because, in the real tests, all users start the discovery protocol of Visual DNS in a time interval of a few seconds, and so, differences between the start times of each user are small. On the other hand, in the simulation tests, the start time of each node is a random value determined by a uniform distribution in the interval *[0, inter-HELLO interval]*. Therefore, the mean value of the protocol's start time should be approximately half the inter-HELLO interval, being closer to this value as the number of samples increases. In our example case, considering an inter-HELLO interval of 10 seconds, the intermediate value in the interval *[0, 10]* is 5 seconds. Looking at Figure 7.9 we see that, indeed, the total setup time for each case is close to this value.

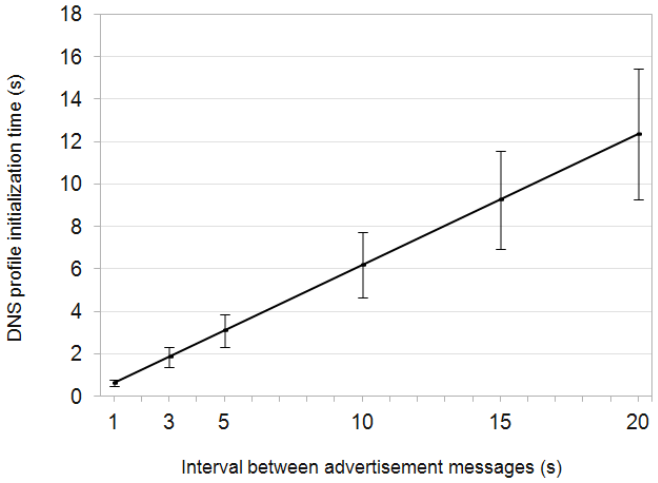

Figure 7.9: Initialization time varying the inter-HELLO interval (nodes at 1-hop).

In the second test, we measured the network reconfiguration time when a new node joins the network. The network topology in this case forces a chain configuration (worst case, considering the dissemination of information for the new node). In this scenario, we consider that between the endpoints of the chain, there will be a distance of 1, 2, 4 and 8 hops. Figure 7.10 shows an example of this topology consisting of 5 nodes. As we can observe, there is a 4-hops distance between the two end nodes. As in the first test, Figure 7.11 shows how the obtained results closely resemble the real case (the trend of the four curves in the chart is the same), but for the same reason, these values are slightly greater than the real ones.

The conclusion we can extract after this initial comparison between real and simulated environments is that, considering small static scenarios, the simulator model of our system can faithfully represent real-world environments.

Therefore, it means that we can now move on to evaluate larger and dynamic scenarios. In the following sections this evaluation is carried out. Note, however, that experiments only focus on the evaluation of the Visual DNS dissemination protocol, considering both setup and reconfiguration times, varying some parameters. Note also that in these experiments, as the sending packet rate caused by the dissemination protocol is low and the network bandwidth is high, there are no collisions between packets, meaning that all packets arrive to their destination on the first attempt, thus generating a packet loss rate equal to zero. If we consider the traffic generated by EasyMANET applications once the discovery protocol has finished, this assumption may no longer be true because it probably would have collisions between packets, but this is beyond the scope of this initial evaluation.

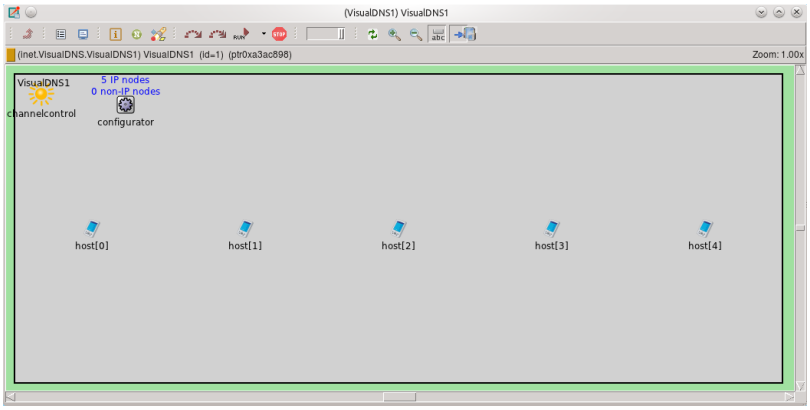

Figure 7.10: Example of chain topology formed by 5 nodes.

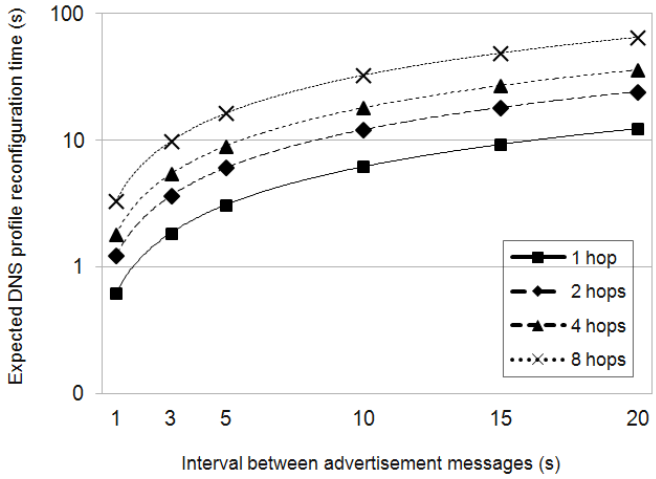

Figure 7.11: Reconfiguration time due to a new node joining in the existing MANET for different network diameters.

7.2.2.4 Baseline scenario.

The next step in the evaluation process was to establish a baseline scenario which could be taken as a first reference for large-scale assessment. This baseline scenario is derived by setting up some parameters of the simulator, which are detailed in Table 7.3. The objective was to measure the setup time of the 80 nodes deployed in the scenario. As mentioned previously, some other parameters, such as the nodes distribution along the scenario layout, or the simulation seed, are different in each run. Therefore, the 80 nodes are distributed randomly along the scenario layout in each run. In addition, as discussed in the basic tests, the time when each node starts broadcasting its list of known nodes using UDP packets is also a random value, in this case obtained by the uniform distribution in the interval *[0, 5]*.

Table 7.3: Parameters of the baseline scenario.

Size	Nodes	Radio range	inter-HELLO interval	Speed	Info size
870x870m	80	250m	5s	3m/s	3KB

Figure 7.12 shows a screenshot of the simulator with the 80 nodes baseline scenario[1]. This Figure also shows the distribution of broadcast packets from one node announcing its list of known nodes (red balls). The objective is to measure the configuration time for the 80 nodes, that is, when the 80 nodes have the profile information for all the other nodes, the simulation stops. In addition, these values will be obtained when the *25, 50, and 75%* of the nodes are configured.

The results for this test are shown in Figure 7.13. As commented previously, the X axis represents the percentage of nodes that have already been configured, from 25% to 100%. We observe how the total setup time for the 80 nodes is less than $8s$, which shows the efficiency of our dissemination protocol (remember that each value is the average of 10 independent random executions). This is because the density of nodes throughout the scenario is balanced. Furthermore, we see that the time interval between the configuration of 25% and 100% of nodes is very small (of about $0.5s$).

7.2.2.5 Scalability analysis of the total setup time.

In this section we address the variation of two parameters at once in each considered scenario in order to measure the total setup time of all nodes conforming the network. As we will see, combining some parameters allows having a more global view of the system behavior on large-scale scenarios. The three parameters that will be analyzed are: (i) the number of nodes, (ii) the inter-HELLO interval, and (iii) the nodes' speed.

[1] Although the figure shows that broadcast packets graphically reach all nodes in the network, this does not imply that disseminated information reaches nodes beyond the radio coverage of source node. That is, the wireless signal can be detected several nodes beyond the data delivery range, but they are not able to decode the signal (e.g. in our case, the scenario size is 870x870 meters, however, the effective radio range for data delivery is 250 meters). In those cases, some simulation parameters representing the physical level are highly relevant (e.g., the network noise or the signal attenuation).

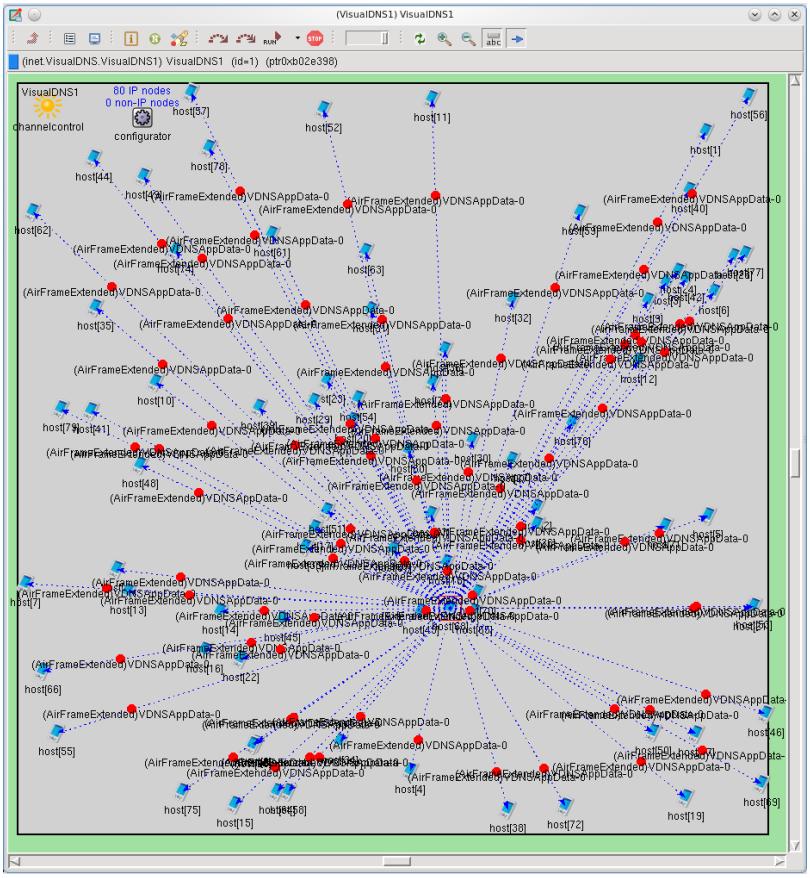

Figure 7.12: Screenshot of a Visual DNS running example in OMNeT with an UDP broadcast.

Test 1: varying the number of nodes and the inter-HELLO interval

In the first example scenario, we test with different values for the number of nodes (ranging from 50 to 400 nodes) and different values for the inter-HELLO interval (ranging from 1 to 20 seconds).

Table 7.4: Scenario sizes for different number of nodes.

Nodes	50	80	100	200	300	400
Size	$688m^2$	$870m^2$	$973m^2$	$1375m^2$	$1685m^2$	$1945m^2$

In addition, by varying these parameters, two different cases must be considered. The first one consists of increasing the scenario size and the number of nodes accordingly, thus keeping the same node density established in the baseline scenario. Table 7.4 shows the required scenario sizes for the first case. The second

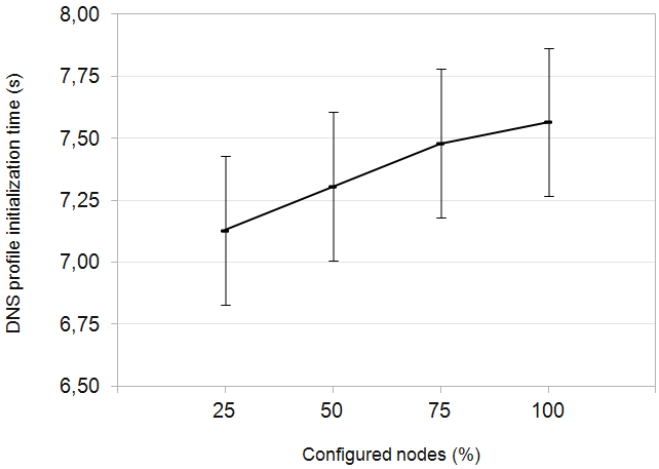

Figure 7.13: Baseline scenario.

case consists of increasing the number of nodes while maintaining the scenario size (this size corresponds to the baseline scenario value, $870m^2$), which implies a higher density of nodes in the scenario as the number of nodes increases. Notice that, in both cases, the remaining parameters have the same value as in the baseline scenario.

Figure 7.14 shows the results of the first case (fixed density, variable size). As we can observe, the total setup time logically increases with a higher inter-HELLO interval. This trend was detected in both the real testbed and the simulated tests conducted. Moreover, the setup time also increases with an increasing number of nodes. Notice that in this particular case, we have increased the scenario size proportionally to the number of nodes. Therefore, the mean hop count for the scenario becomes higher for a greater number of nodes, which explains the results obtained.

If we now consider the second case (variable density, fixed size), Figure 7.15 shows a different behavior. As in the previous case, the setup time increases for greater inter-HELLO intervals. However, for any given interval, the setup time decreases with an increasing number of nodes. To understand the behaviour experienced in this case, notice that the density of nodes in the scenario increases with the number of nodes, thereby reducing the mean hop count for the scenario. This also implies to reduce the possibility of having empty areas. That is, with a greater number of nodes is more likely to cover areas that were empty having fewer nodes. An empty area is one in which a source node does not find any destination node for sending broadcast packets, thus being the node isolated within its radio coverage. In this situation, the source node must spend some time moving around randomly, looking for a more populated area. Therefore, if the scenario does not contain empty areas (sparse scenario), the dissemination process will be more efficient, which means that a larger node density implies a lower setup time.

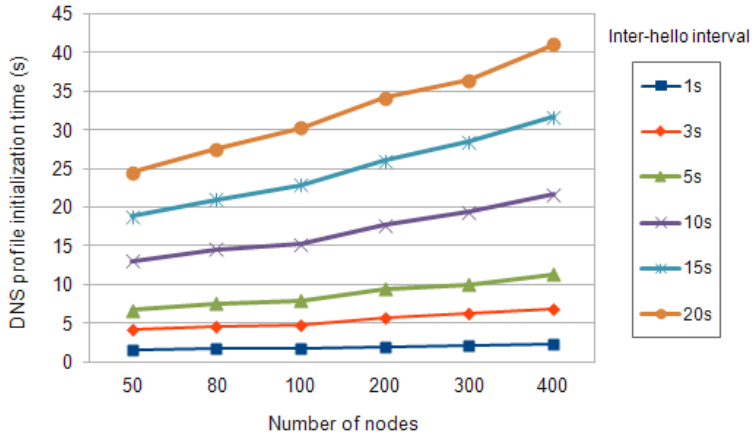

Figure 7.14: Initialization time when varying the inter-HELLO interval and the number of nodes maintaining density.

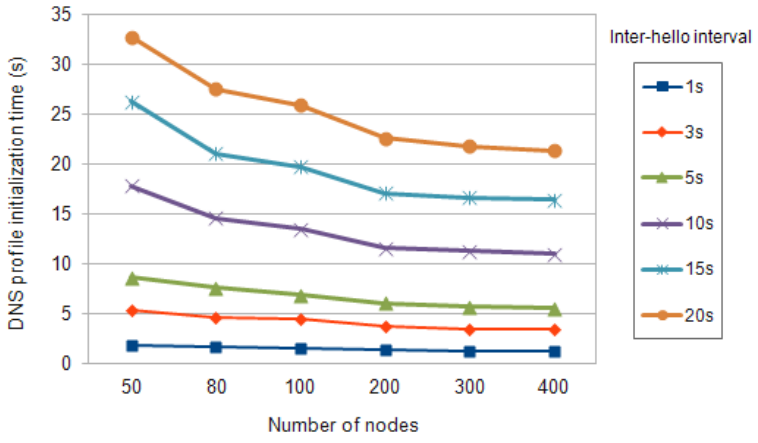

Figure 7.15: Initialization time when varying the inter-HELLO interval and the number of nodes maintaining the scenario size.

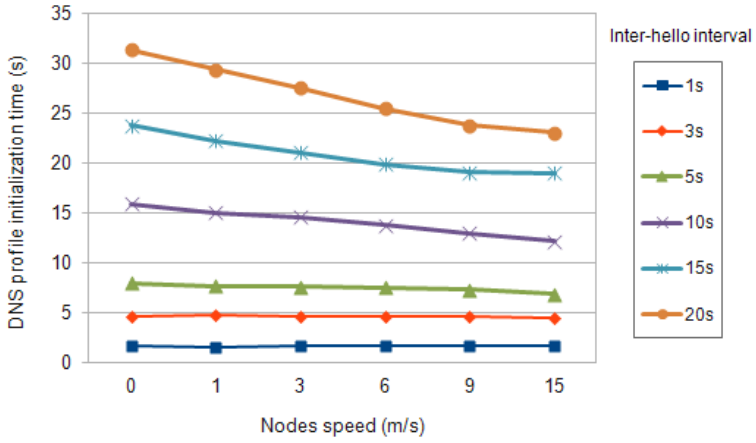

Figure 7.16: Initialization time when varying the inter-HELLO interval and the nodes speed.

Test 2: varying the nodes' speed and the inter-HELLO interval

In the second example scenario, the objective is to measure the total setup time considering different values for the nodes' speed (ranging from 0 to 15 m/s) and different values for the inter-HELLO interval (ranging from 1 to 20 seconds) while maintaining the number of nodes. Figure 7.16 shows the obtained results. As in the second case of Test 1, the setup time increases for longer inter-HELLO intervals. Similarly, for any given inter-HELLO interval, the setup time decreases when the nodes' speed increases. The reason is that, for higher node speeds[2], any isolated node remains in that situation for just a short period of time. Additionally, nodes carry new information to remote places more frequently. Therefore, the dissemination process will be more efficient at higher node speeds. In fact, if both the node speed and the inter-HELLO interval are high, the mobility dissemination effect alone allows improving performance. In summary, looking at Figure 7.16, the overall trend is downward for all inter-HELLO intervals considered.

7.2.2.6 Scalability analysis of the reconfiguration time

In the last set of tests, the objective is to measure the reconfiguration time when a new node arrives to a configured network considering different values for the number of nodes (ranging from 50 to 400 nodes) and different values for the inter-HELLO interval (ranging from 1 to 20 seconds). As established in Test 1 of the previous analysis, two different cases must be considered: (i) increasing the scenario size (fixed density), and (ii) maintaining the scenario size (variable density), setting the remaining parameters to the values of the baseline scenario.

[2]Note that again, the remaining parameters are the same that in the baseline scenario, that is, there are 80 nodes in a $870m^2$

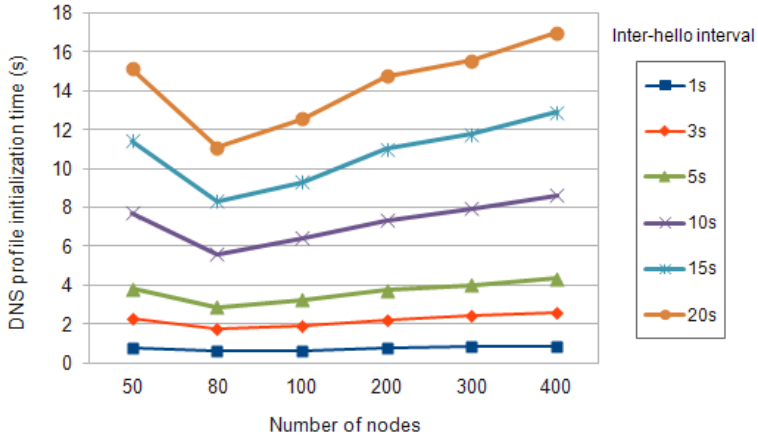

Figure 7.17: Reconfiguration time due to a new node joining in the existing MANET for different number of nodes maintaining density.

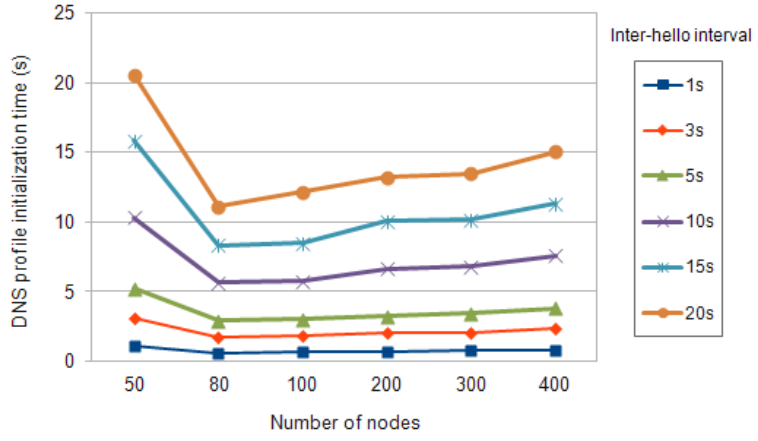

Figure 7.18: Reconfiguration time due to a new node joining in the existing MANET for different number of nodes maintaining the scenario size.

As we observe in Figure 7.17, for the first case (variable size, fixed density) the general behaviour is the same as in the first scenario for the two parameters considered (Figure 7.14), except for the case of 50 nodes. Notice that in the current scenario, a new node reaches a configured network taking a random location on it. Therefore, despite the node density in the scenario is maintained for the increasing number of nodes, the probability of being in a worst case situation in terms of maximum hop count increases for very low numbers of nodes. Additionally, when there are fewer nodes, they can become isolated more easily. This explains the obtained results.

Regarding the second case (fixed size, variable density), Figure 7.18 shows different results when comparing against the first scenario (Figure 7.15), except for the case of 50 nodes. We find that, as the number of nodes in the scenario increases, the dissemination time increases accordingly. Again the 50 nodes case is an exception to this general trend, which is due to the low node density in the network. Notice that, below a certain threshold, low node densities cause at least a few nodes to become isolated from the rest of the nodes at certain times, impeding data dissemination to be completed until mobility moves those nodes within range of the main node set. We also observe that the general behavior differs from that of Figure 7.15, where the information was coming from multiple sources at once, having a higher density of nodes which helped at decreasing the setup time instead.

7.3 Summary

Until now, mobile ad hoc networks were known mostly by researchers and experts in the computer networks field. Standard users are unaware of the applications of MANETs. This is mostly due to a lack of useful commercial applications for common users, despite its popularity in the military field. Given the current state-of-the-art in this area, we believe that the use of MANETs can be extended to non-expert users in a wide variety of scenarios. Such a view is also shared by the One Laptop Per Child project [One05], where OLPC laptops are configured so that they are able to create a MANET easily and effortlessly.

In this chapter, we have presented EasyMANET - an extensible platform that includes automatic setup and provision of useful context-aware services to short-lived MANETs. We have shown that using current off-the-shelf IEEE 802.11 wireless and Bluetooth technologies, along with an autoconfiguration system and Visual DNS, we are able to establish MANETs effortlessly. Visual DNS allows MANET users to identify others through their photos, names, and profiles.

We deployed a real testbed to assess the ease of use of EasyMANET. We observed that 99% of EasyMANET users get their client devices configured typically under 20 seconds, which is a reasonable time without causing user annoyance. From that point onwards, users are able to communicate using the Visual DNS interface. In fact, users have claimed that the tool is intuitive and functional.

Finally, simulations results have shown that EasyMANET is scalable because it works well with medium/large scenario sizes, disseminating information to all the network nodes in a few seconds.

Chapter 8

Conclusions, Publications and Future Work

Throughout this thesis several contributions have been made to the areas of Pervasive Computing and Mobile Ad Hoc Networks. With the main objective of assimilating as much as possible the concept of ubiquitous or pervasive computing, in this thesis we have proposed different prototype applications for PAN and MANET environments based on wireless technologies such as IEEE 802.11 and Bluetooth. Working with these networks requires some degree of experience and knowledge by users. Therefore, the target of the developed prototypes was to automate some of the processes (e.g. setting the parameters for establishing a MANET) so that inexperienced users can work with them in a simple and fast manner. After testing the functioning of the different prototype applications, the work focused on performing experiments in order to estimate some important parameters (e.g. the system performance, the overhead, etc.) as well as validate their correctness. Finally, we have make simulation tests, which allowed us to extrapolate the testbed results to a much greater scope. In particular, we tested with a high number of network nodes and high number of hops between nodes.

We now proceed to summarize the most relevant contributions of this work:

- BlueHospital. A system prototype that provides context-aware information and location based services to clinicians in hospitals' recovery wards.

- MANETs autoconfiguration. Two solutions to the wireless interface configuration problem have been proposed. The objective is to configure the parameters that allow a node to be part of a MANET in a way that is transparent to the user.

- Visual DNS. A solution to the name-to-address translation problem has been proposed. The solution focuses on MANETs, and solves the problem by performing a visual mapping, which makes it more intuitive from the user point of view.

- Service discovery. A solution to the service discovery problem has been proposed. The solution allows any node using Visual DNS to be able to discover what kind of network services are offered by other nodes.
- EasyMANET. Visual DNS has been integrated into a more complete platform (EasyMANET) which makes use of Visual DNS as a basic and mandatory service. EasyMANET intends to make the use of MANETs something simple and intuitive, providing support for different kind of services.
- Large-scale validation. The OMNeT++ simulator has been enhanced in order to perform large-scale evaluations of the Visual DNS discovery protocol. As we have seen, the protocol performs quite well even in scenarios with more than 50 nodes, meaning that Visual DNS is highly scalable.

Having accomplished all of our predefined goals, we consider that the ultimate purpose of this thesis has been achieved successfully, and so we conclude this dissertation.

8.1 Publications Related to the Thesis

The research work related to this thesis has resulted in 11 publications; among them we have 2 book chapters, 2 journal articles (indexed by the Journal Citation Reports (JCR) database), and 7 conference papers (4 of them indexed by the Computer Science Conference Ranking or the Computing Research and Education (CORE) lists). We now proceed by presenting a brief description of each of them.

8.1.1 Book Chapters

[CCCM08] J.-C. Cano, C. T. Calafate, J. Cano and P. Manzoni. "Deploying Pervasive Technologies", in *Encyclopedia of Information Science and Technology*, Second edition, Hershey, PA: Information Science Reference, 2008.

This book chapter presents our vision on ubiquitous computing by reporting our experiences building the BlueHospital system, a pervasive prototype that provides context-aware information and location-based services to clinicians on hospitals recovery wards. BlueHospital leverages Bluetooth technologies and Java services to offer patient information to clinical personnel based on the patient's profile and the clinicians' preferences and requirements.

[CCCM10] J. Cano, J.-C. Cano, C. T. Calafate, and P. Manzoni. "Experiences in Developing Ubiquitous Applications", in *Designing Solutions-Based Ubiquitous and Pervasive Computing: New Issues and Trends (Book)*, Published by Idea Group, Inc., publisher of the Idea Group Publishing, Information Science Publishing, IRM Press, CyberTech Publishing and Idea Group Reference imprints, 2010, pages: 97-112, ISBN13: 9781615208432, DOI: 10.4018/978-1-61520-843-2.ch005.

This book chapter describes the research work we did in the area of pervasive computing to define ubiquitous systems that fit into different types of environments. These systems are prototypes from which more sophisticated versions can be developed when targeting real ubiquitous scenarios. Selected environments include: 1) museums, where visitors can get personalized and context-aware computing information, 2) spontaneous networks in academic and business scenarios, that allow P2P connections to exchange any type of resources, 3) MANETs (Mobile Ad Hoc Networks), that allow an organized team to communicate independently in different situations, and also 4) intelligent hospitals, where it is possible to monitor patients automatically. We provided a detailed description of each prototype, including the system architecture as well as application and implementation details.

8.1.2 Journals

[CCT+10] J. Cano, J.-C. Cano, C.-K. Toh, C. T. Calafate, and P. Manzoni. "EasyMANET: An Extensible and Configurable Platform for Service Provisioning in MANET Environments", in *IEEE Communications Magazine*, December 2010, Vol. 48 Issue:12, pages: 159-167, ISSN: 0163-6804, DOI: 10.1109/MCOM.2010.5673087.

In this article we present EasyMANET, an extensible platform whose main objective is to encourage the widespread adoption and use of MANETs by non-expert users. To achieve this goal, EasyMANET provides two essential elements: an address autoconfiguration system and a name resolution service, known as Visual DNS. The autoconfiguration system allows users to join an 802.11-based MANET by establishing the parameters of their terminal interfaces quickly and transparently. Thereafter, Visual DNS offers a graphical view of MANET participants and gives users the possibility to access the services made available by other users. Examples are communication services (text-based chat, VoIP, videoconference), file sharing, and localization services. We performed several laboratory experiments, and evaluated the performance of EasyMANET based on address autoconfiguration time and Visual DNS performance. Results show that EasyMANET can be established within seconds, and EasyMANET applications are easy to use.

[VCT+12] M.-J. Villanueva, C. T. Calafate, A. Torres, J. Cano, J.-C. Cano, and P. Manzoni. "Seamless MANET autoconfiguration through enhanced 802.11 beaconing", in *Mobile Information systems Journal*, Volume 8, Issue 3, 2012. Status accepted.

In this article we propose a novel solution for the autoconfiguration of IEEE 802.11 based MANETs that relies on SSID parameter embedding. The solution allows users to join an existing MANET without resorting to any additional technology, and even in the presence of encrypted communications. Experimental testbed results using a real implementation of the proposed

solution show that it introduces significant improvements compared to other existing solutions, allowing nearby nodes to be configured in about two seconds, and also enabling multihop dissemination of configuration data to take place quickly and efficiently.

8.1.3 International Conferences

[CCMF05] J. Cano, J.-C. Cano, P. Manzoni, and D. Ferrández. "On the Design of Spontaneous Networks Using a P2P Approach and Bluetooth", in *10th IEEE Symposium on Computers and Communications (ISCC 2005)*, La Manga del Mar Menor, Cartagena, Spain, June 2005.

In this paper, we address the design of spontaneous networks using a peer-to-peer (P2P) architecture and Bluetooth technology. A spontaneous network is a small infrastructureless network formed when a group of people come together to participate in some collaborative activity. We developed a base library and application work ground for easy spontaneous networks development. Based on it, we presented an experimental application that provided spontaneous networks with context and transparent services to interchange resources between peers. We described the overall network architecture and presented details of the implementation steps taken to create our P2P and Bluetooth based application. Finally, we run some experiments in a small testbed to evaluate the performance and system behaviour. We presented our findings in terms of the duration of the inquiry procedure and throughput performance with respect to distance and node speed.

[CCMD06] J.-C. Cano, J. Cano, P. Manzoni, and D. Kim. "On the design of pervasive computing applications based on Bluetooth and a P2P concept", in *1st International IEEE Symposium on Wireless Pervasive Computing (ISWPC 2006)*, Phuket, Thailand, January 2006.

As an example of implementing the pervasive computing concept, in this paper we developed an application framework to deploy an easy, spontaneous, and infrastructureless network. We selected the Bluetooth technology with the peer-to-peer (P2P) concept to develop an experimental application which enables peers to exchange their resources. The overall network architecture and the prototype application are presented. We selected a small test-bed and simulation to evaluate the overall performance and the system behaviour. We then presented our findings in terms of duration of the inquiry procedure and throughput, according to various experimental parameters such as physical distance between nodes and their speeds.

[CBC+06b] J. Cano, E. Burgoa, C. T. Calafate, J.-C. Cano, and P. Manzoni. "A MANET autoconfiguration system based on Bluetooth technology", in *3rd IEEE International Symposium on Wireless Communication Systems (ISWCS 2006)*, Valencia, Spain, September 2006.

In this paper we proposed using Bluetooth technology to solve the configuration problem of the terminals conforming an IEEE 802.11-based ad-hoc network. The main objective of mobile ad-hoc networks (MANETs) is to extend the connectivity range of nodes through packet forwarding, thereby avoiding the use of a fixed infrastructure. However, since configuration of nodes is a complex issue, we provide a fast and reliable solution to autoconfigure MANET terminals. The solution is adequate for the quick setup and deployment of rescue, military or any other sorts of organized teams. We present the architecture of the proposed system by means of a simple example, along with the applications developed for both client and server. We conclude with some experiments to assess the performance of the proposed architecture.

[CCCM07a] J. Cano, J.-C. Cano, C. T. Calafate, and P. Manzoni. "Solving the user-to-host binding problem in ad hoc networks through the dissemination of photographic identifiers", in *4th ACM International Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks (PE-WASUN 2007)*, Chania, Crete Island, Greece, October 2007.

In this paper we presented Visual DNS, which is an extension to the DNS service that meets the requirements of MANET environments. Visual DNS allows any node in the network to discover any other node in a transparent manner, and independently of its location. The proposed application makes use of a distributed user discovery protocol based on reliable data dissemination to share information about users. Information shared includes the user name and photo, along with the host's IP address, among other things. Once the discovery protocol starts, the application offers users the possibility to communicate with other known users through either a text based chat application, a VoIP session or a videoconference session. We present the architecture of the proposed application, along with some implementation details. Finally, experimental results show that, if properly tuned, the proposed data dissemination strategy allows user information to reach all the nodes in a MANET in just a few seconds.

8.1.4 National Conferences

[CCMD05] J. Cano, J.-C. Cano, P. Manzoni, and D. Ferrández. "A peer-level multi-hop based architecture to support spontaneous networks formation", in *Simposio sobre Computación Ubicua e Inteligencia Ambiental (UCAmI'05) part of the 1er Congreso Español de Informática (CEDI'05)*, Granada, Spain, September 2005.

In this work, we developed a base library and application work ground for easy spontaneous networks development. Based on it, we present an exper-

imental application that provides spontaneous networks with context and transparent services to interchange resources between peers. We describe the overall network architecture and present details of the implementation steps taken to create our P2P and Bluetooth based application. Finally we run some experiments in a small testbed. We present our findings in terms of the duration of the inquiry procedure. We observed that at an average of 3s, users can obtain the desired information almost 50% of the times. This waiting period is normally acceptable to a user without causing annoyance.

[CBC+06a] J. Cano, E. Burgoa, C. T. Calafate, J.-C. Cano, and P. Manzoni. "An autoconfiguration method for IEEE 802.11 based MANETs using Bluetooth", in *XVII Jornadas de Paralelismo*, Albacete, Spain, September 2006.

In this paper we proposed using Bluetooth technology to solve the configuration problem of the terminals conforming an IEEE 802.11-based ad-hoc network. The main objective of mobile ad-hoc networks (MANETs) is to extend the connectivity range of nodes through packet forwarding, thereby avoiding the use of a fixed infrastructure. However, since configuration of nodes is a complex issue, we provide a fast and reliable solution to autoconfigure MANET terminals. Our solution is adequate for the quick setup and deployment of rescue, military or any other sort of organized team. We present the architecture of the proposed system by means of a simple example, along with the applications developed for both client and server. We conclude with some experiments to assess the performance of the proposed architecture.

[CCCM07b] J. Cano, J.-C. Cano, C. T. Calafate, and P. Manzoni. "Visual DNS: a pervasive application offering peer collaboration in mobile ad hoc networks", in *XVIII Jornadas de Paralelismo*, Zaragoza, Spain, September 2007.

In this paper we presented Visual DNS, which is an extension to the DNS service that meets the requirements of MANET environments. Visual DNS allows any node in the network to discover any other node in a transparent manner, and independently of its location. The proposed application makes use of a distributed user discovery protocol based on reliable data dissemination to share information about users. Information shared includes the user name and photo, along with the host's IP address, among other things. Once the discovery protocol starts, the application offers users the possibility to communicate with other known users through either a text based chat application, a VoIP session or a videoconference session. We present the architecture of the proposed application, along with some implementation details. We consider that the proposed application could be a candidate solution to encourage the widespread adoption and use of MANETs in a wide variety of scenarios.

8.2 Future work

In the development of this thesis several issues emerged which deserve further scrutiny in a future. The ones we consider most relevant are the following:

- To study new reconfiguration techniques for the proposed autoconfiguration solutions taking into account the restrictions imposed by the considered environment in each case.
- To extend/adapt the proposed prototypes (mainly EasyMANET) considering new wireless technologies (e.g. WiMAX, UMTS, GPS) in order to offer solutions to new areas/environments (e.g. vehicular ad-hoc networks -VANETs-).
- To further test EasyMANET with bigger scenarios (very large scale evaluation considering thousands of nodes) and focusing more on routing issues.
- To add support for multicast communication to EasyMANET considering automatic assignment of multicast addresses, as proposed by Zeroconf. Furthermore, to consider content delivery by using delay-tolerant networking (DTN) techniques.

Bibliography

[AAH+97] Gregory D. Abowd, Christopher G. Atkeson, Jason I. Hong, Sue Long, Rob Kooper, and Mike Pinkerton. Cyberguide: A mobile context-aware tour guide. *Wireless Networks*, 3:421–433, 1997.

[ACK94] Abhaya Asthana, Mark Cravatts, and Paul Krzyzanowski. An indoor wireless system for personalized shopping assistance. In *In Proceedings of IEEE Workshop on Mobile Computing Systems and Applications*, pages 69–74. IEEE Computer Society Press, 1994.

[ARS01] Asis Nasipuri, Robert Castaneda, and Samir R. Das. Performance of multipath routing for on-demand protocols in mobile ad hoc networks. *ACM/Baltzer Mobile Networks and Applications (MONET) Journal, vol. 6*, pages 339–349, 2001.

[Asu06] AsusTek Computer Inc. Asus wl500g premium wireless internet router review. http://www.asus.com, 2006.

[AT99] George Aggelou and Rahim Tafazolli. RDMAR: A bandwidth-efficient routing protocol for mobile ad hoc networks. In *Proceedings of the WOWMOM*, pages 26–33, 1999.

[Bas99] S. Basagni. Distributed clustering for ad hoc networks. In *Proceedings of the IEEE International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN)*, pages 310–315, Perth, Western Australia, June 1999.

[BDM06] Luca Benini and Giovanni De Micheli. *Networks on Chips: Technology and Tools*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2006.

[BDR07] Matthias Baldauf, Schahram Dustdar, and Florian Rosenberg. A survey on context-aware systems. *Int. J. Ad Hoc Ubiquitous Comput.*, 2:263–277, June 2007.

[Ben07] Iván Benito. The bluehospital project: implementation details. Bachelor Thesis (in spanish), Technical University of Valencia, 2007.

[BGW01] Nikita Borisov, Ian Goldberg, and David Wagner. Intercepting mobile communications: the insecurity of 802.11. In *Proceedings of the*

7th annual international conference on Mobile computing and networking, MobiCom '01, pages 180–189, Rome, Italy, 2001.

[BJC+06] B. Ford, J. Strauss, C. Lesniewski-Laas, S. Rhea, F. Kaashoek, and R. Andmorris. Persistent personal names for globally connected mobile devices. In *USENIX 06: Proceedings of the 7th conference on USENIX Symposium on Operating Systems Design and Implementation*, pages 233–248, Seattle, USA, 2006.

[Blu02] IEEE 802.15.1(tm) IEEE Standard for Information technology–Telecommunications and information exchange between systems– Local and metropolitan area networks–Specific requirements Part 15.1: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Wireless Personal Area Networks (WPANs(tm)), 2002.

[BM97] Jonathan Broadbent and Patrizia Marti. Location aware mobile interactive guides: Usability issues, 1997.

[Bol02] J. Boleng. Efficient Network Layer Addressing for Mobile Ad Hoc Networks. In *Proc. of International Conference on Wireless Networks (ICWN02)*, pages 271–277, Las Vegas, USA, June 2002.

[BPS04] Microsoft Bluetooth Protocol Stack. Microsoft support for bluetooth on windows xp and windows ce. Available at http://msdn.microsoft.com/library/, 2004.

[BR99] Bhargav Bellur and Richard G. Ogier. A Reliable, Efficient Topology Broadcast Protocol for Dynamic Networks. Proceedings of the IEEE INFOCOM, The Conference on Computer Communications, New York, USA, March 1999.

[CBC+06a] J. Cano, E. Burgoa, C. T. Calafate, J.-C. Cano, and P. Manzoni. An autoconfiguration method for ieee 802.11 based manets using bluetooth. In *XVII Jornadas de Paralelismo*, Albacete, Spain, September 2006.

[CBC+06b] J. Cano, E. Burgoa, C. T. Calafate, J.-C. Cano, and P. Manzoni. A manet autoconfiguration system based on bluetooth technology. In *3rd International IEEE Symposium on Wireless Communication Systems*, ISWCS '06, pages 674–678, Valencia, Spain, September 2006.

[CCCM07a] J. Cano, J.-C. Cano, C. T. Calafate, and P. Manzoni. Solving the user-to-host binding problem in ad hoc networks through photo-ids. In *Proceedings of the 4th ACM Workshop on Performance Evaluation of Wireless ad hoc, Sensor, and Ubiquitous Networks*, PE-WASUN '07, pages 78–81, Chania, Crete Island, Greece, October 2007.

[CCCM07b] J. Cano, J.-C. Cano, C. T. Calafate, and P. Manzoni. Visual dns: a pervasive application offering peer collaboration in mobile ad hoc networks. In *XVIII Jornadas de Paralelismo*, Zaragoza, Spain, September 2007.

[CCCM08] J.-C. Cano, C. T. Calafate, J. Cano, and P. Manzoni. *Deploying Pervasive Technologies*, volume 2. Information Science Reference, Hershey, PA, USA, second edition, 2008.

[CCCM10] J. Cano, J.-C. Cano, C. T. Calafate, and P. Manzoni. *Experiences in Developing Ubiquitous Applications*. Published by Idea Group, Inc., publisher of the Idea Group Publishing, Information Science Publishing, IRM Press, CyberTech Publishing and Idea Group Reference imprints, 2010.

[CCM+07] J.-C. Cano, C. T. Calafate, P. Manzoni, J.-M. Cano, and E. González. Design and validation of a low-power network node for pervasive applications. In *Proceedings of the 2007 International Conference on Intelligent Pervasive Computing*, IPC '07, pages 527–530, Los Alamitos, CA, USA, 2007. IEEE Computer Society.

[CCMD05] J. Cano, J.-C. Cano, P. Manzoni, and Ferrández D. A peer-level multi-hop based architecture to support spontaneous networks formation. In *Simposio sobre Computación Ubicua e Inteligencia Ambiental (UCAmI 05) part of the 1er Congreso Espanyol de Informática (CEDI 05)*, Granada, Spain, September 2005.

[CCMD06] J.-C. Cano, J. Cano, P. Manzoni, and Kim D. On the design of pervasive computing applications based on bluetooth and a p2p concept. In *Proceedings of the 1st International IEEE Symposium on Wireless Pervasive Computing*, ISWPC '06, Phuket, Thailand, January 2006.

[CCMF05] J. Cano, J.-C. Cano, P. Manzoni, and D. Ferrández. On the design of spontaneous networks using a p2p approach and bluetooth. In *Proceedings of the 10th IEEE Symposium on Computers and Communications*, ISCC '05, pages 125–130, La Manga del Mar Menor, Cartagena, Spain, June 2005.

[CCT+10] J. Cano, J.-C. Cano, C. K. Toh, C. T. Calafate, and P. Manzoni. Easymanet: an extensible and configurable platform for service provisioning in manet environments. *IEEE Communications Magazine*, 48:159–167, December 2010.

[CES01] C. E. Perkins, E. M. Royer, and S. R. Das. Ip address autoconfiguration for ad hoc networks. Internet Draft, IETF MANET Working Group, draft-perkins-manet-autoconf-01.txt, November 2001. Work in progress.

[CES03] Charles E. Perkins, Elizabeth M. Belding-Royer, and Samir R. Das. Ad hoc on-demand distance vector (AODV) routing. Request for Comments 3561, MANET Working Group, http://www.ietf.org/rfc/rfc3561.txt, July 2003. Work in progress.

[CFBM05] Juan-Carlos Cano, David Ferrández-Bell, and Pietro Manzoni. Evaluating bluetooth performance as the support for Context-Aware applications. *Telecommunication Systems*, 28:333–347, March 2005.

[COO99] Richard Cox, Mick O'Donnell, and Jon Oberlander. Dynamic versus static hypermedia in museum education: an evaluation of ilex, the intelligent labelling explorer. In *Proceedings of the Conference on Artificial Intelligence in Education (AI-ED 1999), Le Mans*, pages 181–188. Press, 1999.

[CP94] C. E. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. *ACM Computer Communication Review*, 24(2):234–244, October 1994.

[CQS98] X. Chen, L. Qi, and D. Sun. Global and superlinear convergence of the smoothing Newton method and its application to general box constrained variational inequalities. *Mathematics of Computation*, 67(222):519–540, 1998.

[CRP03] Carlos T. Calafate, Roman Garcia Garcia, and Pietro Manzoni. Optimizing the implementation of a manet routing protocol in a heterogeneous environment. In *The 8th IEEE Symposium on Computers and Communications (ISCC'2003)*, Kemer - Antalya, Turkey, July 2003.

[CSR02] CSR Inc. Bluessite software for bluetooth development. http://www.csr.com/development/bluesuite.htm, 2002.

[CSX+08] Xiaowen Chu, Yi Sun, Ke Xu, Z. Sakander, and Jiangchuan Liu. Quadratic residue based address allocation for mobile ad hoc networks. In *Communications, 2008. ICC '08. IEEE International Conference on*, pages 2343 –2347, May 2008.

[DATY02] D. Chakraborty, A. Joshi, T. Finin, and Y. Yesha. GSD: A Novel Group-based Service Discovery Protocol for MANETs. In *4th IEEE Conference on Mobile and Wireless Communications Networks (MWCN)*, Stockholm, Sweden, September 2002.

[DDY04] David B. Johnson, David A. Maltz, and Yih-Chun Hu. The dynamic source routing protocol. Internet Draft, MANET Working Group, draft-ietf-manet-dsr-10.txt, July 2004. Work in progress.

[DMC+98] Nigel Davies, Keith Mitchell, Keith Cheverst, Gordon Blair, Keith Cheverst, and Gordon Blair. Developing a context sensitive tourist guide, 1998.

[DP01] D. Eastlake and P. Jones. US Secure Hash Algorithm 1 (SHA1). IETF RFC 3174, September 2001.

[DRWT96] Rohit Dube, Cynthia D. Rais, Kuang-Yeh Wang, and Satish K. Tripathi. Signal stability based adaptive routing (ssa) for ad-hoc mobile networks. Technical report, 1996.

[EC96] E. Kaplan and C. Hegarty . *Understanding GPS: Principles and Applications*. Artech House, Boston, 1996.

[FB10] Jose Flich and Davide Bertozzi. *Designing Network On-Chip Architectures in the Nanoscale Era*. CRC Press, Inc., Boca Raton, FL, USA, 2010.

[FCK+05] Daniel Fitton, Keith Cheverst, Chris Kray, Alan Dix, Mark Rouncefield, and George Saslis-Lagoudakis. Rapid prototyping and user-centered design of interactive display-based systems. *IEEE Pervasive Computing*, 4:58–66, October 2005.

[FFF+02] M. Fleck, Margaret Fleck, M. Frid, E. O'Brien-Strain, Marcos Frid, Tim Kindberg, Tim Kindberg, Rakhi Rajani, Rakhi Rajani, Mirjana Spasojevic, and Mirjana Spasojevic. Rememberer: A tool for capturing museum visits. In *Proc. Ubicomp 2002*, pages 48–55. Springer-Verlag, 2002.

[Foo02] Peter Foot. Opennetcf bluetooth library. Available at http://www.opennetcf.org/, 2002.

[FV05] F. Sailhan and V. Issarny. Scalable Service Discovery for MANET. In *Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications (PERCOM '05)*, pages 235–244, Kauai Island, Hawaii, March 2005.

[GC01] G. Guichal and C.K. Toh. Service location architectures for mobile ad hoc wireless networks. In *Proc. IEEE PIMRC*, San Diego, CA, USA, 2001.

[GCNB01] J. Gomez, A. Campbell, M. Naghshineh, and C. Bisdikian. Conserving transmission power in wireless ad hoc networks, 2001.

[GDL+04] Robert Grimm, Janet Davis, Eric Lemar, Adam Macbeth, Steven Swanson, Thomas Anderson, Brian Bershad, Gaetano Borriello, Steven Gribble, and David Wetherall. System support for pervasive applications. *ACM Trans. Comput. Syst.*, 22:421–486, November 2004.

[GHP01] ISAAC Group Home Page. Security of the wep algorithm. Available at http://www.isaac.cs.berkeley.edu/isaac/wep-faq.html, 2001.

[GLAS99] J. J. Garcia-Luna-Aceves and Marcelo Spohn. Source-tree routing in wireless networks. In *ICNP*, pages 273–282, 1999.

[GSM97] Digital cellular telecommunications system (Phase 2+); Terminal Equipment to Mobile Station (TE-MS) multiplexer protocol. ETSI GSM 07.10 version 6.3.0, 1997.

[HAB96] H. Dobbertin, A. Bosselaers, and B. Preneel. RIPEMD-160, a strengthened version of RIPEMD. In *Fast Software Encryption, LNCS 1039*, pages 71–82, 1996.

[HBS06] Thomas Riisgaard Hansen, Jakob E. Bardram, and Mads Soegaard. Moving out of the lab: Deploying pervasive technologies in a hospital. *IEEE Pervasive Computing*, 5:24–31, July 2006.

[HD06] R. Hinden and S. Deering. IP Version 6 Addressing Architecture. IETF RFC 4291, February 2006.

[HH05] R. Hinden and B. Haberman. Unique Local IPv6 Unicast Addresses. IETF RFC 4193, October 2005.

[HHS+99] Andy Harter, Andy Hopper, Pete Steggles, Andy Ward, and Paul Webster. The anatomy of a context-aware application. *WIRELESS NETWORKS, VOL*, 8:187–197, 1999.

[HM02] Matt Henricksen and Bill Millan. Evaluation of rc4 algorithm executive summary, 2002.

[IC08] I. Chakeres and C. Perkins. Dynamic MANET On-demand (DYMO) Routing. Internet Draft, MANET Working Group, draft-ietf-manet-dymo-12.txt, February 2008. Work in progress.

[IEE88] IEEE Ultrasonics, Ferroelectrics and Frequency Control Society . *ANSI/IEEE Std 176-1987, IEEE standard on piezoelectricity*. January 1988.

[IEE99] IEEE 802.11 WG. International Standard for Information Technology - Telecom. and Information exchange between systems - Local and Metropolitan Area Networks - Specific Requirements - Part 11: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications, ISO/IEC 8802-11:1999(E) IEEE Std. 802.11, 1999.

[IET03] IETF. Ad hoc network autoconfiguration (autoconf) charter. http://www.ietf.org/html.charters/ -charter.html, 2003.

[INE04] INET Framework. An open-source communication networks simulation package for the OMNeT++ simulation environment. Available at http://inet.omnetpp.org/, 2004.

[INE09] INETMANET Framework. Mobile and ad-hoc networks simulator for OMNeT++. Available at https://github.com/inetmanet/inetmanet/wiki, 2009.

[Ing99] Davin S. L. Ing. Innovations in a technology museum. *IEEE Micro*, 19:44–52, November 1999.

[IrD01] IrDA Physical Layer Standard version 1.3. http://www.irda.org, March 2001.

[JDP03] Juan Carlos Cano, Dongkyun Kim, and Pietro Manzoni. CERA: Cluster-based Energy Saving Algorithm to Coordinate Routing in Short-Range Wireless Networks. The International Conference on Information Networking (ICOIN) 2003, Jeju Island, Korea, February 2003.

[JJ87] J. Jubin and J. D. Tornow. The darpa packet radio network protocols. *Proceedings of IEEE*, 75(1):21–32, January 1987.

[JJH03] J. Jeong, J. Parkt, and H. Kim. Ndr: Name directory service in mobile ad-hoc network. In *Proc. 5th Int. Conf. Advanced Commun. Tech.*, Phoenix Park, Korea, 2003.

[JJH04] J. Jeong, J. Park, and H. Kim. Auto-Networking Technologies for IPv6 Mobile Ad Hoc Networks. In *International Conference on Information Networking (ICOIN 2004)*, pages 257–267, Busan, Korea, February 2004.

[JM96] David B Johnson and David A Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[JO01] J. Beutel and O. Kasten. A minimal bluetooth-based computing and communication platform. Technical report, Computer Engineering and Networks Lab, Swiss Federal Institute of Technology (ETH) Zurich, 2001.

[JSL08] Jang-Ping Sheu, Shin-Chih Tu, and Li-Hsiang Chan. A distributed IP address assignment scheme in ad hoc networks. *Int. J. Ad Hoc Ubiquitous Computing*, 3(1):10–20, December 2008.

[Kar90] Phil Karn. MACA - A New Channel Access Method for Packet Radio. Proceedings of the 9th ARRL Computer Networking Conference, London, Ontario, Canada, 1990.

[KBM+02] Tim Kindberg, John Barton, Jeff Morgan, Gene Becker, Debbie Caswell, Philippe Debaty, Gita Gopal, Marcos Frid, Venky Krishnan, Howard Morris, John Schettino, Bill Serra, and MIRJANA SPASOJEVIC. People, places, things: Web presence for the real world, 2002.

[KK00] K. Fall and K. Varadhan. ns notes and documents. The VINT Project. UC Berkeley, LBL, USC/ISI, and Xerox PARC, February 2000. Available at http://www.isi.edu/nsnam/ns/ns-documentation.html.

[KM02] K. Weniger and M. Zitterbart. IPv6 autoconfiguration in large scale mobile ad-hoc networks. In *Proc. of European Wireless 2002*, pages 142–148, Florence, Italy, February 2002.

[Kra03] Maxim Krasnyansky. BlueZ: Official linux bluetooth protocol stack. http://bluez.sourceforge.net/, 2003.

[Kum02] C. Bala Kumar. JSR-82: Java APIs for Bluetooth. http://www.jcp.org/en/jsr/detail?id=82, 2002.

[KV98] Y.B. Ko and N.H. Vaidya. Location aided routing (LAR) in mobile ad-hoc networks. In *The Annual International Conference on Mobile Computing and Networking (MOBICOM)*, Dallas, Texas, USA, October 1998.

[Lab01] IBM Research Labs. Bluehoc: Bluetooth ad-hoc network simulator. Available at http://www-124.ibm.com/developerworks/opensource/bluehoc/, 2001.

[LBA01] L. M. Feeney, B. Ahlgren, and A. Westerlund. Spontaneous networking: An application-oriented approach to ad hoc networking. *IEEE Communications Magazine*, 39:176–181, 2001.

[LCWG97] W. Liu, C. Chiang, H. Wu, and C. Gerla. Routing in clustered multihop mobile wireless networks with fading channel. In *Proc. IEEE SICON'97*, pages 197–211, 1997.

[LG01] S. Lee and M. Gerla. Split multipath routing with maximally disjoint paths in ad hoc networks. In Proceedings of the IEEE ICC, pages 3201–3205, 2001.

[LKAA96] Sue Long, Rob Kooper, Gregory D. Abowd, and Christopher G. Atkeson. Rapid prototyping of mobile context-aware applications: The cyberguide case study. pages 97–107. ACM Press, 1996.

[LYM+01] Lei Wang, Yantai Shu, Miao Dong, Lianfang Zhang, and O.W.W. Yang. Adaptive multipath source routing in ad hoc networks. ICC 2001. IEEE International Conference on Communications; Page(s): 867-871 vol.3, 2001.

[Mal98] G. Malkin. RIP Version 2. RFC 2453 (Standard), November 1998. Updated by RFC 4822.

[MD01] M. Marina and S. Das. On demand multipath distance vector routing in ad hoc networks. In Proceedings of IEEE International Conference on Network Protocols (ICNP), pages 14–23, 2001.

[MGLA96] Shree Murthy and J. J. Garcia-Luna-Aceves. An efficient routing protocol for wireless networks. *Mobile Networks and Applications*, 1(2):183–197, 1996.

[Mit05] Mitsumi Electric Co., Ltd. Bluetooth module wml-c11 class 1. http://www.mitsumi.co.jp/english/, 2005.

[MMHH03] M. Aoki, M. Saito, H. Aida, and H.Tokuda. ANARCH: A Name Resolution Scheme for Mobile Ad Hoc Networks. In *Proceedings of the 17th International Conference on Advanced Information Networking and Applications (AINA '03)*, pages 723–730, Xi'an, China, March 2003.

[Moy98] John Moy. Open shortest path first (OSPF) version 2. Internet RFC 2328, April 1998.

[MR02] Mansoor Mohsin and Ravi Prakash. IP Address Assignment in a Mobile Ad Hoc Network. In *Proceedings of IEEE Military Communications Conference (MILCOM 2002)*, volume 2, pages 856–861, Anaheim, California, USA, October 2002.

[MSW+02] M. W. Newman, S. Izadi, W. K. Edwards, J. Z. Sedivy, and T. F. Smith. User interfaces when and where they are needed: an infrastructure for recombinant computing. In *Proceedings of the 15th annual ACM symposium on user interface software and technology (UIST 02)*, pages 171–180, Paris, France, October 2002.

[NTCS99] Sze-Yao Ni, Yu-Chee Tseng, Yuh-Shyan Chen, and Jang-Ping Sheu. The broadcast storm problem in a mobile ad hoc network. In *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 151–162, Seattle, Washington, United States, 1999.

[OAM09] H. Okada, K. Akima, and K. Mase. An overhead reduction strategy for weak duplicate address detection in mobile ad hoc networks. In *Personal, Indoor and Mobile Radio Communications, 2009 IEEE 20th International Symposium on*, pages 2618 –2622, 2009.

[OCPP07] O. Gama, C. Figueiredo, P. Carvalho, and P.M. Mendes. Towards a reconfigurable wireless sensor network for biomedical applications. International Conference on Sensor Technologies and Applications (SENSORCOMM 2007), 2007.

[ODBE02] O. Catrina, D. Thalerr, B. Aboba, and E. Guttmana. ZMAAP: Zeroconf Multicast Address Allocation Protocol. Internet Draft, IETF Zeroconf Working Group, draft-ietf-zeroconf-zmaap-02.txt, October 2002. Work in progress.

[One05] One Laptop per Child Association, Inc. One Laptop per Child (OLPC) project. http://www.laptop.org, 2005.

[Ope04] OpenWrt Project. OpenWrt Linux distribution. http://openwrt.org/, January 2004.

[PGC00] Guangyu Pei, Mario Gerla, and Tsu-Wei Chen. Fisheye state routing: A routing scheme for ad hoc wireless networks. In *ICC (1)*, pages 70–74, 2000.

[PGH00] G. Pei, M. Gerla, and X. Hong. Lanmar: Landmark routing for large scale wireless ad hoc networks with group mobility, 2000.

[QDF09] Nokia Qt Development Frameworks. Qt: Cross-platform c++ development. http://qt.nokia.com/, 2009.

[Ran95] Marcus J. Ranum. One time pad vernam cipher faq. Available at http://www.ranum.com/security/computer-security/papers/otp-faq/, 1995.

[RFIP06] Microsoft TechNet: Resources For IT Professionals. Recommendations for small office or home office wireless networks. Available at http://technet.microsoft.com/en-us/library/bb727047.aspx, 2006.

[RG00] R. Want and G. Borriello. Special issue on information appliances. IEEE Computer Graphics and Applications, June 2000.

[Riv92] Ronald L. Rivest. The MD5 Message-Digest Algorithm (RFC 1321). http://www.ietf.org/rfc/rfc1321.txt?number=1321, April 1992.

[Riv94] Ronald L. Rivest. RC4: Rivest Cipher 4. Available at http://en.wikipedia.org/wiki/RC4/, 1994.

[RRD01] Ratish J. Punnoose, Richard S. Tseng, and Daniel D. Stancil. Experimental Results for Interference between Bluetooth and IEEE 802.11b DSSS Systems. In *Proceedings of IEEE Conf. On Vehicular Technology*, Atlantic City, NJ, USA, October 2001.

[Rud01] Larry Rudolph. Project oxygen: Pervasive, human-centric computing - an initial experience. In *Proceedings of the 13th International Conference on Advanced Information Systems Engineering*, CAiSE '01, pages 1–12, London, UK, UK, 2001. Springer-Verlag.

[Sak98] K. Sakamura. Digital museum. *Journal of Information Processing Society of Japan (JIPS)*, 39(5), 1998.

[San01] Miguel Sanchez. Adaptive Power Control for Ad-hoc Networks. 5th International Conference on Systemics, Cybernetics and Informatics (SCI 2001), Orlando, Florida, July 2001.

[SC05] Daniel Steinberg and Stuart Cheshire. *Zero Configuration Networking: The Definitive Guide*. O'Reilly Media, Inc., 2005.

[SCP+04] S. Pradhan, C. Brignone, Phil Karn, J. Cui, A. McReynolds, and M. Smith. Websign: hyperlinks from a physical location to the web. Technical report, 2004.

[SFD+06] Oliver Storz, Adrian Friday, Nigel Davies, Joe Finney, Corina Sas, and Jennifer Sheridan. Public ubiquitous computing systems: Lessons from the e-campus display deployments. *IEEE Pervasive Computing*, 5:40–47, July 2006.

[SKB+98] Steve Shafer, John Krumm, Barry Brumitt, Brian Meyers, Mary Czerwinski, and Daniel Robbins. The new easyliving project at microsoft research. In *Proc. Joint DARPA/NIST Smart Spaces Workshop*, pages 30–31, 1998.

[SM06] Stuart Cheshire and Marc Krochmal. Multicast DNS. draft-cheshire-dnsext-multicastdns.txt, August 2006.

[Sma07] Smart-its: Interconnected embedded technology for smart artifacts with collective awareness. Lancaster University, ETH Zurich, 2007.

[SS02] Rob Semper and Mirjana Spasojevic. The electronic guidebook: Using portable devices and a wireless web-based network to extend the museum experience. In *In Proceedings of Museums and the Web Conference*, 2002.

[SSKK01] Mirjana Spasojevic, Mirjana Spasojevic, Tim Kindberg, and Tim Kindberg. A study of an augmented museum experience. Technical report, 2001.

[Sun01] Jun-Zhao Sun. Mobile ad hoc networking: an essential technology for pervasive computing. In *International Conferences on Info-tech and Info-net*, volume 3, pages 316–321, 2001.

[SWR98] Suresh Singh, Mike Woo, and C. S. Raghavendra. Power-aware routing in mobile ad hoc networks. In *Mobile Computing and Networking*, pages 181–190, 1998.

[TCJ03] T. Könberg, C. Öhult, and J. Delsing. Measuring breathing-and heart rate data with distribution over wireless ip networks. Instrumentation and Measurement technology conference (AIMTC 2003), May 2003.

[Tex05] Texas Instruments Inc. Msp430 family of ultra-lowpower 16-bit risc processors. http://www.ti.com/lit/ds/symlink/msp430f1611.pdf, 2005.

[THL06] In The Hand Ltd. 32feet.net: Personal Area Networking for .NET. Available at http://inthehand.com/, 2006.

[TKK08] Timothy J. Thompson, C. Bala Kumar, and Paul J. Kline. *Bluetooth Application Programming with the Java APIs Essentials Edition*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2008.

[Toh97] Chai-Keong Toh. *Associativity Based Routing For Ad Hoc Mobile Networks*. Wireless Personal Communications Journal, Special Issue on Mobile Networking and Computing Systems Vol.4 No.2, March 1997.

[Toh07] Chai-Keong Toh. Future application scenarios for manet-based intelligent transportation systems. In *Proceedings of the Future Generation Communication and Networking (FGCN'07)*, pages 414–417, Washington, DC, USA, 2007.

[Tou96] Jean Tourrilhes. Wireless extensions for linux. Hewlett Packard Laboratories, Palo Alto, 1996. Available at the author's home page at http://www.hpl.hp.com/.

[Tou08] Jean Tourrilhes. Wireless tools for linux. Hewlett Packard Laboratories, Palo Alto, 2008. Available at: http://www.hpl.hp.com/personal/Jean_Tourrilhes/.

[TP03] T. Clausen and P. Jacquet. Optimized link state routing protocol (OLSR). Request for Comments 3626, MANET Working Group, http://www.ietf.org/rfc/rfc3626.txt, October 2003. Work in progress.

[Ubi03] Ubisense, Inc. The ubisense precise real-time location system. http://www.ubisense.net, 2003.

[ULMT01] U. Hansmann, L. Merk, M. Nicklous, and T. Stober. *Pervasive Computing Handbook*. Springer-Verlag New York, Inc., January 2001.

[Var01] András Varga. The omnet++ discrete event simulation system. *Proceedings of the European Simulation Multiconference (ESM'2001)*, June 2001.

[VCT+12] M.-J. Villanueva, C. T. Calafate, A. Torres, J. Cano, J.-C. Cano, and P. Manzoni. Seamless manet autoconfiguration through enhanced 802.11 beaconing. *Mobile Information systems Journal. Status: accepted*, 8, 2012.

[VS00] V. Park and S. Corson. Temporally-ordered routing algorithm (TORA) version 1 - functional specification. Internet Draft, MANET Working Group, draft-ietf-manet-tora-spec-03.txt, November 2000. Work in progress.

[Wei91] Mark Weiser. The computer for the 21st century. *Scientific American*, 256(3):94–104, 1991.

[Wei93] Mark Weiser. Some computer science problems in ubiquitous computing. *Communications of the ACM*, 1993.

[Wen05] K. Weniger. PACMAN: Passive Autoconfiguration for Mobile Ad hoc Networks. *IEEE Journal on Selected Areas in Communications*, 23(3):507–519, March 2005.

[WHFG92] Roy Want, Andy Hopper, Veronica Falco, and Jonathan Gibbons. The active badge location system. *ACM Transactions on Information Systems*, 10:91–102, 1992.

[WiG01] WiGLE. Wireless Geographic Logging Engine. Available at http://wigle.net/, 2001.

[WSA+95] Roy Want, Bill N. Schilit, Norman I. Adams, Rich Gold, Karin Petersen, David Goldberg, John R. Ellis, and Mark Weiser. The parctab ubiquitous computing experiment. *IEEE PERSONAL COMMUNICATIONS*, 2:28–43, 1995.

[Wu02] Jie Wu. An Extended Dynamic Source Routing Scheme in Ad Hoc Wireless Networks. 35th Annual Hawaii International Conference on System Sciences (HICSS'02)-Volume 9, Big Island, Hawaii, January 2002.

[XJRY05] X. Hong, J. Liu, R. Smith, and Y. Lee. Distributed Naming System for Mobile Ad Hoc Network. In *Proceedings of International Conference on Wireless Networks (ICWN-05)*, pages 509–515, Las Vegas, USA, June 2005.

[YE03] Y. Sun and E. M. Belding-Royer. Dynamic address configuration in mobile ad hoc networks. Technical report, Computer Science, UCSB, Tech. Rep, 2003.

[YMK07] Seung Yi, Jeff Meegan, and Jae H. Kim. Network Autoconfiguration for Mobile Ad Hoc Networks. In *Military Communications Conference, 2007. MILCOM 2007. IEEE*, pages 1–7, Orlando, USA, October 2007.

[YW03] Y. Yuan and W. Arbaugh. A Secure Service Discovery Protocol for MANET. In *The 14th IEEE 2003 International Symposium on Personal, Indoor and Mobile Radio Communication Proceedings (PIMRC)*, pages 502–506, Beijing, China, September 2003.

[zer99] Zeroconf: IETF Zero Configuration Networking Working Group. http://www.zeroconf.org, September 1999.

[zig05] Zigbee specification. Technical Report Document 053474r06, Version 1.0, ZigBee Alliance, June 2005.

[Zim96] T. G. Zimmerman. Personal area networks: near-field intrabody communication. *IBM Syst. J.*, 35:609–617, September 1996.

[ZLMX06] Z. Gao, L. Wang, M. Yang, and X. Yang. CNPGSDP: an efficient group-based service discovery protocol for MANETs. *Computer Networks*, 50(16):3165–3182, November 2006.

[ZM99] Z. Haas and M. Pearlman. The zone routing protocol (ZRP) for ad hoc networks. Internet Draft, MANET Working Group, draft-ietf-manet-zone-zrp-02.txt, June 1999. Work in progress.