Università degli Studi di Pisa

Dipartimento di Informatica
Dottorato di Ricerca in Informatica

Ph.D. Thesis

Device Interoperability and Service Discovery in Smart Environments

Michele Girolami

Supervisor
Prof. Stefano Chessa

Supervisor
Dr. Francesco Furfari

Contents

List of Figures

List of Tables

List of Acronyms

CORDIAL COllaborative seRvice DIscovery ALgorithm

DLNA Digital Living Network Alliance

EC Energy Cost

EP End Point

GENA General Event Notification Architecture

HA Home Automation

HCMM Home-cell Community-based Mobility Model

KNX Konnex

LDDI Local Device Discovery Integration

MANET Mobile Ad-hoc NETwork

MSN Mobile Social Networks

NO Network Overhead

OSN Online Social Networks

QA Query Answered

QRT Query Response Time

REST Representational State Transfer

SAIL Sensor Abstraction and Integration Layer

SC Service Cache

SE Smart Environments

SIDEMAN ServIce DiscovEry in Mobile sociAl Networks

SOAP (Simple Object Access Protocol

SSDP Simple Service Discovery Protocol

UPnP Universal Plug and Play

ZB4O ZigBee APIs for the OSGi Framework

ZBD ZigBee Base Driver

ZCL ZigBee Cluster Library

ZDO ZigBee Device Object

Abstract

Smart Environments (SE), and in particular Smart Homes, have attracted the attention of many researchers and industrial vendors. In such environments, according to the Ambient Intelligence paradigm, devices operate collectively using any information describing the environment (also known as the context-information) in order to support users in accomplishing their tasks. SE devices are characterized by several properties: they are designed to react autonomously to specific events, they are aware of the context, they manage sensitive information concerning the users, they adopt a service-oriented model in order to interact with other devices, and they interact by means of various applications and communication protocols.

Cooperation with devices in SE is thus complex. This thesis deals with two problems that still represent a barrier to the development of many SE applications. The thesis examines how to interact with low-power devices, which is refereed to as device interoperability, and how to discover the functionalities that mobile devices offer, namely the service discovery problem. The first part of the thesis describes the design of ZB4O an integration gateway for low-power devices based on the ZigBee specification. The growing market for ZigBee-ready appliances makes the ZigBee specification an important technology-enabler for SE. However, accessing such devices entails an easy interaction model with IP-based networks that are already present in most SE. Therefore, this work presents an open source platform that seamlessly integrates ZigBee devices with applications running on SE. The thesis describes the evaluation process of ZB4O with various trials organized over the last year of two EU projects, as well as the integration of ZB4O with UPnP and a RESTful approach. SE devices can also export their functionalities with a service-oriented approach. In fact, every resource offered by a device can be seen as a service available for other devices. The second problem studied in this thesis is the service discovery and it deals with how to advertise and query services in SE. The scenario considered for the service discovery problem is characterized by mobile devices carried by people roaming in SE. Hence, mobility and sociality are two key-factors that make the service discovery problem more complex and challenging. The thesis presents two algorithms, termed SIDEMAN and CORDIAL, for the service discovery in Mobile Social Networks (MSN) which are evaluated with real and synthetic simulation scenarios.

Chapter 1

Introduction

Already in 1991 Mark Weiser observed that "The most profound technologies are those that disappear." [1]. In fact, increasingly miniaturized, powerful, cheap smart phones smart watches, tablets and ultra-thin notebooks have become part of our daily lives. We are surrounded by all these devices daily, at home, work and also in public spaces.

The Ambient Intelligence (AmI, late 1990s) research field embraces the Weiserian vision. AmI is the convergence of two older concepts, Ubiquitous Computing [1] and Pervasive Computing [2] that focus the attention on human needs. In particular, devices that surround us can also be used to assist our common needs, such as mobility, entertainment, health and security. In order to achieve the needs of users, systems with the AmI paradigm must provide features such as context-awareness, service orientation, personalization, adaptation and anticipatory behavior. The Smart Environments (SE) [3] are the incarnation of the such complex systems. Weiser in 1981, defined a Smart Environment as [...]*a physical world that is richly and invisibly interwoven with sensors, actuators, displays, and computational elements, embedded seamlessly in the everyday objects of our lives, and connected through a continuous network.* This definition highlights that an environment becomes *smart* when it helps the users within the environment to accomplish his/her tasks.

A concrete example of a SE is the Smart Home [4] other examples include public spaces such as hospitals, airport terminals, bus stations or (smart) cities. The EU projects DOREMI, GiraffPlus, universAAL, ReAAL and PERSONA have focused on the creation of concrete SE. A Smart Environment basically consists of three entities: humans, the devices and the software modules. Humans (or the end-users) play the key-role in a SE, since they are assisted by the services provided by devices. Devices are hardware components that are optionally equipped with sensors and actuators, and are controlled by the software modules. In the literature there are many applications based on the concept of an SE, most of which are designed to improve the quality of life of users at home, while moving, at work or during entertainment activities.

The Smart Home is the most representative example of an SE. In a Smart Home,

different types of devices are deployed in the same environment; for example appliances, sensors and actuators and general-purpose devices cooperate together offering support to the inhabitants in their daily activities. Some examples include monitoring heating systems, detecting intrusions at home, or sharing digital contents between the TV and PC.

Smart Homes are not the only representative case of SE in fact health-care domain is an emerging market of applications aimed at monitoring the health status of the elderly, patients and caregivers under three main areas: (i) monitoring specific patient's diseases; (ii) controlling the patient's gestures in order to detect failures or accidents, and (iii) reacting to anomalous health-conditions by alerting relatives or caregivers.

The perimeter of a SE can be wider than an indoor location such as a home or a hospital, indeed Smart Cities are further examples of SE that are offer advanced services for citizens, such as controlling traffic jams, monitoring flood detection and pollution agents, or advertising public transport in real-time [5].

Such application domains give rise to a set of properties and research challenges that characterize the SE:

- Autonomous devices: the complexity and dynamics of many application domains in a SE entails distributing intelligence in the system. The challenge is to design (smart) devices that autonomously react to a wide range of different situations in order to reduce human intervention.

- Scalability: the increasing number of connected devices gives rise to various scalability issues: (i) assigning an address to devices installed or dynamically added to a SE, some initiatives such as IPv6 and 6LowPan specifically focus on how to uniquely address devices in a network; (ii) data communication and networking: the goal is to manage the interconnection among a large number of devices that are based on different protocols and communication paradigms.

- Energy-awareness: devices can be disconnected from the power line, therefore they must be designed and programmed to minimize energy consumption and to maximize the battery life-cycle. The results achieved by the Green Computing paradigm [6] can be adopted. Green Computing takes into account the energy consumed during the computation, so that expensive computations are only performed when strictly necessary.

- Embedded security and privacy preservation: devices can manage sensitive information, for example the health conditions of a patient or data gathered from caregivers on the status of an elderly person. Clearly, such data must be protected against unauthorized diffusion or from malicious attacks. A SE has to consider how to protect data by adopting software and hardware countermeasures, such as deploying tamper-proof devices that provide evidence of malicious manipulations, or using public-private keys for encrypting messages exchanged among devices.

- Context-awareness: the context describes the environment and any events that occur. In a SE the devices must be aware of the context in order to react properly. For example a device providing a streaming service can adapt the quality of the video according to context-information such as the network bandwidth, the energy consumption of the device or according to specific constraints imposed by the device invoking the service.

- Interoperability: the range of devices that we have considered so far are extremely heterogeneous in terms of access mechanisms. In particular, devices come from different vendors and are designed for different purposes. They differ with respect to the hardware and the software modules, the communication protocols and the interaction paradigms that they obey. In order to design SE applications that are able to access different devices simultaneously, SE are often equipped with an inter-operable gateway that is able to integrate such devices seamlessly. An inter-operable gateway should hide all the technical complexity concerning the access of devices in the SE, by offering simplified interfaces to devices or humans willing to access such devices;

- Service orientation: the functionalities offered by devices in an SE can be considered as services offered both to humans and to other devices coexisting in an SE. Each service exposes its interfaces through which other devices can discover the service and access it by means of a service discovery and access protocol. The services can also be combined in order to form more complex and rich services. Moreover, devices in an SE can be deployed as stationary or mobile devices. Stationary devices are installed permanently in specific locations for example a smart plug, and some kinds of environmental sensors or appliances. On the other hand, mobile devices can change their position over time; for example a smart phone, a smart watch or a wristband are not deployed in SE hot-spots, but are worn by people within the SE and their mobility is tightly coupled with the mobility of the person carrying them. The numbers of mobile devices are increasing in our daily lives and thus they are even more present in the SE in which we spend most of the time. We consider that the mobility of a device affects the way and the quality of the services that are provided by devices. Indeed, a mobile device is often unplugged from the power line, it is connected to a (unreliable) wireless network and can be switched on or off without any prior notification. These affects make it more complex to discover services provided by mobile devices, which is also known as service discovery problem[7].

1.1 Research Questions and Objectives

The list of properties reported in the previous section, highlights that cooperation among devices in an SE is a complex task. This thesis addresses two research

questions that derive from the cooperation among devices, namely the device interoperability (Section 1.1.1) and the service discovery (Section 1.1.2). Device interoperability results from devices in an SE having different hardware and software features. Some devices have high hardware/software capabilities that allow them to be easily integrated with each other. Conversely, other devices are designed for very specific tasks and are poor in terms of hardware/software capabilities. In particular, low-power devices do not support complex communication protocols and hence they are not able to fully interact with more powerful devices. Therefore, we argue that low-power devices require an inter-operable gateway in order to facilitate cooperation with other devices.

Service discovery algorithms are designed to advertise the existence of services in an SE and to allow end-users (either devices or humans) to query the network for services that match with requirements (for example the quality of the service, the functionalities needed and the expected performance etc.). While for stationary devices, the discovery algorithms proposed so far have reached a reliable and accepted level (some notable examples are SSDP, Bonjour [8] and SLP [9]), such algorithms for mobile devices are still in their youth. This thesis therefore studies the service discovery problem for mobile devices commonly carried by humans in an SE. We focus on understating how the social dimension of the people in an SE can be exploited in order to make the service discovery phase more efficient. We consider the Mobile Social Networks (MSN [10, 11]) as a representative example of an SE where the social dimension of humans is particularly evident. Service discovery in MSN is introduced in more detail in Section 1.1.2.

It is worth to notice that this thesis was inspired by several EU projects on device interoperability and service discovery problems. We exploited these projects in order to test the results described in this thesis with real-world experimentations. The following two sections describe into details the objectives of this thesis.

1.1.1 Device interoperability

Devices in an SE can be split into two categories: general-purpose devices and low-power devices. The first category implements application protocols that enable them to export their functionalities with a service-oriented approach. This class of devices can thus interact with each other autonomously without any intermediary agent, for example the UPnP protocol stack. In this case UPnP-ready devices (or DLNA certified devices such as a TV, set-top-box, Media Server and Smart Phones) are able to interact with each other and to invoke the services that they provide in a seamless way.

On the other hand, improvements in electronics have led to the diffusion of low-power devices everywhere: at home, and at work in cities. Such devices have limited hardware and software capabilities, are programmed to accomplish specific tasks, and often only have one communication protocol stack. The resources installed within such devices are often not enough to implement a service-oriented

communication protocol such as CoAP or UPnP. Low-power devices thus need to be integrated by means of a gateway that is able to automatically connect heterogeneous devices together.

Product silos are prevalent in the market of devices for SE. Today most of the companies offer closed-solutions often based on a proprietary gateway designed only to integrate homogeneous devices. Such market fragmentation is one of the most important barriers to the seamless interoperability among devices produced by different vendors. Thus open standards could play a major role in pushing the interoperability towards a higher level.

The ZigBee [12, 13] industrial standard is an example of a protocol designed for low-power devices based on the IEEE 802.15.4 standard. ZigBee is designed for resource-constrained devices that cannot run more complex application protocols. Such devices need to be integrated into an SE by means of a ZigBee-based gateway that browses the ZigBee network and exports the functionalities provided by the ZigBee devices to a target network (for example to an IP-based network). Some examples of ZigBee devices are for building automation, remote control, smart energy and health care. The importance of ZigBee for SE is also highlighted by several recent works [12, 14] and [15], especially in the field of home automation, personal health-care devices, energy saving and intelligent appliances.

A suitable ZigBee gateway thus needs to guarantee the interoperability between ZigBee and other communication protocols. The design of the ZigBee gateway involves:

- *seamless integration* - ZigBee devices should be easily integrated without requiring any vendor-specific drivers;
- *interoperability* - services provided by the ZigBee devices cooperate by adopting a service-oriented interaction model.

Although several ZigBee gateways have been proposed, there are still some major limitations:

- most of the gateways rely on specific ZigBee hardware, without providing any abstraction of the ZigBee hardware itself;
- most of the gateways convert the ZigBee messages (also called ZigBee frames) to only one specific target technology and only a few of them aim at the generalization of the target network;
- not all the ZigBee gateways fully recognize the ZigBee application profiles or are able to integrate with customized ZigBee devices.

Objectives

This thesis contributes to resolving the problem of device interoperability through the design and creation of an application-level gateway aimed at integrating low-

power devices with several target networks. We show the strength of our approach for the ZigBee protocol.

Although designed for low-power devices, ZigBee has an intrinsic complexity due to its service-oriented design and the richness of its application profiles. We first design and implement an inter-operable gateway for ZigBee named ZB4O (ZigBee API for OSGi Service Platform) based on the OSGi execution model, and then we evaluate it both with qualitative and quantitative metrics. We also present various EU projects that have adopted the ZB4O gateway as a standard mechanism for interacting with ZigBee networks, and we discuss the benefits of the solution we propose in real use-cases.

1.1.2 Service Discovery

The goal of the service discovery is to allow devices to advertise the services they provide and clients to find the services they need. Many service discovery protocols have already been proposed for different purposes, including three pioneering but still widely adopted protocols: Jini, SSDP and SLP. These protocols focus mainly on administrated and IP-based networks that rely on a stable network architecture. In fact, they assume the existence of a network infrastructure ensuring permanent connectivity with every device.

SE are capitalized by the pervasive presence of mobile devices (i.e. wearable devices) which provide a lot of information but whose interaction is made more complex by mobility and sociality. These two key aspects are intimately linked, indeed the mobility of a device is affected by the sociality of the individual carrying it. More precisely, people in SE do not move randomly - their mobility is affected by at least to factors (i) the kinds of social-relationships they are involved in, and (ii) their personal activities.

Concerning the first aspect, the tendency of individuals to associate and bond with similar others introduces additional features in the way people (and hence devices) in a MSN move and behave. As discussed in [16] contacts among similar happen more frequently than that contacts among dissimilar people. For instance, race and ethnicity, sex and gender, age, religion and education are notable aggregation factors. The activities of a person, on the other hand, influence the places that such a person visits over a time frame. For example, going to work, meeting friends, attending a conference, staying at home are activities that underlie why a person goes to the office, a pub, a conference room, or home.

Objectives

This thesis contributes to service discovery by studying algorithms that take into account the mobility of devices in SE and the sociality of people carrying them. In order to reproduce human mobility and social interactions, we consider the MSN as a meaningful example of an SE. We first analyze various aspects concerning human

mobility, in order to extrapolate the requirements for designing discovery algorithms. Then we design and implement two service discovery algorithms - SIDEMAN (ServIce DiscovEry in Mobile sociAl Networks) and CORDIAL (COllaborative seRvice DIscovery ALgorithm). Finally, we evaluate both of the algorithms using real-world mobility traces, as well as synthetic traces obtained from a human mobility model.

1.2 Approach to the Research Questions

This thesis investigates two problems arising from different properties of SE, which need to be tackled with different methodologies and tools.

The goal of the device interoperability problem is to design a gateway for ZigBee devices, therefore approaching it from a software design point of view. We recall the approach adopted with a previous study named SAIL [17], Sensor Abstraction Integration Layer, and we refine SAIL with the ZB4O gateway. We use UML modeling tools to identify the components and their interactions, as well as sequence diagrams in order to define the execution flows between the ZigBee network and the gateway. We assess ZB4O by experimenting with some use-cases.

The objective of evaluating ZB4O is twofold. First, we want to assess various qualitative metrics concerning the usability of ZB4O in real-word deployments. We thus exploit the trials organized over the last few years of the GiraffPlus and universAAL EU projects. Second, we want to verify the performance of ZB4O. We therefore evaluate the integration between ZB4O and the UPnP protocol (a representative protocol for SE) and measure some widely accepted metrics for software profiling, such as CPU usage, memory occupancy, and the resources allocated.

The goal of the service discovery is to study how to advertise services in MSN. In this case, we adopt an experimental approach by studying the service discovery problem by means of visual analytics tools and simulators. More specifically, designing a service discovery algorithm for an MSN entails, first, understanding how people move and interact with each other. We therefore analyze various metrics capturing temporal features concerning encounters among people (duration of contacts, inter-contact time, distribution of contacts over time etc.) as well as the communities formed by people in the MSN. Results concerning the temporal metrics were used to design two discovery algorithms. We then perform some simulation campaigns to execute the algorithms proposed with some mobility traces, using both real-world and synthetic traces.

1.3 The Overall Architecture

Device interoperability and service discovery are fundamental building blocks for realizing most of the application domains previously described. Even if such steps are at the first look disjoint, they can be still considered as consecutive ones and

part of an overall architecture. In fact, most of the applications designed for SEs require at least two basic functionalities: integrating heterogeneous devices in a seamless way and discovering the services offered by them. In particular, the device integration allows to discover the devices available in the SE and to interact with them by means of inter-operable protocols such as HTTP or UPnP. On the other hand, service discovery algorithms enable end-users to discover services that devices in a SE provide.

In most of the application scenarios that inspired this thesis, it is possible to identify the liaison between device interoperability and service discovery. A meaningful example is represented by the SE deployed within an airport terminal. This SE is composed by a heterogeneous network made of a static and a mobile part. Static devices are connected through a reliable network connection like a wired Ethernet connection. The mobile part of the SE is composed by pocket devices carried by people within the terminal and hence their mobility follows the mobility of their owners. The cooperation among pocket devices happens both with the resilient WiFi network but also with ad-hoc networks such as Bluetooth and Wifi Direct interfaces. Such highly dynamic SE requires firstly to make inter-operable the static part of the network so that different kinds of devices can cooperate. To this purpose an integration gateway could ease the interaction between sensors for monitoring the temperature of the terminal and actuators for controlling the heating system of the terminal. Secondly, both the static and the mobile part of the SE provides services targeted to the end-users. The static part is supposed to offer permanent services such as monitoring the physical conditions of the terminal, sharing the Internet connection, accessing facilities like fax, printer. Differently, the mobile devices offer services that are temporary, in particular they are available until the person carrying the device moves away from the terminal. Some meaningful examples are services provided by smart phones for sharing media contents like pictures or movie clips, but also services for sharing the weather forecasts or the time table of departures and arrivals. To this end, mobile devices have to adopt a service discovery algorithm able to cope with the mobility of the nodes and also with the sociality of individuals carrying them.

1.4 Structure of the thesis

The remainder of this thesis is structured as follows:
Chapter 2 reviews the background concepts and the state of the art concerning both of the research questions addressed in this thesis, namely the device interoperability and service discovery. The results of this chapter have been published in [18, 19, 20].
Chapter 3 presents the ZB4O gateway as well as the experimentation phase. We present use-cases in which we evaluate both qualitative and quantitative evaluation metrics. The results presented in this chapter have been published in [21, 22, 23, 24, 25, 26, 27].

Chapter 4 presents the design of SIDEMAN and CORDIAL, and explains how these algorithms exploit the MSN described in Chapter 2.2.1.
Chapter 5 presents the performance evaluation of the discovery algorithms described in Chapter 4. First we describe some features of the selected mobility traces, and then the evaluation metrics adopted, as well as the discovery algorithms used for a performance comparison. Finally, the chapter presents the results obtained both for SIDEMAN and CORDIAL with real-world and synthetic mobility traces.The results presented in Chapter 4 and of Chapter 5 have been published in [28, 29, 30, 31, 32, 33].
Chapter 6 presents the conclusions for device interoperability and service discovery in the MSN as well as a description of future research.

Chapter 2

Background and Related Works

This chapter introduces the background concepts and related works both for the research questions described in Chapter 1, namely device interoperability and service discovery. Concerning the device interoperability, this chapter surveys the ZigBee [13, 34] specification and the OSGi [35] model, two technologies that play a key role for the device interoperability in Smart Environments. Then, two inter-operable middleware for Smart Environments resulting from the GiraffPlus and universAAL EU projects are described. As part of the consortium of both projects, we participated in the design and the implementation phases, and we exploited their trials to validate the solution to the device interoperability described in this thesis. We describe the UPnP and the REST protocols as two notable examples of inter-operable protocols for SE which we also use for further experimentations with our solution. Finally we survey the most noticeable integration gateways designed for ZigBee and we discuss their limitations and weaknesses. Concerning to service discovery, the chapter first surveys Mobile Social Networks (MSN) as being representative of SE, where mobility and sociality are two characterizing aspects. We describe the MSN architectures and how to detect communities in MSN. We then present the kinds of services that are potentially available in MSN and review popular service discovery architectures. We introduce the service discovery problem as a 4-step process. For each step we review the most important results achieved so far. The review of the state of the art of the service discovery problem is discussed at the end of this chapter and it has been published in [18].

2.1 Device Interoperability

2.1.1 Background

The ZigBee Specification

The ZigBee specification defines low-power wireless network [13, 34] based on the IEEE 802.15.4 standard. An overview of the ZigBee protocol stack is shown in Figure

2.1. The ZigBee specification defines the roles of devices in the network. A ZigBee

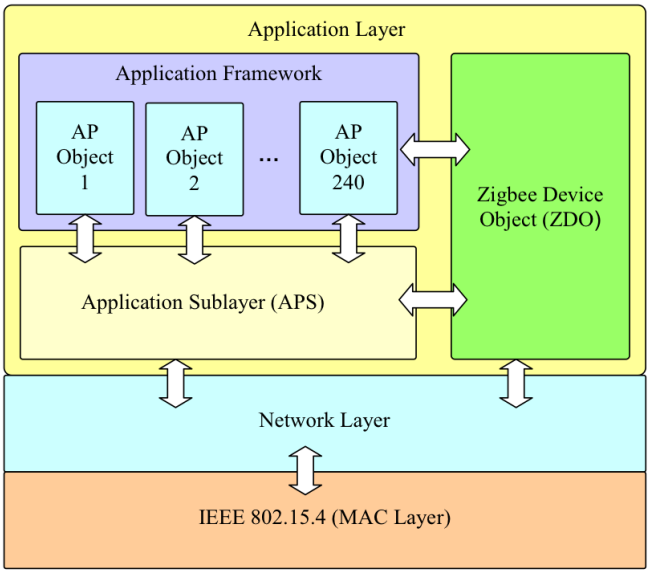


Figure 2.1: The ZigBee Stack.

end-device corresponds to an IEEE RFD (Reduced Functional Device) or FFD (Full Functional Device) device. A ZigBee router is an FFD with routing capabilities. The ZigBee coordinator is an FFD managing the whole network. Besides the star topology (that naturally maps to the corresponding topology in IEEE 802.15.4), the ZigBee network layer also supports more complex topologies like the tree and the mesh. Figure 2.2 shows examples of these topologies. Among the functionalities provided by the network layer there are: multi-hop routing, route discovery and maintenance, security and joining/leaving a network.

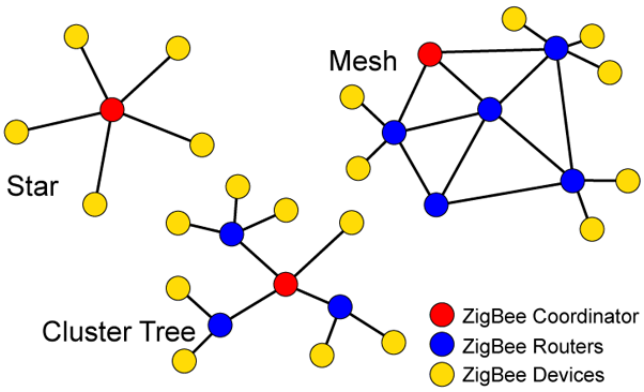


Figure 2.2: The ZigBee network topology.

A ZigBee network is formed according to the join procedure. When a device wishes to join to an existing network, the network layer is requested to start a

network discovery procedure. With the support of the scan procedure of the MAC layer, a node learns about neighboring routers that announce their information to the networks. After the upper layer has decided which network to join (several ZigBee networks may overlap spatially, using different channels), the network layer selects a parent node from his neighborhood, and asks to the MAC layer to start an association procedure. Upon receiving an indication of the association request from the MAC layer, the parent's network layer assigns to the node that is joining a 16-bit short address and lets the MAC layer successfully reply to the association request. Node that is joining will use the short address for any further network communication.

The routing algorithm adopted by the ZigBee nodes depends on the topology used in the sensor network. In a tree topology the routing can only happen along the parent-child links established as a result of join operations (this is called "tree-based routing"). Routers maintain only their address and the address information associated with their children and parents. Given the way the network addresses are assigned, a router that needs to forward a message can easily determine whether the destination belongs to a tree rooted at one of its router children or at one of its end-device children. If so, it routes the packet to the appropriate child; otherwise it routes the packet to its parent.

Route discovery is a process required to establish routing table entries in the nodes along the path between two nodes wishing to communicate. A Route Discovery Table (RDT) is maintained by routers and the coordinator to implement route discovery. Route discovery in ZigBee is based on the well-known Ad hoc On Demand Distance Vector routing algorithm (AODV)[36]. When a node needs a route to a certain destination, it broadcasts a route request (RREQ) message that propagates through the network until it reaches the destination. As it travels in the network, a RREQ message accumulates (in one of its fields) a forward cost value that is the sum of the costs of all the links traversed. The cost of a link can be set to a constant value or it can be dynamically calculated based on a link quality estimation provided by the IEEE 802.15.4 interface. Each RREQ message carries a RREQ ID field which the originator increments every time it sends a new RREQ message. In this way the RREQ ID and the source address of a message can be used as a unique reference for a route discovery process. The reception of a RREQ triggers a search within the RDT for an entry matching with the RREQ message. If no match is found, a new RDT entry is created for the discovery process and a route request timer is started (upon timer expiration the RDT entry will be removed). Conversely, if an entry is found in the RDT the node compares the path cost of the RREQ message and the corresponding value in the RDT entry. If the former is higher, then it drops the RREQ message, otherwise it updates the RDT entry. Finally, if the node is not the route discovery destination, it allocates an RT entry for the destination, with status Discovery, and rebroadcasts the RREQ after updating its path cost field. If the node is the final destination, it replies to the originator with a route reply (RREP) message that travels back along the path.

The application layer of the ZigBee stack comprises the Application Support Sublayer (APS), the ZigBee Device Object (ZDO) and the Application Framework. The APS provides the transport layer functionalities. The Application Framework contains a number of user-defined Application Objects (APOs) (also called application-level devices). The APOs implement the ZigBee applications. The ZDO provides services that allow the APOs to organize themselves into a distributed application. Each APO is uniquely identified by the network address of the hosting network-level device (ZigBee node) and by an EndPoint number (EP). Hereafter the hardware devices are referred as ZigBee nodes, and the APO as ZigBee devices. To enable interoperability of nodes from different manufacturers, the ZigBee Alliance defines the application profiles and the clusters. The application profile is a collection of device descriptions that form a cooperative application. For instance, the Home Automation Profile describes some kinds of ZigBee devices such as the Thermostat, Pump, and Pump controller devices. A device is described as the set of clusters (that are specification of messages) that it can manage. In turn, a cluster is defined as a collection of commands and attributes (data entities which represent a physical quantity or state). Clusters are defined in the ZigBee Cluster Library (ZCL) [37] and each cluster can appear in different profiles. The ZDO provides to the ZigBee device and service discovery functionalities. Device discovery allows an APO to obtain the network address of other nodes. The service discovery exploits the cluster descriptors and the cluster identifiers to determine the capabilities offered by the APOs.

ZigBee also defines a binding mechanism used to connect the APOs. When an APO sends a message with a specific cluster identifier, the message is automatically routed to a set of APOs according to the binding table.

The OSGi model

The OSGi specification [38] defines a service oriented, component-based model for Java developers. This specification defines the component (or OSGi bundles) life-cycle. In particular a component can be installed, removed, started, and stopped at run time. An OSGi bundle is a jar file that contains Java classes, resources and metadata describing the dependencies with other bundles. The main features that OSGi offers are:

- a service model where every component can be registered as service in a registry;
- an execution environment where multiple components can run on the same virtual machine;
- a set of APIs for the control of the component life-cycle;
- a secure environment where multiple components coexist without affecting each other;

- a cooperative and distributed environment where components can discover other components.

The OSGi bundles, wishing to detect the presence of a particular OSGi service, send a service subscription to the Service Registry. As soon as the service needed is available, the OSGi Service Registry notifies the requester, Figure 2.3 shows the service registration mechanism.

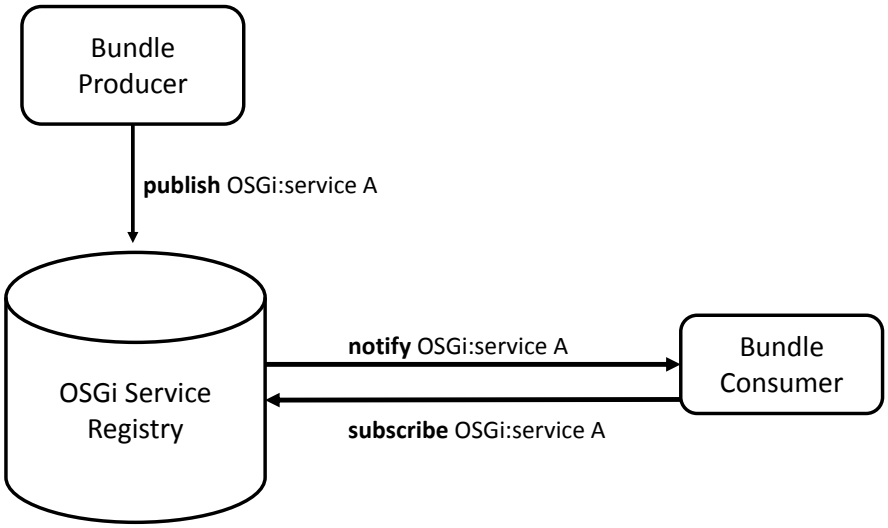


Figure 2.3: The OSGi service model.

Although OSGi was initially thought as a Java platform for realizing LAN/WAN gateways, it became popular also in many other fields. OSGi is now considered the dynamic module system for Java thanks to the introduction of the Java Specification Request 291.2. In 2010, the OSGi Alliance has extended the OSGi specification with the OSGi Remote Services Specification (O-RSS). O-RSS defines how to extend the service registry mechanism to discover and access OSGi services deployed on a remote host. The O-RSS is available with a reference implementation (the DOSGi implementation[1]) that implements the discovery and the access mechanisms based on SOAP over HTTP (or RESTful JAX- RS see section 2.1.1) and the WSDL contracts. A further step forward has been done by some interesting projects [39] that not only focus on the cooperation among distributed OSGi instances, but also on the cooperation among devices and objects that implement the DPWS specification.

The OSGi architecture is composed by several layers running on top of a native OS and a Java Virtual Machine, Figure 2.4 shows the layered architecture.

The OSGi specification is implemented with several OSGi Execution Environment, such as Felix, Karaf and Equinox. All of them adhere to the OSGi specification [35], however some differences are still present among the distributions.

[1]http://cxf.apache.org/distributed-osgi.html

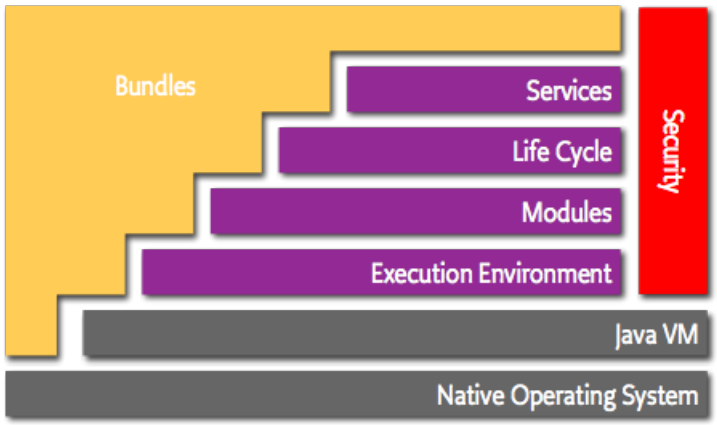


Figure 2.4: The OSGi architecture.

Concerning the Module layer, OSGi defines every software module as an OSGi bundle. A bundle is the atomic software packaging unit for the OSGi framework, the bundles are distributed as Java ARchives (JAR) files and they contain the following resources:

- one or more Java classes organized as Java packages, these classes may implement zero or more OSGi services;
- the Manifest file containing the meta-data describing the bundle properties;
- other resources such as: documentation, source code and extra files.

The bundle life cycle is described with a set of execution states, the transitions among states are performed by the bundle itself, when some events occur, or by other bundles that have the right permissions to change the status. The set of possible states are:

- installed: the OSGi bundles may require some external resources such as software dependencies and configuration files or media contents. The OSGi framework installs an OSGi bundle if all the resources required are available;
- started/stopped: the bundle is activated and de-activated;
- updated: the bundle execution is stopped, and the bundle code, as well as the resources, are replaced with a new version of the OSGi bundle;
- uninstalled: the bundle is stopped and the code and its resources are removed from the system.

Figure 2.5 shows all the status previously described as well as the transitions.

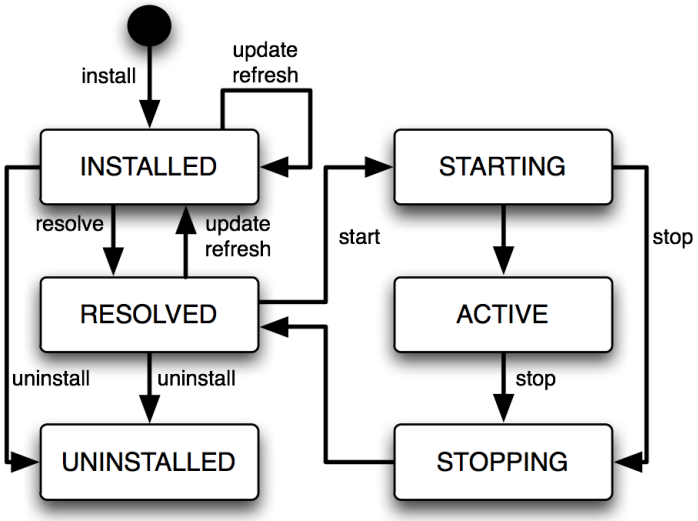


Figure 2.5: The OSGi bundle life cycle.

Another important layer of the OSGi architecture is the Service registry (see Figure 2.5). It implements a cooperation model for bundles. In particular, bundles can cooperate via traditional class sharing but class sharing is not compatible with dynamically installing and un-installing code. Suppose at time t the bundle $B0$ provides the Timer Java class. The Timer class provides methods for manage a simple timer such as start, stop, set and get; at time $t+1$ the bundle $B1$ tries to reference the Timer class, but $B0$ has been already un-installed from the framework, therefore $B1$ catches an exception. A more flexible solution is offered by the OSGi Service Platform that provides the bundle Service Registry to register service objects. A service object is a Java class implementing a set of methods, OSGi defines a rich collection of events to discover new or existing services. With this approach bundle $B1$ checks if a Timer service is available in the framework, only if this operation succeeds $B1$ invokes the Timer service, otherwise $B1$ stops its execution.

The Security layer is a crossing layer of the OSGi Service Platform. The Security layer introduces a set of security policies to provide a trusted execution environment. This set is made of Java 2 Code Security, Package Permission and Service Permission providing mechanisms to developers to design and to implement trusted bundles and ensuring a correct interaction between them. OSGi adopts Java 2 Code Security to protect resources from specific actions, Java 2 Code Security introduces the Permission classes keeping trace of the resources and the available actions on each resource. Every bundle has a set of grants, this set can be modified by adding or by removing grants at run-time; when a bundle invokes the action for the resource x (eg. x is a file) the Security Manager checks if the bundle has the required grant.

Finally, the Service layer defines a vast collection of services in terms of Java interfaces, such services are fully described in [35].

Integration Middleware for Smart Environments

This section surveys GiraffPlus and universAAL projects as notable examples of middleware specifically designed for Smart Environments. Both of the projects take into account the specific requirements coming for AAL (Ambient Assisted Living) scenarios but with two different approaches. GiraffPlus deeply focuses on some specific use-cases based on the Giraff robotic agent for tele-presence. Conversely, universAAL adopts an holistic approach, it provides not only the reference architecture and its implementation, but also tools and services for the design and implementation of AAL-based applications.

Both of the projects require to integrate resource-constrained devices in order to achieve the use-cases needed. In particular, among the project requirements, we cite:

- being able to integrate resource-constrained devices also based on the ZigBee protocol;
- avoid the integration with one single ZigBee vendor, rather adopt an extensible solution;
- the integration with the ZigBee protocol must be designed so that it is possible to interact with ZigBee devices by means of standard communication protocols (such as HTTP and MQTT) as well as platform-specific protocols (such as jGroups).

ZB4O meets all the previous requirements and what thus used in both GiraffPlus and universAAL projects. The next two subsections give an overview for both of the architectures while section 3.3 describes how ZB4O extends them.

The GiraffPlus project

The GiraffPlus system [40] has been developed in the GiraffPlus FP7 project and addresses some challenges. A first issue is the early detection of some possible deteriorations of the health conditions of a patient, in order to detect in advance anomalies in the early stage and to involve timely relatives or caregivers. A second issue is to provide adaptive support which can offer services to assist in coping with age-related impairments. Third, ways of supporting preventive medicine must be found as it has been increasingly recognized that preventive medicine can contribute to promote a healthy lifestyle and delay the onset of age-related illnesses. The GiraffPlus system consists of a network of home sensors that measure e.g. blood pressure or temperature, or detect e.g. whether somebody occupies a chair, falls down or moves inside a room. The data from these sensors are interpreted by an intelligent system in terms of activities, health and wellbeing: e.g. the person is exercising or the person is going to bed, or a fall has occurred. These activities can then trigger alarms or reminders to the primary user or his/her caregivers, or be analyzed off line and over time by a health professional. The main component of the system is

a telepresence robot, the Giraff robot, which can be moved around in the home by somebody connected to it over Internet. The Giraff robot is an example of mobile robotic telepresence technology [41] and it is effectively a mobile communication platform, equipped with video camera, display, microphone and speakers, and a touch screen. When a remote visitor uses the Giraff robot as a communication tool, what has happened in the home in terms of activities and the physiological measurements of the person can be seen and analyzed. Both the remote visitor and the person in the home have access to the information and the system can be modified to assess other aspects with the agreement of the primary user. The robot uses a Skype-like interface to allow caregivers to virtually visit an elderly person in the home. The Giraff robot is enhanced with semi-autonomy in order to increase safety and ease-of-use. The GiraffPlus system also includes a network of sensors. Data from these sensors are processed by an advanced context recognition system based on constraint-based reasoning, which both detect events on-line and can perform inference about long term behaviors and trends. Personalized interfaces for primary and secondary users are developed to access and analyze the information from the context recognition system for different purposes and over different time scales. An important feature of the system is an infrastructure for adding and removing new sensors seamlessly, and to automatically configure the system for different services given the available sensors.

The GiraffPlus Architecture

We now present the specification of the GiraffPlus system in terms of components, functionalities and interfaces among components. Also, we describe how the components are integrated and interfaced with the rest of the system. Figure 2.6 shows an abstract component diagram of the GiraffPlus system. In particular, the system is composed by three building blocks: (a) the Physical Environment and Software Infrastructure, (b) the middleware Infrastructure and (c) the Service Layer. The Physical Environment and Software Infrastructure coupled with the middleware Infrastructure represent the basic modules of the GiraffPlus system. Indeed, all the data services are grounded on them; the middleware infrastructure is also responsible for the inter-operable communication service. In particular, the middleware Infrastructure constitutes a gateway shared among all the system components. Then, the Sensor Network, composed by both physiological and environmental sensors, gathers the information generated at home as well as it provides the data collected. Finally, the telepresence robot provides the GiraffPlus social interaction functionalities enabling remote access in the environment through a pilot software embedded in the visualization and interaction services. The Long-Term Data Storage component is responsible for providing a general database service for all the data generated. Specifically, the main role of this component is to manage a database containing all the data collected through the middleware Infrastructure and generated by the Sensors Network. The Context Recognition and Configuration Planning are responsible for context/activity recognition and system configuration planning, i.e., two high-level reasoning systems in charge of respectively implement-

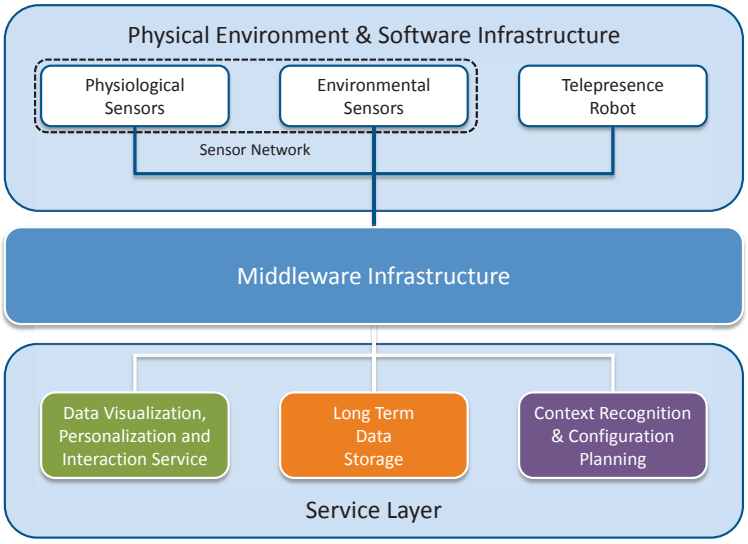


Figure 2.6: The GiraffPlus components.

ing the monitoring activities by means of context/activity recognition and providing suitable configuration settings for the Sensors Network according to the requested monitoring activities. Finally, the Data Visualization, Personalization and Interaction Service is the part of the system responsible for creating user-oriented service.

The universAAL project

The EU-FP7 project universAAL[24] was set up in 2010. Its main goal is the design and development of an open platform that provides a standardized approach for an easy and economic development of AAL (Ambient Assisted Living) solutions. The universAAL platform benefits:

- the end-users, i.e. older adults and people with disabilities, their caregivers, and family members, by making new solutions affordable, simple to configure and to deploy;

- the solution providers, by making it easier and cheaper to create innovative AAL services or to adapt existing ones by using a compositional approach based on existing components, services, and external systems;

- the authorities with responsibility for AAL by introducing a standardized approach for the development of AAL solutions.

universAAL adopts a holistic approach, the whole architecture is shown in Figure 2.7. The system provides support in three main areas: runtime support, development support, and community support. The runtime support provides the reference

model, the reference architecture and a concrete architecture implementing the universAAL specifications. The developer support provides a set of resources for developers such as guidelines, development tools, training and other materials useful to easy the integration with universAAL. The community support deals with tools and facilities for the open source communities. Such tools aim at diffusing as much as possible the universAAL approach and also to create a possible market of AAL applications.

The universAAL project has a strong focus on standardization activities. As for example, the universAAL Framework for user interaction has obtained the official IEC PAS status for its specification. It is documented as a Publicly Available Specification by the International Electrotechnical Commission (IEC) with the reference IEC/PAS 62883 Ed. 1.0 and titled The universAAL Framework for User Interaction in Multimedia Ambient Assisted Living (AAL) Spaces Similarly, the universAAL model and the universAAL architecture aims at follow the same path as the User Interaction model.

Moreover, the universAAL project has attracted the attention of the EU commission and it has been selected for a wide camping of experimentations with a massive number of end-users. To this purpose, the ReAAL FP7 EU project [2] has the goal of testing several market applications based on the universAAL platform with more than 5000 users in 6 countries.

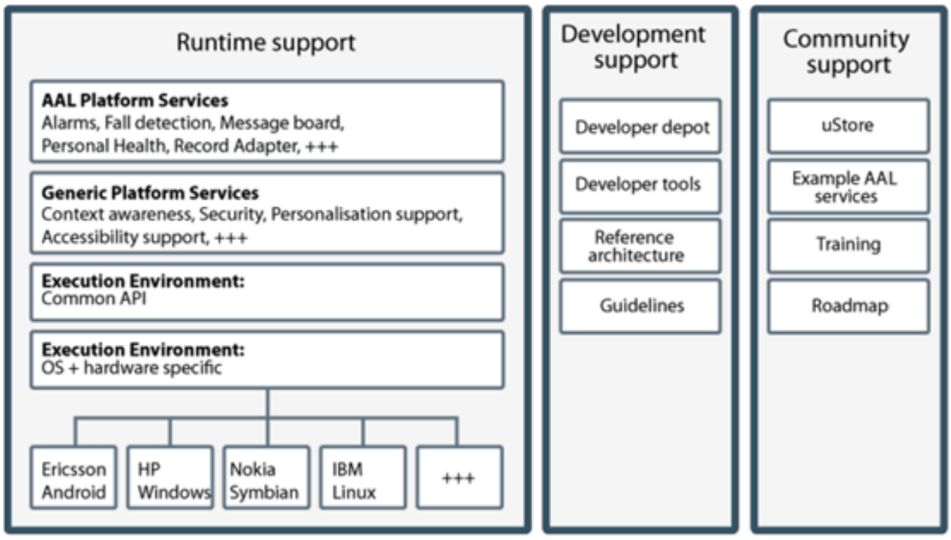


Figure 2.7: The universAAL system overview.

The universAAL architecture

The universAAL architecture has been designed, tested and implemented with the Runtime Support (RS). The RS is one of the building block of the universAAL

[2] http://www.cip-reaal.eu/about/project-description/

eco-system, it consists of a set of software components that can be installed on different types of hardware such as smart phones, laptops and tablet PCs. Moreover universAAL supports the integration of three low-power technologies, namely ZigBee, Bluetooth and FS20. The RS it is composed by different building blocks as shown in Figure 2.8.

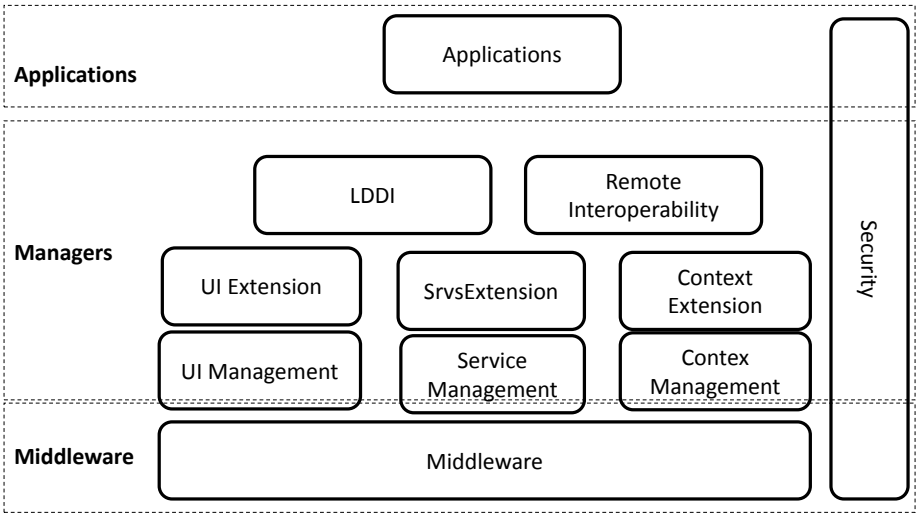


Figure 2.8: The universAAL components.

The building blocks of the RS are:

- the middleware: it acts as a broker between the nodes (also known as AAL-aware nodes). In particular, it handles the exchange of messages among nodes and it hides the heterogeneity of the hardware devices. It is decomposed by different sub-blocks: the Container that provides an abstraction to the execution environment used for running universAAL, the Discovery and Peering responsible for discovering nodes and to build an overlay-network among nodes (the AAL Space), Communication and Data Representation Model implementing the communication primitives;

- the Local Device Discovery Integration (LDDI): it implements a sensor and abstraction layer in order to integrate sensors and actuators with the universAAL platform. The universAAL RS supports the most important assistive technologies such as Konnex, ZigBee, Continua devices via Bluetooth and a prototype integration of FS20;

- Context Management: it deals with the management of the context information. The context information is used by the AAL Services to adapt themselves to the environment or the end-users status;

- service Management: it deals with the definition and implementation of the universAAL service infrastructure, in particular it offers a service-oriented model to the functionalities provided by the AAL applications.

- User Interaction (UI) Management: it offers a framework for the user interaction that separates the content exchanged between the users and the applications from its actual presentation;

- Remote Interoperability: it implements a mechanism for remotely administrate an AAL Space, for example to remotely configure or install an application;

- Security provides for trust, privacy-awareness and access control. This building block addresses the protection of the privacy of personal data related to people involved in the universAAL experimentation. As a notable example, the security components aims to prevent unwanted disclosure of health details, personal preferences, habits, and lifestyle leads to discrimination.

The RS implements a service-oriented cooperation model based on two core communication bus: the context and the service bus. The context-bus is used by any application running on top the universAAL in order to publish context events. The events are published without specifying a receiver, rather any other AAL-aware node can be notified asynchronously about the events generated. The nodes generating context events are named context publisher, while nodes receiving events are context subscribers. Examples of context events are the installation of new devices in the Smart Environments, any kind of sensor readings, the anomalies of the devices or more complex events such as the health status of an elderly and the posture/activity of a person at home. The context bus is implemented by the Context Management building block.

The service bus is used to register any functionality that an AAL-aware node wants to share with any other node. As an example, an AAL-aware node can register a health monitor service in order to control remotely a patient. Services are registered by announcing the service profile on the service bus. The service profile is also named service advertisement and it contains some information about: the kind of service, the node providing the service and how to invoke the service. A node willing to access a service, first discovers the service by querying the service bus with a service query.

Service Interoperable Protocols

The UPnP protocol

UPnP (Universal Plug and Play) [42] is a widely-accepted protocol designed to enable the integration of consumer electronics, intelligent appliances and mobile devices from many different vendors. UPnP defines how to interact with UPnP

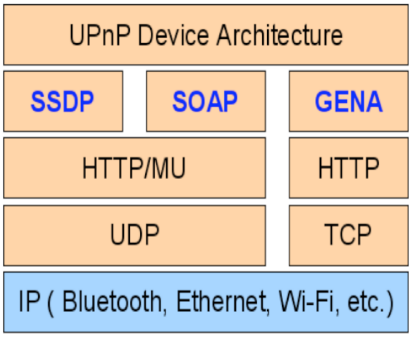


Figure 2.9: The UPnP protocol stack.

devices and the Digital Living Network Alliance (DLNA) standard[3] defines guidelines for the standardization of the profiles and of the content offered by devices (including multimedia devices) of different manufacturers. Under this respect the two standards are complementary: UPnP offers management services and DLNA standardizes the profiles and contents. UPnP offers services for the management of intelligent devices in the home environment, it is adopted by many (multimedia) devices already on the market, such as televisions, hi-fi, video cameras, etc., and it is the de-facto solution for infotainment. The UPnP technology is an architecture for unmanaged peer-to-peer networks composed by intelligent devices (Routers, PCs, Videocamera, TV, etc.) in local network environments. The goal of UPnP is to create a vendor independent (but vendor extensible), user friendly and robust standard. In order to achieve the previous goal the UPnP Forum (the industry initiative promoting UPnP) defined the UPnP protocol stack shown in Figure 2.9. It provides some features such as device and service discovery, device description, device control, eventing management, and presentation of the devices through user friendly interfaces (using HTML). The UPnP forum defines a set of standard profiles describing different categories of devices. Up to now the forum identified, among others, the categories of Basic, Audio/Video (aimed at standardization of multimedia devices), Home Automation (for domotic applications), Remoting (for intelligent remote commands) etc. From a network point of view UPnP can be seen as a layer constructed over HTTP in a standard TCP/IP and UDP/IP network. UPnP exploit the SSDP protocol (Simple Service Discovery Protocol) to support discovery, SOAP (Simple Object Access Protocol) to support control of devices, and GENA (General Event Notification Architecture) to enable listening and event management as showed in Figure 2.10 and 2.11.

The REST Style

REST stands for REpresentational State Transfer and it implements a widely accepted Web Service paradigm based on the HTTP protocol [43]. In the REST

[3]http://www.dlna.org/en/industry/home/

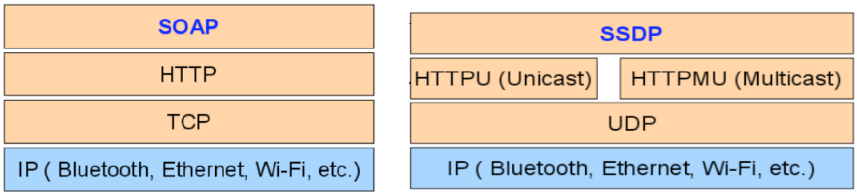


Figure 2.10: The UPnP stack for discover and control phases.

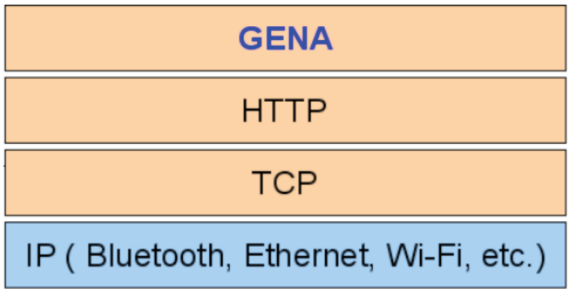


Figure 2.11: The UPnP stack for event phase.

architectural style, data and functionality are considered resources and are accessed using Uniform Resource Identifiers (URIs), typically links on the Web. The resources are acted upon by using a set of simple, well-defined operations. The REST architectural style constraints an architecture to a client/server architecture and it is designed to use a stateless communication protocol, typically HTTP. In the REST architecture style, clients and servers exchange representations of resources by using a standardized interface and protocol. The following principles characterize the RESTful services:

- Resource identification through URI: a RESTful web service exposes a set of resources that identify the targets of the interaction with its clients. Resources are identified by URIs, which provide a global addressing space for resource and service discovery.

- Uniform interface: resources are manipulated by using four operations: create, read, update and delete implemented with the four HTTP commands, namely PUT, GET, POST, and DELETE. PUT creates a new resource, which can be then deleted by using DELETE. GET retrieves the current state of a resource in some representation. POST transfers a new state onto a resource.

- Self-descriptive messages: resources are decoupled from their representation so that their contents can be accessed in a variety of formats, such as HTML, XML, plain text, PDF, JPEG, JSON, and others. Metadata about the resource is available and used, for example, to control caching, detect transmission errors, negotiate the appropriate representation format, and perform authentication or access control.

- Stateful interactions through hyper links: every interaction with a resource is stateless; that is, request messages are self-contained. Stateful interactions are based on the concept of explicit state transfer. Several techniques exist to exchange state, such as URI rewriting, cookies, and hidden form fields. State can be embedded in response messages to point to valid future states of the interaction.

2.1.2 Related Works

The ZigBee gateways have been subject of intensive, independent studies in the recent past [14, 15] and [44]. Some of these works are mainly addressed to the design and the implementation of gateways for protocol translation. The first specification of a ZigBee Gateway is proposed by the ZigBee Alliances in late 2011 [45]. The idea is to expose a web service (typically based on SOAP/REST) for every ZigBee device. Such web-service represents an end-point for the external applications in the IP network that needs to interact with the ZigBee device, for example to query the state of a device or to request a service. The queries to ZigBee devices are expressed in terms of XML-based messages, and they are translated into the appropriate cluster by the gateway. Consequently, the external applications have to comply with the format of ZigBee messages as well as with the protocol mechanisms.

Lee et al. [14] describes a gateway between Konnex (KNX) and ZigBee that translates KNX telegrams into ZigBee frames and vice-versa. The implementation relies on a multi- component gateway able to define such a mapping, but it does not provide any general solution for the node interaction or the hardware abstraction. The solutions provided by Kawamoto et al. [46] and Kim et al. [47] are both focused on the implementation of UPnP (DLNA) to ZigBee gateway. Kawamoto et al. [46] take into account only the integration of ZigBee in the DLNA networks. One of the components of the gateway is responsible for creating virtual UPnP devices as soon as it acquires relevant information on the ZigBee network. In a similar way, it creates virtual ZigBee application objects for every UPnP device found. This approach has the limitation that the gateway must be configured as the ZigBee network coordinator. Furthermore it does not implement an abstraction of the ZigBee nodes.

A similar approach is described by Kim et al. in [47]. The authors implement an inter-networking gateway between UPnP and ZigBee focusing on the discovery mechanisms. In this solution the key components are the ZigBee Device Manager (ZDM) and the Virtual UPnP Proxy Manager (VUPM). The ZDM controls the ZigBee nodes, it reflects all the relevant changes to the VUPM. The VUPM is responsible for creating or removing Virtual UPnP Proxy (VUP) as soon as some events occur on the ZigBee network. The authors provide a mapping between the meta-data of the ZigBee devices and the ones of VUP. With respect to the DLNA-ZigBee Gateway [46], in this paper the authors discuss how the ZigBee devices are abstracted by the VUPs, however both of the mentioned solutions strictly rely on

specific ZigBee hardware without introducing any hardware abstraction layer.

Guozhen Hu [48] proposes a gateway for protocol translation that maps the IEEE 802.15.4 layers to the corresponding Ethernet ones. However, this gateway does not abstract the ZigBee protocols and it does not provide any hardware abstraction mechanisms. Peng et al. [49] provide a web-sensor gateway to enable the access to a ZigBee network by means of a web browser, and a web server that maps ZigBee devices in dynamic web pages. This solution requires that the end-users addresses the nodes of the WSN individually. De Silva et al. [50] present a vertical solution based on SmartBee. SmartBee lets ZigBee nodes be accessible via multi-channel solutions (by means of regular web interfaces, mobile applications etc.). On the other hand this solution is bound to SmartBee, which is a significant requirement for a general-purpose solution.

The approach proposed by Young-Guk Ha [15] models the ZigBee devices as OSGi services. In this work, the gateway acts as the ZigBee coordinator, and it assumes that nodes in the network periodically announce their ID and profile to the gateway. In turn, the gateway downloads a software service for new ZigBee devices (wrapped within an OSGi bundle) and registers this service in OSGi.

The solution presented in [44] describes a gateway for the integration between ZigBee and the OSGi framework. This solution is composed by 3 main OSGi bundles. The ZKernel manages the home appliances, the events and the warning messages moreover the Zkernel performs log routines. The ZPanel provides a graphical interface of the system management and it provides the contact point between remote clients and the ZKernel. Lastly the Updated Bundle is in charge of system maintenance and of the update of ZKernal and ZPanel. This software module is able to recognize ZigBee devices that implement some ZigBee standard clusters. Moreover it is possible to extend the module with support for custom ones. The authors provide support for Sensing and Measurement, Lighting, HVAC, Security and Safety profiles. This work implements a Web-based GUI to access to the ZigBee network via HTTP protocol. However, it seems not possible to extend the software with other kinds of connectors. Moreover the ZigBee module interacts only with a XBee-based device attached to the PC without any hardware abstraction layer.

2.1.3 Discussion

We identify the criteria to classify the ZigBee gateways. Table 2.1 summarizes the ZigBee gateways according to the previous criteria, in particular:

- Gateway architecture: either service-oriented (where the ZigBee devices are modeled as services), multi-component (where the ZigBee devices are accessed by interacting with a component of the gateway), or mixed solutions;

- Integration mechanisms: how the ZigBee network can be accessed from outside network and which technologies can be used for the interaction;

		[14]	[46]	[47]	[48]	[49]	[50]	[15]	[44]
System features	**Gateway architecture**	Component	Component	Component	Component	Mixed	Service	Component	Component
	Integration mechansim	ZigBee to Knx	ZigBee to UPnP	ZigBee to UPnP	ZigBee to Ethernet	ZigBee to Web service	ZigBee to Web service	ZigBee to Web services	ZigBee to OSGi services
	Hardware abstraction	✗	✗	✗	✗	✗	✗	✗	✗
	Device abstraction	HA Profile	HA Profile	HA Profile	No ZigBee profile	No ZigBee profile	HA Profile	No ZigBee profile	Support custom profile
	Deployment constraints	✓	✗	✗	✓	✗	✓	✓	✓
	Extension mechansism	✗	Future work	✗	✗	✗	✗	✗	✓

Table 2.1: Comparative tables of ZigBee gateways.

- ZigBee hardware abstraction: how the gateway abstracts from specific ZigBee hardware;
- ZigBee device abstraction: how the gateway manages the ZigBee devices that adhere to a standard or custom ZigBee profile;
- Extension mechanisms: how the gateway discovers and manages customized ZigBee devices.
- Deployment constraints: the possibility of deploying the ZigBee gateway on resourced-constrained nodes (e.g. Raspberry PI or BeagleBoard).

The solutions presented in Table 2.1 are representative examples of architectures for the integration of the ZigBee protocol with other protocols. However, such solutions present some limitations leading us to consider the design and the implementation of the solution presented in Chapter 3.

Most of the papers presented in Table 2.1 adopt an integration mechanism that allows to convert the ZigBee messages to only one specific target technology. As for example, [14] converts ZigBee messages to Konnex frames while [46] and [47] allows the integration between ZigBee and the UPnP protocol. Under this respect, we consider that an integration gateway has to overcome such first limitation by offering a dynamic mechanism to integrate ZigBee devices with various protocols.

Moreover, all the solutions previously discussed do not implement a hardware abstraction mechanism in order to be independent from the hardware vendor. The absence of hardware abstraction means that the gateway is intrinsically bound with

a hardware vendor and this reduces the portability of the solution to application scenarios in which ZigBee devices come from different manufacturers.

The third limitation of the solutions presented so far concerns the compliance of the gateway to the ZigBee profiles. Indeed, the ZigBee Alliance is following a standardization initiative in order to cluster devices withing a set of profiles. Each profile lists the features that all devices must provide. Gateways that do not recognize the profiles will integrate only a subset of the features that such device actually provide, hence reducing the potentiality of such devices.

Furthermore, from Table 2.1 emerges that the current literature does not consider the possibility of integrating custom ZigBee devices, more precisely devices that do not adhere to any profile. Such feature is desirable in application scenarios in which vendors deploy prototype devices not yet adhering to any profile. Finally, some of the papers presented have been designed in order to be deployed on resource-constrained boards, as for example [14, 46] explicitly tested their solution on boards acting as gateway. Concerning the gateways based on the OSGi environment it is worth to notice that there exist several distributions of the OSGi suitable for low-cost, low-power devices such as Raspberry PI or BeagleBoard hardware. This is the case for the gateways described in [15, 44].

In summary, we started from the previous discussion in order to design and implement an integration gateway addressing all the previous points, chapter 3 details our solution.

2.2 Service Discovery in MSN

2.2.1 Background

Mobile Social Networks are build over concepts inherited from conventional Online Social Networks (OSN, such as Facebook, Google+ or Twitter) and wireless networks based on opportunistic communications such as ad hoc networks or delay tolerant networks. Specifically, a MSN is a network of mobile devices (typically smart phones, smart watches, tablets etc.) that communicate opportunistically and that are carried by people. The mobility traits and the interactions between the mobile devices can be investigated as a reflection of human social interactions, enabling social ties detection by means of the analysis of interactions between devices, the user personal data and the time and space dependent context data that each device can collect.

MSN are enabled by the tremendous diffusion of mobile devices in the planet. In fact, the average number of devices per-person exceeded 0.5 in 2013, which means that there are currently billions of mobile devices in the world [51]. These devices generally embed several network interfaces, including short-range (e.g. WiFi, Bluetooth or ZigBee) and long-range (e.g. GPRS, 3G and LTE). Short-range interfaces allow devices to communicate with each other without relying on any network infrastructure, while long-range interfaces guarantee broadband connectivity mostly

everywhere. The devices are also equipped with advanced multi-core CPUs and dedicated GPUs, flash memories and external storage, and they embed a plethora of sensors such as accelerometer, gyroscope, GPS receiver etc. Such features make these devices ideal for the exploitation of a new form of distributed computing, such as the opportunistic computing [52].

As compared to OSN, MSN offer complementary opportunities to users, as they exploit the mobility of humans carrying the devices of the MSN. The human mobility can be characterized by three key-aspects: *(i)* common activities (e.g., we all go to work, go back home, travel toward office), *(ii)* visiting a limited number of locations (e.g., home, the office, a movie theater), and *(iii)* traveling more often along short paths instead of long routes. On the other hand, homophily among people introduces additional features in the way people (and hence devices) in MSN move and behave. In particular, people tend to meet more frequently and/or for longer periods with other people with whom they share similarities. As discussed in [16] contacts among similar happen more frequently than that contacts among dissimilar people. For instance, race and ethnicity, sex and gender, age, religion and education are some notable aggregation factors that tend to cluster similar people together. A common way to model all these aspects is to group devices into communities and to profile them according to the interests of their users or according to other criteria such as time spent together, places visited or common acquaintances. The detection of communities can result helpful in order to study the effect of mobility on the effectiveness of different strategies for resources and information diffusion in MSN. In this section we give an overview of these aspects of MSN, before introducing the service-oriented approaches.

Communities in Mobile Social Networks

Human mobility is driven by several factors, among which, social interactions are predominant [53]. People that meet frequently for long periods have robust social ties, and they form *communities*. There is not a unique definition for the term community [54, 55], however it can be informally described as clustering of entities, e.g. people, that are closely linked to each other and have non-volatile social ties. Examples of communities are the employees of the same company that work in the same office, students attending the same lectures or members of a family. Communities are a useful tool for analyzing and exploiting the potentialities of a MSN. In fact, members of a community are connected by similar interests, habits or mobility patterns. For example, classrooms attending to the same lecture may share interests for the university campus events or nightlife in the city where they live. Similarly, people commuting by train might be interested for the timetables, delays or the weather forecast at the destination. Note that, belonging to the same community does not necessary imply to be often in proximity, and communities composed by people that share the same interests but that have different mobility patterns are also admissible in MSN.

In general, algorithms and applications designed for MSN can exploit the existence of communities to optimize the diffusion of information, the routing and the discovery and access of services available the MSN.

The community detection can be implemented with centralized [56, 57, 58] or with distributed algorithms [59, 60, 61]. The centralized algorithms need full knowledge of the whole network and of its ties, while in the distributed ones each device has a local view of the network, and it needs to detect the communities to which it belongs. The mobile and dynamic nature of the MSN leads us to consider the distributed community detection algorithms as the natural choice. Indeed, with distributed algorithms, the devices need to know the quality of encounters with other devices, in terms of:

- temporal metrics: measure of the temporal dimension of the device encounters. Some notable examples are the contact duration, the average inter-contact time, and the last time a device has been in contact;

- similarity metrics: measure of the similarity degree with the encountered devices. For example, the similarity function can measure the interests, the number of similar contacts or the similarity with the places visited.

Hui et al. [60] propose a family of community detection algorithms, namely SIMPLE, k-CLIQUE and MODULARITY. As a general principle, each device is maintaining a familiar set, in which encountered devices are imported when the cumulative contact duration exceeds a certain threshold. The community of each device contains all devices in the familiar set plus devices selected by the detection algorithm. If two devices have quite similar familiar sets, then they add each other into their communities. The two communities will merge in SIMPLE if they have more common devices than a threshold, in k-CLIQUE if the other device's familiar set contains at least $k-1$ devices of the root device's community and in MODULARITY if the familiar sets of the other device's community members, not included in the root device's familiar set, belong to the root's device community. One of the drawbacks of the SIMPLE algorithm is that it does not remove old contacts from the familiar set. In particular, once a device has been added to a community it will be never removed. AD-SIMPLE algorithm [59] extends SIMPLE by adding a mechanism for aging contacts so that it is possible to remove devices from the community. Another distributed community detection algorithm is presented by Li and Wu [62], as part of the community-based epidemic forwarding scheme called LocalCom. The detection algorithm is initiated by a self-selected device, called initiator. In order to use a broader definition than cliques, the notion of the virtual link is introduced and two devices are considered to be socially related if they share a strong connection with a common neighbor. Moreover, by adopting normalized cuts, devices are able to form communities in a distributed way. This is achieved by selecting the detected community with the smallest normalized cut value.

DRAFT [61] is a distributed spatio-temporal detection algorithm. It uses the cumulative contact duration to add or remove a device from a community. Moreover, it implements a decay mechanism to prune devices that no longer belong to a community. DRAFT takes three parameters: cumulative threshold, decay rate and length of the time frame, the combination of these parameters affects the cardinality of the communities detected by each device.

Distributed MSN Architecture

MSN are mobile and dynamic in nature, people establish contacts according to their social activities, and, consequently, devices carried by people are able to interact only for short and intermittent periods. The network architecture required by MSN differentiates form the OSN. In fact OSN rely usually on a web-based centralized architecture. In this case, a cluster of replicated servers provides the back-end of the social network; the servers are supposed to always be on-line backed by ultra-fast and reliable network infrastructures. The (mobile) clients of the OSNs connect to the servers by means of Wi-Fi or by long-range network interfaces such as LTE. In MSN such architecture is not feasible, hence we do not assume the existence of a stable network infrastructure rather all the communications happen in an opportunistic way. More precisely, we refer to MSN based on a distributed architecture as shown in Figure 2.12. The figure shows the snapshot of a MSN along 24 hours. The snapshot

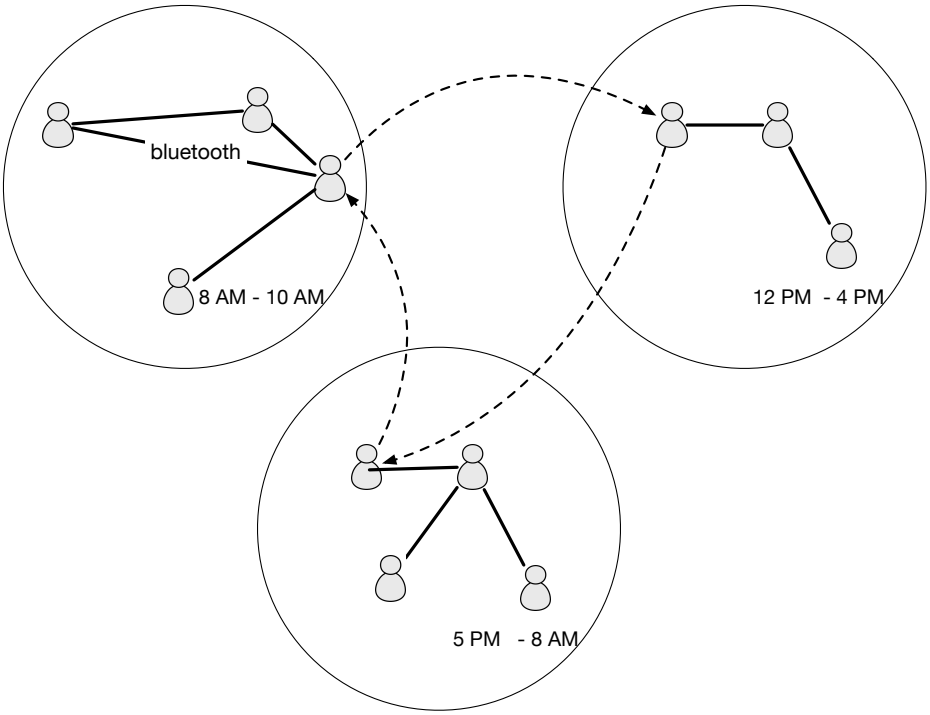


Figure 2.12: Snapshot of a distributed MSN.

shows the connections established by group of devices in different periods of the day. Some of them are connected directly by means of short range network interfaces (for example Bluetooth or WiFi Direct), other devices are (partially) disconnected.

The key feature of distributed MSN is the total absence of centralized servers. Mobile devices are able to communicate and access the information only by connecting to other devices. Therefore, the network itself has to store and route data until the correct destination is found. Distributed MSN can be further divided into two different categories. In the first one, devices share contents directly or by using some deployed infrastructure (but not centralized anyway). In the second one (which is the most challenging), intermediate devices might be required to provided inter-communication between two devices (they are also named rely devices). Because no infrastructure is considered, the mobile devices themselves are designated to route or carry the messages until the final destination is found.

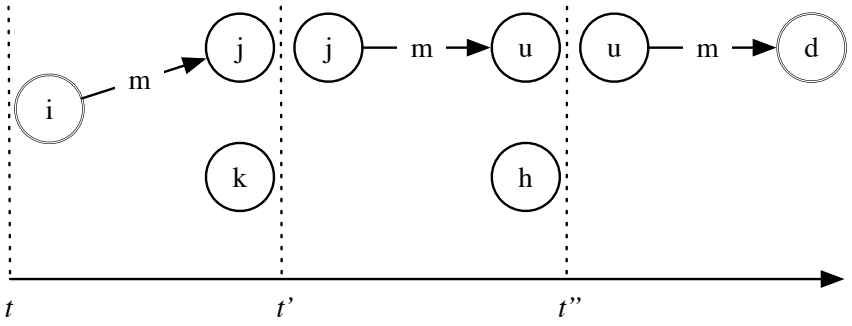
Routing and Data Dissemination in MSN

Routing and data dissemination are two complementary key-problems in MSN. The routing aims at delivering a message from the source to the destination, in this case the sender and the device receiver are known. Data dissemination aims at disseminating data produced from a provider to devices that might be interested in using such data, in this case the sender is known but the receivers are not known. In general, the goal of a routing protocol is to deliver a message quickly toward its destination, while the goal of a dissemination protocol is to diffuse efficiently a data among the interested devices. Figure 2.13 shows two simple strategies for routing and data dissemination.

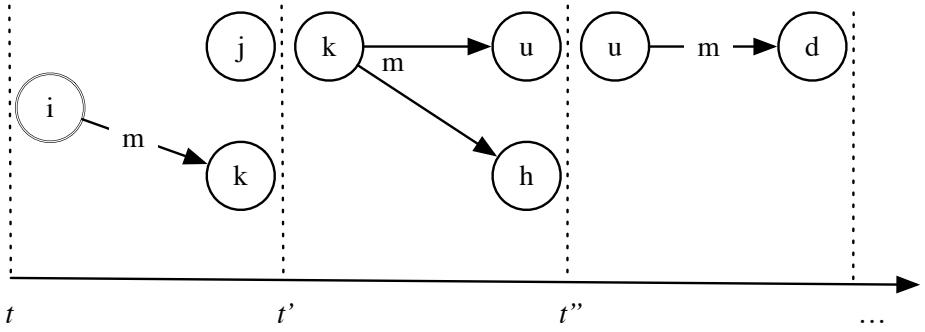
Figure 2.13a shows a simple routing strategy from device i to device d. Devices receiving the message m forward it to another device according a local decision, for example device j forwards m to u because u is *often* in contact with the destination d. Such decision is taken on the basis of a local routing policy.

Figure 2.13b shows a controlled-flooding dissemination strategy. Device i generates and propagates the message m among devices that are interested in receiving m; in particular i selects k at time t', k in turn, selects both u and h at time $t''$and, finally, u selects d. In this latter case, there is not a final destination, rather the message m is propagated until a certain rule applies (for example when it reaches its maximum hop count).

In MSN there are generally no stable end-to-end delivery paths. Therefore, delivering messages to a destination or diffusing a data towards interested devices becomes more challenging. Most of the existing solutions for routing and data forwarding adopt the store-carry-and-forward approach, which consists in storing a message locally to a device and carrying it until the destination or a *promising* device is encountered (i.e. a device that will hopefully meet the destination in the future, according to some metric). It is thus important to adopt a smart relaying selection and forwarding decision in order to reduce the latency of delivery. Based



(a) Routing message m from i to d.



(b) Disseminating message m from i.

Figure 2.13: Routing and data dissemination.

on a chosen strategy, policies for routing and data forwarding varies from epidemic replication of all the messages to every node like Epidemic routing [63], through to multi-copy and single-copy forwarding. Flooding-based protocols with unlimited replicas of messages cause high demand on network resources, such as storage and bandwidth and they cause congestion. However, multi-copy protocols typically aim to limit the number of replicas of the message in order to leverage a trade-off between resource usage and probability of message delivery. On the other hand, single-copy strategies require routing algorithms to implement a next-best-hop heuristic that forwards the messages to those devices with a highest probability to deliver the message to its destination.

Service-Oriented MSN

The social nature of MSN leads people to form and maintain communities of similar. Communities are structures offering a great opportunity for people's devices to interact and exchange useful information. Devices can gather environmental data, download and share contents from the web, store multimedia contents and exploit

sensing capabilities available on the device. Such information is continuously refreshed and updated according to the input of the end-users. When data available on a device are offered to the MSN, the computation paradigm may become service-oriented. In fact, every data can be exposed as a service offered to devices located in proximity or not directly connected but that are part of the same community. However, the service oriented paradigm cannot be applied *as-is*, rather MSN introduce several restrictions to such paradigm. The absence of a network infrastructure, the opportunistic nature of MSN and the human behavior typical of these networks draw a challenging but exciting scenario. In particular, the problem of finding and accessing services (in one word, *s*ervice discovery) has been widely studied in many other contexts very close to MSN [7, 64].

With the term service, we refer to every hardware and software resource provided by a device in the network. In this section we propose a spectrum of services potentially available in the MSN Figure 2.14 shows a non-exhaustive classification of application services in MSN. Services are classified in three categories:

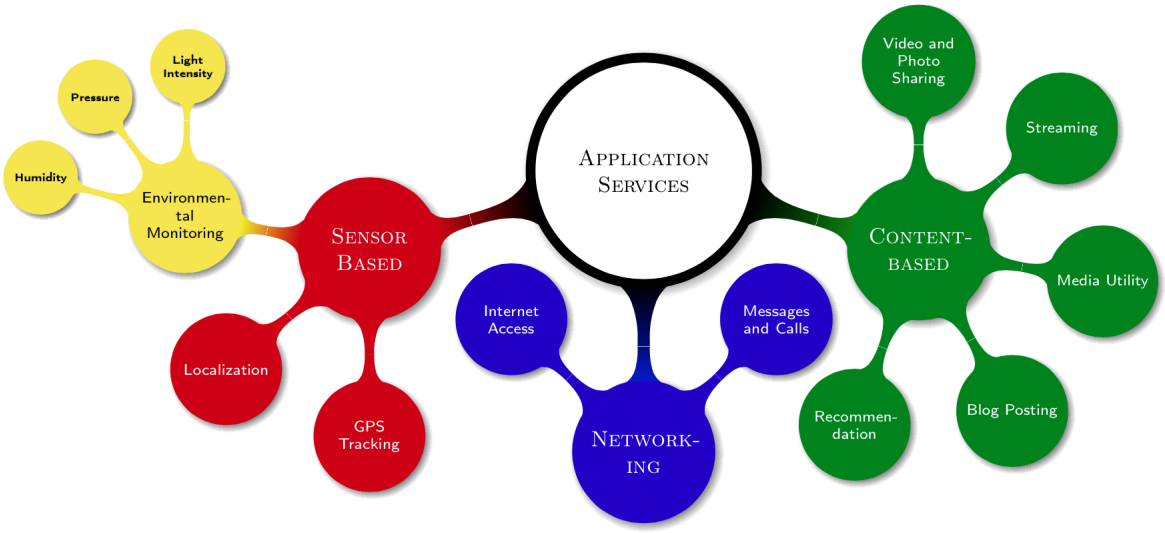


Figure 2.14: Classification of services in MSN.

1. content-based;
2. networking;
3. sensor-based.

Content-based services are designed for sharing media contents with other devices; notable examples in this category are services for sharing video and photo, streaming of short videos, cooperative blogging platform [65], utility services for media contents such editing or compression of images, recommender systems for recommending places to visit (e.g. restaurants, pubs or others) [66].

The category of networking services does not only include the traditional services referred by [9, 8, 42] namely printer, fax, scanner and memory storage, but it also wraps services designed for sharing networking functionalities with mobile devices. Notable examples in this category are services for sharing the Internet connection [67] and services for enabling text messages or phone calls. The category of sensor-based services comprises all the services that exploit sensors installed on the device. When the sensing capabilities of a device are combined with the mobility and sociality aspects, the range of application services increases. Some notable examples are the opportunistic sensing and crowd-sensing [68, 32] platforms designed to gather information from mobile devices by exploiting sensors on the device itself.

Moreover, the way in which the application services are provided is something that distinguishes the MSN from other application scenarios. In particular, some services are meaningful only if they are geographically and temporally tagged. For example, during a social event like a concert, people can start exchanging comments and thoughts about the events in a shared virtual dashboard. The dashboard service is not provide by a server, rather all devices roaming around the event implement a distributed service that will exist until someone will keep using it. We define the way of creating and providing service on the fly *here & now*.

Service Advertisements and Queries

Devices part of a service oriented architecture play different roles: service provider, service client or service registry, moreover devices can play multiple roles simultaneously. The service providers announce of the existence of the services that they provide. This phase is achieved by propagating a service advertisement message to the devices that are met opportunistically in the MSN, this is commonly referred as proactive service discovery. The service advertisement (or for brevity the advertisement) is a compact data structure that summarizes the core features of the service, such as:

- the identifier of the service provider;
- the service interests that classify the service according to the functionalities it provides[4];
- the Quality of Service (QoS) features that describe how the service is provided; for example the estimation of the service competition time, the estimation of the length of the pending requests and other metrics useful to describe the performance of the service provider.
- input and output parameters that describe how to invoke the service and the kind of expected results.

[4]There are well-known methods for service classifications, relying on syntactical or ontology-based techniques [69, 70, 71]. These kinds of classifications is beyond the scope of this thesis.

The service clients are the devices that discover and access the services in the MSN. The clients propagate a service query in the network, this phase is commonly referred as reactive service discovery. The query is a message that describes the kind of service needed by a client. The query wraps a set of parameters that must be directly compared with an advertisement message, in particular a query may contain:

- the service interests that specify the kind of functionalities needed by a client;
- the QoS features that require a specific level of performance of the service provider;

Devices in MSN implement a distributed service registry. A service registry is a (set of) device(s) storing the service advertisements diffused in the network (Section 2.2.2 describes both the centralized and the distributed implementations). In particular, a device carries the service advertisements of the services it provides, as well as the service advertisements (provided by other devices) that the device received proactively or reactively. Since every device in the MSN is part of the registry, as soon as a device receives a query it checks if one of the advertisements stored locally matches with the query. If a match is found, than the device carrying the answer replies to the client with the service advertisement.

2.2.2 Related Works

Service discovery architectures can be classified in: directory-based, directory-less or hybrid solutions [72]. The directory-based architectures assume the existence of a directory whose role is twofold:

- management of advertisements: devices providing one or more services store the advertisements in the directory;
- management of queries: devices looking for a service send the query to the directory. If the query matches with one or more advertisements, the directory replies to the service client with the matching advertisements.

The directory-based architectures can be implemented in a centralized or distributed way. In the centralized solutions (among them we cite some results [9, 73, 74]) there is only one device acting as directory that is known in advance by every device in the network. This is the simplest solution, but these architectures are not designed to scale well in large-scale systems. Indeed, the directory is a single point of failure: if it is not available then the clients are not able to discover any service. In distributed solutions (an old but meaningful example is the Jini architecture [73]), a set of devices acts as a distributed directory. Such pool of devices can implement methodology for synchronizing the advertisements stored locally or to split the advertisements inside the pool. A distributed directory can be implemented

by statically configuring devices that implement the directory or by dynamically electing the devices [75]. In this last case, elected devices are those that have the best features for playing the role of directory [76, 77]. The distributed directories are more robust than the centralized ones, as several devices participate to the directory role.

The directory-less architectures, (some notable example are [78, 79]), are simpler than the directory-based ones. With directory less architectures the role of the registry is shared among all the devices in the MSN. As soon as a device receives a query, it checks if any of the advertisements stored locally matches the query. In this case, the device replies to the client, otherwise it propagates the query to other devices selected with a strategy. In a similar way, the providers diffuse the advertisements in the network with a proper strategy. Directory-less architectures are more suitable for MSN, since they do not assume the existence of always-on devices acting as directory. However, the strategy adopted by devices for querying and advertising is a crucial part of the discovery process, and simple solutions based on flooding or gossiping can revel too aggressive in terms of use of the network bandwidth, overheard of the protocol and energy consumption of devices. The next subsections describe the advertisement and query strategies available for service discovery in MSN.

When directory-based and directory-less are combined together, the discovery architecture is hybrid. In this case, the architecture supports the existence of a directory both centralized and distributed. Clients first query the directory; if no advertisements are found then clients query the whole network in a similar way to the process described for the directory-less ones.

Advertisement and Query Strategies

This section discusses the approaches for the dissemination of advertisements and queries in MSN. Some of the works here discussed are specifically designed for service discovery problem, while others are generically designed for data dissemination, but they can be applied to the dissemination of advertisements and queries. The diffusion strategies presented in this section are organized within 3 categories: namely interest-based, flooding-based and social-based strategies.

Table 2.2 summaries the works surveyed in this subsection according to the following features:

- Forwarding rule: the strategy adopted for the message forward;
- Mobility: the application scenario considered by the authors to evaluate the strategy. We distinguish between real-word scenarios (obtained with real-world mobility traces) from synthetic scenarios (obtained with mobility models);
- Community Detection: the algorithm used for detecting communities (if adopted in the strategy);

- Evaluation Metrics: some commonly used metrics used for evaluating the strategy.

Interest-based strategies

In interest-based strategies each device is associated with a set of interests representing the user profile, and each query or advertisement message is also described by a set of interests. The message diffusion strategy exploits only relay devices that have interests similar to the ones of the discovery message. To this purpose, these strategies adopt a similarity function whose goal is to measure the similarity between the interests of the message to be forwarded and the interests of the potential relay devices. The higher the similarity, the more likely the person carrying the device will use the information (query or advertisement) contained in the message. Examples of similarity functions are the Jaccard-index, the cosine similarity or the Hamming distance.

The authors of [80] propose a peer-to-peer service discovery protocol. They use a pull-based, (on-demand) query generation and propagation mechanism, that merges two distinct overlay networks: one made by the *interests* of devices, i.e. information about the types of contents the devices are interested in, and one made of contacts built with information about physical proximity between devices. Query are represented with a type (music, video, etc) and they are sent to all interested neighbors; these are discovered by previous messages received by the device and cached locally, or by propagation of interests using a gossip protocol. The authors study via simulation the hit-rate of the service discovery protocol and the influence of mobility and density of the network. The queries are based on a static set of interests without others metadata used in other works.

The SocialCast [81] algorithm implements a publish/subscribe paradigm as done in [82]. SocialCast relies on the observation that people with similar interests tend to meet more frequently with respect to people without overlapping interests. The data forwarding strategy is implemented by observing the mobility patterns of people and also the interests of devices running SocialCast. SocialCast adopts an interpolation function based on the Kalman filter to predict the movement of people by tracking their previous movements. The SocialCast algorithm is implemented in several steps: *(i)* the dissemination of user interests, each device broadcasts the list of its interests to the ego-network. Then *(ii)* every device computes the utility function for all its interests and lastly *(iii)* every device checks the contents it carries with respect to the subscriptions of the devices, and eventually the contents are forwarded to the subscribers.

Mei et al. [83] focus on a forwarding strategy based on the sociality of users in a MSN. This paper proposes the so-called interest-cast forwarding schema. The aim of interest-cast is to efficiently diffuse information to the highest number of interested devices (the authors use the cosine similarity function). The selection of the device matching with the interests is based on the similarity function. Under this respect, the interest-cast service can be seen as a directory-less service in which

			Mobility		Evaluation Metrics						
	Paper	**Forwarding rule**	**Real-World**	**Syntehtic**	**Community Detection**	**Delivery Delay**	**Delivery Ratio**	**Message Sent**	**Power Consumption**	**Fairness**	**Other**
Interest	[80]	Similarity among interests	infocom 06	SWIM	not addressed	✓		✓			Coverage (% of relevant destinations reached
	[81]	Similarity among Interests	PMTR	HCMM	Louvain algorithm	✓	✓	✓			Coverage
	[83]	Similarity among interests and energy awareness	SIGCOM	SLAW	not addressed	✓	✓	✓	✓	✓	Effectiveness Fairness
	[85]	FM: encounter time with the final destination IB: intetrest-based	infocom 06	SWIM	not addressed	✓					
	[86]	interest-based		CMM	Girvan-Newman			✓			Time To leave
	[87]	interest-based		CMM, RWP	not addressed		✓	✓			
Flooding	[88]	Flooding-based		ad hoc simulation	not addressed	✓		✓			
	[89]	Location-based		ad hoc simulation	not addressed			✓			Number of clients that discover a provider
	[90]	Flooding-based		ad hoc simulation	not addressed	✓		✓			
Social	[92]	General-purpose framework	infocom 06, MIT Reality, UCSD	not addressed	not addressed	✓	✓	✓			
	[93]	Publish/Subscribe within communities	MIT Reality, UCDS, CAM, WirelessRope	not addressed	K-Clique, SIMPLE	✓	✓				
	[96]	Centrality degree of nodes	infocom 06, MIT Reality, Cambridge, Hong-Kong	not addressed	K-Clique, Newman WNA		✓	✓			Centrality Betweeness
	[97]	Duration of the enconters among nodes	NUS, infocom 06 MIT Reality	no	yes, assume the existence						Number of nodes that can retrieve a copy of a data in a given time slot.
	[82]	several utity functions: MFV,MLN,F,P,US		HCMM	Based on the HCMM model	✓	✓			✓	

Table 2.2: Comparative table of advertisement and query strategies.

the service selection is performed by the potential recipients of the content that filter the input when it is received. The assessment of interest-cast addresses the performance and overhead of the content diffusion, and especially focuses on the coverage, i.e. the percentage of relevant destination that hold a copy of the content within a time frame. The assessment is performed over real mobility traces and on synthetic traces built with the SWIM mobility model [84].

In [85] the authors describe a comparison between two forwarding strategies used to deliver a message from a source towards a destination in MSN. The paper presents two simple but effective strategies, namely FirstMeeting (FM) and InterestBased (IB). The first strategy is social-oblivious in the sense that it does not use any information related to the interests of the devices in order to take the forwarding decision. Conversely, the second strategy is based on the interest profiles of devices. With FM the source device, for example the one generating an advertisement for another device, always generates two copies of the message. One copy is stored locally while the second one is forwarded to device R, only if R is met before the final destination. Similarly to [83], the IB strategy relies on the assumption that people with similar interests tend to visit more frequently. With the IB strategy a device delivers a message to another device if the similarity between the interests is greater than a fixed threshold. The authors present a comparison between FM and IB by using two simulation scenarios based on real and synthetic mobility traces. The traces reproduce two realistic scenarios where the mobility of devices is affected by the interests of people carrying them.

The authors of [86] defines the BehaviourCast problem for the diffusion of information in MSN. BehaviourCast is based on four key-features:

- the validity: forwarding a message to a subset of the interested device;
- the effectiveness: forwarding the message in order to achieve total coverage of devices interested;
- the efficiency: involving the smaller number of relying devices;
- the query termination: interrupting the forwarding of a message after a given time.

The authors also propose two interest-based strategies solving the BehaviourCast problem, namely the basic InterestCast and the weighted InterestCast. With the basic InterestCast, every device i executes an utility function that counts the total number of devices encountered and sharing the same interests of i. The authors note that such basic strategy has no memory of past encounters, for this reason the authors propose an enhanced version, namely the weighted utility function. This last strategy is based on the Shannon's Entropy principle, it counts separately the number of encounters the device i had in the past with device j. The weighted utility function, hence, keeps track of the number of encounters with every single devices

met. The authors simulate the two strategies proposed both with real and synthetic mobility traces and they compare the results obtained with respect to ProfileCast and SocialCast algorithms.

The authors of [87] present PIPER an interest-aware social-based forwarding algorithm for MSN. PIPER extends the IPER algorithm by taking into account also the energy consumption of devices involved in the forwarding process. The IPER forwarding strategy combines together several factors in order to rank all possible candidates for forwarding a message towards its destination. In particular, given the device i the IPER function combines the similarity between the interests of the device and the interests of the message to be forwarded, a dumping factor to determine the amount of reliance on opportunistic forwarding, the social rank of device i and the similarity between the interests of the advertisements and the interests of all the friends of device i. PIPER extends IPER by adding another parameter used for ranking a candidate forwarded, namely the battery level. The authors state that during the forwarding process, devices with low battery level should be excluded form the forwarding process in order to reduce as much as possible unattended battery depletion. In particular a device is highly ranked if one of the following conditions holds: its battery level is above a threshold, the person carrying the device is linked with popular friends and also if the set of friends of the person caring the device are also interested in the message. The paper also discusses the performance of PIPER with respect to IPER and a benchmark algorithm based on a Epidemic forwarding strategy with real an synthetic mobility traces.

Flooding-based strategies

Flooding-based strategies aim at maximizing the number of recipients of queries and advertisements messages but they do not use any social-based metric to select the rely devices of messages. The main advantage of these solutions is the effectiveness and the simplicity of the diffusion strategy, but they may incur in a high overhead in terms of messages forwarded, network bandwidth and the devices battery depletion.

The authors of [88] present a service discovery protocol for MANET with low mobility of devices. The protocol is based on the reliable broadcast. Services are not described by id's nor by a service type, rather they are labeled with the input/output parameters. The parameters of the services available in the network are spread using a proactive exchange of the local tables stored locally by each device. Such tables contain a mapping between the service and the I/O parameters needed. Moreover, the protocol is supposed to be embedded with the neighbors discovery protocol of the underlying MANET. In this way, the services are advertised together with the discovery of devices found in proximity. Queries are diffused reactively by flooding the neighborhood, hence this happens as soon as a device needs to access to a specific service. The queries are described with a set of parameters, the authors propose a simple but effective solution for the composition of multiple services together. The use of parameters are described using a common taxonomy, that includes the possibility of hierarchies among parameters. The authors classify the

queries within two categories, namely exact or generic. The protocol is evaluated via simulations by means of the NS-2 network simulator, statistics about the delay of service discovery and the amount of overhead of the protocol are studied. This enables service composition protocols and generic query of service providers.

In [89] the authors address the problem of service discovery in delay tolerant networks. The scheme adopts a proactive propagation of service advertisements, where each service provider announces periodically its own services to the entire network. The advertisement is composed by a list of keywords, and all the devices in the network cache the received advertisements and associate them to their latest time of reception. When a client looks for a service, it first checks into its local cache. If there are no matches, then it starts a reactive service discovery by broadcasting a service query packet. Each device receiving the query looks for a service matching with the query in its internal cache, hence the query can receive an early response even from intermediate devices. The client then selects the matching replies to its query based on the latest time of reception. The solution proposed [89] is evaluated by means of the NS-2 network simulator with an ad hoc mobility model to assess the efficiency and the overhead of service discovery. Although this scheme does not consider sociality aspects in the advertisement and query distribution, its basic architecture of service discovery is also valid for MSN, and its mechanisms of diffusion of advertisements and service queries can be easily combined with knowledge about the communities and user interests to be adopted in MSN.

OLFServ (Opportunistic and Location-aware Forwarding protocol for Service delivery) [90] considers a scenario in which devices are geo-localized. The authors refer explicitly to an application scenario sparse, hence the network may become disconnected for some periods. It assumes that any device can advertise a service, by using a multicast-based scheme that limits the area of propagation of the advertisements. Service advertisements have an expiration time and they contain information about the position of the emitter (the service provider) and the geographic area where the service can be accessed. Furthermore, the service advertisements also include a list of potential recipients (service clients). The paper does not discuss how the list of potential receivers is created and maintained. Beyond making public a new service, the purpose of the advertisement is also to make more efficient the access to the service by providing geographical information about the service provider. The authors evaluate the efficiency of OLFServ in distributing the advertisements against their rate of success (in terms of number of clients that find correctly a service provider).

Social-based strategies

Social-based strategies exploit information about users social relationships to drive the dissemination of queries and advertisements. However, the human social relationships are difficult to capture and to measure, since they are influenced by many factors. According to Granovetter [91], "the strength of a tie is a (probably linear) combination of the amount of time, the emotional intensity, the intimacy (mutual confiding) and the reciprocal services which characterize the tie". Relationships among devices can be determined by using specific utility functions, such

as contact duration among devices, inter-contact time or remaining inter-contact time or by means of well-known centrality metrics such as the betweenness centrality or the closeness centrality. Generally speaking, the social-based strategies are characterized by the capability of the strategy in discovering devices among which there exist non-occasional social ties. In turn, such devices are used to optimize the diffusion of information among social-linked devices.

The authors of [92] describe a general forwarding algorithm for the diffusion of advertisement and query messages, namely Delegation Forwarding (DF). DF relies on a basic idea: a message is forwarded from the sender to an intermediate device only if the receiver is *better* than the sender with respect to a specific metric. The authors of DF does not define any new metric for taking the forwarding decision, rather they propose a general framework for deciding who should receive a message copy. The authors discuss the effectiveness of DF when implemented with some strategies:

- Epidemic [63]: a device forwards a message to all devices it is in contact with only if they did not have already a copy of it stored locally;
- Frequency [93]: a device forwards the message to another device if it has more total contacts with respect to the sender device;
- Last Contact: a device forwards a message to another device if it has contacted any device more often than the sender device;
- Destination Frequency: a device forwards a message to another device if it has contacted any device more frequently than the sender device;
- Destination Last Contact: a device forwards a message to another device if it has contacted any device more recently than the sender device;
- Spray and Wait [94]: the message source s creates l replicas for the same message. If, along the time, s carries $k > 1$ replicas than if s encounters another device that has no replicas, then s forwards half of its replicas to the encountered node. Otherwise the Destination Last Contact rule is applied.
- SimBet [95]: the message source forwards a message to another device only if the SimBet metric of the encountered devices is higher than that the SimBet metric of the source node. The SimBet metrics is basically a linear combinations of the betweenness and similarity graph metrics.

The BUBBLE Rap algorithm [96] implements a social-based forwarding strategy in delay tolerant networks. Users are given a global ranking and a local ranking, computed according to their importance in the network. The information (either service query or an advertisement) is first forwarded from the sender through users with higher global ranking, until the message reaches a user in the same community

of the receiver. Then, the message is forwarded only among users in the same community according to the local ranking, until the information reaches its destination.

Socio-aware [97] is a publish subscribe strategy that relies on a overlay structure based on communities. Communities are detected by means of two algorithms proposed by the same authors [60]. The overlay structure assumes that devices can play different roles: the brokers, the subscribers and the publishers. The brokers have high centrality degree inside the community. The brokers receive all the subscriptions and un-subscriptions from other devices as well as the list of the centrality values form the devices in contact with a time stamp. The broker device evaluates the centrality values previously received in order to decide which device should take the role of the broker. If a change is needed, then the broker transfers the subscription list to the new broker with the highest centrality degree. Then, an update message is sent to all the brokers in the overall network. During the gossiping stage, subscriptions are propagated towards the community's broker. When a subscription reaches the broker, it is propagated to all other brokers, and then the broker checks its own subscription list. In the case there are members in its community that must receive the subscription, the broker floods the community with the information.

In ContentPlace [82] devices advertise the data objects they are interested in, and data objects that they carry around the network. Data objects are contents of different types, such as media content that are transferred with some communication channels. The protocol proactively exchanges information about the channels provided and subscribed by devices encountered along the time. Moreover, the protocol must decide on the basis of a utility function which data objects to replicate. The utility is a function of weights, the weights are function of groups, channels and the availability of the object in the network. The authors analyze via simulation ContentPlace with a number of different utility policies for computing such weights.

The paper [98] presents a forwarding schema by taking into account the time needed to transfer a piece of information (especially multimedia content). The authors observe that the time needed for such transfer could be not negligible, and that the transfer success is related to the duration of the contact time between the devices involved in the information transfer. In fact, typical forwarding schema for MSN may fail to forward the content if the transfer time is longer than the contact time between the sender and the chosen rely node.

Service Selection

Service Selection is another important step of every service discovery protocol. After the advertisement and query steps, a device might receive a number of advertisements whose services provide the same or similar functionalities. The device needs to select the best candidate and, eventually, compose multiple services together [99]. In [7] service selection is presented as the phase that comes after all the replies from a query are collected at the client and the right service provider must be selected among the alternatives. As discussed in [7], most of the effort in the literature focus

on service advertisement and query, while the selection phase has not been studied in depth. In the following, we review how service selection has been studied so far by focusing on how to describe a service and how to implement the proper selection.

First, service selection is based on the evaluation of the properties of the service. In MSN different formalisms (languages) are used to describe service properties, they range from a simple key-value pairs, to XML or even more complex description. Moreover other formalisms are needed to describe user preferences and other context-information such as users-location or scope awareness. There is a constant trend in raising the complexity of these formalisms, moving away from simple keywords based service description and syntactic-matching of attributes, towards a semantically richer matching [88]. This, in turn, enables the use of very descriptive queries that perform most of the selection phase by carefully describing what are the real interests of a client. Most of the works on service discovery lack this point of view and study the query phase only as an information diffusion process; only few works measure metrics like the precision and recall of a query [100].

Second, as reported in [7] service selection protocols differentiates between user's assisted or automatic selection. In MSN, with pocket devices and mobile users, user's assisted selection might result not appropriate because users are typically not aware of the internal mechanism regulating the selection policy. Moreover, the MSN are supposed to implement an autonomic cooperation model, where user's intervention should be avoided as much as possible. Conversely, the automatic selection policies are more widely used and we expect that has noted previously, better description of the user needs and of the context, will reduce the gap between the automatic and manual choice, with respect to overall user's satisfaction. For this reasons, we focus on the automatic service selection protocols. We therefore present a first attempt to classify the most interesting selection protocols along two categories: manual and automatic selection as shown in Figure 2.15.

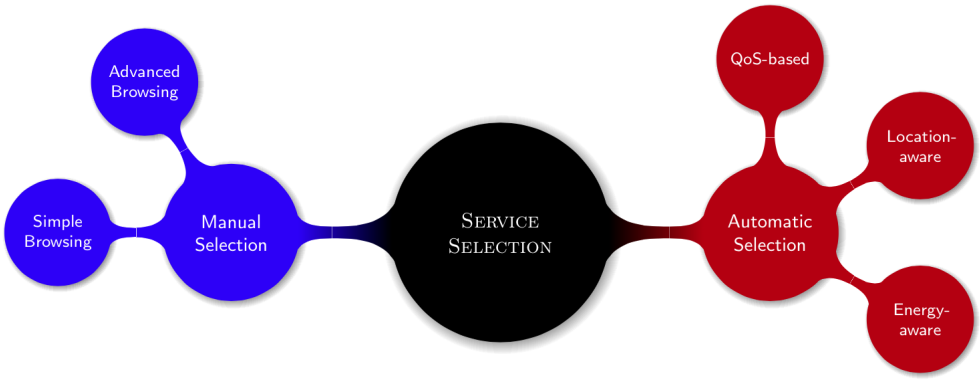


Figure 2.15: Service selection strategies.

A general-purpose service selection protocol is proposed in [101]. This selection protocol does not depend on any specific discovery protocol, rather it can be po-

tentially applicable with any existing discovery protocol. The solution described in [101] implements a service selection based on dynamic attributes such as the size of the print queue or the computational load of the host that a service may be located on. Each client has a specific set of weights that measure the relevance, for that client, of the dynamic attributes. The weights are used to compute an overall score used to rank different service providers. Given the attributes and the ranking for a client, the service selection becomes mostly automated. In fact the selection is obtained by selecting the first provider in the ranked list of every client.

The authors of [102] propose several policies for the service selection. The policy proposed are: (i) Minimum Expected Value that minimizes the time required to receive the first response of the service invocation, (ii) the Random policy that selects randomly the provider, (iii) the Always First policy that selects the first suitable provider that is in contact with the client, if no such provider exists, the client waits to encounter it, (iv) the Atomic policy that selects the provider that offers a single service satisfying the request of the client.

The authors of [103] propose a cross-layer service discovery and service selection architecture. The solution proposed is based on the extension of two routing protocols for MANET, namely the DSR and DSDV by adding the service discovery and selection features. The SDL is defined as the Service Discovery Layer and the RLD is defined as the Routing Layer Driver. The SDL stores information about known servers in a service table. Table entries have five fields: service description, service location (e.g IP of the provider), minimum hop count from the current host to a service provider, optional routing protocol specific information provided by RLD (e.g., a list of available routes to the destination), and optional service-specific metrics supplied by a service provider (e.g., current load, CPU usage). The service selection is performed by ranking the client with respect to the information available in the SDL. The authors do not specify any policy, rather they propose a general-purpose framework for the service selection. Moreover, the authors propose a simple method for re-evaluating the SDL metrics and to re-select the provider as soon as the SDL metrics change significantly.

In [104, 100] presents a Location Aware Service Discovery Protocol (LADS), and a corresponding Service Selection Protocol (LASS). Both protocols assume that devices know their geographic location and their motion speed. The service discovery phase is based on a geographic scoped broadcast, i.e. only devices sufficiently close to the requester and not moving too fast, will receive the query and eventually send a reply. A node running LASS stops a query if it already forwarded k replies for the same query, the threshold k is chosen by the requester. Alternatively, a node running LASS stops the current reply if it does not satisfy a quality criteria, i.e. the distance/speed ratio of its service provider is worse than one already forwarded.

The authors of [105] propose two methods for the service selection. The methods proposed assume that every node in the network (the authors refer to nodes in a Wireless Sensor Network) know their position in advance. The first method is named Closest service selection and it selects the provider whose euclidean distance is the

smallest among all the providers available in the network. The second method is named Nearby service selection and, given a threshold of the distance, it selects the provider whose distance is the nearest to the threshold.

In [106] the authors present a middleware for service selection in MANET, with a particular focus on emergency scenarios. The authors classify services in two complementary classes, namely comfort and safety-related services. The authors show that common service selection algorithms fail to distinguish the differences on the two classes. The solution proposed in [106] is a context-aware M2M middleware for service selection that incorporates client's realistic expectations, which are predefined in the middleware. The selection of comfort services is achieved by selecting the most popular services. The popularity is computed by considering the feedbacks reported by other clients, such feedbacks are in turn combined with the hop-count metric. The selection of safety-related services is achieved by selecting the most reliable providers, also in this case the authors of [106] adopt the hop-count metric.

In [28] we describe CoDA, a cost-based service discovery algorithm for Smart Environments. CoDA is based on a directory-based discovery architecture with a distributed implementation of the service registry. The network is configured with a number of service registries, and both the clients and the providers directly interact with the nearest registry in order to find a service or to announce a service. Despite of the kind of architecture, CoDA addresses the problem of which provider selects when multiple options are available. Our solution proposes minimizes the energy consumption required by a client in order to invoke the services available. The energy consumption is estimated by considering the distance between the client and the provider in terms of number of hops, together with the energy cost of the path traversed. The lower the energy cost of the path toward provider x, the higher the probability that the client will select x as best option. CoDA considers different costs for the paths traversed by a client, such costs are defined according to the technology of the path. For example CoDA assigns different costs to wireless, wired or hybrid links.

Finally, Table 2.4 compares the service selection strategies previously described with respect to three criteria:

- selection policy: the policy used for selecting the provider, we distinguish among QoS, Localization or Energy-based;

- Selection in Query: this criteria describes if the selection strategy implements a pre-filtering mechanism in order to exclude in advance services that are appropriate for the service query;

- Network Architecture: the kind of architecture (either ad hoc network, WSN, DTN, etc.) considered by the authors;

	Paper	Selection Policy	Seletion in Query	Network Architecture
QoS	[101]	Relevance of service attribute	✗	Infrastructure-based
	[102]	Minimize response time, random selection, first choice, best-match policies	✗	Opportunistic network
	[103]	Description of the service, service location, minimum hop count	✗	MANET
Localization	[100,104]	Proximity of the provider	✓	Minimum-distance and speed ratio
	[105]	Closest provider and proximity provider	✗	Wireless Sensor Networks
Energy	[106]	Most popular provider and minimum hop-count	✗	MANET
	[28]	Minimize energy cost for the service access	✗	Infrastructure-based

Table 2.3: Comparative table of service selection strategies.

Service Access

In the service access, the client requests the service to the provider chosen in the service selection phase and it receives the results. In a MSN both the client requests and the responses from the providers may travel along the MSN using the available mechanisms. As there may not be a path constantly available between the client and the provider, the service access protocol should be tolerant to delays and disconnections. However, differently than the diffusion of advertisements and service queries, the communications involved in the service access depend on the nature of the service. In some cases the service is stateless, thus service access protocol is limited to the exchange of a request and a response message. In other cases instead, the request (or the response) may require the transmission of a large amount of data (think for example to a video streaming service). As a result, the phase of service access may be (even by far) the most expensive in terms of bandwidth and energy required. Despite this fact, only few works address this phase.

In particular, service access is discussed in some recent works [89, 107, 90, 108], although not specifically designed for MSN. Most of these works [89, 107] implicitly assume services are stateless, that is, they can be delivered without memorization of the state at the service provider, and thus they require a simple exchange of request and response messages. Differently, the work of [108] instead considers a wider scenario including services with state and stateless. The service access scheme of [90] assumes a relatively simple service, which can be delivered by exchanging a single request and response message. This work focuses exclusively on the routing of these messages. In particular, it aims at optimizing the routing by using a different routing scheme than that the one used for service discovery. The proposed routing scheme exploits the information about geo-localization of the client and of the provider, and their estimated speeds to deliver the messages in the right place and (possibly) at the right time.

The Time-Aware Opportunistic middleware (TAO) [107] addresses hybrid networks, defined as infrastructure-based networks with opportunistic extensions. Also its service access scheme (called TAO-INV, where INV stands for invocation) assumes relatively simple services, and it exploits request and response messages that are forwarded by using a store, carry and forward principle. Also in TAO the forwarding scheme for service access differs from the forwarding scheme adopted for service discovery, in the attempt to limit the cost of epidemic forwarding by reducing the number of copies of these messages that are generated in the network. Specifically, to increase the chance of fast delivery of a service request message, the routing scheme classifies the potential rely devices as good or bad based on the last date of contact with the service provider (the smaller is this date, the highest is the goodness of a rely device). To improve the reliability of the scheme, TAO-INV also forwards the message to some bad relay devices. Since the request message keeps the path traversed from the client to the service provider, the provider uses a reverse source routing scheme to send back the response.

Although [89] addresses resource/service discovery in delay tolerant networks, its results may also apply to service access in MSN. Differently than the previous works, the authors of [89] propose to merge the service query and request phases. In this approach, when a service query reaches the service provider it implicitly requests the service. In this case, the service response adopts an epidemic routing scheme, but when a copy of the response message reaches the client, the client immediately performs a network-wide broadcast to stop the further forwarding of other response message copies in the network in order to limit the overhead of the protocol.

In [108] the authors focus on service access (which is called service invocation in the paper) in disconnected ad hoc networks. The service access relies on a publish-subscribe mechanism, in which the clients publish a service description, and the providers subscribe for specific service descriptions. When there is a match, the service provider(s) transfer back the response by using a publish subscribe scheme. In particular, the subscriber publishes a message with the identifier of the client, so that the client can subscribe for this message and receive the responses. Although there are presumably limitations in the parameters that the client can use to request a service, the scheme is stateful and thus the response is not limited to a single message, but it can develop over several messages from the provider to the client. For this reason, this mechanism is associated to timeouts in order to let the provider clean its state of outdated subscriptions. Furthermore, the publish/subscribe scheme enables multiple providers to offer the same service to a client at the same time. This is partly a desired behavior of the protocol, as it enforces redundancy and reliability, but it requires a mechanism to limit the number of providers that connect to the client.

The entire mechanism proposed in [108] is also designed to be delay tolerant as the underlying network may be (temporarily) disconnected, and thus messages can be delayed. Furthermore, to manage situations in which the client and the provider become permanently disconnected during a service access, the authors propose to keep the state of the session at the client-side, so that the client can recover by looking for an alternative provider.

Apart the work of [108], the other works focus only on the aspects of routing related to the messages involved in the service access. Furthermore, as they are not specifically designed for MSN, they do not introduce any optimization in data forwarding related to the social aspects of devices mobility, which, as observed in [83], brings significant advantages on the performance of the communication protocols. On the other hand, they observe that the interaction between the client and the provider requires different data diffusion mechanisms than those used to diffuse advertisements and queries. In particular, they all adopt routing schemes that tend to unicast-like communications by limiting the epidemic effect in the message forwarding.

Based on the latter observation, all the existing literature on general data diffusion schemes for social-based networks can be reconsidered for service access. As the review of such results is beyond the scope of the this section, we limit here to

Paper	Network Architecture	Stateful Service	Forwarding Schema for Service Access	Use of Sociality Metrics	Use of Temporal
[108]	Mobile ad hoc networks	✓	publish/subscribe	✗	✗
[90]	Mobile ad hoc networks	✗	unicast/geo-localization of devices	✗	✗
[107]	Infrastructured with opportunistic extensions	✗	epidemic	✗	✗
[89]	Delay tolerant networks	✗	epidemic with limiting rules	✗	✗
[98]	Pocket switched networks	✗	unicast	✓	✓
[109]	Mobile social networks	✓	not addressed	✗	✗

Table 2.4: Comparative table of service access strategies.

mention [98, 109]. Specifically, [98] considers the contact time among devices and the time required to transfer the content in order to choose the best forwarding device to the destination. This can be particularly relevant in the access to services that have state and that require the transfer of large amount of data.

The optimization of the output bandwidth of a service provider in the context of a MSN is the focus of [109]. It formalizes a global optimization problem that aims at distributing the output bandwidth to several clients by taking into account the freshness of the content transferred to the clients, in a model where the content continuously flushes from the provider to the clients.

Table 2.4 compares the service access strategies previously described with respect to the following criteria:

- network architecture: the kind of architecture (either ad hoc network, MSN, DTN, etc.) considered by the authors;
- stateful service: whether the service access protocol requires a service provider to keep a state. This is necessary when the service is complex and its access requires several interactions between the client and the provider;
- forwarding schema: the message forwarding strategy adopted to access the services;
- use of sociality metrics: if the proposed approach exploits the human sociality to enhance the service access;
- use of temporal metrics: the use of temporal metric such as contact duration or inter-contact time, to enhance the service access;

2.2.3 Discussion

From the description of the MSN described in Section 2.2.1 and from the works previously surveyed we derive some important requirements for the design of an efficient discovery algorithm, which are discussed in the rest of this chapter.

Routing and multiple forwarding strategies

Discovering services in networks of mobile devices requires several steps. Each of the discovery steps has some specific goals that can be achieved by exploiting techniques coming from routing and data forwarding in MSN. In details:

- service query: the goal of this strategy is to propagate a service query to devices that can potentially answer to the query in short time. The query strategy cannot rely on a registry that stores all or a subset of the advertisements; rather, it has to explore the network to find devices that provide the service themselves, or devices that already know an advertisement matching with the query. For example, a device could have high probability to answer to a query if the person carrying the device already accessed to similar services in the past;

- service advertisement: the goal of this strategy is twofold. Primarily, it aims at diffusing advertisements in the MSN only to devices that might be interested in accessing the advertised service. The advertisement phase is successful if people potentially interested in a service receive (at the right time) the advertisement required. In this case data forwarding strategies can be used because the set of recipients of the advertisement is not known in advance (Section 2.2.2 surveys some available results). The diffusion of an advertisement might also be directed to a specific receiver (if the device that owns the advertisement knows the device to which deliver it). In this last case, the goal is to deliver quickly the advertisement to the destination. Here, the routing strategies are helpful to route the message towards its final destination.

  Secondly, the service advertisement strategy should avoid forwarding messages towards devices that already received them in the past from previous encounters. Indeed, flooding-based solutions may result too aggressive in terms of use of the network resources and energy costs, and the diffusion of advertisements has to find a balance between effectiveness of the diffusion and management of resources of the MSN.

Discovery mode

Devices must support both the reactive and proactive discovery modes (Section 2.2.1). With the reactive mode, a device propagates the query as soon as it needs to access to a service; with the proactive mode a device is proactively notified with some service advertisements during the occasional encounters with other devices. In a MSN it is important to adopt both the discovery modes in order exploit the occasional encounters with other devices to exchange service advertisements.

Community-based service discovery

Service discovery in MSN can exploit the community structure (see Section 2.2.1) in order to make more efficient the discovery phases previously described. In fact as discussed in [83, 85, 110] people with similar interests tend to stay in touch for longer period of times, with respect to people without overlapping interests. Such groups of *similar* people can be detected by means of community detection algorithms, that measure several metrics for identifying communities. Hence, service clients can propagate a query first among members of the same community in order to find people with matching interests.

Chapter 3

A Service-Oriented ZigBee Gateway

The pervasive presence of low-power devices in SE increases the smartness of the environments where they are deployed however, their presence gives rise to numerous issues. We are interested in affordable and open-source solutions for the interoperability of low-power devices in a SE. Our approach to the device interoperability is to install an integration gateway within an SE. This gateway exports the functionalities provided by low-power devices, such as sensors and actuators to other devices, and hides all the technical details concerning their access. This chapter presents the design and the implementation of ZB4O (ZigBee API for OSGi Service Platform), an integration gateway for ZigBee devices based on the OSGi model (see Section 2.1.1).

The key aspects of ZB4O [21] are:

- it provides a richer and flexible gateway for the ZigBee network;
- it extends the OSGi framework with a mechanism that represents the ZigBee devices as standard OSGi services.

ZB4O was inspired by SAIL [17], our early proposal for a general-purpose architecture for the integration of devices. ZB4O refines the idea of SAIL and provides an implementation for the ZigBee standard. The interoperability between ZigBee devices and other applications installed in an SE (for example IP-based applications) is not a standard process. Such interoperability is often implemented with vertical solutions that are not sufficiently flexible to be extended or customized. On the other hand, the OSGi Service Framework, defined by the OSGi Alliance, provides a service-oriented architecture. Therefore the design of a gateway based on these two key-technologies (ZigBee and OSGi) could represent a driver for integrating low-power devices in an SE.

We first present some use-cases that define our reference scenario and then we introduce the design of ZB4O as a layered-architecture. The second part of this chapter we evaluate two use-cases. We describe the use of ZB4O with two EU projects

and with UPnP protocol and the RESTful web-services. The results presented in this chapter have been published in [21, 22, 23, 24, 25, 26, 27].

3.1 The Reference Scenario

Our reference scenario is driven by two use cases, named UC1 and UC2, which all focus on the possibility of accessing ZigBee devices in a Smart Environment.

UC1 - Seamless plug & play

A new ZigBee device is installed at home, for instance a standard smart plug for monitoring home energy consumption. As soon as the user plugs in the ZigBee device, the user is able to immediately discover it without installing any specific driver. All the ZigBee devices are recognized and integrated autonomously.

UC2 - Multi-protocol bridge

The user plugs in a new ZigBee medical device. Similarly to UC1, the device is integrated within the Smart Home. In this case the device has to be available not only in the local network, but also remotely. A remote client, which supports a standard protocol (such as IEEE 11073 or HL7 as outlined in the Continua Health guidelines[1]), can interact with the ZigBee medical equipment installed at home.

ZB4O is designed by keeping in mind a typical scenario of the SE, namely the Smart Home. The reason for such design choice is twofold. Firstly the Smart Home is the natural environment in which the ZigBee devices might be deployed and tested. Secondly, the ZigBee Alliance has standardized the set of devices specifically designed for the home, the result of such standardization process is the Home Automation profile. Nevertheless we consider that UC1 and UC2 might be applicable also outside the home domain. More specifically both of the use cases are meaningful also in other environments such as hospitals, airport terminals or even in wider places like cities. The experiments done in this thesis do not attempt to validate our solution in such contexts, rather we focus on the more controllable scenario like a home.

3.2 The ZB4O Gateway

The ZB4O gateway has been designed keeping in mind the following guidelines:

- Dynamic discovery of nodes: ZB4O exploits the discovery mechanisms of ZigBee. As soon as a new node joins the network, ZB4O registers OSGi services that represent the APOs implemented by the node;

[1] http://www.continuaalliance.org/

- abstraction of ZigBee devices: ZB4O recognizes ZigBee devices adhering to the ZigBee profiles (e.g. On/Off Switch device, Remote Control device, Light Sensor device) and it abstracts them. This allows external applications to ignore how to create a ZigBee frame and to focus only on how to gather data from the ZigBee nodes with more intuitive APIs. As already observed in Section 2.1.2 the ZB4O approach is rather different from the solution adopted from the ZigBee Alliance [45];

- extension mechanisms for ZigBee devices: the ZigBee Cluster Library [37] defines an extended set of clusters to be used with the ZigBee devices. However, it enables also the definition of customized clusters by third parties. ZB4O uses this feature and it allows including customized clusters. Such clusters will be used during the refinement process of a ZigBee device;

- modular integration mechanisms: ZB4O maps the ZigBee devices to several OSGi services, which may expose the access to ZigBee applications with high-level protocols.

ZB4O is based on a three layered architecture: namely the Access, Abstraction and Integration layer according to the model proposed in [17]. Figure 3.1 shows an overview of ZB4O model.

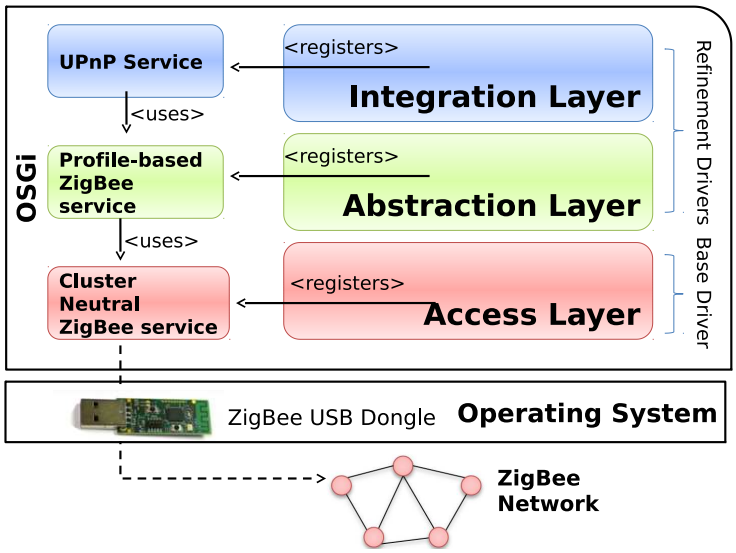


Figure 3.1: The ZigBee service model.

The Access Layer directly communicates with the ZigBee network by means of a network adapter (called USB dongle, see Figure 3.1). According to the OSGi Device Access Specification, the component implementing the Access Layer is called Base Driver (in this thesis it is named ZigBee Base Driver), whereas the components

of the upper layers are called Refinement Drivers. The Access Layer registers for every ZigBee device a proxy that does not implement any specific cluster rather it provides some simple methods for messaging with the ZigBee device. The Abstraction Layer adds semantic to the proxy services registered by the Access Layer. The Abstraction Layer detects the service proxies of the Access Layer and it registers new OSGi services according to the ZigBee profile implemented by the ZigBee devices (i.e. Light devices, Thermostat devices). Note that, although the Abstraction Layer is designed as a generic layer, it should include a specific refinement driver for each ZigBee profile that is implemented by nodes of the network. Currently ZB4O implements the Home Automation (HA) profile, but it is planned to implement other refinement drivers starting from the healthcare profile. The Integration Layer, finally, maps the the ZigBee services refined by the Abstraction Layer to an application-level protocol (Figure 3.1 shows the UPnP exporter as described in Section 3.3.3). The Integration Layer follows the standard OSGi event mechanism. As soon as a new proxy is installed, the exporter acts as protocol translator by injecting the ZigBee devices into the appropriate network. For example, ZigBee devices can be wrapped as virtual UPnP devices or as REST end-points.

3.2.1 The Access Layer

The core component of this layer is the ZigBee Base Driver (ZBD) that:

- implements the ZigBeeDevice APIs;
- uses the Simple Driver APIs.

The ZBD uses the Simple Driver APIs to interact with one or more ZigBee devices plugged to the PC. The Simple Driver APIs provide the hardware abstraction layer of ZB4O. Figure 3.2 gives an overview of the Access Layer.

The ZigBeeDevice APIs provide a model for the ZigBee nodes and EndPoints (EP). A ZigBee node is described in terms of network attributes such as IEEE address, network address, node type and PAN (Personal Area Network) ID. An EP is described in terms of attributes such as profile ID, input cluster ID, output cluster ID, endpoint ID and device category. The ZBD instantiates an OSGi service as soon as it discovers a new EP in the ZigBee network. For each EP, the ZBD creates and registers an OSGi service called `ZigBeeDevice`. The `ZigBeeDevice` acts as a proxy for the EPs. In particular, when an application interacts with a `ZigBeeDevice` service, the ZBD forwards the messages to the corresponding EP on the ZigBee network. Vice versa, the messages generated from the EPs are forwarded by the ZBD to the applications waiting for them. The Simple Driver APIs define an industry-independent hardware interface including most of the common mechanisms needed to interact with a ZigBee network. The Simple Driver APIs are implemented by the ZigBee network drivers that directly interact with the ZigBee network. The

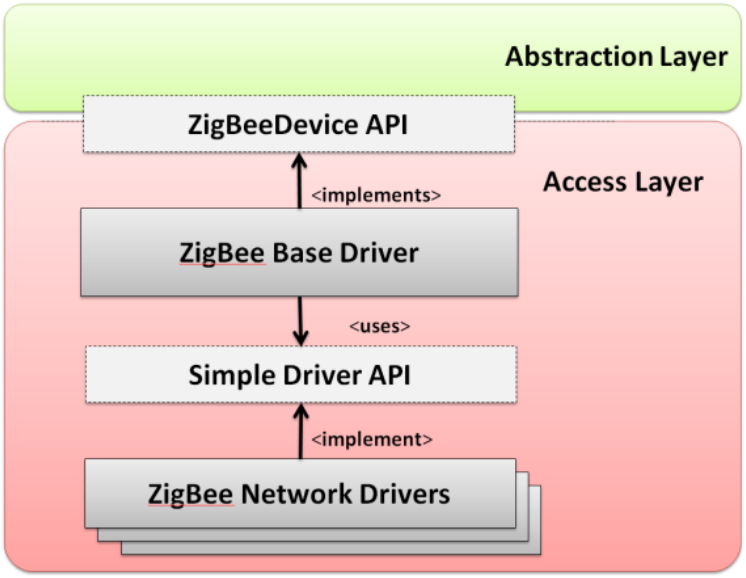


Figure 3.2: The Access Layer.

Simple Driver APIs have been designed by taking into account a set of high-level primitives for the interaction with the network:

- Create/join the network: this primitive includes some operations to configure the dongle in the ZigBee network. The configuration can be performed by specifying the channel to use, the security key and the network identifier (pan ID). It is possible to interact with the dongle to create a new network or to join an existing one as a coordinator, router or as a end device;

- inspect the ZigBee node: by means of this primitive, the ZBD inspects the IEEE address of a given node. Moreover ZBD can fetch the list of EPs available on the node together with the description of each EP;

- binding to an EP: this primitive allows the ZBD to bind to the ZigBee hardware plugged to the PC (as an example the USB dongle) with one or more EPs in the network. Moreover it is possible to bind two remote EPs with each other without the enrollment of the USB dongle.

- send/receive messages to/from an EP: these primitives enable the ZBD to send messages to the EPs or to receive messages from the EPs. The communication with an EP can be synchronous or asynchronous. In the first case the operation (either send or receive) is implemented by means of a conventional synchronous Java method invocation, whereas in the latter case the communication is part of a more complex communication flow implemented by means of the observer pattern;

- Inspect the status of the driver: these primitives allow the ZBD to inspect the properties of the driver such as the assigned IEEE network address or the channel in use. Such configurations exploit the management and the monitoring interface of the ZigBee protocol stack.

Discovery of ZigBee nodes

The node discovery can be implemented in an reactive or proactive mode. The proactive discovery mode allows to be notified directly from a ZigBee node as soon as it becomes available in the network. Conversely, with a reactive discovery mode, the ZBD queries the network to discover the nodes. Note that the proactive mode is optional in the ZigBee specification.

The ZBD implements both of the discovery modes with periodical browsing and event listening algorithms. First, the periodical browsing algorithm exploits only the ZigBee clusters that are mandatory for the ZigBee compliant devices. This ensures the compatibility of the ZB4O gateway with all the ZigBee networks. The periodical browsing exploits the addressing tree defined by the ZigBee standard. The ZBD is a logical tree (see the addressing link arrows in Figure 3.3) rooted on the coordinator of the network. The intermediate nodes act as routers and the leaves act as end-devices or routers. The addressing tree is browsed by means of the `IEEE_addr_req` and `IEEE_addr_rsp` messages, which are mandatory in the IEEE standard (an alternative is the `MGMT_lqi` request and response). The `IEEE_addr_req` message is used to request the IEEE address of a node together with the network address list of all the nodes connected to it. The `IEEE_addr_rsp` message carries the answer. The periodical browsing starts forming the network coordinator (whose address is always $0x0000$). The result is an accurate view of the network. However the drawback of this discovery mode is a non-negligible network overhead. A reasonable configuration is to execute the periodical browsing with a low frequency (for example every hour) or to run such discovery algorithm only on demand.

Second, the event listening algorithms uses some optional ZigBee clusters. Such clusters provide an automatic notification mechanism of new nodes that join the network. Hence, this discovery mode results more efficient than the periodic browsing. Specifically, this mode relies on the `Device_annce` message. A ZigBee node sends via broadcast the `Device_annce` message as soon as it joins the network. Unfortunately, the dual message to notify when a ZigBee node leaves the network does not exist in the ZigBee specification. For this reason the network status may become inconsistent and less precise than the periodical browsing.

Discovery of EPs and ZigBee devices

Once the ZBD discovers the new ZigBee nodes, it can inspect the EPs available in the ZigBee nodes. To this purpose the ZBD fetches from every node some relevant information that are needed to detect the EPs. The ZBD exploits two pair

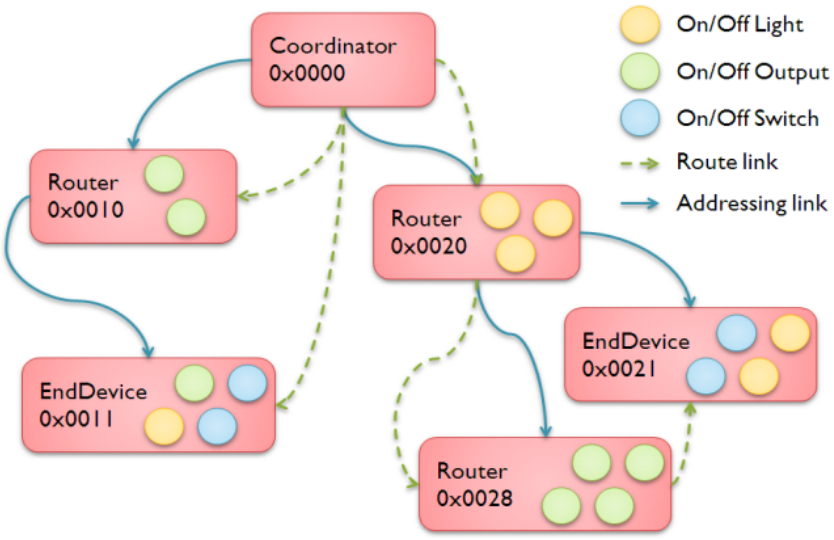


Figure 3.3: An example of addressing tree and device discovery.

of mandatory messages on the ZigBee protocol: `Active_EP_req`, `Active_EP_rsp` and `Simple_Desc_req`, `Simple_Desc_rsp`. The first pair (`Active_EP_req`, `Active_EP_rsp`) of messages is used to retrieve the list of ZigBee EPs available on the node. The second pair of messages (`Simple_Desc_req`, `Simple_Desc_rsp`) retrieves the list of the clusters available on a specific EP. Once the inspection of a ZigBee node is completed, the ZBD registers a `ZigBeeDevice` service for every EPs found. Figure 3.3 shows an example in which the ZBD browses a ZigBee network. Except for the coordinator, these nodes implement On/Off Light and On/Off Switch. Note that the solid lines in the figure define the network links among the nodes on the routing tree, and the dotted lines represent the sequence in which the ZBD queries the nodes.

3.2.2 The Abstraction Layer

The Abstraction Layer is responsible for refining all the `ZigBeeDevice` services registered by the Access Layer. The refinement process of ZB4O is one of the core-aspects of our solution. Indeed, ZB4O is able to recognize ZigBee devices adhering to a set of pre-installed profiles as well as devices implementing custom ones. The final result of the refinement process is the registration of an OSGi service providing all the features that the ZigBee device actually provides.

In order to emphasize the added-value of the Abstraction Layer, we provide an example. Suppose a Pump Controller device implements four mandatory clusters, namely Pump Configuration, On/Off, Scenes and Groups clusters. The Pump can implement some optional clusters, for example the Pressure, Temperature and Flow measurement clusters. The refinement process of the Abstraction Layer selects the set of clusters that are actually implemented by the ZigBee device. This check allows to manage both ZigBee devices adhering to a the standard ZigBee profiles but also

devices implementing a custom profile. In the previous example, the manufacturer of the Pump Controller device might define some proprietary diagnostic clusters. In this case, the ZB4O gateways only requires a component that registers the diagnostic cluster in the cluster factory registry. In turn, the Abstraction Layer is able to refine the Pump Controller with all the clusters that the device implements.

The architecture of the Abstraction Layer is composed by:

- a number of refinement drivers, specifically one for every ZigBee profile;
- the ZigBee Cluster Library;
- (optionally) a number of third-party components that can be used to register non-standard clusters.

Figure 3.4 gives an overview of the Abstraction Layer with the Home Automation profile driver (HA Driver).

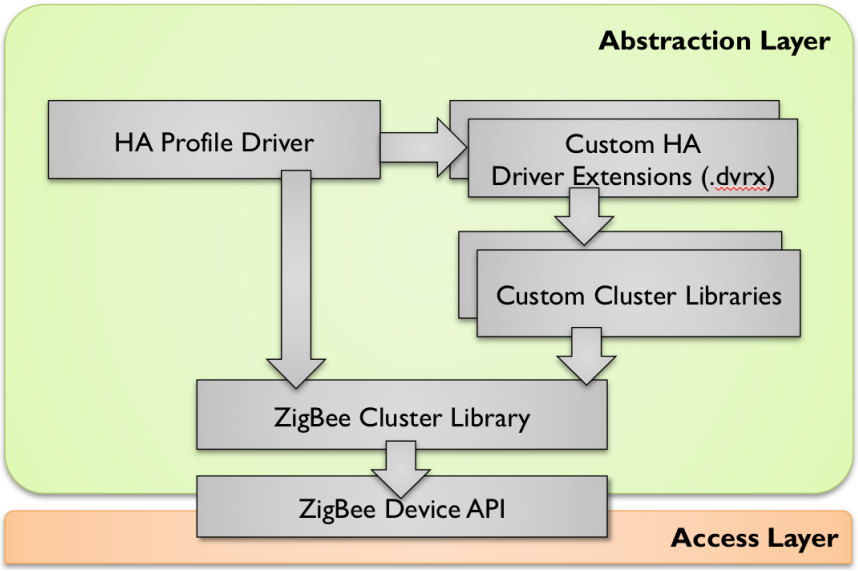


Figure 3.4: The Abstraction Layer.

The HA profile driver defines some common devices of a Smart Home, such as On/Off Switch, Remote Control or Door Lock. They are represented as a set of hierarchical Java classes that model commons HA devices.

The refinement process of the HA profile is shown in Figure 3.5. The HA Driver monitors the `ZigBeeDevice` services registered by the Access Layer. More precisely it inspects the ProfileID and DeviceID of the `ZigBeeDevice` so that to select the proper device factory to invoke. In turn, the factory can install a proxy instance of the remote ZigBee EP (see Figure 3.5). To this end, the Abstraction Layer provides a registry for the device factories. The registry is used to keep device factories for both standard and non-standard profiles. The HA Driver registers all its device

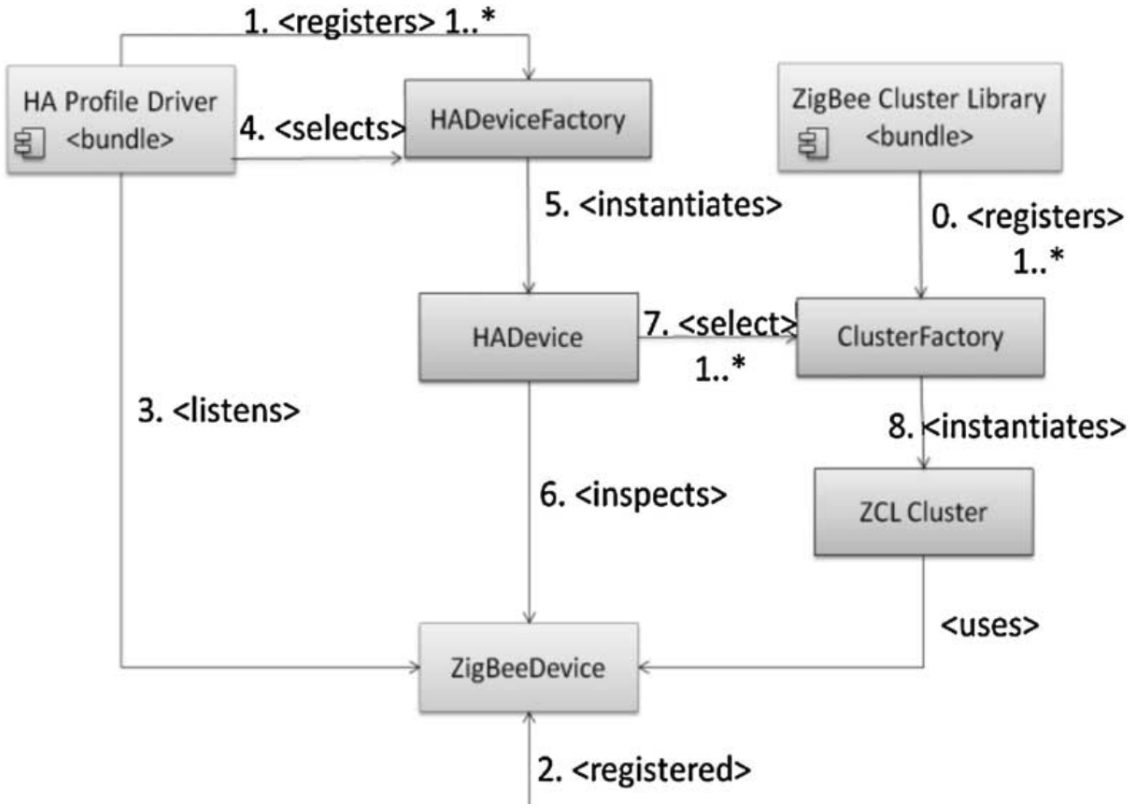


Figure 3.5: The refinement process of the Abstraction Layer.

factories during the start-up phase. In turn, every device factory verifies that the `ZigBeeDevice` implements all the mandatory clusters defined by the ZigBee profile.

In summary, the refinement process of the Abstraction Layer is achieved in two steps:

1. the instantiation of the device factories specific of the ZigBee profiles;
2. the instantiation of the custom clusters implemented by a ZigBee device.

The ZigBee Alliance defines the ZigBee Cluster library. Such library is used to ease the interoperability of devices and to re-use messages among devices. According to this approach, the ZigBee devices adhering to the same profile that have to reuse clusters already defined in the ZCL. ZB4O adopts a similar approach for the Abstraction Layer. All the HA devices use the clusters defined by the ZCL bundle, whose factories are registered at the bundle start up. The development of new profiles is simplified because they can reuse the ZCL clusters. Moreover new versions of the ZCL library can either be upgraded by using the OSGi life-cycle mechanisms or extended by registering only the missing clusters with the proper registry. The ZCL bundle offers several abstract classes and utilities to ease the design of new clusters. In particular, the abstract class Attribute is used to define

new cluster attributes. It inherits the common commands (read, write and report). The ClusterBase class refines a cluster with a collection of attributes and commands. The Serializer and Deserializer classes, respectively, parse and un-parse payload of the cluster commands and of the responses. Moreover, generic device and cluster factories are available to easily implement the plug-in mechanism for both new devices and clusters.

The Abstraction Layer hides the internal details of the ZigBee protocol in order to simplify the interaction with the ZigBee network. To this purpose, the HA Profile Driver bundle implements glue code that re-defines the clusters of the ZCL library with simpler versions. An interesting example is a high-level application willing to be notified of the status change of the On/Off Light Device. By adopting the clusters defined in the ZCL (namely the OnOff cluster), the application needs to know at least the following items:

- how to configure the Reporting command: this command is used to be notified about the status change;
- how to send the Reporting command;
- how to bind the Light Device to the USB dongle;
- how to receive the notification messages.

Such complexity can be avoided by using a simplified version of the cluster provided by ZB4O. Indeed, with our approach the caller has only to instantiate a Java listener object. As soon as the value of the Light Device changes, the listener is called back with. The current implementation of ZB4O offers simplified versions for a large number of standard ZigBee clusters.

3.2.3 The Integration Layer

The role of the Integration Layer is to export the ZigBee devices to different target networks. To this purpose ZB4O adopts a general-purpose solution. This feature goes beyond the existing solutions described in Section 2.1.2. Indeed, most of the existing works are designed to implement a protocol translation from ZigBee to one specific target network.

The OSGi bundles that implement the Integration Layer are named exporters. During this thesis we designed and implemented several exporters, with the goal of testing ZB4O in real application scenarios. Currently, ZB4O provides four exporters in particular:

- the GiraffPlus exporter [23] that integrates the GiraffPlus middleware with ZigBee network;
- the universAAL exporter [24] that integrates the universAAL architecture with the ZigBee network;

- the UPnP exporter that maps the ZigBee HA devices as UPnP devices;
- the REST exporter that maps the ZigBee HA devices as REST web-services.

Section 3.3 describes each of the exporters previously mentioned.

3.3 Evaluation of Use Cases

We evaluate the ZB4O gateway in use cases UC1 and UC2 from both a qualitative and quantitative perspectives. The objective of the evaluations is twofold:

- UC1: verify the integration of new ZigBee devices plugged at runtime;
- UC2: verify the interaction with ZigBee devices previously integrated by using different communication protocols.

The approach we follow differs according to the two perspective. In particular, a qualitative evaluation studies how our solution is effective in real deployment scenarios, such as the installation of ZB4O in a Smart Home, the interoperability of ZB4O with a robotic agent, and the integration with different protocols. We specifically focus on an assessment of the design complexity and on the effort needed to integrate ZB4O with existing platforms such as GiraffPlus and universAAL projects (see Section 2.1.1), as well as with the UPnP and REST protocols (see Section 2.1.1).

Quantitative metrics study various software metrics and provide an indication of the ZB4O performance in real deployments. In particular, we study typical software profiling metrics such as the use of the CPU, the number of threads allocated, the RAM occupancy and the number of classes loaded at runtime. This last analysis aims to verify also the possibility of installing the ZB4O gateway in embedded hardware devices, such as RaspberryPI or BeagleBone board[2].

Finally, it is worth to notice that the analysis of the existing solutions for interoperable gateways (reported in Section 2.1.2) revealed us the impossibility of making a direct comparison between ZB4O and a benchmark solution. This is due for two main reasons:

- the gateways that we analyzed implement only a subset of the features provided by ZB4O. This means that such a comparison of software artifacts different in their nature may easily be un-fair. In particular, we observed that not all the gateways provide a hardware abstraction layer for interacting with ZigBee dongles, not all of them support the ZigBee profiles, and most of the gateways are just designed to bridge ZigBee one single target network.
- The software of the most meaningful ZigBee gateways (those analyzed in table 2.1) is unavailable. Furthermore, these gateways require specific configurations

[2]raspberrypi: https://www.raspberrypi.org/, BeagleBone: http://beagleboard.org/bone

of the ZigBee network that are not always reported (or reported only at high level) in the published papers.

Hence, we decided to not compare ZB4O with respect to one or more benchmark gateways, rather we decided to extend as much as possible the experimentations so that to provide to the community robust results concerning its performance.

In summary use cases UC1 and UC2 described in Section 3.1 are evaluated as reported in Table 3.1.

Table 3.1: Evaluation table of ZB4O.

	qualitative evaluation	**quantitative evaluation**
UC1	✓	✓
UC2		✓

3.3.1 The GiraffPlus exporter

The GiraffPlus exporter is implemented within the ZB4O Integration Layer and it interacts with the GiraffPlus middleware installed on the robot. Figure 3.6 shows how the exporter exploits the functionalities provided by the middleware in order to implement the integration of the GiraffPlus middleware with the ZigBee network. The exporter uses a subset of methods from the APIs provided by the middleware to announce the presence of a new ZigBee devices and to publish the status readings of the sensors. The integration between ZB4O and GiraffPlus is made easier by the OSGi framework shared by the two architectures. As soon as the ZB4O stack notifies the exporter of a new device turned on, the exporter creates a descriptor compliant to the GiraffPlus formalism. The exporter then invokes the announce method specifying such descriptor as argument. In this way the information about the existence of a new device is shared among all the clients listening on the service bus. If a client subscribes (or has already subscribed) to the service bus, it will be notified with the relative descriptor. After the announcement of the ZigBee devices, it starts publishing messages regarding its readings and status changes. Each service subscribed to the relative context bus topic will receive the message. Figure 3.7 shows the sequence diagram of the scenario previously described. Similarly to the announcement, when a device is turned off the exporter calls the remove method with the relative descriptor as argument. In this way all the subscribers will be notified of the unavailability of that device. Our main goal is to test the whole architecture in order to let GiraffPlus be able to: discover and interact with several ZigBee networks installed in a Smart Environment. The hardware we adopted is shown in Figure 3.8 and described below:

- the GiraffPlus robot equipped with the CC2531 USB ZigBee dongle. The USB dongle is used as access point for the ZigBee networks;

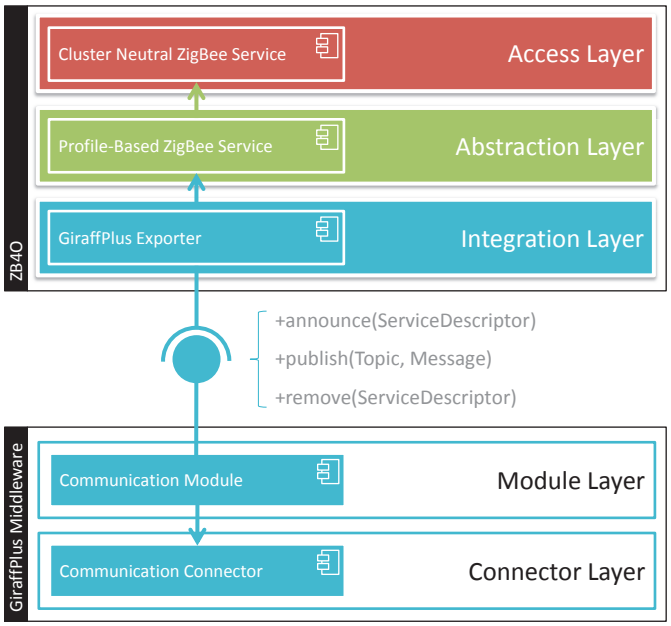


Figure 3.6: The component diagram of the integrated architecture.

- a collection of ZigBee devices implementing the Home Automation profile. In particular: *(i)* Generic devices such as On/Off switch, Remote Control, Door Lock, *(ii)* Lighting devices such as On/Off Light, Dimmable Light and Occupancy Sensor, *(iii)* HVAC devices such as Temperature Sensor and *(iv)* Intruder Alarm devices such as IAS Zone system.

A picture of the Smart Homes we tested in shown in Figure 3.9. The environment is equipped with several ZigBee networks deployed in the bathroom, living room, bedroom and in the kitchen. The figure also shows the path followed by the Giraff robot during the test. As soon as GiraffPlus moves close to a ZigBee network, the ZB4O framework performs the following actions:

- the Access Layer discovers the new devices from the ZigBee network, for every newly discovered device it creates a `ZigBeeDevice` object that acts as a proxy for the device;
- the Abstraction Layer checks if the `ZigBeeDevice` implements the Home Automation Profile. If it is the case, the Abstraction Layer refines the proxy;
- the GiraffPlus exporter announces the existence of a new service with the service bus.

3.3.2 The universAAL exporter

The LDDI (Local Device Discovery Integration) building block (see Section 2.1.1) is the universAAL component responsible of the integration of heterogeneous sensing

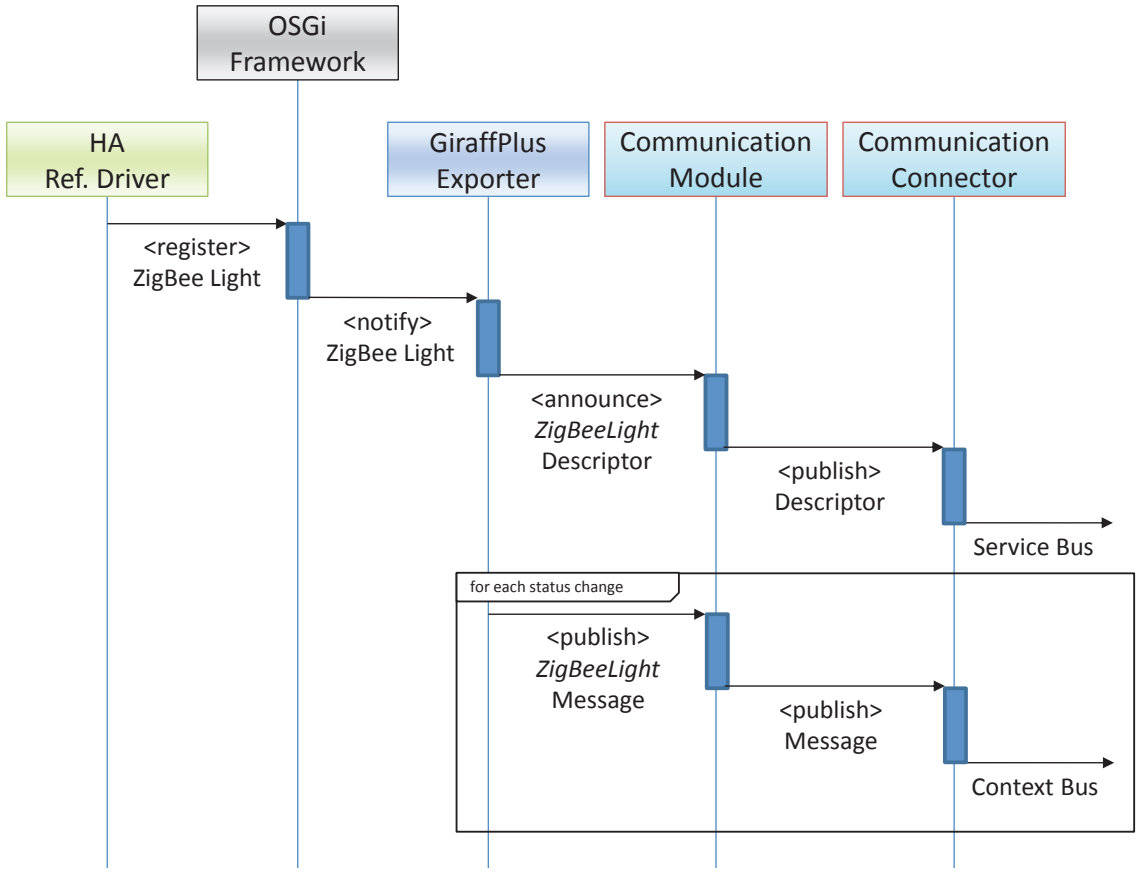


Figure 3.7: The sequence diagram of the announcing and publishing mechanisms for a sample device (ZigBee Light) in the integrated scenario.

technologies with the universAAL platform. LDDI is composed by several exporters, every export implements a protocol bridge between the universAAL Runtime Support and a specific technology. A simplified schema of the LDDI building block is shown in Figure 3.10.

We design a new exporter, namely the uAAL exporter, that integrates the ZB4O gateway with the universAAL platform. Such exporter interacts both with the Abstraction layer of ZB4O and with the Runtime Support of the universAAL middleware. More precisely, the uAAL exporter receives a notification from the Abstraction layer of ZB4O (see Section 3.2.2), some examples of relevant events are the registration or the removal of a ZigBee device from/to the network. A soon as the uAAL exporter receives an event concerning the registration of a new ZigBee device, then it generates a service registration message that, in turn, is published to the service bus. Similarly, for every event concerning a status change of a ZigBee device, the uAAL exporter generates a context event that is published to the context-bus. Such events (service registration and context events) are received by all the universAAL nodes that are connected to the service and context bus. The uAAL exporter al-

Figure 3.8: The robot and the sensors used: 1) The GiraffPlus robot, 2) The ZigBee dongle connected to the robot, and 3) The environmental sensors.

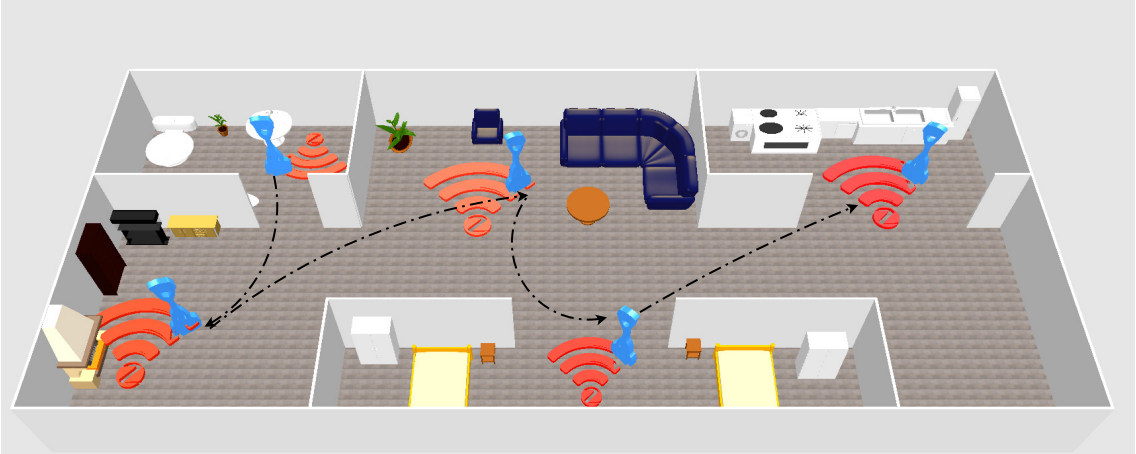


Figure 3.9: The Integrated scenario.

lows nodes running universAAL to interact with ZigBee devices by means of the service and the context bus so that all the complexity concerning the interaction with ZigBee specification is hidden. The implementation of the service and context bus is made more efficient with the possibility of specifying filters that regulate the notifications that a universAAL node will receive. Figure 3.11 shows how filters on the service bus are applied.

First, the universAAL node registers to the service bus without specifying any

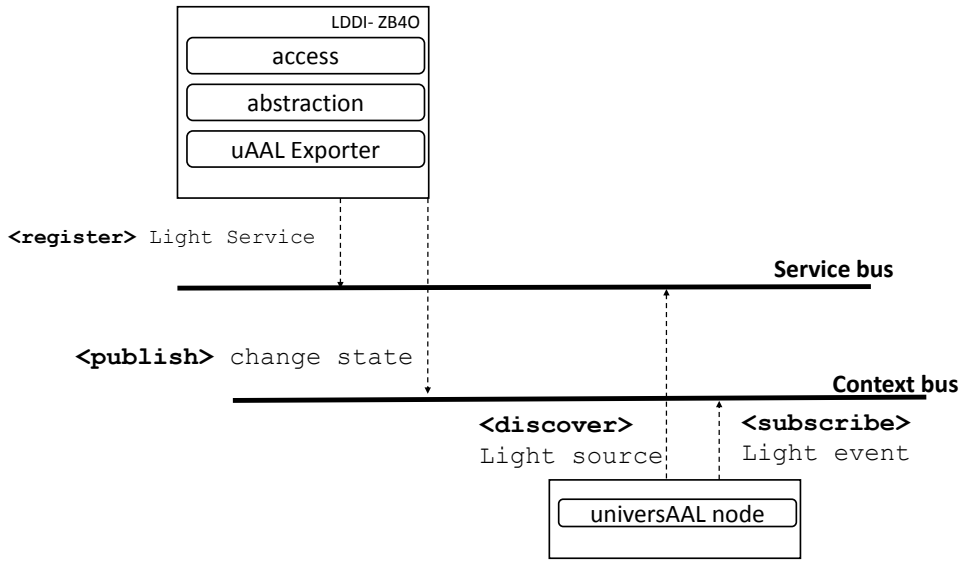


Figure 3.10: The LDDI building block.

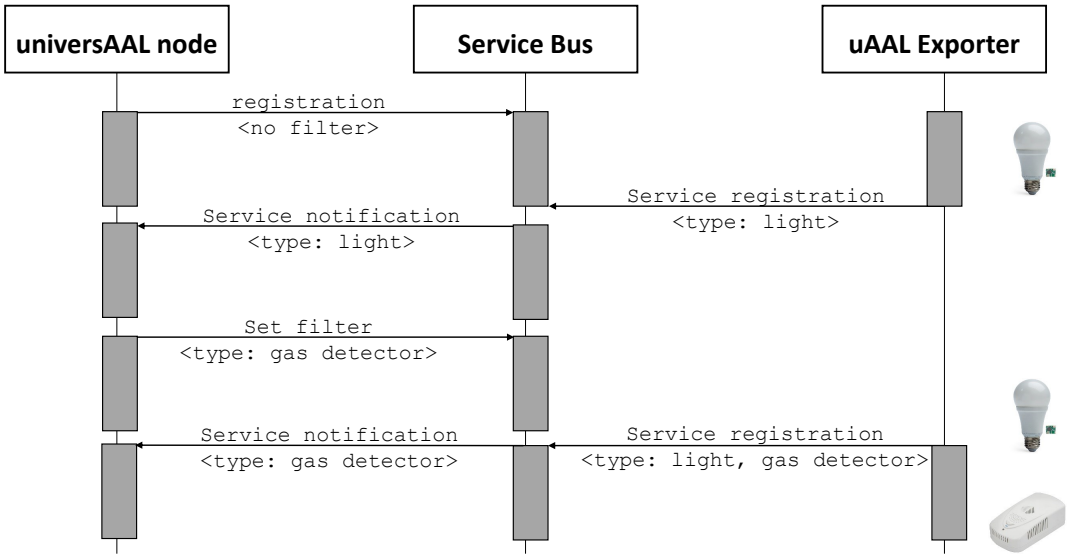


Figure 3.11: Example of filters for the service bus.

filter. As soon as the uAAL exporter generates a service registration message (ex. a new ZigBee light is switched on), the service bus notifies the universAAL node. Then, the universAAL node sets a filter in order to be notified only with service registrations concerning the gas detector device. Later on the uAAL exporter generates two service registration events concerning a new light and a new gas detector.

At this point the Service Bus applies the filters set by the universAAL node and the bus notifies the node only with the gas detector event.

3.3.3 The UPnP exporter

The UPnP protocol stack (see Section 2.1.1) is widely used in home automation and it is commonly considered as an enabling technology for application scenarios typical of SE. Figure 3.12 shows the integration of UPnP with the ZigBee technology.

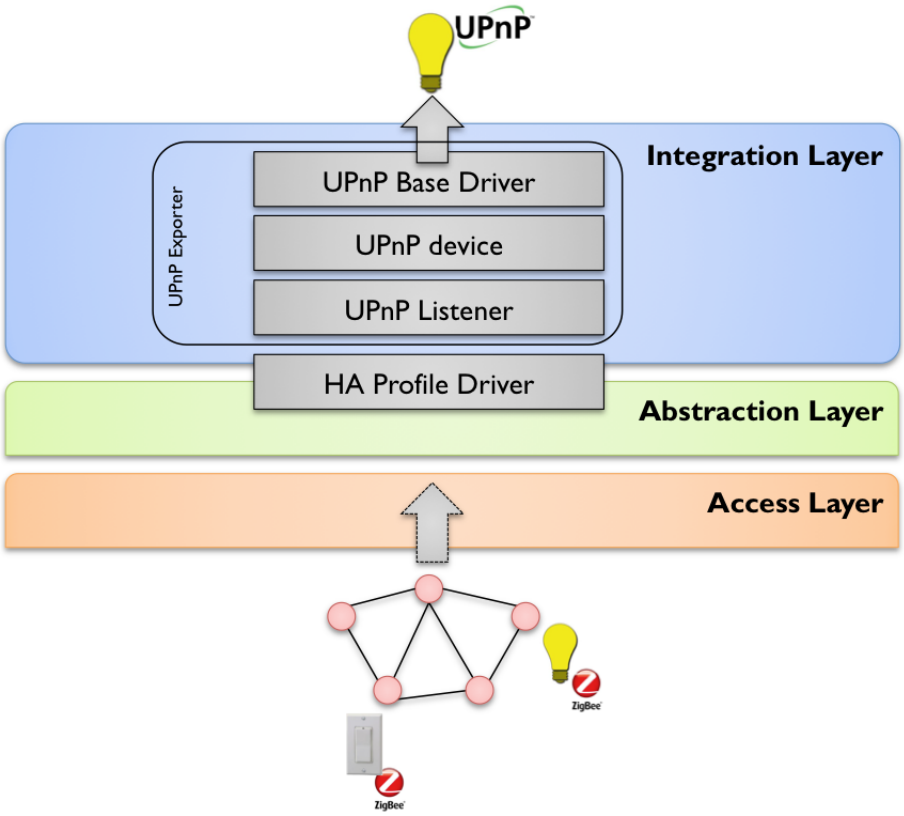


Figure 3.12: Integration between UPnP and ZigBee.

Figure 3.12 shows two ZigBee devices installed in the networks, namely the Binary Switch and the OnOff Light. Such devices are detected by ZB4O and they are abstracted.

Figure 3.13 depicts the sequence diagram describing how to export an OnOff Light as UPnP BinaryLight device. The starting point is the `announce` cluster sent by the ZigBee OnOff Light. The Access Layer receives the `announce` cluster and it reacts by registering a ZigBee Device in the OSGi framework. The Home Automation refinement driver detects the registration of the ZigBee Device, and it registers a ZigBee OnOff Light Device that refines the `ZigBeeDevice`. At the end, the UPnP exporter wraps the ZigBee OnOff Light Device as a `UPnPDevice`.

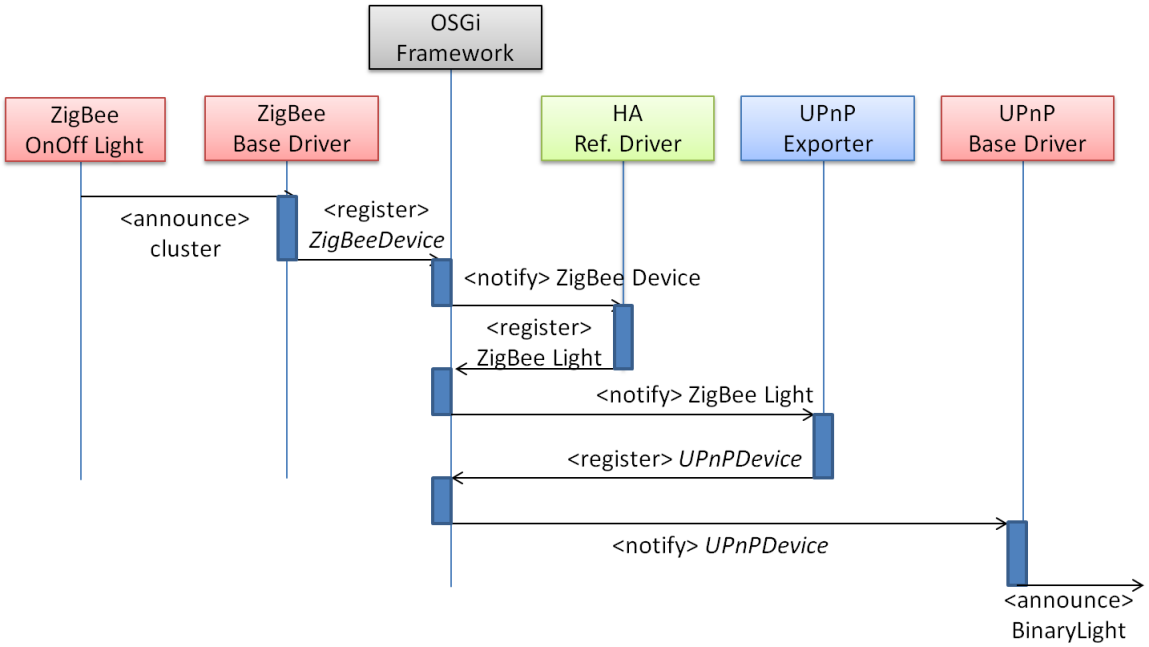


Figure 3.13: ZigBee OnOff Light as UPnP BinaryLight.

The `UPnPDevice` implements a general-purpose UPnP device as a Java objects. It provides all the features of the UPnP devices. In particular the UPnP exporter implements the `UPnPDevice`, `UPnPService`, `UPnPAction` and `UPnpStateVariable` java classes. The role of the `UPnPDevice` is to translate the behavior of an UPnP device in the respective ZigBee Device. In particular the `UPnPDevice` translates the UPnP state variables as ZigBee attributes. The final step is performed by the UPnP Base Driver [111] (provided by Apache Felix), whose role is to detect all the `UPnPDevice` and to announce them with the UPnP protocol. In this way, a UPnP Control Points can discover new UPnP devices, and they can interact with them via the UPnP protocol. The UPnP exporter will translate the UPnP commands in the respective ZigBee ones.

We test the UPnP exporter by taking into account the use cases presented in Section 3.1, in particular UC1 and UC2. Every use case is studied by using a quantitative evaluation by first describing the goals, the actors involved and the actions taken by actors. All the use cases have been configured with the ZB4O gateway deployed on a Windows 7, Intel i7 at 3.4 Ghz with 16 GByte of memory Ram. The version of OSGi was Apache Felix 4.2 and the Oracle JDK 1.6.

UC1: Seamless Plug & Play

The goal of UC1 is to test the performance of the ZB4O provisioned with the UPnP exporter when a number of devices join the ZigBee network. The actors involved are: the ZB4O gateway and 6 ZigBee devices of different types:

- 5 ZigBee devices operating at 2.4 GHz IEEE 802.15.4 compliant and implementing the Home Automation Profile (On/Off switch, Smart Plug and Gas Detector devices) ;
- 1 USB CC2531 dongle ZigBee device acting as ZigBee network entry point.

Figure 3.14 shows the hardware setup used for UC1.

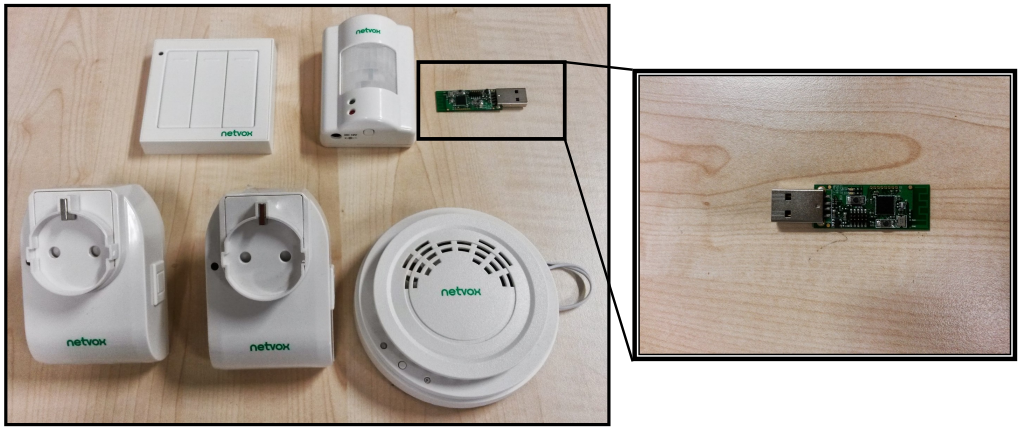


Figure 3.14: Hardware used in UC1.

UC1 is implemented by reproducing the steps shown in Figure 3.15. The 5

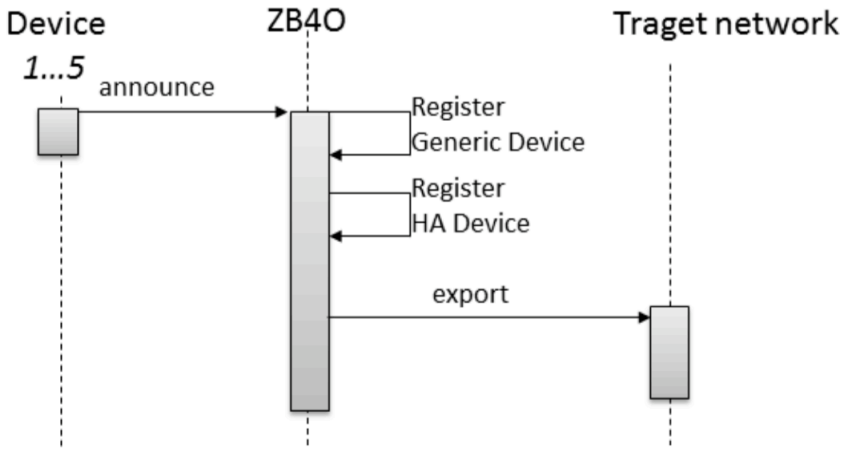


Figure 3.15: The UC1 execution flow with the UPnP exporter.

ZigBee Devices announce themselves in the ZigBee network, and the ZB4O gateway

registers the Generic Device and the refinement driver, in particular the HA Device. At the end, the ZB4O gateway exports the devices to a generic target network. The following metrics have been taken into account to evaluate the performance of ZB4O:

- average load of the CPU due to the ZB4O process;
- number of active threads in the ZB4O process;
- RAM memory occupation of the ZB4O process;
- number of loaded classes during the execution of ZB4O.

The results of UC1 are reported in Table 3.2.

Table 3.2: Performance metrics for UC1 with the UPnP exporter.

ZigBee Device/Metrics	1	2	3	4	5
CPU(%)	0.10	0.5	0.5	0.5	0.5
Thread (no.)	27	27	27	27	27
RAM (MByte)	6.7	6.9	6.9	6.8	6
Classes loaded (no.)	3364	3435	3437	3443	3443

The measurements show that the CPU average load is below 0.5%, with a peak of 1%. The number of threads is fixed to 27, but only 4 of them are created by ZB4O. In particular, 1 thread is used by the RXTX library (it is employed for controlling the serial port), and 3 threads are used by ZigBee Base Driver for the network discovery (see Section 3.2.1). The memory RAM footprint is always below 7 Mbyte, with a base footprint of 6.1 Mbyte used by the OSGi execution environment. The increase of the number of ZigBee devices has a negligible impact on the memory RAM and the CPU load. This highlights that ZB4O does not affect significantly the performance of the hosting PC. Finally it is worth noticing that the number of Java classes loaded by the ZB4O is around 3000. The number of classes loaded increases only when new ZigBee profiles are detected (around 5 classes for the HA profile). Hence ZB4O can control an increasing number of ZigBee device without requiring extra load. Such profile-based classes are loaded once, and used multiple times.

UC2: Multi-protocol bridge

The goal of UC2 is to test the interaction between a remote client and the ZigBee devices recognized by ZB4O. The actors involved are:

- the ZB4O gateway equipped with the UPnP exporter;

- a remote client able to browse the UPnP devices and to invoke UPnP actions;
- 1 ZigBee Device implementing the Home-Automation profile;
- 1 USB CC2531 dongle acting as network entry point.

UC2 is implemented by reproducing the steps shown in Figure 3.16.

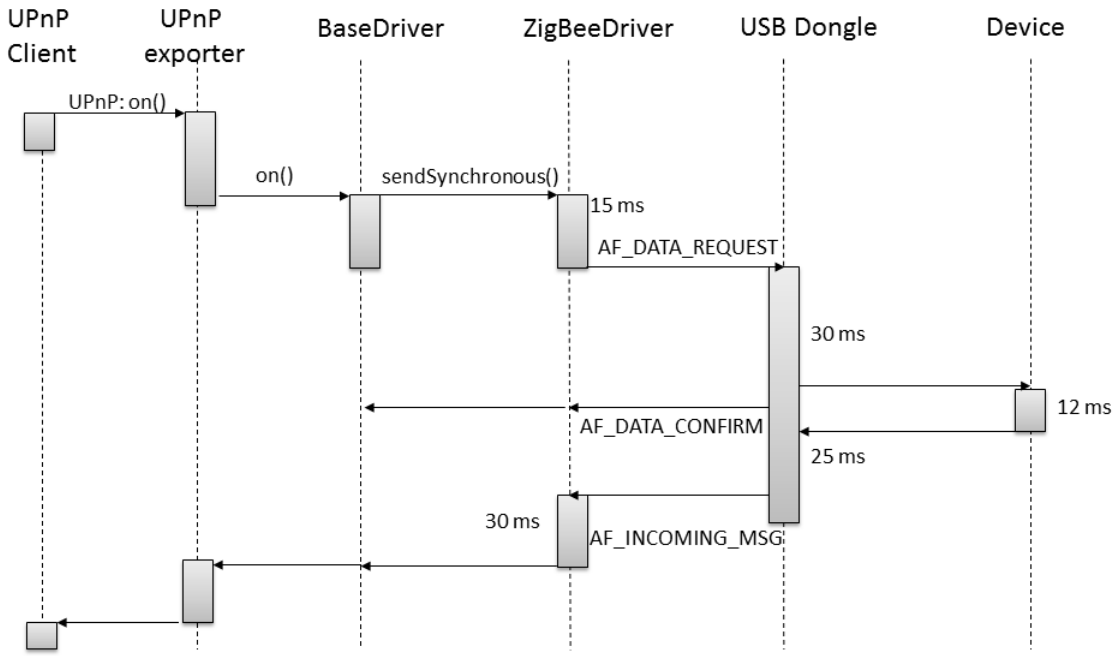


Figure 3.16: The UC2 execution flow with the UPnP exporter.

The UPnP client invokes the On (and Off) command, requesting the switching of a light, at varying request rate. It has been monitored the overhead introduced by ZB4O during the access of the device invoked by the UPnP client. The measures ignore the network latency from the UPnP client to the UPnP exporter (in the order of 300 ms) since such delay is not caused by the ZB4O gateway. The system works as expected for requests rates less than 10 per second. Higher request rates saturate the system, which discards the excess requests. Figure 3.17a and Figure 3.17b illustrate the system behavior.

Figure 3.17a reports the average number of requests served by increasing the service rate. The average number of requests is $7.67 req/sec$. The whole system (exporter, ZB4O and ZigBee network) scales up to the limit of 10 requests per

second after which the curve remains stable. This hard limit depends on the ZigBee dongle that we used for our experimentations, as detailed in the next. The average service time is 100 ms (see Figure 3.17b), and the components are illustrated in Figure 3.16. In the figure, the UPnP exporter sends the requests to the ZB4O BaseDriver and the BaseDriver invokes the appropriate function to interact with the ZigBee driver. This driver spends 15 ms to send the packet through the serial port. Once received the packet, the USB Dongle spends 30 ms to send the packet through the ZigBee network and to acknowledge the BaseDriver about the result of the operation. Similarly, the USB Dongle spends 25 ms to receive the response back from the ZigBee network and to notify the ZigBee Driver about the incoming message. Finally, 30 ms are needed to receive the message from the serial port and notify the BaseDriver. The latency introduced by the serial port is proportional to the serial bit rate, which is set at 38400 b/s because of instability due to the dongle hardware. As a result, the delay introduced by the hardware is 55 ms whereas the delay introduced by the serialization alone is 45 ms. The overhead introduced by the ZB4O stack is negligible with respect the described delays. Preliminary results obtained with a new version of the USB dongle here used, show that instability problems are gone and serialization can be speed up to 115200 b/s with a threefold reduction of the 45 ms serialization overhead. It is expected that the 55 ms hardware delay is reduced too, as the USB Dongle provides an optimized 8051 MCU core and a second generation IEEE 802.15.4 compliant system on chip. The above described latencies can be reduced on the dongle side, both by increasing the communication speed with the controlling PC and by increasing the processing times. The 12 ms time illustrated in Figure 3.16 is relative to networking communication and device processing, and cannot be reduced on the dongle side.

3.3.4 The REST exporter

In this last evaluation case our goal is to export the services offered by ZigBee devices with a RESTful approach. The RESTful approach offers a mapping between the CRUD operations and the HTTP methods, in this way we can use e.g HTTP POST method to create a resource or the HTTP GET method to read the status of a resource (note that with the REST approach every resource is uniquely identified with a URI). We implemented the REST exporter with two modules, namely REST4ZB and WebZB, as shown in Figure 3.18.

The REST4ZB implements the back-end and the WebZB implements the front-end. In particular a client interacts with the WebZB module to download a HTTP page (named the presentation page) with the user interface and the references pointing to the REST4ZB module. The presentation page has been designed to render a mobile-friendly GUI optimized for smart phones and tablets based on the jQuery framework.

The REST4ZB component is based on the API JAX-RS and it implements the

REST services. REST4ZB offers three basic REST services [3]:

- list of nodes: this service returns the list of nodes joining the ZigBee network;
- list of devices: this service returns the list of ZigBee devices refined by the Abstraction Layer. As described in Section 3.2.2, ZB4O refines the ZigBee nodes found in the network as profile-based devices according to one of the ZigBee profile supported;
- list of ZigBee services: this service returns the list of APOs provided by every ZigBee device, this list offers the services that a client can invoke.

For every node, device or service shown in the GUI, the REST client can invoke several methods in order to: read the status of the device, perform some operations of the device (according to the type of device) and subscribe to events generated by the device. Each of these operations are REST services exported by the REST4ZB component.

Similarly to the UPnP Exporter, the tests are organized in two use cases with goals, the actors involved and the actions taken by actors.

All the UCs have been configured with:

- the ZB4O and REST4ZB gateway deployed on Felix 4.2 with Oracle JDK 1.6 running on Windows 7 920 Quadcore, Intel i7 at 2.6 Ghz with 8 GByte of memory RAM;
- the REST UI deployed on a smart phone DualCore 1.5 GHz running Android KitKat 4.4.3 with 600Mb or memory RAM.

UC1: Seamless Plug & Play

The goal of UC1 is to test the performance of the ZB4O provisioned with the REST4ZB exporter when a number of devices join the ZigBee network. The actors involved are the same in UC1 with the UPnP exporter. In particular the ZB4O gateway and 6 ZigBee devices of different types:

- 5 ZigBee devices implementing to the Home Automation Profile (On/Off switch, Smart Plug and Gas Detector devices);
- 1 USB CC2531 dongle ZigBee device acting as ZigBee network entry point.

UC1 is implemented by reproducing the steps shown in Figure 3.19.

The 5 ZigBee Devices announce themselves in the ZigBee network, and the ZB4O gateway registers the Generic Device and the refinement driver, in particular the HA Device. At the end, the ZB4O gateway exports the devices to the HTTP network by registering three REST end-points (ad previously described). The following metrics have been taken into account to evaluate the performance of REST4ZB:

[3]The data-format used for the service invocation is based on the JSON format.

- average load of the CPU due to the ZB4O process;
- number of active threads in the ZB4O process;
- RAM memory occupation of the ZB4O process;
- number of loaded classes during the execution of ZB4O.

The results of UC1 are reported in Table 3.3.

Table 3.3: Performance metrics for UC1 with the REST exporter.

ZigBee Device/Metrics	1	2	3	4	5
CPU(%)	0.11	0.4	0.4	0.5	0.5
Thread (no.)	31	31	31	31	31
RAM (MByte)	4.6	4.6	4.6	4.8	4.8
Classes loaded (no.)	2344	2245	2134	2984	2925

The measurements show that the CPU average load is below 0.5%. The number of threads is fixed to 31, but only 8 of them are created by ZB4O. In particular, 5 threads are used by the Apache CXF library and by the Jetty Web Server library used to create REST web-services. Similarly to the UPnP exporter, 3 threads are used by ZigBee Base Driver for the network discovery (see Section 3.2.1). The memory RAM footprint is always below 5 Mbyte, with a base footprint of 4.6 Mbyte used by the OSGi execution environment. The increase of the number of ZigBee devices has a negligible impact on the memory RAM and the CPU load. This highlights that ZB4O does not affect significantly the performance of the hosting PC.

UC2: Multi-protocol bridge

The goal of UC2 is to test the interaction between a mobile client (a smart phone running REST UI) and the ZigBee devices recognized by ZB4O with the REST exporter. The actors involved are:

- the ZB4O gateway provisioned with the REST exporter;
- a smart phone running the REST UI;
- 1 ZigBee Device implementing the Home-Automation profile;
- 1 USB CC2531 dongle ZigBee device acting as network entry point.

UC2 is implemented by reproducing the steps shown in Figure 3.20a.

The REST UI client invokes one of the web services provided by the REST4ZB module. In particular the smart phones invokes the GET ATTRIBUTE service that

retrieves one of the attributes offered by a ZigBee device. The invocation of this web-service involves all the actors: the smart phone, the REST exporter, ZB4O and the ZigBee network. The GET ATTRIBUTE is implemented with the HTTP method GET and it requires two parameters: the IEEE address of the ZigBee device and the ID of the attribute to read. Note that REST4ZB also provides REST services for retrieving both the list of ZigBee devices as well as the list of clusters and attributes of every device. The invocation of the GET ATTRIBUTE introduces a latency that depends on the WiFi network used for the tests. We experienced latencies ranging in the interval $[100 - 500]$ milliseconds but such values are subject to high variance due to e.g. network load and on the strength of the signal of the WiFi network. Once the REST UI client invokes the web-service, REST4ZB processes the invocation and it interacts with the BaseDriver as described for Figure 3.20a with the same latencies described for UPnP exporter. We analyzed the average number of requests served by increasing the service rate, the graph is shown in Figure 3.20b. Similarly to Figure 3.17a the average number of requests served scales up to the limit of 10 requests (in this case the average is $7.72 req/sec$) after which the curve remains stable. Also in this case, this hard limit depends on the ZigBee dongle that we used for our experimentations.

3.4 Summary

Concerning device interoperability, this thesis describes the ZB4O (ZigBee API for the OSGi Service Platform) gateway which provides a rich and flexible gateway for the ZigBee network. ZB4O extends the OSGi framework with an open mechanism to integrate the ZigBee standard with a service-oriented approach. The most important achievements of ZB4O are:

- Access to the ZigBee network: ZB4O implements a hardware-independent mechanism for accessing the ZigBee network. We define a general-purpose API to access the ZigBee devices in a standard-way without introducing any constraints for a specific ZigBee vendor.

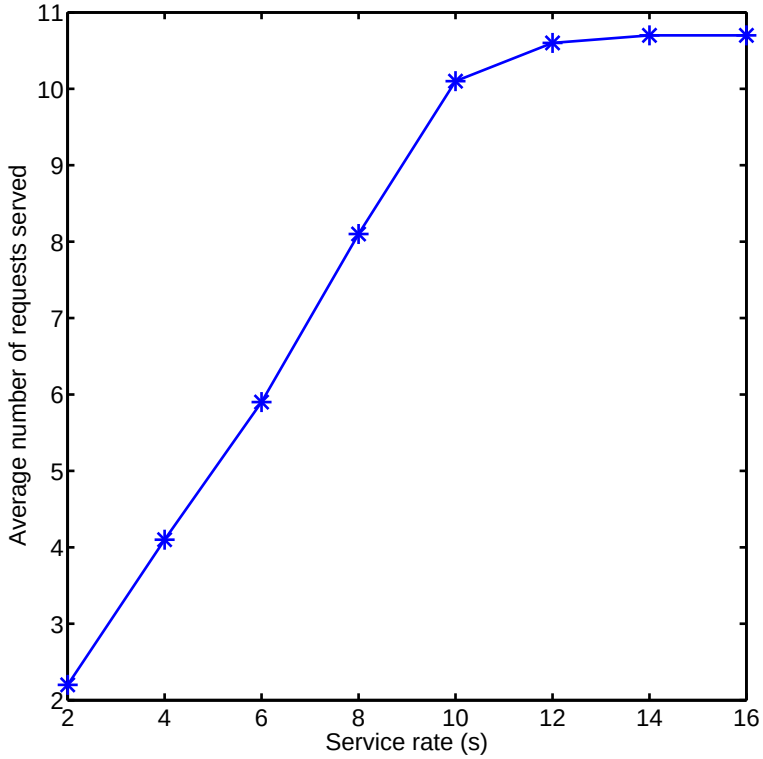
- Abstraction of ZigBee devices: ZB4O recognizes ZigBee devices adhering to the ZigBee profiles (e.g. On/Off Switch device, Remote Control device, Light Sensor device) and abstracts them. This enables external applications to ignore how to create a ZigBee frame, and to focus only on how to gather data from the ZigBee devices using intuitive APIs.

- Extension mechanisms for ZigBee devices: the ZigBee Cluster Library (ZCL [37]) defines an extended set of clusters to be used with ZigBee devices. This library also facilitates the definition of custom clusters, which can be defined for third-party vendors. ZB4O offers the possibility of integrating custom clusters in the gateway with a mechanics named as the refinement process.

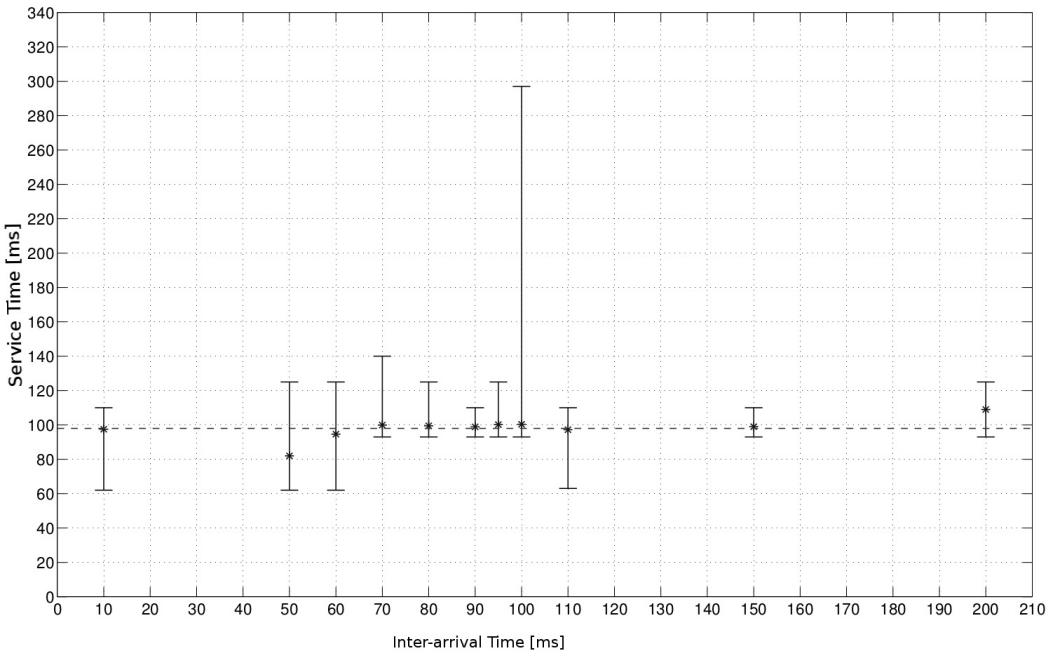
- Modular integration: ZB4O represents each ZigBee device as an OSGi service. The OSGi service representing the physical device is wrapped with one or more exporters thus implementing a software bridge between ZigBee and different technologies. We design and develop several exporters so that it is possible to access the ZigBee network from: UPnP, REST web services and using GiraffPlus and universAAL platforms.

ZB4O has attracted the attention of several researchers and universities as well as various important private companies. Finally, ZB4O[4] is now an open-source project with a growing community of developers and passionate supporters, it is hosted under the AALOA[5] association and is released under the Apache Software License.

[4]http://zb4osgi.aaloa.org/
[5]http://www.aaloa.org/



(a) Requests served with increasing service rate in UC2 with UPnP exporter.



(b) Service Time in UC2 with UPnP exporter.

Figure 3.17: Performance evaluation in UC2 with UPnP Exporter.

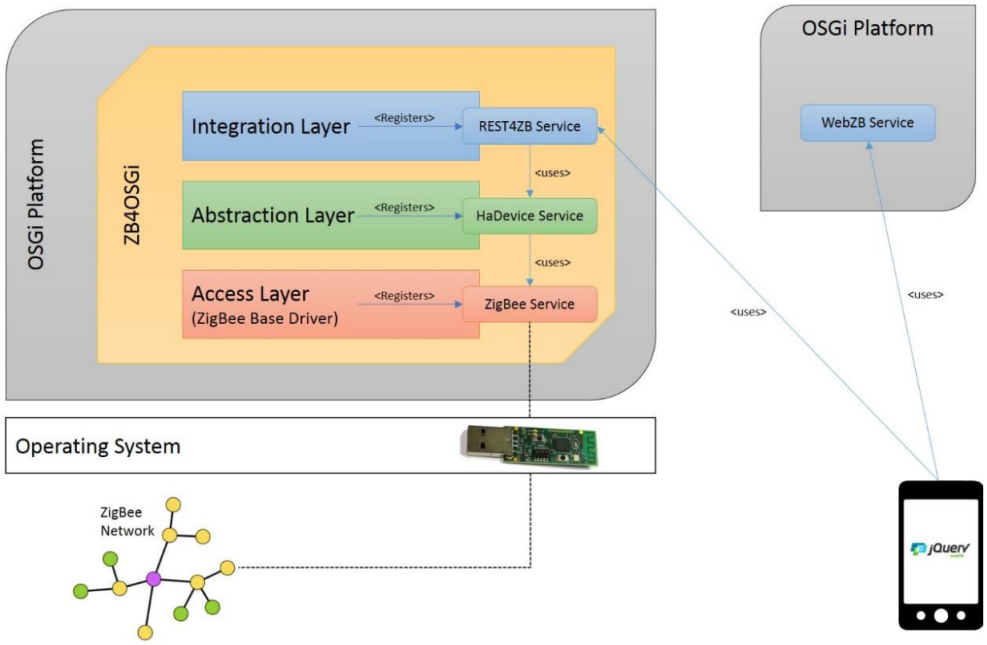


Figure 3.18: The REST exporter.

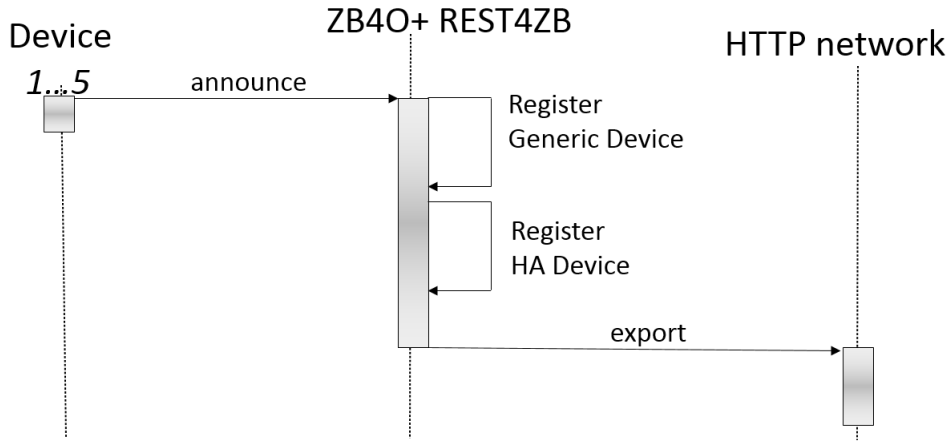
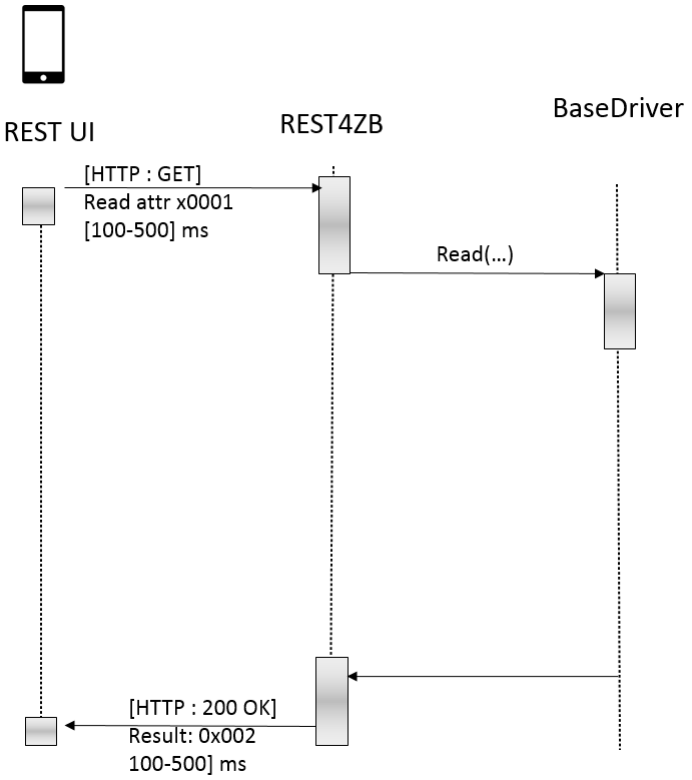
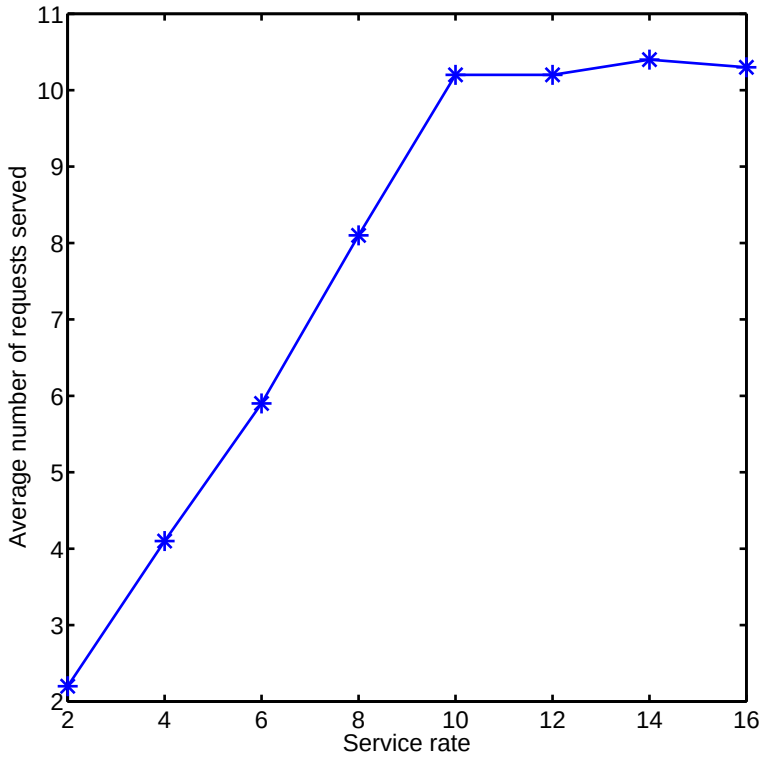


Figure 3.19: The UC1 execution flow with the REST exporter.



(a) The UC2 execution flow with the REST exporter.



(b) Requests served with increasing service rate in UC2 with REST exporter..

Figure 3.20: Performance evaluation in UC2 with REST exporter.

Chapter 4

Service Discovery in Mobile Social Networks

The progress of knowledge on MSN (Section 2.2.1) has raised several novel research aspects. The richness of the resources (sensors, hosted and self-generated data, connectivity etc.) of the devices that make up an MSN leads to the problem of how to share and distribute such resources among users. A natural solution is to organize the MSN according to a service-oriented model, in which resources are made available in terms of services offered by the devices themselves. In fact, a recent work [112] cites service discovery as a challenging open issue in MSN, and another work on opportunistic networking [52] describes the management of resources and services as an open research issue. On the other hand, the development of a service-oriented MSN is still in its infancy and relatively few works have addressed this issue [113].

This chapter describes two new algorithms for service discovery in MSN. The algorithms we propose, namely SIDEMAN (ServIce DiscovEry in Mobile sociAl Networks) and CORDIAL (COllaborative seRvice DIscovery ALgorithm), implement a service advertisement and query strategy specifically designed for MSN. In particular:

- *service advertisement*, is the advertisement of services available in the MSN. This is achieved by discovering and recognizing social communities and by the proactive diffusion of service advertisements among people with similar interests;

- *service query*, is the process of requesting a missing service from a fellow user. This is implemented by a controlled query propagation mechanism aimed at avoiding the extensive use of indiscriminate flooding.

We first present our reference scenario by describing how mobility and sociality affect service discovery algorithms in MSN, then we present a simple model for describing the service discovery problem more formally. We state the service discovery

problem and first describe SIDEMAN and then CORDIAL algorithms. The performance evaluation of SIDEMAN and CORDIAL algorithms is presented in Chapter 5. The results presented in this chapter have been published in [28, 29, 30, 31, 32, 33].

4.1 The Reference Scenario

Our reference scenario is made up of people who move in an environment and establish meaningful social relationships with other people. The environments we consider have different geographical dimensions e.g. a university campus, an airport terminal, a district or a city. People in our scenario carry smart objects in their pockets such as smart phones, smart watches or sensorized wrist bands, all of them forming an MSN. We will refer to the smart objects in the MSN both as nodes and devices.

As discussed in Section 2.2.1, our scenario is characterized by two aspects: *(i)* devices offer the functionalities they provide with a service-oriented approach (Section 2.2.1), and *(ii)* movements of people affect the communication opportunities among devices.

Devices offer different types of hardware and software resources. They have different types of network interfaces, such as short-range interfaces (WiFi, Bluetooth or ZigBee) or long-range interfaces (UMTS, 3G and LTE). Devices are also equipped with sensors such as accelerometers, 3-axis gyroscopes and GPS receivers. We are interested in studying the possibility of sharing such resources (hardware and software) in terms of services with other devices in the environment.

Human mobility is the second aspect that characterizes our reference scenario. In our scenario, people (and the devices associated with them) move in the environment over time. We consider that people move according to the objectives and activities arising from their social relationships [16]. The mobility of people profoundly affects the communication opportunities among the devices.

We do not assume a centralized MSN architecture with a reliable and stable network infrastructure. Instead, we consider that two devices can interact only if they lie in the transmission range of each other (which is typical of a distributed MSN). As in the Opportunistic Networks [52], a device with information to transmit, carries it within its cache, until a communication opportunity arises, i.e. a wireless link with the device of another person can be established. At that point, the information is transferred. This paradigm also applies for the service discovery problem. In this case, nodes exchange service advertisements and service queries opportunistically, as soon as they meet other nodes. If a device does not find an answer to a query, it stores the query in a local buffer and waits until the next encounter.

Figure 4.5 shows our reference scenario both form the human and node perspective. The human perspective represents the real social interactions happening among people, while the node perspective represents the communication opportunities among devices carried by people. Both of the perspectives change along the

time.

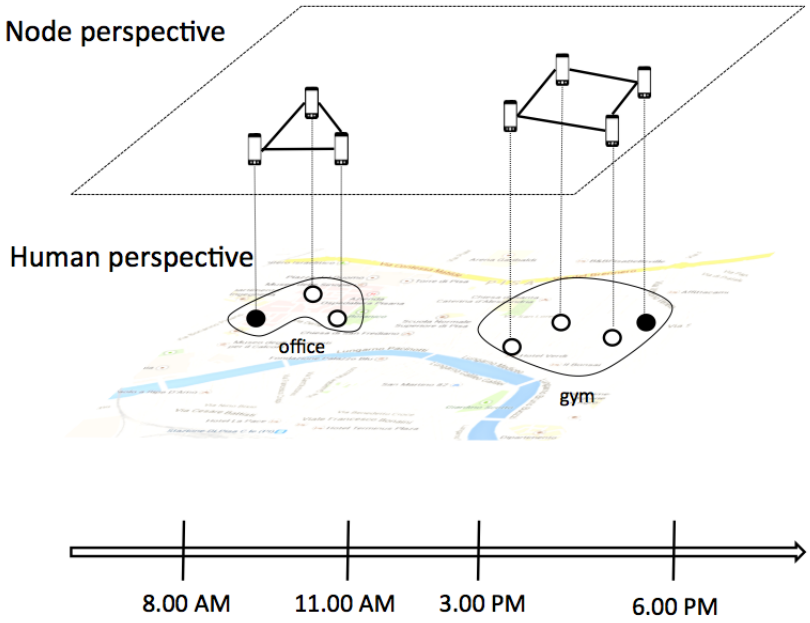


Figure 4.1: Social interactions during a day.

4.2 Service Discovery Model for MSN

4.2.1 Mobile Social Networks and Community Detection

A MSN is seen as a set of mobile nodes, each representing an individual moving within a bounded region. Nodes move driven by one or more objectives, e.g., traveling to the office, going back home or meet friends out. Occasionally, a node establishes a contact with another node by using a wireless communication interface (e.g., Bluetooth or WiFi). We model the connections among the nodes at time t as a directed graph $G_t = (V, E_t)$, where $V = \{n_1, n_2, \ldots, n_v\}$ is the set of the v nodes of the MSN and $E_t = \{e_{ij} = (n_i, n_j) : n_i, n_j \in V\}$ is the set of links among the nodes at time t. Links in E_t can be directed, indicating one-way communications, typical of opportunistic networks. The neighborhood N_t^i of node n_i at time t is the set of nodes $n_j \in V$ such that $e_{ji} \in E_t$. In other words, the neighborhood of node n_i is made up of all nodes that can communicate with n_i at time t.

As a node n_i moves and comes into contact with other nodes, it keeps a contact history for every node it connects with. For each node n_j the contact history is given by the following parameters:

- the cumulative contact time $t_{cum}(n_i, n_j)$, i.e., the total time nodes n_j was in contact with node n_i;

- the average contact time $ct(n_i, n_j)$, defined as the average duration of a contact between n_j and n_i;
- the inter-contact time $ict(n_i, n_j)$, which is the average time between successive contacts between n_j and n_i (as defined in [114]);
- the last-time seen $lt(n_i, n_j)$, which is the last time node n_i entered in contact with n_j.

Figure 4.2 depicts the times when node n_i hears from two other nodes, namely node n_2 and node n_3.

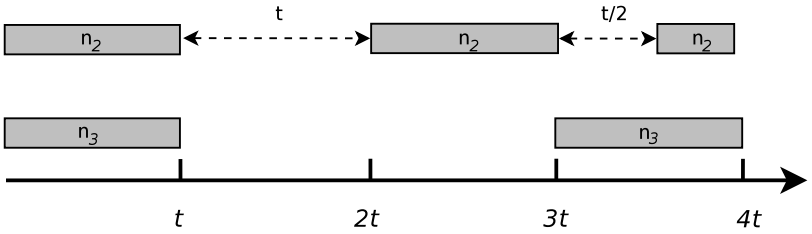


Figure 4.2: Contact times of nodes n_2 and n_3 with node n_i.

Based on these encounters, the contact history of node n_2 as stored by node n_i at time $4t$ is: $t_{cum}(n_1, n_2) = \frac{5}{2}t$, $ct(n_1, n_2) = \frac{5}{6}t$, $ict(n_1, n_2) = \frac{3}{4}t$ and $lt(n_2, n_3) = 4t$. For node n_3 node n_i would store $\langle 2t, t, 2t, 4t\rangle$. The contact history of all nodes is stored by node n_i in a table T^i that for each node $n_j \neq n_i$ lists:
$\langle t(n_i, n_j), ct(n_i, n_j), ict(n_i, n_j), lt(n_i, n_j)\rangle$.

The contact history is used to detect communities (refer to section 2.2.1). More precisely, in order to determine the community C where it resides, node n_i executes at time t a community detection algorithm A by using the contact history T_t^i and the neighborhood N_t^i of node n_i at time t as input. The community C is a subset of node n_i current in the neighborhood N_t^i. The selection of which neighbors are part of the community of node n_i depends on algorithm A. Our discovery algorithms are independent of the specific algorithm used to detect communities, and can use any of the detection algorithms proposed for MSN [60, 61].

Node n_i keeps track of the communities it has been a part of in a table $CT^i = \{C_1, \ldots, C_w\}$. Such table contains all the communities visited in time. Every time a community C is detected (by running algorithm A) it is inserted in CT^i only if there is no other community C' in CT^i that is *similar* to C. To determine similarity among communities C_1, C_2, we use the Jaccard index [115] as commonly done by community detection algorithms [59];

$$\gamma = \frac{|C_1 \cap C_2|}{|C_1 \cup C_2|}, \tag{4.1}$$

Observe that $\gamma \in [0, 1]$; if $\gamma = 1$ then the intersection among community members coincides with the union of C and C', and hence C and C' are identical. If $\gamma \geq \tau$

then C and C' are similar in the sense that they share a certain number of nodes. In both cases ($\gamma \geq \tau, \gamma = 1$) node n_i already visited C and it does not need to store C in CT^i again. Differently, if $\gamma < \tau$ then community C is new and n_i has to store C in CT^i .

Every node $n_i \in V$ is characterized by a profile that expresses its interests $\mathcal{I}_i \in \mathcal{I}$, and $\mathcal{I} = \{t_1, \ldots, t_n\}$ is the set of all possible interests. The interests are labels tagging a node in terms of a common classification. For example a node might be interested in: *news*, *forecast*, *national football team* service.

4.2.2 Service Discovery

Service discovery is a process composed by four steps, namely advertisement, query, selection and access as shown in Figure 4.3. Each phase requires a different type of message as explained in the follow.

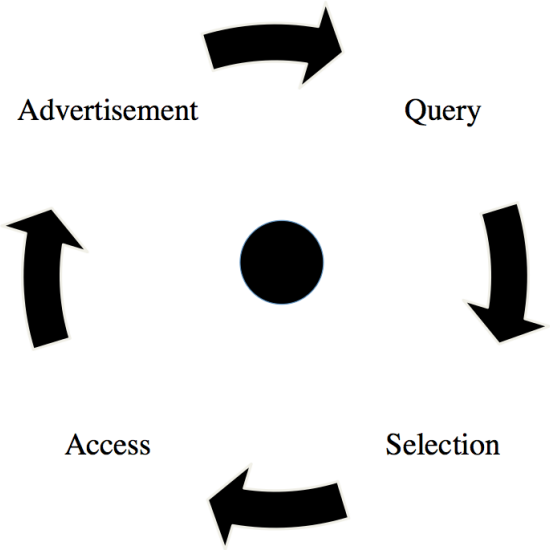


Figure 4.3: Service discovery process.

Advertisement

Nodes advertise the services they provide and the service advertisements they are aware of. A service advertisement is a compact data structure describing the most important features of the service, i.e. the functionalities it provides, the non-functional characteristics of the service[1], the identifier of the service provider. We assume that the advertisement adv_j comprises at least the following fields:

[1]The actual formalism used to describe service advertisements is out of the scope of this thesis and we will not discuss it.

- Advertisement interests: $\mathcal{I}_{adv_j} \subseteq \mathcal{I}$, which is the set of interests associated to the advertisement adv_j. Similarly to the node's interests, the interests $\mathcal{I}_{adv_j}$ tag the advertisement with respect to a common classification. For example the interests associated to a service for sharing media contents might be: *media*, *entertainment*, *social* etc.

- Service provider: $sp_k, k \neq j$ identifies the node that provides the service described with the advertisement adv_j. We consider that the service provider sp_k can provide more than one service.

The node n_i stores the advertisements received along the time in the node cache $\mathcal{A}_i = \{adv_1 \cdots adv_m\}$.

Query

The query $q \in Q$ is craft by the node n_i to discover the advertisements matching with it. We model a query as a message containing a list of features that specify the type of advertisement required by n_i. Given the query q, n_i checks if its cache contains at least one advertisement matching with q, this operation is done with the $f : Q \longrightarrow \mathcal{A}_i$ function. Such function returns the set of advertisements stored in $\mathcal{A}_i$ whose interests match with the query interests. The similarity among query interests and advertisement interests is determined with the γ similarity index, as also done with the communities (see Section 4.2.1). We model the query q as a message containing the following fields:

- Query interests $\mathcal{I}_q \subseteq \mathcal{I}$: the set of interests describing the kind of service required by n_i;

- Service requester n_r: the identifier of the node that craft the query;

The set of nodes to which a client forwards the query q is named the forwarding set (FS). After the submission of a query, the client can receive zero or more advertisements matching with the query. Such advertisements are stored in the node's cache with the *Update* function. If the node n_i does not receive any answer for a query, then it marks the query as a *pending query* and it stores it in the pending query set PQ_i.

Service selection and access

The last two steps of the service discovery process are the selection and the access phases (refer to section 2.2.2 and 2.2.2). The goal of the selection phase is to select the best-matching service advertisement according to an objective function. As an example, the goal of the client might be selecting the advertisement whose node provider minimizes the response time or selecting the provider that applies the lowest

charge to the service access. Differently, the goal of the access phase is to discover the best route in order to access the provider of the service previously selected.

SIDEMAN and CORDIAL focus on the first two steps of the service discovery, namely advertisement and query since they represent a complex problem for the reference scenario to which we refer to. Moreover, we consider that the selection phase might be implemented with well-known method for optimizing an objective function (e.g. best-fit or local greedy heuristics) and that the access phase can be delegated to out-of-band protocols.

4.3 SIDEMAN Algorithm

4.3.1 Overview of the algorithm

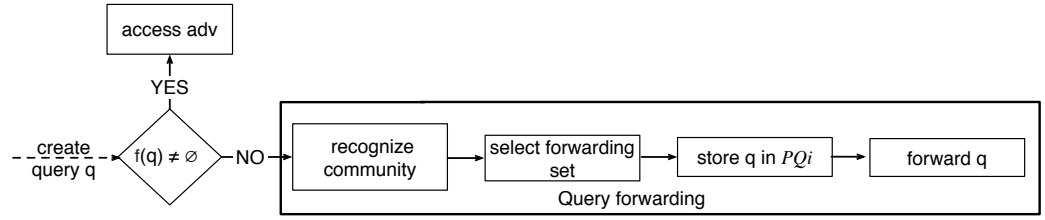
SIDEMAN [30, 33] is an algorithm used by a node n_i for discovering service advertisements available in a MSN. To this purpose SIDEMAN enables nodes *(i)* to reactively disseminate queries for advertisements they do not currently have, and *(ii)* to proactively exchange advertisements they might be interested in, so that a node has that advertisement when it needs it and *(iii)* to manage the reception of queries and advertisements (the discovery messages). The design of SIDEMAN exploits a typical feature of MSN: people tend to form communities made of similar individuals (Section 2.2.1). In particular, SIDEMAN relies on the detection of communities for optimizing operations (i) and (ii) as follows:

- the dissemination of queries is realized only among members of the same community, thus avoiding their indiscriminate flooding through the whole neighborhood;
- the exchange of services is performed only inside a community. The rationale of this choice is that since a community is often formed by similar individuals, it is more likely to find services of interest inside the community.

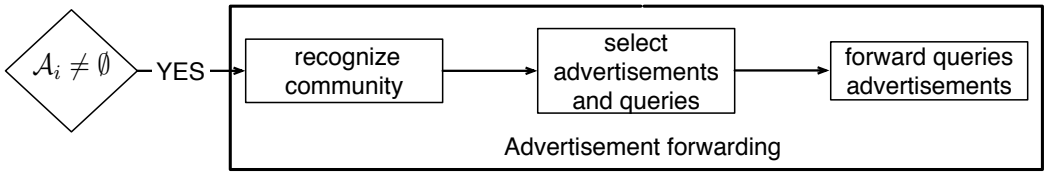
An overview of the three phases of SIDEMAN is shown in Figure 4.4.

Whenever a node needs a service (reactive phase of SIDEMAN), it crafts the query q and it checks if its cache stores an advertisement matching with q (Figure 4.4a). If the node finds an advertisement matching with q ($f(q) \neq \emptyset$) then it accesses the service provider specified in the advertisement. Otherwise, n_i runs an algorithm for detecting the community to which it belongs. Once a community is detected, n_i checks for a similar community in its community table. If such a community exists, i.e., if n_i has already visited that community in the past, n_i recognizes it. Otherwise, n_i stores the new community in its community table.

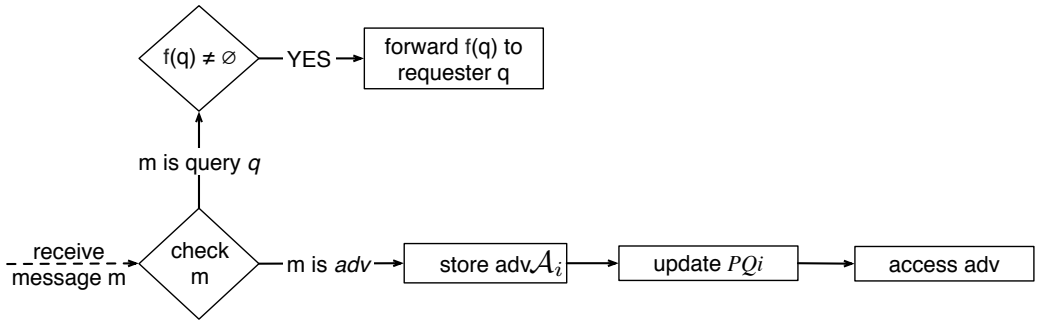
After the community detection phase, n_i builds a forwarding set from members of its current community whose interests match those in q. In other words, the query q is forwarded only to the members of the community of n_i sharing at least



(a) Reactive phase



(b) Proactive phase



(c) Message Reception

Figure 4.4: Overview of SIDEMAN.

one interest with q. In this way, the node n_i prevents from sending a query to a node whose interests are completely different with respect to the query q. At this point, node n_i stores the query q in the set of its pending query, such set represents the queries that are still waiting for an answer.

At regular intervals, node n_i executes the proactive phase of SIDEMAN, for exchanging services with members of its community (Figure 4.4b). If n_i carries an advertisement whose interests match those of at least one member of its community, then n_i exchanges such advertisement. During this phase, node n_i also forwards the queries left in PQ_i to the members of the community previously detected.

Figure 4.4c depicts the situation in which node n_i receives a discovery message (either query or an advertisement) from another node. If the message received is a query q, node n_i checks if it can provide an answer to q. If this is the case, node n_i replies to the requester node with the set of advertisements matching q. Otherwise, if the message received is the advertisement adv_j, then n_i stores adv_j in $\mathcal{A}_i$ and removes all the pending queries now answered from the advertisement adv_j.

4.3.2 SIDEMAN

Before describing the reactive and proactive phases of SIDEMAN (Figure 4.4a and 4.4b), the following algorithm describes how n_i recognizes the community C.

Algorithm 1 RecognizeCommunity(T, N, CT, τ).

1: $C = A(T, N)$
2: **if** $\exists\ C' \in CT \mid \gamma(C, C') \geq \tau$ **then**
3: $\quad \mathcal{I}_C =$ interests of C' from CT
4: **else**
5: $\quad \mathcal{I}_C = getInterests(C)$
6: $\quad CT = CT \cup C$
7: return $< C, \mathcal{I}_C >$

A community C is determined by running any community detection algorithm A [60] (line 1). A community C is *recognized* if there is a community C' in CT^i that is *similar* to C according to a given similarity index. As mentioned, we used the Jaccard index with threshold τ [115]. If a community C' is found that is Jaccard-similar to C (i.e., $\gamma(C, C') \geq \tau$) then the set of interests $\mathcal{I}_C$ of community C are those of community C' (line 3). If no such a community exists, the interests of the members of the community must be collected by asking to every community member its interests. This set of communication is performed through executing the function $getInterests(\cdot)$ (line 5). In particular the $getInterests(\cdot)$ function iterates over the members of C and asks to every member the list of its interests. The recognized community C is finally returned (line 7).

We stipulate that Algorithm 1 returns the pair $< C, \mathcal{I}_C >$, where C is the community (the set members of the community) and $\mathcal{I}_C$ is the list of interests shared by the members of the community. We recall that the community table CT^i is used for storing the communities visited by n_i and the interests of the community members. Through this table we can remarkably reduce the number of times the function $getInterests(\cdot)$ is invoked by n_i. Our experiments show that communities are recognized always over 50% of the times, and that, at steady state, the function $getInterests(\cdot)$ is called in about half of the cases.

The reactive phase of SIDEMAN is implemented by Algorithm 2.

Node n_i starts by checking whether it has an advertisement matching the query q (line 1). The f function verifies if node n_i stores in its cache a service adv_j matching q. In the positive, n_i accesses it. Otherwise, n_i recognizes the local community to which it belongs (Algorithm 1). Node n_i then computes the set $V_q \subseteq C$ of nodes that can potentially answer to query q (line 6). Note that the set V_q is composed by nodes in C that share at least one interest with q. Finally, the query is forwarded to nodes in V_q (line 7). We observe that the operation of creating the set V_q (based on checking that the interests in q also belong to the interests $\mathcal{I}_j$ of node n_j) can be expensive (in time, energy, etc.). In order to optimize it, we implemented the sets $\mathcal{I}_j$ as Bloom

Algorithm 2 Reactive phase of SIDEMAN.

1: **if** $f(q) \neq \emptyset$ **then**
2: $adv_j = f(q)$
3: access adv_j
4: **else**
5: $< C, \mathcal{I}_C >=$ RecognizeCommunity(T^i, N^i, CT^i, τ)
6: $V_q = \{n_j \in C \mid \exists\ t_k \in \mathcal{I}_q$ s.t. $t_k \in I_j\}$
7: Forward q to V_q

Filters (BFs) [116]. A Bloom Filter is a data structure implementing the following two operations: (i) checking whether or not an element belongs to the data structure (membership), and (ii) adding an element to the data structure (addition). Both operations are performed in constant time thanks to use of h distinct hash functions. We used Bloom Filters for implementing the set $\mathcal{I}_c$ of interests assigned to node n_j, so that checking if the interest t_i in $\mathcal{I}$ belongs or not to $\mathcal{I}_j$ is performed efficiently.

The proactive phase of SIDEMAN that takes care of advertisement exchange and forward of pending queries is performed at node n_i by executing the following Algorithm 3.

Algorithm 3 Proactive phase of SIDEMAN.

1: $< C, \mathcal{I}_C >=$ RecognizeCommunity(T^i, N^i, CT^i, τ)
2: **for all** $t \in \mathcal{I}_C$ **do**
3: $V_t = \{n_j \in C \mid t \in \mathcal{I}_j\}$
4: $S_t = \{adv_j \in \mathcal{A}_i \mid t \in SI_j\}$
5: **if** $V_t = \emptyset$ **then**
6: $\mathcal{I}_C = \mathcal{I}_C \setminus \{t\}$
7: Forward S_t to V_t
8: **for all** $q \in PQ_i$ **do**
9: **if** $t \in q$ **then**
10: Forward q to V_t

Node n_i starts with recognizing its community. To this purpose it uses Algorithm 1 described above that returns C as well as $\mathcal{I}_C$.

For every interest $t \in \mathcal{I}_C$ node n_i performs the following actions.

1. It computes the set $V_t \subseteq C$ of nodes in C also interested in t and the set $S_t \subseteq \mathcal{A}_i$ of advertisements matching t (lines 3 and 4). If V_t is empty, n_i removes t from $\mathcal{I}_C$ (line 6). In this way, if the same community is recognized at a later time, a node can avoid considering interests that are no longer shared by the members of C.

2. It forwards the set S_t to nodes in its community sharing the same interests (line 7).

3. It selects those pending queries from PQ_i that concern interest t, and forwards them to members of its community sharing that interest (line 10).

As for Algorithm 2, the constructions of sets V_t and S_t in Algorithm 3 are implemented through Bloom Filters. In this way, the complexity of computing V_t and S_t is $O(|C|)$ and $O(|\mathcal{A}_i|)$, respectively.

Whether reactively sending out service queries (Algorithm 2) or proactively exchanging advertisements and disseminating pending queries (Algorithm 3), some of the nodes in the neighborhood of n_i might transmit responses. The following Algorithm 4 describes node n_i reaction to receiving response m.

Algorithm 4 OnMessageReception(m).

```
1: if m is a query then
2:     n_r is the requester of q
3:     D = f(m)
4:     Forward D to r
5: else
6:     Update(A_i, m)
```

Upon receiving response m, node n_i checks whether it is a query or an advertisement. If m is a query then the node determines all its advertisements that answer that query, if any. These advertisements are then sent directly (i.e., through a unicast transmission) to the node n_r requesting the query m. If m is instead an advertisement, node n_i updates its service cache $\mathcal{A}_i$. Specifically, the function Update($\mathcal{A}_i$, m) checks whether m is already in $\mathcal{A}_i$. If this is the case, the corresponding entry is updated. Otherwise, m is added to the cache.

4.4 CORDIAL Algorithm

This section describes the CORDIAL algorithm. Compared to SIDEMAN, CORDIAL differs with respect to two points:

- an extended application scenario;
- the improvement of the forwarding policy of queries and advertisements.

Concerning the first point, CORDIAL is designed by keeping in mind scenarios in which people move outdoor in large urban or rural areas (see Section 4.1). For example we consider the mobility of a group of persons roaming in a city or a group of students roaming in a university campus for long period time. Under this

respect CORDIAL better exploits than SIDEMAN some mobility features in order to optimize the diffusion of queries and advertisements.

For what concerns the second point, CORDIAL implements different policies for forwarding queries and advertisements. The policy of SIDEMAN for forwarding discovery messages is based on the similarity of the interests between a node and the query and on the similarity between a node and the advertisement to forward. CORDIAL splits these in two cases. The policy for forwarding queries is based on the idea of selecting those nodes from the community that are similar in terms of interests and that are encountered periodically. In fact, nodes that meet periodically have higher changes to exchange a query and, later in time, to exchange an advertisement matching with that query. Concerning the policy for forwarding advertisements, CORDIAL aims at exchanging the advertisements to nodes that are potentially interested in the advertisement and that previously in time submitted a query matching with the advertisement. Section 4.4.1 describes these two observations. Finally, CORDIAL avoids to forward queries and advertisement to nodes that already received them in the past, with the benefit of reducing energy consumption of the devices, the overhead of the protocol and the computational resources of devices.

4.4.1 Routines in MSN

The scenario described in Section 4.1 does not consider a typical behavior of the human mobility, namely the *routines*. As previously discussed, the human mobility follows three basic rules: common activities, restricted number of locations and short paths. Moreover, people often adopt a routinary pattern along the day as discussed in [117]. Intuitively humans tend to visit periodically the same places (e.g. home, office, pub) and hence to join periodically the same communities.

Figure 4.5 shows a simple example. Alice (depicted as a full black circle) joins three communities that are visited at different hours along day. Throughout the day, Alice joins the Office community (from about 8AM to about 5PM), pub community (after hours) and the home community (night hours). Within each community, Alice meets colleagues and friends (indicated by hollow circles). This simple routine repeats for the whole week (e.g Monday to Friday).

Routines can be exploited for the service discovery problem when it is needed to forward either a query and an advertisement message. In particular, the query forwarding strategy aims at disseminating a query to other nodes with the goal of receiving in short time an advertisement matching with it. Such strategy can be extended by considering a measure of how often two nodes enter in contact. Nodes that meet periodically have higher changes to exchange a query and, later in time, to exchange an advertisement matching with that query. Hence the strategy we seek is to give more priority to those nodes visited more frequently and routinary along the time.

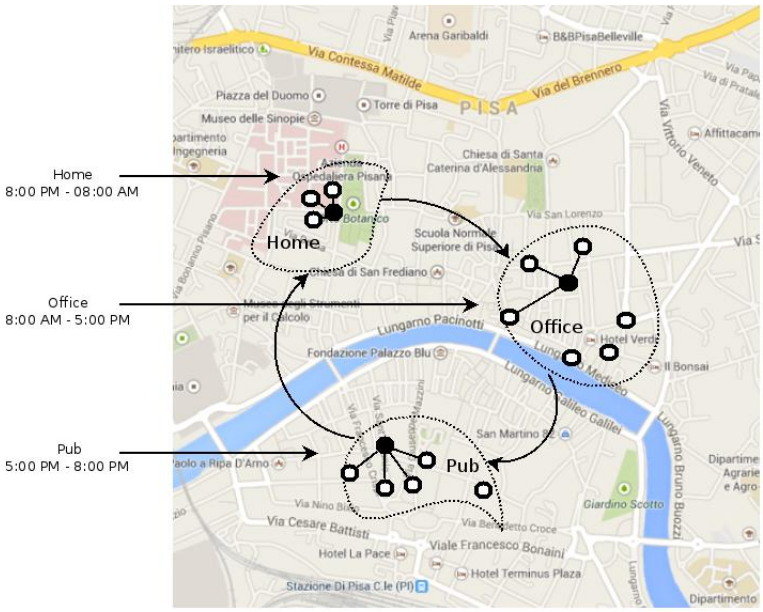


Figure 4.5: Example of routine in a day.

Similarly, the advertisement forwarding strategy aims at exchanging the advertisement adv_j to nodes that:

- are potentially interested in adv_j;
- that previously in time submitted a query matching with adv_j.

If node n_i carries an advertisement adv matching with a query submitted by node n_k, then n_i has two choices: (1) if n_k is in contact with n_i, then n_i delivers adv to n_k, (2) if n_k is not in contact with n_i, then n_i selects the node n_w that will encounter n_k more rapidly than n_i.

Figure 4.6 shows how routines can improve the forwarding of queries and advertisements. In the figure, the node n_i joins the community C_i composed by one single node, namely n_j. Every node adopts the similarity function γ that, given two set of interests, it measures the similarity degree. The node n_j does not share any interest with the query q craft by n_i ($\gamma(\mathcal{I}_q, \mathcal{I}_j) = 0$), however n_j joins the community C_j composed by nodes n_k, n_h. The interest sets of n_k, n_h are γ-similar with q, in particular $\gamma(\mathcal{I}_q, \mathcal{I}_k) = 1$ (perfect match) and $\gamma(\mathcal{I}_q, \mathcal{I}_h) = 0.5$ (partial match). Even if $\gamma(\mathcal{I}_q, \mathcal{I}_j) = 0$, n_i decides to still forward q to n_j since the goal of n_i is to reach the C_j's members.

CORDIAL exploits the strategy previously described in order to optimize the forwarding of queries and advertisements. With such strategy, a node may receive in time *direct* and *indirect* responses, as shown in Figure 4.7.

Figure 4.7a shows the direct response mechanism. In all the tree cases a, b and c, the node receiving the query, namely n_j is the same that returns the advertisement

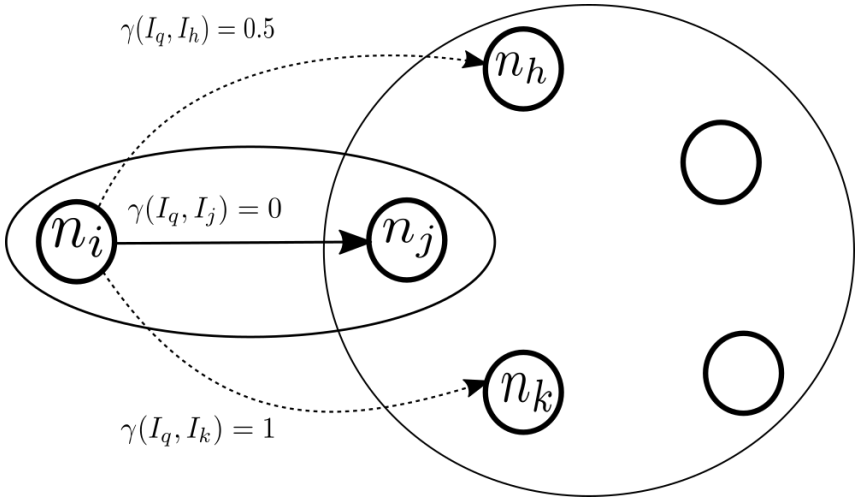
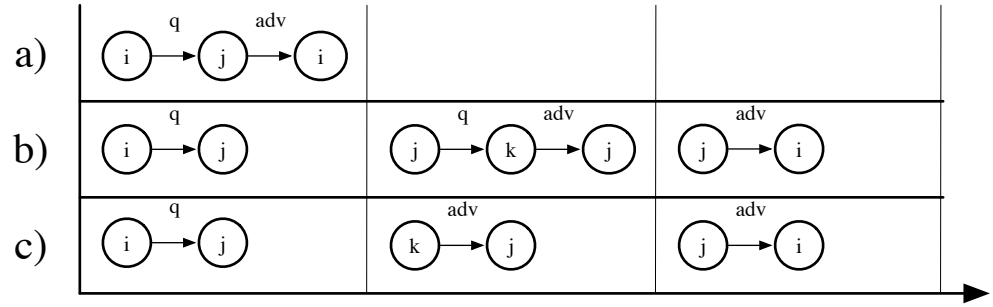
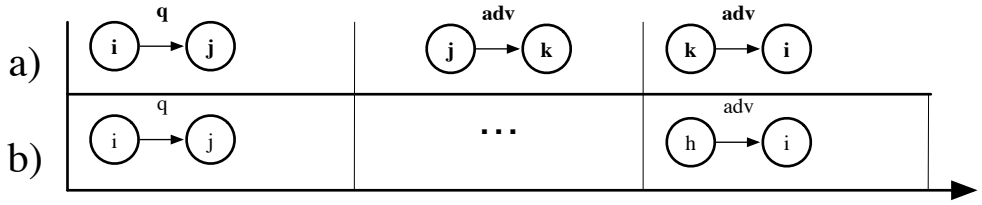


Figure 4.6: Similarity cross-communities.



(a) Direct responses.



(b) Indirect responses.

Figure 4.7: Direct and indirect interaction schema.

back to n_i. The cases *a,b* and *c* show how n_j discovers the advertisement *adv* matching with q:

(a) n_i forwards the query q to n_j and n_j answers immediately with an advertisement matching with q. This is the optimal case, because n_j already carries an

advertisement matching with q;

(b) n_i forwards the query q to n_j, later in time n_j takes care of forwarding q on behalf of n_i to the node n_k. The node n_k carries an advertisement matching with q that it sends to n_j. At the end of the process, n_j encounters n_i again and n_j sends the advertisement back to n_i;

(c) n_i forwards the query q to n_j, later in time n_j receives proactively an advertisement matching with q from n_k. At the end of the process, n_j encounters n_i again and n_j sends the answer to n_i.

Figure 4.7b shows the indirect response mechanism. In this case the node receiving the query, namely n_j, is different with respect to the node that sends the advertisement adv matching with q to the node n_i. We define two cases:

(a) n_i forwards the query q to n_j, n_j carries an advertisement matching with q and it forwards the advertisement to n_h because n_h will encounter n_i more rapidly with respect to n_j;

(b) n_i forwards the query q to n_j, later in time n_i receives proactively an advertisement adv from node n_h. In this last case, n_h did not receive q from n_j, rather n_k advertises n_i proactively.

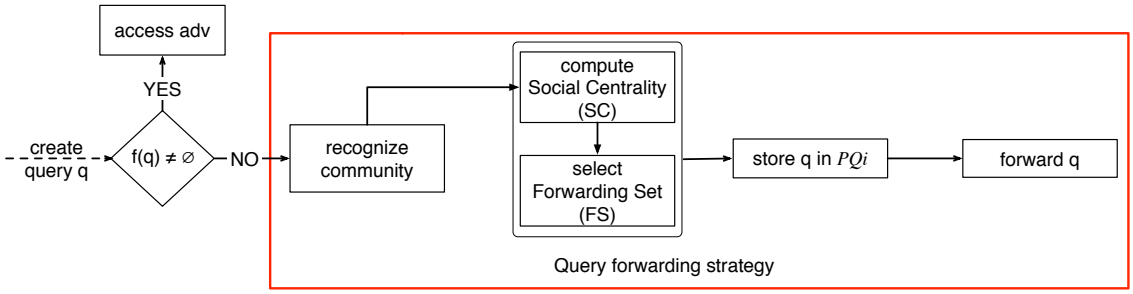
4.4.2 Overview of the algorithm

CORDIAL is the algorithm used by a node n_i for discovering and advertising services available in MSN. To this purpose, CORDIAL enables nodes: *(i)* to reactively disseminate queries for service advertisements they are looking for, and *(ii)* to proactively exchange service advertisements with nodes that might be interested in such advertisements and *(iii)* to manage the reception of queries and advertisements. The reactive and proactive phases of CORDIAL are depicted in Figure 4.8.

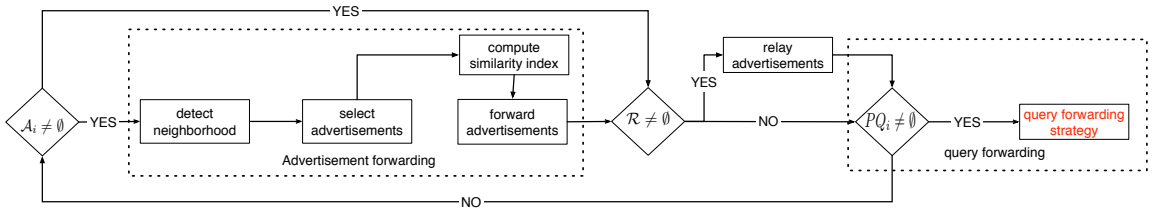
Reactive phase

Whenever the node n_i needs a service, it executes the reactive phase of CORDIAL, see Figure 4.8a. The reactive phase first creates the query q needed by n_i. Then, it executes the f function in order to look up for any advertisement stored in the service cache $\mathcal{A}_i$ of n_i and matching with the query q. If at least one advertisement is found, then n_i accesses to the service provider specified in the advertisement, otherwise it executes the query forwarding strategy (in Figure 4.8a it is denoted with the red box). The query forwarding strategy first recognizes the community of n_i and then it applies the following three rules for selecting the nodes to which forward the query q:

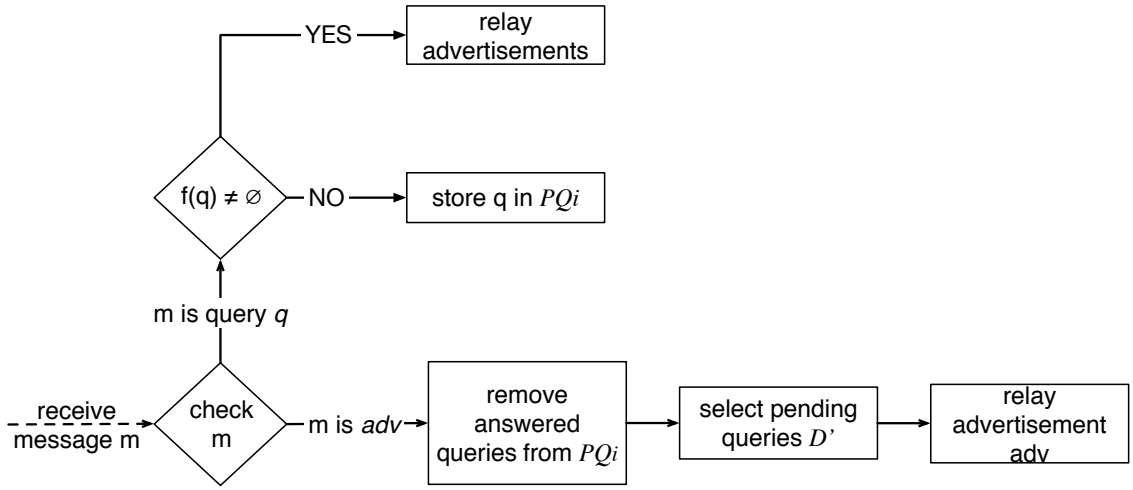
- selection of nodes in the community of n_i whose interests match with the query;



(a) Reactive phase.



(b) Proactive phase.



(c) Message reception.

Figure 4.8: Overview of CORDIAL.

- selection of nodes in the community of n_i whose friends might answer to the query (Social Centrality rule);

- avoidance unnecessary forward of queries.

After the selection of the candidate nodes, n_i store q in the pending query set and it forwards q to the nodes previously chosen.

Proactive phase

The proactive phase is executed at fixed intervals, it is shown in Figure 4.8b. Its goal is to disseminate proactively queries and advertisements carried by n_i to nodes potentially interested in such discovery messages. The proactive phase distinguishes how to forward advertisements from how to forward the pending queries (such steps are shown in Figure 4.8b with two dotted boxes):

The node n_i first checks if its service cache is empty ($\mathcal{A}_i \neq \emptyset$). In the negative case it executes the advertisement forwarding policy whose goal is to disseminate proactively those advertisements carried by n_i to nodes that might be interested in using them. In order to maximize such goal, the proactive phase detects the neighborhood of n_i and it applies the following two rules:

- selection of nodes from the neighborhood of n_i that did not receive the same advertisements in the past;
- selection of nodes from the neighborhood of n_i whose interests match with the advertisement's ones (by means of the similarity index γ).

After the selection of the candidate nodes, n_i forwards the advertisement to the nodes previously chosen. Moreover, n_i acts as relay node for the advertisements that must be forwarded to a specific node, we call this node the final destination of the advertisement. These advertisements are stored in the forwarding cache $\mathcal{R}_i$, a data structure containing advertisements to be forwarded only the final destination. If $\mathcal{R}_i$ is not empty then n_i check if it is in contact with the final destination and (in the positive case) it delivers the advertisement to the final destination, otherwise n_i identifies the nodes in the neighborhood that can deliver the advertisement *quickly* to its final destination.

Finally, the proactive phase concludes with the forwarding of pending queries stored in the set PQ_i. The goal of this step is to forward those queries stored in the pending query set PQ_i that are not jet answered. Queries in PQ_i can be generated either form n_i or to any other node n_j that forwarded q to n_i during the reactive phase (see Figure 4.8a). For every query $q \in PQ_i$, n_i adopts the same strategy used in the reactive phase, in particular n_i first builds the forwarding set and then n_i forwards q to such set.

Message reception

Node n_i can receive asynchronously a message from another node. If the message received is a query q, then n_i executes the function $f(q)$ in order to check if it carries an advertisement matching with q. If this is the case, n_i acts as relay node in order to forward the advertisements matching with q toward its final destination (relay advertisements). Otherwise, n_i stores the query in the pending query set PQ_i waiting for a matching advertisement. If the message m received is the advertisement adv, n_i first removes all the pending queries now answered from adv. Then, n_i selects

the queries received from other nodes that are now answered from adv, namely the the set D'. After this step, n_i checks if the requester of the queries in D' is in contact with n_i, if this is the case n_i delivers the advertisement to the final destination otherwise n_i selects one of the nodes to which it is currently in contact with and that can *quickly* deliver the advertisement to the final destination.

4.4.3 CORDIAL

The reactive phase of CORDIAL is shown in Algorithm 5.

Algorithm 5 Reactive phase of CORDIAL.

1: **if** $f(q) \neq \emptyset$ **then**
2: access adv_j
3: **else**
4: $C = \text{RecognizeCommunity}(T^i, N^i, CT^i, \tau)$
5: $FS = \{SC_j, \forall n_j \in C | q \notin PQ_j\}$
6: forward q to top k nodes in FS

Node n_i starts by checking whether it has an advertisement matching the query q (line 1). The f function verifies if node n_i stores in its cache a service adv_j matching q. In the positive, n_i accesses adv_j. Otherwise, the node n_i executes the community detection algorithm *RecognizeCommunity* as described in Algorithm 1. Given the community C_i (line 4 Algorithm 5), the node n_i builds the forwarding set FS. In order to build FS, n_i first assigns to every node $n_j \in C_i$ its Social Centrality SC score, the SC metric is given by:

$$SC_j = \frac{\gamma(\mathcal{I}_q, \mathcal{I}_j)}{1 + ict(i,j)} + \sum_{\substack{w \in C_j - C_i \\ ict(i,w) > ict(j,w)}} \frac{\gamma(\mathcal{I}_q, \mathcal{I}_w)}{1 + ict(j,w)} \tag{4.2}$$

The SC metric in 4.2 measures the capability of n_j of answering to the query q even with a direct or with an indirect response (see Figure 4.7). In particular the first part of 4.2 measures the direct responses, it assigns to $n_j \in C$ a score that is the ratio between the similarity of the interests between the query q and the interests of n_j with respect to the inter-contact time between n_i and n_j. The first part of 4.2 is high when the similarity index γ is high (n_j has many interests matching with q) and when the inter-contact time between n_i and n_j is low (n_i encounters n_j frequently). The second part of 4.2 is computed as follows. We consider the set difference between the communities C_j and C_i and we apply the constraint that the inter-contact time between $w \in \{C_j - C_i\}$ and n_i is higher than that the inter-contact time between $w \in \{C_j - C_i\}$ and n_j. In this way we consider the nodes that belong to C_j only, and such that n_j visits them more frequently with respect to n_i. With the previous constraint we prevent from delivering to n_j a query directed to

a node that n_i will encounter more rapidly with respect to n_j. The second part of 4.2 is the ratio between the similarity of the interests of query q and the interests of every node $w \in \{C_j - C_i\}$ with respect to the inter-contact time between n_j and w. This last part measures the capability of node n_j to enter in contact with nodes that might answer the query q. The node n_i assigns the SC score to every node $n_j \in C_i$ such that it did not receive the same query in the past ($q \notin PQ_j$, see line 5 algorithm 5). This technique implements the avoidance of query duplication. Then n_i selects the top k nodes with the highest SC and n_i forwards q to them. CORDIAL

The proactive phase of CORDIAL is described with Algorithm 6.

Algorithm 6 Proactive phase of CORDIAL.

1: **for all** $adv \in \mathcal{A}_i$ **do**
2: $V_t = \{n_j \in N_t^i \mid \gamma(\mathcal{I}_j, \mathcal{I}_{adv}) > \tau \wedge adv \notin \mathcal{A}_j\}$
3: Forward adv to V_t
4: **for all** $adv \in \mathcal{R}_i$ **do**
5: **if** final destination of adv is in contact **then**
6: Forward adv to final destination
7: **else**
8: $n_k = TemporalForwarder(adv)$
9: Forward adv to n_k
10: **for all** $q \in PQ_i$ **do**
11: $C = \text{RecognizeCommunity}(T^i, N^i, CT^i, \tau)$
12: $FS = \{SC_j, \forall n_j \in C \mid q \notin PQ_j\}$
13: forward q to top k nodes in FS

The proactive phase implements the forwarding strategy for queries and advertisements. The first part of Algorithm 6 exchanges the advertisements stored in the cache $\mathcal{A}_i$ (lines $1-3$). For every adv stored in $\mathcal{A}_i$, n_i builds the set V_t composed by the nodes whose interests match with those of adv and such that they did not receive adv previously. Note that V_t is computed by considering the neighborhood and not the community. Similar to SIDEMAN, we used the Bloom Filters in order to optimize the creation of set V_t. In particular checking if $adv \in \mathcal{A}_j$ is executed in constant time, thus the complexity of determining V_t is $O(|N_t^i|)$.

Once V_t has been computed, n_i forwards adv to the set of recipients in V_t. After this first round of exchanges, the node n_i checks if some of the advertisements stored in the forwarding cache $\mathcal{R}_i$ can be delivered to the final destination (lines $4-9$). If $adv \in \mathcal{R}_i$ is directed to n_j and n_i is currently in contact with n_j, then n_i delivers the advertisement to n_j (direct answer). Otherwise n_i selects the relay node $n_w \in N_t^i$ with the lowest remaining inter-contact time to the final destination of adv, this is the node n_k. As discussed in [114], at time t the remaining inter-contact time $r - ict$ is (expected) the remaining time before nodes n_w and n_k meet again. We propose a

simple rule for computing the $r - ict$ between n_w, n_k:

$$r - ict(n_w, n_k) = |[t - lt(n_w, n_k)] - ict(n_w, n_k)| \tag{4.3}$$

Where t is the current time, lt it the last time n_w saw n_k and ict is the inter-contact time between n_w and n_k. The smaller the $r - ict$, the more likely n_w will meet n_k again, hence n_w will deliver adv to the final destination. The function $TemporalForwarder$ detects the node the lowest $r - ict$ value with respect the final destination of the advertisement adv. If such node exists, then n_i forwards adv to n_w. The last part of Algorithm 6 exchanges the pending queries carried by n_i (line 10-13). For every $q \in PQ_i$, node n_i applies the same mechanism described in Algorithm 5 lines 5-7.

After the forwarding of queries and advertisements, n_i might receive responses from other nodes. The following Algorithm 7 describes node n_i reaction to receiving a response m. Algorithm 7 distinguishes the reception of queries from the reception of advertisements. If m is a query (lines 1-10) then n_i executes $f(q)$ function to look up for any advertisement matching with q. If this is the case, then n_i checks if the node originating the query, namely the node originator n_r, is currently in contact. The node n_r is the node waiting for an answer for the query q. If n_r is in contact with n_i then n_i forwards the set of matching advertisements to n_r, otherwise n_i selects the nodes n_k with the lowest remaining inter-contact time (see Equation 4.3). If n_i does not carry any advertisement matching with q, the n_i stores $q \in PQ_i$. This last case implements the collaborative strategy adopted by CORDIAL: nodes take care not only of their queries, but also of queries submitted by other nodes.

If m is the advertisement adv (lines 11-30) then n_i checks all the pending queries $q \in PQ_i$ that are now answered from adv, such set is named D (lines 12). The set D is composed by the queries craft by n_i matching with the advertisement adv just received. D is removed from PQ_i, because D contains queries now answered. Moreover, n_i checks all pending queries carried on behalf of other nodes (queries whose requester is not n_i) that are answered by adv just received, such set is named D' (lines 14). For every query q in D', if the requester n_r of $q \in D'$ is in contact with n_i, then n_i forwards adv to n_r. If n_r is not in contact with n_i, then n_i selects the node n_k with the lowest remaining inter-contact time and it forwards adv to n_k. Note that n_k has been selected as relay node for the advertisement adv. After this round of checks, the node n_i verifies who is the final destination of adv. If the final destination is not n_i, then n_i received an advertisement addressed to another node. In this case, the node n_i has been selected as relay node. Node n_i forwards adv to the final destination of adv (if it is in contact), otherwise n_i forwards adv to the node with the lowest remaining inter-contact time. The last operation in Algorithm 6 stores the advertisement adv in the cache $\mathcal{A}_i$ only if the interests of adv match with the interests of n_i.

Algorithm 7 OnMessageReception(m).

1: **if** m is the query q **then**
2: n_r is the requester of q
3: **if** $f(q) \neq \emptyset$ **then**
4: **if** n_r is in contact **then**
5: Forward $f(q)$ to r
6: **else**
7: $n_k = TemporalForwarder(r)$
8: Forward $f(q)$ to n_k
9: **else**
10: store q in PQ_i
11: **if** m is the advertisement adv **then**
12: dst is the final destination of adv
13: $D = \{q \in PQ_i \mid requester(q) = n_i \wedge \gamma(\mathcal{I}_q, \mathcal{I}_{adv}) \geq \tau\}$
14: Remove D from PQ_i
15: $D' = \{q \in PQ_i \mid requester(q) \neq n_i \wedge \gamma(\mathcal{I}_q, \mathcal{I}_{adv}) \geq \tau\}$
16: **for all** $q \in D'$ **do**
17: n_r requester of q
18: **if** n_r is in contact **then**
19: Forward adv to r
20: **else**
21: $n_k = TemporalForwarder(r)$
22: Forward adv to n_k
23: **if** $dst \neq n_i$ **then**
24: **if** dst is in contact **then**
25: Forward adv to dst
26: **else**
27: $n_k = TemporalForwarder(dst)$
28: Forward adv to n_k
29: **if** $\gamma(\mathcal{I}_{adv}, \mathcal{I}_i) \geq \tau$ **then**
30: Update($\mathcal{A}_i, adv$)

Chapter 5

Evaluation of Service Discovery Algorithms

The goal of SIDEMAN and CORDIAL algorithms is to advertise services and propagate queries in MSN. We evaluate the performance of these algorithms via simulations, in order to reproduce Smart Environments that are closer to the reality. In order to achieve this, we first analyze properties concerning the mobility of humans in indoor and outdoor environments by real-world and synthetic mobility traces. This preliminary study enabled us to take design decisions concerning the advertisement and query-forwarding strategies implemented by SIDEMAN and CORDIAL. This chapter is organized as follows. We first present the co-location traces in Section 5.1. In Section 5.2 we describe various real-word mobility datasets available in the literature, as well as the HCMM mobility model, and we present an initial analysis of these datasets using a simple evaluation framework. In Section 5.4 we present the evaluation metrics for our service discovery. Finally Sections 5.5 and 5.6 report the results we obtained for both algorithms.

5.1 Human Mobility Traces

Understanding the human mobility first requires the reproduction of the mobility of people in real-world environments. There are basically two approaches for studying the human mobility: real-world datasets and/or synthetic datasets. Both of them are useful tools when the goal of the experiment is to reproduce a scenario where people move according to some criteria. The first choice we made is to adopt both real-world and synthetic datasets. We give particular emphasis to real world datasets for two main reasons: *(i)* real-world datasets represent the ground truth of human mobility, in fact they can be used to emulate the real mobility of a set of people for a time period and *(ii)* real-world datasets are commonly used in the research field of Mobile Social Computing and, more generally, in Opportunistic Computing hence we can compare our results against existing ones with the same scenario. During this

thesis, we also evaluate our discovery algorithms with a synthetic dataset obtained from a human mobility model, namely the HCMM model.

Real-world and synthetic mobility datasets are commonly available in the form of co-location traces. Co-location traces do not track the global or the relative position of the devices (for example by means of GPS coordinates), rather they track the start and end time of a contact between a device pair with a given time resolution. Typically, the datasets are obtained by using a simple application running on a smart phone that periodically scans the Bluetooth or WiFi network looking for other devices in proximity. If a device is found, then the application runs a simple protocol for double-checking the connectivity with such device and to confirm their co-location. The scan period of the application is a crucial parameter, indeed it regulates how often the application scans the network. Intuitively, the shorter the scan period the higher the resolution, but also the higher the battery depletion of the smart phone or the device running the application. Common scan periods range from $120s$, $150s$ to $300s$.

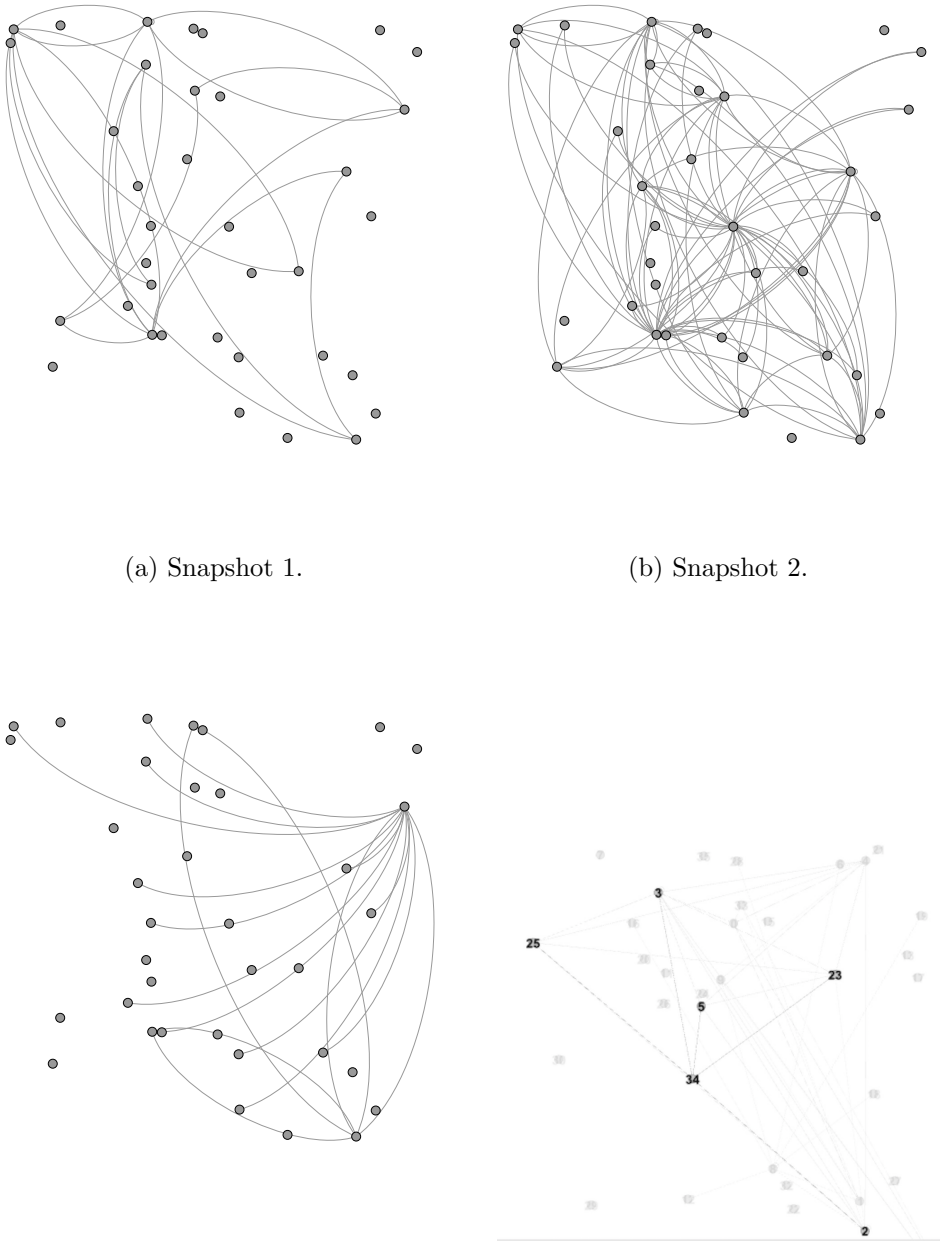
We present below a snapshot of the co-location trace of the Cambridge dataset, the syntax of every row is $[time, tag, node_i, node_j, event]$, in particular the time stamp of the log, a tag, the node pair and the type of event (*up* the node pair is connected, *down* the node pair is disconnected):

```
51679.00 CONN 34 23 up
51679.00 CONN 34 23 down
52218.00 CONN 14 9 up
52218.00 CONN 14 9 down
53517.00 CONN 28 2 up
53517.00 CONN 28 2 down
```

For example, at time 51679.00 device 34 can communicate with device 23, note that often the co-location traces are not symmetric hence we cannot assume that device 23 can communicate also with device 34. The co-location traces provide a compact representation of a dynamic encounter graph. It is a graph composed by devices in the dataset connecting and disconnecting along the time, the accuracy of the graph is given by the scan period used for collecting the co-location traces. Figure 5.1 shows a sequence of 4 snapshots of the encounter graph for the Cambridge dataset, in particular Figure 5.1d is the snapshot of the graph corresponding to the co-location trace previously shown.

5.2 Human Mobility with Experimental Datasets

During this thesis we use 4 real-world datasets, namely Infocom06 [118], Cambridge [118], MIT Reality [119] and the MDC Nokia dataset [120] as well as a synthetic dataset obtained from the HCMM mobility model [121]. In the following the provide a description of each dataset.



(a) Snapshot 1.

(b) Snapshot 2.

(c) Snapshot 3.

(d) Snapshot 4 at time 51679.00.

Figure 5.1: Snapshots of encounter graph with Cambridge dataset.

Infocom06

The Infocom06[1] dataset reproduces an indoor simulation scenario, where 78 conference attendants participate to the IEEE Infocom conference from April 24 to April

[1]Haggle project, http://www.haggleproject.org/

27 2006. The participants were given Intel iMote devices equipped with a Bluetooth transceiver with a transmission range of about 30m. Traces were collected each day from 7.00AM to 9.00PM. Each participant filled a survey about her/his interests, nationality, language spoken, etc (we use this survey to assign interests to the devices). The sampling scan period of the application running on the iMote is of 120s.

Cambridge

The Cambridge dataset reproduces the mobility of 36 students moving in the Cambridge University Campus for 12 days. The participants were given Intel iMote devices equipped with a Bluetooth transceiver with a transmission range of about 30m. The devices assigned to the students have been configured with a sampling scan period of 120s. Some extra devices have been statically placed in specific points of interest, we exclude them from our analysis since not relevant for studying the mobility of people involved in the experimentation.

MIT Reality

The MIT Reality dataset reproduces the mobility of 94 students moving in the MIT University Campus in Boston from 2004 to 2005. The participants were given an application running on the phone tracking several parameters such as Bluetooth contacts, SMS sent/received and phone calls. For the purpose of this thesis, we consider the co-location traces obtained from the Bluetooth contacts. The sampling scan period of the application running on the phone is of 300s.

MDC Nokia

The Mobile Data Challenge Nokia dataset has been collected from 2009 to March 2011 with 185 participants in the Lake Geneva region (Switzerland). Participants were mostly in the age of 22 to 35 years old. They were given an application designed for Nokia N95 phone, that periodically collected several information such as GPS position, Bluetooth sightings, places visited, SMS and phone calls as well as various sensor readings. All the previous information were reported with a scan period of 600s. This dataset does not provide the co-location trace, hence we extracted them by means of the the Bluetooth sightings, in particular we assumed that if a device is in contact with another device for at least 600s then they are co-located and they can communicate. For the purpose of this thesis we extracted a trace of the duration of 20 days in order to have comparable results.

Analysis of experimental datasets

We compare the experimental datasets with respect to several metrics commonly used when analyzing the human mobility in MSN. The metrics we select are indica-

tive both for the mobility of people and for the performance of algorithms designed for MSN such as service discovery algorithms. The goal of this section is not ranking the datasets according to some metrics, this because the datasets have been collected in different epoch, with different sensing technologies and for different purposes. Rather, our aim is to identify the most important features of each of the datasets and to summarize which social scenario the datasets best capture. We study the following metrics:

- Cardinality of neighborhood and communities. This metric measures the average number of devices found in proximity and the cardinality of communities locally detected by each device. The communities are detected by means of the DRAFT algorithm [61]. DRAFT is a distributed spatio-temporal detection algorithm. It requires to configure three parameters: the cumulative contact duration τ is used to decide to add or remove a device form the community of a device. The cumulative contact duration is measured at fixed intervals, the length of the interval is governed by the length t. Moreover, it implements a decay mechanism δ to prune devices that no longer belong to a community. The choice of these parameters affects the cardinality of the communities detected by each device, for our experiments we configured DRAFT with $\tau = 120, t = 3600, \delta = 0.9$ (as discussed in [61]).

- Number of contacts. This metric represents the distribution of the encounters of the users per hour. It gives an indication of the social activity of a user, providing information about when people meet and with how many people. Moreover, as discussed in [96] the number of contacts among pairs of individuals is a first indicator of the mobility pattern characterizing such pair. For example, students in contact daily from 9 to 11 AM are more likely students attending the same lecture, and they have a daily routinary mobility. Concerning the service discovery problem, the number of contacts measures the number of communication opportunities arising among devices carried by users and, in turn, the capability of diffusing queries as well as advertisements.

- Contact Duration. This metric measures the average duration of the encounters among people. The contact duration can be considered as a rough estimator of the familiarity among people. Generally, the more people spend time together, the more they are involved in a non-occasional social relationship. However, the nature of such relationships cannot be determined by only measuring the contact duration. In fact, people that meet for long periods could be relatives, friends or colleagues. Service discovery algorithms designed for MSN can take into account the distribution of the contact duration to take decisions about the diffusion of the discovery messages.

- Inter-contact Time (ict). This metric measures the time elapsed between two consecutive encounters among a pairs of users. It is a good indicator of the

frequency of the encounters and hence of the mobility of people. In datasets with short ict, users tend to often meet the same group of users. Service discovery algorithms designed for MSN that require delivering a message to specific target device, can exploit ict to take local decisions. For example, a routing strategy may deliver to device j a message to device i if the ict between j and i is short.

- The total number of encounters and the unique number of encounters. These metrics measure, for each user, the number of other users met in a period and the number of other users met only once in the same period, respectively. These metrics are indicative of the social attitude of a person in entering in contact with many other people, or conversely to interact only with a few people.

We first measure the average cardinality of the neighborhood (N) and the average cardinality of the dimension of the communities (C) detected. As explained

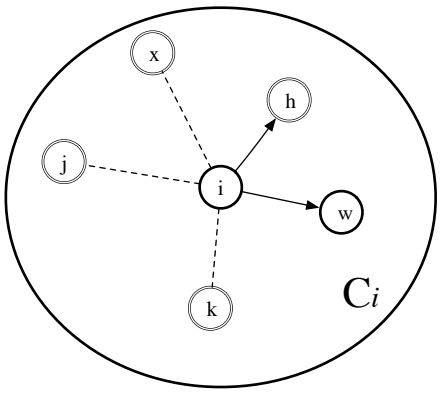


Figure 5.2: Example of community and neighborhood.

in Section 4.2.1, neighborhood and community do not always overlap. The neighborhood is composed by devices in contact i.e. such that there is a link between n_i and n_h, while the community of n_i is composed by devices that have a kind of social tie with n_i (temporal, social, spatial or a combination of them). Figure 5.2 shows the devices in the neighborhood of n_i (marked with a black arrow) and the devices in the community of n_i (marked with dotted lines), note that device n_w is in the neighborhood of n_i but not in its community.

Results concerning the cardinality of the neighborhood N and of the communities C are shown in Figure 5.3. The metrics N and C vary along the simulation time for all the datasets. In particular in the Infocom06 scenario, N and C change according to the time schedule of the Inform06 conference. During the plenary session of the conference, people are more likely co-located in the same room, hence N and C increase (with DRAFT algorithm the more people spend time the more likely they

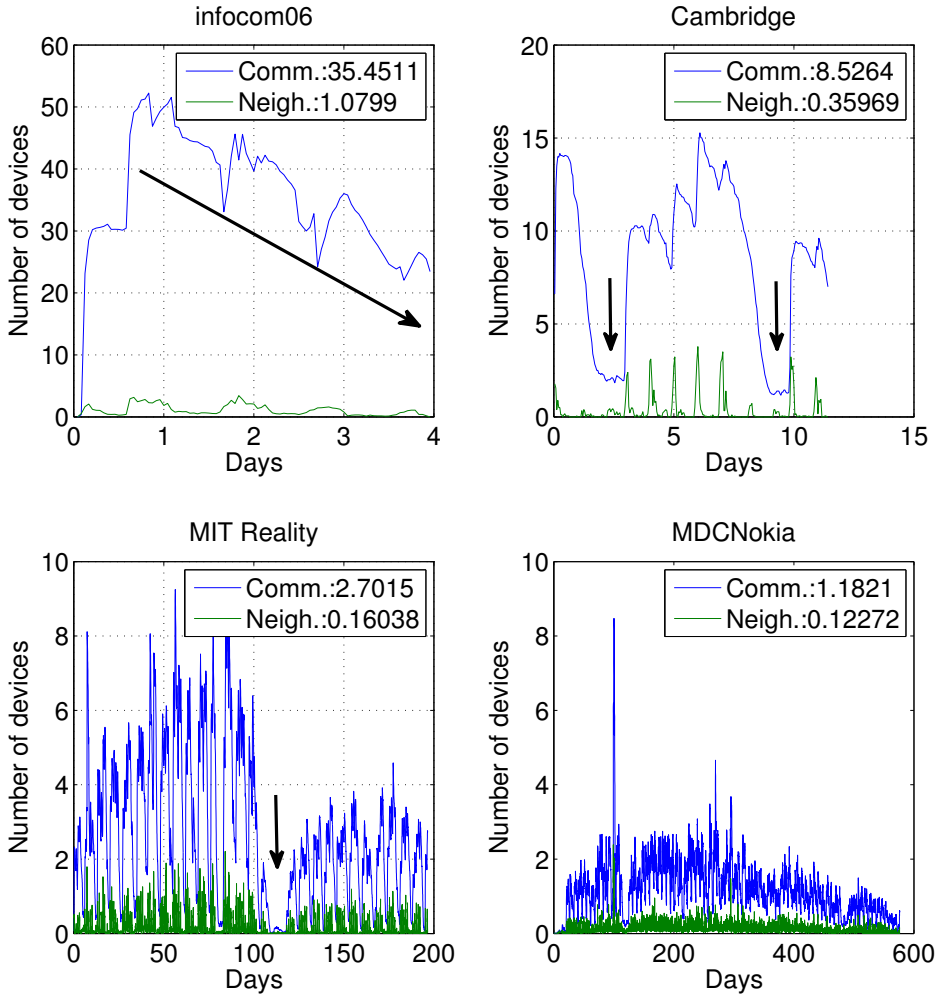


Figure 5.3: Average dimension of communities and neighborhood.

are in the same community). Differently, during specific conference sessions N and C decrease because only few people attend to the session. Moreover, N and C decrease as time passes, since people start leaving the conference. The Cambridge, MIT Reality and MDC Nokia datasets have a different nature with respect to Infocom06 dataset. Such datasets have been collected for a longer time period. The metrics N and C also change along the time but their values are determined by routinary patterns of people, i.e. going to office, coming back home, office meetings, free time etc. For example, the fluctuations of N and C in the Cambridge dataset are tightly coupled with the working day, at the end of every day N and C quickly decrease. The two local minimum in Cambridge scenario (marked with the black arrow) happen in correspondence of the the week-ends. Similar considerations apply

to the MIT Reality dataset and MDC Nokia, at the end of every day N and C decrease. Moreover, the minimum value in the MIT Reality (marked with the black arrows) happen in correspondence to the summer time.

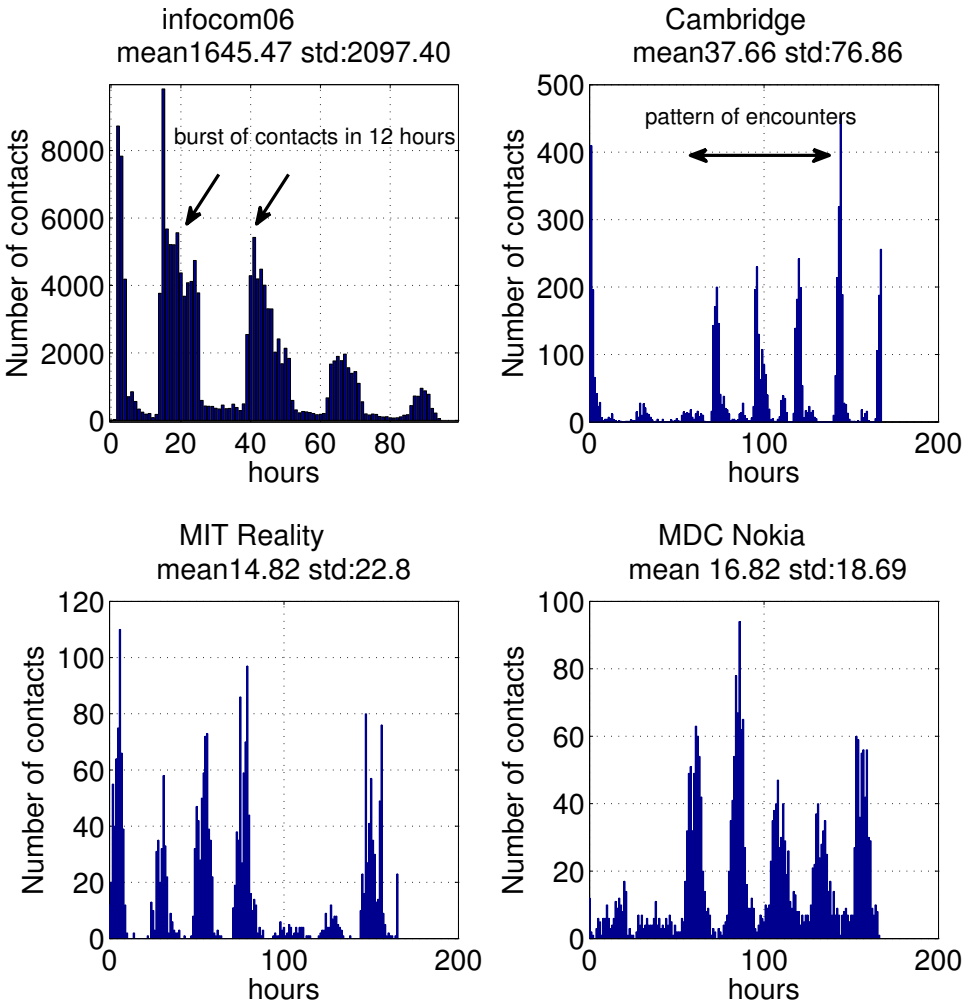


Figure 5.4: Contacts per hour.

Results concerning the number of hourly contacts in period of 7 days are shown in Figure 5.4. The histograms show that all the datasets have a similar trend in the distribution of the contacts along the day. In particular, people tend to meet other people in bursts during the daily hours. Moreover, the number of encounters repeats over the days, giving rise to an intuitive pattern of encounters. In the Cambridge and the MIT Reality the busts are denser in the first part of the daily hours. This probably happens because the environments in these datasets are bounded (university campus) and people are somehow forced to interact during specific working

hours. In the MDC Nokia datasets, instead, the bursts of contacts cover all the daily hours. In fact, such datasets better capture the social interactions also after the working hours.

We measure the complementary cumulative distribution function (CCDF) of the duration of contacts for all the datasets in a logarithmic scale, as shown in Figure 5.5. The curves show that as the duration t increases, the probability of having

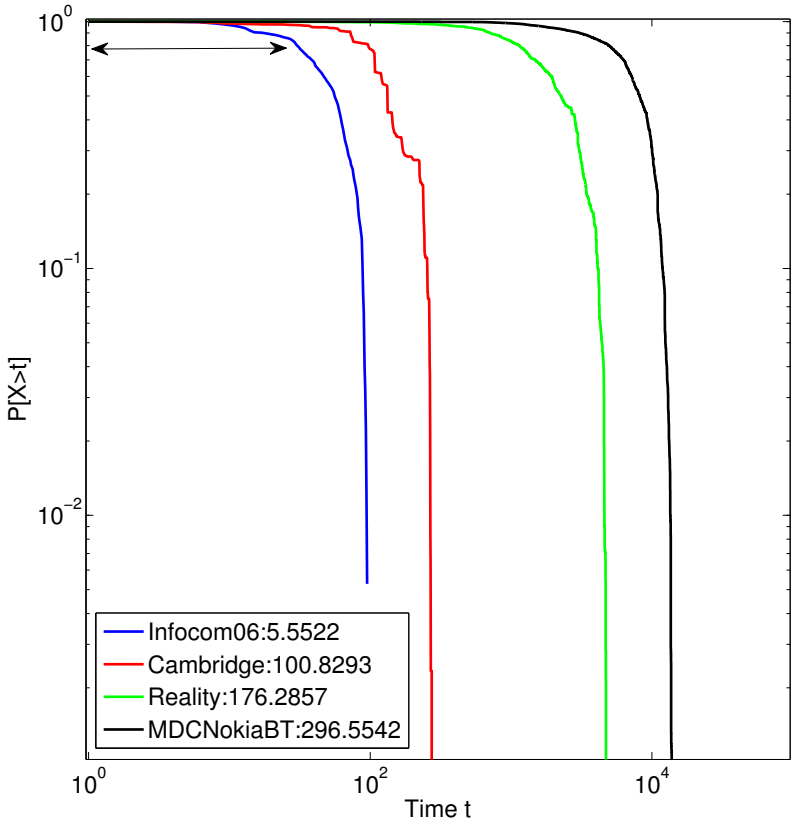


Figure 5.5: CCDF of contact duration.

contacts greater than t decreases. Such decrease is slow in interval $[0 - 100]s$, after which it follows an exponential decay rule. The dataset with the shortest contact duration is Infocom06 (with an average of 5.52 seconds per contact), while MDC Nokia is the dataset with the highest duration of contacts (with an average of 296.5 seconds). The other datasets, namely Cambridge, MIT Reality are bound between them.

We also measure the inter-contact time (ict) among devices. Figure 5.6 shows the CCDF of the ict for all the datasets in a logarithmic scale. All the datasets have a similar CCDF trend. In particular, the CCDF follow a power-low up to roughly 12 hours, after which it decays exponentially. In all the cases, the inter-contact time

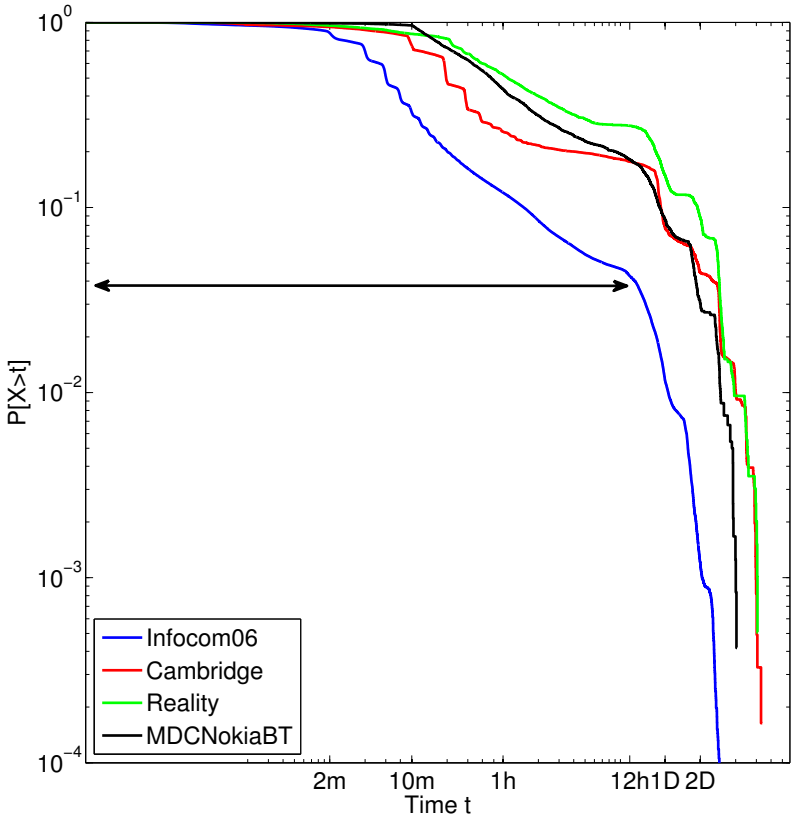


Figure 5.6: CCDF of inter-contact time.

is greater than 120s with high probability, but the curves assume different behaviors after 120s. In particular, users in Infocom06 tend to meet the same people more frequently, while users of MIT Reality meet the same less often. Cambridge and MDC Nokia, are sandwiched between Infocom06 and MIT Reality.

Results concerning the heterogeneity of encounters shown in Figure 5.7. Figure 5.7 represents the heterogeneity of encounters, by plotting a point for each user at coordinates given by the total number of encounters (x axis) and the number of unique encounters (y axis). From the figure, it is seen that in Infocom06, Cambridge and MIT Reality, which are collected in limited geographical areas, users meet many other users more often. This is confirmed by the distribution of the points. In fact, lots of points are placed in the upper part of the diagram that characterizes people with many unique encounters. In the case of MIT Reality, a number of points are also placed in the upper right corner; hence, in this case, people have also many total encounters. The percentage of the population visited by every person in Infocom06, Cambridge and MIT Reality datasets is of respectively 91.02%, 83.48% and 67.60%. MDC Nokia dataset has a distribution of the points shifted in the

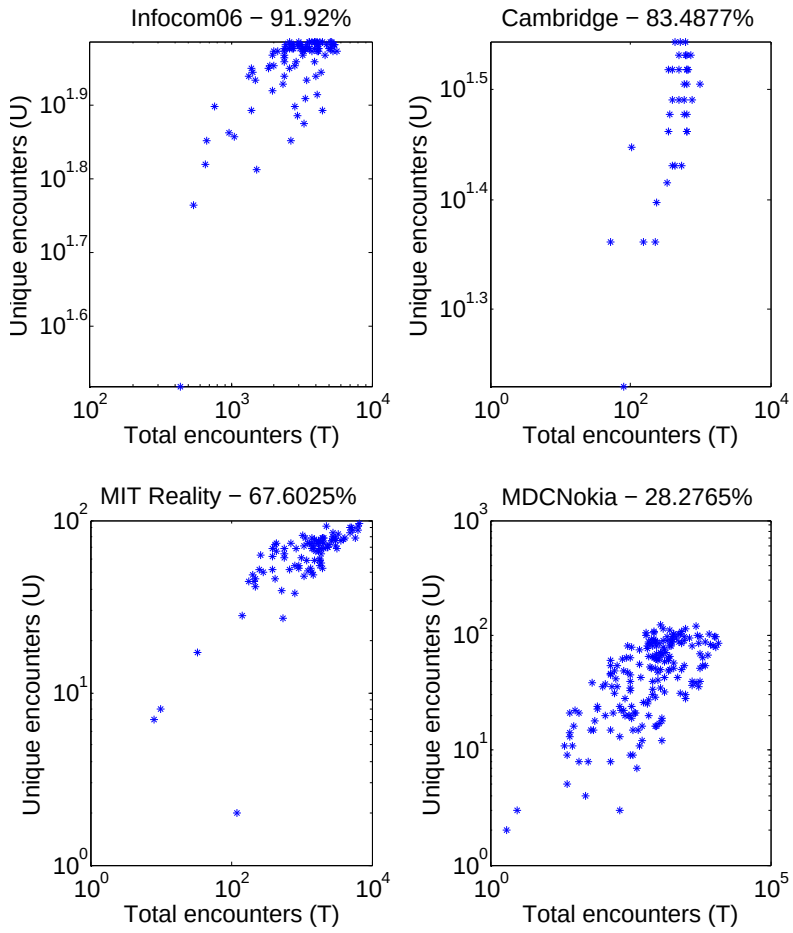


Figure 5.7: Number of encounters w.r.t unique encounters.

lower left corner. This area characterizes people that have few total encounters and few unique encounters. MDC Nokia well reproduce the fact that people, in wide geographical areas, have encounters with a limited number of individuals (an average of 28.27%).

The HCMM Mobility Model

During this thesis we perform some experiments with synthetic co-location traces obtained according to the Home-cell Community-based Mobility Model (HCMM) [121], defined to mimic human mobility. We consider three different deployment areas, small, medium and large, with side of 800m, 1400m and 2000m, respectively. Each area is configured as a grid made up of 5 HCMM groups. Every device is associated to a home cell. Devices in the same home cell share social ties. Some devices also have social ties with other devices from different cells. These devices are called traveler devices. The strength of the social ties of the travelers is determined by the so-called rewiring probability. The rewiring probability models the relationship

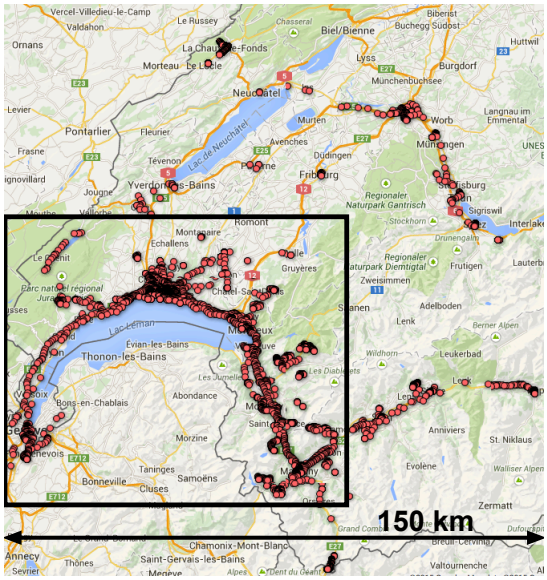


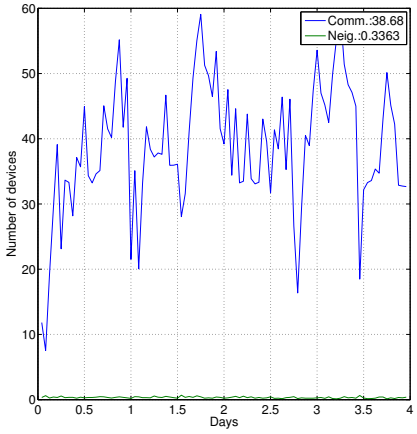
Figure 5.8: Geographical extension of the MDC Nokia dataset.

between devices of different cells and it drives the mobility of devices; it is set to 0.75 for the simulation scenario. The mobility speed is typical of pedestrian walking, i.e., from 1 to 1.86m/s. We call this scenario the HCMM scenario whose metrics are summarized in Figure 5.9. Results concerning the evaluation metrics with the HCMM dataset follow the considerations given for the real-world dataset previously discussed.

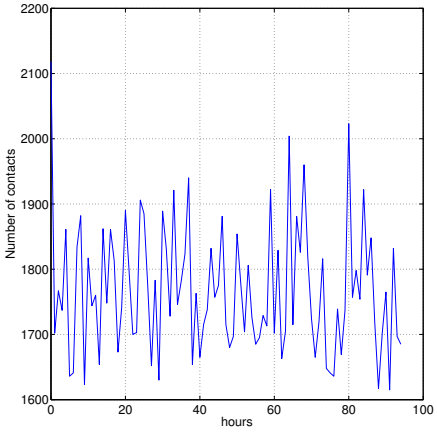
Summary

Table 5.1 summarizes the most important features of the datasets selected. We evaluate SIDEMAN and CORDIAL not with all the datasets, rather we evaluate the two algorithms in different scenarios. In particular, we evaluate SIDEMAN with Infocom06 and the HCMM scenarios to show the benefits of the forwarding policy of SIDEMAN, which is based on the similarity between message and node. In particular, we are interested to verify that even if a node has many other contacts per hour, the SIDEMAN policy selects only a reduced set of candidates to which forward a query or an advertisement. Given this, the average number of contacts in Infocom06 and HCMM is respectively of 1645.47 and 1754, the highest among the 5 dataset, moreover the percentage of nodes visited is respectively of 91.92% and 98.71%.

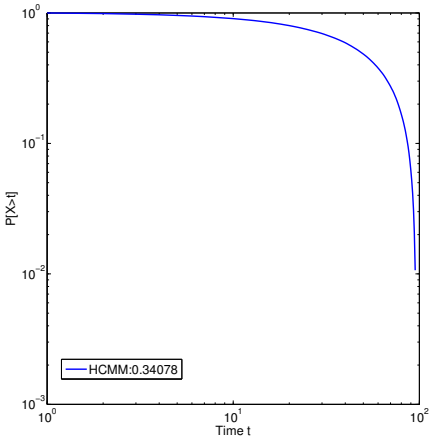
Differently, we evaluate CORDIAL in wider scenarios characterized by a lower number of contacts but more stable. Under this respect we select Cambridge, MIT Reality and MDC Nokia in which a node encounters an average of receptively 37.66, 14.82 and 16.82 nodes, such contacts are more stable with respect to Infocom06 and HCMM indeed the average duration (in seconds) is respectively of 100.82 176.28 and



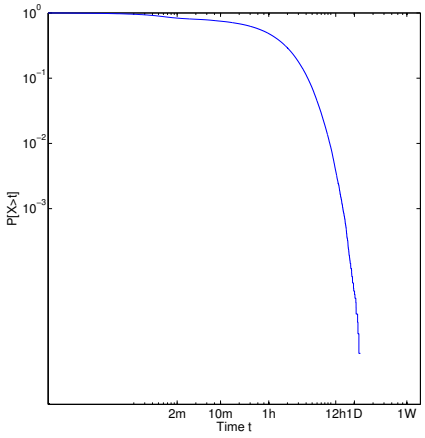
(a) Communities and Neighbourhood



(b) Contact per hour.



(c) CCDF contact duration.



(d) CCDF inter-contact time.

Figure 5.9: Evaluation metrics with HCMM dataset.

296.55. We consider that with thee 3 datasets selected nodes establish more intimate relationships, hence the forwarding policy of CORDIAL can be better analyzed.

We summarize in with Table 5.2 the evaluation process done with SIDEMAN and CORDIAL showing the datasets used and the benchmark algorithms that we compared.

5.3 Benchmark algorithms

To demonstrate its effectiveness in discovering and advertising services, we compared the performance of SIDEMAN and CORDIAL against that of two algorithms for service discovery in wireless networks, which we modified to work in MSN. The two

Table 5.1: Features of the mobility datasets.

	Infocom06	HCMM	Cambridge	MIT Reality	MDC Nokia
Device type	iMote	device	iMote	smart phone	smart phone
Location	Conference	NA	Campus	Campus	City
Duration (days)	3	3	11	246	270
Radio	Bluetooth	NA	Bluetooth	Bluetooth	Inferred
Number devices	78	78	36	97	185
Granularity(s)	120	120	600	300	600
Avg. community	25.4	38.68	8.5	2.7	1.18
Contacts per hour	1645.47	1754	37.66	14.82	16.82
Duration contacts (s)	5.52	0.34	100.82	176.28	296.55
% of total encounters	91.92	98.71	83.48	67.60	28.27

Table 5.2: Summary of the evaluations of SIDEMAN and CORDIAL

	Infocom06	**HCMM**	**Cambridge**	**MIT Reality**	**MDC Nokia**
SIDEMAN	✓	✓			
CORDIAL			✓	✓	✓

algorithms, termed s-Flooding and s-Gossip, are the "social" version of traditional flooding [122] and gossiping [123] algorithms. By using traditional flooding a node would spread queries and services among all its neighbors. Through gossiping a node would instead send those queries and services only to a random subset of its neighbors. The social component is introduced in s-Flooding and s-Gossip by having nodes exchanging services and queries within communities, rather than within their neighbors.

The behavior of s-Flooding and s-Gossip is similar to the one given for SIDEMAN and CORDIAL (see Figure 4.4 and 4.8), in particular the two algorithms are composed by a reactive phase or discovering the service matching with a given query, a proactive phase for disseminating queries ad services stored at a node, and a message management phase for managing the reception of queries and services. The main differences of s-Flooding and s-Gossip with respect to SIDEMAN and CORDIAL are the implementations of the *select forwarding set* of the Reactive phase (Figure 4.4a and 4.8a) and *select services and queries phase* of the Proactive phase (Figure 4.4b and 4.8b). Table 5.3 shows the main differences among the compared algorithms.

We choose s-Flooding and s-Gossip for three reasons: (i) they represent useful performance benchmark for service dissemination in MSN, (ii) they implement a simple but effective strategy for the propagation of information, being both designed to maximize the number of queries and services exchanged among nodes, and (iii), similarly to SIDEMAN and CORDIAL, they are implemented for propagating

Table 5.3: Benchmark algorithms.

	Select forwarding set	**Select advertisements and queries**
SIDEMAN	Members of the current community with interests matching the query.	Advertisements and queries whose interests match those of the community members.
CORDIAL	Members of the current community with highest social centrality metric	Advertisements and queries with highest similarity index
s-Flooding	All members of its current community.	All advertisements and queries to all community members.
s-Gossip	Random number of members of its current community.	Advertisements and queries to a random number of members of the current community.

messages (either queries and services) only inside a community, without flooding the whole network.

5.4 Service Discovery Evaluation Framework

We compare the discovery algorithms proposed in this thesis, namely SIDEMAN and CORDIAL with respect to several evaluation metrics. In particular, we are interested in measuring the relevance of the service advertisements exchanged with respect to the device interests (the accuracy metric) and how many times a device carries a service advertisement that it will access later in time (the proactivity metric). Then, we measure some standard metrics commonly used in the service discovery problem, in particular the average number of advertisements stored in the service cache, the delay between the submission of a query and the reception of an advertisement and the energy cost due to the overhead introduced by the discovery algorithm.

Moreover, the design of the CORDIAL algorithm requires to better analyze the benefits of the query and advertisement forwarding strategies described in Section 4.4.2, hence we define several metrics whose goal is to evaluate the capability of the algorithm to diffuse efficiently queries and advertisements among devices. The metrics used for the evaluation of the discovery algorithms are described below.

- *Accuracy*, this metric measures the accuracy of the service discovery algorithm in propagating advertisements of interest for the recipient devices. It is defined

as the ratio between the number of service advertisements stored in the cache of device n_i that are of interest for n_i and the total number of advertisements stored in its cache. Clearly, the value of the accuracy is in $[0, 1]$. A value of 0 means that none of the service advertisements in which device n_i is interested are in its service cache (worst case). A value of 1 indicates the best case: device n_i stores only service advertisements in which it is interested.

- *Proactivity*, this metric is defined as the ratio between the number of times a device finds the advertisement adv_j already in its service cache and the number of times a device has to query for it. This metric indicates how well a service discovery algorithm propagates to a device the advertisements that might be of interest for it. The value of the Proactivity is in $[0, 1]$. A value of 0 means that every time a device needs a service advertisement it has to query for it (worst case). A value of 1 indicates the best case: every time a device needs a service advertisement, that advertisement is in its cache.

- *Service Cache, SC*, this metric estimates the average number of advertisements stored locally in the device cache.

- *Energy Cost per device, EC*, this metric provides an estimation of the average energy consumption (express in Joule) incurred by each device during network operations (such as forwarding advertisements and queries). The energy cost is estimated by computing the dimension in byte of every type of message, i.e. queries and advertisements. Then, we compute the average number of messages exchanged for every type of message, and we derive an estimation of the energy consumption of every device by considering the energy consumption per byte of the WiFi/ Bluetooth chip Broadcom®BCM4330 of the Samsung Galaxy S III smart phone (this radio was chosen because of the availability of specifications of energy consumption). Since SIDEMAN and CORDIAL are independent from any specific community detection algorithm, we decide not to consider the energy cost of such phase. This choice allows us to focus the attention only on the overhead due to the service discovery algorithms;

- *Query Response Time, QRT* (in seconds), this metric is defined as the average time elapsed between when a query is sent and the reception of the first service advertisement matching that query. This metric informs us about how effective a discovery algorithm is letting a device receive the service it needs swiftly.

- *Query Answered, QA*, this metric measures the average number of queries that have been answered during all the simulation time even with direct and indirect responses.

- *Network Overhead , NO*, this metric measures the average number of packets that a node exchanges by running each of the discovery algorithms. This met-

rics provides an indication of the overhead introduced by each of the discovery algorithms in terms of network operations.

5.5 Evaluation of SIDEMAN

We evaluate SIDEMAN and two other strategies for service discovery with a custom Java-based simulator (resulting from this thesis). The simulation scenarios we consider are Infocom06 scenario, obtained by using the dataset described in Section 5.2, and the HCMM scenario. The simulator we develop implements two important components of service discovery in MSN: *(i)* community detection and *(ii)* the service discovery algorithms. First community detection is run every $p = 300$s and it implements a well known solution, called AD-SIMPLE [59]. AD-SIMPLE is an enhanced version of SIMPLE, an algorithm for community detection that has been shown to outperform previous solutions such as k-CLIQUE, MODULARITY and SIMPLE [60]. The reasons for this choice are multifold: first of all, AD-SIMPLE is a distributed algorithm, thus being suitable to run locally to every device by using only local information gathered by the device (refer to the temporal metrics described in Section 4.2.1). Secondly, despite being distributed, the communities it detects are very similar to those detected by SIMPLE [60]. In fact, the similarity index between the communities detected by AD-SIMPLE and SIMPLE is far higher than that of the communities detected by k-CLIQUE and MODULARITY and their distributed counterparts. Finally, AD-SIMPLE implements an effective mechanism for removing devices from a community that they have not joined for a while. In our simulations we stipulate that two communities are recognized as the same community if the Jaccard similarity index γ is higher than $\tau = 0.8$.

Second, our simulator implements SIDEMAN (Section 4.2.1) and the benchmark algorithms described in Section 5.3. Device behavior is determined by considering query generation rate, service advertisement generation rate and the distribution of interests to devices. The query generation rate determines how many queries are generated by the devices. We model this rate as a Poisson process of intensity λ queries per second. In our simulations we set $\lambda = 3$. In particular, every second a number q of query is generated and assigned to q devices selected randomly and uniformly among all devices. The service advertisement generation rate concerns the service advertisements the devices store in their service cache without using the service discovery algorithm. Initially, a device cache is empty. In time, devices start storing new service advertisements in their cache in order to be able to exchange them among the devices during the opportunistic encounters. The service generation rate models how many new advertisements are stored in the device cache. It is also a Poisson process of intensity μ service advertisements per second. The value of μ is set to 3. Every second a number h of service advertisements is generated and assigned to h devices selected randomly and uniformly among all devices. Each time the h service advertisements are drawn randomly and uniformly from a set $\mathcal{S}$ of m

service advertisements.

The distribution of interests among the devices depends on the scenario. For both of the scenarios used in SIDEMAN, we assume that the interests assigned to devices do not change along the simulation time. We consider such assumption realistic, since people (and hence devices carried by them) do not change interests so frequently along the time, ad discussed in [124]. In the Infocom06 scenario we use the association between interests and devices that comes with the dataset. In particular, every participant to the conference filled out a questionnaire regarding its working interests. The questionnaire asks a set of a multiple choice answers that we parse and use as interests for the devices. Some examples of interests are: Mobile Ad Hoc Networks, Network Architectures, Optical Networks, Quality of Service, Routing Protocols etc.

In the HCMM scenario interests are not provided with the co-location trace, therefore we given them to devices proportionally to the time spent by each device in a specific device region (HCMM cell). In particular:

- every cell x is assigned k_x interests according to a Zipf's distribution (with parameter $skew = 1$). The parameter k_x is the ratio between the total number n of interests in the simulation (set to 35 in our simulations) and the number of cells in the HCMM grid. In our experiments $k_x = \frac{35}{5} = 7$. The interests assigned to cell x are denoted with the set $\bar{\mathcal{I}}_x$.
- Device n_i is assigned $|\mathcal{I}_i|$ interests from $\bar{\mathcal{I}}_x$ proportionally to the time it spends in cell x. In this way, devices visiting the same cell for a long time share similar interests.

The values of λ and μ have been selected consistently with those of similar scenarios in previous works [82, 125, 126].

Our simulation results for the HCMM scenario are obtained by averaging the outcomes of 1000 runs, each running for 300000s (around three days and a half), each time on a different HCMM trace. This number of experiments achieves a statistical confidence of 95% within a 5% precision. The confidence interval are not shown in the graphs since they are too small to be appreciated.

5.5.1 Results

Results concerning Accuracy and Proactivity metrics in HCMM scenario are shown in Figure 5.10. The figures refer to 78 devices roaming in a $800 \times 800\text{m}^2$ area. Results for devices traveling in bigger areas show similar trends.

Accuracy

Results concerning Accuracy are shown in Figure 5.10a. SIDEMAN always obtains an Accuracy equal to 1, meaning that devices store in their caches all and only

service advertisements to which they are interested in. This is a consequence of the very nature of SIDEMAN strategy according to which an advertisement is sent to a device only if that device is interested in it. The Accuracy of s-Flooding and s-Gossip instead decreases in time, reaching a value as low as 0.5 by the end of the observation period. In other words, half of the service advertisement stored in the service cache of devices running s-Flooding and s-Gossip will never be used by those devices. This is because s-Flooding and s-Gossip advertisement exchange is not interest-based. We notice that s-Gossip slightly outperforms s-Flooding. This is because a device running s-Gossip sends service advertisements to a number of devices that is lower than the number of devices involved in advertisement exchange in s-Flooding.

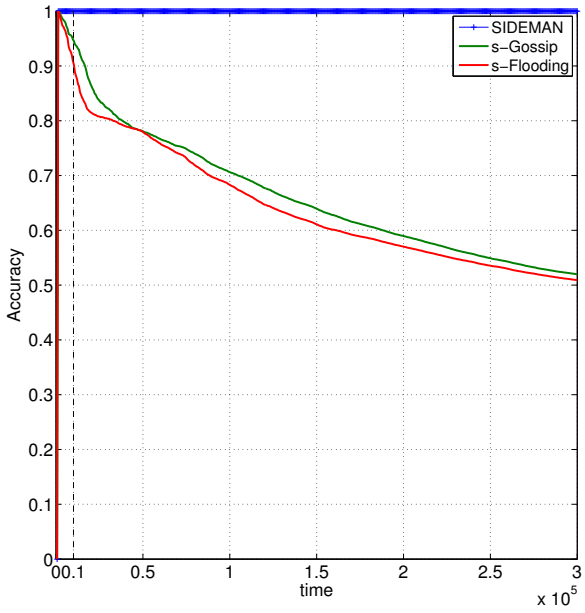
Proactivity

Results for the Proactivity achieved by the three algorithms are shown in Figure 5.10b. The trend is similar for SIDEMAN, s-Flooding and s-Gossip. In particular, with passing time the Proactivity increases and tends to its maximum 1. Moreover, by the end of the observation period it is 0.96 for each of the three algorithms. As expected, the Proactivity increases rapidly during the initial time interval $[0, 0.5 \cdot 10^5]$s because devices start with no service advertisements in their cache and then they begin moving and exchanging advertisements as they meet and form communities. By the end of the observation time they have exchanged enough service advertisements to find the ones they need in their cache. Results about this metric show clearly how effective the discovery strategy followed by SIDEMAN is for the distribution of the service advertisements. The Proactivity of SIDEMAN is never noticeably lower than that of s-Flooding and s-Gossip, two algorithms that represent our benchmark and that are designed to maximize (in our case) the diffusion of the service advertisements. In fact, we notice that the Proactivity of SIDEMAN is always (slightly) better than that of s-Gossip throughout the network operation time. This is because, despite the number of service advertisements distributed by s-Gossip is far higher that that of those distributed by SIDEMAN, most of these service advertisements are not of interest to the querying device.

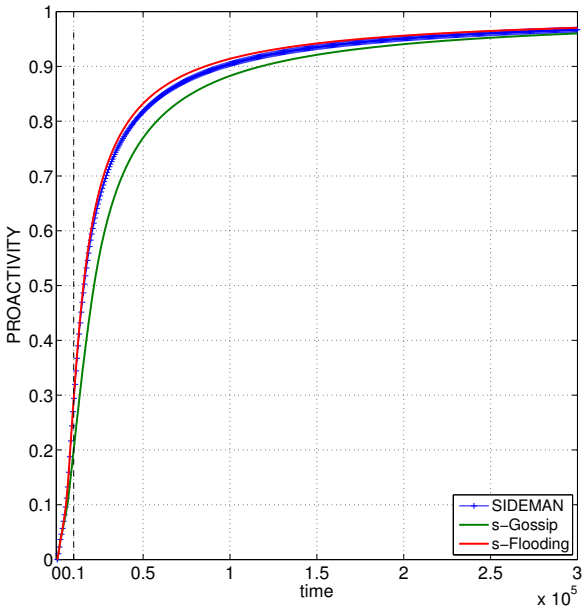
Query Response Time

Results for the query response time of the three algorithms are shown in Figure 5.11a. We notice that the time needed to respond to a query grows in time. This is because as time progresses, devices tend to visit roughly the same communities all over again, and if a device does not find a service advertisement in its cache right away, and has to query for it, it is unlikely that it will receive this advertisement from the devices that it has already met and keeps meeting. In time, it might enter a new community whose members might have the required service, but that happen with low probability. The performance of SIDEMAN is bound between that of s-Flooding

and s-Gossip.



(a) Accuracy.



(b) Proactivity.

Figure 5.10: Accuracy and Proactivity metrics in HCMM scenario.

Since s-Flooding distributes the largest number of service advertisements inside

a community, devices carry more advertisements around, increasing the probability of being able to respond to a query. That is why its query response time is the lowest. The nature of s-Gossip is to distribute advertisements to a subset of the devices in a community selected randomly. The effect is that devices exchange a lot of advertisements in which they are not interested, and therefore the probability of a device to respond to a query decreases, increasing the response time. We observe that by the end of network operation, SIDEMAN has exchanged 84.32% (82.84%) less service advertisements than s-Flooding (s-Gossip). However, since devices running SIDEMAN exchange only those advertisements in which they are interested, the probability of a device to find a service in its cache is fairly high, as confirmed by checking the number of queries sent by devices throughout the observation time. As a consequence the query response time for SIDEMAN soon becomes lower than that of s-Gossip.

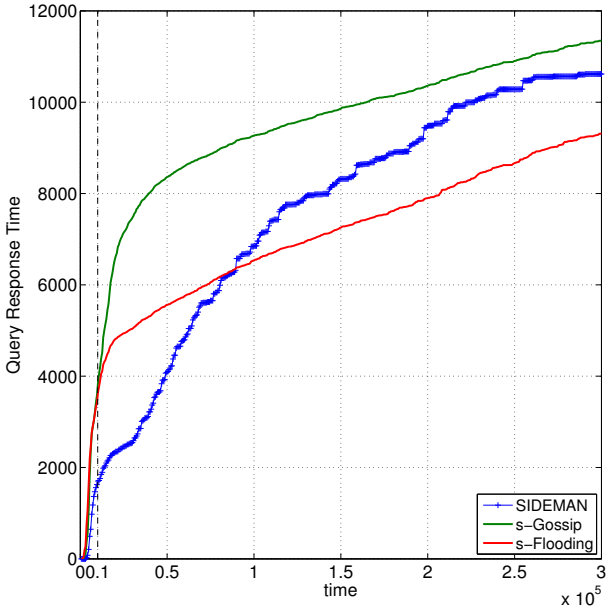
Energy Cost

In terms of energy cost SIDEMAN remarkably outperforms s-Flooding and s-Gossip (Figure 5.11b). The energy consumption of SIDEMAN is at least 7 times lower than that of s-Flooding, and 6 times lower than that of s-Gossip. The energy cost of s-Gossip is slightly lower than that of s-Flooding because, as mentioned already, for its very nature s-Gossip exchanges less service advertisements that s-Flooding. To show the impact on device lifetime of running the three algorithms, we consider devices powered by a standard battery pack with an average consumption of 7.9Watt/hour (i.e., a battery pack with a capacity of 2100mAh and voltage of 3.8V). By the end of the observation time devices running s-Flooding have consumed and average of 87.7% of their initial battery charge, devices running s-Gossip have consumed $80,2\%$ of it, while devices running SIDEMAN have consumed only 12.5% of the initial battery charge.

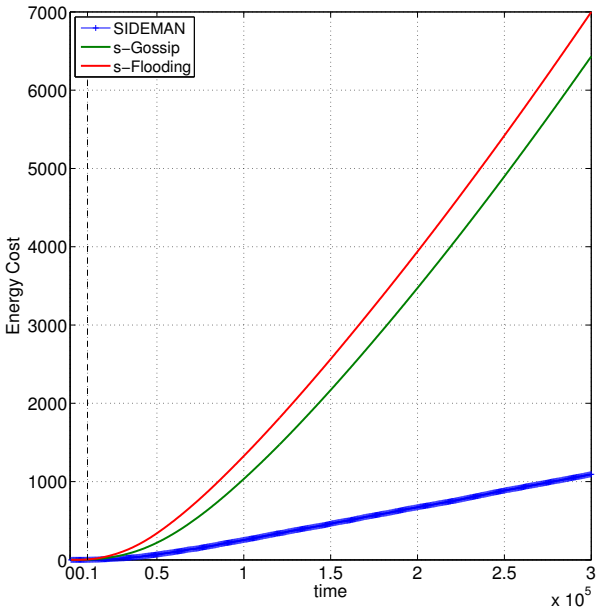
Service Cache

Results concerning the SC metric are shown in Figure 5.12. The Service Cache metric is consistent with the Accuracy metric, in particular SIDEMAN controls the dimension of the service cache during the simulation time. More precisely, the dimension of the cache for the devices running SIDEMAN increases during a limited time interval (e.g. $[0-0.5]\times 10^4 sec.$ for the HCMM configurations 800×800), after which the dimension of the cache is stable until the end of the simulation. In a different way, the Service Cache metric for s-Flooding and s-Gossip increases linearly with the time in all the HCMM configurations, hence the more the simulation last, the bigger is the service cache of the devices running such algorithms. The Service Cache graph shows that devices running SIDEMAN exchange fewer service advertisements than s-Flooding and s-Gossip algorithms, this result has no any consequence on the performance metrics discussed so far, in particular the Accuracy

and the Proactivity metrics.



(a) Query response time.



(b) Energy cost.

Figure 5.11: QRT and EC metrics in HCMM scenario.

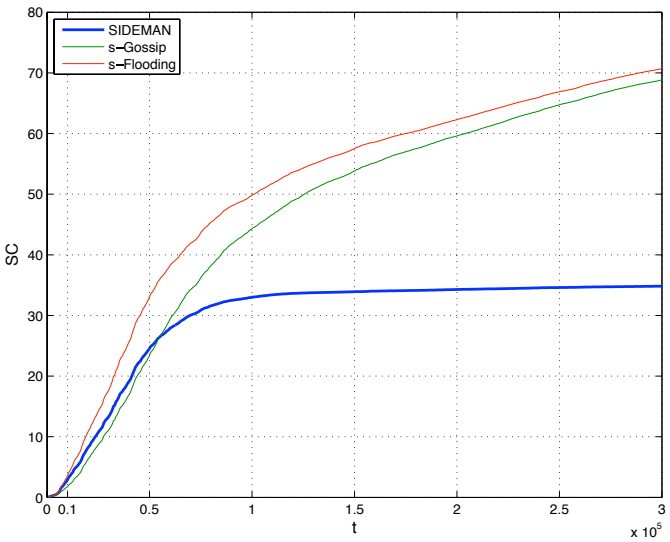


Figure 5.12: Service Cache metric in HCMM scenario.

Network Overhead

Results concerning the network overhead are shown in Figure 5.13. We observe that the average number of packets needed for a single run of SIDEMAN (comprising the reactive and proactive phases and the management of incoming messages) is lower than that of s-Flooding and s-Gossip. During the first part of the simulation $[0, 0.5\dot{1}0^5]$ the network overhead of the three algorithms is comparable, because nodes start with empty caches. As time progresses, nodes begin exchanging queries and service advertisements and the algorithm performances differ. In particular, we observe that s-Flooding is the algorithm with the highest network overhead because its flooding strategy (Table 5.3) requires to forward each query and each service to every member of the node community. Differently, s-Gossip incurs lower complexity than s-Flooding because queries and services are forwarded to a number of community members less than or equal to the size of the node community. SIDEMAN achieves the lowest network overhead along with perfect Recall and comparable Gain, which makes it suitable for application scenarios with nodes that are resource constrained.

Performance in larger deployment areas

We have also evaluate the performance of the three algorithms in larger deployment areas, namely, 1400m $\times$1400m and 2000m $\times$2000m. We observe that by increasing the dimension of the scenario without increasing the number of devices they tend to travel shorter routes and restrict their mobility to few locations. Such a more limited mobility implies that a device enters in contact with a small sub-set of the other devices. In particular, we observe that devices tend to form smaller commu-

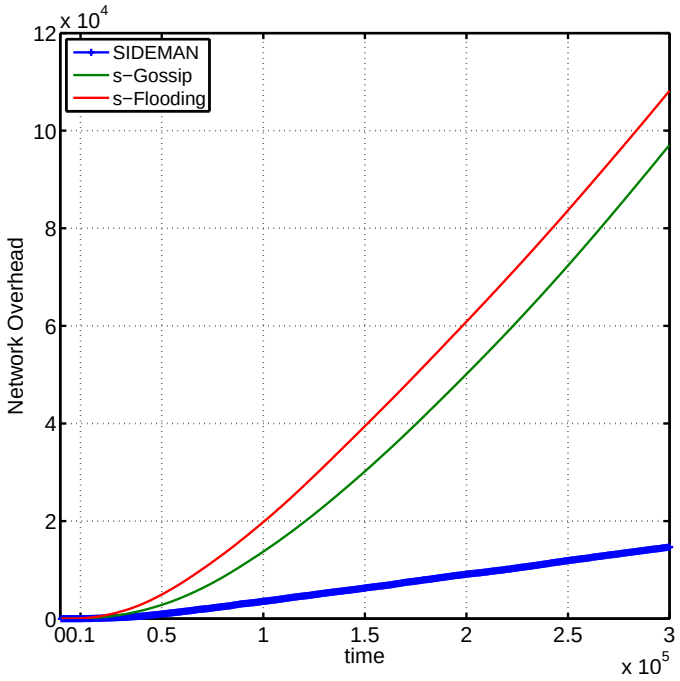


Figure 5.13: Network Overhead metric in HCMM scenario.

nities with respect to scenarios in smaller areas Performance results are affected as follows: the Accuracy of SIDEMAN is equal to 1, irrespective of the size of the scenario. s-Flooding and s-Gossip show a Accuracy that decreases slower than that in the original scenario (Figure 5.10a). This is because nodes encounter fewer devices and therefore exchange fewer advertisements. As a result the probability of storing advertisements not of interests to the devices decreases. The Proactivity of the three algorithms is slightly lower than that shown in Figure 5.10b (less than 2%). This is due, again, to the lower mobility of the devices, which decreases the probability of exchanging advertisements with other devices, and therefore of the probability of a device of finding an advertisement in its cache. As expected, the energy cost for larger scenario decreases, because of the lower number of advertisements exchanged: the fewer the device exchange advertisements and queries, the lower the energy consumption. The query response time increases with the size of the scenario, since decreased mobility leads to a slower propagation of the advertisements. Therefore, as soon as a device queries for a service it has to wait a longer time before receiving the matching service. Lastly, the average size of the service cache also decreases with increasing the size of the scenario, since low mobility reduces the probability of exchanging advertisement and therefore also the probability of storing advertisements in the cache.

The rest of this subsection shows the Accuracy and Proactivity metrics of SIDEMAN, s-Flooding and s-Gossip in the Infocom06 scenario.

Accuracy

Results concerning Accuracy are shown in Figure 5.14a. The Accuracy of SIDEMAN is 1, as expected. The Accuracy of s-Flooding and s-Gossip rapidly decreases to 0.2. This depends on the nature of the Infocom06 traces, where devices in the same community share a low percentage of interests (this percentage in time is as low as 0.8%). For instance, there are times when all participants are at a common event, such as a meal break, or a plenary session. In this case, the community is very large. The interests shared by the community members are instead very few. This is in contrast with the results from the HCMM scenario, where devices in the same community would share instead up to 80% of their interests. This explains why, eventually, in the HCMM scenario s-Flooding and s-Gossip show value of Accuracy higher than those in the Infocom06 scenario (i.e., around 0.5 vs. 0.2). In the Infocom06 scenario, devices running s-Flooding and s-Gossip are more likely to exchange service advertisements in which they are not interested. As a result, the Accuracy decreases more rapidly than that in the HCMM scenario.

Proactivity

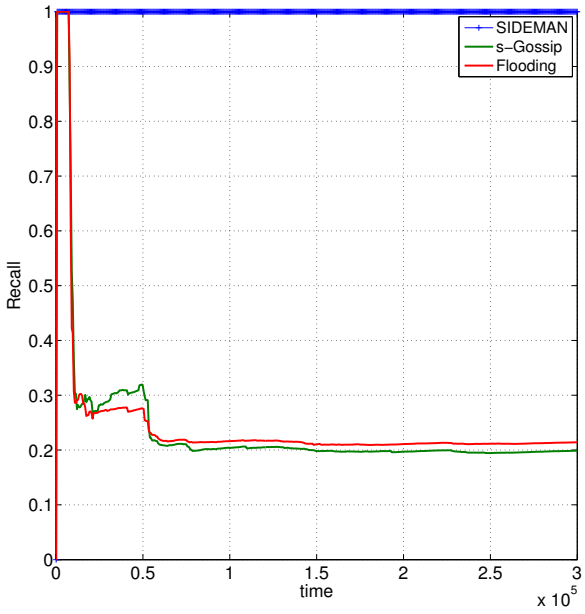
Results for the Proactivity of the three algorithms are shown in Figure 5.14b. The trend is the same for all algorithms and by the end of the observation time the Proactivity reaches the value 0.94. Similarly to Figure 5.10b, the Proactivity increases rapidly in the interval $[0 - 1 \cdot 10^5]$s because devices start with no service advertisements in their cache and then they begin moving and exchanging service advertisements as they meet and form communities. By the end of the observation time, devices have exchanged enough advertisements to find the one they need in their cache. Although showing the same trend observed for the HCMM scenario, the values of the Proactivity in the Infocom06 scenario are lower than that for the HCMM one. This is because the communities at Infocom06 share fewer interests than those shared by the HCMM communities. As a consequence, when a device queries for an advertisement, the probability of receiving a response to its query is low, and consequently the Proactivity grows more slowly.

Figure 5.15 shows the QRT and EC metrics for the Infocom06 scenario.

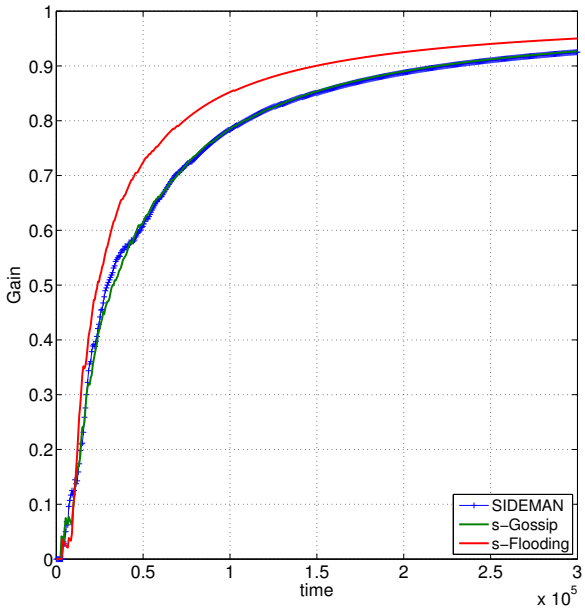
Query Response Time

Results for QRT of the three algorithms are shown in Figure 5.15a. The time needed to respond to a query grows in time. As in the HCMM scenario, devices tend to keep visiting the same communities. As such, if a device has to query for a service advertisement, it is unlikely that it will receive this advertisement from the devices that it has already met and keeps meeting.

We consider that the query response time of SIDEMAN shown in Figures 5.11a and 5.15a is acceptable for two main reasons. First, the two simulation scenarios taken into account are composed by mobile devices with a limited battery autonomy,



(a) Accuracy.



(b) Proactivity.

Figure 5.14: Accuracy and Proactivity metrics in Infocom06 scenario.

hence it is important to design a discovery algorithm that finds a trade-off between the energy consumption and the service discovery performance. Indeed as shown in

Figures 5.11b and 5.15b, the use of flooding-based algorithms (e.g. s-Flooding and s-Gossip) leads rapidly to an uncontrolled battery depletion even before the end of the observation period (see Figure 5.15b). In a different way, SIDEMAN consumes at most 12.5% of the battery charge after 3 days of simulation. Second, the query response time graphs show that it is important to reduce the number of queries sent, by maximizing the probability that devices will find advertisements in their cache (in other words it is important to maximize the Proactivity). SIDEMAN, as shown in Figures 5.10b and 5.14b, obtains a Proactivity that is always comparable with respect to s-Flooding and s-Gossip, moreover the number of queries sent by devices implementing SIDEMAN is lower than that of the devices implementing s-Flooding and s-Gossip.

Energy Cost

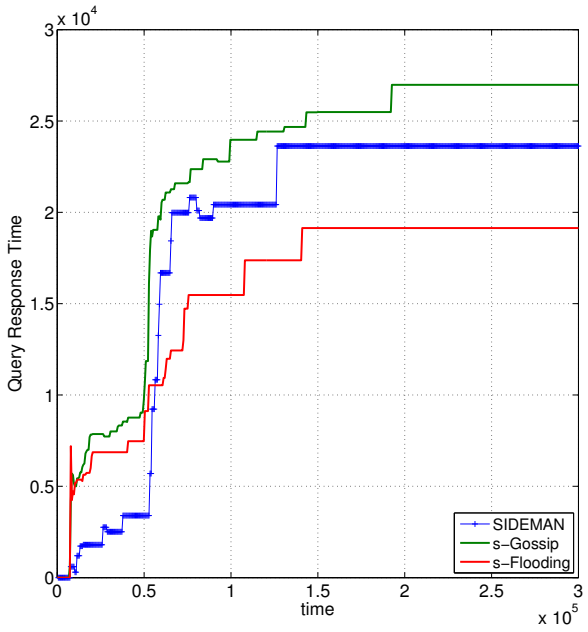
Figure 5.15b shows the energy cost incurred by running the three algorithms. As in the HCMM scenario, SIDEMAN outperforms s-Flooding and s-Gossip. In particular, the energy cost of SIDEMAN is far lower than that of the other two algorithms. If we consider a standard battery pack as described for the HCMM scenario, we observe that devices running s-Flooding and s-Gossip deplete their battery approximately after $1.5 \cdot 10^5$s (half of the observation time), while at the end of the observation time devices running SIDEMAN have consumed only 12% of their energy.

Service Cache

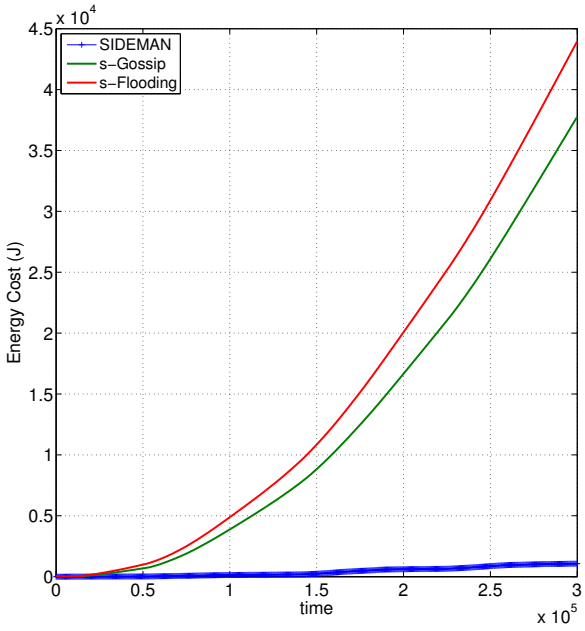
Results concerning the Service Cache metric is shown in Figure 5.16. The Service Cache metric is consistent with the Accuracy metric, in particular SIDEMAN controls the dimension of the service cache during the simulation time. More precisely, the dimension of the cache for the devices running SIDEMAN slowly increases during the time, the end of the simulation time devices store 100 service advertisements in their cache. In a different way, the Service Cache metric for s-Flooding and s-Gossip increases more significantly with the time, hence the more the simulation last, the bigger is the service cache of the device running such algorithms.

Network Overhead

The network overhead of the three algorithms in shown in Figure 5.17. The number of packets exchanged grows similarly to what observed in the HCMM scenario. In particular, s-Flooding and s-Gossip incur the highest network overhead, while SIDEMAN obtains the lowest network overhead. Since in the Infocom 06 scenario nodes tend to encounter more nodes that in the HCMM scenario, the number of services and queries exchanged is higher, which explains the higher network overhead in this case. By the end of the simulation time nodes running SIDEMAN incur a network overhead 77.88% lower than that of s-Flooding and 75.71% lower than that of s-Gossip.



(a) Query response time.



(b) Energy cost.

Figure 5.15: *QRT* and *EC* metrics for Infocom06 scenario.

5.6 Evaluation of CORDIAL

In order to better identify the weaknesses of SIDEMAN we require to study more closely several mobility datasets: Infocom05 and Infocom06, MIT Reality, MDC

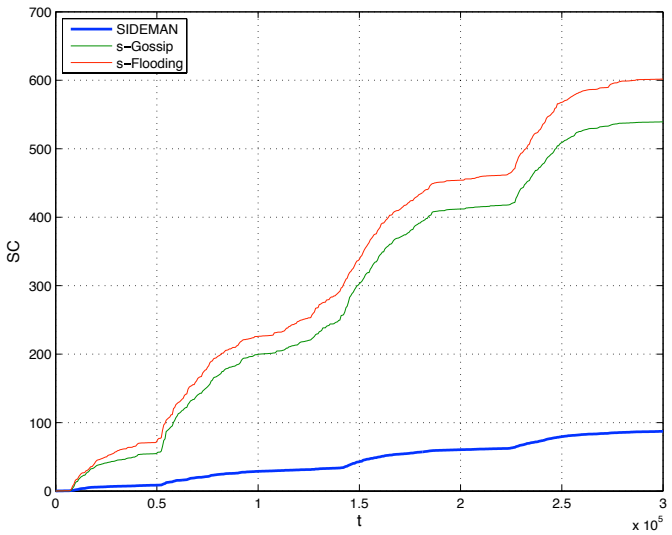


Figure 5.16: Service Cache metric in Infocom06 scenario.

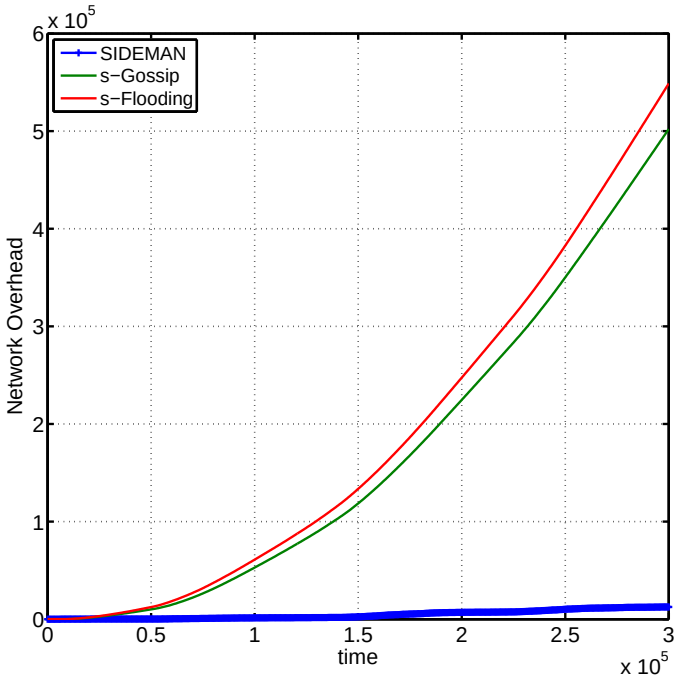


Figure 5.17: Network Overhead metric in Infocom06 scenario.

Nokia and Cambridge as well as synthetic datasets generated through mobility simulators like HCMM (see Table 5.1). To this purpose we use The ONE Opportunistic Network Simulator [127] that offers a stable tool-chain for reading co-location traces. The ONE is designed with a plug&play mechanism to that it is easy to add to the simulation the components needed for reproducing the scenario required. For exam-

ple, The ONE provides the implementation of several routing protocols for DTN as well as a number of mobility models and syntax parsers for using real-world datasets.

We configure The ONE with *(i)* a community detection algorithm and *(ii)* three service discovery algorithms.

First, the variety of the simulation scenarios that we chose for the evaluation of CORDIAL requires to adopt an easy-to-configure community detection algorithm. To this purpose, we use the DRAFT algorithm [61]. As a general rule, by increasing the δ value, DRAFT detects communities of devices that meet for longer periods of time, conversely with lower values of δ, DRAFT captures devices that meet for short time. Our goal is to detect long-lasting communities hence, according to [61], we configure DRAFT with the following tuple $< \tau = 3600s, \delta = 0.9, t = 3600s >$.

Second, we compare CORDIAL with respect to the SIDEMAN algorithm as well as with the s-Flooding strategy described in Section 5.3. The device behavior is determined by considering the same parameters taken into account for the evaluation of SIDEMAN, namely query generation rate and service advertisement generation rate. These parameters have been configured as discussed in Section 5.5. Differently from the evaluation of SIDEMAN, all the datasets taken into account do not provide information about the interests of people carrying the devices. For this reason, we use a Zipf distribution for the assignment of interests to devices. The Zipf distribution reproduces an intuitively but common behavior of humans: many people are interested in a small set of interests, while only few people have specific interests. We use a Zipf distribution with the *skew* parameter set to 1 (as done in [82]) and with a total number of interest set to $\mathcal{I} = 100$. Figure 5.18 shows the distribution of interests given for the 3 datasets, interests with the highest occurrences are within the range $0, 10$.

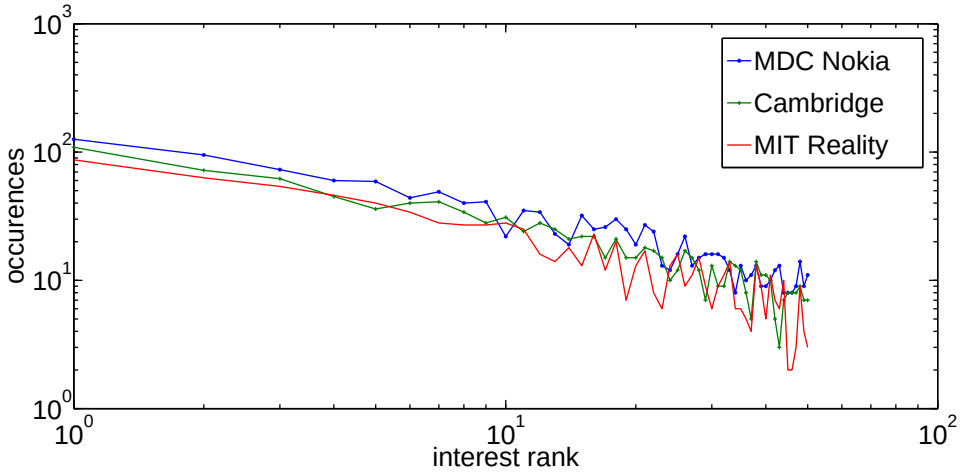


Figure 5.18: Distribution of interests in the datasets.

5.6.1 Results

We report in this section the results of the CORDIAL algorithm compared with respect to SIDEMAN and s-Flooding in Cambridge, MIT Reality and MDC Nokia scenarios.

Accuracy

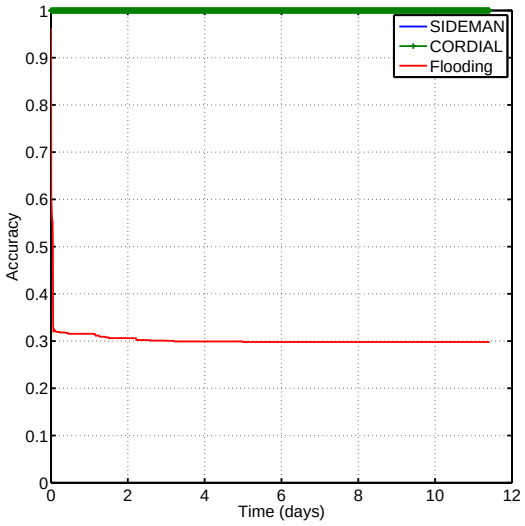
Results concerning the Accuracy metric are shown in Figure 5.19. In all the scenarios, CORDIAL and SIDEMAN obtain a perfect value of Accuracy. In fact, devices running such algorithms carry in their service caches only advertisements whose interests match with the devices's ones. As expected, the Accuracy metric of s-Flooding decreases in time. Hence devices running the s-Flooding algorithm carry advertisements considered off-topic for the device and never accessed during all the simulation time. By the end of the observation time, in the Cambridge scenario the Accuracy of s-Flooding decreases down to 0.3, in the Reality scenario to 0.39 and in the MDC Nokia scenario to 0.33.

Proactivity

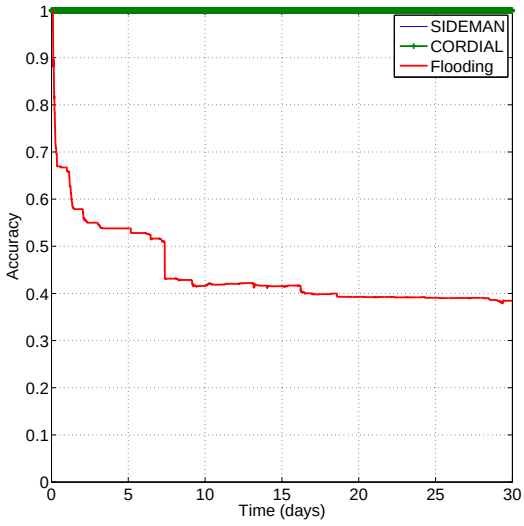
Results concerning the Proactivity metric are shown in Figure 5.20. In all the simulation scenarios, the Proactivity value increases with the simulation time. In particular, at the beginning of every simulation, devices start without any advertisement stored locally and, as time passes, they exchange advertisements with other devices. The results of the Proactivity metric are affected by the strategies used by three algorithms as well as by the mobility traces used (refer to Table 5.1 for the most important features of mobility traces adopted).

Figure 5.20a shows the results of the Proactivity metric in the Cambridge scenario. In this case the Proactivity value tends to its optimal value 1. As discussed in Section 5.2, devices in the Cambridge scenario roam in a restricted geographical area (university campus), they meet frequently (low ict, see Figures 5.6) and with the same set of devices (high percentage of unique encounters, see Figure 5.7) hence the cardinality of the resulting communities is high. As a consequence, the probability of exchanging advertisements among devices is higher than that scenarios with a lower number contacts and with smaller communities. The Proactivity quickly increases during the first 2 days of simulation, after which the curve continues growing but with a slower slope. CORDIAL obtains a value of Proactivity always comparable with respect to the s-Flooding algorithms (our benchmark) and always higher than that the SIDEMAN algorithm. By the end of the observation time, the Proactivity of the three algorithms is 0.95 meaning that a device willing to access an service finds, with high probability, in its cache the service advertisement needed. In MIT Reality and MDC Nokia the Proactivity metric increases slower than that the Cambridge scenario, as shown in Figures 5.20b and 5.20c. With the MIT Reality scenario devices are also bounded in a restricted geographical area,

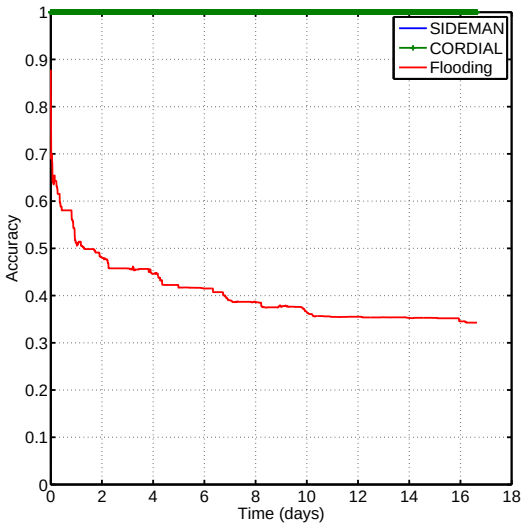
but the number of contacts per hour (see Figure 5.4) and the percentage of devices encountered (see Figure 5.7) is lower than that the Cambridge scenario, giving rise to communities also smaller. For these reasons, the Proactivity values of the three algorithms do not reach its optimal vale of 1. With the MIT Reality scenario CORDIAL outperforms both s-Flooding and SIDEMAN, meaning that our algorithm, even in scenario with medium-size communities, implements an effective forwarding strategy. The MDC Nokia dataset reproduces a more challenging and interesting scenario. In this case, the mobility of devices is not limited to a specific region, rather devices are free to roam in large area (refer to Figure 5.8). However devices meet only a small portion of the whole population (see Figure 5.7) and the number of contacts per hour is even smaller that Cambridge and MIT Reality scenarios. These aspects affect the Proactivity value, giving rise to a very slow increase during the simulation time. Also in this case, by the end of the simulation time, CORDIAL outperforms both s-Flooding and SIDEMAN. Hence, the strategy of CORDIAL for the diffusion of advertisements is effective both in scenarios highly connected (e.g. Cambridge) and in scenarios highly disconnected (e.g. MDC Nokia)



(a) Accuracy in Cambridge.

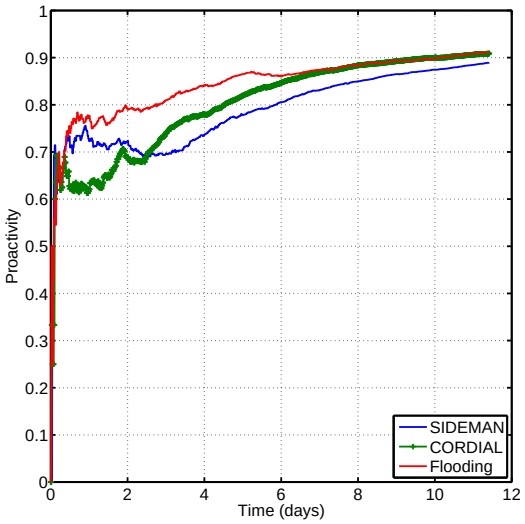


(b) Accuracy in MIT Reality.

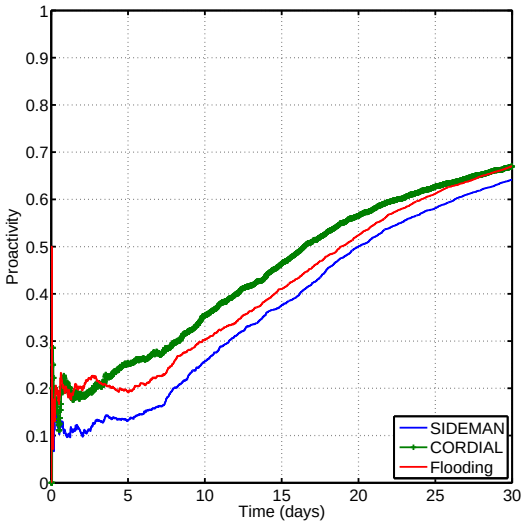


(c) Accuracy in MDC Nokia.

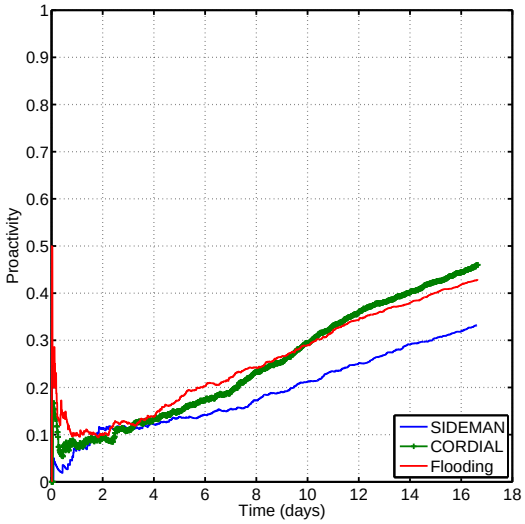
Figure 5.19: Accuracy metric in different simulation scenarios.



(a) Proactivity Cambridge.



(b) Proactivity MIT Reality.

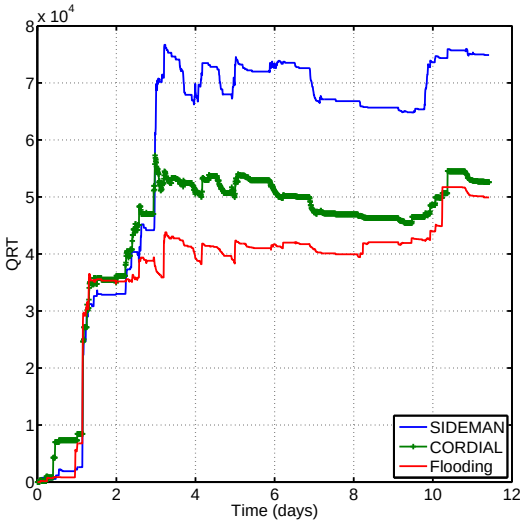


(c) Proactivity MDC Nokia.

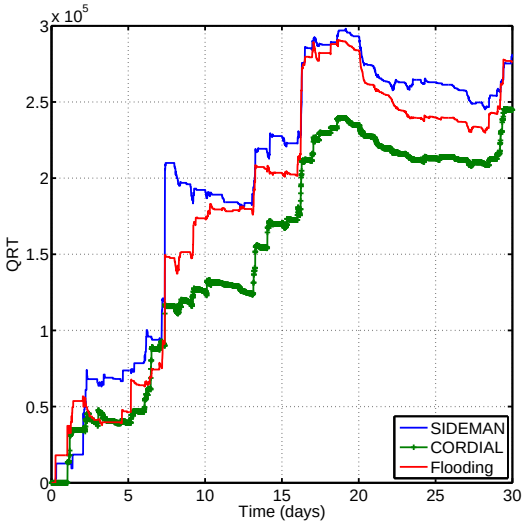
Figure 5.20: Proactivity metric in different simulation scenarios.

Query Response Time

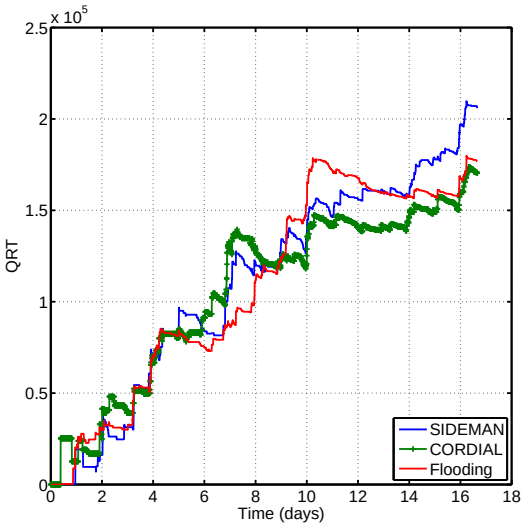
Results concerning the Query Response Time are shown in Figure 5.21. The trend for the three algorithms is the same: the QRT increases quickly during the warm-up period of the simulation after which the QRT value remains stable until the end of the simulation. Figure 5.21a shows the QRT value in the Cambridge scenario. As previously discussed (see consideration given for Figure 5.15a), the s-Flooding algorithm exchanges the highest number of advertisements among devices, hence devices have also the highest probability of answering to a query. However, the drawback of such strategy is an un-controlled diffusion of advertisements, giving rise to a value of Accuracy very low in every scenario, as discussed for Figure 5.19. CORDIAL



(a) QRT in Cambridge.



(b) QRT in MIT Reality.



(c) QRT in MDC Nokia.

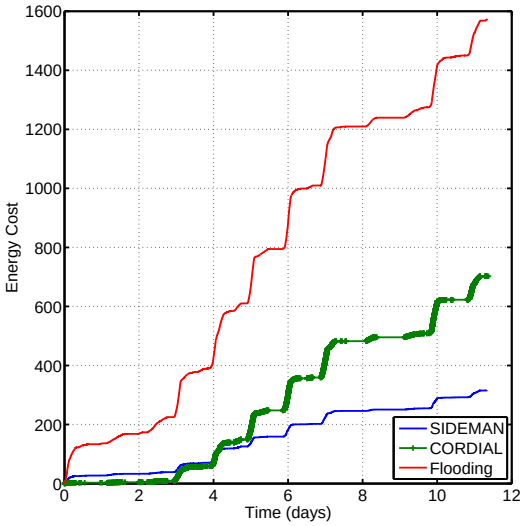
Figure 5.21: QRT metric in different simulation scenarios.

obtains a value of QRT never noticeable worst than that of the s-Flooding strategy (our benchmark), but with optimal values for both the Accuracy and of Proactivity metrics. The results of the QRT metric demonstrate the effectiveness of CORDIAL in controlling the diffusion of advertisement, without affecting negatively the responsiveness of the service discovery. When compared with SIDEMAN, CORDIAL has a QRT always far lower, this provides a further indication of the improvement of CORDIAL with respect to SIDEMAN. Such improvement is given by Algorithms 6 and 7 described in Section 4.4.2. Such algorithms exploit the remaining inter-contact time to forward an advertisement to devices that will *quickly* encounter the final destination, in turn such strategy affect positively the QRT value. Figures 5.21b and

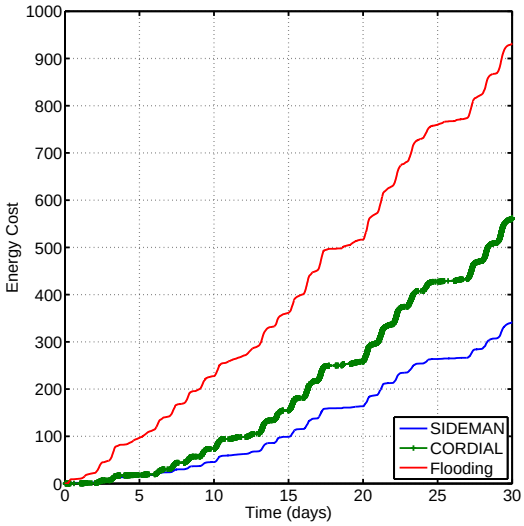
5.21c show the results of respectively the QRT with the MIT Reality and MDC Nokia scenarios. In both of the cases CORDIAL always outperforms SIDEMAN and, in the MIT Reality scenario, it outperforms SIDEMAN and also s-Flooding. In the MDC Nokia scenario the QRT value of CORDIAL is always comparable with respect to s-Flooding strategy and lower at the end of the simulation time.

Energy Cost

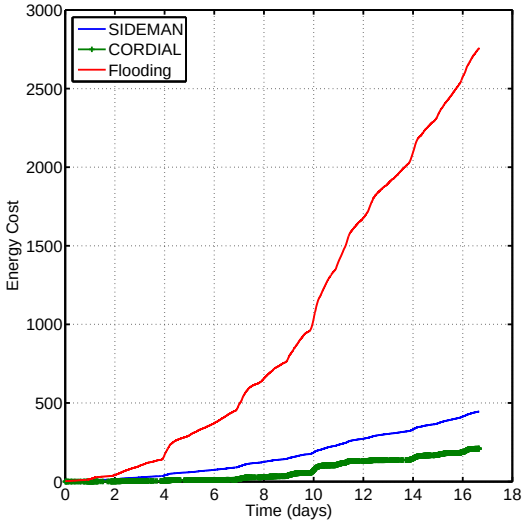
Results concerning the Energy Cost metric are shown in Figure 5.22. In order to



(a) EC in Cambridge.



(b) EC in MIT Reality.



(c) EC in MDC Nokia.

Figure 5.22: EC metric in different simulation scenarios.

estimate the energy consumption of the devices, we consider a standard battery pack

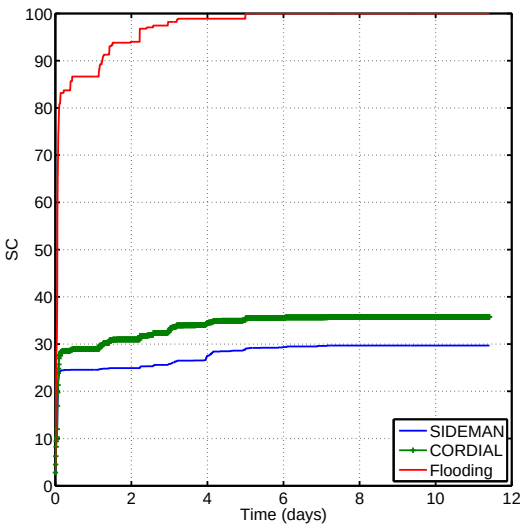
with an average consumption of 7.9Watt/hour (i.e., a battery pack with a capacity of 2100mAh and voltage of 3.8V, as done for the evaluation of SIDEMAN). The trend of the *EC* metric is similar for all the algorithms, *EC* increases as the time passes. In particular, by the end of the observation time both in Cambridge and in MIT Reality, the *EC* of CORDIAL is always sandwiched between s-Flooding (the upper bound) and the SIDEMAN curves. Flooding consumes 8.18%, CORDIAL consumes $2,64\%$ and SIDEMAN consumes 1.10% of the full battery charge. We note that the energy costs of SIDEMAN and CORDIAL do not affect significantly the battery charge. Similar considerations apply for the *EC* metric in the MIT Reality scenario, as shown in Figure 5.22b. With the MIT Reality scenario the *EC* s-Flooding consumes 11.79%, CORDIAL consumes $4,76\%$ and SIDEMAN consumes 1.26% of the battery pack.

The MDC Nokia scenario is an interesting case where it is possible to observe the benefits of the CORDIAL strategy on the *EC* metric. This scenario is characterized by few encounters among devices and very often with the same devices. Since the s-Flooding strategy does not implement any *smart* forwarding strategy for the diffusion advertisements, devices keep forwarding the same discovery messages (queries and advertisements) over and over again resulting with high battery depletion. By the end of the simulation time, s-Flooding spends 15.68% of the full battery charge. The forwarding strategy implemented with the SIDEMAN algorithm forwards discovery messages only to devices whose interests match with such messages. This strategy gives a good result in terms of battery depletion, in fact by the end of the observation time SIDEMAN consumes 4%. CORDIAL performs even better for two reasons: (1) the use of the social centrality metrics described with Equation 4.2 and (2) the avoidance of duplicate advertisements, which limits the number of discovery messages exchanged and hence reduces the energy cost. By the end of the simulation CORDIAL consumes 1.78% of the full battery charge.

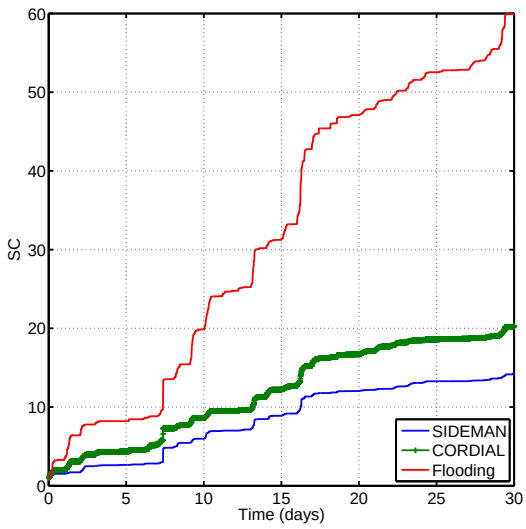
Service Cache

Results concerning the Service Cache are shown in Figure 5.23.

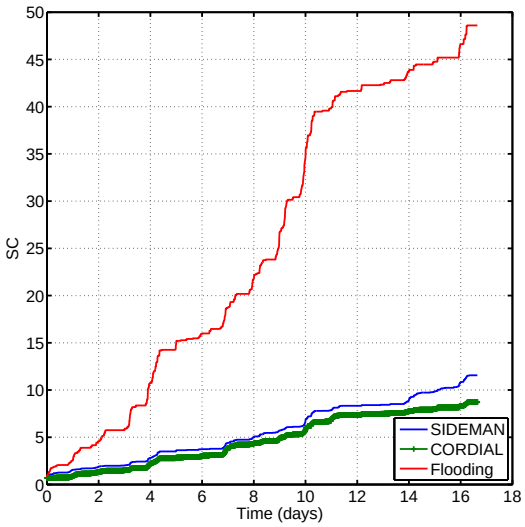
Figure 5.24a shows the SC in the Cambridge scenario. As expected, devices running the s-Flooding algorithm store the highest number of advertisements. After approximately 2 days of simulation, devices running the s-Flooding carry every advertisement available in the simulation. However, most of such advertisements are off topic for the device, as shown by the Accuracy metric in Figure 5.19a. We note that a s-Flooding requires an excessive storage capacity for a device, and this represents a non-negligible constraint for the application scenario to which we refer to. Differently, devices running SIDEMAN and CORDIAL control the number of advertisements stored their cache. More precisely, devices running SIDEMAN store an average of 30 advertisements while devices running CORDIAL store an average of 37 advertisements. CORDIAL requires a bit more storage capacity with respect to SIDEMAN because of the advertisement forwarding strategy (see Section 4.4.2).



(a) SC in Cambridge.



(b) SC in MIT Reality.



(c) SC in MDC Nokia.

Figure 5.23: *SC* metric in different simulation scenarios.

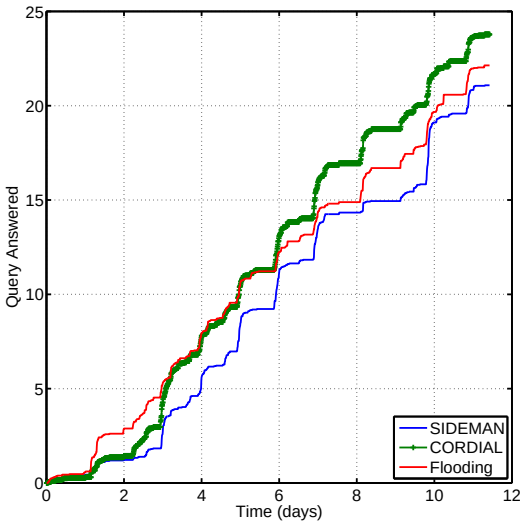
In particular, devices running CORDIAL use the forwarding cache $\mathcal{R}$ for carrying advertisements directed to a specific device (the final destination). Such strategy differs from SIDEMAN and it requires more storage capacity, in particular 23.3% more than that of SIDEMAN. However, the benefits deriving from the forwarding cache $\mathcal{R}$ are an increase of Proactivity, as shown in Figure 5.20b, and a notable reduction of the Query Response Time as shown in Figure 5.21.

Results concerning the Service Cache metric in the MIT Reality and MDC Nokia scenario are shown in Figure 5.24b and 5.24c. The trend of the algorithms is similar to the ones presented for the Cambridge scenario, but with a lower value of the number of advertisements stored. In particular, by the end of the observation time,

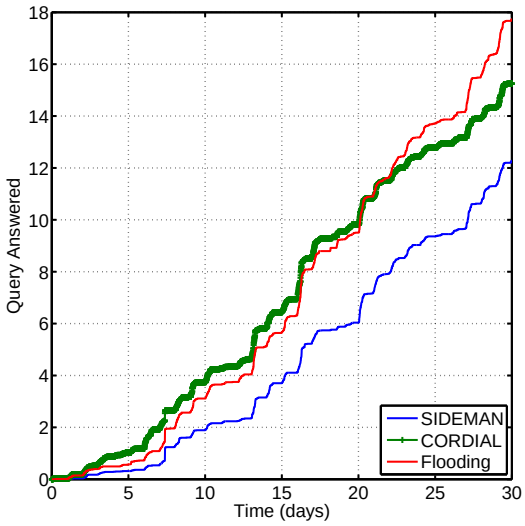
devices running s-Flooding store an average of 60 advertisements, SIDEMAN 15 and CORDIAL 20 (33% more SIDEMAN). With the MDC Nokia scenario, the low number of encounters and the low dimension of communities affect the total number of advertisements exchanged. By the end of the simulation time, devices running s-Flooding carry an average of 49 advertisements, SIDEMAN 12 and CORDIAL 9.

Query Answered

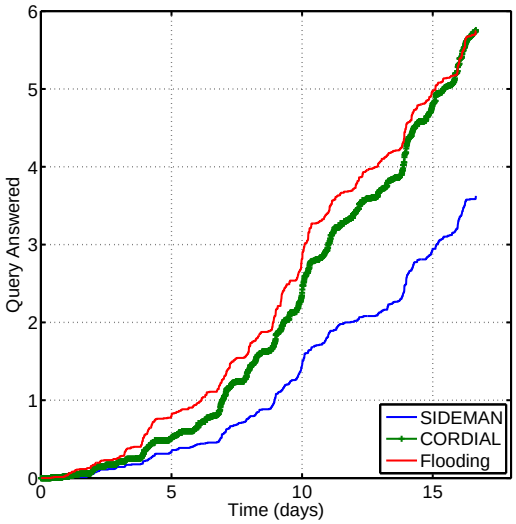
Results concerning the average number of queries answered are shown in Figure 5.24.



(a) QA in Cambridge.



(b) QA in MIT Reality.



(c) QA in MDC Nokia.

Figure 5.24: QA metric in different simulation scenarios.

The QA metrics grows along with the simulation time. In particular, in the Cambridge scenario CORDIAL obtains the highest value of the QA metric, meaning

that a device answers to a number of queries higher than that of devices running s-Flooding and $SIDEMAN$. With the MIT Reality and MDC Nokia scenarios, CORDIAL outperforms SIDEMAN and it obtains values of QA always comparable with the s-Flooding (our benchmark). Also in this case, the forwarding strategies implemented with CORDIAL for the diffusion of queries and advertisements are effective in terms of discovery performance.

Network Overhead

The last metric we analyze is the network overhead of the three algorithms, the results are shown in Figure 5.25.

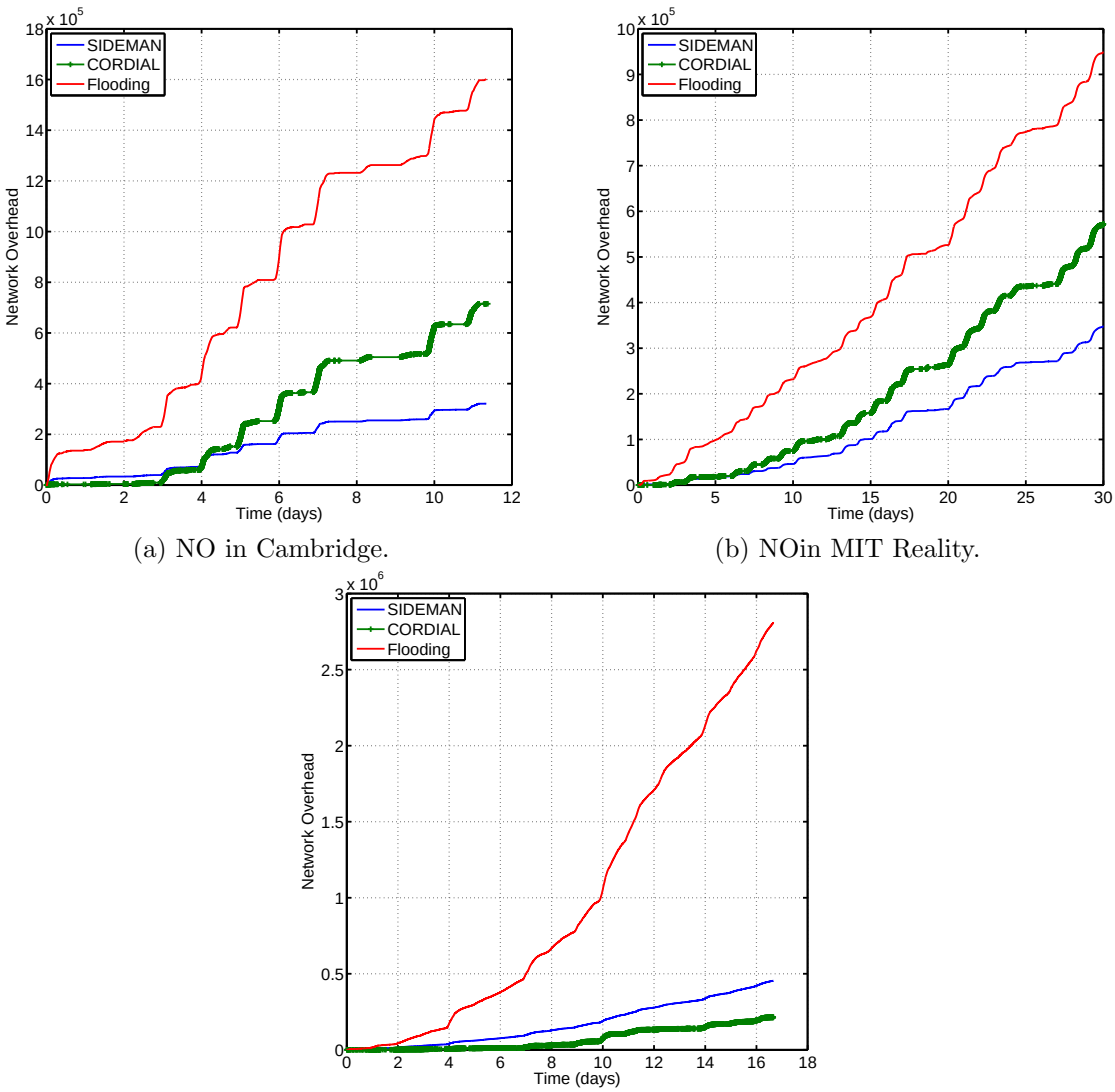
The trend of the NO metric is similar for all the algorithms and the considerations given for the EC metric apply also in this case. In particular, the network overhead increases as the time passes. By the end of the observation time both in Cambridge and in MIT Reality scenario, the NO of CORDIAL is always sandwiched between s-Flooding (the upper bound) and the SIDEMAN curves (the lower bound). In these two scenarios, the s-Flooding strategy is the most expensive one in terms of messages exchanged, while CORDIAL incurs in a network overhead slightly higher than that our first discovery algorithm.

The MDC Nokia scenario is an interesting case in which the benefits of CORDIAL are evident. This scenario is characterized by few encounters among devices and very often with the same devices. Since the s-Flooding strategy does not implement any *smart* forwarding strategy for the diffusion advertisements, devices keep forwarding the same discovery messages (queries and advertisements) over and over again resulting with high network overhead. The forwarding strategy implemented with the SIDEMAN algorithm forwards discovery messages only to devices whose interests match with such messages. This strategy gives a good result in terms of network overhead, however CORDIAL performs even better because:

- the use of the social centrality metrics allows to reduce the number of nodes to which forward a query without affecting the Proactivity metric (see Figure 5.20);
- the mechanism for avoiding duplicate of messages (e.g. advertisements) reduces the overall network overhead of the discovery protocol.

5.7 Summary

This thesis studies the service discovery problem in MSN in terms of how to discover services offered by mobile devices in SE. We study the service discovery in a specific kind of SE, namely Mobile Social Networks and we propose two algorithms named SIDEMAN and CORDIAL. The most important achievements of these algorithms presented are:



(a) NO in Cambridge.

(b) NOin MIT Reality.

(c) NO in MDC Nokia.

Figure 5.25: *NO* metric in different simulation scenarios.

- the use of both reactive and proactive discovery modes. The reactive mode enables a device to actively propagate a query in the network by avoiding flooding-based strategies, but by exploiting the sociality of people carrying the devices. The proactive mode enables a device to be notified passively of service advertisements from the devices that are encountered over time. The proactive mode aims to provide a device with the service advertisements that it is likely to need.

- The use of a community-based diffusion strategy for the propagation of advertisements and queries. Human nature tends to make people form groups of similar individuals, and such groups of people are commonly named com-

munities. SIDEMAN and CORDIAL detect the community to which a device belongs, so that the diffusion of advertisements and queries is achieved by interacting with similar individuals. SIDEMAN and CORDIAL also keep track of communities detected in the past in order to avoid detecting communities that have already been visited.

- CORDIAL extends the design of SIDEMAN by introducing a metric able to measure the capability of a device to answer a query either directly or indirectly. The direct answer is the simple case in which the advertisement matching with the query is already available in the device. The indirect answer is a more complex case in which the device receiving a query forwards it to other devices that can potentially answer the query. CORDIAL measures both cases with the social centrality metric, by assigning a score to each device encountered. This score is then used to select which device is the best candidate to forward a query to.

This thesis presents a preliminary study concerning human mobility. We study some important metrics such as the distribution of the inter-contact time, the number of hourly contacts and the distribution of the contact duration, in order to learn more about the foundation of the algorithms proposed. The algorithms proposed were tested both with real-world mobility traces and with traces obtained from a mobility model.

Finally, we also considered the possibility of studying a formal proof of the discovery algorithms proposed. However, the nature of the underlying network (delay tolerant, opportunistic and typically disconnected) prevents any direct proof of correctness. As a matter of a fact, such proofs in this research field require a probabilistic approach. In the specific case of CORDIAL, such probabilistic approach would have to take into account a large number of parameters (including human mobility, behavior of users and their interests), which becomes easily unmanageable unless reverting to very abstract probabilistic models. We believe this to be a very interesting topic of research that deserves studies, but it is beyond the scope of the thesis. In fact, the thesis focused on the characterization of the protocol in *realistic* conditions (using real mobility traces for example), an approach that is largely used by the research community in this area.

Chapter 6

Conclusions

The diffusion of pocket and wearable devices, the advances in device manufacturing and the use of open platforms are important steps towards Mark Weiser's vision of *providing intelligence to the surrounding environment*. However, the complexity in the implementation of this vision raises many issues. This thesis focuses on two of these issues, device interoperability and service discovery in MSN.

Device interoperability arises from the observation that devices deployed in SE can have very different hardware and software features. Some devices have powerful hardware/software capabilities that enable them to be easily integrated with each other. Conversely, other devices are designed for very specific tasks and are poor in terms of hardware/software resources. We focus on this second class of low-power devices, in particular on devices based on the ZigBee specification. Our first objective is the design and development of an inter-operable gateway that allows access to low-power ZigBee devices with simple interfaces. The gateway hides all the complexity concerning the ZigBee protocol and implements a plug-and-play mechanism so that the functionalities of the devices can be accessed by different technologies (e.g. UPnP, Bluetooth and REST web-services etc.).

The second problem concerns the heterogeneity of deployment and in particular how the service discovery problem is influenced by the mobility of devices roaming in SE. Mobile computing devices (such as smart phones, smart watches, wristbands) are being diffused even more rapidly than traditional PCs and workstations. Such devices offer an increasing number of untapped resources that can be considered as services to offer to other devices or to people in SE. Our aim is to study one main building block of this challenging scenario, i.e. how to advertise and query for services in SE, which is commonly known as the service discovery problem. Our second objective is therefore the design of a service discovery algorithm specifically designed for mobile devices that are carried or worn by people. We take into consideration some basic principles concerning human mobility in SE and how the sociality of people affects the mobility of devices. In turn, such considerations are exploited for the design of our service discovery algorithm.

6.1 Future Works

Our results do not solve all the problems concerning the heterogeneity of access and deployment in SE. Some barriers still need to be removed both in terms of device interoperability and service discovery.

Concerning device interoperability, our results highlight the need to study the problem as a whole, in order to merge the problem of how to access devices with how to discover and advertise the services offered by them. In the future, it may be possible to consider every pocket device in a Smart Environment (such as a smart phone or a smart watch), as a tiny gateway for accessing low-power sensors installed in the device itself, thus giving rise to a mobile gateway. The purpose of this gateway would be to access the sensors behind the gateway and to advertise the functionalities provided by the devices as services to other devices occasionally encountered. Thus the algorithms we have proposed, could be integrated with the ZB4O integration gateway in order to facilitate the convergence between sensing and opportunistic communications. ZB4O could also be enhanced by supporting not only low-power devices (such as ZigBee devices) but also other kinds of access protocols, such as Bluetooth or EnOcean. The challenge, in this case, is to define one single access model for all sensors and to apply it to heterogeneous sensing technologies.

Concerning the service discovery, we have identified some areas of investigation for the dissemination of discovery messages, namely multiple forwarding strategies, understanding human mobility, end-user selfishness and the construction of a formal proof of the discovery algorithms proposed in this thesis. Advertisements and queries are messages that require different diffusion strategies. In fact, as discussed in Section 2.2.3, queries should be distributed to devices that have the highest probability of finding an answer in a short time. Thus the diffusion of advertisement messages could exploit relay devices that might be interested in accessing the service or that easily meet other devices that are particularly interested in these messages. The selection of relay devices with such properties could become more accurate by using community detection algorithms that combine temporal, spatial and social attributes. To the best of our knowledge, the distributed community detection algorithms suitable for MSN only address one of these attributes, giving rise to communities that only partially reflect the complexity of human behaviors. In fact, algorithms that exploit only temporal or spatial metrics tend to bring together people into communities that meet frequently or who often visit the same places. However, disregarding weak temporal or spatial correlations among people, these algorithms may fail to connect devices carried by strangers (i.e. users that have social ties that are too weak to enter a community) which instead may act as bridges among communities [86]. By means of more accurate community detection algorithms, a device that is ready to distribute a service query or an advertisement can better assess the potential of other devices as relays for its messages.

A second important aspect is how to exploit human mobility in the diffusion of

queries and advertisements. Here the key is how to predict human mobility based on past encounters with other people, and how to exploit this prediction in the dissemination of discovery and advertisement messages. If a device k finds that h might answer a query with a high probability, then k may analyze its past encounters with h in order to predict when it will meet h again, and thus decide whether to wait for a meeting with h or to choose another relay device. Some recent works address the problem of understating human mobility patterns [128, 129], however these results are not integrated in the service discovery loop.

Third, the selfishness of individuals taking part in the discovery process is a natural but limiting feature of an MSN. Most of the works surveyed in this thesis assume that all devices in an MSN collaborate with the service discovery process. However, in real scenarios, this assumption is a concrete barrier. People are skeptical with respect to unknown mobile applications that may cause an uncontrolled battery depletion, and that take autonomous decisions. People may also not agree to reveal personal interests and personal habits in order to detect communities or to propagate a query to a similar person. Thus, the service discovery process needs to meet additional requirements such as energy awareness and privacy/security concerns. The strategy to select the target device to which to forward a discovery message can be enhanced with the awareness of the energy consumption of the discovery process and by introducing a reward mechanism. In addition, devices with a low battery level or devices with scarce computational resources could be temporarily excluded from the service discovery.

Lastly, we plan to extend the work done so far with a formal proof of the discovery algorithms proposed in this thesis. In particular we plan to draw an analytic study of the diffusion strategies of queries and advertisements of SIDEMAN and CORDIAL as well as a probabilistic model of some of the evaluation metric proposed in Section 5.4. We consider that having a probabilistic model for some of the metrics proposed is helpful to evaluate the benefits of the algorithms proposed without running any new simulations.

Bibliography

[1] M. Weiser, "Human-computer interaction," ch. The computer for the 21st century, pp. 933–940, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995.

[2] M. Satyanarayanan, "Pervasive computing: vision and challenges," *Personal Communications, IEEE*, vol. 8, pp. 10 –17, aug 2001.

[3] D. J. Cook and S. K. Das, "How smart are our environments? an updated look at the state of the art," *Pervasive Mob. Comput.*, vol. 3, pp. 53–73, Mar. 2007.

[4] S. Helal and C. Chen, "The gator tech smart house: enabling technologies and lessons learned," in *Proceedings of the 3rd International Convention on Rehabilitation Engineering & Assistive Technology*, i-CREATe '09, (New York, NY, USA), pp. 13:1–13:4, ACM, 2009.

[5] M. Naphade, G. Banavar, C. Harrison, J. Paraszczak, and R. Morris, "Smarter cities and their innovation challenges," *Computer*, vol. 44, pp. 32 –39, june 2011.

[6] R. Murphy, T. Sterling, and C. Dekate, "Advanced architectures and execution models to support green computing," *Computing in Science Engineering*, vol. 12, pp. 38–47, november 2010.

[7] C. Ververidis and G. Polyzos, "Service discovery for mobile ad hoc networks: a survey of issues and techniques," *IEEE Communications Surveys & Tutorials*, vol. 10, no. 3, pp. 30–45, 2008.

[8] Apple, Inc., "Bonjour technology white paper," 2007.

[9] E. Guttman, "Service location protocol version 2," 1999. IETF RFC 2608.

[10] P. Bellavista, R. Montanari, and S. K. Das, "Mobile social networking middleware: A survey," *Pervasive and Mobile Computing*, vol. 9, no. 4, pp. 437–453, 2013.

[11] N. Vastardis and K. Yang, "Mobile social networks: Architectures, social properties, and key research challenges," *Communications Surveys Tutorials, IEEE*, vol. 15, pp. 1355–1371, March 2013.

[12] K. Gill, S.-H. Yang, F. Yao, and X. Lu, "A zigbee-based home automation system," *Consumer Electronics, IEEE Transactions on*, vol. 55, pp. 422 –430, May 2009.

[13] Z. Alliance, "The zigbee specification, ver. 1.0," 2005.

[14] W. S. Lee and S. H. Hong, "Implementation of a knx-zigbee gateway for home automation," in *Consumer Electronics, 2009. ISCE '09. IEEE 13th International Symposium on*, pp. 545 –549, May 2009.

[15] Y.-G. Ha, "Dynamic integration of zigbee home networks into home gateways using osgi service registry," *Consumer Electronics, IEEE Transactions on*, vol. 55, pp. 470 –476, May 2009.

[16] M. McPherson, L. S. Lovin, and J. M. Cook, "Birds of a Feather: Homophily in Social Networks," *Annual Review of Sociology*, vol. 27, no. 1, pp. 415–444, 2001.

[17] M. Girolami, S. Lenzi, F. Furfari, and S. Chessa, "Sail: A sensor abstraction and integration layer for context awareness," in *Software Engineering and Advanced Applications, 2008. SEAA '08. 34th Euromicro Conference*, pp. 374–381, September 2008.

[18] M. Girolami, S. Chessa, and A. Caruso, "On service discovery in mobile social networks: Survey and perspectives," *Computer Networks*, vol. 88, pp. 51 – 71, 2015.

[19] P. Cassara', F. Potorti', P. Barsocchi, and M. Girolami, "Choosing an RSS Device-Free localization algorithm for ambient assisted living," in *IPIN 2015 Sixth International Conference on Indoor Positioning and Indoor Navigation (IPIN 2015)*, (Banff, Canada), Oct. 2015.

[20] P. Cassara', F. Potorti', P. Barsocchi, M. Girolami, and P. Nepa, "Lessons learned on device free localization with single and multi channel mode," in *IPIN 2015 Sixth International Conference on Indoor Positioning and Indoor Navigation (IPIN 2015)*, (Banff, Canada), Oct. 2015.

[21] F. Furfari, M. Girolami, S. Lenzi, and S. Chessa, "A service-oriented zigbee gateway for smart environments," *Journal of Ambient Intelligence and Smart Environments*, vol. 6, no. 6, pp. 691–705, 2014.

[22] M. Girolami, F. Furfari, and S. Chessa, "An integration gateway for sensing devices in smart environments," *ERCIM News*, vol. 2015, no. 101, 2015.

[23] M. Girolami, F. Palumbo, F. Furfari, and S. Chessa, "The integration of zigbee with the giraffplus robotic framework," *Communications in Computer and Information Science*, vol. 413 CCIS, pp. 86–101, 2013.

[24] E. Ferro, M. Girolami, D. Salvi, C. Mayer, J. Gorman, A. Grguric, R. Ram, R. Sadat, K. M. Giannoutakis, and C. Stocklöw, "The universaal platform for aal (ambient assisted living)," *Journal of Intelligent Systems*, 2015.

[25] C. Stocklw, A. M. Medrano Gil, A. Fides Valero, M. Girolami, and S. Lenzi, "Multi-tenancy aware ambient assisted living platforms in the cloud," in *Ambient Intelligence* (E. Aarts, B. de Ruyter, P. Markopoulos, E. van Loenen, R. Wichert, B. Schouten, J. Terken, R. Van Kranenburg, E. Den Ouden, and G. O'Hare, eds.), vol. 8850 of *Lecture Notes in Computer Science*, pp. 80–95, Springer International Publishing, 2014.

[26] L. M. Broberg and M. Girolami, "A common platform for aal services and a common future the universaal project," pp. 135 – 155, February 2013.

[27] R. Ram, F. Furfari, M. Girolami, G. Ibaez-Snchez, J.-P. Lzaro-Ramos, C. Mayer, B. Prazak-Aram, and T. Zentek, "universaal: Provisioning platform for aal services," in *Ambient Intelligence - Software and Applications*, vol. 219 of *Advances in Intelligent Systems and Computing*, pp. 105–112, Springer International Publishing, 2013.

[28] M. Girolami, F. Furfari, and S. Chessa, "A cost-based model for service discovery in smart environments," in *Ambient Intelligence* (F. Patern, B. de Ruyter, P. Markopoulos, C. Santoro, E. van Loenen, and K. Luyten, eds.), vol. 7683 of *Lecture Notes in Computer Science*, pp. 397–402, Springer Berlin Heidelberg, 2012.

[29] M. Girolami, P. Barsocchi, S. Chessa, and F. Furfari, "A social-based service discovery protocol for mobile ad hoc networks," in *Proceedings of IEEE Med-Hoc-Net 2013*, pp. 53–60, June 24–26 2013.

[30] M. Girolami, S. Chessa, S. Basagni, and F. Furfari, "Service discovery in mobile social networks," in *Personal, Indoor, and Mobile Radio Communication (PIMRC), 2014 IEEE 25th Annual International Symposium on*, pp. 1464–1468, Sept 2014.

[31] M. Girolami, S. Chessa, and E. Ferro, "Discovery of services in smart cities of mobile social users," in *Fifth International Workshop on Management of Cloud and Smart City Systems 2015 (MoCS 2015)*, (Larnaca, Cyprus), July 2015.

[32] M. Girolami, S. Chessa, L. Foschini, R. Ianniello, and A. Corradi, "Social amplification factor for mobile crowd sensing: The ParticipAct experience," in *20th IEEE Symposium on Computers and Communications (ISCC2015)*, (Larnaca, Cyprus), July 2015.

[33] M. Girolami, S. Basagni, F. Furfari, and S. Chessa, "Sideman: Service discovery in mobile social networks," *Ad Hoc and Sensor Wireless Network*, 2015, to-appear.

[34] P. Baronti, P. Pillai, V. W. C. Chook, S. Chessa, A. Gotta, and Y. F. Hu, "Wireless sensor networks: A survey on the state of the art and the 802.15.4 and zigbee standards," *Comput. Commun.*, vol. 30, pp. 1655–1695, May 2007.

[35] O. Alliance, "About the osgi service platform," tech. rep., OSGi Alliance, June 2007.

[36] C. Perkins and E. Royer, "Ad-hoc on-demand distance vector routing," in *Mobile Computing Systems and Applications, 1999. Proceedings. WMCSA '99. Second IEEE Workshop on*, pp. 90–100, February 1999.

[37] Z. Alliance, "Zigbee cluster library specification," tech. rep., ZigBee Alliance, May 2008.

[38] O. Alliance, "Osgi service platform release 4, version 4.1," tech. rep., OSGi Alliance, May 2007.

[39] O. Dohndorf, J. Kruger, H. Krumm, C. Fiehe, A. Litvina, I. Luck, and F.-J. Stewing, "Towards the web of things: Using dpws to bridge isolated osgi platforms," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on*, pp. 720–725, March 2010.

[40] S. Coradeschi, A. Cesta, G. Cortellessa, L. Coraci, J. Gonzalez, L. Karlsson, F. Furfari, A. Loutfi, A. Orlandini, F. Palumbo, F. Pecora, S. von Rump, A. Stimec, J. Ullberg, and B. Otslund, "Giraffplus: Combining social interaction and long term monitoring for promoting independent living," in *Human System Interaction (HSI), 2013 The 6th International Conference on*, pp. 578–585, June 2013.

[41] A. Kristoffersson, S. Coradeschi, and A. Loutfi, "A review of mobile robotic telepresence," *Advances in Human-Computer Interaction*, vol. 2013, 2013.

[42] M. Corporation, "Universal plug and play: Background," 1999. http://www.upnp.org/resources/UPnPbkgnd.htm.

[43] M. Athanasopoulos and K. Kontogiannis, "Extracting rest resource models from procedure-oriented service interfaces," *Journal of Systems and Software*, vol. 100, pp. 149–166, 2015.

[44] M. Chen and C. Wu, "A zigbee-based home control system using osgi management platform," *International Journal of Smart Home*, vol. 6, no. 4, pp. 15–28, 2012.

[45] Z. Alliance, "Understanding zigbee gateway," 2011.

[46] R. Kawamoto, T. Emori, S. Sakata, K. Furuhata, K. Yuasa, and S. Hara, "Dlna-zigbee gateway architecture and energy efficient sensor control for home networks," in *Mobile and Wireless Communications Summit, 2007. 16th IST*, pp. 1 –5, July 2007.

[47] S. H. Kim, J. S. Kang, H. S. Park, D. Kim, and Y. joo Kim, "Upnp-zigbee internetworking architecture mirroring a multi-hop zigbee network topology," *Consumer Electronics, IEEE Transactions on*, vol. 55, pp. 1286–1294, August 2009.

[48] G. Hu, "Design and implementation of industrial wireless gateway based on zigbee communication," in *Electronic Measurement Instruments, 2009. ICEMI '09. 9th International Conference on*, pp. 1–684 –1–688, August 2009.

[49] P. Qiu, U. Heo, and J. Choi, "The web-sensor gateway architecture for zigbee," in *Consumer Electronics, 2009. ISCE '09. IEEE 13th International Symposium on*, pp. 661 –664, May 2009.

[50] G. De Silva, L. De Silva, P. Ishara, M. Kumara, and T. Ginige, "Smartbee; multi-channel access zigbee gateway with plug and play device interface for smart home/office automation," in *Information and Automation for Sustainability, 2008. ICIAFS 2008. 4th International Conference on*, pp. 251 –256, December 2008.

[51] G. Bigwood, D. Rehunathan, M. Bateman, T. Henderson, and S. Bhatti, "Exploiting self-reported social networks for routing in ubiquitous computing environments," in *Networking and Communications, 2008. WIMOB '08. IEEE International Conference on Wireless and Mobile Computing,*, pp. 484–489, October 2008.

[52] M. Conti and M. Kumar, "Opportunities in opportunistic computing," *Computer*, vol. 43, pp. 42–50, January 2010.

[53] D. Wang, D. Pedreschi, C. Song, F. Giannotti, and A.-L. Barabasi, "Human mobility, social ties, and link prediction," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, (New York, NY, USA), pp. 1100–1108, ACM, 2011.

[54] M. Girvan and M. E. Newman, "Community structure in social and biological networks," *Proceedings of the National Academy of Sciences*, vol. 99, no. 12, pp. 7821–7826, 2002.

[55] P. Dhakan and R. Menezes, "The role of social structures in mobile ad-hoc networks," in *Proceedings of the 43rd Annual Southeast Regional Conference - Volume 2*, ACM-SE 43, (New York, NY, USA), pp. 59–64, ACM, 2005.

[56] M. Newman, "Detecting community structure in networks," *The European Physical Journal B - Condensed Matter and Complex Systems*, vol. 38, no. 2, pp. 321–330, 2004.

[57] M. E. J. Newman, "Analysis of weighted networks," *Phys. Rev. E*, vol. 70, p. 056131, Nov. 2004.

[58] A. Clauset, "Finding local community structure in networks," *Phys. Rev. E*, vol. 72, p. 026132, Aug 2005.

[59] E. Borgia, M. Conti, and A. Passarella, "Autonomic detection of dynamic social communities in opportunistic networks," in *Proceedings of IEEE Med-Hoc-Net 2011*, (Favignana, Italy), pp. 142–149, June 12–15 2011.

[60] P. Hui, E. Yoneki, S. Y. Chan, and J. Crowcroft, "Distributed community detection in delay tolerant networks," in *Proceedings of 2Nd ACM/IEEE International Workshop on Mobility in the Evolving Internet Architecture*, MobiArch '07, (New York, NY, USA), pp. 7–14, ACM, 2007.

[61] M. Orlinski and N. Filer, "The rise and fall of spatio-temporal clusters in mobile ad hoc networks," *Ad Hoc Networks*, vol. 11, pp. 1641–1654, July 2013.

[62] F. Li and J. Wu, "LocalCom: A community-based epidemic forwarding scheme in disruption-tolerant networks," in *2009 6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, SECON 2009*, 2009.

[63] A. Vahdat and D. Becker, "Epidemic routing for partially connected ad hoc networks," tech. rep., 2000.

[64] C. Campo, M. Munoz, J. C. Perea, A. Marin, and C. Garcia-Rubio, "PDP and GSDL: A new service discovery middleware to support spontaneous interactions in pervasive systems," in *Proceedings of the IEEE Percom Workshops 2005*, pp. 178–182, 2005.

[65] S. M. Allen, G. Colombo, and R. M. Whitaker, "Uttering: Social micro-blogging without the internet," in *Proceedings of the Second International Workshop on Mobile Opportunistic Networking*, MobiOpp '10, (New York, NY, USA), pp. 58–64, ACM, 2010.

[66] I. Konstas, V. Stathopoulos, and J. M. Jose, "On social networks and collaborative recommendation," in *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, (New York, NY, USA), pp. 195–202, ACM, 2009.

[67] J. Lee, C. Shao, H. Roh, and W. Lee, "Price-based tethering for cooperative networking," in *Information Networking (ICOIN), 2013 International Conference on*, pp. 379–384, January 2013.

[68] G. Cardone, L. Foschini, P. Bellavista, A. Corradi, C. Borcea, M. Talasila, and R. Curtmola, "Fostering participaction in smart cities: a geo-social crowdsensing platform," *Communications Magazine, IEEE*, vol. 51, pp. 112–119, June 2013.

[69] D. Oberle, A. Barros, U. Kylau, and S. Heinzl, "A unified description language for human-to-automated services," *Information Systems*, vol. 38, pp. 155–181, March 2013.

[70] M. Pantazoglou and A. Tsalgatidou, "A generic query model for the unified discovery of heterogeneous services," *Services Computing, IEEE Transactions on*, vol. 6, pp. 201–213, April 2013.

[71] A. Nedos, K. Singh, R. Cunningham, and S. Clarke, "Probabilistic discovery of semantically diverse content in manets," *Mobile Computing, IEEE Transactions on*, vol. 8, pp. 544–557, April 2009.

[72] C. Cho and D. Lee, "Survey of service discovery architectures for mobile ad hoc networks. term paper," *Department of Computer and Information Science and Engineering (CICE), University of Florida, Fall*, vol. 5531, 2005.

[73] A. B. O. Sullivan, R. W. Sheifler, J. Waldo, and A. Wollrath, "The jini specification." Sun Microsystems, Inc.

[74] F. Zhu, M. Mutka, and L. Ni, "Splendor: A secure, private, and location-aware service discovery protocol supporting mobile services," in *Pervasive Computing and Communications, 2003. (PerCom 2003). Proceedings of the First IEEE International Conference on*, pp. 235–242, March 2003.

[75] M. Klein and B. Knig-Ries, "Multi-layer clusters in ad-hoc networks an approach to service discovery," in *Web Engineering and Peer-to-Peer Computing* (E. Gregori,

L. Cherkasova, G. Cugola, F. Panzieri, and G. Picco, eds.), vol. 2376 of *Lecture Notes in Computer Science*, pp. 187–201, Springer Berlin Heidelberg, 2002.

[76] K. Seada and A. Helmy, "Rendezvous regions: a scalable architecture for service location and data-centric storage in large-scale wireless networks," in *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, p. 218, April 2004.

[77] U. Mohan, K. Almeroth, and E. Belding-Royer, "Scalable service discovery in mobile ad hoc networks," in *Networking 2004* (N. Mitrou, K. Kontovasilis, G. Rouskas, I. Iliadis, and L. Merakos, eds.), vol. 3042 of *Lecture Notes in Computer Science*, pp. 137–149, Springer Berlin Heidelberg, 2004.

[78] O. Ratsimor, D. Chakraborty, A. Joshi, and T. Finin, "Allia: Alliance-based service discovery for ad-hoc environments," in *Proceedings of the 2Nd International Workshop on Mobile Commerce*, WMC '02, (New York, NY, USA), pp. 1–9, ACM, 2002.

[79] S. Helal, N. Desai, V. Verma, and C. Lee, "Konark - a service discovery and delivery protocol for ad-hoc networks," in *Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE*, vol. 3, pp. 2107–2113 vol.3, March 2003.

[80] T. D. Nguyen and S. Rouvrais, "A socially inspired peer-to-peer resource discovery service for delay tolerant networks," in *Proceedings of the 2007 OTM Confederated International Conference on On the Move to Meaningful Internet Systems - Volume Part II*, OTM'07, (Berlin, Heidelberg), pp. 960–969, Springer-Verlag, 2007.

[81] P. Costa, C. Mascolo, M. Musolesi, and G. Picco, "Socially-aware routing for publish-subscribe in delay-tolerant mobile ad hoc networks," *Selected Areas in Communications, IEEE Journal on*, vol. 26, pp. 748–760, June 2008.

[82] C. Boldrini, M. Conti, and A. Passarella, "Contentplace: Social-aware data dissemination in opportunistic networks," in *Proceedings of ACM MSWiM 2008*, pp. 203–210, Octoberober 27–31 2008.

[83] A. Mei, G. Morabito, P. Santi, and J. Stefa, "Social-aware stateless forwarding in pocket switched networks," in *Infocom, 2011 Proceedings Ieee*, pp. 251–255, IEEE, 2011.

[84] S. Kosta, A. Mei, and J. Stefa, "Large-Scale Synthetic Social Mobile Networks with SWIM," *IEEE Transactions on Mobile Computing*, vol. 13, no. 1, pp. 116–129, 2014.

[85] J. Díaz, A. Marchetti-Spaccamela, D. Mitsche, P. Santi, and J. Stefa, "Socially-aware forwarding improves routing performance in pocket switched networks," in *Proceedings of ESA 2011*, pp. 723–735, 2011.

[86] E. Pagani, L. Valerio, and G. P. Rossi, "Weak social ties improve content delivery in behavior-aware opportunistic networks," *Ad Hoc Networks*, vol. 25, Part B, no. 0,

pp. 314 – 329, 2015. New Research Challenges in Mobile, Opportunistic and Delay-Tolerant Networks Energy-Aware Data Centers: Architecture, Infrastructure, and Communication.

[87] S. A. Al Ayyat, S. Aly, and K. A. Harras, "PIPeR: Impact of power-awareness on social-based opportunistic advertising," in *Proceedings of IEEE WCNC 2014*, Apr. 2014.

[88] U. Aguilera and D. Lpez-de Ipia, "A parameter-based service discovery protocol for mobile ad-hoc networks," in *Ad-hoc, Mobile, and Wireless Networks* (X.-Y. Li, S. Papavassiliou, and S. Ruehrup, eds.), vol. 7363 of *Lecture Notes in Computer Science*, pp. 274–287, Springer Berlin Heidelberg, 2012.

[89] Z. Wang, E. Bulut, and K. Boleslaw, "Service discovery for delay tolerant networks," in *GLOBECOM Workshops (GC Wkshps), 2010 IEEE*, pp. 136–141, IEEE, 2010.

[90] N. Le Sommer and Y. Mahéo, "OLFServ: an Opportunistic and Location-Aware Forwarding Protocol for Service Delivery in Disconnected MANETs," in *Fifth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (Ubicomm 2011)* (X. P. Services, ed.), (Lisbon, Portugal), pp. 115–122, Nov. 2011.

[91] M. Granovetter, "The Strength of Weak Ties," *The American Journal of Sociology*, vol. 78, no. 6, pp. 1360–1380, 1973.

[92] V. Erramilli, M. Crovella, A. Chaintreau, and C. Diot, "Delegation forwarding," in *Proceedings of ACM MobiHoc 2008*, pp. 251–260, 2008.

[93] V. Erramilli and M. Crovella, "Diversity of forwarding paths in pocket switched networks," in *in Proc. ACM IMC 07*, pp. 161–174, 2007.

[94] T. Spyropoulos, K. Psounis, and C. Raghavendra, "Efficient routing in intermittently connected mobile networks: The single-copy case," *Networking, IEEE/ACM Transactions on*, vol. 16, pp. 63–76, February 2008.

[95] E. M. Daly and M. Haahr, "Social network analysis for routing in disconnected delay-tolerant manets," in *Proceedings of the 8th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc '07, (New York, NY, USA), pp. 32–40, ACM, 2007.

[96] P. Hui, J. Crowcroft, and E. Yoneki, "BUBBLE Rap: Social-based forwarding in delay-tolerant networks," *IEEE Transactions on Mobile Computing*, vol. 10, pp. 1576–1589, November 2011.

[97] E. Yoneki, P. Hui, S. Chan, and J. Crowcroft, "A socio-aware overlay for publish/-subscribe communication in delay tolerant networks," in *Proceedings of the 10th ACM Symposium on Modeling, Analysis, and Simulation of Wireless and Mobile Systems*, MSWiM '07, (New York, NY, USA), pp. 225–234, ACM, 2007.

[98] K. C.-J. Lin, W.-T. Lin, and C.-F. Chou, "Social-based content diffusion in pocket switched networks," *IEEE Transactions on Vehicular Technology*, vol. 60, no. 9, pp. 4539–4548, 2011.

[99] C. Groba and S. Clarke, "Opportunistic service composition in dynamic ad hoc environments," *IEEE T. Services Computing*, vol. 7, no. 4, pp. 642–653, 2014.

[100] J. Kniess, O. Loques, and C. V. Albuquerque, "Location aware discovery service and selection protocol in cooperative mobile wireless ad hoc networks," in *INFOCOM Workshops 2009, IEEE*, pp. 1–2, IEEE, 2009.

[101] S. Cuddy, M. Katchabaw, and H. Lutfiyya, "Context-aware service selection based on dynamic and static service attributes," in *Wireless And Mobile Computing, Networking And Communications, 2005.(WiMob'2005), IEEE International Conference on*, vol. 4, pp. 13–20, IEEE, 2005.

[102] M. Conti, E. Marzini, D. Mascitti, A. Passarella, and L. Ricci, "Service selection and composition in opportunistic networks," in *Wireless Communications and Mobile Computing Conference (IWCMC), 2013 9th International*, pp. 1565–1572, July 2013.

[103] A. Varshavsky, B. Reid, and E. de Lara, "A cross-layer approach to service discovery and selection in manets," in *Mobile Adhoc and Sensor Systems Conference, 2005. IEEE International Conference on*, pp. 8 pp.–466, Nov 2005.

[104] J. Kniess, O. Loques, and C. V. Albuquerque, "Service discovery with time constraints in mobile ad hoc networks," *Earth Science Informatics*, pp. 1–14, 2014.

[105] N. Santoro and I. Stojmenovic, "Localized Distance-Sensitive Service Discovery in Wireless Sensor and Actor Networks," *IEEE Transactions on Computers*, vol. 58, pp. 1275–1288, Sept. 2009.

[106] N. Surobhi and A. Jamalipour, "A context-aware m2m-based middleware for service selection in mobile ad-hoc networks," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 25, pp. 3056–3065, Dec 2014.

[107] A. Makke, Y. Mahéo, N. Le Sommer, *et al.*, "Towards opportunistic service provisioning in intermittently connected hybrid networks," in *Proceedings of the 4th International Conference on Networking and Distributed Computing (ICNDC 2013)*, 2013.

[108] Y. Mahéo and R. Said, "Service invocation over content-based communication in disconnected mobile ad hoc networks," in *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*, pp. 503–510, IEEE, 2010.

[109] S. Ioannidis, A. Chaintreau, and L. Massoulié, "Optimal and scalable distribution of content updates over a mobile social network," in *INFOCOM 2009, IEEE*, pp. 1422–1430, IEEE, 2009.

[110] W. jen Hsu, D. Dutta, and A. Helmy, "Csi: A paradigm for behavior-oriented profile-cast services in mobile networks," *Ad Hoc Networks*, vol. 10, no. 8, pp. 1586 – 1602, 2012. Special Issue on Social-Based Routing in Mobile and Delay-Tolerant Networks.

[111] F. Furfari and S. Lenzi, "The felix upnp documentation, technical report tr-02," tech. rep., ISTI-CNR, 2008.

[112] Y. Wang, A. Vasilakos, Q. Jin, and J. Ma, "Survey on mobile social networking in proximity (msnp): approaches, challenges and architecture," *Wireless Networks*, vol. 20, no. 6, pp. 1295–1311, 2014.

[113] N. Jabeur, Z. S, and S. B, "Mobile Social Networking Applications.," *Communications of the ACM*, vol. 56, pp. 71–79, 2013.

[114] P. Hui, *People are the Network: Experimental Design and Evaluation of Social-based Forwarding Algorithms*. PhD thesis, University of Cambridge, 2007. Published as UCAM Computer Laboratory technical report no. 713.

[115] N. H. Sulaiman and M. Daud, "A Jaccard-based similarity measure for soft sets," in *Proceedings of IEEE SHUSER 2012*, pp. 659–663, 2012.

[116] S. Tarkoma, C. Esteve Rothenberg, and E. Lagerspetz, "Theory and practice of Bloom filters for distributed systems," *IEEE Communications Surveys & Tutorials*, pp. 131–155, 2012.

[117] T. Henderson, D. Kotz, and I. Abyzov, "The changing usage of a mature campus-wide wireless network," *Computer Networks*, vol. 52, no. 14, pp. 2690 – 2712, 2008.

[118] J. Scott, R. Gass, J. Crowcroft, P. Hui, C. Diot, and A. Chaintreau, "CRAWDAD data set cambridge/haggle (v. 2006-01-31)." Downloaded from http://crawdad.org/cambridge/haggle/, 2006.

[119] N. Eagle and A. (Sandy) Pentland, "Reality mining: Sensing complex social systems," *Personal Ubiquitous Comput.*, vol. 10, pp. 255–268, Mar. 2006.

[120] J. K. Laurila, J. Blom, O. Dousse, D. Gatica-perez, O. Bornet, J. Eberle, I. Aad, and M. Miettinen, "The mobile data challenge: Big data for mobile computing research," in *Proceedings of Mobile Data Challenge Workshop (MDC)*, 2012.

[121] C. Boldrini and A. Passarella, "HCMM: Modelling spatial and temporal properties of human mobility driven by users social relationships," *Computer Communications*, vol. 33, no. 9, pp. 1056–1074, 2010.

[122] H. Lim and C. Kim, "Flooding in wireless ad hoc networks," *Computer Communications*, vol. 24, no. 3–4, pp. 353 – 363, 2001.

[123] H. C. Li, A. Clement, E. L. Wong, J. Napper, I. Roy, L. Alvisi, and M. Dahlin, "BAR gossip," in *Proceedings of USENIX OSDI 2006*, pp. 14–14, 2006.

[124] P. Sermpezis and T. Spyropoulos, "Not all content is created equal: Effect of popularity and availability for content-centric opportunistic networking," in *Proceedings of the 15th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc '14, (New York, NY, USA), pp. 103–112, ACM, 2014.

[125] C. Boldrini, M. Conti, and A. Passarella, "Context and resource awareness in opportunistic network data dissemination," in *Proceedings of IEEE WoWMoM 2008*, (Newport Beach, CA), pp. 1–6, June 23–26 2008.

[126] C. Liu and J. Wu, "Routing in a cyclic mobispace," in *Proceedings of ACM MobiHoc 2008*, pp. 351–360, 2008.

[127] A. Keränen, J. Ott, and T. Kärkkäinen, "The ONE Simulator for DTN Protocol Evaluation," in *SIMUTools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, (New York, NY, USA), ICST, 2009.

[128] M. D. Domenico, A. Lima, and M. Musolesi, "Interdependence and predictability of human mobility and social interactions," *Pervasive and Mobile Computing*, vol. 9, no. 6, pp. 798 – 807, 2013. Mobile Data Challenge.

[129] T. M. T. Do, O. Dousse, M. Miettinen, and D. Gatica-Perez, "A probabilistic kernel method for human mobility prediction with smartphones," *Pervasive and Mobile Computing*, vol. 20, pp. 13 – 28, 2015.