Department of Electronic and Telecommunications, University of Florence, Italy

Ph.D. Thesis - XXI Cycle - DIIMT

Systems and Solutions for Mobile Pervasive Computing Environments

Ph.D. Candidate

David Tacconi

Advisors

Prof. Romano Fantacci

Ing. Laura Pierucci

Prof. Imrich Chlamtac

December 2008

To Francesca and Cecilia

ACKNOWLEDGEMENTS

During this three years, I have been considering my self a privileged for having the opportunity of doing research. Research, and this is particularly true during a PhD, is a work where humans can express all their creativity and talent, since there is no limit to what you can realize from one day to another and every moment and occasion is an opportunity to think about a new issue to be addressed or a solution to a known problem.

From one side this means that your work becomes strongly integrated with your private life. For this reason, I have to thank first of all my wife Francesca, since she has always encouraged me to go on and to face all the difficulties, with love and with a lot of patience. Moreover, she has married me at the beginning of these three years and we have had our wonderful daughter Cecilia just few months ago. Then, I would like to thank some of my numerous friends in Trento that have shared this adventure with me: Gabriella, Giuseppe, Fabio, Francesco, Stefano, Giorgio, Maria, Alessandro, Francesca, Matteo, Alessandra, Lorenzo and many others. Last but not least, I want to thank my parents Paolo and Marzia for all the opportunities they have given me in these years and my sister Chiara and her family for their lovely support.

At the same time, when you are new to research as I was three years ago, you really need somebody to follow, somebody you can share your ideas with, somebody who, step by step, makes you find the right way. Luckily, I found these persons in Daniele, Francesco and Iacopo, my colleagues at Create-Net research center in the pervasive computing area. I have been staying with them every day, learning more than I have ever learned before, at first listening to their discussions without saying a word and then starting to share my ideas and thoughts. Furthermore, I would like to thank all the people in Create-Net and in particular: Oscar, Hagen, Roberto, Tinku, Antonio, Cristina, Federico, Paolo, Anna, Sabrina, Alessandro, Elio and all the others who have shared these three years with me.

A special thanks goes to my advisor at the University of Florence professor Romano Fantacci for letting me enter the research world a second time and for all the stimulating inputs he has provided me during this PhD. I would like to thank also dr. Francesco Chiti for his precious contribution in my research and all the people at the “Lart” lab of the University of Florence.

Finally, I would like to thank Professor Imrich Chlamtac for hosting me at Create-Net research center and for having introduced and guided me into research in a more complete and wise way.

Abstract

Mobile pervasive computing and communication environments are characterized by an extremely large number of networked embedded devices and by mobile nodes gathering information from these embedded sensing devices. Such environments are receiving a significant attention from the research community, given the high proliferation of both embedded sensing and mobile devices.

In this thesis work, it is assumed a mobile pervasive computing environment, where nodes do not communicate with a central server. A system architecture is presented, with three different types of nodes: (i) mobile nodes (ii) gateway nodes (iii) sensing nodes. Sensing nodes communicate among them via multi-hop communications as in traditional Wireless Sensor Networks (WSN). Mobile nodes communicate through one hop peer to peer communication, and the information gathered from sensing nodes is diffused in an opportunistic fashion. Gateway nodes represent the interface among mobile nodes and sensing nodes and do not communicate among them in any occasion.

Given this system architecture, the application scenarios of Intelligent Transportation Systems (ITS) and Mobile Social Networks (MSN) are considered with the relative challenges. At first, we design and evaluate a routing framework for the ITS scenario able to deal with sinks querying a network while moving (e.g. cars), in order to retrieve information from a large WSN (e.g. providing information about parking places). Furthermore, we address the challenges related to management of data at the mobile nodes layer. We assume in fact that mobile nodes gather data from gateway nodes and exchange information among them at every meeting, so that the amount of data stored in each device and exchanged need to be controlled in order to optimize the scarce resources of mobile environments.

In the scenario of MSNs, it is initially presented a real implementation of the proposed system architecture for Bluetooth enabled java smartphones. By using such implementation, a real experimentation is performed, where mobile users exchange information through the developed application at every meeting. Users are asked to compile a personal profile by expressing preferences and selecting some interests. Information diffusion is therefore controlled by users'

profile and the way sociality impacts the information diffusion process is investigated. Furthermore, we define a routing framework that leverages on the degree of friendship of mobile users, i.e. depending on how close the interests of users are. Finally, we show how mobility of users can be leveraged in order to have a service evolving in a distributed and cooperative way.

Results presented throughout the thesis show that the system architecture is able to face and overcome the challenges and issues arising in a mobile pervasive computing environment, especially when mobile nodes gather information from a largely deployed WSN and those deriving from high mobility of nodes in the absence of a central server.

Sommario

Gli ambienti mobili e pervasivi sono caratterizzati da un numero estremamente vasto di dispositivi connessi in rete e inclusi nell'ambiente e da nodi mobili che recuperano le informazioni da questi stessi dispositivi. Tali ambienti stanno ultimamente ricevendo un'attenzione significativa dalla comunita' di ricerca, visto l'elevato proliferare sia di nodi sensori che di nodi mobili.

In questo lavoro di tesi, si assume un ambiente mobile e pervasivo, dove i nodi non comunicano con nessun server centralizzato. Si presenta un'architettura di sistema con tre tipi di nodi: (i) nodi mobili (ii) nodi gateway (iii) nodi sensori. I nodi sensori comunicano tra di loro tramite comunicazioni a piu' salti come nelle tradizionali reti di sensori wireless (Wireless Sensor Networks - WSN). I nodi mobili comunicano tramite comunicazioni peer to peer ad un salto, e le informazioni recuperate dai nodi sensori sono diffuse in maniera opportunistica. I nodi gateway rappresentano l'interfaccia tra i nodi mobili e i nodi sensori e non comunicano tra di loro in nessuna occasione.

Data questa architettura di sistema, vengono considerati gli scenari applicativi di Intelligent Transportation Systems (ITS) e di Mobile Social Networks (MSN) con le relative problematiche. Inizialmente, viene progettato e valutato un routing framework per lo scenario ITS, capace di affrontare dei sink mobili che interrogano la rete mentre si muovono come ad esempio delle macchine, in modo tale da recuperare le informazioni da una vasta WSN che ad esempio fornisce informazioni su dei parcheggi. Inoltre, si affrontano le problematiche collegate alla gestione dei dati al livello dei nodi mobili. Si assume infatti, che i nodi mobili recuperino le informazioni dai nodi gateway e si scambino le informazioni tra di loro ad ogni incontro, cosicche' l'ammontare di dati memorizzati in ogni dispositivo e poi scambiato deve essere controllato, vista anche la scarsita' di risorse degli ambienti mobili.

Nello scenario delle MSNs, viene inizialmente presentata una implementazione reale della architettura di sistema proposta per telefoni cellulari con Bluetooth e dotati di piattaforma Java. Usando questa implementazione, viene svolta una sperimentazione reale dove i nodi mobili si scambiano le informazioni ad ogni incontro grazie all'applicazione implementata. Agli utenti

viene richiesto di compilare un profilo personalizzato esprimendo delle preferenze e selezionando degli interessi. La diffusione delle informazioni e' pertanto controllata dai profili utente, e viene investigato il modo in cui gli aspetti sociali impattano la diffusione dei dati. Inoltre, abbiamo definito un framework di routing che sfrutta il grado di amicizia tra i nodi mobili a seconda cioe' di quanto i loro interessi sono similari. Infine viene mostrato come la mobilita' degli utenti possa essere sfruttata per ottenere dei servizi che evolvono in maniera distribuita e cooperativa.

I risultati presentati durante la tesi mostrano che l'architettura di sistema e' capace di affrontare e superare le sfide e i problemi tipici degli ambienti mobili e pervasivi, specialmente quelli riguardanti nodi mobili che recuperano informazioni da vaste WSN installate nell'ambiente e quelli derivanti dall'elevata mobilita' dei nodi in assenza di un server centralizzato.

Contents

1 Introduction

1.1 Motivation of the work

The field of mobile computing was born with the appearance of full-function laptop computers and wireless LANs in the early 1990s. With mobile computing we usually refer to the possibility of being able to use a computing device even when being mobile and therefore changing location. Portability is one aspect of mobile computing.
The exponential growth of mobile devices and the ascendant interest toward mobile services led research and industry to confront the problems that arise in building a distributed system with mobile clients. Although many basic principles of distributed system design continued to apply, according to [1], four key constraints of mobility forced the development of specialized techniques: unpredictable variation in network quality, lowered trust and robustness of mobile elements, limitations on local resources imposed by weight and size constraints, and concern for battery power consumption.

At the same time, a pervasive computing environment is commonly referred to as one saturated with computing and communication capability, yet so gracefully integrated with users that it becomes a ”technology that disappears.” According to the definition of Wieser [2], pervasive computing is a halo of embedded devices immersed into reality able to provide information to a human or to another device on the environments he is immersed in.
It is a matter of fact that pervasive computing is a nowadays a trend in which computing devices are increasingly ubiquitous, numerous and mobile. Therefore, it is common to refer to an environment of mobile devices interacting among them and with distributed embedded devices as a mobile pervasive computing environment.

In such an environment, the common paradigm of devices always connected to a central server cannot always be applied. From a networking point of view, scalability problems arise when considering a large set of mobile devices looking or sharing information through centralized communications. However, the work of Tse and Grossglauser [3] have shown that it is

possible to obtain a scale-free network by exploiting network mobility and node to node communications to convey the information, at the cost of an increased delay in packet delivery.
From an application point of view, it should be considered that although there is a proliferation of advanced mobile devices and services, we are still far to have users always connected to the web, given the technological and business constraints still existing for mobile web access through UMTS, Wi-Fi or WiMax connections. By considering device to device communication or direct access to the pervasive computing environment, an alternative networking communication is provided for mobile pervasive computing environment.
Therefore, the interest of the research community toward device to device communication has received a great attention in the last few years and this new paradigm of communication it is now commonly referred as opportunistic networking [4], [5], [6].
It is a matter of discussion of this thesis to understand the challenges arising in a mobile pervasive computing environment where the absence of a centralized connection is a constraints and information is retrieved from devices embedded in the surroundings. The main goal of this work is to define a system architecture able to overcome such challenges and define new communications technique and services in the emerging application fields of Intelligent Transportation Systems (ITS) and Mobile Social Networks (MSN).

1.2 Description of the work.

The main contribution of this work is the definition of a system architecture able to overcome the challenges arising in a mobile pervasive computing environment, its application to emerging scenarios such as those of ITS and MSN and the definition of new systems and techniques specifically designed to solve the issues of these scenarios.
In Sec. 2 the environment under evaluation and the system architecture are presented in details, together with the challenges. Three different types of nodes are taken into account: nodes embedded in the environment and sensing a particular phenomenon, able to communicate in a multi-hop fashion in order to reach the boundaries of the network; mobile nodes, gathering information from the surroundings and exchanging it among them; gateway nodes, in charge of creating a relationship among mobile and sensing nodes. This system architecture is specifically tailored for the application scenario of Intelligent Transportation Systems [7] where cars query a Wireless Sensor Network (WSN) [8] while on the move, and the one of MSNs, where nodes exchange gathered data according to common interests and preferences.
In Sec. 3 the system architecture is applied to ITS. In particular two different issues are treated

in this chapter. At first, the problem of mobile nodes querying a WSN while moving to gather interesting information is faced. In such a scenario, a geographic routing framework for WSN with mobile gateways is proposed and evaluated. The system parameters are dimensioned and particular strategies to overcome energy consumption related issues in WSN are shown in details. Then, data management issues in an ITS scenario are presented. In particular, a data management technique based on wavelet analysis is presented for multiple mobile nodes querying a large amount of sensing nodes and exchanging information among them, in order to provide the different level of details to the user depending on the distance among the node and the gathered data.

Sec. 4 deals with the application scenario of mobile social networking, when mobile nodes interact among them in an opportunistic fashion and exchange information based on the interests decided at the application layer. Initially, it is presented a possible implementation for smartphones of the proposed system architecture, together with the implementation issues deriving from the complexity of using a service developed for such devices in a real environment. It is depicted then a social analysis deriving from the use of this mobile service in an office environment, where interests and preferences of users are taken into account and used for filtering information diffusion among mobile nodes. Consequently, a routing framework based on similar interests, i.e. social friendship, is presented and evaluated through an extensive emulation study. Finally, it is also presented an example in which mobility of nodes and opportunistic information exchange is used not only to diffuse information, but also as a mean for having a service evolving without human intervention.

Finally, sec. 5 presents thesis conclusions and some promising future directions.

2 System architecture, scenario, and application

2.1 System Architecture

With the advent of pervasive computing, a huge amount of tiny sensing devices is embedded into the environment for providing information to querying users on the surrounding conditions or about a specific object. Such devices organize themselves into clouds of nodes, adopting multihop communication technique to deliver data to a central server.
The classical system architecture for modern pervasive computing take into account a large number of sensing devices somehow connected to central server, which delivers the information to mobile or non mobile users equipped with smart embedded devices. It is worth to mention several initiatives in the research community ranging from volcanic monitoring [9] to habitat and structural monitoring [10].
We are here considering a different system architecture in which mobile nodes avoid the connection to a centralized authority. The increasing capabilities of mobile devices together with the growing competences of mobile users are nowadays posing new challenges to the research community. Users are willing to access information in the surrounding environment to enhance their user experience of a mobile service. At the same time, they want to avoid connections to a central web server, mainly for privacy reasons, since a centralized system is commonly thought to violet user's privacy, and for economical reasons, since mobile web access is not yet accessible by everybody in every moment.

In this work we propose a different pervasive computing environment, in which the mobile user with its running service is posed at the center of the system and the lack of a backhaul connection is considered a requirement. From now on, then, we will refer to a mobile pervasive environment with the following characteristics:

- a large number of embedded devices with few intelligence onboard and wireless com-

munication capability

- mobile users bringing a handheld device, capable to gather information from embedded devices and communicate among them
- the absence of a central authority and/or the willing not to use any networking centralized system, so that devices exchange information in a peer to peer manner while on the move through proximity communications
- users cooperate in order to diffuse information throughout the distributed network of mobile cooperative users.

In the considered environment, mobile users gather information from sensing nodes and exchange this information among them exploiting their mobility. Therefore, we define a system architecture as the one shown in Fig. 2.1 where we identify:

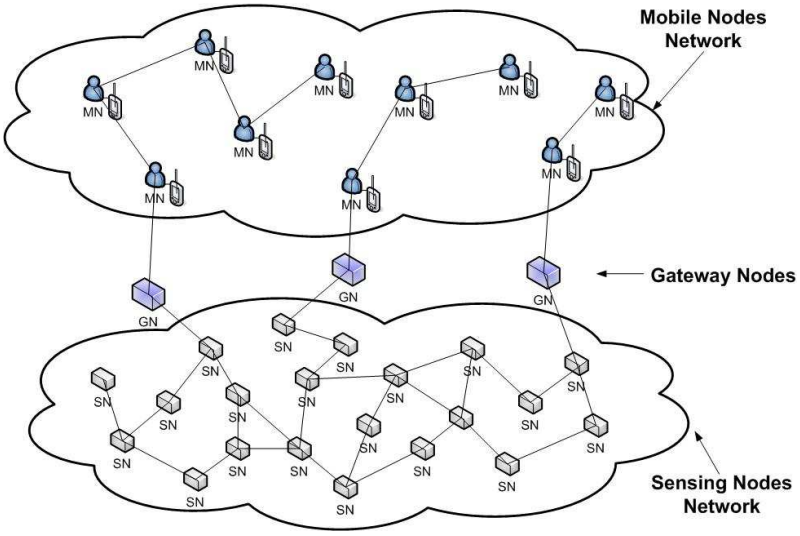


Figure 2.1: The general system architecture: mobile nodes network, composed by mobile nodes (MN), gateway nodes (GN) and the sensing nodes network, composed by sensing nodes(SN)

- *Sensing Nodes*, i.e. static nodes responsible of sensing a physical phenomenon (such as a parking place free or occupied or a video of the surroundings) and sharing it with other sensing nodes through multihop wireless communications

- *Gateway Nodes*, i.e. enhanced sensing nodes responsible of the communications among sensing nodes and mobile nodes; a sensing node is elect to be a gateway node either for its geographical position in the network or for an application requirement

- *Mobile Nodes*, i.e. nodes usually identified by handheld devices carried by mobile users and communicating with sensing nodes and other mobile nodes through on the fly one hop communications

In particular, a mobile node can query the network of sensing nodes by injecting a query through a gateway node. Then, a gateway node will reply to the mobile node with the response information that is either gathered from the network of sensing nodes through multihop communications or sensed by the gateway node itself. Once the information has reached the mobile node, it enters the distributed diffusion process of the network of mobile nodes.
Therefore, two layers of network exists: (i) the layer of sensing nodes, where queries and sensed information are diffused through multihop communication, i.e. the *sensing nodes network* (ii) the layer of mobile nodes, where moving nodes communicate through simple one-hop communication in a peer to peer manner, i.e. the *mobile nodes network*. Gateway nodes are part of the first network and realize the inter-communication among the two different networks.

2.2 Application Scenarios

The depicted system architecture could be adopted in several application scenarios, where mobility is a system requirement and embedded devices can enhance mobile users with sensed information.
The first application scenario is that of intelligent transport systems (ITS). The fast-growing field of ITS spans from flight transport and traffic management to in-vehicle services like driver alert or traffic monitoring. As a consequence, transportation information collection and communication plays a key role in all intelligent transport application. Unfortunately, most conventional ITSs can only detect the vehicle in a fixed position, and their communication cables and power cables elevate the cost of construction and maintenance. In fact, nowadays, collecting traffic data for traffic planning and management is achieved mostly through wired sensors. The equipment and maintenance cost and time-consuming installations of these existing sensing systems prevent large-scale deployment of real-time traffic monitoring and control. Wireless sensor network (WSN)[11],[10] offer the potential to significantly improve the

efficiency of existing transportation systems as for instance in [12], [13]. Because of the advantages of the WSN such as low power consumption, wireless distribution, and flexibility without cable restrictions, the usage of WSN in ITSs is expected to be able to overcome the above difficulties.

Usually, a WSN is typically constituted by (i) a set of resource-constrained nodes, which are deployed over the spatial region to be observed and capable of performing user-defined data gathering tasks, (ii) a sink node, where information of the WSN can be accessed by the end-user.

However, in ITSs, the usage of a traditional WSN may result extremely inefficient in terms of infrastructure to be deployed, and management overhead to be performed in order to guarantee the correct functioning of the network. In fact, when shifting to an urban setting, the scenario differs significantly from the previous one. Indeed, let us assume the WSN to be deployed along a road, or in city area covered by a specific pervasive service. Let us further assume the end-user, while moving around the city, to be in direct contact with the WSN, thus acting both as end-user and sink at the same time. In this case, the mobile user can directly access services offered by the ubiquitous WSN, without the need to resort to a remote repository where data is first stored, and then accessed. This is the case of ITSs, where deployed WSNs enable cars to obtain information on the conditions of the surrounding environment, and to use this information for taking appropriate decisions. The WSN could be for instance used for detecting the formation of ice over the road, or monitoring the status of parking spots along the streets of the city center.

Therefore, we can apply to the described application scenario, those paradigm designed for the system architecture under evaluation, in particular for the mobile nodes to sensing nodes communication.

However, the gathered data is usually not sufficient for providing a complete information about the surrounding environment. For instance, we can think to multiple cars looking for a free parking place: each car gathers information from sensing nodes about the closest parking lots, but it is not aware of any free place in more remote regions. Then, once the information has reached the mobile devices, we also want to leverage on users' mobility to enhance data diffusion. In fact, since cars are continuously moving and meeting with other cars, they could share gathered information and each user's experience would result enhanced. As a consequence, devices have to store a large amount of sensed data and exchange it when meeting another user on the fly. Given the short time of these opportunistic meetings, it is extremely important to treat information in such a way that its store and forward has a high efficiency in

terms on time and consumed resources. Therefore, before going into the details of the mobile user to mobile user communications and applications, efficient data management for the ITS application scenario techniques needs to be introduced.

Another application scenario of the presented system architecture is that of mobile peer to peer (P2P) applications and mobile social networks. When thinking of P2P systems, it is common to refer to distributed data sharing systems over IP networks, such as the well known initiatives of Napster, Gnutella [14], Kazaa [15] or Bittorrent [16]. However, the increased need for mobile service and the proliferation of advanced mobile devices have lead the research community to the introduction of mobile peer to peer systems, in which the lack of a central authority is a strict requirement [17], [18], [19], [20]. Frequently, mobile peer to peer systems only refer to file sharing for small localized mobile communities. By leveraging on the introduced system architecture, we want to go a step ahead.
First of all, in our application scenario mobile users will run a mobile service on his/her device. Such service will drive the gathering of information from the environment by querying sensing devices when on the move. This information will be filtered from an application point of view by user's interests and preferences. Furthermore, users communicate on the fly and upon meeting each other and consequently share any kind of information they are interested in, providing in such a way a controlled data diffusion for mobile communities. Therefore, each user exchanges information only with those users with similar interests and preferences: this throws the basis of a particular kind of mobile social network, where people get in touch while performing daily activities, without the need of being connected to the web or to any centralized server and especially by simply editing a personal profile. No particular constraints are posed on the type of information and on the size of the community.

2.3 Challenges

In the described system architecture some challenges need to be addressed.

- The first challenge can be identified with the communication issues between mobile nodes and sensing nodes in a high mobility environment. Embedded sensing devices are usually able to communicate among themselves in a distributed fashion when they are static or slowly moving as for instance in Wireless Mesh Network (WMN) or WSN. These nodes organize themselves into multihop networks and are then able to provide information to a remote server for offline analysis. Communication issues arise when

the information want to be retrieved in a real time manner and while on the move even with significant speeds as in an ITS application scenario. In other words, what are the communication issues related to accessing information on a distributed network from a mobile server?

- The second challenge is related to data management for mobile nodes. In fact, users' devices must be able to store data and exchange on the fly information useful for other users in a cooperative manner. In particular, it is challenging to store gathered and exchanged data in order to optimize the information diffusion process given the limited resources of each user (mainly energy and bandwidth) in this environment.

- The third challenge is related to the influence of sociality into networking. Given the fact that users cooperate to diffuse information, they will always seek for the information they are interested in, when running a service that allows each user to express a set of preferences and edit a personal profile. So, if we consider groups of users with similar interests to be "friends", the information diffusion process needs to be investigates as the friendship definition rule changes. In addition, it is worth to explore such a process in a real environment so that it can be understood how contact pattern among users affect the information diffusion process.

- The fourth issue is related to the software implementation of the described system architecture. Mobile devices nowadays coincide with mobile smartphones. Such devices are becoming more powerful as time goes by, both in terms of communication and computing capabilities. It is then challenging to understand which are the technological issues related to bringing the above mentioned concepts into those devices.

2.4 Related work

2.4.1 WSN and Intelligent Transportation Systems

There are several works addressing the problem of a mobile sink within the more traditional concept of a static wireless sensor network (WSN) deployed in the environment. We recall that differently from several other solutions and studies, our idea is to have a mobile user querying

directly the WSN while it is moving and the network is replying to that user with interesting data for the given application and we are interested in evaluating the performance of different forwarding strategies.

In SENMA [21] and MSSN [22] the mobile user gathers information through one-hop communications from either a sparse or dense WSN. SENMA (SEensor network with Mobile Agent) exploits nodes redundancy using mobile agents (e.g. mobile sinks) that collect data from WSN by means of one hop communications. The application depicted for SENMA is an aircraft flying above a dense deployed sensor network that wants to collect data regarding the sensed environment or performs some network monitoring techniques. MSSN (Mobile Sink in Sensor Network) is based on the same concept of one hop communications between sensor nodes of a sparse WSN and the mobile sink. The envisioned application is a police car moving along a highway for instance once a day to gather data from sensors deployed to collect traffic and surveillance information.

These work avoid the problem of multihopping and consequently of querying information and routing data by defining application where one hop relaying is the only way for the sink to gather data from sensors; it is also stated that multihop communication have worst performance in terms of energy consumption, although it regards the specific application scenarios depicted.

In [23], [24], [25] the sink acts as a mobile relay (i.e. data mules) that gathers data from a location and brings them somewhere else, while in [26],[27], [28] the mobile node takes place of a static node to let it save some energy. Both approaches have the final goal to maximize network lifetime exploiting sink mobility. In [29] an analysis of controlled mobility versus uncontrolled mobility is performed.

In Hyper [30] a routing strategy is designed to allow mobile users to gather information from the WSN while performing maintenance work or environmental evaluation. The goal is to relay packets to the mobile sink by efficient routing strategies, with the concern of throughput and energy efficiency.

In TTDD [31] the issue of multiple mobile sinks in a large and dense WSN is considered. Differently from other works, TTDD divide the network in a grid where each node is location-aware and tries to minimize the overhead due to multiple and frequent queries from the sinks using a hierarchical approach. This protocol is proven to perform as efficient as commonly used routing protocols in the presence of fixed sinks, in terms of energy consumption, throughput and end to end delay.

When considering the area of ITS, WSNs have been proposed as a solution for gather-

ing real-time information about road conditions [32]. In [7], the authors present a WSN for proactively detecting and advertising possible dangerous situations on roads. In [33], a traditional WSN architecture is optimized in order to achieve a high cars detection accuracy, while preserving a sufficiently high network life-time. Several works in the field of ITS focus on enhanced parking management strategies by enabling each parking space with a sensor node. In [34] a combination of magnetic and ultrasonic sensors for accurate and reliable detection of vehicles in a parking lot is used. In [35] the status of the parking field detected by sensor nodes is reported periodically to a database via the deployed wireless sensor network and its gateway. Reported information can be then used by a centralized system to perform various management functions, such as finding vacant parking lots, auto-toll, security management, and statistic report.
Finally, a market example is provided by Streetline technologies [36], a company located in San Francisco that deploy thousands of sensor nodes on parking lots and provides information on each parking lot (available, occupied or violated) to customers paying for the service. In this case the information is provided by a centralized system that collects all the data and send information to users' devices. We are here interested in investigating the case where customers are in direct contact with the deployed WSN

2.4.2 Opportunistic Networking and Mobile Social Networks

In data–centric architectures for opportunistic communications, the reference forwarding scheme is represented by epidemic routing [37],[38]. This is rooted in the ease of implementation and robustness with respect to network conditions typical of such schemes. Epidemic routing share many principles with controlled flooding [39], which has been extensively described through fluid approximations and infection spreading (see for example [40]). The control of forwarding has been addressed in the ad-hoc networks literature, e.g in [41] and [42]. In [41], the authors describe an epidemic forwarding protocol based on the *susceptible-infected-removed* (SIR) model [40]. Authors of [41] show that it is possible to increase the message delivery probability by tuning the parameters of the underlying SIR model.

Notice that in the case of end-to-end opportunistic networks, the target is to deliver messages with high probability to the intended destination: all message copies stored at nodes other than the destination represent overhead. Conversely, in the case of data centric forwarding, the trade-off involves a notion of *utility*, because not all message copies are redundant, as they can be relevant for a variety of users. In [43], this trade-off has been explored by applying a *publish-subscribe* paradigm, where each user takes into account other users "subscriptions"

for making appropriate data caching decisions. However, due to the overhead related to the regular update of users interests, such approach is not viable for large-scale and extremely dynamic systems.

As such systems depend heavily on the user behaviour (in terms of contact patterns as well as set of interests), various groups have recently started investigating the impact of social aspects on system design and performance [44],[45],[46],[47]. In particular, the structure of the social network the user is in was proved to significantly influence the performance of the system. Starting from this observation, social interactions have been accounted for in order to design opportunistic forwarding and data diffusion schemes. In [44], the community structure behind the social interactions has been studied in order to improve forwarding algorithms. The authors showed that there exists a limited set of nodes, called *hubs*, which play a central role in the diffusion of information. Being aware of the community structure, in [44] the authors showed that an extremely efficient trade-off between resources and performance can be achieved.
In [47], the impact of different social-based forwarding schemes were evaluated in the case of a DTN routing protocol. Similar to our work, real world mobility patterns were obtained from Bluetooth proximity measures. The authors of [47] showed that incorporating a friend/strangers classification in the forwarding policies can be beneficial in different application scenarios.

Opportunistic communications have been the focus of intensive studies in the past year, addressing the many facets of this innovative communication paradigm, including networking performance, software architectures [48],[49], social dimension [44],[47], etc..
Recently, different software architectures have been proposed for enabling the opportunistic diffusion of data in mobile environments. In [48], the haggle software architecture is presented in all its components. The described platform enables a seamless diffusion of data in mobile environments by separating the the application layer form the networking one, and confining each of the two to its functional role. This permits to maximize the performance of both layers.

The importance of the social network regulating the diffusion of data in opportunistic environments, has received a great attention in the past years.
In [44], the community structure behind the social interactions driving the data diffusion process has been studied in order to improve the performance of forwarding algorithms. Starting from experimental data sets collected in real-world experiments, it was first shown that the contact graphs were far from regular. In particular, it was shown that it is possible to identify a

limited set of nodes that were much more active than the others. Such nodes represent *hubs* of the network. Further, authors show how it is possible to divide the network into overlapping communities of nodes, and how these communities depend on the social relations of the nodes themselves. Finally, it was shown that by incorporating the knowledge of both nodes centrality and community belonging into forwarding algorithms an extremely efficient trade-off between resources and performance could be achieved.

The relevance of social behavior in opportunistic networks was further assessed by the work of Miklas et al. [47], where the impact of different social-based forwarding schemes were evaluated in the case of a DTN routing protocol, worm infection and a mobile P2P file-sharing system application scenarios. Also in this case the evaluation is based on real world mobility patterns, obtained from bluetooth proximity measurements. In particular, by applying a threshold on the periodicity of meetings it is possible to classify encounters in terms of strangers or friends, and analyze the properties of these 2 classes of meetings separately. Starting from this classification, authors conducted a statistical analysis of these 2 categories of meetings, concluding that (i) while most of the encounters are between strangers, meetings among friends account for almost two-thirds of the overall meetings, (ii) networks of both strangers and friends are scale-free and (iii) the network of friends have a very high-clustering coefficient. Finally, the authors showed that incorporating this friend/strangers distinction in the forwarding policies can be beneficial in different application scenarios, such as P2P file sharing or worms spreading prevention.

3 Application to ITS

3.1 Introduction

The aim of this chapter is to address the challenges and issues arising when using the proposed system architecture in an Intelligent Transportation System scenario. In particular, we address in Sec. 3.2 the communication challenges of mobile nodes querying a large Wireless Sensor Networks (WSN) and in Sec. 3.3 the problems of keeping large amount of data coming from sensing nodes at the mobile nodes layer.

WSN [8],[10] are receving a significant attention from the research community, thanks to their capability of being fully immerse into the physichal phenomhenom to be monitored, thus giving the possibility to collect data with a granularity not possible before. Tipically, a WSN is constituted by (i) a set of resource-constrained nodes, which are deployed over the spatial region to be observed and capable of performing user-defined data gathering tasks, (ii) a sink node, where information of the WSN can be accessed by the end-user. Often WSNs are deployed in remote hostile regions, and the sink is the only node through which the WSN is first queried and then accessed for data gathering operations. Hence, the sink is expected to be connected to the backend through some form of long-range connection (i.e., satellite communication, Wi-Fi, Wi-Max, etc.). This may result extrememly inefficient in terms of infrastructure to be deployed, and management overhead to be performed in order to guarantee to the network the correct functioning. However, when shifting to an urban setting, the scenario differs significantly from the previous one. Indeed, let us assume the WSN to be deployed along a road, or in city area covered by a specific pervasive service. Let us further assume the end-user, while moving around the city, to be in direct contact with the WSN, thus acting both as end-user and sink at the same time. In this case, the mobile user can directly access services offered by the ubiquitous WSN, without the need to resort to a remote repository where data is first stored, and then accessed. This is the case, for instance, of Intelligent Transportation Systems (ITS), where deployed WSNs enable cars to obtain information on the conditions of the surrounding environment, and to use this information for taking appropriate

decisions. The WSN could be for instance used for detecting the formation of ice over the road, or monitoring the status of parking spots along the streets of the city center.

This application domain, while intuitively clear, requires a general refinement of the standard WSN architecture and protocols. In particular, we are moving from a centralized architecture, where nodes self-organize at boostrap phase for the delivery of data to the sink node, to a fully distributed one, where no pre-determined gateway can be identified.
The considered scenario consists of a WSN deployed over an urban area. Multiple mobile sinks injecting queries into the WSN, which answers, later on, with the requested information, if available. No particular requirement is imposed over the WSN deployment geometry, and packets are routed within the network according to an adaptable geographical routing mechanism, where the position of final destination (sink) is adapted to the mobility pattern of the mobile user querying the network. This adapatation process is achieved through a mobility prediction scheme, which takes into account speed variations, sudden direction changes, etc., when forwarding packets.

At the same time, the increasing pervasiveness of mobile devices is enabling totally new networks and mobile applications, where information is diffused with no need for dedicated network infrastructures by employing epidemic-like information dissemination techniques [50],[51],[52],[53]. By taking advantage of proximity communications, any communication opportunity, or "contact" [51], is exploited for exchanging information. Mobile nodes contributing to the diffusion of information may range from cars [52], to people [51], or buses [53]. Such enlarged networking possibility is enabling a wide range of innovative mobile application scenarios such as vehicular sensor networks [54], where sensed data is distributed by means of local communications among carstributed context provisioning, where information is shared among mobile nodes in an ad-hoc fashion [55].
As the number of mobile users increases, and more rich content (i.e., images) enters the application scene [56], in order to apply such techniques, it is of paramount importance to make a parsimonious use of the limited resources of mobile nodes, i.e., storage and battery. Hence, efficient mobile data management techniques need to be devised for limiting the data to be exchanged.

We consider an application scenario where mobile nodes are running context-aware applications [55]. The information contributing to the creation of the surrounding context [57] is

originating from sensors embedded in the environment, and is exchanged by mobile nodes whenever in mutual communication range. Context-aware applications are typically requiring fine grained data on the surrounding environment, and only a coarse approximation of remote regions. It is therefore a natural choice to manage the stored data in a way that data originating from nearby sensors is kept (and stored) at full resolution, while information originating from remote sensors is compressed by reducing its resolution. This multi-resolution property is achieved by exploiting the characteristics of the Discrete Wavelet Transform, which is applied to the stored information. The benefits of the proposed approach are twofold: first, it becomes possible to trade off the accuracy of the information for the required storage resources (i.e., memory allocation required on mobile devices for storing the context information) depending from the specific application scenario; second, by allowing a compact representation of the stored information, it becomes possible to apply epidemic-like dissemination mechanisms for diffusing data among mobile nodes.

3.2 Mobile Sinks Querying a Large WSN: Protocols and Techniques

3.2.1 System Model

The operative scenario we refer to is constituted by an *information retrieval* area[1] with one or more mobile users moving around it and querying for data.
As an example, this includes the case of cars moving around at rush hours, and looking for a free parking spot in a particular area (i.e., close to a subway station, close to user's favorite shop). To this purpose a car acts as a sink: along its trajectory it gets temporarily connected to the network, and sends a query asking for information on certain geographical region, and then waits for the corresponding data.

We have modeled this reference scenario with a simple system architecture, by logically separating three categories of nodes. As it can be seen in Fig. 3.1, we have identified the following types of nodes in our network:

- *Sensor Nodes (SNs)*, in charge of "sensing"[2] a certain zone; in particular we assume a large number of SNs deployed around a building in a block of a city town or in a trading centre, detecting a parking lot around the building to be free or occupied; furthermore

[1] It is comprised of nodes making the sensing of a phenomenon of interest.
[2] Clearly, the sensing operation depends from the specific service supported.

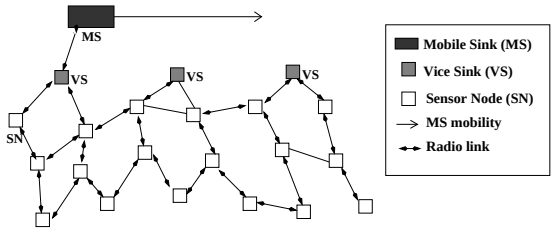


Figure 3.1: System architecture: a Mobile Sink (MS) querying a WSN while moving, while nodes closer to the street are the so called Vice Sinks (VSs), that are in charge for communicating with MSs. The nodes in the inner WSN are the so called Sensor Nodes (SNs) communicating in a multihop fashion and reaching VSs.

we assume each SN to be aware of its geographical position, for instance storing in each node its coordinates during deployment;

- *Vice Sinks (VSs)* that constitute the edge nodes managing the communications from and to MSs; we assume each VS to be aware of its position and of the position of the closest VSs and to have a unique progressive identifier (ID)

- *Mobile Sinks (MSs)*, comprising the nodes moving along the deployment area where *VS* are deployed; we assume each MS to be equipped with a satellite receiver like GPS, so that information on position, direction and speed is always available.

VSs are disseminated along the WSN network perimeter, but no particular constraint is applied to their density. In particular, it is not assumed that the MS is always reachable, thus resulting in several disconnections experienced from the MS, and it is not assumed that the VSs form a subnetwork, as they are not necessarily connected to each other.
MSs are moving according to a *constrained* random waypoint mobility model, where their position is constrained to stay along the peripheral area inside which the WSN is deployed. This includes the case of mobile users driving around a city block, and looking for a free parking spot as depicted in Fig. 3.2.

The communication architecture needs to be designed in such a way that a MS is allowed to send a query and to receive related responses, managing in a transparent way possible disconnections. It implies defining proper interfaces among MS and VSs, and SNs as well.

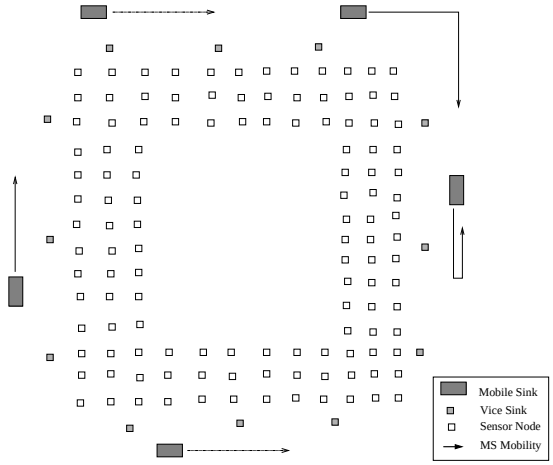


Figure 3.2: An example scenario: cars move around a WSN deployed around a building and look for a free parking lot.

3.2.2 Routing Framework Design

The routing framework we propose for the outlined architecture is composed by a geographic routing forwarding strategy enhanced by mobility prediction. In fact, after query injection in the network by a MS, a response message is expected to reach the outer nodes of the network by predicting the new position of MS, according to the mobility information included in the original query message. Typically, the VS node closest to the estimated position will be reached by the response packet. Then, if the MS is effectively in local proximity of the VS the response is delivered with success, otherwise the packet needs to be routed toward the most likely actual position of the MS. In order to support that, we propose a geographic forwarding strategy coupled with an efficient mobility prediction scheme, able to use, at the VSs, latest mobility information available at the MS. The reference scenario is briefly summarized in Fig. 3.3, where the MS injects a query on the WSN through the first VS. The query is then forwarded to the interested region (highlighted in grey) where the destination node (the closest to the center of the region) aggregates data of the region of interest by querying other nodes belonging to the same region and then sends collected data toward the target destination. The requested information is first forwarded from the SNs by means of multi-hop communications, and, finally, delivered from the last VS to the mobile sink.

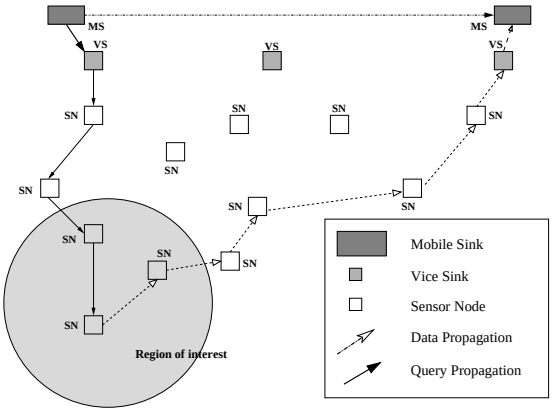


Figure 3.3: Example of the considered application scenario. The Mobile Sink injects a query in the WSN through the closest Vice Sink . The query is forwarded to the queried region (highlighted in grey). The queried data is finally delivered through another VS.

Packets format

Before going into details of the routing strategy, let us introduce five different types of packets in order to support both the geographic forwarding and the mobility prediction strategies. The packets, their fields and the actor of the network managing them are the followings:

- *HELLO* packet: a simple packet sent periodically containing node's ID, geographical position, a flag to specify whether it is a VS node or not, the remaining energy (used for energy aware forwarding) and the current duty cycle (used for delay aware forwarding);
- *MOBILIY* packet: a simple packet sent by the MS to every VS in local proximity, containing Direction of movement, Geographic coordinates, Speed and a Global Timestamp;
- *ALERT* packet: a message generated by every VS upon notification by a MS of a change occurred in its mobility pattern. It contains all the information contained also in the MOBILITY packet, plus the ID of the originating VS, the network address of the sender of the packet and the geographical coordinates of the destination of the ALERT packet;
- *QUERY* packet: a message generated by the MS after selecting the target region of the query itself. It contains the mobility information as in the MOBILITY packet, the geographical coordinates of the centre of the target region and its radius of interest, the network address of the sender and the TTL of the packet;

- *REPLY* packet: a message generated by the SN closest to the centre of the target region of the QUERY packet that contains all the mobility information of the originating MS as in the MOBILITY packet (copied from the QUERY packet), the actual position of the MS evaluated hop by hop according to mobility information and elapsed time, the network address of the sender and the TTL of the packet (copied from the QUERY packet).

The way these packets are handled by the routing framework is shown in the following sections.

Geographic forwarding

As stated in Sec. 3.2.1, we assume each node of the network to be aware of its position and each MS to be enabled with a satellite receiver such that it is able to know its coordinates, speed, direction and global timestamp. For the sake of simplicity, let us identify the coordinates of the target region stored in the QUERY packet with *TargetPos*, the coordinates of the MS stored in the REPLY packet with *MsPos* and the coordinates of the node that is currently taking the decision on the next hop with *CurrentPos*.

We describe the routing framework by separating the phases shown in the following. Network Topology Construction In the network bootstrap phase each SN builds its one-hop neighbor table by means of reciprocal HELLO packets exchange. Every node at bootstrap sends a *HELLO* packet described in Sec. 3.2.2. HELLO messages are scheduled at random instants in order to avoid collisions. Once the bootstrap phase is concluded every node has created a routing table containing neighborhood's geographical information. After the bootstrap phase is completed, each node periodically evaluates its remaining energy and its current duty cycle, stores this information in the next *HELLO* packet and then periodically broadcasts it. In such a way, each node in the network is aware of the position, the energy and the current duty cycle of its neighbors.

Packet Forwarding strategy

A node forwards the packet towards the destination by selecting the neighbor in the direction that maximizes the scalar product φ with the versor $vers(\cdot,\cdot)$ in the direction of the *targetPos*, i.e. the center of the target region for *QUERY* packets or the estimated position of the MS for *REPLY* packets. The scalar product is evaluated as follows:

$$\varphi_i = vers(\textit{neighbourPos}_i, \textit{targetPos}) \cdot vers(\textit{currentPos}, \textit{targetPos}) \tag{3.1}$$

where φ_i is the scalar product evaluated among the versors of the current node position *currentPos* and the *i-th* neighbour position $neighbourPos_i$ with the target position *targetPos*.
When a node finds out that the next hop coincides with the previous one, a DEADLOCK is discovered and the forwarding strategy enters the *back-up* mode: the current node stores in the packet the incoming direction in a field named *back-up angle* (e.g. the angle among the previous and the current node) and selects the neighbor that maximizes the scalar product with the versor in this direction. At every step then, if the *back-up angle* is set, a node tries at first to find a neighbor that maximizes the scalar product toward the destination, otherwise if another DEADLOCK occurs, it keeps on following the *back-up angle* direction previously set, in order to overcome the hole. When a neighbor closer to the destination and different to previous hop is found (i.e. the hole is overcome) the *back-up angle* can be reset and nodes keep forwarding the packet toward the destination. This simple strategy allows each packet to reach the target region avoiding holes and deadlocks. The forwarding strategy and the DEADLOCK event are shown in Fig. 3.4.

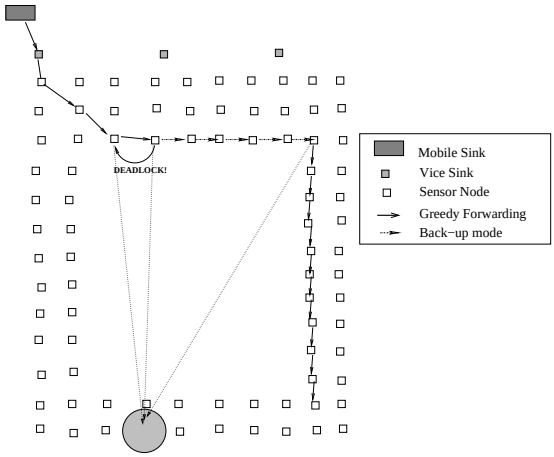


Figure 3.4: A deadlock occurring when forwarding the packet toward a target region and founding a hole: the scalar product would be maximized by the previous hope, so the *back-up* mode is entered and the packet is forwarded around the hole. The *back-up* angle is reset when a node different from the previous hop that maximizes the scalar product toward the destination is found, and this happens exactly at the end of the hole.

Query propagation A MS sends a query specifically to obtain information about a selected region. The region is specified by geographic coordinates and the query is forwarded toward the center of that region by each node according to the described packet forwarding strategy.

Using a SQL-like syntax, Alg. 1 presents an example of a query injected by a MS in the WSN.

Algorithm 1 Example of a query injected in the Wireless Sensor Network by the Mobile Sink.

SELECT parkingInfo
FROM sensors
WHERE region C_r, R_r
MOBILITY POS_{MS} , V_{MS} , DIR_{MS} , T_{MS}

Once the query is accepted by the target VS, the MS continues moving along the road, while expecting to receive the requested information in a reasonable amount of time. The *targetPos* (i.e. the centre of the target region) of a QUERY packet selected by the MS remains unchanged after each hop of QUERY packet forwarding. Once the destination is reached by the query, the forwarding process is stopped and the source prepares to send the requested information. Mobility information reguarding the original MS will be used to re-route the response toward the new position of the MS.

Query response

The *MsPos* for REPLY packet's forwarding is evaluated hop by hop according to the mobility information sent by the MS and originally included in the QUERY packet. The adaptive routing strategy implements the followings at each SN node:

1. evaluate the target destination based on the MS mobility information and the actual time;
2. prepare the REPLY packet to forward including MS mobility information, data and next hop ID;
3. check among the neighbors if there is a VS; if yes send the message toward it, if no select the closer node to the target destination according to the packet forwarding strategy;

Information delivery Only VSs are responsible for delivering information to MSs. Once the information has reached a VS, if the MS has not already passed by the current VS, a timestamp is set in order to wait for the MS for a reasonable time. If instead the MS has already passed by, the VS will use the received mobility information to re-route the packet towards the next target destination. The packet will reach the SN one hop further, and following the

previously described strategy, it will go through the SNs in the direction where MS is moving, till the packet will be received by the next VS. This process will be iterated till an application dependent timestamp expires. This can happen for highly irregular movements of the MS.

The whole geographic forwarding strategy is summarized in Alg. 2.

Algorithm 2 Packet forwarding strategy

```
receive msg;

if (msg is HELLO)
   update neighbors table;
else if(msg is QUERY)
   find next hop;
   forward QUERY to next hop;
else if(msg is REPLY)
   find next hop;
   if (TypeOfNode is VS)
      if(has updated mobility info)
         update mobility info;
         find next hop;
         forward REPLY to next hop;
      else
         store msg for a given time;
   else
      forward REPLY to next hop;
endif
```

Mobility management

The main goal of the mobility management strategy is to the inform the VSs with the newest mobility information of the MSs. This is done since REPLY packets will certainly reach the outer part of the WSN and then the VS closer to the estimated position of the MS. If the MS is not directly reachable by this VS, a decision on REPLY packet forwarding has to be appropriately taken. For this purpose, mobility information is sent by MSs every time they can communicate with a VS. In particular a MOBILITY packet is sent including fresh information about position, speed, direction and global timestamp. Each VS maintains this data in a structure that is updated upon receiving a fresher packet. When a REPLY packet reaches a VS two different decisions can be taken:

1. if no information fresher than the one currently stored in the packet are present at the

VS, the packet waits a predefined amount of time for the MS to pass by or for fresher information (we will show later on how this can be achieved);

2. if fresher information is present at the VS, the mobility fields of the REPLY packet are updated and the packet is immediately forwarded toward the new destination.

Since a MS can invert the direction of mobility or simply make a turn it is crucial that close enough VSs get informed about the new mobility information if they cannot be directly reached by a new MOBILITY packet. Therefore we have introduced an algorithm that is able to inform a given number of VSs about the new mobility information, so that a REPLY packet can efficiently be forwarded toward the appropriate destination after reaching a VS. Whenever a VS detects a drastic change of direction in the mobility pattern it alerts close by VSs with the new mobility information.
In particular 2 different situations can occur:

- if the MS informs a VS of a just occurred inversion of direction, the VS sends an ALERT packet with the new mobility information towards the VSs in the previous direction of the MS. In such a way, a REPLY packet routed to a destination where the MS is expected to be found (according to the original mobility information) is immediately forwarded in the opposite direction, therefore increasing the probability of success;
- if the MS informs the VS of a just occurred change of direction while keeping the same direction around the WSN (for instance it has turn to another side of the parking lot area), the VS informs the other VSs of the previous side of the occurred change, e.g. the VSs previously encountered by the MS. This helps a REPLY packet being forwarded towards one side to immediately being routed according to the new information.

It is clear that such a technique introduces an additional communication overhead, but at the same time it allows the management of critical situations with a higher message delivery ratio and a lower latency. However, a Time to Live (TTL) field for the packets needs to be properly set in order to avoid useless information to be propagated in the network. In this case, a user looking for a parking lot in a specific geographic region could consider the information to expire after a given amount of time; it then forwards a new query until the reply arrives. In such a way we are able to evaluate the time needed by each user to receive the queried information (i.e., to find a free parking) in different conditions of mobility and network topology, as we will show in the following section.

The combination of geographic forwarding and mobility prediction strategy is reported in Alg. 3

Algorithm 3 Strategy applied at every VS for REPLY packet forwarding when receiving mobility information

```
receive msg;

if (msg is MOBILITY || msg is ALERT)
   if(MS is connected)
      send stored REPLY to MS;
   else
      find next hop;
      send stored REPLY to next hop;;
endif
```

3.2.3 Load Balancing Techniques

Energy consumption is one of the main problems in WSN and especially in large scale deployment as those we are considering here. A WSN is composed by several nodes that are battery-supplied and which cooperate for distributing and delivering sensed information to querying nodes. Ideally, all the nodes should consume the same amount of energy and should die almost at the same time. It is obvious that depending on network deployment and topology, as well as on traffic load, some nodes are more stressed than others and happen to die with a higher probability. When a node dies, all the network has to re-configure itself with a high consumption of resources. Energy aware strategies aim at reaching network balancing with smart forwarding strategies or efficient MAC protocols, prolonging in such a way network lifetime, i.e. the time before which the first node in the network dies.
Given the previously described routing framework, we propose now two different techniques for load balancing in our system model: energy-aware forwarding, a strategy that works at the network layer and delay-aware forwarding, a cross-layer technique that involves the MAC layer. In particular when taking a decision on the next hop, each node evaluates a metric $dist_{x-i}$ by taking into account energy consumption or packet delay and decide for the neighbour that minimized the value of $dist_{x-i}$.

Energy-aware forwarding

We have shown in Sec. 3.2.2 that each node decides the next hop of a message by maximizing the progress towards destination. Each node broadcasts its position at network bootstrap and collects information about its neighbours through *HELLO* packets exchange.
We recall that each node periodically broadcasts its position together with information about

its battery consumption as described in Sec. 3.2.2. In order to keep the proposed strategy as general as possible, we further assume that energy consumption directly depends on the number of transmitted and received packets, i.e. at node i:

$$E_i = E_{init}\left(1 - N_i\alpha_{pkt})\right) \tag{3.2}$$

where E_i is the energy available at node i, E_{init} is the available energy at bootstrap, N_i is the number of received and transmitter packets at node i and α_{pkt} is the percentage of energy consumed at each transmition. By periodically broadcasting this information, each node is then aware of the available energy of its neighbours with a good approximation, depending on the HELLO packet period.
We introduce now a different metric for packet forwarding decision. Let us denote by φ the scalar distance evaluated as described previously. The distance $dist_{x-i}$ among node x and its neighbour i is then:

$$dist_{x-i} = \varphi_{x-i}\left(\frac{E_{init} - E_i}{E_{init}}\right) = \varphi_{x-i}\left(1 - N_i\alpha_{pkt}\right) \tag{3.3}$$

Next hop decisions are then taken accoring to this metric.

Delay aware forwarding

We introduce now a cross-layer strategy based on an adaptive duty cycle at each sensor node, according to its energy consumption. We refer to the A-MAC protocol described in [58], where the adaptive duty cycle δ_c is defined at each node according to the following metric τ:

$$\tau = \frac{T_{elap}}{T_{init}} - \frac{E_{elap}}{E_{init}} \tag{3.4}$$

wehre T_{elap} is the elapsed time since network bootstrap and E_{elap} is the elapsed energy since network bootstrap.
Ideally, τ is equal to 0 and the network is completely balanced; when τ is above 0 it means that the node is consuming less than expected, while when τ is below 0 the node is excessively stressed. δ_c is adaptively varied according to τ. In particular, as respect to a starting value of $\overline{\delta_c}$:

- δ_c is increased by 2 times when $\tau < 0$
- δ_c is decreased by 2 times when $\tau > 0$

- δ_c is mantained when $\tau = 0$

As explained in [59], an average delay corresponds to a given duty cycle for each packet transmition. We evaluated a delay of $130ms$ for $\delta_c = 1.5\%$ and a packet rate of $521\ packet/s$ [59]. As we increase the duty cycle this delay is decreased while it grows when the duty cycle is reduced. As described in Sec. 3.2.2 a node broadcasts to its neighbours $delay_i$, i.e. the delay a node x should expect when transmitting a packet to a node i. We can now define a new metric:

$$dist_{x-i} = \varphi_{x-i} delay_i \tag{3.5}$$

If next hop decision is taken according to this metric, we expect a higher network balancing and the average latency to be reduced with respect to geographic forwarding.

3.2.4 Performance Evaluation

System model evaluation

In order to evaluate the proposed system performance we resort to extensive numerical simulations with an open source simulation tool [60]. We assumed N_s^2 sensors to be uniformly deployed around a hole placed in the center of a $1000m \times 1000m$ square as shown in Fig. 3.2 in accordance to a regular grid disposition, with the distance between two consecutive sensors being 25 m. The size of the hole is $750m \times 500m$. A number of equally spaced vice-sinks (VSs) are also installed along each side of the road. Mobile users are moving with a speed that is uniformly distributed between a minimum and maximum value and invert direction of movement (clockwise or counterclockwise) with a certain probability. The current speed of the mobile users is updated every 5 s and the decision about an inversion is taken every 30 s. MSs move along the outer road and can communicate only with VSs and have a communication range of 25 m. Every time a MS reaches a corner of the outer road, it changes its direction, while preserving the way of movement, either clockwise or counterclockwise.

We initially evaluated the performance of the network with a single MS that randomly selects the starting position in the outer square and the initial direction of movement. It also selects a region in the network and sends a *QUERY* packet toward it through a selected VS. Every time a *QUERY* is injected in the network the MS starts a timer and accordingly sets up a Time To Live (TTL) field of the packet, after which it is dropped. The MS sends the same *QUERY* packet again until it does not receive *REPLY* before the timer expires. We have set the expiration time to be 90 s. Nodes communicate at a rate of 250 Kb/s, and adopt CSMA/CA

mechanism for managing medium access.
We have identified three different parameters to evaluate the proposed system architecture and a related routing framework:

- the number of VSs deployed on each side of the outer square, $N_{vs\text{-}side}$;
- the number of VSs alerted by a VS $N_{vs\text{-}alerted}$, when occurring a change in the mobility pattern of the MS;
- the mobility pattern of the MS, differentiating the contribution of the speed, $K_{speed} = v_{max} - v_{min}$, and the probability of inversion, $p_{inversion}$.

At first we have evaluated how each of the three key parameters introduced above affects the performance of the network, by fixing two of them and varying the remaining one. When varying $N_{vs\text{-}side}$, we consider $N_{vs\text{-}alerted}$ to be the ceiling of $N_{vs\text{-}side}/2$, then keeping a symmetry in different simulation. Each plot has been obtained running 500 different simulations and evaluating the average latency for received packets between query injection and reply reception, with a confidence interval of $98\ \%$.

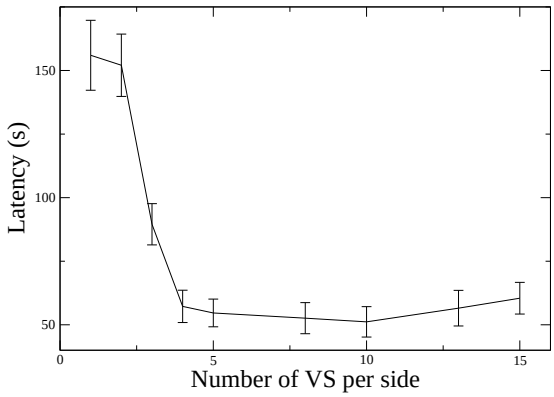


Figure 3.5: Average latency of packet with confidence interval of 98% while varying the number of VS per side, with 1000 SNs reguarly deployed in a $1000m \times 1000m$ square around a hole of $750m \times 500m$, distance among SNs $25\ m$, 1 MS, $K_{speed} = 10m/s$, $p_{inversion} = 1/2$ and $N_{vs\text{-}alerted}$ equal to the ceiling of $N_{vs\text{-}side}/2$.

We initially fixed the mobility of MS and $N_{vs\text{-}alerted}$ as stated above. The results in Fig. 3.5 show that increasing the number of VS per side from 1 to 15, e.g from a case where the MS

is almost always disconnected to a strongly connected one, decreases the latency of packets until $N_{vs\text{-}side} = 4$, after which the performance does not change significantly. This shows the behavior of the proposed routing framework for such a scenario, where disconnections from the network of the MS are supported in an efficient way even in condition of high mobility of the MS.

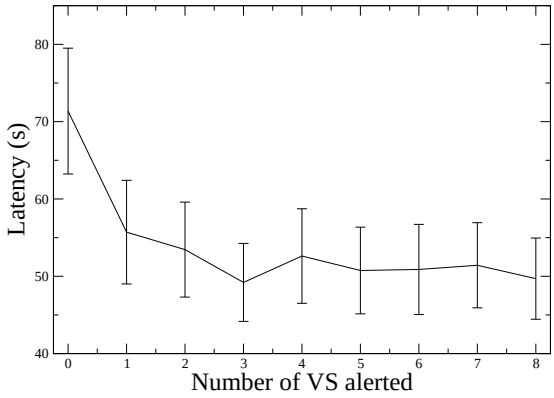


Figure 3.6: Average latency of packet with confidence interval of 98% while varying the number of VS alerted, with 1000 SNs reguarly deployed in a $1000m \times 1000m$ square around a hole of $750m \times 500m$, distance among SNs 25 m, 1 MS, $K_{speed} = 10m/s$, $p_{inversion} = 1/2$ and $N_{vs\text{-}side} = 8$.

Then we fixed $N_{vs\text{-}side}$ to 8 and the mobility pattern of the MS, by varying the $N_{vs\text{-}alerted}$ from 0 to 8. It can be seen from Fig. 3.6 that latency sensibly decreases when the mobility tracking strategy is introduced (moving from $N_{vs\text{-}alerted} = 0$ to 1), reaching a floor value for $N_{vs\text{-}alerted} = 3$. This shows that it is sufficient to alert a limited number of close by VSs to efficiently deliver packets to the MS, without flooding the network with *ALERT* packets.
Finally we fixed $N_{vs\text{-}side} = 8$ and $N_{vs\text{-}alerted} = 4$ and varied the mobility pattern of the MS, i.e. V_{max} from 10 m/s to 20 m/s and $p_{inversion}$ from $1/4$ to 1, keeping $V_{min} = 5$ m/s. It can be easily observed in Fig. 3.7 that the speed of the MS does not increase the latency as the probability of inversion does. As a consequence, the routing framework and in particular the mobility management protocol is not dependent on the speed variation, while the probability of inversion strongly affects the latency of packets.

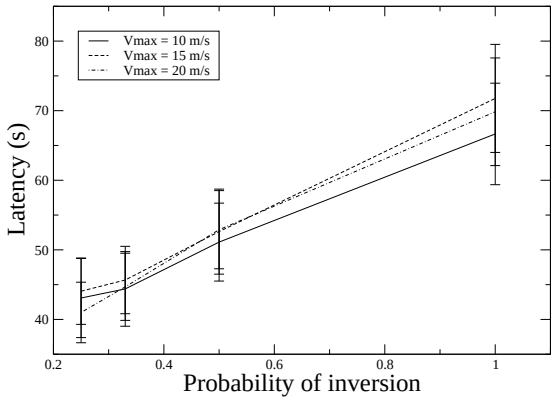


Figure 3.7: Average latency of packet with confidence interval of 98% while varying the mobility pattern, with 1000 SNs in the network reguarly deployed in a $1000m \times 1000m$ square around a hole of $750m \times 500m$, distance among SNs 25 m, 1 MS, $N_{vs\text{-}side} = 8$ $N_{vs\text{-}alerted} = 4$.

After dimensioning the network, we finally introduce multiple MSs that query the network until they receive a *REPLY* packet while moving around. In this scenario, the energy consumption of nodes in the network has evaluated, by averaging over 100 different simulations where 50 MSs query the network (one query per MS per each simulation run) with $N_{vs\text{-}side} = 8$, $N_{vs\text{-}alerted} = 4$, $K_{speed} = 10\ m/s$ and $p_{inversion} = 1/2$. In Fig. 3.8 a plot of the network, with coordinates of nodes in the x and y axis and average energy consumed by each node at position x, y is shown. In order to evaluate the performance of the routing framework in a platform independent way, we were considering a counter for every node increasing by one at each transmission or reception of a packet. In such a way we do not include also the aspects of MAC layer that affect battery consumption, such as carrier sensing and packet retransmission due to collisions.

It is easy to observe that the routing framework concentrates packets around the hole, since every time a packet discovers a *DEADLOCK* it enters the *back-up* mode until the hole is not overcome. Also, it can be observed that nodes close to the outer road are very stressed, and this fact has a two-fold reason: first, once a packet reaches one of these nodes, the routing framework is such that a MS is followed according to the newest mobility information arrived at a VS, then being forwarded on SNs deployed along the road; second, multiple MSs moving around cause a large amount of *ALERT* packets to be forwarded to close by VSs, always fol-

lowing the paths along the road.
It is straightforward to notice that an energy distribution algorithm should be introduced here. A technique as the one proposed in [58] such that the duty cycle of each SN is adapted to the current remaining amount of battery power of the node itself, could force packets to be spread all around in the network, avoiding the showed diversity in energy consumption. The evaluation of such an algorithm is part of our future work.

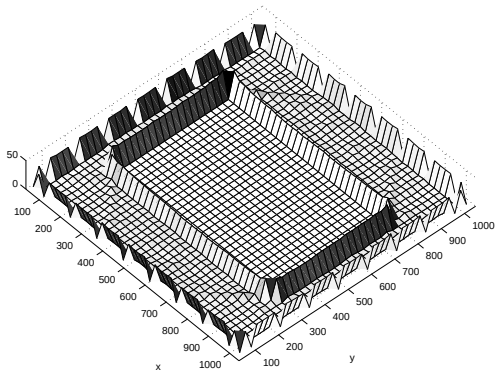


Figure 3.8: Average energy consumption for 1000 SNs reguarly deployed in a $1000m \times 1000m$ square around a hole of $750m \times 500m$, distance among SNs $25\ m$, with 50 MSs, $K_{speed} = 10m/s$, $p_{inversion} = 1/2$, $N_{vs\text{-}side} = 8$, $N_{vs\text{-}alerted} = 4$

Load balancing techinques evaluation

Furthermore, we want to address through an extensive simulative study a comparative performance evaluation among simple geographic forwarding and load balancing techniques. Once again, we assumed N_s^2 sensors to be uniformly deployed around a hole placed in the center of a square as shown in Fig. 3.2 in accordance to a regular grid disposition, with the distance between two consecutive sensors being 25 m. The size of the hole is $500m \times 500m$. A number of equally spaced vice-sinks (VSs) are also installed along each side of the road, and the distance among VSs is fixed to 125 m.
Mobile users are moving with a speed that is uniformly distributed between a minimum and maximum value and invert direction of movement (clockwise or counterclockwise) with a certain probability. The current speed of the mobile users is updated every 5 s and the decision

about an inversion is taken every 30 s. MSs move along the outer road and can communicate only with VSs and have a communication range of 25 m. Every time a MS reaches a corner of the outer road, it changes its direction, while preserving the way of movement, either clockwise or counterclockwise.
An MS randomly selects the starting position in the outer square and the initial direction of movement. It also selects a region in the network and sends a *QUERY* packet toward it through a selected VS. Every time a *QUERY* is injected in the network the MS starts a timer and accordingly sets up a Time To Live (TTL) field of the packet, after which it is dropped. The MS sends the same *QUERY* packet again until it does not receive *REPLY* before the timer expires. We have set the expiration time to be 90 s. Nodes communicate at a rate of 250 Kb/s, and adopt CSMA/CA mechanism for managing medium access.
Also, the duty cycle operation has been taken into account by introducing an additional delay to each packet transmission.
We aim now at evaluating and comparing the performance of the proposed routing strategies while varying the size of the network and preserving MS mobility characteristics. From now on, the simple geographic routing is labeled as *Geo-forwarding*, the energy aware routing with *Energy-forwarding* and the delay aware routing with *Delay-forwarding*.
The parameters under evaluation are:

- *latency*, the delay between transmission of a query and reception of a reply by a MS
- *hop-count*, the number of hops passed by each packet
- *time to first failure*, the time before which the first node in the network dies

We fixed the initial energy of a node to an adimensional value $E_{init} = 50$ and the percentage of consumed energy by a transmitted or received packet to $\alpha = 1\%$. Furthermore, the initial duty cycle is fixed to 1.5% with a corresponding delay of $130ms$ and the network lifetime is estimated to be 5000 s, a parameter used by the delay aware forwarding strategy.

The MS is moving with a minimum speed v_{min} of 5 m/s and a maximum speed v_{max} of 20 m/s, while the probability of inversion $p_{inversion}$ is fixed to 0.5. The size of the network is increased by adding new lines of sensors around the central hole. Furthermore, we keep the distance among consecutive VS fixed, and consequently the number of VSs increases accordingly together with number of $VS_{alerted}$. The various network topologies with the corresponding simulation parameters are described in Tab. 3.1.

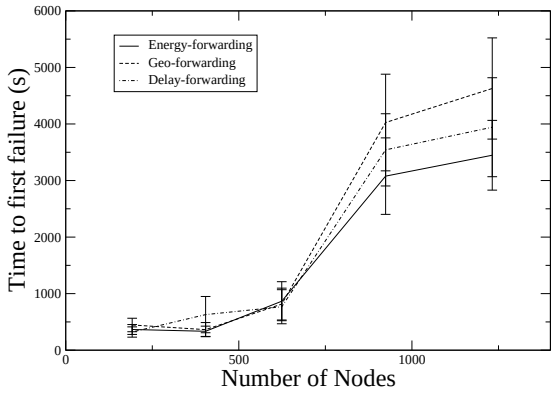


Figure 3.9: Average time to first failure versus number of nodes N with confidence interval of 98%. v_{min}=5m/s, v_{max} = 20 m/s, $p_{inversion}$=0.5, VSs spaced by 125m, 50 runs for each point.

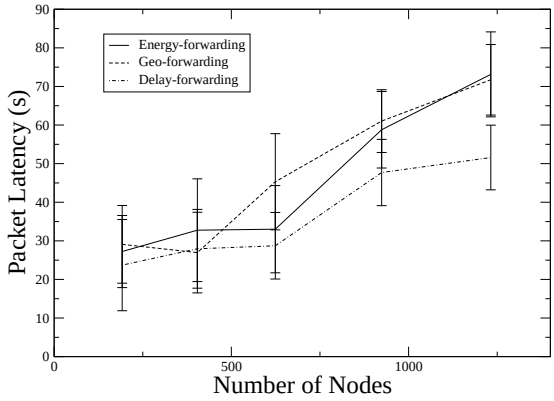


Figure 3.10: Average latency of packets versus number of nodes N with confidence interval of 98%. v_{min}=5m/s, v_{max} = 20 m/s, $p_{inversion}$=0.5, VSs spaced by 125m, 50 runs for each point.

Fig. 3.9 shows the time to first failure with the three forwarding strategies. The results of the simulation have been obtained by running 50 different simulations for each value of N (i.e. the number of nodes in the network) and registering the time in which the first node consumed all its initial energy. It can be observed that, as the size of the network increased, load balancing

N	SN	VS	$VS_{alerted}$	$Zone_x$	$Zone_y$
192	176	4	2	650	650
404	384	5	2	750	750
648	624	6	3	850	850
924	896	7	3	950	950
1232	1200	8	4	1050	1050

Table 3.1: Different network topologies with variable number of nodes N : SN is total number of Sensor Nodes, VS the number of Vice Sinks, $VS_{alerted}$ the number of VS alerted by the mobility algorithm and $zone_x$ and $zone_y$ the dimensions of the network.

techniques overcome the simple geographic routing. In particular, for the biggest network under evaluation, *delay-forwarding* has an average time to first failure 10% larger than that of *geo-forwarding*, while *energy-forwarding* is 20% higher. On the other side, for smallest sizes of the network, the three strategies performs almost identically. In fact, in small network (i.e. $N \leq 648$) there are not enough possible routes toward the destination to find an alternative when a node is consuming all its energy, while in biggest network (i.e. $N \geq 924$) load balancing techniques can benefit from several paths and decide for less stressed nodes.

In Fig. 3.10 the latency of packets is plotted for the same set of simulations. It can be observed that in this case, while *geo-forwarding* and *energy-forwarding* performs similarly, *delay-forwarding* decreases considerably the packet latency for any size of network.
In fact, in this strategy the duty cycle is adaptively reduced or augmented according to consumed energy, and consequently, at the beginning of the simulation, less stressed nodes can forward packets with a reduced delay.

Fig. 3.11 shows the number of hops a packet has passed through between query injection and reply reception. It can be easily seen that the hop-count is approximately the same for the three strategies and that it increases with the size of the network. This implicitly shows that *QUERY* and *REPLY* packets do not weigh upon energy consumption among nodes. Furthermore, we can conclude that *ALERT* packets and in general mobility prediction affects the time to first failure parameter and highlight the difference among load balancing strategies and simple forwarding.

We are now interested in evaluating the performance of the three forwarding strategies when varying the mobility conditions of the MS. In particular, we keep fixed the size of the network to $N = 924$, set V_{min} to $5m/s$, V_{max} to $20m/s$ and we vary the probability of inversion from 0.125 to 1. In this set of simulation, we still measure the time to first failure, packet latency

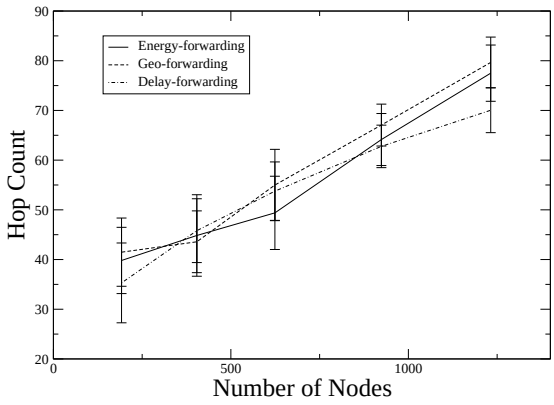


Figure 3.11: Average hop-count versus number of nodes N with confidence interval of 98%. v_{min}=5m/s, v_{max} = 20 m/s, $p_{inversion}$=0.5, VSs spaced by 125m, 50 runs for each point.

and packet hop-count.

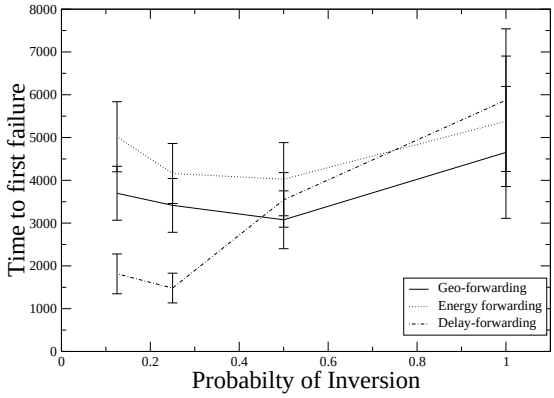


Figure 3.12: Average time to first failiure versus $p_{inversion}$ with confidence interval of 98%. v_{min}=5m/s, v_{max} = 20 m/s, N=924, VSs spaced by 125m, 50 runs for each point.

Fig. 3.12 shows the *time to first failure* while increasing the probability of inversion at the MS. At first, we observe that *energy-forwarding* always outperforms *geo-forwarding*, while *delay-forwarding* has a shorter time to first failure for small network and a longer one for

larger networks.
Then, it is interesting to observe that we have a time to first failure that decreases toward a floor value that for simple forwarding and energy aware forwarding is reached at $p_{inversion} = 0.5$ while for the delay aware strategy is reached at $p_{inversion} = 0.25$. From this point on, we observe an increase in the time to first failure. This can be explained considering that, since for high $p_{inversion}$ values the node is often changing its direction while remaining close to a VS. Therefore, after notifying a VS of its position the MS does not perform an inversion before reaching another VS. In such a way, the alerting strategy does not starts and less packets are injected into the network, with a consequent saving of energy.

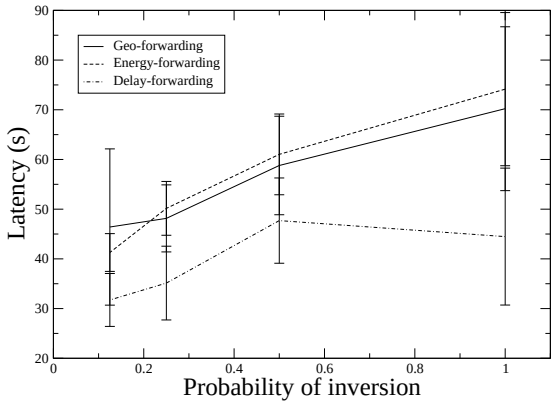


Figure 3.13: Average latency versus $p_{inversion}$ with confidence interval of 98%. v_{min}=5m/s, v_{max} = 20 m/s, N=924, VSs spaced by 125m, 50 runs for each point.

At the same time, in Fig. 3.13 we observe how increasing the $p_{inversion}$ the latency of packets increases accordingly, as well as the hop-count, as it can be seen from Fig. 3.14. This is a confirmation of the above explained fact: by increasing the $p_{inversion}$, the mobility strategy injects into the network a smaller number of *ALERT* packets and in such a way, VSs that are reached by a reply packet happen to be not aware of the movement pattern of the MS. The *REPLY* packet starts then to follow the MS without information on its mobility and in such a way the latency of a packet is increased. This is a confirm of the goodness of the mobility prediction strategy, but at the cost of a higher energy consumption.

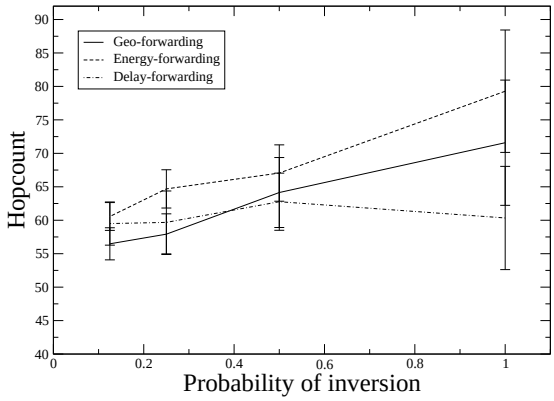


Figure 3.14: Average hop-count versus $p_{inversion}$ with confidence interval of 98%. v_{min}=5m/s, v_{max} = 20 m/s, N=924, VSs spaced by 125m, 50 runs for each point.

3.3 A Multi resolution data management scheme

3.3.1 Overview

The considered application scenario consists of mobile users moving around and running context-aware applications. Contextual information is originating from sensor embedded in the environment, and is gathered by mobile nodes when in the mutual communication range of sensors. We are therefore considering a network scenario where the architecture has 2 distinct tiers [50]. The lowest layer is constituted by miniaturized devices embedded in the environment; the task of such sensor devices is to measure a physical phenomenon upon demand and to transmit the collected information to mobile nodes within radio range. Opposed to sensor nodes, the other layer is made by mobile users, which carry with them personal handheld device, where context-aware services are running. Such services are fed with information originating from sensors and dynamically build what is usually referred as the *context* for user-situated services [61].

Furthermore, in order to diffuse environmental information with no need for dedicated network infrastructures, epidemic information dissemination techniques are employed [50], [51],[52]. A central issue in order to apply such techniques, anyhow, is to make a parsimonious use of the limited resources of mobile nodes, i.e., storage and battery. For this scenario, the original contribution of this paper is an entire *location-dependent degrading data man-*

agement model, which operates a wavelet-transform of data with a variable compression ratio (and thus memory allocation), depending on the current location of the mobile user. This let us trade off accuracy of the information for the resources needed to store it. Notice that this is a reasonable trade-off, since context-aware services are expected to require precise information on the surrounding environment, and a brief summary of outer regions.

The same degrading storage model is then exploited for efficiently exchanging the stored information between two mobile users encountering while moving.

3.3.2 Wavelets Decomposition

Wavelet techniques have been widely adopted in image and audio processing for dynamically varying the resolution and compression of data, or for efficiently denoising signals. As an example, the wavelet transform is the basis of the video MPEG and image JPEG standards [62], [63]. The wide adoption of JPEG standard is a testament of its success. A similar approach is at the basis of the $JPEG2000$ [64] standard, which varies the applied wavelet decomposition depending on the target compression ratio.

Wavelets technique have being recently considered in the area of Wireless Sensor Networks for providing in-network wavelet-based aggregation of information originating from sensors [65]. In response to sink queries, only a compressed aggregate information is sent. Since the wavelet decomposition preserves the properties of the original signal (i.e., trends, peaks, average, etc.), the sink can successively ask for detailed information only from the subset of sensor nodes that presented interesting data in the aggregate response. This results in a significant reduction of data traffic circulating in the network and, thus, in an extension of the network life-time. The work of [65] was successively extended by [66], where a wavelet transform, specifically tailored to non-regular deployments of sensor nodes, is defined.

Wavelet analysis represents a powerful tool to obtain a compact and highly flexible representation of signals. Wavelet representation offers significant advantages over, e.g., Fourier series analysis, thanks to its capability of carrying a joint time-frequency representation of the considered signal [67],[68]. A wavelet transform consists in decomposing an input signal over a set of basis of wavelets, which are functions with a *compact support* (i.e., they are defined over a finite and convex interval) and are oscillatory (i.e., their integral over the support is zero), from which it comes the name “wavelets”, small waves.

The Discrete Wavelet Transform (DWT) is the mostly used practical tool in wavelet applications. The DWT uniquely associates a signal (square-integrable over a n-dimensional space $\mathbb{R}^n$) to a sequence of coefficients, representing the projections of the signal over the

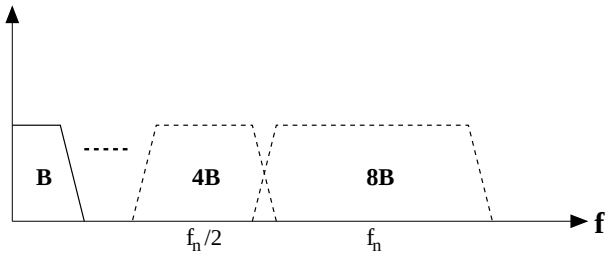


Figure 3.15: Wavelet decomposition in the frequency domain.

basis of $\mathbb{R}^n$. This operation is called *wavelet series decomposition*. Decomposing a signal over a wavelet basis corresponds to cut the original signal in different pieces and analyze each piece separately. Each piece is cut by means of a fully scaled and modulated window, which is applied over the signal to be transformed. This is also referred in the literature as *Multi-Resolution Analysis (MRA)* [67]. Representing the full original signal information would require an infinite number of wavelet functions (and coefficients), thus being able to analyze the original signal over the full range of frequencies and for all its time duration. In order to overcome this problem, wavelet decomposition is conducted down to a certain frequency f^*, whereas all the lower frequencies are "wrapped up" in a $trend$ component obtained by applying a low-pass filter. The described process is briefly summarized in Fig. 3.15.

The $k-level$ multiresolution analysis of an input signal f can be synthetically summarized as follows:

$$f \rightarrow A^k + D^k + \ldots + D^2 + D^1, \tag{3.6}$$

where A^k represents the $trend$ component of the k-MRA, while D^i the $detail$ component of the $i-th$ level of decomposition. It is important to notice that it is possible to reconstruct perfectly the trend coefficients at level $k-1$ by applying the inverse DWT to A^k and D^k. It follows that it is possible to reconstruct perfectly the original signal starting from (3.6).

A similar approach applies to the 2-dimensional case. In this case, the 1-level wavelet decomposition of an input signal f can be summarized as follows:

$$f \rightarrow \left(\begin{array}{c|c} LL & HL \\ \hline LH & HH \end{array} \right), \tag{3.7}$$

where (a) LL is computed by calculating the trend along the rows of f, followed by computing the trend over the columns; (b) HL is the subimage computed by calculating the trends along the rows, followed by computing the fluctuations along the columns; (c) LH is the subimage

computed by calculating the trends along the columns, followed by computing the fluctuations along the rows; (d) HH is the subimage computed by calculating the fluctuations along the columns, followed by computing the fluctuations along the rows.

In the rest of the paper, we will use, for our numerical results, the Haar wavelet [68]. The Haar wavelet is indeed the simplest wavelet function, and enables a rapid implementation while capturing all the advantages of DWT.

3.3.3 Compressing a Random Field

Problem Formulation

Let us associate the surrounding context to a physical phenomenon to be monitored. This physical phenomenon is modeled as an isotropic random field X, defined on a suitable probability space $\{\Omega, \mathcal{F}, \mathbb{P}\}$ [69]. Assuming a two-dimensional space, the process X can be written as $X(x, y, t)$, (x, y) representing a location and $t \in [0, +\infty)$ the time index.

We assume the Space/Time Random Field (S/TRF) X to be stationary and ergodic, which eliminates the possibility of any spatial or temporal trend. While this assumption may look restrictive, it is generally possible to separate the trend component from the stationary one, and deal with them separately. In this work, we will concentrate on the stationary component, which accounts for the variability of the physical phenomenon to be monitored.

Let us assume N^2 sensors to be deployed in a regular fashion (so that their topology forms a grid) over an $L \times L$ square playground, as depicted in Fig. 3.16 for $N = 5$. Without loosing in generality, let us further assume N to be a power of 2. The complete sensor status, representing the contextual information, is stored in the *Sensor Map* (SM), a square matrix of size $N \times N$:

$$SM = \begin{pmatrix} s_{1,1} & s_{2,1} & \dots & s_{N,1} \\ \vdots & \vdots & \vdots & \vdots \\ s_{1,N} & s_{2,N} & \dots & s_{N,N} \end{pmatrix}, \tag{3.8}$$

where $s_{i,j}$ represents a sample of the random field at position (i, j) over the $N \times N$ square grid. As we are not interested in monitoring historical trends of data, we neglect for the moment the temporal domain and simply assume that a value of the sensor map corresponds to the most recent reading from the corresponding sensor. We are now interested in understanding how well DWT can be used to compress the SM matrix.

For the sake of simplicity, we limit our analysis to the case of the field X constituting a Gaussian random field with zero mean and unit variance. The process X is therefore com-

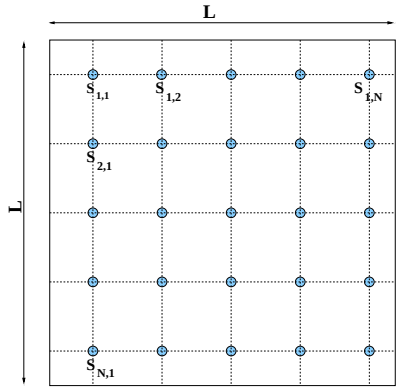


Figure 3.16: Example of grid deployment of $N^2 = 25$ sensors over a $L \times L$ square playground.

pletely specified (in the statistical sense) by its covariance matrix K. [3].

As it is commonly assumed for environmental processes [69], we adopt separable space-time models, so that for two locations s, s' and two time epochs t, t' the process normalized covariance can be written as:

$$\rho(s, t; s', t') = \rho_1(s, s')\rho_2(t, t'), \tag{3.9}$$

where $\rho_1(s, s')$ and $\rho_2(t, t')$ are the normalized covariance of the spatial and temporal components, respectively. Further, we have assumed that the spatial component of the covariance follows an exponential (isotropic) decay:

$$\rho(\zeta) = \exp\left(-\frac{\zeta}{\lambda}\right), \tag{3.10}$$

where ζ is the (Euclidean) distance between two points on the grid and the parameter λ describes the rate at which correlation varies. The parameter λ is usually referred to as the "correlation length" or "correlation scale" of ρ. We assume also to sample the temporal component of (3.9) with a sufficiently large period, so that we can consider the space/time random field as a stochastic image, where time is seen as a quality index of each spatial component in terms of freshness of data.

We have used stochastic simulations [71], [72] for generating alternative, equally probable,

[3] It is worth recalling that the gaussian RF presents the highest entropy of *any* random field with zero mean and covariance matrix K [70]. Hence, sine we are interested in compressing the SM and since to lower entropies correspond a better compressibility, the gaussian assumption represents an upper bound.

realizations of the space/time random field. By adopting a grid topology for the sensors, each realization represents a possible *stochastic image* of the space/time distribution of the random field, and satisfies the constraints imposed on the covariance model.

Varying the correlation length results in a different *entropy* [70], and, thus, in a different compressibility of the stochastic image. The entropy decreases while increasing the correlation length [70].

Compressing the Random Field Through Standard Wavelet Techniques

An important property of the wavelet transform is the *compaction of energy*. Although the wavelet decomposition maintains the total energy of the original signal, typically, the trend component accounts for a large percentage of the energy of the transformed signal. This results in the magnitude of its coefficients being significantly larger than the magnitude of the detail ones.

We considered the repartition of energy among trend and detail components for various levels of decomposition. Varying the level of decomposition consists in stopping the iterative DWT decomposition at a certain level k. As an example, in Fig. 3.15 we could stop the decomposition at the first iteration, with the detail component obtained with a $8B$ band-pass filter and the trend with a $\frac{f_n}{2} + 2B$ low-pass filter, or further iterate the process to deeper levels.

The considered setting encompasses a 64×64 sensors, equally spaced by a distance of $D_S = 20$ m and distributed over a 1280×1280 m^2 square playground. The results obtained averaging 50 simulations are reported in Fig. 3.17 for $\lambda = ND_S$ (the entire playground size). As it is intuitively clear, most of the energy is concentrated in the trend component. This effect is particularly evident for low decomposition levels. When we attempt to squeeze the energy into ever smaller time intervals (higher band-bass filters) some energy inevitably leaks out.

In general, wavelet-transformed images are then compressed by means of *Thresholding Schemes*, which consist in comparing each DWT coefficient with a fixed threshold T_h, and in keeping only the coefficients above the defined threshold, while setting to 0 the remaining ones [4]. The processed signal results to be composed by a high number of 0s, and can then be compressed adopting standard compression methods (i.e., run-length encoding, etc.) [73], [74]. Obviously, by selecting a large threshold value, a high compression rate is achieved at the cost of a corrupted reconstructed image. The main objective in wavelet compression is

[4] We are not considering soft thresholding schemes, since they are typically applied to image denoising, rather then image compression.

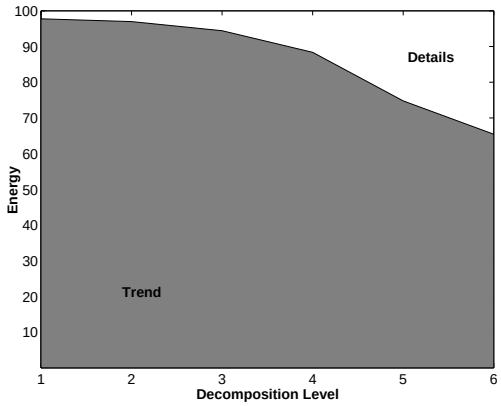


Figure 3.17: Energy repartition between the trend and details components at various decomposition levels.

selecting a good threshold value, depending from the specific application scenario considered, that can guarantee either a given compression rate, or a chosen quality of the reconstructed image.

In our case, we decided to select a threshold so that at least 95% of the energy of the original sensor map is kept, applying the method in [73].

We now consider the percentage of detail coefficients that are set to zero at each level of decomposition by applying the described thresholding scheme. In Fig. 3.18, the number of zeros in the case of a 64×64 sensors regularly deployed over a 1280×1280 m^2 playground are presented in the case of a correlation length of 320 m and 1280 m. Results are obtained averaging 50 simulations. Clearly, different correlation lengths lead to different compression ratios, since the correlation length influences the resulting entropy of the random field. As we can see from Fig. 3.18, the first levels of decomposition are characterized by an extremely high percentage of null values in the detail coefficients and this percentage increases with larger deployments (i.e., larger values of N) or higher correlation lengths [75].

The Multiresolution Data Compression Scheme

From the previous analysis it possible to conclude that, when compressing a sufficiently high correlated random field and for sufficiently large deployments ($N \gg 1$) [75], only a minimum amount of energy of the original signal will be lost when removing the details coefficients. This suggested us an alternative approach that we termed *Multiresolution Data Compression (MDC)* scheme. The MDC scheme consists in removing all the details coefficients up to a cer-

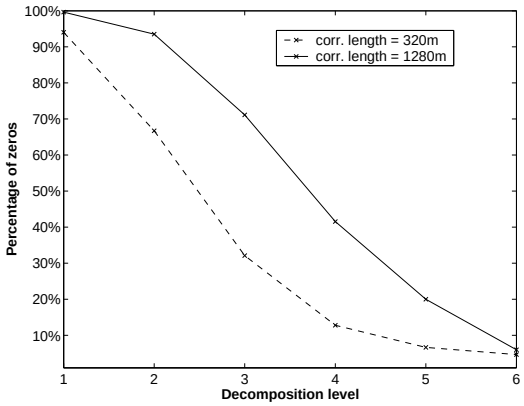


Figure 3.18: Percentage of zeros in the details components applying an adaptive thresholding.

tain level of decomposition, and in keeping only the trend component of the DWT transform. The level of decomposition reflects the compression ratio of the sensor map (and therefore the memory required for storing the compressed image), since the deeper is the decomposition level, the more are the coefficients that will be dropped. As an example, let us consider an $N \times N$ image, with N power of 2. The lowest compression ratio can be obtained by performing a single level DWT of the SM, and maintaining the resulting trend component: the trend matrix is a $\frac{N}{2} \times \frac{N}{2}$ matrix. Conversely, the highest compression ratio can be obtained by iterating the DWT down to the maximum decomposition level, which is $log_2\,(N)$ for N power of 2. In this case, the trend matrix consists of a single coefficient, which represent the maximum compression achievable with the MDC scheme. Obviously, the deeper the level of decomposition, the higher the compression rate at the cost of a lower quality of the reconstructed image.

The benefits deriving from the MDC mechanism are twofold. First, it allows for a great level of control over the memory resources allocated for storing the compressed image. In fact, at each level of decomposition the trend component has a well defined number of coefficients; second, it presents an extremely low-complexity implementation, and can therefore run on processing-constrained devices. This allows us to easily trade off information accuracy with resources, depending from the specific application scenario.

Ideally, the processed signal is then transmitted and reconstructed on the receiver side. The reconstructed signal will be an estimate $\widehat{S}$ of the original stochastic image S. In fact, while reconstructing the signal, all the details components that were previously removed are set to zero, and this obviously results in a degradation of the reconstructed image. In order to

quantify this degradation we considered the distortion D, defined as:

$$D = \frac{1}{N^2} \sum_{i=0}^{N} \sum_{j=0}^{N} (S_{i,j} - \widehat{S}_{i,j})^2, \tag{3.11}$$

where $S_{i,j}$ and $\widehat{S}_{i,j}$ are, respectively, the elements (i, j) of the original and reconstructed sensor map, and N is the size of the SM matrix. In Fig. 3.19, the distortion of the reconstructed

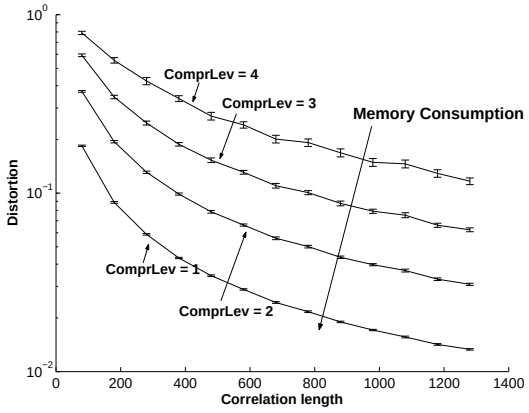


Figure 3.19: Distortion of the reconstructed stochastic image varying the compression ratio.

signal $\widehat{S}$ of a 64×64 sensors image, with sensors equally spaced by $D_S = 20$ m and distributed over a 1280×1280 m^2 playground, is depicted. The setting encompasses a variable correlation length λ and a variable level of compression $ComprLev$. Results are obtained running 50 simulations and considering the corresponding 98% confidence interval. The $ComprLev$ represents the decomposition level down to which details coefficients are discarded. As it is intuitively clear, for any fixed correlation length λ the distortion increases while increasing the compression level of the image (thus reducing the memory resources needed for storing the SM), since we are discarding a higher number of DWT coefficients. Additionally, for any given compression level, increasing the correlation length results in a higher correlation of the sampled random field and, thus, in a higher significance of the trend (low-pass) componenet. This affects the capacity of the Haar DWT to representing the RF, an is reflected in a reduced distortion of the reconstructed signal.

3.3.4 A Multi-Resolution Data Management Scheme

Scheme Description

Let us assume N^2 sensing devices to be uniformly deployed over a $L \times L$ square playground, with the aim of providing a precise estimation of the random field X when asked to. Let us further assume the playground to be logically divided into $T \times T$ square tiles, each tile containing $l = \frac{N^2}{T^2}$ sensors. M mobile nodes are moving over the playground and reading sensor nodes when in mutual communication range. Each sensor reading is constituted by the tuple: $< value, time, pos >$, with $value$ being the reading of the sensor, $time$ the relative reading time [5] , and pos the sensor location [6]. The gathered information is then stored in the internal memory of the mobile users' portable device.

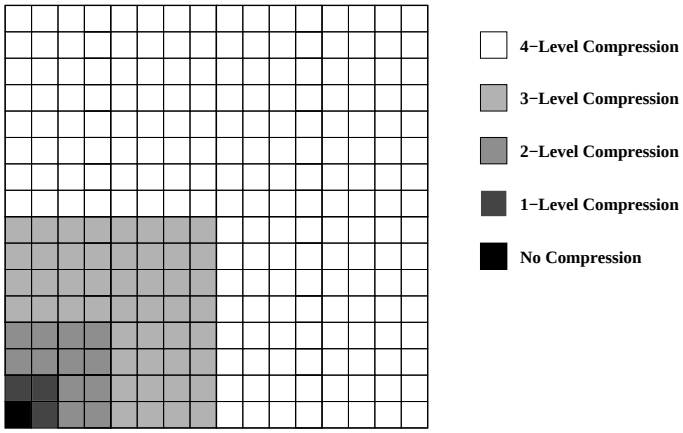


Figure 3.20: Tiling and multi-resolution structure applied over the playground.

Multi-Resolution Data Management Scheme

In order not to exhaust the limited storage resources of mobile nodes, information is managed according to a *Multi-resolution Data Management (MDM)* scheme , which implements a lossy and location-dependent degrading storage model. Starting from its current position, each mobile node builds a quad-tree hierarchy of the stored information, compressing different tiles

[5] This does not require all nodes to be synchronized to a common clock, since we are only interested in storing the *reading age*, which can be calculated at any time instant by subtracting the timestamped value from the local clock.

[6] Sensors' position can be programmed during the installation phase, or, given a limited sensors communication range, simply assumed as the mobile user position.

along the quad-tree with different compression ratios. Finer representations will be kept for localized information, while only coarse approximations will be left for remote regions. Hence, each tile is wavelet transformed and compressed separately. Different compression ratios are achieved by varying the wavelet decomposition level, as detailed in Sec. 3.3.2. This is briefly summarized in Fig. 3.20, where the scale ranges from black to white, with black corresponding to the maximum resolution (no compression), and white to a coarser approximation of the original signal (maximum compression). As an example, Fig. 3.20 presents the information view of a mobile user positioned in the left corner tile.

Clearly, by adopting a tiled management of the stored information, we are trading off the efficiency of the Discrete Wavelet Transform with the level of flexibility that is needed for applying different compression ratios to information deriving from different spatial regions. Threating the entire sensor map as a single *image* would lead to better (but flat) compression ratios, without the possibility of separately compressing/decompressing portions of the overall SM.

The information is stored in the *Information Map* (IM), which is a $T \times T$ tuple matrix containing the overall information stored by a mobile user. Except for the tile where the mobile user is moving in, the following fields are associated to each tile contained in the IM:

- *Readings Number (RN)*: represents the number of sensor readings over which the DWT is calculated;
- *Readings Timestamp (RT)*: when reading a sensor, each mobile node stores the reading time, so that it is possible to keep track of the *age* of each reading. The RT is an average over the timestamp of the sensor readings, and represents the "freshness" of the tile information;
- *Compression Ratio (CR)*: is the compression ratio at which the tile is compressed;
- *DWT coefficients (DC)*: contains the coefficients of discrete wavelet transform of the tile, in accordance with the compression ratio CR applied.

No compression is applied instead to the tile where the user is moving in. For this tile only, the complete information ($< value, time, pos >$) on each sensor reading is kept. Indeed, this is the most localized information and, thus, the most important for services running on users' portable device.
The *Information Map Summary (IMS)* is defined as the information map purged from the DWT coefficients DC. Hence, the IMS contains a summary of the information stored in the

IM (the DWT coefficients constitute the largest part of the IM storage), and it will be used, as explained in the following, by mobile nodes for a preliminary exploratory phase of the stored information.

The introduced MDM scheme requires a *Memory Usage (MU)* that asymptotically scale as $\Theta\left(N \log_2 N\right)$ [75]. This represents the tradeoff between the storage resources and quality of the information, and can be efficiently mapped to the requirements of the specific application scenario requirements.

Mobile nodes are exchanging their IMs according to a simple P2P protocol, whenever in mutual communication range [75]. When merging the received IM, the most updated information is kept (greater Readings Timestamp and Readings Number).

Performance Evaluation

In order to evaluate the performance of the proposed multi-resolution data management scheme, we have run extensive simulations utilizing a freely available simulation tool [60]. We considered a square playground of size $L = 2.56$ km, over which 128×128 sensors are uniformly deployed in a grid fashion. A variable number M of mobile nodes are moving over the playground, and are applying the multi resolution data management (MRDM) scheme, for storing data read from sensors, and exchanging the stored information at every communication occasion. The MRDM scheme is implemented by means of Haar discrete wavelet transform, with a 8×8 tiles structure. Hence, each tile contains 128 sensors, allowing for 4 levels of decomposition in the MRDM scheme. In our implementation, we applied the MRDM down to level 3, keeping 4 DWT coefficients at the highest compression ratio.
Mobile nodes are moving according to a Random Waypoint (RWP) mobility model [76] with a pausing time uniformly distributed between $(0, 10)$ s, a speed uniformly distributed in $(10, 15)$ m/s. Nodes' initial location is drawn according to the steady-state distribution [77], thus reproducing a "perfect simulation". This eliminates the time needed for reaching a stationary regime.
It is assumed an ideal channel, where mobile nodes can communicate if their mutual distance is less than 75 meters, and sensors are read if located at a distance less than 10 meters. Nodes apply a CSMA/CA for resolving collisions, and communicate at a rate of 11 Mb/s.
The *Weighted Distortion (WD)* is the metric utilized for evaluating the performance of the

system, and is defined as:

$$WD = \frac{1}{MN^2} \sum_{i=1}^{M} \sum_{j=1}^{N^2} w_{i,j}(R_{i,j} - S_j)^2, \tag{3.12}$$

where $R_{j,i}$ is the value stored by node i on sensor j, S_i is the value of sensor i. $w_{i,j} = \exp(-\gamma d_{i,j})$ with $d_{i,j}$ being the spatial distance of node i by sensor j and γ the weighting memory. Hence, γ is an exponentially decaying correcting factor taking into account the weight that context-aware applications apply to information: higher for local data, lower for information originating from remote locations (the lower γ, the larger the impact of remote data on the WD). The weighting memory is application-dependent, and tunes the level "of information locality" that the specific application scenario can tolerate.

As a first step, we considered a static scenario, with sensor nodes taking values according to a static Random Field $RF = C + X_0$, with $C = 4$ and X_0 a sample of the Gaussian random field. In Fig. 3.21, the distortion D is depicted in the case of a variable number of mobile nodes, a correlation length of 1280 and no weighting memory ($\gamma = 0$). It can be seen that a higher number of mobile nodes leads to a better performance of the system. An initial phase is needed for the mobile nodes to first read the sensors, and then diffuse the gathered information. This is reflected in the distortion being constant for the first 100 s, and then dropping significantly down. Fig. 3.21 presents also the lower bound, calculated as:

$$LB = D_0 + \sum_{i=1}^{log(N)} (D_i \cdot 3 \cdot 4^{i-1}), \tag{3.13}$$

where D_i is the distortion at decomposition level i obtained averaging 100 Matlab simulations. It is possible to observe that, in the long run, the distortion approaches the lower bound although never reaches it. This fact can be easily explained as a consequence of missing information: it is very likely that nodes, when leaving (and compressing) a tile, will have only the information on a subset of sensor nodes. Hence, only this partial information will be compressed and determine the distortion of the reconstructed image. Conversely, the LB is calculated under the assumption that information on all sensors within the tile to be compressed is available.

We have then introduced a dynamic scenario, with sensor nodes taking values according to:

$$RF = C + sin(2\pi T_f t) + X, \tag{3.14}$$

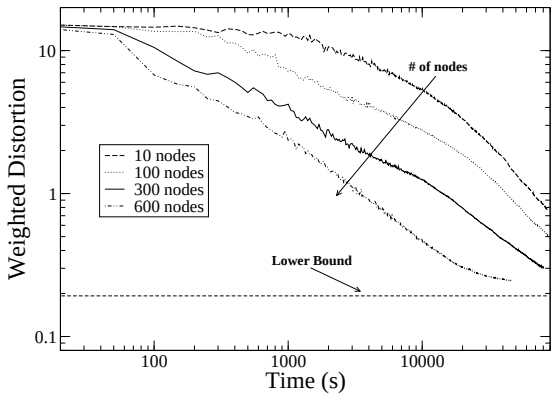


Figure 3.21: Distortion in the case of a variable number of mobile nodes moving according to the RWP mobility model, and a static RF.

with C being constant, $sin(2\pi T_f t)$ a periodic deterministic temporal component and X a Gaussian random field (with zero mean, unit variance and an exponentially decaying, in the space domain, covariance) updated every T_{RF} seconds. For our simulations, we set T_f to 1 day, C to 4, the exponential decay factor of the random field to 1280 and the weighting memory γ to 0.001.[7]
In Fig. 3.22, the WD is depicted in the case of 100 and 600 mobile nodes, and a random field varying every $T_{RF} = 600$ s. After an initial bootstrap phase, needed for the MRDM scheme to adjust to the average value C, the system is able to track the RF changes. Clearly, given the limited time available for adjusting to new RF values, the WD is constantly fluctuating, with peaks corresponding to RF updates.

Finally, Fig. 3.23 presents the WD in the case of 600 mobile nodes moving according to the RWP mobility model and a RF update time T_{RF} of 60, 600 and 6000 s. Clearly, the lower T_{RF} the less the time available for the system to read diffuse sensor information. This is evidently clear from Fig. 3.23.

[7] By setting $\gamma = 0.001$, we are weighting approximatively $\frac{1}{3}$ information originating from sensors that are 600 m away from the current position of a mobile node.

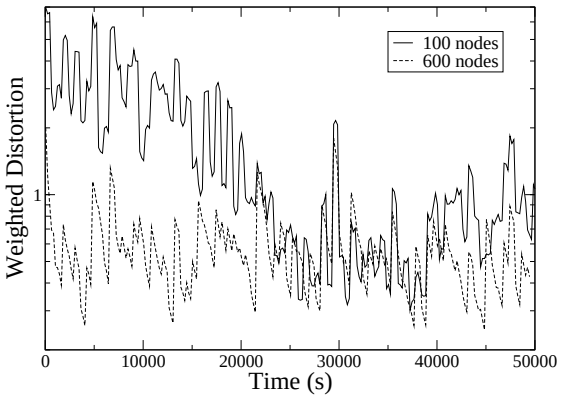


Figure 3.22: Weighted distortion in the case of 100 and 600 mobile nodes, and a RF varying every 600 s.

3.4 Closing Remarks

In this chapter, we have initially proposed a system architecture and the related routing framework for a large-scale WSN, with users moving around it and querying the network. This application scenario well represents the case of cars moving around a hot parking place area, where sensors able to detect a parking place to be free or occupied organize themselves in a WSN. MS can in general experience periodical disconnections from the network, depending on the deployment of nodes along the road where they are moving. In order to efficiently react to unpredictable mobility pattern of the MSs we have couple a geographic routing strategy able to overcome holes with a mobility prediction strategy.

We have introduced in the first part of this chapter also two simple load balancing techniques: an energy-aware forwarding strategy and a delay-aware forwarding, where, energy and delay are taken into account when deciding about the next hop, respectively. Such strategies have been tested and compared to simple geographic forwarding while varying the size of the network and the mobility of the MS.

A simple scenario has been considered, with SNs regularly deployed in a grid disposed around a hole representing a building or a trading centre and VSs regularly disposed along a road where MSs are moving. We have shown by means of an extensive simulation study that the proposed solutions enable the introduction of novel intelligent transportation system applications, with different results for energy-aware, delay-aware and simple geographic forwarding in terms of time to first failure of a node and latency of packets.

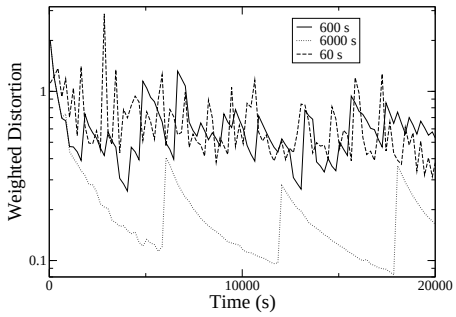


Figure 3.23: Weighted distortion in the case of a 600 mobile nodes and a random field varying every 60, 600 and 6000 s, respectively.

We have furthermore addressed the scalability problem deriving from the management of context data originating from sensors embedded in the environment. We have defined Multi-Resolution Data Management scheme that trades off the resolution of data for its storage resources. The mechanism has been implemented and evaluated by means of a simulative study. Results show that for a sufficiently large number of mobile nodes, the proposed scheme is capable of tracking the random field changes, while significantly reducing the storage and communication resources required on mobile nodes.

4 Application to Mobile Social Networks

4.1 Introduction

The proliferation of mobile technologies (such as mobile phones, gaming consoles and MP3 players) equipped with short-range wireless connectivity (such as Bluetooth and WiFi) has encouraged the development of applications that allow users to produce, access and share digital resources in a wide number of everyday occasions and without the support of a fixed infrastructure [78],[79],[80]. The development of such applications presents challenges in terms of both user interaction and technical feasibility, as users' behaviors needs to be taken into account, especially the mobility of users and the variability of contexts traversed, and technological limitations, especially in terms of battery/processing power and wireless bandwidth, cannot be underestimated.

From an user-interaction perspective, applications have addressed the possibility for users to access data from certain locations (location-based) or to share data with other users in proximity (mobile peer-to-peer).

From a technical perspective, the use of opportunistic communication systems [6] to support mobile peer-to-peer applications[49],[48], exploiting the proximity of mobile nodes for exchanging personal or contextual information, is becoming an increasingly popular research topic. Originally, such a paradigm emerged as a way to provide connectivity in intermittently connected scenarios, such as the cases of Interplanetary Internet and developing regions [81]. In cases where connectivity could not be taken for granted, nodes were able to temporarily buffer data, and forward it at the next communication opportunity. The paradigm applied was therefore a "store-and-forward" one, where nodes were expected to first store, and then forward any information destined outside the local network.

Recently, significant attention has been devoted to mobile application scenarios, where the carriers of information are represented by people with their personal handheld devices. In this case, the characteristics of the social network in which the data is being diffused is extrememly important and can significantly influence the performance of such systems. This consideration has led to the studies on how to exploit social interactions in mobile systems. In [44], the community structure behind the social interactions driving the data diffusion process has been studied in order to improve the performance of forwarding algorithms. Starting from experimental data sets collected in real-world experiments, it was shown that it is possible to identify a limited set of nodes that were much more active than the others. Such nodes represent *hubs* of the network. Further, the authors show how it is possible to divide the network into overlapping communities of nodes, and how these communities depend on the social relations of the nodes themselves. Finally, it was shown that by incorporating the knowledge of both the social activity of the nodes and the communities to which they belonged, an extremely efficient trade-off between resources and performance could be achieved.
The relevance of users' behavior in opportunistic networks was further assessed in [47], where the impacts of different social-based forwarding schemes were evaluated in the case of a DTN routing protocol scenario, a worm infection scenario, and a mobile P2P file-sharing system application scenario. Also, in this case, the evaluation was based on real world mobility patterns, obtained from bluetooth proximity measurements. In particular, by applying a threshold on the periodicity of meetings it was possible to classify encounters in terms of strangers or friends, and analyze the properties of these 2 classes of meetings separately. Starting from this classification, the authors conducted a statistical analysis of these 2 categories of meetings, concluding that (i) while most of the encounters were between strangers, meetings among friends account for almost two-thirds of the overall meetings, (ii) networks of both strangers and friends were scale-free and (iii) the network of friends had a very high-clustering coefficient. Finally, the authors showed that incorporating this friend/strangers distinction in the forwarding policies can be beneficial in different application scenarios, such as P2P file sharing or in the prevention of the spreading of worms.

What appears to be missing from current studies on opportunistic communication systems is the realization that while mobility and proximal encounters might provide opportunities for improving the performance of the network, these mobile encounters are inherently wed to the social world in which they take place. Therefore, opportunistic communication systems must take into account the types of applications which they are meant to support, and to recognize

the significance of the users' preferences regarding the data that they are interested in accessing and sharing through these systems. Further, with respect to real-world implementations of such systems, we believe it is also important to acknowledge the effects of current technical limitations (such as the bandwidth provided by short-range wireless technologies like Bluetooth) on the potential performance of an opportunistic communication based application scenario.

In this chapter we present a study which is a first step towards addressing such concerns; we take into account both technical limitations of the specific technology adopted, as well as the preferences expressed by users. It is presented an opportunistic content distribution application scenario, where information is diffused solely by means of opportunistic P2P data exchanges among the users participating in the service.
The main contribution of this chapter it is provided here is as follows. In Sec. 4.2 we present a system architecture and the relative software implementation for bluetooth enabled java phone. In Sec. 4.3 we introduce a system model on top of the presented software architecture. Here, we present at first an analysis of the connectivity patterns emerging between people in office environment which have been recorded by means of off-the-shelf technologies. This provided insights into the real-world performance level of opportunistic services running over commercially available devices such as smartphones. Next, we gathered actual user-expressed preferences through the means of a questionnaire. Third, we combined these technical and social parameters to develop a simulator, capable of reproducing the measured pattern of inter-user contact, and of emulating the content distribution network based on the preferences of the users. This allowed us to see not only when data *could* be distributed, but when it actually *would* be, and we were able, to investigate how the size and format of different types of content significantly impacted the diffusion process. Finally, starting from the assumption that people tend to share information within communities of similar interests, we develop the concept of *affinity*, which is a metric measuring the similarity of users' preferences. We analyze how such an affinity metric influences the structure of the network, and the circulation of the content. By combining user constraints and realistic contact patterns, we are able to accurately simulate how content is distributes over the network, thus providing valuable insights into the design and dimensioning of opportunistic content distribution systems.
On top of this analysis, we present in Sec. 4.4 a class of opportunistic data diffusion algorithms, which we call k–friends–based relaying. The idea is the following one. Each node selects a set of k "nearest neighbours", based on the contact pattern (number of contacts within a given timeframe and their duration). It then builds a list of interests by merging its own

ones with the ones of its best friends (properly weighted and/or ordered). Upon a meeting, it communicates such list to the other node, asking for data matching such interests. In this way, it aims at receiving both the content it is interested in and the content its nearest neighbours are interested in.
Finally, in Sec. 4.5 the introduced framework is used in localized mobile community for the evolution of a service such as the intuitive dictionary function for mobile phones, i.e. the *t9* service.

4.2 Software Architecture

Opportunistic networking [6] refers to the possibility of delivering data applying an epidemic-like forwarding mechanism, without the need for any dedicated infrastructure. Such communication paradigm received great attention in the last few years as an emerging technology for disseminating data in challenged environments, where due to environmental constraints it is not possible to build an alternative communication infrastructure, or in pervasive environments, where data exchanges are driven by the "social interactions" of mobile users [82]. In particular, the latter case is a direct consequence of the fact that mobile devices (e.g., smartphones or PDAs) are nowadays largely available among people and of the constantly increasing computing, communication and storage power of such devices. Several mobile phones are in fact equipped with Bluetooth, Wi-Fi and Wibree (in the near future), technologies that are directly accessible for programmers through freely available and easy-to-use APIs. Further, mobile phones are now capable of intensive processing operations and of storing large amounts of data in their internal memory.
However, although opportunistic networking has been deeply investigated from a theoretical point of view, only few real deployments have been proposed, with the main goal of understanding the networking performance of the implemented protocols [83] or the social aspects related to proximity communications [82].
Following these considerations, we have developed a *User-centric Heterogeneous Opportunistic Middleware* (U-Hopper), running on any java-enabled smartphones and leveraging Bluetooth connectivity for exchanging data. Such platform combines the user preferences with the requirements imposed by the pervasive services hosted on the users' portable device. The final goal is that of gathering and disseminating data in a totally transparent way to the user.

4.2.1 System Overview

The considered system architecture consists of two classes of nodes: *User-Nodes* (U-Nodes), which are resource-rich mobile devices (e.g., smartphones, PDAs) carried around by users during their daily activities, and *Tiny-Nodes* (T-Nodes), which are resource-constrained devices embedded in the environment and providing localized information [84]. A part from their technological differences, the 2 classes of devices play a different role in the network. T-Nodes act as *providers of information*, constantly broadcasting localized information such as advertisements, or snapshots of the surrounding environment. Conversely, U-Nodes act as *consumers of information*, reading T-Nodes in their communication range, and augmenting pervasive services with such data. The data generated by T-Nodes is first stored in the U-Nodes internal memory, and then diffused by means of opportunistic peer-to-peer data exchanges. Users' mobility is therefore exploited in order to achieve system-wide communications. In this demo,

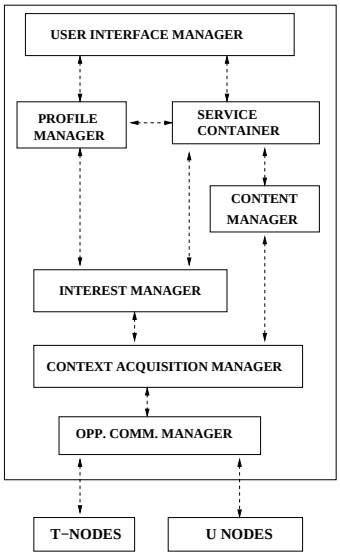


Figure 4.1: Block diagram and interconnections of the U-Hopper platform.

we will present a U-Hopper, a User-centric Heterogeneous Opportunistic Middleware initially introduced in [85]. Such platform resides on U-Nodes and exploits any proximity communication interface (i.e., Bluetooth, Wi-Fi) in order to (i) gather localized information originating from T-Nodes embedded in the environment (ii) opportunistically disseminate the stored data to other U-Nodes. The information diffused includes data received from T-Nodes as well as any other information shared by the user (e.g., music, videos, etc.).

As depicted in Fig. 4.1, the U-Hopper middleware is composed by six distinct components. The *User Interface Manager* (UIM) handles any human-to-machine interaction such as data insertion and visualization. The *Service Manager* (SM) is the execution environment where pervasive services are running. This component allows services deployment, deprecation and update.
The *Interest Manager* takes into account (i) the user preferences and (ii) the requirements deriving from the pervasive services hosted by the SM, and produces a list of "interests", which are a high-level description of the information the user is interested in. Such interests regulate the way according to which information is exchanged between any 2 U-Nodes accordingly.
The *Content Manager* (CM) manages the persistent storage available on mobile devices. In particular, it provides context data insertion/deletion, update and search functionalities. In addition, the CM runs appropriate "data aging" algorithms that are needed in order to discard outdated information and preserve the available resources. Such techniques implement information filtering rules that trade off data locality (both in the time and space domains) for available resources (i.e., storage, communication, etc.) [86]. As an example, we can think at a special sale offer ending at 5 pm of the current day. Clearly, as soon as the offer is no longer valid, it is useless to store the corresponding information. The CM is in charge of detecting such situations and of determining when to remove data from the users' device permanent storage.
The *Context Acquisition Manager* (CAM) takes into account the information deriving from the Interests Manager, and applies data filters on the incoming and outgoing data flows.
The *Opportunistic Communication Manager* (OCM) monitors the availability of data sources in the surrounding environment, and seamlessly performs any networking operation needed for gathering the discovered data. This includes data originating from T-Nodes as well as from other U-Nodes. Please, refer to [85] for a more detailed description of U-Hopper system components.
Fig. 4.2 depicts the handshake regulating the data exchange between any 2 nodes meeting. The data exchange is triggered by a Node 1 receiving a beacon message used for discovering neighboring peers. In response, Node 1 sends its own interests (Interests 1 MSG), which are a description of the information Node 1 is interested in. Node 2 responds with the data stored in its own internal memory matching Node 1 interests (DATA 2 MSG), and subsequently, with its own interests (Interests 2 MSG). Finally, the data exchange is terminated with Node 1 sending any data matching Node 2 interests. The corresponding system components interaction flow, when generating the user interests, is presented in Fig. 4.3. When a beacon message is

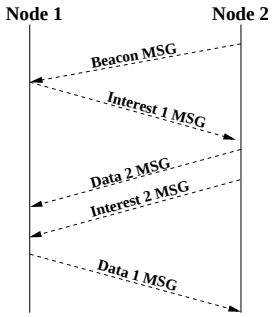


Figure 4.2: Data handshake between any 2 nodes meeting.

received by the OCM, a request for the user interests is invoked. Such request is then captured by the IM, which gathers the user profile, the service constraints and returns the user interests. It is worth remarking that the described actions are performed by the U-Hopper platform transparently to the user, thus increasing the system usability of the system, since it does not require any human intervention. On the counterpart, when the interests from an encountered

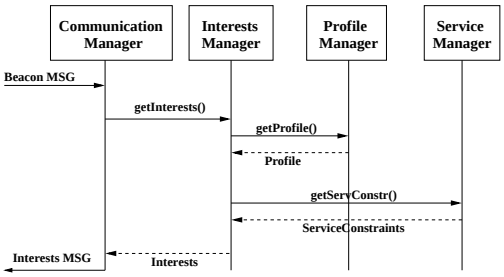


Figure 4.3: System components interaction flow, when generating the user interests.

node are received, the data stored in the internal memory is searched accordingly, and information matching the received interests is send back. The corresponding system components interaction flow is depicted in Fig. 4.4.

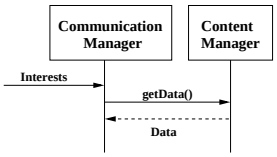


Figure 4.4: System components interaction flow when retrieving data, starting from user's interests.

4.2.2 Implementation Details

In order to embrace the largest number of "potentially available" mobile devices, we selected smartphones as the target platform over which we developed U-Hopper. In fact, smartphones are nowadays typically carried around by users during their daily activities and at the same time, they have reached a sufficiently large computing and communication power to perform very complex operations. We also decided to use some laptops for broadcasting advertisement information or collecting application's statistics.
In order to leverage on a widely diffused and standardized computing environment, we choose to develop U-Hopper as a java Midlet running over J2ME (MIDP profile 2.0) [87], which is currently available on most of the smartphones shipped today. By implementing U-Hopper as a java component, we are guaranteed that the software will be portable over a large set of devices.
The fist technological issue to be solved in developing a middleware for opportunistic environment is given by the selection of the proper network interface. In fact, as detailed in Sec. 4.2.1, opportunistic communications are the primary mean by which information is diffused in the described environment. Currently, Bluetooth is the largest available network interface on mobile phones and can be easily accessed through J2ME dedicated APIs, such as the well-known JSR 82 ([88]). Hence, we choose to rely on this technology for achieving localized peer-to-peer data exchanges among mobile nodes. Obviously, Bluetooth is not properly designed for opportunistic communications, given the amount of time typically required for establishing a connection between two devices. To shorten up this connection time, we leveraged on some assumptions and on few properties of the Bluetooth technology. At first, we have assumed that all the devices running U-Hopper are assigned with a friendly name starting with a given prefix (for instance they all start with 'uhopper'): this simple assumption allows U-Hopper clients to recognize immediately a candidate that runs the same service, avoiding useless attempts of connection with other devices. We assume also that all devices keep a list of recently visited devices in order to avoid too frequent connections with the same peers. A peer is periodically removed from this list, when a given timer is elapsed.
In U-Hopper, each device is always working in server mode, i.e. it is waiting for incoming connections from U-Hopper clients. Alternately, a device can work in client mode, periodically inquiring close-by devices, and consequently inhibiting its server mode until the client operations are not completed. Whenever another device is found during an inquiry, if its friendly name starts with the given prefix and it was not recently visited, the inquiry process is stopped and the service discovery operation is performed on that device. This peer is now

inserted in the list of recently visited devices. If the U-Hopper service is found and it is available, a connection is established and the communication handshake previously described can start. The node working in server mode updates its list of recently visited devices, in order to avoid a new exchange of information with the same peer in a brief time. Following this procedure, two peers can establish a connection in approximately 1 or 2 seconds (depending on the number of Bluetooth devices in radio range), a time suitable for having opportunistic communication on Bluetooth enabled devices.
The second issue we had to face was to design a proper persistent storage on each device. Typically, in smartphones data are saved locally using the Record Management Store (RMS), where information can be easily stored as an array of bytes and retrieved using easy-to-use matching methods, similar to common data base queries. We design then a RMS storage for each device, allowing U-Hopper to maintain persistent data, interests and profiles through different execution of a service running on top of the middleware.
Concerning the implementation of U-Hopper on laptops, we used the Avetana Bluetooth library, an open source Bluetooth stack that allows applications designed for JSR-82 to run on Linux devices. Although we had to adapt some of the components to a not-embedded environment, as for instance the local storage and the user interfaces, most of the modules of U-Hopper were easily installed and run on several laptops. Laptops were used as T-Nodes generating advertisement information or as collection points, for showing application statistics, running the same communication module described before.

4.2.3 The System at Work

A demonstration of the proposed software architecture is depicted in Fig. 4.5. It consists in (i) few smartphones and laptops acting as data sources (T-Nodes), (ii) 4 smartphones running U-Hopper (U-Nodes), (iii) 1 laptop acting as a mobile node (U-Node) for collecting application's statistics. In order to reproduce the considered pervasive application domain, we have used a few Bluetooth enabled smartphones acting as environmental data sources (T-Nodes). Such smartphones are regularly broadcasting physical data, such as measured temperatures or snapshots of the conference site periodically taken by smartphones' cameras. Few laptops are broadcasting context data such as conference social and technical activities. Context data is then collected by other smartphones (U-Nodes) passing by, as for instance *UN1* gathering data from *Tiny Node 1* in Fig. 4.5. Information is then epidemically diffused by means of opportunistic P2P data exchanges among U-Nodes that take place transparently to the user.

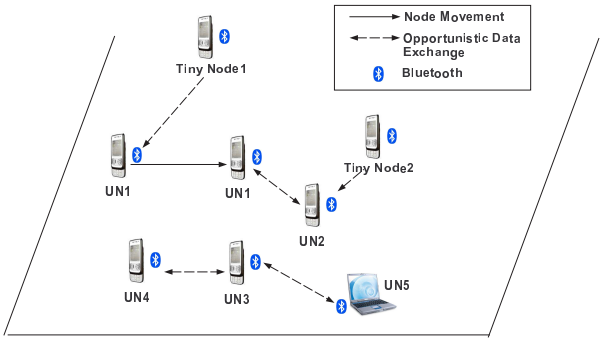


Figure 4.5: Demonstration scenario.

After installing and starting the application, users are asked to select their preferences. Such preferences will influence the data gathering and data exchange process. Users can then visualize the collected information, such as snapshots and advertisement information, at any time on the smartphone's graphical interface Furthermore, U-Hopper users will have the opportunity to share some information in a totally P2P fashion, directly acting as source of data. For instance they can share some personal midi files (e.g. ring tones), exchange them with other peers if interested in entertainment information and then listen to the received files through a simple UI. In Fig. 4.6 two users are shown while using U-Hopper.

In order to show the outcomes of the demonostration scenario (such as collected information

Figure 4.6: U-Hopper on Nokia E65 smartphones.

and statistics on users P2P communications) in a more user-friendly way, users can visualize at any time such information through an application running on the laptop acting as a user node shown in Fig. 4.7), available for everyone interested in our application .

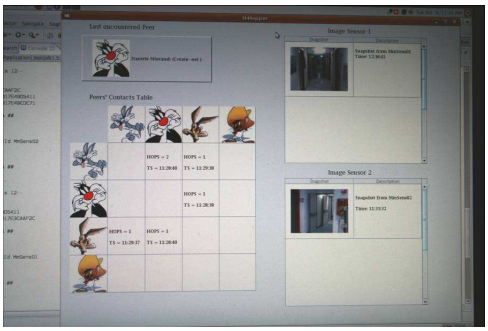


Figure 4.7: The laptop interface, showing contacts among 4 users and data gathered from T-nodes taking snapshots of the site.

4.3 System Design

The aim of our study is to investigate the technological and user-defined constraints which pertain to the design of an opportunistic content distribution application scenario, that allows users to access and distribute digital content such as music, videos or news over Bluetooth-based Smartphones in a given social environment. We have assumed that people always bring with them their smartphone with Bluetooth always on. With a good approximation networking interactions occurred through Bluetooth can be assumed as social interaction among people.

The methodology we applied in the study is briefly depicted in Fig. 4.8. First, we conducted a real-world experiment within an office environment, monitoring the encounters between co-workers equipped with Bluetooth-enabled mobile phones thanks to a simple Java application. Such encounters were then analyzed in order to understand the network of interactions generated by the encounters between co-workers. At the same time, we conducted a survey to assess people's preferences, in terms of preferred news topics and data formats.
In the second phase, we tested the performance of the network by taking into account both user preferences and current technical limitations of the Smartphones, and ran a series of simulations in which we measured the consequences of introducing different data formats (such as text, audio and video) over a Bluetooth-based network. In addition to this, we evaluated the impact of distributing content over the network depending on the preferences selected by users.
In the last phase, we combined technical constraints and user preferences, looking at how the performance of the network is affected by users transmitting heterogeneous data formats and only accepting the data they might be interested in.

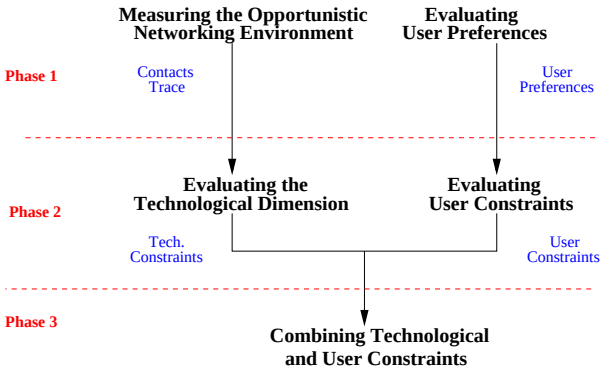


Figure 4.8: The methodology applied in the study consisted of 3 parts: in the first phase, we evaluated a specific office environment in terms of opportunistic networking characteristics, and user preferences; in the second phase, we evaluated how technological constraints and the user constraints influence the design of opportunistic communication systems; in the third part, we evaluated the combination of both constraints.

The social environment we have selected for our real-world investigation is a work environment, which has been chosen for several reasons. First, an office represents a relatively enclosed environment where people are likely to interact with each other frequently. Within this setting, then, we monitored people's natural behavior without having to impose an artificial set of rules for interactions. Second, this environment allowed us to monitor social interactions between the same group of people occurring over an extended period of time. Finally, an office represents a socially-rich environment where encounters often occur not only for work but also for socializing purposes.

Fig. 4.9 presents the organizational structure of the office environment under consideration. It is comprised of 21 people organized into 5 groups, with each group composed by a group leader and a variable number of staff members. The workers chosen for the study had different roles within the organization and were working on different floors in the building. With respect to the social characterization of the people involved in the experimentation, 23% were women, and the ages of participants ranged from 25 to 56, with 28% of the participants being younger than 30.

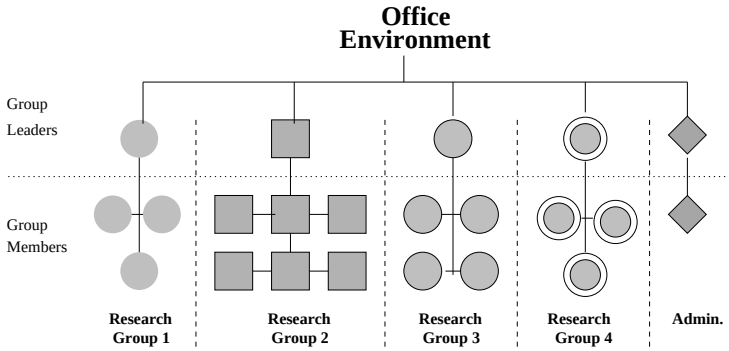


Figure 4.9: Working environment organizational structure.

4.3.1 First Phase: Understanding The Technological and User constraints

Assessing contact opportunities of an office environment

Opportunistic communication systems are typically characterized by the contact patterns of the nodes of the network [51],[86]. A "contact" is defined as the communication opportunity deriving by the physical proximity of two nodes. Clearly, such contact patterns depend from several technological aspects, i.e., the communication technology adopted, the noisiness of the environment, the mobility of nodes, and represent an extremely relevant aspect to study. We have then run a set of experiments to measure the connectivity pattern of nodes in the office environment under consideration.

Similarly to the experiments conducted in [46],[51], people's encounters have been monitored by tracing their proximity for a 4 weeks period. During the experiment, 21 workers - with different roles within the organization and working on different floors of the same building - were equipped with a mobile phone running a java application discovering and tracing neighboring peers approximately every 60 seconds. Whenever the proximity of another device was detected, its bluetooth address, together with the meeting timestamp [1], was saved in the permanent storage of the device for a later processing. In order not to overload the devices memory, a bluetooth enabled laptop acted as a gateway toward a centralized database, gathering the stored information from any smartphone in proximity and transmitting such information to

[1] Nodes are using the phone's internal clock for determining the timestamp. Since SIM cards are inserted in the phones, it possible to synchronize such clock to the GSM network. This ensures a sufficient level of precision, especially when compared with the granularity of the peer discoveries.

a remote repository. The result of this experiment is a trace which includes a series of contacts, with each contact fully characterized by a timestamp, the IDs of the met nodes and the duration of the meeting. Tab. 4.6 presents a summary of the experimentation settings.

Participants	21
Experiment Duration	4 weeks
Registered Contacts	179332
Effective Contacts	14100

Table 4.1: Summary of the experimentation settings. Totally, 21 people were equipped with a bluetooth-enabled phone searching for nearby peers once every 60 s. This resulted in 179332 meetings over a 4 weeks time period. The effective number of contacts is obtained by aggregating consecutive contacts into a longer one.

The meetings duration is inferred from consecutive positive peer discovery inquiries performed by nodes. In fact, the java application is simply tracing the proximity of nodes. This means that a 5 minutes contact is identified by 5 consecutive mutual discoveries (nodes are running a discovering phase approximately every 60 sec.). A post-processing routine has then been applied to the collected raw data in order to aggregate multiple contacts into a single one, with duration equals to the sum of consecutive peer discovery inquiries. The effective number of contacts (Tab. 4.6) is the result of this operation.

Fig. 4.10 depicts the distribution of the contacts duration. It can be easily observed that most of the meetings are relatively short in duration, lasting no more than 100 sec., although few contacts persist for a very long time. This is due users not carrying the mobile phone with them, and leaving it on their desk.

As in any real socially environment, there are couple of nodes meeting very often (i.e., people of the same research group), and people meeting rarely (i.e., people of different groups). This behavior is captured by the intermeeting time among nodes, which measures the time elapsed between the consecutive meeting of nodes couples. The statistical properties of the intermeeting time is represented in Fig. 4.11.

Tab. 4.2 summarizes the overall properties of contacts duration and intermeeting time.

Metric	**Mean**	**Variance**	**Min.**	**Max.**
Inter-meetings	24156	101588	90	1812085
Contacts Duration	473	1047	45	55717

Table 4.2: Intermeetings and contacts duration.

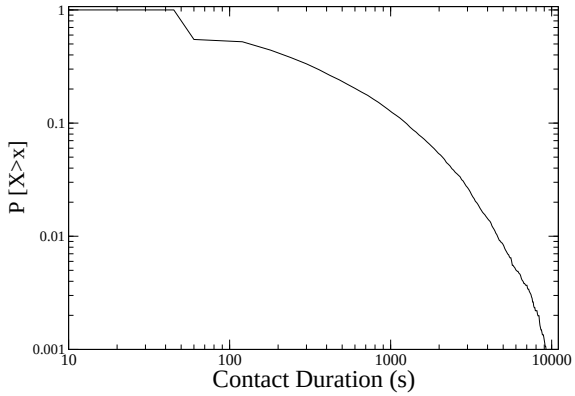


Figure 4.10: Contacts duration distribution.

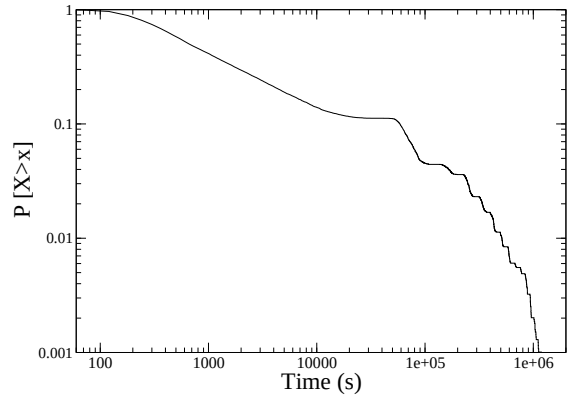


Figure 4.11: Intermeeting time distribution.

We have then considered the network's *Contact Graph* (CG) [45], which is a graph-based representation of the network of contacts, as obtained from the experimentation. In such graph, vertexes represent the nodes of the network, and edges a contact (or a series of contacts) between a couple of nodes. The presence of each edge is regulated by some metric such as, e.g., the cumulative contact duration between nodes couples, or periodicity of such contact. In our work, the edges of the CG are regulated by the cumulative contacts duration, which measures the overall time that 2 nodes i and j have been in contact during the entire duration of the experimentation. In other words, an edge exists between any 2 nodes if they have been close

to each other, over the entire duration of the experimentation, for a period of time longer than a predefined threshold d_{thr}. As such, the CG shows the stronger relations existing among the people in the office environment.

Fig. 4.12 presents the CG in the case of a cumulative duration threshold d_{thr} equals to 40000 s., which corresponds to nodes being in proximity for approximately 30 minutes per day. Given this threshold, Fig. 4.12 shows strong relations among nodes such as, i.e., people working in the same group or going regularly at lunch together. This is confirmed by the different shapes of nodes [2] correspond to different groups within the office environment. As expected, people working together tend to have stronger relations and to be somehow isolated with respect to other colleagues. A few nodes (e.g. nodes 6 and 18) guarantees the connectedness of the network, and represent those people working in collaboration with different groups.

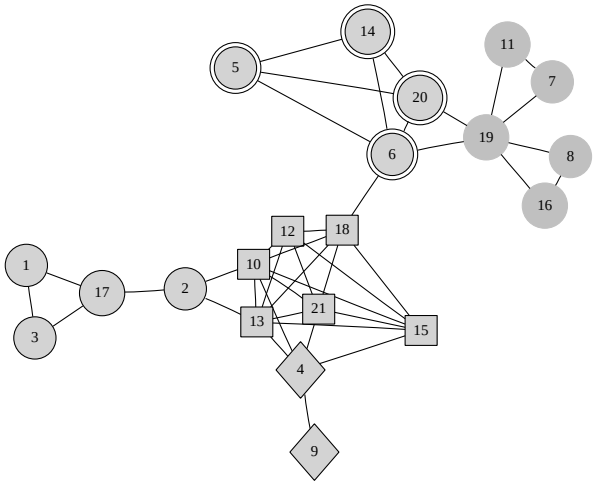


Figure 4.12: Graph-based representation of the network of contacts, as obtained from the real-world experimentation. An edge exists between any 2 vertexes if the corresponding persons have been in proximity for approximately 30 minutes per day.

4.3.2 Assessing users expectations

As in any socially-rich environment, people tend to behave differently, depending on their specific role played in the work environment, their preferences, their attitude toward socialization, and many other factors. In order to assess realistic user preferences, we have conducted a

[2] In Fig. 4.9 the different shapes are explained in correspondence of the office environment organizational structure.

survey with the users involved in the experimentation. The questionnaire consisted of 10 questions addressing their preferences in terms of content (e.g.. music, cinema and news), formats (e.g., text, audio and video) and granularity (i.e., more or less focused) of the information they could access and distribute over such network.
The first part of the questionnaire regarded the users preferences in terms of content. In particular, we asked users to rank the different music, cinema and news categories according to their preferences. Further, we asked them to provide a list of their favourite 5 music bands and 5 actors. This provides us a first feedback on common interests.
The second part of the questionnaire addressed the expectations of users in terms of content potentially deriving from the proposed opportunistic content distribution application scenario. As an example, users were asked to rank the information format they would expect to receive, or the level of granularity of the information circulating over the network. A brief summary

Cinema							
Thriller	Science fiction	Dramatic	Romantic	Comedy	Horror	Documentary	Italian
19%	14%	14%	10%	15%	9%	11%	7%
Music							
Etnic	Rock	Pop	Disco	Rap/ Hip-Hop		Jazz	Classic
13%	18%	16%	12%	10%		15%	14%
News							
Meteo	Politics	Chronicle		Economy	Sport		Culture
11%	21%	18%		16%	14%		15%
Data Format				**Content Level of Detail**			
Video	Audio	Textual		Extremely Focused		Medium level	High Level
28.95%	36.85%	28.95%		45%		21%	34%

Table 4.3: Users preferences expressed for the cinema and music question categories.

of the questionnaires results is presented in Tab. 4.3. As it can be seen, there is a wide heterogeneity in the preferences expressed by users. As an example, for the music category, no specific genre can be excluded a priori. Similar conclusions apply for the other categories.

4.3.3 Opportunistic Content Distribution Application

In an office environment people already use various digital technologies to communicate and exchange data. Compared to established networked technologies, opportunistic communications and mobile peer-to-peer applications add to the richness of this environment as they allow people to share resources only when in physical proximity and often without their full

awareness of any data sharing happening. Various applications can be supported by an opportunistic communication paradigm, and - as we already claimed - these need to be taken into account when assessing the potential performance of the network.
For this study we chose to consider a quite simple application scenario where users download (from an unspecified destination that could be the Internet or a particular location) daily content such as, e.g., music, news, on their mobile phones depending on what they usually are interested in. Such information can be then exchanged between users when they happen to be in proximity. When an encounter occurs, from a networking perspective, users can exchange all the data available on their phone, or keep only the one that matches their interests and forward the other to other users that might be interested, or they can only exchange the data they are interested in.
In order to fully understand the performance of such a content distribution network, we have followed a two steps approach: first we analyzed the technological constraints imposed over such network, initially abstracting users preferences and interests. To this extent, we have developed a trace-based emulator of the diffusion process, able to replay the nodes contact pattern measured during the real-world experimentation, and to simulate a data diffusion process. Second, starting from the preferences expressed by users in the questionnaires, we have reproduced a simplified behavior of users, taking into account their preferences, and the content they could be potentially be interested in. We introduced the concept of *affinity*, which is is a measure of the likelihood they will exchange content of mutual interest, and we evaluated the impact of affinity over the opportunistic diffusion of data.

4.3.4 The Technological Dimension

Given the described office environment, we have simulated an epidemic data diffusion process over the collected contacts trace. Epidemic-style forwarding [89] is based on a "store-carry-forward" paradigm: a node receiving a message buffers and carries that message as it moves, passing it on to new nodes upon encounter. Alike the spread of infectious diseases, each time a message-carrying node encounters a new node not having a copy thereof, the carrier may decide to infect this new node by passing on a message copy; newly infected nodes, in turn, behave similarly. Epidemic diffusion, while being far from optimal in terms of utilized resources [40], is the only viable approach for those application scenarios where information is not delivered from a source to a destination, but rather seamlessly diffused among users.

In order to evaluate the impact of different application constraints over the diffusion process, we have build a simulator of the content distribution process. The simulator replays

the contacts trace gathered during the experiments, and integrates several realistic parameters such as, e.g., bluetooth service discovery time, data transfer rate [3]. With the simulator, we have investigated how the content diffusion varies, depending on different system parameters, as imposed by the opportunistic content distribution application scenario.
Since the aim of the considered application is to distribute heterogeneous content to the interested users, we assumed different content formats to be injected into the network. In particular, as summarized in Tab. 4.4, we assumed three type of data: text, music and video. Each is characterized by a specific size, and a corresponding transfer time, which is determined by the data rate of the bluetooth.

Data Format	**Size**	**Transfer Time**
Text	$100Kb$	$12s$
Audio	$5Mb$	$90s$
Video	$20Mb$	$300s$

Table 4.4: We assumed the opportunistic content distribution application to diffuse different formats of contents: text, music and videos. Each format is characterized by a specific size, and by the time needed for transferring it over the bluetooth communication medium.

As a first step, a randomly chosen node injecting a message at a random time in the network. From that instant the epidemic diffusion starts, and is stopped when the 90% of the nodes have been reached by that message. We assumed all nodes to be equally interested in the content circulating in the network, and we measured the time that is needed in order to infect different fraction of nodes (*Network Infection Ratio*). For each node we run a set of 150 simulations, and we repeated the simulations for all the 21 nodes of the network. Fig. 4.13 presents the results of this first experiment, together with the 98% confidence interval obtained over the 21×150 runs. As it can be observed, the infection proceeds fast up to the 80%, and slows down above this value. This is due to few nodes missing, from time to time, for some days from the office. Video diffusion is slower, since shorts contacts are not enough to exchange $20Mb$ of data over bluetooth, but, in any case, is fast enough to reach most of the network nodes in 1 day (80000 s.).

In the second experiment, we moved one step forward towards the addressed opportunistic content distribution application scenario. We assumed that contents are regularly injected into the system, as required by any content distribution application. Further, contents are

[3] We have run separate measurements for evaluating the bluetooth performance that is possible to experience through the $JSR82$ java APIs for bluetooth. Measurements show that is possible to obtain up to 600 Kb/sec for sufficiently long data transfers, and an average of 25 s. service discovery time.

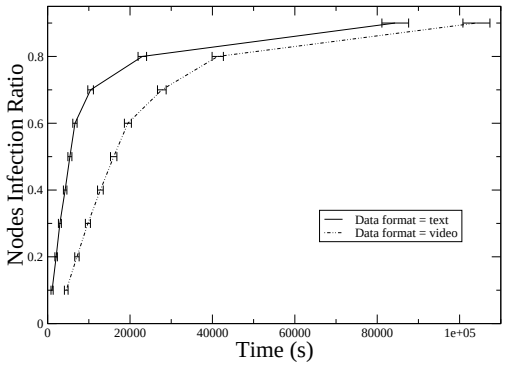


Figure 4.13: Network Infection Ratio over time in the case of text and video content.

fully characterized by their format, e.g., text, music or video, and by the Time To Live TTL, which represents the validity in time of the specific content. The TTL, while being useful for avoiding users to exchange useless information, is also extremely important in mobile networks for avoiding the system to collapse due to the exponential growth of data circulating in the network. The specific format of contents is chosen in accordance with the preferences expressed by the users in the questionnaires (Tab. 4.3).
Contents are sorted in the local storage of nodes according to their generation time, i.e. from the freshest to the oldest, and this order is respected when selecting the data to be sent. At each meeting, two nodes merge the respective storage, avoiding duplication of data[4] and with respect to the contact duration time. In particular, we assume that the contact duration T is equally shared between the 2 nodes meeting, and that each node disposes of $T/2$ seconds of time for sending data to the encountered peer [5].

We have then run a second set of experiments, where a limited set of messages are injected in the network according to a predefined *Message Injection Rates* (MIR). Each simulation is stopped when all messages injected have expired, due to their TTL constraint. We measured the NIR in the case of different and of different TTLs, and different MIRs. In order to obtain a sufficiently small confidence interval, for each setting considered we run 10 simulations, varying the instant at which messages are injected in the network, and averaging results over

[4] In reality, nodes can avoid duplication of data by first running an information discovery protocol. This would introduce an additional overhead that is in any case negligible, with respect to the size of the data being exchanged.
[5] This assumption can be easily removed by assuming nodes to alternatively transmit predefined chunks of data. If the size of the chunk is sufficiently small, this corresponds to equally share the available contact duration T between the 2 nodes.

all the messages injected in the simulation. In Fig. 4.14, the results of this experiments are presented. Increasing the MIR corresponds to increase the network load, and this is reflected, independently from the specific TTL considered, in a significant decrease of the experienced NIR for high MIR values. Also the value of the contents TTL significantly impacts the diffusion of contents in the network: the lower its value, the smaller the time available for a content to diffuse. For the considered office environment, where the dynamism of nodes is relatively small, it is necessary to adopt high TTL values, e.g., greater than 12 hours, in order to reach a sufficiently large number of users.

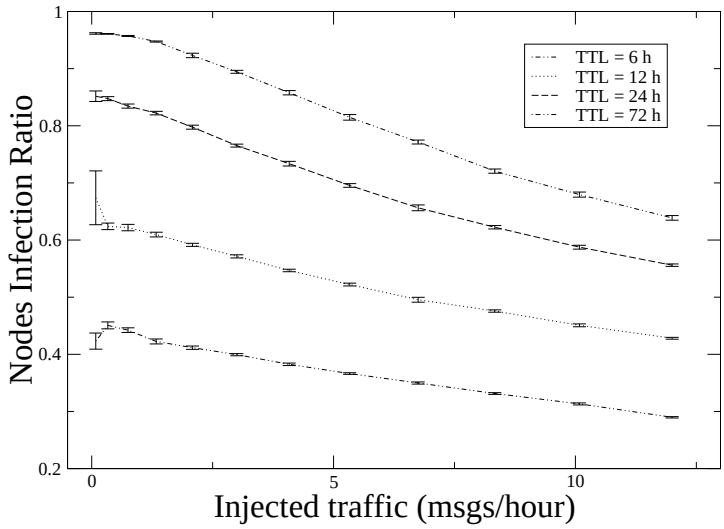


Figure 4.14: *Network Infection Ratio* over time in the case of different *Message Injection Rates*, and different TTLs.

Finally, we investigated how well the different content formats diffuse, when varying the MIR. Fig. 4.15 presents the NIR in the case of a TTL equals 24 hours, for a MIR spanning from 1 to 15 messages per hour. As it can be observed, textual information, due to the limited content size, is insensitive to increases of the MIRs, and also in the case of 15 messages injected per hour is able to reach the 85% of the nodes. Differently, when a high number of video contents are pushed into the system, the opportunistic network saturates, as the limited duration of the contacts is not enough for the nodes to exchange all data. This is clear from Fig. 4.15, where the video MIR decreases down to 50% very soon.

From this first analysis, we can easily conclude that, for the office environment under con-

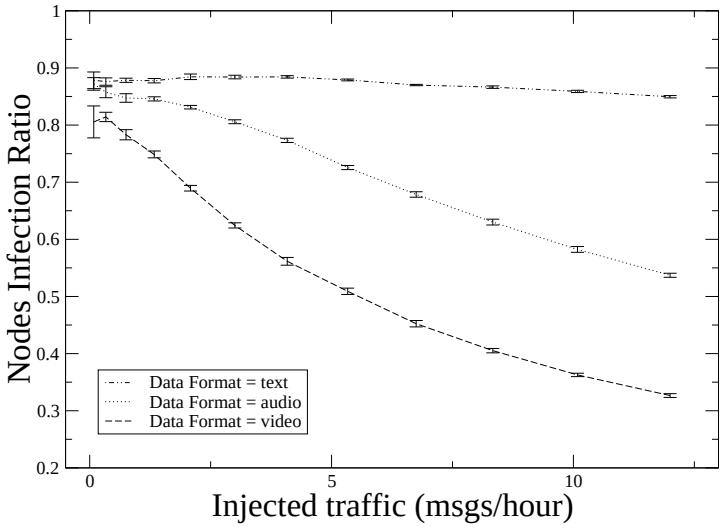


Figure 4.15: Number of users infected by the packet injected the network vs message generation rate in the case of textual, audio and video data format, $TTL = 24$ hours, 10 runs for each point, 98%confidence interval.

sideration, the opportunistic content distribution application can safely support textual information, but not video. Further, given the limited users dynamism, content TTL should be at least 24 hours in order not to expire before having reached a sufficiently high number of users.

4.3.5 Evaluating User Preferences

Starting from the assumption that people tend to share information within communities of similar interests, we reproduced a simplified behavior of users, starting from the preferences expressed by users in the questionnaires. We evaluated then how his impacts the considered application scenario.

User Interests and Affinity Measure

We have assumed each user to be characterized by a set of interests, consisting of the union of different content categories (i.e., music, cinema, news). More formally, to each user i we associate the interests $I_i \bigcup \{I_{i,0}, \ldots, I_{i,N}\}$, where $I_{i,k}$ is the k_{th} interest of user i. Each content category I_k is fully characterized by a finite hierarchical set of nested subcategories (i.e., rock, pop, etc.). We have then associated a weighted tree data structure to each content category,

with weights representing the relevance given by a specific user to that sub-category. Each weight is relative to specific sub-categories level, and detail of each sub-category increases while navigating the tree.
An example of such structure is reported in Fig. 4.16. In this case, the user is interested in three distinct content categories: sport, cinema and music. For the case of music, the user is interested in rock music, and his favourite bands are $U2$ and $Moby$.

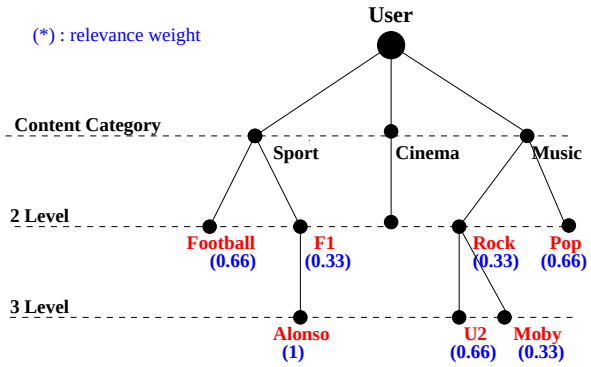


Figure 4.16: User interests tree representation.

For each content category, we have then defined the *affinity* between 2 users as the similarity between the trees representing the corresponding interests for that specific category. The affinity between 2 users i and j is then defined as follows:

$$a_{i,j} = \sum_{k=1}^{N} \sum_{l=1}^{\min(h_{I_{i,k}}, h_{I_{j,k}})} w_l r_l(I_{i,k}, I_{j,k}),$$

where $r_l(I_{i,k}, I_{j,k})$ is the Pearson coefficient [6] of the subset of elements of trees $I_{i,k}$ and $I_{j,k}$ at depth l, $h_{I_{i,k}}$ represents the depth of the content tree k for user i, and w_l is the weight given by users to depth l of the content tree. Clearly, the affinity ranges from 0 to 1, and reflects the similarity between the trees representing the preferences of two users.

Fig. 4.17 presents an example of how such affinity metric is evaluated. In this case:

[6] Pearson's product moment correlation coefficient is defined as $r = \frac{\sum(x_i-\overline{x}_i)(y_i-\overline{y}_i)}{\sqrt{\sum(x_i-\overline{x}_i)^2 \sum(y_i-\overline{y}_i)^2}}$, and measures the linear correlation between two variables.

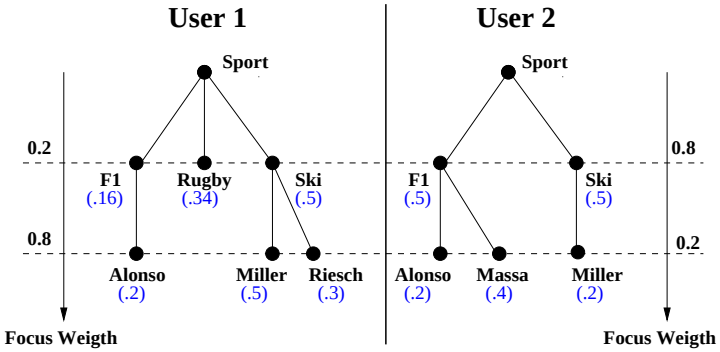


Figure 4.17: Example of evaluation of users affinity.

$$a_{1,2} = 0.2r_1(I_{1,1}, I_{2,1}) + 0.8r_2(I_{1,2}, I_{2,2}) = 0.50,$$
$$a_{2,1} = 0.8r_1(I_{1,1}, I_{2,1}) + 0.2r_2(I_{1,2}, I_{2,2}) = 0.15,$$

and the difference is due to the fact that user 1 retains more relevant less focused information, for which there is a higher match.
Starting from the user preferences expressed in the questionnaires, we evaluated the affinity among users for the different content categories considered. The results are summarized in Tab. 4.5, where the last case is the total affinity between 2 users. As it can easily observed, users interests are very close with respect to news, while the music content category leads to larger differences.

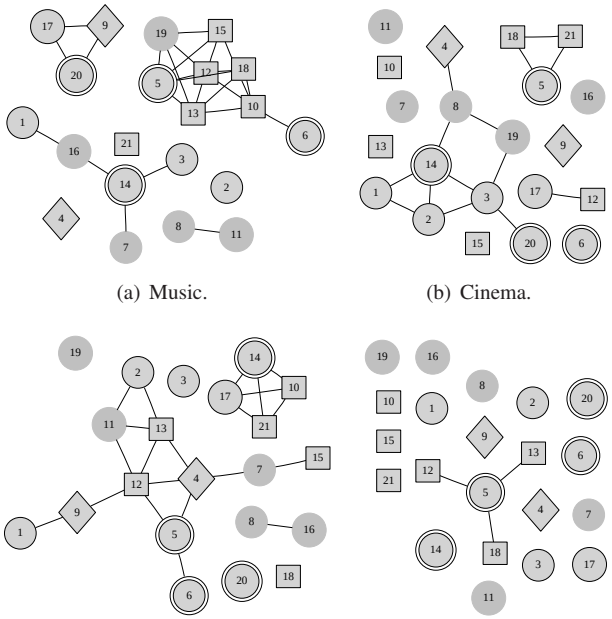
Content Cat.	Min	Mean	Max	Variance
Music	0.004	0.36	0.98	0.26
Cinema	0.003	0.39	1	0.23
News	0.003	0.54	1	0.23
Total	0.38	0.36	0.41	0.02

Table 4.5: The affinity among users, for the content categories considered, music, cinema and news, and for the total category.

We have then introduced the *Affinity Graph* (AG), in which an edge is drawn between two vertexes if the corresponding nodes presents an affinity above a given threshold λ_{thr}. Differently from Fig. 4.12, in this case constraints are derived

1.

solely by the affinity among users, and not from the strength of the "contact" links.



(a) Music. (b) Cinema.

(c) News. (d) Total.

Figure 4.18: Affinity Graph for the music, cinema, news and total categories. The affinity graph is build by drawing an edge between two vertexes if the corresponding nodes presents an affinity above a given threshold $aff_{thr} = 0.75$.

Fig. 4.18 shows the AG for the different content categories, and an affinity threshold $aff_{thr} = 0.75$. As it can be easily observed, depending on the specific content category and affinity threshold, the topology of the graph changes significantly. For an affinity of 0.75, all the graphs are already partitioned, with many isolated nodes. This is particularly true for the $Total$ case where, a part for a subgraph of 3 nodes, the remaining ones are completely isolated from the rest of the network. The reason for this is that the affinities for different categories tend to compensate each other, and users showing similar preferences in terms of, e.g., music have opposite ones with respect to cinema. This situation is mapped into a lower value of the total affinity, and heavily influences the resulting network structure.

4.3.6 Phase 3: combining users and technological constraints

In this last part of the study, we have evaluated how the combination of both the technological and user-defined constraints impact the opportunistic content distribution application.
Let us now assume users to exchange data only if they share common interests. At the system level, this requirement is mapped to users exchanging data only if their affinity exceeds a certain threshold aff_{thr}, otherwise not. Clearly, the higher the threshold, the closer must users interests have to be for a data exchange to occur. We have then simulated the opportunistic content diffusion over the collected traces taking into consideration the interests expressed by users in the questionnaires. Differently from the previous evaluation, messages are now injected in the system and tagged with a specific category (belonging to the set of those present in questionnaires). When two nodes start an information exchange, data is organized in the storage from the most to the less relevant for the other node, so that in the available meeting time the most interesting information are delivered first.

Fig.4.19 shows, for different $TTLs$ and for a fixed infection ratio of 1 message per hour, how the node infection ratio varies in correspondence of different affinity thresholds. Again, each point is evaluated as the average of 10 different runs and with a confidence interval of 98% and simulations are stopped only when all the contents are expired. As intuitively clear, for lower values of aff_{thr} a loose filter is applied to the data exchange and the information is influenced by the technological dimension only. Results change when increasing aff_{thr} up to 0.7, above which no sufficient match is present among users preferences for a data exchange to occur. We can then conclude that the opportunistic content distribution performance consistently degrade as the aff_{thr} increases. In fact, for high values of aff_{thr} (i.e. higher that 0.7) users collect only content they are really interested in, without storing contents that could be of interest for other nodes in the network, showing in such a way a selfish behaviour.

We have then evaluated the behaviour of the network if data of a single content category are injected and the affinity among users is evaluated only with regards to that single category. The simulation settings are the same as before. Fig. 4.20 depicts network performance (i.e. node infection ratio) for the three content categories we have assumed throughout this paper: it can be observed that the behavior is the same independently from the specific category, but that nodes still exchange some contents for the highest values of aff_{thr}. This is due to the fact that, as shown in the previous section, the affinity among users regarding a single content category is obviously higher than the total one, suggesting that opportunistic content distribution applications performs better with more specific contents.

Finally, we have analyzed a particular case of content distribution in order to show the

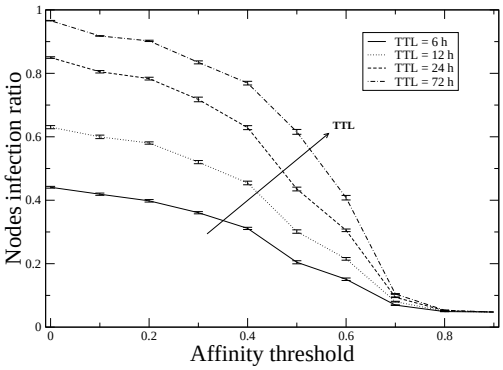


Figure 4.19: Nodes infection ratio vs affinity threshold in the case of heterogeneous data formats, TTL of 6, 12, 24 and 72 hours and injection rate of 1 message per hour.

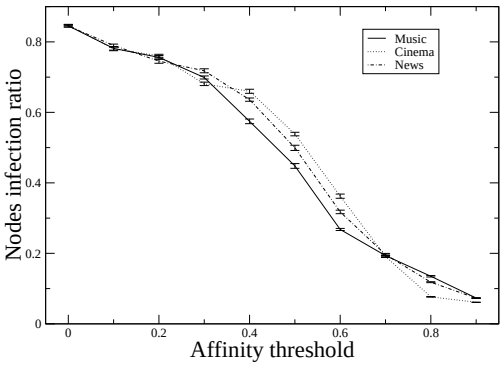
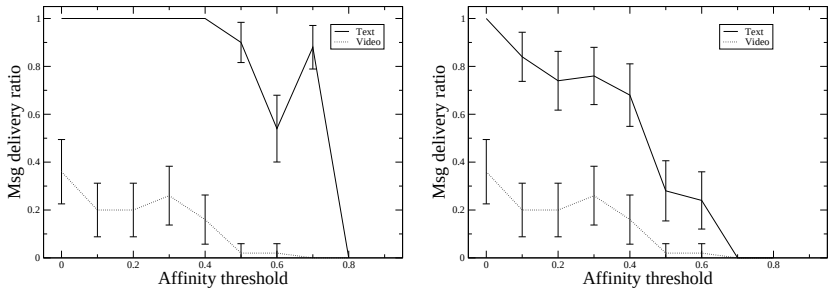


Figure 4.20: Nodes infection ratio vs. affinity threshold, for the three content categories considered, TTL of 24 hours and injection rate of 1 message per hour.

importance of combining the technological and users constraints in the considered application. We have preliminarily isolated 3 users in the network, whose interests were very close in the content category "News" (Fig. 4.18). These users correspond to nodes '2', '4' and '5'. As a consequence, an information generated by '5' will almost certainly interest also node '2' and '4'. At the same time, we have observed that in Fig. 4.12 these nodes belong to different groups, and with a high probability a message delivery among them would occur only if the content distribution network performs well, e.g. all the nodes encountered by '5' store the

message and forward it finally to ‘2’.



(a) $5 \rightarrow 2$. (b) $5 \rightarrow 4$.

Figure 4.21: Graph resulting from an affinity threshold of 0.6, in the case of music and news categories, respectively.

In a highly loaded network (i.e. injection rate of 10 messages per hour) with a TTL of 24 hours we have evaluated for increasing values of aff_{thr} the message delivery ratio for the above mentioned nodes, i.e. the number of times one specific message injected by ‘5’ reaches ‘2’ and ‘4’. In particular, we have monitored a single message injected by ‘5’ and stopped the simulation either when it reached both ‘2’ and ‘4’, or whrn it expires. The separated results for each nodes couple (‘5’-‘2’ and ‘5’-‘4’) are reported in Fig. 4.21 for the text and video formats. Each point in figure has been averaged over 50 different runs with a confidence interval of 98%. A value of 1 for the message delivery ratio means that all the messages generated by ‘5’ have been forwarded to ‘2’ or ‘4’, while a value of 0 means that none of the considered information have been delivered before expiration. We can observe the different behaviour between the two formats for each source-destination couple: it is interesting to notice how, for the same simulation settings, a textual information can be easily delivered even for high values of aff_{thr} (e.g. up to 0.7 for ‘2’) while a video not. At the same time we notice that two nodes that should apparently behave similarly, results with a different infection. This is due to the fact that when the network is congested, contents diffusion paths differs significantly among each other.

4.3.7 Discussion

Reflecting on these results, we begin to see that considering technical constraints in isolation from expressed user preferences, or vice versa, does not paint a complete picture. When we begin to consider not merely inter-user encounters, but the duration of these meetings, coupled

with the format and size of data which the users might want to share, a network graph like that of (insert ref) Figure 5 begins to emerge. The length of an encounter, then, becomes extremely important when we consider not only *if* people were in proximity with one another, but how regularly, or how briefly they met. In recognizing that data transmission is, as of yet, not instantaneous, the graph of potential data exchange becomes less complete. Further, when we incorporate user-expressed preferences about the types of data they might wish to exchange, the graphs based on these affinities become even more fragmented. From these results we do not mean to say that opportunistic content distribution systems are not feasible, but rather to motivate the importance of both current technological limitations as well as user desires.
In future work we plan to investigate how these important factors might be effectively incorporated into the design of future systems. One interesting avenue for exploration we have considered is to attempt to identify hubs of these social networks and to lower their level of affinity in order to aid in the diffusion of data. Alternatively, we also propose to make the inner-workings of the network more transparent to users, encouraging an awareness of the trade-off between requesting only very specific types of data, or content which is of considerable size (e.g. video), and the amount of data that user is likely to receive. By attempting to incorporate social factors into the parameters of the applications performance, and, on the other hand, by making users more aware of the technical implications of their expressed choices, we hope to be able to create an opportunistic content distribution system which achieves an optimal level of real-world performance.

4.4 Interests-Driven Opportunistic Data Diffusion Schemes

We are now interested in understanding whether the social network the user is part of (exemplified by a set of "nearest neighbours") can be leveraged to increase the system performance. In such respect, we introduced a class of opportunistic data diffusion algorithms, which we call k–friends–based relaying. The idea is the following one. Each node selects a set of k "nearest neighbours", based on the contact pattern (number of contacts within a given time-frame and their duration). It then builds a list of interests by merging its own ones with the ones of its best friends (properly weighted and/or ordered). Upon a meeting, it communicates such list to the other node, asking for data matching such interests. In this way, it aims at receiving both the content it is interested in and the content its nearest neighbours are interested in. As nearest neighbors are nodes with which frequent/long transmission opportunities oc-

cur, the net effect is to build "fast" paths (mainly two–hop ones) between the content sources and the users interested in it. Performance is measured in terms of two parameters. The first one is represented by the utility perceived by the user, measured as the fraction of interesting content available in the system the user actually receives. The second one is a measure of the overall system efficiency, expressed as the average ratio between the number of nodes which received a message and the number of nodes potentially interested in that message. Emulation results are presented, showing that k–friends relaying schemes are able to offer an interesting trade–off between utility and efficiency across a variety of traffic loads.

Our evaluation is based on emulations, performed using traces obtained from real–world experiments

4.4.1 System Model and Problem Formulation

In this work, we focus on an opportunistic content sharing and distribution system. Users in the system carry a hand-held device with some form of proximity wireless communications. No infrastructure (e.g., 3G, WiFi-based access points etc.) is assumed to be present. Information can be exchanged among two nodes whenever they get within mutual communication range. Each user has a list of well–defined interests, which is specified through a suitable application interface. Data is exogenously injected into the system by some fixed devices. Each data unit (from now on called also, with a slight abuse of terminology, a message or an item) is marked with some metadata describing its content. The metadata associated to a message and the user's interests are expressed using a common language (as an example, standard ontology languages such as RDF or OWL could be used). For the sake of simplicity, we assume in the following that each message is marked with one single tag, describing its content (in principle, a message could include content belonging to various categories). The set of possible categories is assumed to be finite and is denoted by Ψ. Each message has also a time–to–live field, expressed in seconds, which determines the persistence of the message in the system.

Each item c injected into the network is characterized by:

1. the item category; we use the notation $\psi(c) \in \Psi$ addressing the category of item c;
2. the item *content generation time*, which represents the generation time of the specific item;
3. the item *time to live* (TTL), which determines the temporal validity of such item (in seconds), starting from its content generation time.

Each user is characterized by:

- a set of interests $\varphi_i \subseteq \Psi$, specifying the content user i is interested in;
- an interest vector, which is a list of pairs $< category, weight >$ where *category* represents a specific interest and *weight* is a numerical value in the range $(0, 1]$ expressing the level of interest. All categories represented in φ_i are present in the interest vector with unitary weight.

Data are exchanged upon meetings following a pull–based technique. When node i meets node j, it sends a query message containing its interest vector. Node j checks the availability of such type of content in its internal buffer and transmits the items matching the query, in decreasing order of weight.

Two performance metrics are used. First, we define a network-wide *utility* figure aimed at capturing the perceived quality of service by the users. This accounts for the ability of the mechanism to deliver the content present in the system to the users interested in it. Of course, we also need to account for the resources spent in order to deliver such contents: in order to do so, we introduce another performance figure, that we refer to as *efficiency*, aimed at describing the ability of the proposed mechanism to limit the usage of resources. For the application scenario under consideration, indeed, the optimal operating condition is to deliver each content item *only* to those users interested in it. This ensures that any content injected in the network will reach the intended (= interested) destinations, and that no redundant messages are generated, thus minimizing the resources that are used for running the opportunistic service. We refer to this as the *ideal* case with obvious meaning. For each content category ψ we define set S_ψ as the set of nodes which are interested in category ψ.
Now, when item c is injected in the network, over time it will reach a set I_c of the network nodes. At any instant in time we denote by S_c the set of infected nodes being interested in the content category $\psi(c) : S_c = S_{\psi(c)} \cap I_c$.
Given a set of M messages, we define the following figures:

- *Network Infection Ratio* (NIR): measures the average fraction of infected nodes per message, namely

$$NIR = \frac{1}{M} \sum_{i=1}^{M} \frac{|I_{c_i}|}{N}, \tag{4.1}$$

  where $|\cdot|$ identifies the cardinality of a set and N denotes the number of users in the system.

- *Utility* (U): measures the average number of *interested* nodes infected by a message, i.e.,

$$U = \frac{1}{M} \sum_{i=1}^{M} \frac{|S_{c_i}|}{|S_{\psi(c_i)}|} \tag{4.2}$$

- *Efficiency* (E): measures the tradeoff between message delivery and resources consumption. It is defined as the average ratio between the fraction of interested nodes infected and the fraction of overall nodes infected. In symbols:

$$E = \frac{1}{M} \sum_{i=1}^{M} \frac{|S_{c_i}|}{|S_{\psi}(c_i)|} \frac{N}{|I_{c_i}|} \tag{4.3}$$

In order to characterize this metrics, we start noticing that $|S_{c_i}| \leq |S_{\psi(c_i)}| \; \forall c_i$ so that $0 \leq U \leq 1$. Furthermore, as $\frac{|I_{c_i}|}{N} \leq 1 \; \forall i$, $U \leq E$. Equality holds only if, before the TTL timer expires, any message c_i gets received by all nodes.
In the ideal case $|S_{c_i}| = |S_{\psi(c_i)}|$, so that $U = 1$. We also notice that, $S_{c_i} = I_{c_i}$, so that from (4.3) the maximum efficiency is

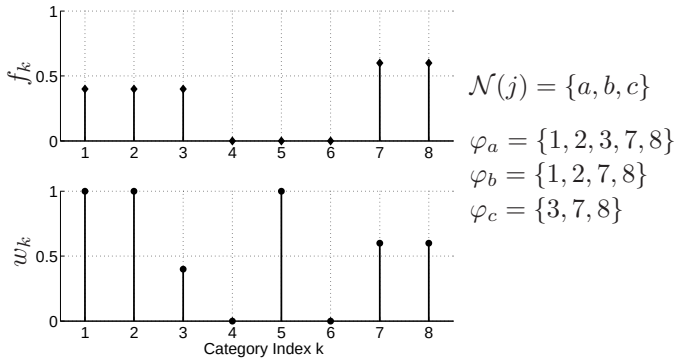
$$E_{\max} = \frac{N}{M} \sum_{i=1}^{M} \frac{1}{|S_{\psi}(c_i)|} \tag{4.4}$$

From the definition of the above metrics, it is clear that in order to improve the efficiency, nodes should forward data c_i only to nodes in $S_{\psi(c_i)}$. Nevertheless, this severely limits $|I_{c_i}|$, especially when data has a finite lifetime (as this hinders the possibility of exploiting relaying), potentially harming the utility figure.

4.4.2 Algorithms and Protocols

In this section we propose an algorithmic approach aimed at trading off utility for redundancy with the aim to increase the efficiency. The key intuition is rather simple: we leverage local exchange of data among nodes which have stronger encounter patterns. We call them *persistent neighbours*. We also account for the categories that are mostly represented in such set of nodes and try to forward first data that correspond to more popular categories. This technique has two appealing features from a distributed implementation standpoint:

- it is receiver based, since the filtering is operated at the receiver side;
- only local information on each node encounter pattern is needed.



Sorted list of categories sent to i: $O = (1, 2, 8, 7, 3)$

Figure 4.22: Example of construction of the category weights at node j. For example, the second item stored belongs to category 1; list O represents the list of categories of items received at i.

Algorithmic description

Our proposed scheme is based on two distinct procedures that run in parallel: the first one is used to build the interest vector; the second procedure leverages a receiver–driven diffusion scheme for data exchange.

Persistent neighbors discovery: this procedure builds an estimate of the set of persistent neighbors $\mathcal{N}(i)$ of a node i. Any node in the network traces the nodes met during a given time window and the length of the respective contact. Nodes are then sorted according to the sum of the contact durations, and the first k of them are considered as persistent neighbours. The parameter k is used to control the utility/efficiency tradeoff. For $k = 0$ we limit the usage of resources, but relaying is not exploited. Larger values of k lead to the use of larger amount of resources, but potentially increase the utility figure as well.

As a second step, this procedure evaluates the relative frequency of the *categories* within $\mathcal{N}(i)$, $\varphi_{\mathcal{N}(i)} = \cup_{i \in \mathcal{N}(i)} \varphi_i$, building the histogram of such categories with respect, i.e., measuring the relative frequency f_ψ of interest in ψ within $\mathcal{N}(i)$:

$$f_\psi = \frac{|S_\psi \cap \mathcal{N}(i)|}{|\varphi_{\mathcal{N}(i)}|}, \quad \psi \in \varphi_{\mathcal{N}} \tag{4.5}$$

Forwarding based on interests vector: the second procedure is used to drive opportunistic forwarding as follows. Once two nodes meet, after a customary discovery phase, nodes exchange their interest vector. The interest vector contains a set of categories with an associated weight. The weights are determined at node i as follows. For category ψ, we have:

$$w_\psi = \begin{cases} 1 & \text{if} \quad \psi \in \varphi_i; \\ f_\psi & \text{if} \quad \psi \in \varphi_{\mathcal{N}(i)} \setminus \varphi_i; \\ 0 & \text{otherwise.} \end{cases} \tag{4.6}$$

An example of the construction of the category weights is depicted in Fig. 4.22, where weights are calculated based on the set of persistent neighbors $\varphi_{\mathcal{N}} = \{1, 2, 3, 7, 8\}$ relative to nodes $\mathcal{N}(j) = \{a, b, c\}$.

After receiving the interest vector, the other node starts sending the matching items present in its internal buffer. This is done by following (deterministically) the weight associated to each category. Items whose category is ranked higher are transmitted first. If multiple items stored belong to the same category, the order of transmission among them is sorted according to a LIFO principle. Of course, an item c such that $f_{\psi(c)} = 1$ will be transmitted first. In the example depicted in Fig. 4.22, node j starts transmitting two items belonging to category e. When all such items have been transmitted, node i will start forwarding sequentially items c such that $0 < f_{\psi(c)} < 1$ (in this case, message 3, which belongs to category a). The process lasts until either all data of interest are transmitted, or the contact ends. We notice that, due to the finite duration of a contact, it may happen that items (with lower weight) will not be transmitted.

Protocol Implementation

The aforementioned scheme has been implemented in a java–based middleware. All transactions reported in Fig. 4.23 have to be understood at the application level. Batches of messages are passed to the Bluetooth L2CAP protocol. Piggybacking is employed to maximize the system performance.

At each node, data are maintained in a buffer, where new data is added on the head. Upon meeting a node, a routine is triggered to eliminate from the buffer the messages whose TTL has expired.

Upon meeting a node, a message containing the interest vector is build and transmitted to the neighboring node (Alg. 4). Upon receiving an interest vector, the set of data to be transmitted

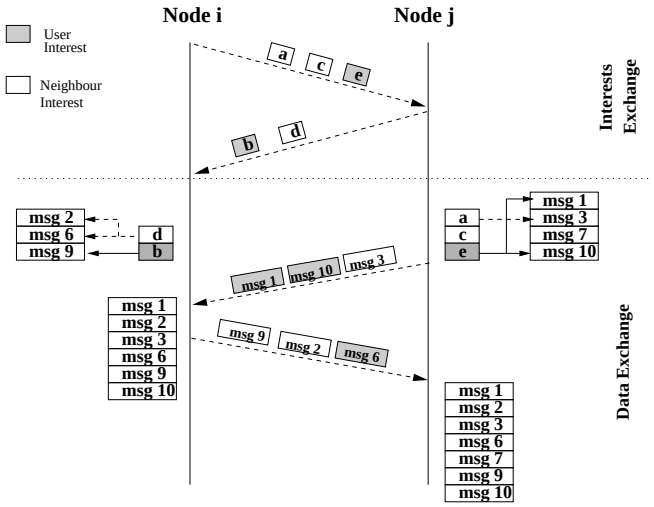


Figure 4.23: Graphical representation of the data flows in the proposed scheme.

is created as follows. First, the categories with the highest weight in the interest vector are selected. The set of available data is scanned; if a positive match is found, the corresponding item is sent to the L2CAP. Once the buffer has been scanned, the set of categories corresponding to the second highest weight are considered. The procedure is repeated iteratively until either all entries in the interest vector have been considered or the contact has ended (Alg. 5).

Algorithm 4 Interests Generation and Transmission

Require: neighContactTime[ ]

1: neighbours ← updateNeighbours(neighContactTime[ ]);
2: $\psi \leftarrow$ evaluateInterests(neighbours);
3: sendInterests(ψ);

Algorithm 5 Messages Transmission

Require: contVector[ ]

1: $\psi \leftarrow$ receiveInterests( );
2: sort(contVector[ ],ψ);
3: **for** $i = 0$ to $contentVector.size$ **do**
4: m ← contVector[i];
5: send(m);
6: **end for**

4.4.3 Experimental Evaluation

In order to evaluate the performance of the proposed algorithm and protocols, we have developed an emulator that reproduces the opportunistic diffusion of contents, exploiting the contact trace and users preferences collected during the first phase of this work.
The emulator, at startup phase, loads the users, together with their interests, and then processes one content at a time, as measured during the field experimentation. At regular rate, new content items are injected into the network. Each content item is characterized by (i) a specific category, which is determined starting from the interests of the user receiving that item, (ii) a Time To Live (TTL), which determines the time validity of such item, starting from the instant when such item is injected into the network (iii) a specific format, which specifies the size (expressed in bits) of the item being exchanged and, therefore, the time that is needed for exchanging it given the bluetooth data rate. [7] Once a content is expired, it is not diffused any longer by the nodes. At each contact, nodes exchange messages according to the protocol described in Sec. 4.4.2, and applying the interests-driven forwarding mechanism.
The performance of the system is evaluated according to the metrics introduced in Sec. 4.4.1, *Network Infection Ration* (NIR), *Utility* and *Efficiency*, and averaging over 20 runs. Results are reported in terms of 95% confidence intervals. In order to randomize emulations and obtain sufficiently tight confidence intervals of the results, each run starts at a random time instant, uniformly distributed over the entire duration of the contact trace. The trace is then iteratively reproduced until all messages injected in the network have expired. At the end of each emulation, a routine processes the results and evaluates the NIR, Utility and Efficiency of the run.

Data Format	Audio ($5Mb$ of size and $90s$ of transfer time)
Data TTL	48 h
Interest	1 and 3
Diffusion Mechanisms	flooding, k–neighbours
Message Rate	1–50 msgs/h

Table 4.6: Summary of the experimental settings: audio contents, with a $48h$ TTL are emulated applying the flooding and k–nearest neighbours–based diffusion mechanisms.

As a first experiment, we have assumed audio contents, with $5Mb$ size and $90s$ of transfer

[7] We have run separate measurements for evaluating the bluetooth performance that is possible to experience through the $JSR82$ java APIs for bluetooth. Measurements show that is possible to obtain up to 600 Kb/s for sufficiently long data transfers, and an average of 25 s service discovery time.

time, to be injected into the network with a $TTL = 48h$. All users are assumed to be interested in all the three considered categories (music, news, cinema). For each category, each user is assumed to be interested in two sub-categories (e.g., music → rock, music → jazz, etc.) Each user is therefore interested in 6 types of content among the possible 21 subcategories (see Tab. 4.3). Users are assigned to categories according to the outcomes of the questionnaires, as reported in Tab. 4.3. We have evaluated the case of flooding and k–nearest neighbours–based diffusion for various network loads (measured in messages/hr).

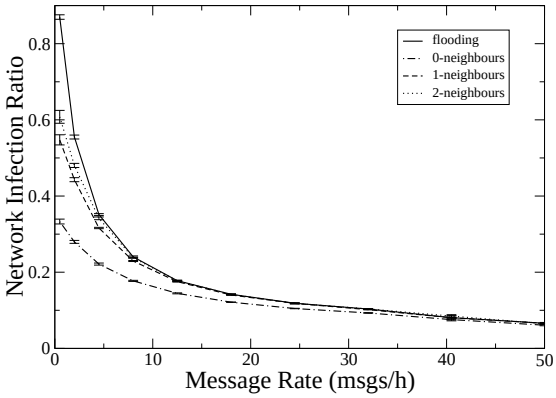


Figure 4.24: *Network Infection Ratio* in the case of flooding and k–nearest–neighbours–based diffusion. Contents are injected at a variable rate, and with an expiration time equals to $TTL = 48h$. Each user is interested in 3 categories (out of 3).

The results of this first experiment are depicted in Figs. 4.24, 4.25 and 4.26. As intuitively clear, the highest NIR is obtained by the flooding mechanism. In this case, no specific policy is applied to the content diffusion process, and all nodes exchange all currently stored in their buffers, being constrained only by the limited contact duration. The NIR of the interest–based diffusion mechanism depends on the size of the neighbourhood. The case of 0–neighbours corresponds to a node behaving in a selfish manner, storing and exchanging contents that are relevant to its own interests only. For larger values of the neighbourhood size, nodes start to work in a cooperative manner, exchanging, with lower priority, messages destined not to themselves only, but also to their persistent neighbours. This is reflected in a higher value of the NIR.

Fig. 4.25 presents the Utility results for the same settings as before. As expected, flooding performs best in the presence of very low loads, but it performance degrade quickly, and k–

nearest–neighbours scheme take over. In this case, exchanging messages according to the users interests provides a better Utility, since this metric takes into account how many interested users a message with a specific category was able to reach. Clearly, increasing the size of the neighbourhood corresponds a less selfish behavior and provides a better utility.

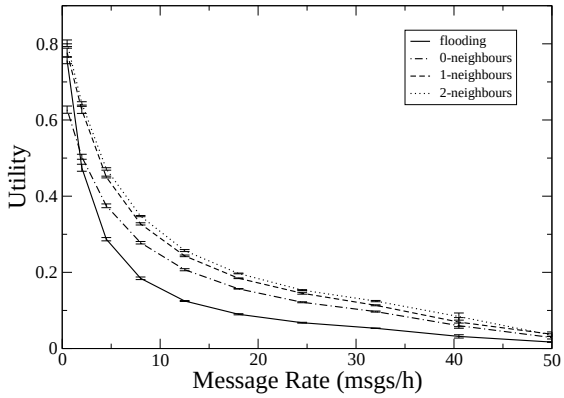


Figure 4.25: *Utility* in the case of flooding and k–nearest–neighbours–based diffusion. Contents are injected at a variable rate, and with an expiration time equals to $TTL = 48h$.

In Fig. 4.26, the resulting system efficiency is plotted for the same settings. The efficiency reflects how well the system behaves in delivering contents to interested users only. In this respect, the flooding is outperformed by the nearest neighbours–based scheme, independently from the size of the neighbourhood. This is due to the completely blind diffusion mechanism, which propagates messages to any node, independently from its interests. The nearest neighbours–based diffusion mechanism efficiency changes depending on the size of the neighbourhood and of the message rate. At low network load, low values of neighbourhood size lead to better efficiency. Under such conditions, the benefit offered in terms of utility is not sufficient to pay for the increase usage of resources. However, the beneficial impact of persistent neighbours become evident for message rates greater than 10 msgs/h. In this case, diffusing messages also for persistent neighbours allows nodes to maximally exploit any contact opportunity, and the larger the size of the neighbourhood, the greater the gain (at least up to $k = 2$).

As a second experiment, we have assumed nodes to be interested in 1 category of content only: either music or cinema or news. This corresponds to increasing the diversity of users

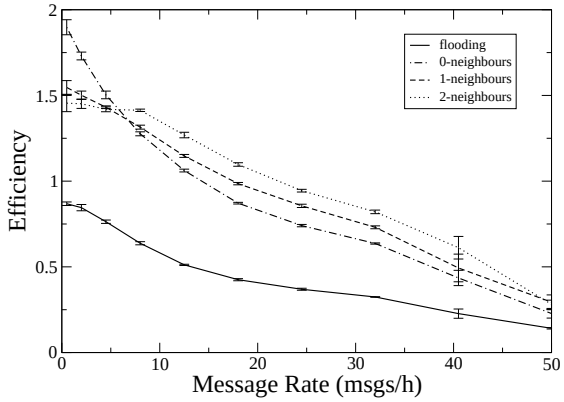


Figure 4.26: *Efficiency* in the case of flooding and k–nearest–neighbours–based diffusion. Contents are injected at a variable rate, and with an expiration time equals to $TTL = 48h$.

interests, and therefore to reducing the likelihood that any 2 nodes meeting will share a common interest. As before, each node is interested in 2 sub-categories. In this case, each node is interested in 2 out of 21 possible types of content. The assignment of (sub-)categories to nodes is done again by exploiting the data gathered through the questionnaires.
Fig. 4.27 presents the NIR results for this second case. As before, the flooding is able to maximally diffuse a message over the global network, outperforming k–nearest neighbours–based diffusion schemes.

Fig. 4.28 presents the utility for this second setting. Also in this case, the interests-based diffusion with 2 neighbours performs best, but in this case the gain is much higher. This is due to the limited number of encounters with nodes sharing similar interests. In this case, a purely selfish behavior (neighbourhood size equals to 0) severely limits the utility of the diffusion mechanisms and, for low values of the message rate, is even outperformed by the flooding. Differently, even a minimal cooperative behavior leads to a much more effective diffusion of data.

Finally, Fig. 4.29 presents the efficiency of the system for this second setting. For this second setting, the 1–nearest–neighbour diffusion mechanisms performs best. This is due to the optimal trade-off between the level of cooperation and the message redundancy that is introduced into the network. Indeed, especially for high values of message rate, the 1–nearest–neighbour interests-based diffusion scheme is able to achieve a very high utility (Fig. 4.28),

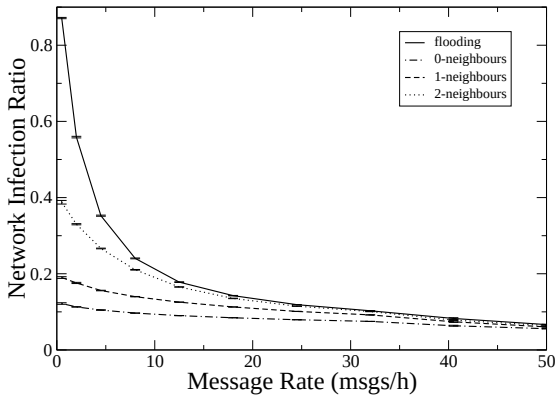


Figure 4.27: *Network Infection Ratio* in the case of flooding and k–nearest–neighbours–based diffusion. Contents are injected at a variable rate, and with an expiration time equals to $TTL = 48h$. Each user is interested in 1 category (out of 3).

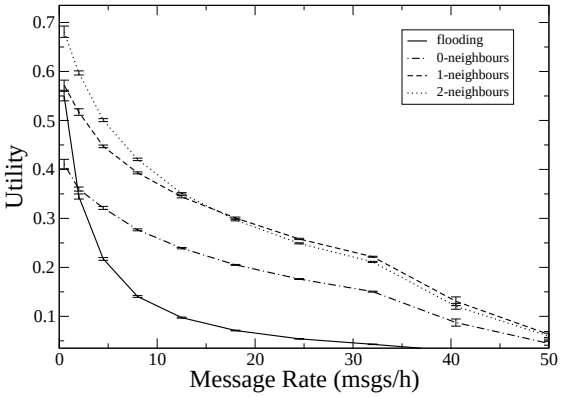


Figure 4.28: *Utility* in the case of flooding and k–nearest–neighbours–based diffusion. Contents are injected at a variable rate, and with an expiration time equals to $TTL = 48h$.

but introducing a limited number of redundant messages (Fig. 4.27), if compared, e.g., with he 0–nearest–neighbour mechanism.

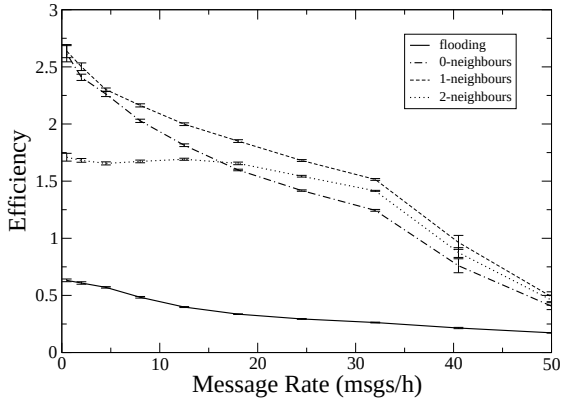


Figure 4.29: *Efficiency* in the case of flooding and k–nearest–neighbours–based diffusion. Contents are injected at a variable rate, and with an expiration time equals to $TTL = 48h$.

4.5 Cooperative Evolution of Services in Ubiquitous Computing Environments

Autonomic Services for Pervasive Computing

In the recent years, an increasing attention has been devoted to the design and deployment of services offered to the end users by means of the novel generation of devices. Recent technologies, in fact, offer the possibility of remote access, thus having the service following the user, even when in motion and regardless of her/his actual location, e.g., at office or at home. Thus, we assist to what is commonly referred as to the shift towards *ubiquitous services* [2].

While the notion of service itself stems from the business science community (where it is widely identified with an intangible activity taking place in the form of an interaction between the customer and the service provider [90], [91]), in the language of computer science and information science such terminology is mostly related to the support of a set of applications and functionalities, mainly provided over the Internet [92].

But, despite it has become customary in literature to refer to the notion of a *service*, it is difficult to adopt a widely accepted definition, due to the fact that the interpretation of what a service is depends on the context of use. Providing a common understanding of the word

service is out of the scope of this paper, but, in sight of the above considerations, it is indeed worth recalling the terminology adopted in some related fields, as outlined for example in [92] and [93]. The definition used in the Business Science community usually contains some recurring elements: a service is mostly identified with an intangible activity taking place in the form of an interaction between the customer and the service provider [90], [91]. More in general, any service can be described as an action performed by one entity on behalf of another, where this action involves a transfer of value [93]. In the language of computer science and information science, conversely, the service terminology is mostly related to the support of *web-service* and *e-service*, meaning a set of applications and functionalities related to the Internet [92].

We remark that the definition of service instead, even in the case of web-services and e-services, is very often conveniently identified with a *business-service*, especially where dealing with issues of network pricing and quality of service, because what rules the service provision is ultimately the degree of satisfactions of the end user.

In the remainder of this work, we will use a broad definition of "services", intending them as software components which perform a given task, which is valuable to the user. Naturally, we associate to a service a level of "fitness", described as the quality of experience perceived by the user (in terms of both functional and non-functional aspects). We consider a homogeneous set of users, so that the level of quality associated to a given service does not depend on the specific user taking advantage of it. In this paper, we focus on the actual performance of a service, rather on the mechanisms underpinning the software performing the given task.

Hence, this will be the first feature of a service that we will account for in the following, i.e., with vague notion, the quality or the level of a service; we denote briefly *fitness* of a service the satisfaction perceived by the end user when providing the service. Despite it sounds a subjective definition totally depending on the user, in the networking literature it is preferred to rely on a set of objective parameters, often represented by physical quantities such as delays or energy. In an equivalent way, when referrring to a particular service in the reminder of this paper, we will define the fitness as an objective parameter to measure the satisfaction of each user to the actual service. Second, we will mostly consider a service as provided by a software, running locally or possibly remotely and displayed on a device owned by the customer. We will concentrate more on the algorithmic and performance aspects rather than on the implications of what a service is. For example, it is quite common to refer to a *web-service* and an *e-service*, meaning the *program* or *application* instance which is the implementation running on some device. In this paper, we will deal especially with the above mentioned

concrete notion of a service.

With respect to their actual implementation, several problems arise from the way services for mobile and non mobile users are currently designed. The design requires defining the operating conditions, i.e., the conditions under which a program is designed to provide a service to a customer. User requirements are typically captured by statistics and analysis are performed via interviews or via field observation. Of course, these operations are extremely time-consuming, impacting the design and the test of the service. Furthermore, as soon as customers start using the service, their needs become more demanding and they start seeking for more advanced functionalities. The process of adaptation to users' needs is of strategic importance even after the service is provided on a large scale: market analysts are typically able to understand the degree of satisfaction of users in a continuous interaction with a statistical sample of them. Designers are then alerted that new functionalities are required and they provide new releases in the shortest possible time.
In an always-connected scenario, providers remotely furnish customers with upgrades or updates of the software in order to enhance the perceived quality-of-experience and satisfaction level. Let us think for example of the existing services for automatic mechanisms that are commonly employed to update softwares running on desktop PCs and laptops. Typical examples are Windows OS updates, Java plug-ins and a plethora of services where a new update is automatically installed to the customer as soon as providers release them . This is also the case of the updates of the maps of a GPS device. Service update processes are cited often as an example of autonomicity in centrally controlled system [94], [95], [96].
In general, it is common to refer to these examples as "autonomic systems". Such systems are expected to represent one of the major technological breakthroughs in the next decades, and to lead to a deep change in the way we interact with technologies nowadays. The term "autonomic" comes from the computing field [97], where it is used to define systems which are inherently self-configuring, self-optimizing, self-healing and self-protecting. As the term itself suggests, an autonomic system should show the same properties of the human autonomic nervous system, which controls routine tasks such as blood pressure, hormone levels, heart and breathing rate without requiring direct control of the central nervous system. Of course, the human nervous system could be taken as a benchmark reference system, able of automated operations, showing also remarkable robustness and reliability characteristics, but the mapping of such desirable features in man-crafted artificial systems appears at present still vague and far from a well-established practice.

Methods for Distributed Service Evolution

The rapid growth of the number of handheld devices carried by mobile users poses new challenges in the way the above described autonomic features can be made available for massively distributed applications, where centralized management is unpractical for scalability reasons. Mobile devices, in fact, are becoming a place where users keep track of their appointments, save their contacts, keeps their favorite playlist of songs, look for the closest path to a restaurant through GPS enabled services, play stand-alone games, share their profile and even data with close by users with local connectivity such as Bluetooth, IrDA or WiFi [98], [99] [79]. Handling such a halo of services in a centralized fashion is out of question, but how to make them autonomous, represents still an open challenge. The situation is made worse by the fact that customers, while acquiring more and more expertise with their devices, become increasingly aware of the offered services and capable of using them in an advanced way. Owners of mobile phones, for example, are continuously seeking for more complex and personalized functionalities that can better satisfy their requirements. An autonomic service able to upgrade without any direct action required to the user (such as remote download of patches and released upgrades) would be strongly recommended, somehow understanding the needs of the user and transparently adapting its functionalities to the user of the device.

In the presence of high mobility and increased need of user-situated services, a centralized control for every single instance of a service running on top of users' device would not work for various reasons. In fact, customized options might require a too fine grained resolution in the design of a service. Also, several softwares deployed on mobile phones or handheld devices are designed to work without the need of remote connectivity, providing services that range from entertainment to office organization or even wellness and healthcare (as for instance in [100], [101]). The continuous adaptation of these and other locally running services relies only on the ability and willingness of the user to update the service itself. Furthermore, the application scenario shows a high level of variability, with requirements possibly dependent on the geographic location of the user and the local community where she/he is actually immersed. It is almost impossible to follow such a continuous variation of needs and update the software providing the service in a customized, personalized way. The common feeling nowadays is more that a user has to adapt to what the service provides rather than offering an increasing satisfaction thanks to transparent adaptation of the service to her/his needs, if she/he is running a program with any centralized connection. A scenario of high mobility of customers is then seen as a strong limitation to the opportunity of introducing autonomicity for the previously described services.

In what follows we will show that mobility can be leveraged in order to make services adaptable to users' needs, and we will refer to a class of services able to evolve in a distributed fashion, e.g. adapting to better satisfy the user expectations. We will show that, under proper conditions, such services can be designed in order to drift automatically to match certain requirements, with no need of advanced intelligence on the device nor centralized control. In our scheme, the engine for such distributed optimized scheme is the local intermittent connectivity of mobile users which share the same service. We denote the whole process as recombination, where exchange of "genetic material" (in the form of service bluepring) takes place upon meetings between devices running similar services, mimicking what happens in biological evolution. Evolution is driven by natural selection, embedded in recombination policies which work on the basis of the service fitness, intended as ability of the service to match the user request. In a communication-oriented application scenario, this principle translates to the survival of the most satisfying versions of a service. The data exchange assumes the larger meaning of snippets of code, modules of software or even parameters, basically all that information that are needed to increase or simply change the functionality of a software that offers a service.

The T9 Service Example

In the following, we propose a simple example to make the previously described approach more concrete. In our benchmark example, we consider a reduced keyboard disambiguation system for mobile phones, where the limited keyboard has typically three or more alphabet letters associated with each key. This limitation has brought designers to introduce the well-known T9 service, a predictive text software that makes it faster and easier to type on small mobile devices when writing SMS, emails, notes or also while chatting with other users [102]. Pre-loaded with thousands of words, emoticons, and punctuation, T9 predicts what the user wants to say while typing. Every time the user types in a word that T9 does not recognize, this word is saved and it will be recognized next time. The correspondence between a sequence of number and related words is not unique: for example the sequence 568359 corresponds to the words 'lovely' and 'loudly'. It is then up to the user to switch between these words and possible alternatives, which are ordered following a ranking, decided by the T9 service. Obviously, the dictionary has to be as wide as possible, so that users from all around the world can experience a good average degree of satisfaction with the same set of words, also when starting using T9 from scratch. On the other side, it is common that a community of customers geographically or socially related to each other use a particular dictionary for instance related

to slang or local places or activities. It is clear that all these terms cannot be included by mobile phones developers and the progressive customization of the dictionary may become a lengthy and cumbersome process. Notice that in this case, the fitness can be objectively measured as a function of the number of times the user switches to find the desired word related to the digitized sequence and of the number of times she/he has to add a new word to complete her/his sentence. Users that are geographically close or that share the same interests (e.g. horse riding, rock climbing) are expected to change the set of words in a similar way, since they should have a similar dictionary, and they could benefit from a cooperative exchange of information among them when in local proximity. Let us assume that the T9 service is able to measure its user's degree of satisfaction and at the same time it is possible to know the fitness of a close by user in a totally transparent way. The service could be augmented if the user with the lower fitness value could merge its dictionary with the one of the other user that experiences a higher degree of satisfaction. If this process is repeated every time two users meet each other, we expect the average fitness of a local community to increase as time goes on. The higher the mobility, the more useful words spread and the average fitness increases. After meeting with several mobile users, the T9 service will present to its user an updated dictionary, such that the number of times she/he is expected to introduce new words to complete sentences is considerably reduced. Efficient and optimal techniques for merging will be presented in the reminder of the paper.

4.5.1 System Model

According to the framework previously outlined, we are interested in modeling the evolution process of services over a disconnected wireless network architecture. It is worth remarking that, from the user's point of view, the process of distributed service evolution is completely transparent, since what it experiences, in reality, the change in quality of experience provided by the service.

In this paper, we are interested in understanding how some factors (i.e., the number of nodes, the nodes speed, the mobility model) affect the evolution process. In terms of fitness, we expect that services with a higher degree of fitness will have a higher chance to survive, so that, in the recombination process, their *genes* (e.g., routines, code parameters etc.) are likely to be inherited by the offsprings.

In the T9 service example, the dictionaries that are more complete and then let the users experience a higher degree of satisfaction are most likely to survive after recombination. The *genes*, in fact, represents the words of the dictionaries, and those that will be spread among

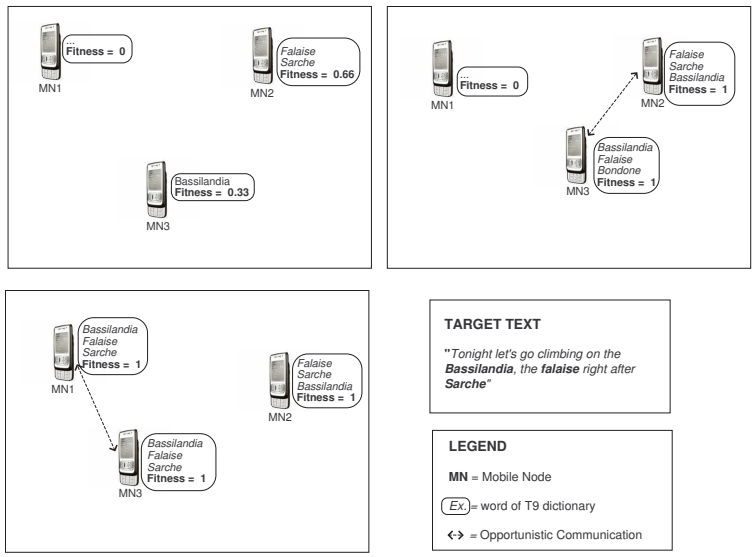


Figure 4.30: The distributed evolution of the T9 service.

T9 users are the most used among a localized community. As a particular case, we are interested in understanding the behavior of this service as a concrete example of the proposed distributed cooperative evolution process. A simple example of the T9 service distributed evolution is show in Fig. 4.30, where a user in the area of Trento, Italy (MN1 in figure) , is sending a message to his friends inviting them for climbing to a falaise called “Bassilandia”, located close to a village named “Sarche”, as indicated in the “Target Text” box. Unfortunately, his T9 dictionary does not contain the words *falaise, Sarche, Bassilandia*, strictly related to rock climbing and the particular area involved, respectively. In order to keep the example simple, in Fig. 4.30 we consider *falaise, Sarche, Bassilandia* as the target dictionary for the T9 service. When meeting other users, MN3’s T9 service will find MN2’s fitness higher (MN2 has also enriched his dictionary after meeting with MN3) and will start the dictionary exchange, getting all the three missing words. After a while, without entering any new word, he will try to send the sms including the target text, immediately finding all the needed words and experiencing then a higher fitness level.

The framework used in the following builds upon the assumptions and notation we are here

describing. We denote the fitness level of user i at time t as $I_i(t)$, and assume that $0 \leq I_i(t) \leq 1 \; \forall i = 1, \ldots, N$, where N is the total number of users in the network. A fitness level equal to '1' denotes perfect adaptation to the user needs and environment features, whereas a fitness level equal to '0' denotes a useless service. We can group the fitness level of all users into an N-dimensional vector $\underline{\hat{I}}(t) = [I_1(t), \ldots, I_N(t)]$. Assuming that the users' requirements and the environmental features are slowly changing over both time and space dimensions, $\underline{\hat{I}}(t)$ will change only at the recombination instants. If the mobile speed is finite, the recombination instants can be grouped to form a stricly monotone sequence $\{t_k\}_{k \in \mathbb{N}}$, $t_{k+1} > t_k$. In general, $\underline{\hat{I}}(t)$ will then be a random jump process defined on a suitable probability space $\{\Omega, \mathcal{F}, \mathbb{P}\}$. We denote by $\mathbb{E}[\cdot]$ the expectation taken with respect to the measure induced by $\mathbb{P}$. By standard arguments, we can transform $\underline{\hat{I}}(t)$ into a right-continuous left-limited (*càdlàg*) process, that will be denoted by $\underline{I}(t)$. In order to study the system evolution, we can then limit our scope to the embedded process $\underline{I}(t_k)$, where t_k denotes the k-th recombination time. Please note that the recombination times will be defined as a subset of the meeting times (i.e., the time instants two or more nodes get into mutual communication range), depending on the actual service recombination policy employed (see below for the definition of the three cases considered).[8] Also, note that the mobility pattern of nodes plays a crucial role, in that it determines the intensity (and distribution) of the sequence of meeting times, and thus the convergence rate of the service evolution process.

4.5.2 Performance Evaluation and Numerical Results

In this section, we present numerical results on the convergence time of the aforementioned recombination policies. All simulations were performed using a freely available software tools [103]. We simulated, for the whole range of simulation parameters, the two optimal service recombination policy outlined above, namely, clone-and-mutate and combine-and-mutate. In all simulations, nodes are assumed to use IEEE 802.11b-compliant PHY and MAC protocols; the communication range is set to 50 m. To model the mobility of users and devices, a synthetic mobility pattern, namely the Random Waypoint Mobility (RWM) [104] is used. In RWM nodes initially generate a destination (chosen uniformly over the simulation area) and subsequently move along a straight line at (constant) speed v until the destination is reached. The process is then repeated. In our simulations, we assumed v to be uniformly drawn, for each path, in the set $\{6, 15\}$ m/s. We assume nodes to be initially distributed according to a

[8] This accounts for the fact that not all meeting instants may be used for performing recombination, depending on the fitness value of the nodes getting within mutual communication range as well as on the particular recombination policy employed.

uniform distribution over the simulation area.[9] For all simulations, 95% confidence interval is reported, averaged over the set of simulation runs. We run 100 simulations for each point and each simulation was lasting for at most 72 hours. We denote by N the total number of nodes in the system. In order to measure the performance of the system, we introduce two metrics related to the *convergence time* of the evolutionary processes. First, fixing a threshold ξ (in the simulation, we will use $\xi = 0.90$), we want to measure the time it takes for the average fitness level to exceed ξ. Formally, $T_{conv}^{avg} = \min \left(t : \frac{\sum_{i=1}^{N} I_i(t)}{N} \geq \xi \right)$. Then, we are interested in the time it takes for all the fitness values to exceed ξ, i.e., $T_{conv}^{min} = \min \left(t : \min_{i=1,\ldots,N} (I_i(t)) \geq \xi \right)$. Clearly, $T_{conv}^{min} \geq T_{conv}^{avg}$. Further, the smaller such convergence times, the more efficient the evolution process and the ability of the service to adapt to rapidly changing environmental conditions.

In order to understand the performance of a realistic service, we also implemented and tested the performance of the proposed evolutionary mechanisms, when applied to the T9 service example.

We model the complete T9 dictionary with $D = \{(1, f_1), (2, f_2), \ldots, (K, f_K)\}$, where i is an integer number representing a given word and f_i is the corresponding frequency of usage. We assume that the frequency of usage of words follow a Zipf's Law [106], which is known to be a good model for the English language. We have therefore:

$$f_i = \frac{1}{i \sum_{n=1}^{N} \frac{1}{n}},$$

The amount of memory on each device allows to store a maximum of M words, where M is assumed to be one third of K. Further, we denote by D^* the optimal dictionary, which is again of size M. [10]

[9] Please note that our simulation of RWM is *not* a perfect simulation [105]. Indeed, we do not start the simulation with nodes distributed spatially according to the steady-state distribution, but, rather, with a uniform one. This represents a pessimistic assumption, in that, as it may be easily understood, a uniform distribution of nodes over the area of interest is the distribution yielding the lowest probability of having nodes connected to each other (or, alternatively, the highest node isolation probability). Nonetheless, we believe that such an assumption leads to meaningful results in terms of comparison of the performance obtainable with RWM.

[10] Note that the optimal dictionary does not correspond to the one including the most highly ranked words, but the one which includes the words which are expected to be more used in a given area.

The fitness level I_i of user i is evaluated as follows:

$$I_i = 1 - \sum_{j \in D^*} \eta_j \chi \left(j \notin D_i \right) \tag{4.7}$$

where $\eta_j = \frac{f_j}{\sum_{l \in D^*} f_l}$ and the function $\chi(\cdot)$ is the indicator function of the event $\cdot$. In other words, the fitness I_i equals one minus the weighted frequency of usage of those words that are in D^* but not in D_i. In particular, it can be noticed that the fitness is 1 if D^* and D_i are identical, 0 if none of the words of D^* belongs to D_i.

The mutation operator is implemented by substituting, with a given probability, a word in D_i (randomly chosen) with one in D (randomly chosen among the ones not present in D_i to avoid duplication). The crossover operator is applied by drawing at random a crossover point in $\{1, \ldots, M-1\}$ and substituting the tail of the dictionary (i.e., all words occurring after the crossover point) with the corresponding ones in the dictionary of the other node involved in the recombination.

At the beginning of the simulations, each node is assigned a dictionary constituted by M words randomly chosen among the ones in D. First, we consider (as in the previous section) the time necessary for the algorithm to converge. The convergence is in this case understood as the time necessary for the average fitness to exceeds the 0.9 value. In this case, the playground size is set to $1000 \times 1000\ m^2$.

We start assuming the dictionary length equal to $K = 3 \times M = 150$ words, set the crossover probability to 0.25 and vary the mutation probability. The results are reported in Fig. 4.31. As it can be seen, the performance of the system turns out to be not very sensitive to the value of the mutation probability, provided that the latter exceeds 0.02. For lower values of the mutation probability (and in particular for the case in which no mutations are performed) the performance drop significantly, and becomes also less predictable (as can be seen by the very large confidence interval).

We then evaluated the sensitivity of the system with respect to the crossover probability. We set the mutation probability to 0.05. The results are plotted, in semilogarithmic scale, in Fig. 4.32. As for the simple chromosome experiments, also in this case higher values for the crossover probability lead to an enhancement in the system's performance, in terms of reduced convergence time.

We also checked the scalability in terms of dictionary size. The results are plotted in Fig. 4.33, for mutation probability equal to 0.05 and crossover probability 0.25. It can be noted that the convergence time grows linearly with the dictionary length. This is consistent

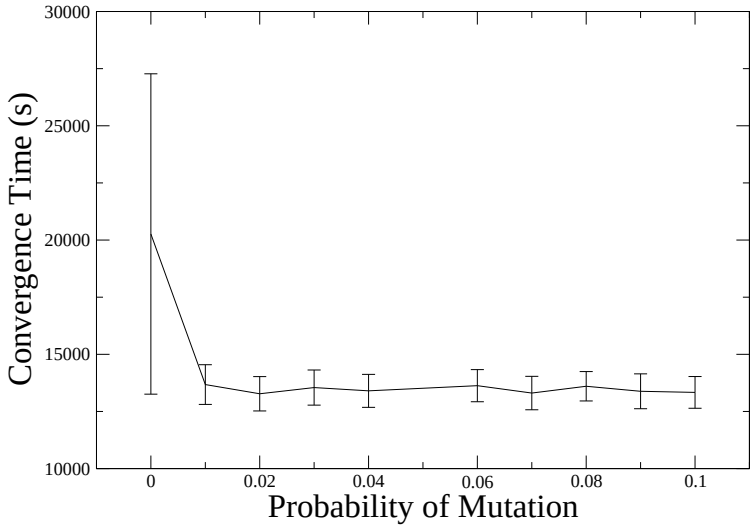


Figure 4.31: Convergence time as a function of the mutation probability for the T9 service, crossover probability set to 0.25.

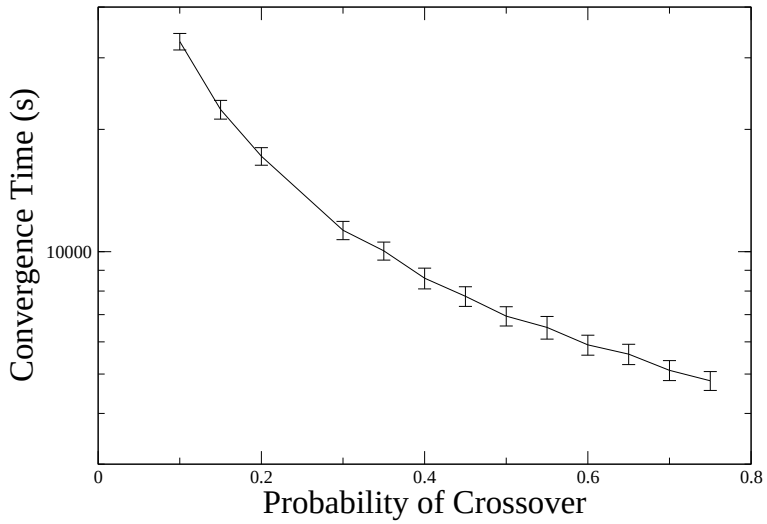


Figure 4.32: Convergence time as a function of the crossover probability for the T9 service, mutation probability set to 0.05.

with the results in the simple chromosome case, in that the dimension of the search space is here equal to three times the dictionary length.

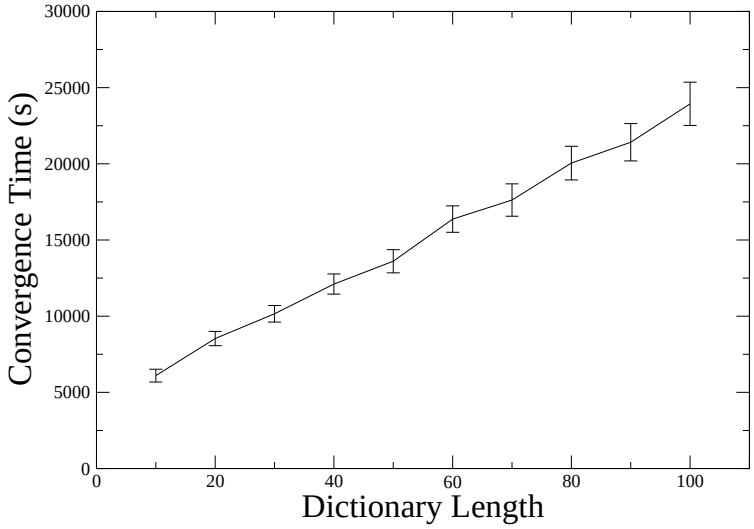


Figure 4.33: Convergence time as a function of the dictionary length M, crossover probability set to 0.25 and mutation probability set to 0.05.

Last, we performed some tests with a more complex simulation scenario. In this case, the simulation area is set to 2000×2000 m^2 and divided in four zones, as:

Z_1	Z_2
Z_3	Z_4

Nodes move according to the RWM. With a given probability (called "migration probability"), they select a suitable destination over the whole simulation area; otherwise, the destination is randomly selected in the zone in which the node is located. The dictionary length M is set to 100 words. The optimal dictionary corresponding to each zone is drawn independently of the one of nearby zones. For this set of experiments, the convergence threshold is set to 0.9.

The results, for a crossover probability of 0.5 and mutation probability varying in the set $[0.05, 1]$ are plotted in Fig. 4.34 against the migration probability. It can be seen that, in general (and as expected), the lower the migration probability, the better the performance of the system. In this case, indeed, the migration process has the net effect of introducing some

form of "noise" in the evolutionary process. In terms of impact of the mutation probability, it can be seen that the best performance is achieved for a value of 0.2. We recall here that all simulations that did not complete were stopped after 72 hours: this explains the saturation that can be observed when the probability of migration tends to 0.3.

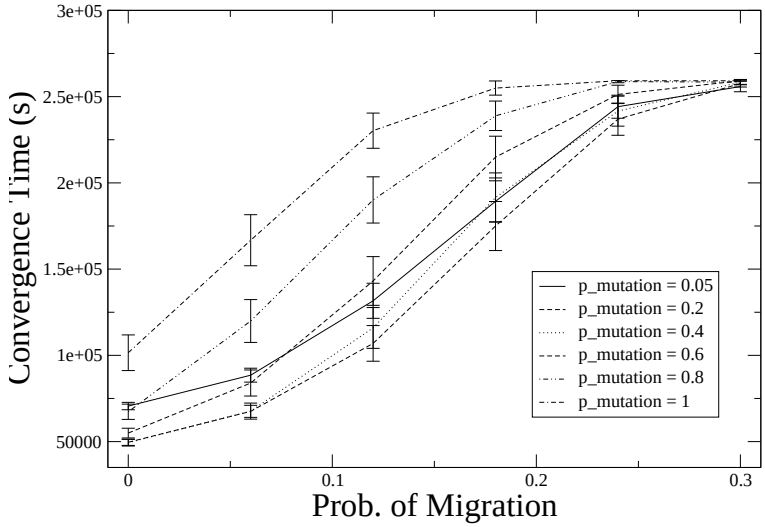


Figure 4.34: Convergence time as a function of the migration probability for the T9 service with four zones, crossover probability set to 0.5, mutation probability in the range $[0.05, 1]$.

On the other hand, it may be expected that the migration process has a beneficial effect for those cases in which some randomness is needed to ensure the convergence to the optimal dictionary set. We therefore explored the performance in the presence of very low mutation probability, taken in the set $[0, 0.1]$. For such case, we considered a convergence threshold of 0.8. (With a convergence threshold of 0.9, most of the simulation runs were either converging extremely slowly or not converging at all - for null migration probability.) As it can be seen from Fig: 4.35, a minimum level of migratory nodes indeed turns out to have a positive effect for the case no mutation operator is employed.

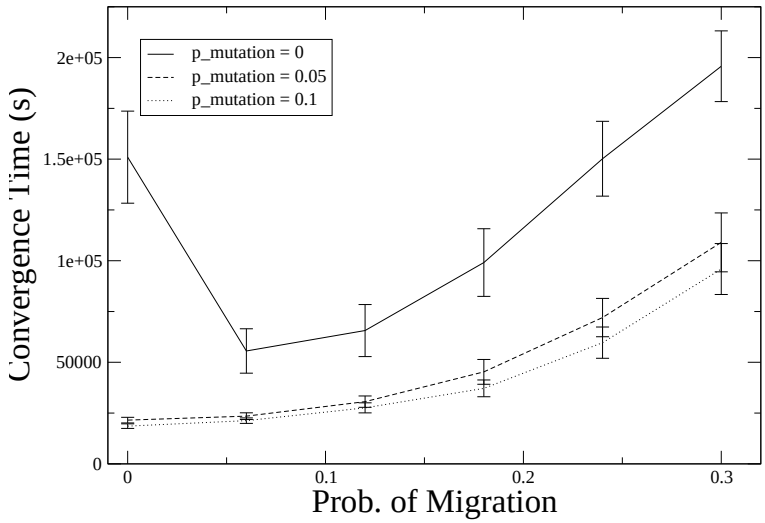


Figure 4.35: Convergence time as a function of the migration probability for the T9 service with four zones, crossover probability set to 0.5, mutation probability in the range $[0, 0.1]$.

4.5.3 Discussion

We have proposed a framework for building autonomic services in ubiquitous computing environments. The proposed approach is based on the use of operators commonly used in Evolutionary Computing for building methods (recombination policies) for cooperative distributed service evolution. A characterization of optimal recombination policies, in terms of sufficient conditions, is provided. Recombination policies are also studied in terms of their dynamic behavior (i.e., speed of convergence), and optimality conditions are provided. Numerical simulations are performed to evaluate the impact, on the system design, of some system parameters.
Among the various research directions worth being pursued, the most promising one represent the application, to the proposed framework, of a non-linear genotype-to-phenotype mapping. Of particular interest appears the application, to the proposed framework, of techniques from Artificial Embryology [107], which have the potential to turn our approach into a viable one for more complex and realistic services.

4.6 Closing Remarks

In this chapter we have presented an application of the proposed system architecture to a Mobile Social Network scenario where nodes exchange information in a totally opportunistic manner.
In particular we have initially addressed the issues arising when developing a real prototype of the proposed system architecture for Bluetooth enabled java phones, and we have presented a software architecture specifically tailored for opportunistic diffusion scenarios as the one of mobile social network. In fact, the developed U-Hopper is an attempt to provide a reusable software platform capable to transparently handle opportunistic data dissemination among mobile nodes over off-the-shelf hardware and software platforms.
By leveraging on the developed software platform, we evaluated a specific use case in an office environment, taking into consideration the realistic contact patterns of users, as measured in a 4 week period, as well as users' preferences, as obtained from questionnaires. We then evaluated how data diffusion varies, depending on these two system design dimensions. Furthermore, we have presented and evaluated a class of opportunistic data diffusion mechanisms, tailored to a data–centric architecture, that exploit the concept of "neighbourhood" for offering a robust trade-off between user–perceived performance and usage of resources. Results are evaluated by leveraging on real world contact pattern traces and emulating the diffusion of

data among users according to previously collected preferences and interests.
Finally, we have proposed a framework for building autonomic services in ubiquitous computing environments. The proposed approach is based on the use of operators commonly used in Evolutionary Computing for building methods (recombination policies) for cooperative distributed service evolution. The described approach is applied to the T9 service for mobile phones and the performance evaluated through an extensive simulation study.

5 Conclusions

In this thesis, a system architecture specifically tailored for mobile pervasive computing environments is presented. We have initially assumed a scenario in which mobile nodes gather data from sensing nodes embedded in the environment and exchange information in a totally distributed fashion, without the help of a centralized connection. The proposed system architecture is therefore composed by three different layers: (i) the layer of sensing nodes, where queries and sensed information are diffused through multihop communication, i.e. the *sensing nodes network* (ii) the layer of mobile nodes, where moving nodes communicate through simple one-hop communication in a peer to peer manner, i.e. the *mobile nodes network*,(iii) the layer of gateway nodes that realize the inter-communication among the two mobile nodes and the sensing nodes networks.

This system architecture has been applied to the application scenario of Intelligent Transportation Systems (ITS) and to that of Mobile Social Networks (MSN).

In ITSs, mobile nodes gathers sensing nodes through the gateway nodes in order to retrieve information on the surroundings and enhance the driver experience. We have considered in particular the example of cars querying a large WSN in order to find free parking lots. Embedded nodes are assumed to sense a parking place to be free or occupied and to deliver information through multi-hop communication, by following the mobile nodes (the so called mobile sinks) while moving around. A geographic routing framework specifically tailored for the described scenario is presented and the performance evaluated through an extensive simulation study. We have shown that the presented routing framework is able to overcome the challenging scenario of a mobile sink querying a network, by delivering information even if the mobile sink itself experience disconnections from the gateway nodes disposed along the road. We have also presented two different techniques in order to balance the load of traffic along the network, based on the knowledge of neighbours energy or delay. These techniques are shown to prolong the network lifetime while guaranteeing the same performance in terms of latency and packet delivery.

By going a step further in the same application scenario, we have considered the data manage-

ment issues arising at the mobile nodes network layer, when mobile nodes gather information from sensing nodes and exchange it among them in an opportunistic fashion. We have assumed the sensing nodes to be organized as a grid and we have then modeled the sensing image as a random field. By leveraging on wavelet compression techniques, we have shown through a simulation study that nodes are able to store a large amount of information with different compression levels, depending on the geographic distance between the node itself and the gathered data. In particular, results show that for a sufficiently large number of mobile nodes, the proposed scheme is capable of tracking the random field changes, while significantly reducing the storage and communication resources required on mobile nodes.
Future work in this application scenario will be devoted to a real implementation of the proposed system architecture and to a deeper investigation of the routing framework and the data management techniques.

We have then applied the proposed system architecture to the MSN application scenario. At first we have faced the challenges arising when developing a real prototype for off the shelf mobile devices. We have presented U-Hopper, a software architecture designed as a middleware and an attempt to provide a reusable software platform capable to transparently handle opportunistic data dissemination among mobile nodes over off-the-shelf hardware and software platforms. In particular, it is the capability of U-Hopper to exploit the physical mobility of user in order to gather information and diffuse it to other mobile nodes. The platform has been developed over java-enabled smartphones, leveraging Bluetooth connectivity for achieving proximity communications.
By leveraging on the developed software architecture, we evaluated a specific use case in an office environment, taking into consideration the realistic contact patterns of users, as measured in a 4 week period, as well as users' preferences, as obtained from questionnaires. We then evaluated how data diffusion varies, depending on these two system design dimensions. Results show how the design of such systems is a complex, non-trivial tasks and needs to account for many factors, both technological and user-defined, in order for the system to support the desired amount of traffic, and for users the effectively receive the content they are interested in. Future work will be devoted to the implementation and evaluation of the entire system architecture, with the aim of understanding the broader social impacts of the everyday use of such a system.
Furthermore, we have designed a class of opportunistic data diffusion mechanisms, tailored to a data–centric architecture, that exploit the concept of "neighbourhood" for offering a robust trade-off between user–perceived performance and usage of resources. The basic idea

is to let nodes gather content of potential interests for their local neighbourhood (defined, roughly speaking, as the set of nodes with which they can best communicate), trading off resources (storage and communication, hence energy) for performance (number of interested nodes reached by a given message). Algorithmic solutions have been proposed and implemented. Performance results were presented, exploiting the outcomes of an experimental measurement campaign carried out involving real users. The work at hand sheds some light on the possible advantages that designers can take by leveraging the "social" aspects of opportunistic communication systems. Again, the natural next step would be the experimentation of the proposed scheme in a real–world setting, with real users and real content.

Finally, we have proposed a framework for building autonomic services in ubiquitous computing environments, with the particular example of the T9 service for mobile phones. The proposed approach is based on the use of operators commonly used in Evolutionary Computing for building methods (recombination policies) for cooperative distributed service evolution. A characterization of optimal recombination policies, in terms of sufficient conditions, is provided. Recombination policies are also studied in terms of their dynamic behavior (i.e., speed of convergence), and optimality conditions are provided. Numerical simulations are performed to evaluate the impact, on the system design, of some system parameters.

Among the various research directions worth being pursued, the most promising one represent the application, to the proposed framework, of a non-linear genotype-to-phenotype mapping.

Bibliography

[1] M. Satyanarayanan, "Pervasive computing: vision and challenges," *Personal Communications, IEEE [see also IEEE Wireless Communications]*, vol. 8, no. 4, pp. 10–17, 2001.

[2] M. Weiser, "The computer for the 21st century," *ACM Mob. Comput. Commun. Rev.*, vol. 3, no. 3, pp. 3–11, 1999.

[3] M. Grossglauser and D. Tse, "Mobility increases the capacity of ad hoc wireless networks," *IEEE/ACM Trans. on Netw.*, vol. 10, no. 4, pp. 477–486, Aug. 2002.

[4] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott, "Impact of human mobility on the design of opportunistic forwarding algorithms," in *Proc. of INFOCOM*, Barcelona, Spain, April 23–29, 2006.

[5] J. Leguay, A. Lindgren, J. Scott, T. Friedman, and J. Crowcroft, "Opportunistic content distribution in a urban setting," in *Proc. of ACM Chants*, Florence, IT, September 15, 2006.

[6] L. Pelusi, A. Passarella, and M. Conti, "Opportunistic networking: data forwarding in disconnected mobile ad hoc networks," *IEEE Comm. Mag.*, vol. 44, no. 11, Nov. 2006.

[7] M. Karpinski, A. Senart, and V. Cahill, "Sensor networks for smart roads," in *PerCom Workshops*, Pisa, Italy, 2006, pp. 306–310.

[8] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, pp. 393–422, 2002.

[9] G. Werner-Allen, J. Johnson, M. Ruiz, J. Lees, and M. Welsh, "Monitoring volcanic eruptions with a wireless sensor network," in *Wireless Sensor Networks, 2005. Proceeedings of the Second European Workshop on*, pp. 108–120.

[10] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson, and D. Culler, "An analysis of a large scale habitat monitoring application," in *In Proc. of SenSys*, New York, NY, USA, 2004, pp. 214–226.

[11] I. Akyildiz and I. Kasimoglu, "Wireless sensor and actor networks: research challenges," *Ad Hoc Networks*, vol. 2, no. 4, pp. 351–367, 2004.

[12] J. Bohli, A. Hessler, O. Ugus, and D. Westhoff, "A secure and resilient WSN roadside architecture for intelligent transport systems," in *Proceedings of the first ACM conference on Wireless network security*. ACM New York, NY, USA, 2008, pp. 161–171.

[13] W. Chen, L. Chen, Z. Chen, and S. Tu, "WITS: A Wireless Sensor Network for Intelligent Transportation System," in *Proceedings of the First International Multi-Symposiums on Computer and Computational Sciences-Volume 2 (IMSCCS'06)-Volume 02*. IEEE Computer Society Washington, DC, USA, 2006, pp. 635–641.

[14] S. Saroiu, K. Gummadi, and S. Gribble, "Measuring and analyzing the characteristics of Napster and Gnutella hosts," *Multimedia Systems*, vol. 9, no. 2, pp. 170–184, 2003.

[15] J. Liang, R. Kumar, and K. Ross, "Understanding KaZaA," *Manuscript, Polytechnic Univ*, 2004.

[16] D. Qiu and R. Srikant, "Modeling and performance analysis of BitTorrent-like peer-to-peer networks," in *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*. ACM New York, NY, USA, 2004, pp. 367–378.

[17] A. Yu and S. Vuong, "MOPAR: a mobile peer-to-peer overlay architecture for interest management of massively multiplayer online games," in *Proceedings of the international workshop on Network and operating systems support for digital audio and video*. ACM New York, NY, USA, 2005, pp. 99–104.

[18] T. Hopfeld, K. Tutschku, F. Andersen, H. de Meer, and J. Oberender, "Simulative performance evaluation of a mobile peer-to-peer file-sharing system," *Next Generation Internet Networks, 2005*, pp. 281–287, 2005.

[19] N. Kotilainen, M. Weber, M. Vapa, and J. Vuori, "Mobile Chedar-a peer-to-peer middleware for mobile devices," in *Pervasive Computing and Communications Workshops, 2005. PerCom 2005 Workshops. Third IEEE International Conference on*, 2005, pp. 86–90.

[20] G. Ding and B. Bhargava, "Peer-to-peer file-sharing over mobile ad hoc networks," in *Proceedings of IEEE Annual Conference on Pervasive Computing and Communications Workshops*, 2004, pp. 104–109.

[21] L. Tong, Q. Zhao, and S. Adireddy, "Sensor networks with mobile agents," in *Proc. of IEEE MILCOM*, 2003.

[22] L. Song and D. Hatzinakos, "A cross-layer architecture of wireless sensor networks for target tracking," *IEEE/ACM Transactions on Networking (TON)*, vol. 15, no. 1, pp. 145–158, 2007.

[23] R. Shah, S. Roy, S. Jain, and W. Brunette, "Data mules: Modeling a three-tier architecutre for sparse sensor networks," in *Proc. of IEEE SNPA*, 2003.

[24] K. Lan and Z. Wu, "On the feasibility of using public transport as data mules for traffic monitoring," in *Intelligent Vehicles Symposium, 2008 IEEE*, 2008, pp. 979–984.

[25] G. Anastasi, M. Conti, and M. Di Francesco, "Data Collection in Sensor Networks with Data Mules: an Integrated Simulation Analysis," in *Proc. of IEEE ISCC 2008*, 2008, pp. 1096–1102.

[26] W. Wang, V. Srinivasan, and K. Chua, "Using mobile relays to prolong the lifetime of wireless sensor networks," in *Proc. of ACM MobiCom.*, 2005.

[27] ——, "Extending the Lifetime of Wireless Sensor Networks Through Mobile Relays," *Networking, IEEE/ACM Transactions on*, vol. 16, no. 5, pp. 1108–1120, 2008.

[28] J. Luo, J. Panchard, M. Piorkowski, M. Grossglauser, and J. Hubaux, "MobiRoute: Routing Towards a Mobile Sink for Improving Lifetime in Sensor Networks," *LECTURE NOTES IN COMPUTER SCIENCE*, vol. 4026, p. 480, 2006.

[29] S. Basagni, A. Carosi, and C. Petrioli, "Controlled Vs. Uncontrolled Mobility in Wireless Sensor Networks: Some Performance Insights," in *Proc of IEEE VTC-2007*, 2007, pp. 269–273.

[30] D. E. Thomas Schoellhammer, Ben Greenstein, "Hyper: A routing protocol to support mobile users of sensor networks," CENS, Tech. Rep., January 2006.

[31] H. Luo, F. Ye, J. Cheng, S. Lu, and L. Zhang, "TTDD: Two-tier data dissemination in large-scale wireless sensor networks," *Wireless Networks*, no. 11, pp. 161–175, 2005.

[32] F. Zhao and L. Guibas, *Wireless Sensor Networks: An Information Processing Approach*. Elsevier/Morgan-Kaufmann, 2004.

[33] S. Coleri, S. Y. Cheung, and P. Varaiya, "Sensor networks for monitoring traffic," in *Proc. of Allerton Conf.*, September 2004.

[34] S. Lee and D. Y. A. Ghosh, "Intelligent parking lot application using wireless sensor networks," in *International Symposium on Collaborative Technologies and Systems*, May 2008, pp. 38–57.

[35] V. Tang, Z. Yuan, and C. Jiannong, "An intelligent car park management system based on wireless sensor networks," in *International Symposium on Pervasive Computing and Applications*, August 2006, pp. 65–70.

[36] Streetline, city infrastructure technologies. [Online]. Available: http://www.streetlinenetworks.com/

[37] A. Vahdat and D. Becker, "Epidemic routing for partially connected ad hoc networks," Duke University, Tech. Rep. CS-2000-06, 2000.

[38] A. Khelil, C. Becker, J. Tian, and K. Rothermel, "An epidemic model for information diffusion in MANETs," in *Proc. of ACM MSWiM*, Atlanta, Georgia, Sept. 28, 2002, pp. 54–60.

[39] K. Harras, K. Almeroth, and E. Belding-Royer, "Delay tolerant mobile networks (DTMNs): Controlled flooding schemes in sparse mobile networks," in *Proc. of IFIP Netwoking*, Waterloo, Canada, May, 2005.

[40] X. Zhang, G. Neglia, J. Kurose, and D. Towsley, "Performance modeling of epidemic routing," *Comput. Netw.*, vol. 51, no. 10, pp. 2867–2891, 2007.

[41] M. Musolesi and C. Mascolo, "Controlled Epidemic-style Dissemination Middleware for Mobile Ad Hoc Networks," in *Proc. of ACM Mobiquitous*, July 2006.

[42] A. E. Fawal, J.-Y. L. Boudec, and K. Salamatian, "Performance analysis of self limiting epidemic forwarding," EPFL, Tech. Rep. LCA-REPORT-2006-127, 2006.

[43] P. Costa, C. Mascolo, M. Musolesi, and G. P. Picco, "Socially-aware Routing for Publish-Subscribe in Delay-tolerant Mobile Ad Hoc Networks," *IEEE JSAC*, vol. 26, no. 5, June 2008.

[44] P. Hui and J. Crowcroft, "Bubble rap: Forwarding in small world dtns in ever decreasing circles," Univ. of Cambridge, Computer Laboratory, Tech. Rep. UCAM-CL-TR-684, May 2007.

[45] M. Musolesi and C. Mascolo, "Designing mobility models based on social network theory," *ACM SIGMOBILE Mobile Computing and Communication Review*, vol. 11, no. 3, July 2007.

[46] Reality Mining, "Machine perception and learning of complex social systems." http://reality.media.mit.edu/.

[47] A. G. Miklas, K. K. Gollu, S. Saroiu, K. P. Gummadi, and E. de Lara, "Exploiting social interactions in mobile systems," in *Proc. of UBICOMP*, Innsbruck, Austria, Sept 2007.

[48] J. Su, J. Scott, P. Hui, J. Crowcroft, E. de Lara, C. Diot, A. Goel, M. Lim, and E. Upton, "Haggle: Seamless networking for mobile applications," in *Proc. of Ubicomp*, Sept. 16-19 2007, pp. 391–408.

[49] O. Riva and S. Toivonen, "The dynamos approach to support context-aware service provisioning in mobile environments," *Elsevier Journal of Systems and Software*, vol. 80, no. 12, pp. 1956–1972, 2007.

[50] I. Carreras, I. Chlamtac, F. De Pellegrini, and D. Miorandi, "Bionets: Bio-inspired networking for pervasive communication environments," *IEEE Trans. Veh. Tech.*, 2006, in press. [Online]. Available: http://www.create-net.org/$\sim$dmiorandi

[51] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott, "Impact of human mobility on the design of opportunistic forwarding algorithms," in *Proc. of INFOCOM*, Barcelona, Spain, April 23–29, 2006.

[52] U. Lee, E. Magistretti, B. Zhou, M. Gerla, P. Bellavista, and A. Corradi, "MobEyes: smart mobs for urban monitoring with vehicular sensor networks," UCLA CSD, Tech. Rep. 060015, 2006. [Online]. Available: http://netlab.cs.ucla.edu/wiki/files/mobeyestr06.pdf

[53] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine, "MaxProp: Routing for Vehicle-Based Disruption-Tolerant Networks," in *Proc. IEEE INFOCOM*, April 2006. [Online]. Available: http://prisms.cs.umass.edu/brian/pubs/burgess.infocom2006.pdf

[54] U. Lee, E. Magistretti, B. Zhou, M. Gerla, P. Bellavista, and A. Corradi, "Efficient data harvesting in mobile sensor platforms," in *inProc. of PERCOM*. Washington, DC, USA: IEEE Computer Society, 2006, p. 352.

[55] O. Riva and C. di Flora, "Contory: A smart phone middleware supporting multiple context provisioning strategies," in *In Proc. of ICDCSW*. Washington, DC, USA: IEEE Computer Society, 2006, p. 68.

[56] M. Srivastava, M. Hansen, J. Burke, A. Parker, S. Reddy, G. Saurabh, M. Allman, V. Paxson, and D. Estrin, "Wireless urban sensing systems," CENS, Tech. Rep. 65, April 2006.

[57] A. K. Dey, "Understanding and using context." *Personal and Ubiquitous Computing*, vol. 5, no. 1, pp. 4–7, 2001.

[58] Y. Nam, H. Lee, H. Jung, T. Kwon, and Y. Choi, "An adaptive mac (a-mac) protocol guaranteeing network lifetime for wireless sensor networks," in *Proceedings of 12th European Wireless Conference (EW 2006)*, Athens, Greece, April 2006.

[59] D. Tacconi, I. Carreras, D. Miorandi, I. Chlamtac, F. Chiti, and R. Fantacci, "Supporting the sink mobility: a case study for wireless sensor networks," in *Proceedings of IEEE ICC07 (to appear)*, Glasgow, Scotland, June 2007.

[60] OMNeT++ discrete event simulation system. [Online]. Available: http://www.omnetpp.org

[61] D. L. Lee, J. Xu, B. Zheng, and W.-C. Lee, "Data management in location-dependent information services," *IEEE Pervasive Computing*, vol. 1, no. 3, pp. 65–72, 2002.

[62] J. L. Mitchell, W. B. Pennebaker, C. E. Fogg, and D. J. Legall, Eds., *MPEG Video Compression Standard*. London, UK, UK: Chapman & Hall, Ltd., 1996.

[63] G. K. Wallace, "The jpeg still picture compression standard," *Commun. ACM*, vol. 34, no. 4, pp. 30–44, 1991.

[64] D. S. Taubman and M. W. Marcellin, *JPEG 2000: Image Compression Fundamentals, Standards and Practice*. Norwell, MA, USA: Kluwer Academic Publishers, 2001.

[65] D. Ganesan, B. Greenstein, D. Estrin, J. Heidemann, and R. Govindan, "Multiresolution storage and search in sensor networks," *ACM Trans. on Storage*, vol. 1, no. 3, pp. 277–315, 2005.

[66] R. Wagner, S. Sarvotham, and R. Baraniuk, "A multiscale data representation for distributed sensor networks," in *In Proc. of ICASSP*, March 2005.

[67] I. Daubechies, *Ten lectures on wavelets*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1992.

[68] C. K. Chui, *An introduction to wavelets*. San Diego, CA: Academic Press Professional, Inc.,, 1992.

[69] E. VanMarcke, *Random fields: analysis and synthesis*. Cambridge, MA: MIT Press, 1983.

[70] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York: John Wiley & Sons, 1991.

[71] F. Alabert, "The practice of fast conditional simulationsthrough the LU decomposition of the covariance matrix," *Mathematical Geology*, vol. 19, no. 5, p. 369385, 1987.

[72] M. W. Davis, "Production of conditional simulations via the LU triangular decomposition of the covariance matrix," *Mathematical Geology*, vol. 19, no. 5, pp. 91–98, 1987.

[73] J. Walker, *A Primer on Wavelets and Their Scientific Applications*. NY, USA: Chapman and Hall:CRC, 1999.

[74] R. Rao and A. Bopardikar, *Wavelet Transforms: Introduction to Theory and Applications*. Addison-Wesley Publications, 1998.

[75] I. Carreras, D. Tacconi, D. Miorandi, and I. Chlamtac, "Multi-resolution wavelets," CREATE-NET, Tech. Rep. NA-92-08, 2006.

[76] T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for ad hoc network research," *Wiley Wireless Communications and Mobile Computing*, vol. 2, no. 5, pp. 483–502, August 2002.

[77] W. Navidi and T. Camp, "Stationary distributions for the random waypoint mobility model." *IEEE Trans. Mob. Comput.*, vol. 3, no. 1, pp. 99–108, 2004.

[78] "Bedd: Bringing people together," http://www.bedd.com/.

[79] "Juicecaster: Know first ... show first," http://www.juicecaster.com/.

[80] "imity: Social situations to go," http://www.imity.com/.

[81] S. Burleigh, L. Torgerson, K. Fall, V. Cerf, B. Durst, K. Scott, and H. Weiss, "Delay-tolerant networking: an approach to interplanetary internet," *IEEE Comm. Mag.*, vol. 41, no. 6, pp. 128–136, 2003.

[82] M. Nicolai, N. Behrens, and E. Yoneki, "Wireless rope: An experiment in social proximity sensing with bluetooth," in *Proc. of PerCom*, March 2006.

[83] X. Zhang, J. Kurose, B. N. Levine, D. Towsley, and H. Zhang, "Study of a Bus-Based Disruption Tolerant Network: Mobility Modeling and Impact on Routing," in *Proc. of Mobicom*, September 2007, pp. 195–206.

[84] BIONETS, "Biologically-inspired autonomic networks and services," http://www.create-net.eu, 2005.

[85] I. Carreras, D. Tacconi, and D. Miorandi, "Data-centric information dissemination in opportunistic environments," in *Proc. of MASS 2007 - Demo Session*, Pisa, Italy, October 2007.

[86] I. Carreras, I. Chlamtac, F. D. Pellegrini, and D. Miorandi, "Bionets: Bio-inspired networking for pervasive communication environments," *IEEE Trans. on Vehicular Technology*, vol. 56, no. 1, pp. 218–229, Jan. 2007.

[87] Jsr-000118 mobile information device profile 2.0. [Online]. Available: http://jcp.org/aboutJava/communityprocess/final/jsr118/

[88] Jsr-000082 javatm apis for bluetooth. [Online]. Available: http://jcp.org/aboutJava/communityprocess/final/jsr082/

[89] A. Vahdat and D. Becker, "Epidemic routing for partially connected ad hoc networks," 2000. [Online]. Available: citeseer.ist.psu.edu/vahdat00epidemic.html

[90] C. Gronroos, *Service Management and Marketing: A Customer Relationship Management Approach*. Chichester, UK: John Wiley & Sons, 2000.

[91] P. Kotler, *Marketing Management: Analysis,Planning, Implementation and Control, 6th edition*. Englewood Cliffs, NJ: Prentice Hall, 1988.

[92] Z. Badia, J. Gordijn, and B. Omelayenko, "A shared service terminology for online service provisioning," *Proceedings of ACM Sixth International Conference on Elettronic Commerce*, May 2004.

[93] J. OSullivan, D. Edmond, and A. H. M. ter Hofstede, "Service description: A survey of the general nature of services," Centre for Information Technology Innovation, Queensland University of Technology, Tech. Rep. FIT-TR-2003-02, 2002.

[94] R. Haas and P. D. B. Stiller, "Autonomic service deployment in networks," in *IBM SYSTEMS JOURNAL*, vol. 42, no. 1, 2003.

[95] M. G. Merideth and P. Narasimhan, "Retrofitting networked applications to add autonomic reconfiguration," *ACM SIGSOFT Software Engineering Notes archive*, vol. 30, pp. 1–7, July 2005.

[96] S. Sivasubramanian, G. Pierre, and M. van Steen, "Autonomic data placement strategies for update-intensiveweb applications," *First International Workshop on Advanced Architectures and Algorithms for Internet Delivery and Applications (AAA-IDEA'05)*, vol. 30, pp. 2–9, 2005.

[97] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *IEEE Comp. Mag.*, vol. 36, no. 1, pp. 41–50, Jan. 2003.

[98] "Nokia Sensor," http://europe.nokia.com/A4144923.

[99] "Shockfish Spotme," http://www.spotme.com/.

[100] W. Healthcare inc., "101 things to do with a mobile phone in healthcare." [Online]. Available: http://www.wirelesshealthcare.co.uk/wh/report_101.htm

[101] A. Prentza, S. Maglavera, and L. Leondaridis, "Delivery of healthcare services over mobile phones: e-vital and chs paradigms," *Proceedings of the 28th IEEE EMBS Annual International Conference*, pp. 3250–3253, August 2006.

[102] "T9 text input," http://www.t9.com.

[103] AA.VV., "Omnet++." [Online]. Available: www.omnetpp.org

[104] C. Bettstetter and G. R. ad Paolo Santi, "The node distribution of the random waypoint mobility model for wireless ad hoc networks," *IEEE Trans. on Mobile Computing*, vol. 2, no. 3, pp. 257–269, July-September 2003.

[105] J.-Y. Le Boudec and M. Vojnović, "Perfect simulation and stationarity of a class of mobility models," in *Proc. of IEEE INFOCOM*, Miami, FL, 2005.

[106] W. Li, "Random texts exhibit zipfs-law-like word frequency distribution," *IEEE Transactions on Information Theory*, vol. 38, p. 18421845, 1992.

[107] P. J. Bentley and S. Kumar, "Three ways to grow designs: A comparison of embryogenies for an evolutionary design problem," in *Proc. of GECCO*, Orlando, Florida USA, 1999, pp. 35–43.

6 Publications

1. David Tacconi, Daniele Miorandi, Iacopo Carreras, Francesco Chiti and Romano Fantacci “Using Wireless Sensor Netowrks to support Intelligent Transportation Systems” subm. to Elsevier journal on Ad Hoc Networks, Special Issue on Vehicular Networks

2. D. Tacconi, D. Miorandi, I. Carreras and F. De Pellegrini, Cooperative Evolution of Services in Ubiquitous Computing Environments, submitted to ACM Trans. Aut. Adapt. Systems.

3. Iacopo Carreras, Francesco De Pellegrini, Daniele Miorandi, David Tacconi and Imrich Chlamtac “Embedding User Interests into Forwarding Schemes: an Experimental Approach” In Proceedings of ACM MobiCom Workshop on Challenged Networks (CHANTS 2008). San Francisco, California, USA, September 2008.

4. Iacopo Carreras, David Tacconi and Arianna Bassoli “Social Opportunistic Computing: Design for Autonomic User-Centric Systems” sumbitted as book chapter to Springer Autonomic Communication

5. Arianna Bassoli, J. Brewer, K. Martin, I. Carreras and D. Tacconi “Undersound and the Above Ground” Poster Presentation at the 5th International Mobile Music Workshop (MMW 2008). Vienna, Austria, May 2008.

6. David Tacconi, Oscar Mayora, Paul Lukowicz, Bert Arnrich, Cornelia Setz, Gerhard Troster and Christian Haring. Activity and Emotion Recognition to Support Early Diagnosis of Psychiatric Diseases. In Proceedings of 2nd Pervasive Health Conference. Tampere, Finland, January 2008.

7. Luca Paolino, Alessandro M. Martellone, David Tacconi, Monica Sebillo, Genoveffa Tortora and Giuliana Vitiello, “Dynamic User Modeling for Personalized Advertisement Delivery on Mobile Devices” to appear in Springer Proc. of ecomo 2009

8. Alessandro Martellone, David Tacconi, Vincenzo Del Fatto and Giuliana Vitiello “LUNA AdsSustaining Wireless Access for Mobile Users”, in Springer Proc. of 10th International Conference Visual 2008,

9. Iacopo Carreras, David Tacconi, Margarita Anastassova and Oscar Mayora ”BluePlanner: a personal support for opportunistic mobile environments” Mobile Experience - In 10th International Conference on Human-Computer Interaction with Mobile Devices and Services (MOBILEHCI 2008). Amsterdam, the Netherlands, Sept. 2008.

10. David Tacconi, Oscar Mayora, Paul Lukowicz, and Bert Arnrich. ”A Context Aware Framework for Trend Analysis of Long Term Diseases”. Interaction with Ubiquitous Wellness and Healthcare UBIWELL’07. UBICOMP’ Workshop Innsbruck, Austria 2007

11. Iacopo Carreras, David Tacconi, Daniele Miorandi “Data-Centric Information Dissemination in Opportunistic Environments” Demo Session - In Proceedings of MASS 2007, Pisa, Italy, October 8-11, 2007.

12. Iacopo Carreras, David Tacconi “U-Hopper: User-centric Heteogeneous Opportunistic Middleware” Demo Session - In Proceedings of SAC (BIONETICS Workshop), Budapest, Hungary, December 2007.

13. David Tacconi, Iacopo Carreras, Daniele Miorandi, Francesco Chiti, Antonio Casile and Romano Fantacci “A System Architecture Supporting Mobile Applications in Disconnected Sensor Networks” In Proceedings of IEEE Global Communications Conference (IEEE GLOBECOM 2007), Washington, USA, November 2007.

14. I.Carreras, D.Tacconi, D.Miorandi, I.Chlamtac, “A Multi-Resolution Data Management Scheme for Opportunistic Information Diffusion” in Proc. of MDM, Mannheim (Germany), May 2007

15. D. Tacconi, I. Carreras, D. Miorandi, I. Chlamtac, F. Chiti and R. Fantacci, “Supporting the Sink Mobility: a Case Study for Wireless Sensor Networks”, in Proc. of IEEE ICC, Glasgow (UK), June 2007.

16. David Tacconi, Hagen Woesner, Imrich Chlamtac and Chao Xie “Evaluation of Diversity Techniques for the Indoor Diffuse Infrared Channel” in Proc. of IEEE Wireless Communications and Networking Conference, 2007. WCNC 2007.

List of Figures

List of Tables