ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Ph.D. Course in
ELECTRONICS, COMPUTER SCIENCE AND TELECOMMUNICATIONS
Cycle XXV

Examination Sector 09/H1
Scientific Disciplinary Sector ING-INF/05

PERVASIVE SENSING IN FUTURE NETWORKS

Candidate:
GIUSEPPE CARDONE

Supervisor:
PROF. ANTONIO CORRADI

Ph.D. Course Coordinator:
PROF. ALESSANDRO VANELLI-CORALLI

Final Examination Year 2013

Giuseppe Cardone: *Pervasive Sensing in Future Networks,* © 2013.

WEBSITE:
http://middleware.unibo.it/people/gc

E-MAIL:
giuseppe.cardone@unibo.it

«I've never seen the Icarus story as a lesson about the limitations of humans. I see it as a lesson about the limitations of wax as an adhesive.»

— Randall Munroe

To my family, my girlfriend and my friends. I am what I am because of what we are. Thank you.

ABSTRACT

PERVASIVE Sensing is a recent research trend that aims at providing widespread computing and sensing capabilities to enable the creation of smart environments that can sense, process, and act by considering input coming from both people and devices. The capabilities necessary for Pervasive Sensing are nowadays available on a plethora of devices, from embedded devices to PCs and smartphones.

The wide availability of new devices and the large amount of data they can access enable a wide range of novel services in different areas, spanning from simple data collection systems to socially-aware collaborative filtering. However, the strong heterogeneity and unreliability of devices and sensors poses significant challenges. So far, existing works on Pervasive Sensing have focused only on limited portions of the whole stack of available devices and data that they can use, to propose and develop mainly vertical solutions.

The push from academia and industry for this kind of services shows that time is mature for a more general support framework for Pervasive Sensing solutions able to enhance frail architectures, promote a well balanced usage of resources on different devices, and enable the widest possible access to sensed data, while ensuring a minimal energy consumption on battery-operated devices. This thesis focuses on pervasive sensing systems to extract design guidelines as foundation of a comprehensive reference model for multi-tier Pervasive Sensing applications. The validity of the proposed model is tested in five different scenarios that present peculiar and different requirements, and different hardware and sensors. The ease of mapping from the proposed logical model to the real implementations and the positive performance result campaigns prove the quality of the proposed approach and offer a reliable reference model, together with a direction for the design and deployment of future Pervasive Sensing applications.

PUBLICATIONS

Some ideas and figures have previously and partially appeared in the following publications:

[1] G. Cardone, A. Corradi, and L. Foschini, "Translucent Middleware Approach to Facilitate WSN Access Management," in *ISCC '10: Proceedings of the Fifteenth IEEE Symposium on Computers and Communications*, pp. 595–598, 2010.

[2] P. Bellavista, G. Cardone, A. Corradi, and L. Foschini, "The Future Internet convergence of IMS and Ubiquitous Smart Environments: an IMS-based Solution for Energy Efficiency," *Journal of Network and Computer Applications*, vol. 35, no. 4, pp. 1203 – 1209, 2012.

[3] G. Cardone, A. Corradi, and L. Foschini, "Reliable Communication for Mobile MANET-WSN Scenarios," in *ISCC '11: Proceedings of the Sixteenth IEEE Symposium on Computers and Communications*, pp. 1085 –1091, 2011.

[4] G. Cardone, A. Corradi, and L. Foschini, "Cross-Network Opportunistic Collection of Urgent Data in Wireless Sensor Networks," *The Computer Journal*, 2011.

[5] G. Cardone, P. Bellavista, A. Corradi, and L. Foschini, "Effective Collaborative Monitoring in Smart Cities: Converging MANET and WSN for Fast Data Collection," in *K-2011: Proceeding of ITU Kaleidoscope 2011: The Fully Networked Human? - Innovations for Future Networks and Services*, pp. 1 –8, dec. 2011.

[6] T. Wang, G. Cardone, A. Corradi, L. Torresani, and A. T. Campbell, "WalkSafe: a Pedestrian Safety App for Mobile Phone Users Who Walk and Talk while Crossing Roads," in *HotMobile '12: Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications*, pp. 5:1–5:6, 2012.

[7] M. Lin, N. Lane, M. M. Rabbi, X. Yang, L. Hong, G. Cardone, S. Ali, A. Doryab, E. Berke, T. Choudhury, and A. T. Campbell, "BeWell+: Multi-dimensional Wellbeing Monitoring with Community-guided User Feedback and Energy Optimization," in *WH '12: Proceedings of the Third Conference on Wireless Health*, 2012.

[8] G. Cardone, A. Corradi, L. Foschini, and R. Montanari, "Socio-technical Awareness to Support Recommendation and Efficient Delivery of IMS-enabled Mobile Services," *IEEE Communications Magazine*, vol. 50, pp. 82 –90, june 2012.

[9] G. Cardone, A. Cirri, A. Corradi, L. Foschini, and D. Maio, "MSF: An Efficient Mobile Phone Sensing Framework," *International Journal of Distributed Sensor Networks*, 3 2013.

CONTENTS

1 INTRODUCTION

PERVASIVE Sensing is a recent research trend that aims at providing widespread computing and sensing capabilities to enable the creation of smart environments that can sense, process, and act by considering input coming from both people and devices. Pervasive Sensing is driven by the progress of three different, but linked, technological areas: computing, networking, and sensing. First, enhancements to the manufacturing process of integrated circuits provides portable devices, such as smartphones, with powerful processors and plenty of memory and storage resources. Second, as computing devices have gotten smaller, they have become personal and portable terminals that can participate in dynamically configured networks. Finally, recent progress of microelectromechanical systems has enabled the manufacturing of small low-power low-cost sensors and actuators that allow devices to be connected not only to other devices via heterogeneous network, but also with the physical world.

The capabilities necessary for Pervasive Sensing are nowadays available on a plethora of devices, spanning from embedded devices to PCs and smartphones. Notably, personal devices, like tablets and smartphones, usually have access also to data generated by users, such as pictures, information on social networks, device usage statistics, and so on. This information can be managed as sensed data and allows devices to potentially use it to gather additional knowledge about the current real world context. This large amount of data that they can access paves the way to a wide range of novel services in different application areas, such as health care, environmental monitoring, and collaborative filtering, that will likely have a huge impact on our everyday lives.

The potential and challenges of Pervasive Sensing systems can be studied at three different levels of increasing abstraction. At the lowest level, that we cal *data-centric*, Pervasive Sensing focuses mostly on collection, routing, and processing of data, with particular for energy-efficiency and collection latencies. Having technologically and economically viable techniques to collect and dispatch environmental data is a prerequisite for successful deployment of Pervasive Sensing systems, and it is an essential building block of other, higher-level management functions. At the medium-level, that we call *person-centric*, Pervasive Sensing systems deal with processing sensed data to provide people-centric services, such as healthcare monitoring and lifestyle tracking. These kind of services, that are more complex than straightforward environment monitoring services, make Pervasive Sensing systems perceivably important for individuals. Finally, at the highest *social-centric* level, Pervasive Sensing systems scale up sensing and processing to human communities: data that emerge within social communities can be collected and processed to provide services that are relevant for specific social groups that spontaneously form in larger communities.

Supported by the large amount of literature by both academia and industry, we believe that time is mature for a support framework for Pervasive Sensing solutions based on reliable architectures that promote well-balanced usage of resources on different devices and enable the widest possible access to sensed data, while ensuring a minimal energy consumption on battery-operated devices. However, the wide range of devices, communication protocols, technologies, and scenarios involved in Pervasive Sensing systems pose significant issues that still have to be overcome for their successful deployment in the real world, that we group along three directions:

- *Heterogeneity*. Pervasive Sensing systems feature extremely heterogeneous devices that have severely different energy, computing, and network resources. Getting all these different devices working together requires a careful engineering of workloads assignment and communications among heterogeneous devices, that should use cross-layer informations to coordinate sensing tasks.
- *Dynamic environments*. Many devices involved in Pervasive Sensing systems exist in an inherently dynamic environment (e.g.: due to user moving and low-cost hardware failing) whose failures and inaccuracies should be tolerated by sensing systems. This causes distributed resources not to be consistently available, thus requiring an opportunistic approach to their usage.
- *Resource constraints*. Even if in certain scenarios they can rely on the Cloud, Pervasive Sensing systems unavoidably need to access embedded and mobile devices to collect sensed data. These devices have very strict constraints for processing, communicating, storing data, and battery consumption. It is important to stress that working around these limitations is not simply desirable, rather it is a core requirement for Pervasive Sensing systems.
- *Scalability*. Large scale Pervasive Sensing systems feature thousand of fixed and mobile sensors, that stream large amounts of sensed data. As the system scale grows, data sensing, routing, and processing introduce a fundamental scalability bottleneck, that should be addressed by properly tailoring and distributing sensing tasks, exploiting locality principles to improve scalability.

Several efforts in the research literature have tried to address these challenges; however, to the best of our knowledge, previous works are mostly focused on specific applications of Pervasive Sensing systems, leaving out the great deal of complexity, from the management of data collection at the low-level to the provisioning of services at a very high level. Hence, although previous works provide valid and interesting solutions, this thesis work is motivated by the fact that additional research is needed to make large-scale Pervasive Sensing systems viable in real-world scenarios.

Our thesis fills this void by highlighting through an in-depth analysis of the challenges of integrated large scale Pervasive Sensing systems composed of heterogeneous devices connected by next generation networks, and by proposing novel architectural models and design guidelines for their development and deployment. One of the main claims is that Pervasive Sensing systems must have a layered architecture with cross-layer visibility to make it as easy as possible to calibrate communications between different devices. In addition to this major claim, the contributions of this thesis can be grouped in the following main areas:

- *The introduction of a new unifying logical model for large-scale Pervasive Sensing systems*. The thesis proposes a high-level logical architecture, detailing its modules, issues and design trade-offs. We accurately analyze design choices, discuss their impact and compare them to current state of the art implementations, by highlighting how network and devices dynamic behaviour can be tolerated and even exploited via opportunistic communication protocols.
- *The design and the implementation of different Pervasive Sensing systems, targeted for five different significant case studies*. Considering the different levels of abstraction of Pervasive Sensing systems, we apply our logical model to five different case studies, that can be classified in three main research direction: the first direction is data collection and sensing devices management, the second one is data processing to provide services to individuals, the third and last research direction is scaling sensed data-based to social communities.
- *Thorough evaluation of Pervasive Sensing systems*. To assess the validity of technical contributions of this thesis we implemented the facilities described by our logical model. In particular, we used mathematical models, simulation approaches, and real-world testbeds to collect extensive performance statistics about several performance indicators, such as CPU load, memory overhead, sensing task completion rate. The joint usage of simulators and real world deployment allows to evaluate network performances for large-scale scenarios while considering real-world resource loads.

- *Performance evaluation of the performances for all considered case studies*. Exploiting both simulator-based and real-world implementations we thoroughly evaluate the performances of the presented case studies, to measure the effectiveness and the efficiency of proposed protocols and techniques.

The thesis is organized in six chapters, that analyze Pervasive Sensing systems starting from low-level data collection up to social-aware sensing systems. Chapter 2 gives background information about current state of the art of sensing systems and how they interact with network architectures expected to be commonplace in the next future. Chapter 3 describes the requirements of Pervasive Sensing systems, analyzes the limitations of current proposals, presents sound design principles, and derives from them our novel logical model, describing its high-level tiered architecture, network deployment and functional components; in addition, it compares the proposed model to existing solutions, to better highlight the need of research in this area. To assess the validity of our logical model, we present five different case studies, all relying on our proposal, that explore different Pervasive Sensing systems, starting from systems focused on data collection, then moving on to systems that exploit sensed data to provide personalized services, and finally to systems that provide services to social communities.

Chapters 4 and 5 prove that the device heterogeneity that makes Pervasive Sensing systems hard to design can actually enable large-scale systems with better performance guarantees compared to existing solutions. In particular, they analyze the design and implementation of two systems for data collection: the former provides efficient data routing in Wireless Sensor Network (WSN) by opportunistically exploiting mobile devices, the latter stems from the logical model a systems that manages sparsely connected WSN by leveraging mobile devices as opportunistic controllers of WSN nodes. Then, we shift our focus on systems that exploit collected sensed data to provide personalized services. In particular, Chapter 6 derives from the proposed logical model the architecture of an application that opportunistically measures physical events on smartphones to infer information about its owner wellness, thus providing a personalized service based on pervasive opportunistic sensing. Scaling up services based on pervasive sensing, Chapter 7 presents two Pervasive Sensing solutions, called Social-Aware IMS-enabled Recommender (SAIR) and McSense, that use techniques described in previous chapters and show two different integrations with social groups: SAIR is a service based on sensing proof of concept that implements collaborative filtering based on distributed sensing, whereas McSense integrates with human groups by asking them to be sensors themselves, thus providing sensing data that would be very hard to gather without willing participation of people in Pervasive Sensing systems.

Finally, Chapter 8 ends this dissertation by highlight main thesis contributions and by detailing still open challenges and future research directions.

2 PERVASIVE SENSING IN FUTURE NETWORKS

MARK D. Weiser postulated long ago that *"the most profound technologies are those that disappear"* [1]: his vision was establishing the blooming of *Ubiquitous Computing*, the idea that computing devices will get so powerful, compact, energy-efficient, and self-managed that eventually they will stop needing active attention and will vanish in the background while still providing their services, transmitting information ready to use. The research by Weiser at the Xerox Palo Alto Research Center lead the development of several experimental portable devices, similar to modern tablets and interactive whiteboards that were deeply integrated with the supporting infrastructure of the research center itself (Figure 2.1).

The system architecture pioneered in those early days of research on Ubiquitous Computing is very close to the ideas of present day tablets and smartphones whose data is backed up on the cloud. However, that vision is strongly centered on considering Ubiquitous Computing for large systems that receive input from user and route it wherever and whenever needed, but have little or no direct knowledge of the real world environment: they can only manage data that is already available. A big step forward for the usefulness of Ubiquitous Computing systems is the possibility to wire them sensors, namely to devices that are capable of measuring stimuli and transduce them in a machine-readable format. Adding sensors to a computing system unlocks new information layers that were previously impossible to manage, enabling a deep integration between the digital world and the real world by producing a *Pervasive Sensing* scenario.

Several important research topics are being promoted by the growing trend of Pervasive Sensing and Ubiquitous Computing: the former has pushed the evolution of new sensors and their integration in electronic devices, vehicles and buildings; the latter has stimulated the development of novel network architectures to dynamically integrate new mobile devices with commodity, wired networks. This chapter analyzes the recent results of these research topics and makes the case for the importance of their integration to provide pervasive sensing in future networks.

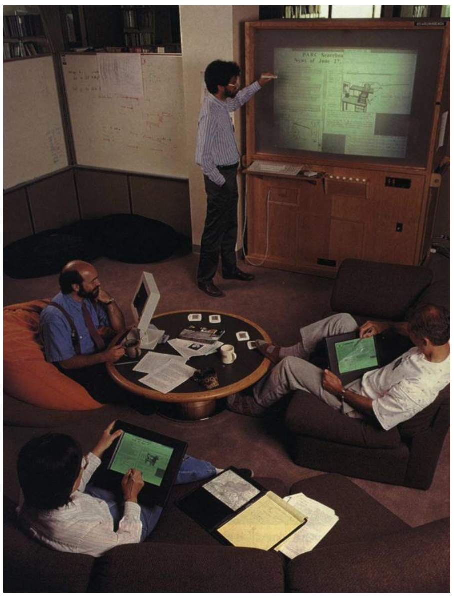


Figure 2.1: Experimental electronic "chalkboards" and "tablets" at the Xerox Palo Alto Research Center in 1991. Courtesy of [1].

2.1 SENSING DATA

The capability of gathering data via sensors is the core feature that sets apart Pervasive Sensing systems from traditional networked systems. Thus, it is important to define exactly what we consider a sensor and what are sensed data. Generally speaking, a sensor is any device for the detection or measurement of a physical property to which it responds. This definition is correct, but it is also too broad to be useful in the context of Pervasive Sensing systems for two reasons: first, it does not capture the key aspect of sensors as electronic devices connected to a system, second, it excludes all the sensors that do not directly access the physical world. For example, the fact that two people have a social relationship is not physically-sensed, nonetheless it is real and can be concretely identified and sensed by networked systems (e.g., processing social networks data).

For these reasons, here we define sensors as *any physical or logical data source that provides information about events in the real world in a machine readable format*. This definition captures the fact that sensors are gateways that reflect events in the real (not necessarily physical) world to the digital world of Pervasive Sensing systems [2].

The distinction between physical and logical sensors directly mirrors a fundamental difference between collected data. Physical sensors detect events of the physical world (e.g., temperature, noise level), that we call *physical data*. Logical sensors collect data about events that can not or are very hard to be physically be measured, but nonetheless are real (e.g., a person stress level, a relationship between friends). These kind of events are not tied to an unambiguous physical mea-

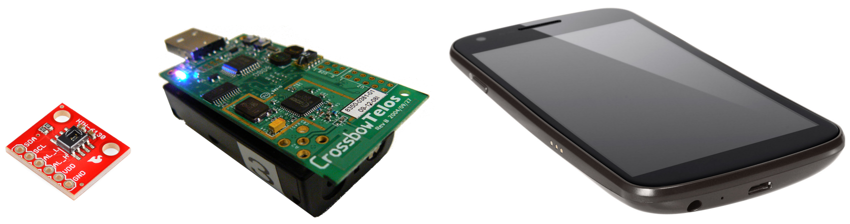


Figure 2.2: Examples of physical sensors: a humidity sensor (courtesy of Sparkfun.com), a TelosB sensor node, and a Galaxy Nexus Android Smartphone.

surement, rather, they are part of human emotions and social relationships, hence we call them *social data* and, for the same reason, we call logical sensors also *social sensors*.

Even though the goal of sensors is to remove humans from the loop and enable automatic data collection, nonetheless users can voluntarily input physical and social data. Of course, humans are neither hardware physical sensor nor software logical sensor, hence, we regard human as a special category of sensors. In the following subsections we describe the most important and recent works about physical sensors, social sensors and humans as sensors, and their applications.

2.1.1 Physical Sensors

Physical data are collected by any device or *physical sensor* that can react to physical stimulation and translating them in a machine readable format. Physical sensors can be classified in three categories: *sensors*[1], *sensor nodes*, namely autonomous sensors provided with computing and communication capabilities, and *mobile sensor nodes*, namely sensor nodes that can either move autonomously or are carried by vehicles and people (Figure 2.2).

Sensors are chronologically the first devices that allowed collection. Sensors are composed by a transducer and a signal processor. The transducer is a device that converts variations of a physical quantity (e.g., light, pressure) into variations in another, which often is a voltage or current change. The signal processor receives the output of the transceiver and processes it, cleaning it up, by possibly running other refining algorithms that gather specific information from the raw sensor data; the resulting data is then sent to the transceiver that transmits it to the device that uses the sensor. The most important change brought by integrating sensors in a computer system is the possibility of taking humans out of the loop in all those cases where humans

1 We use the term *sensor* to refer to sensors as the actual hardware devices for this subsection only, throughout the rest of this work we will instead use the generic definition given in Section 2.1.

had to measure properties and then manually input them into an application. Being able to perceive some physical properties, process them, and act accordingly allows computing systems to reduce their reliance on users, coming closer to the the vision of Pervasive Sensing systems. It is important to stress that, depending on the application domains, sensors sometimes can not substitute humans. For example, sensors are very accurate when measuring physical properties such as temperature and length, but they are much more error prone with some tasks that humans can do effortlessly, such as recognizing complex-shaped objects and classifying emotions.

Sensors are a basic building block of Pervasive Sensing systems, however they present several limitations: they have no processing power (except for the signal processor), no storage resources, they must be configured one by one, and they must be connected to a computing system to be useful. These drawbacks are especially severe when sensing has to be run on a very large area, or has to be very detailed, thus involving hundreds or thousands of nodes. Academic research has tackled this problem with the development of *sensor nodes*. Sensor nodes are low cost, small, autonomous devices, that enhance sensors adding new features such as computing power, memory, persistent storage, and wireless communication capabilities [3]. Sensor nodes can be sparsely deployed, thus requiring an external node to harvest their data; in alternative, through a cooperative effort and exploiting ad hoc networking protocols, they can form a Wireless Sensor Network (WSN), a decentralized multi-hop wireless network that uses low-power wireless links and can be deployed very close to the monitored phenomenon to collect, process, and route sensed data. WSNs require no (or very limited) network engineering and predetermined routing and can work on almost randomly deployed sensor nodes. The greater flexibility of sensor nodes compared to sensors makes them suitable for a very large spectrum of applications, such as environmental (e.g., fire and flood detection, pollution analysis, monitoring of animal behavior), health (e.g., physical activity monitoring, heartbeat monitoring), military (e.g., surveillance, targeting, damage assessment, attack detection), domestic (e.g., home automation, smart environments), and civil (e.g., heating, ventilation, and air conditioning management, structure flexure detection, traffic management) applications [4]. Data collected by WSNs can be used directly within the network itself [5], by routing it to actuators, but it can be also routed to specific gateway sensor nodes that connect WSNs to a network or to the Internet, thus making collected data available to other computing systems [6].

Sensor nodes are key technology for the realization of Pervasive Sensing systems, but there is still room for improvements. In particular, WSNs take time to be deployed in new areas, which can be a lengthy process, and they are limited to monitor only the area they have access to: any event that is even slightly out of their monitoring range has no chance of being detected. This limitation can be removed and sensing coverage can be improved by *mobile sensor nodes* that are sensor nodes that can move either actively or because they are passively carried by vehicles and people [7]. Examples of mobile sensor nodes with autonomous movement capabilities are the so called *data MULEs*, namely controlled nodes that harvest data from WSNs [8], and robots, that are often used only in emergency scenarios and not only for data collection [9]. A very relevant example of mobile sensor nodes that are passively carried are current generation smartphones. In fact, any off the shelf smartphone is provided with several sensors, such as accelerometer, magnetometer, gyroscope, Global Positioning System (GPS), proximity sensor, microphone, and camera. Smartphones have been provided with these sensors for straightforward tasks, such as rotating their screen view and rotate maps of navigation applications; however, recent research has shown that signal processing and machine learning techniques can infer very high-level data from them. Those data can be person-centric or urban-centric: examples of person-centring smartphone sensing tasks are the estimation of quantity and quality of physical activity using the accelerometer [10], detection of social situations such as being in a cafeteria, on the road or listening to music using the microphone [11], and estimation of user heartbeat and lung function via smartphone camera [12, 13]; examples of urban-centric smartphone sensing tasks are noise pollution mapping via microphone [14], road condition monitoring via accelerometer [15], automatic tagging of indoor environments via camera [16]. This opportunistic explotation of smartphones has been called in literature *opportunistic crowdesensing*, hinting at *crowdsourcing*, namely the process that outsources data elaboration tasks to human operators [17–19].

Sensors, sensor nodes, and mobile sensor nodes are not a replacement of one another, altought their features partially overlap, because are mostly complementary, as summarized in Table 2.1. Consequently, Pervasive Sensing systems should exploit and integrate these different physical sensors, to get a data collection coverage as large as possible.

2.1.2 Social Sensors

Social data is harvested by any sensor, that we called *social sensor*, that can collect socially-relevant data, either inferring it by processing the output of physical sensors or by extracting metadata from data manually input by users [20, 21]. A prominent example of social data derived from physical data stems from processing information provided by smartphones. For example, user mobility patterns can be processed to automatically identify places of interests (e.g., a con-

Table 2.1: Summary of features of sensors, sensor nodes, and mobile sensor nodes.

	Sensor	Sensor node	Mobile sensor node
Autonomous	✗	✓	✓
Direct connection to the Internet	✗	✗	✓
Low power	✓	✓	✗
Variable sensing area	✗	✗	✓
Continuous sensing of an area	✓	✓	✗

cert hall, a gym), which are likely to be shared with people that have similar interests [22], user proximity collected via Bluetooth scanning can help identifying user cliques [23, 24]; audio processing can collect social data, for example how often and how long the user talks [25], if she talks to groups of people [26], and if she is under stress or presents other relevant emotions [27, 28].

Smartphones are also an entry point for real social interactions, such as phone calls, text messages, emails and interactions via social networks. Logical sensors can tap into these data streams, making smartphones a privileged platform to gather social information about users [29]. Traditional phone interactions, namely phone calls and text messages, have been used for different goals, such as to automatically classify contacts as family members, friends or colleagues and to provide a smart phonebook that predicts which contact the user is going to use [30, 31]. More recently, the widespread usage of social networks readily available on mobile devices has unlocked a potentially very large dataset to harvest social information about users. Some data is immediately accessible and needs little or no processing, for example friendship relationships (Facebook), professional relationships (LinkedIn), topics of interest at any given time (Twitter hashtags and Pinterest), musical preferences (Last.fm) and even health conditions (PatientsLikeMe) [32]. By further processing data collected from social networks allows inferencing higher level data, from estimating personality traits [33] to aiding mental health assessment [34], from very accurate prediction of user mobility based on friendship [35], to optimization of data delivery and content distribution networks based on social relationships [36].

2.1.3 Humans as Sensors

Even though physical and social sensors can collect large amounts of data, human can still contribute to data harvesting and act as sensors, a process called *crowdsensing* or *participatory crowdsensing*. People, by using personal devices such as smartphones, can take pictures, audio recording or directly input data that can be fed to Pervasive Sensing systems [17, 19, 37]. Let us stress that we call crowdsensing the sensing processes based on voluntary actions of users that actively input sensing data, not to data that is just collected by devices carried by users. The data provided by users can be a supplement or a complement to automatically collected data. Supplementary data may also be picked up by physical or social sensors; for example, traffic congestion could be detected by aggregating and processing users position and speed via GPS; however the same data can also be very quickly provided by a user that simply signals the problem by using an appropriate software. Complementary data is very hard or impossible to collect for automated sensors; for example signaling that air is polluted in an area that is not monitored by environmental sensors.

Crowdsensing applications can be classified in two categories depending on their main goals: *environment-centric* and *human-centric*. *Environment-centric applications* leverage humans as sensors by asking them to measure, signal, or take pictures of environmental phenomena, such as noise and air pollution, water levels, and natural hazards [38, 39]. *Human-centric applications* encourage people to sense their surroundings to generate data to share amongst themselves, for example to compare habits and interests [19, 40]. Due to its human factor, crowdsensing can not generate the same volume of data that physical and social sensor can, unless it is distributed on a very large population; nonetheless, it is a product of human intelligence and compensates its relative scarcity with its high quality level: a single human inference can save hundreds of measurements on simpler sensors; thus crowdsensing is an important data source for Pervasive Sensing systems.

2.2 PERVASIVE SENSING IN FUTURE NETWORKS

The overview about sensing techniques of the previous section shows that current techniques are capable of collecting and processing large amounts of data that can be used in several different domains, that span from urban monitoring to healthcare, from civil engineering to social network data mining. Despite the many fields that can benefit from Pervasive Sensing, academic and industrial research efforts have yet to make the breakthrough that makes it available to the large public. According to the recent NSF Workshop on Pervasive Computing at Scale [41], this is in part due to the fragmentation of research efforts in very specific niches, which is caused by the lack of largely accepted reference testbeds, datasets, models, and software, that would ease the development of new sensing systems and allow to fairly compare them. However, a holistic approach that would solve the fragmen-

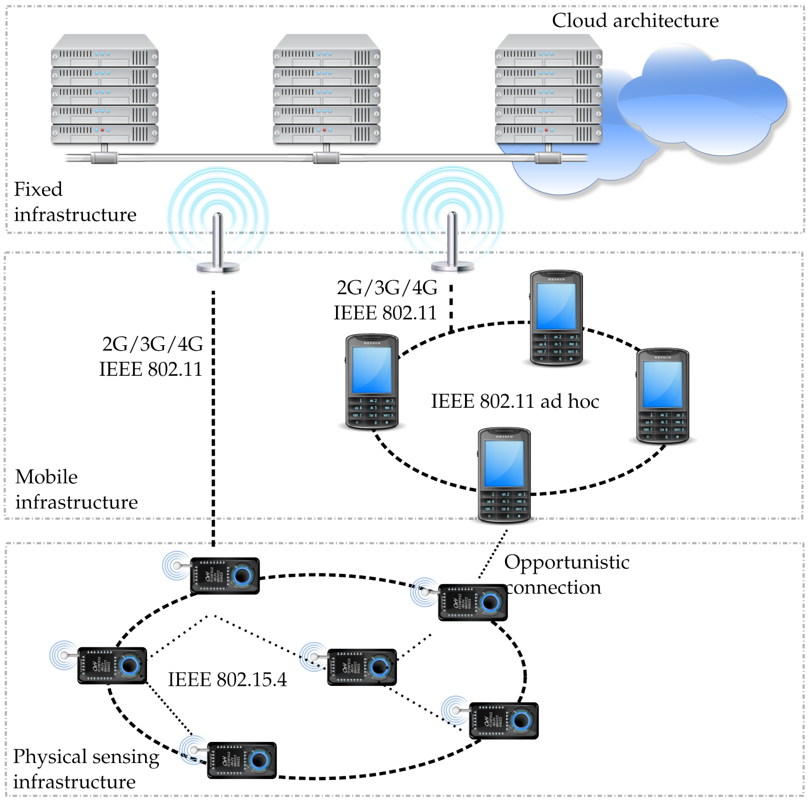


Figure 2.3: Pervasive Sensing system in a Future Network.

tation problem has to deal with the technical difficulties of designing a system that manages very different devices, from sensor nodes for physical sensing, to smartphones for physical and social sensing, to high end servers for data processing. In addition to their computational differences, these systems also use different network types to communicate, that can further complicates coordination. Managing this complex landscape requires a deep knowledge of technological details, with many complex cross-layer consequences. For example, the usage of different physical networking protocols constraints the communication windows between different devices, the computational resources available on them will require specific processing to be demanded to other, more suitable devices, and so on. Thus, it is important to describe our main, complete, Pervasive Sensing system deployment scenario and how it conforms to Future Networks that are likely to be more and more common in the next decade.

Pushed by the significant interest by academia and industry, the sensing technologies described in the previous section are expected to become widespread. In particular, WSNs market has steadily grown for the last few years, and with no slow down: thus WSNs will be a common technology [42–45]. At the same time, smartphone market share is constantly increasing and is expected to take over the global mobile phone market [46, 47]; to avoid the congestion of mobile telecommunication infrastructures, smartphones will rely, when possible, on peer-to-peer network architectures called Mobile Ad-hoc NETworks (MANETs). In addition, smartphones that are already equipped with multiple wireless interfaces (IEEE 802.11, Bluetooth and cellular 2G/3G), are expected to host on board also low-power wireless interfaces that will enable direct communication between them and sensor nodes, unlocking a potential deep integration of MANETs and WSNs [48–50]. Fixed infrastructure, too, has gone through deep changes: traditional, consolidated backends have been enhanced along the *cloud computing* direction that exploits hardware virtualization to provide computing resources on-demand. In particular, solutions such Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) allow programmers to develop applications based on software services, frameworks, and infrastructures that can be easily tuned to scale up and to provide the desired level of reliability. The cloud eases the management of large amounts of data and are capable of quickly reacting to sudden request spikes, hence we can expect new devices to integrate with both existing infrastructures and with Cloud architectures that provide large-scale data processing and storing capabilities [51].

Figure 2.3 reports an example of Pervasive Sensing system in a Future Network, and highlights its inherently three-tiered architecture based on fixed infrastructure at the top level, that comprises servers and Cloud services that are linked to the Internet via commodity wired connections, on mobile infrastructure at the mid level, namely by smartphones and other mobile sensor nodes, and on physical sensing infrastructure at the lower level, namely WSNs and sensor nodes. Let us stress that none of the three tiers in Figure 2.3 is mandatory: a Pervasive Sensing system can comprise any pair of the three tiers and still be useful, as we will show with the case studies presented in the next chapters.

Future developments in electronic and telecommunications will certainly deeply modify the implementations of these three tiers. Researches in energy scavenging and novel radio protocols will make sensor nodes truly ubiquitous, enabling every object to have a network and computing resources, thus realizing the vision of Internet of Things (IoT) that predicts that all objects will be eventually connected to the Internet. Smartphones, that currently are the most important mobile device, may completely change their appearance (e.g., the Google Project Glass project is currently experimenting with substituting smartphones with augmented reality head-mounted displays) and be integrated with other technologies that provide healthcare and wellbeing data, such as Body Area Networks (BANs). Fixed infrastructure will become more distributed to be more quickly available to users, for example research about "cloudlets" is currently investigating the possibilities of offering on-demand virtualization-based computing resources as support for resource-rich mobile computing [52]. The research trends of the technologies in the three tiers promote a sharp increase of the number of networked nodes. Managing them will require to encourage, whenever possible, local interactions between nodes because they do not require support infrastructure to scale as the number of nodes grows. Despite the possible profound changes that the tiers will go through, they will still provide the functionalities the we can reasonably expect from Pervasive Sensing systems:

- smart environments and smart objects, realized via sensor nodes;
- automatic inference of user context, realized via artificial intelligence algorithms on mobile devices;
- background processing and ubiquitous availability of personal data, realized via fixed infrastructure.

We claim the relevance of the scenario described in this chapter as long term reference because it represents a system capable of providing the functionalities required by Pervasive Sensing systems and highlights the need for localized interactions.

The reference scenario of Future Networks features extremely heterogeneous devices, that differ in terms of computational resources, spanning from the very limited sensor nodes, to the powerful but battery-bound smartphones, to high end servers, and in terms of network resources, from slow, low-powered WSN, to the many telecommunication protocols available to smartphones, to the low-latency high-bandwidth wired connection available on the fixed infrastructure. In addition, the scenario is highly dynamic; for example mobile devices can change their position, thus changing the neighbourhood of other devices they can directly connect to, and sensor nodes deployed in difficult condition or for a long time can run out of battery causing changes in the topology of WSN. The heterogeneity of devices and the dynamic conditions force Pervasive Sensing systems to support heterogeneous networking and encourage opportunistic behavior.

2.2.1 Heterogeneous Networking

The reference scenario in Figure 2.3 includes four different types of networks: commodity wired networks for the fixed infrastructure, data-oriented mobile telecommunications protocols such as General Packet Radio Service (GPRS), Enhanced Data Rates for GSM Evolution (EDGE), and 3G connect smartphones and sensor nodes to the fixed infrastructure [53, 54], finally, ad-hoc networks interconnect smartphones and sensor nodes, thus creating the so called MANETs and WSNs. While wired connectivity and GPRS/EDGE/3G protocols are commodity networking solutions, MANET and WSN are still very active research areas. Due to their relevance for Pervasive Sensing in Future Networks, this section sketches their main features.

Mobile Ad-hoc NETworks

A MANET is an IP-based impromptu peer-to-peer network of co-located wireless mobile nodes, such as smartphones, laptops, and Personal Digital Assistants (PDAs)s, which cooperate to route network traffic to compensate for the lack of a fixed network infrastructure (e.g., access points, routers, switches) [55]. MANETs are versatile tool with several application areas, such as network extension (i.e., extending network access to areas not covered by fixed infrastructure), local interconnection networks (i.e., spontaneously forming a network to interconnect devices, for example in emergency response scenarios), ubiquitous computing, and vehicular networking [56]. However, to be a viable technology, MANET have still to solve many challenges. Some of these stem from the fact that all the communication is transmitted via wireless channel, which has several well known problems:

- no clear propagation boundaries;
- unprotected from external signals and noises;
- less reliable than wired links;
- asymmetric;
- subject to hidden terminal and exposed terminal.

In addition to these problems, which are common to all wireless networks, MANETs have a number of additional constraints and complexities [55, 57]:

- *Autonomous and infrastructure-less*. MANETs do not require any fixed infrastructure, all supporting services are run in a completely distributed fashion. This approach greatly increases the difficulties of management of nodes and fault detection.
- *Multi-hop routing*. The infrastructure-less approach means that there are no routers that manage packet dispatching, hence a subset or every node of a MANET must dedicate part of its resources to route data packets on behalf of other nodes.
- *Dynamic topology and link capabilities*. Nodes of a MANET have usually little or no movement constraints, which causes the network topology to change continuously, and the characteristics of established links, such as symmetry and packet loss ratio, vary with time.
- *Network scalability*. Theoretical studies show that the bandwidth between two arbitrary nodes in a MANET does not scale as the MANET size increases [58], thus techniques as clustering must be employed, further increasing management complexity.

The large pool of applications of MANETs has promoted many research efforts to overcome these limitations, mostly by focusing on routing protocols, that have a pivotal role to overcome all the complexity previously listed [56, 59]. We can identify four main routing protocol categories by aggregating them based on similar features, shortcomings and suitable scenarios: *reactive*, *pro-active*, *hybrid*, and *geographical*. *Reactive protocols* evaluate routing paths on-demand, when a node tries to send a packet to another one. The reactive approach is a good choice for dynamic networks and has a low overhead, but it has high latencies and leads to frailer routes compared to other approaches [60–62]. *Pro-active protocols* maintain always updated routes in distributed tables, even for routes that have never been used. The protocols of this category have low latencies, but are not suitable to highly dynamic networks and have a high overhead [63–65]. *Hybrid protocols* try to balance the features of reactive and pro-active protocols: each node pro-actively maintains up-to-date routing tables of its neighbor nodes, to which it has stronger links, while routes to farther nodes are evaluated reactively. Hybrid protocols are more suitable to large networks, while for smaller networks is usually possible to use simpler reactive or pro-active protocols [66–68]. Finally, *geographical protocols* exploit the knowledge of node position to route data following a path that is geographically short. These protocols have generally good performances, but they require location information, that can often be unreliable or not available (e.g.: indoor) [69, 70]. All these protocols manage MANET nodes as a flat collection, but it can be shown that as the network size increases the total throughput increases, but the end-to-end throughput approaches 0, making very hard to maintain robust, large MANETs [58]. This additional problem can be mitigated by organizing hierarchically MANET nodes in clusters managed by clusterhead nodes. In fact, many existing flat routing algorithms have been modified to support clustering, allowing routing to scale well to large networks. However, this approach strains clusterhead nodes, because they bear the additional load of managing nodes in their cluster, causing a quicker depletion of their battery; hence, it is advisable to keep MANETs small whenever possible, and resort to clustering only when MANET size is too large for flat routing protocols [56, 71].

Wireless Sensor Networks

WSNs, whose application areas have been already described in Section 2.1.1, share some similarities with MANET, especially the limitations of the wireless medium. However, they also have remarkable differences [72]:

- *Network scale*. Some WSN applications require hundreds or thousands of sensor nodes, deployed much more densely than the nodes of a MANET, causing a potential overcrowding of wireless channels.

- *Failures*. Sensor nodes are much more prone to failures than MANET nodes, due to the lower quality of their components, which is needed to keep the cost per node low, and to harshness of the environment in which they are deployed.
- *Computing resource limitation*. The low cost constraint on sensor nodes, which is necessary to make sensor nodes disposable, limits severely the computational, storage, and memory resources of sensor nodes. A typical sensor node has a 5-10MHz microcontroller, 8-128KB for storage, and 1-10KB for RAM [73]. These critical limitations constrain the complexity of algorithms and protocols that can run on sensor nodes.
- *Energy constraints*. Sensor nodes are battery-powered and, due to the potentially very large number of sensor nodes in a network, battery replacement could be very hard or infeasible.

All these constraints make WSNs less adaptable than MANETs, because the specific usage scenario has consequences on all layers of software and network stacks, making it very hard to develop generic protocols suitable for all use cases. Hence, the goal of a WSN must be defined before its deployment, to configure it to use the most suitable protocol and avoid costly upgrades after the deployment. Usage scenarios of WSNs fall into one of the following tasks: *collection*, *dissemination*, and *point-to-point* communication; each of these communication tasks requires specialized networking protocols [74]. *Collection* is the most important use case, needed by all the monitoring applications based on WSNs: it is the collection of data sensed by nodes, that is processed and dispatched, according to an *anycast* policy, to any gateway node in the WSN, that then forwards it to the Internet [75, 76]. *Dissemination* is the opposite use case: data from the Internet reaches the gateway sensor nodes, which then is sent to all sensor nodes, according to a *broadcast* policy [77–80]. An example of dissemination is the propagation of firmware images to update the software running on sensor nodes. It is important to note that protocol for collection and dissemination are usually not IP-based, thus sensor nodes forming a WSN are not directly addressable neither by other sensor nodes nor by clients on external networks. Finally, *point-to-point communication* allows direct communication of sensor nodes in a *unicast* fashion. An example of networks that can benefit from point-to-point communication are the so called Wireless Sensor and Actor Network (WSAN), namely WSNs which are connected to actors that can be directly controlled by sensors nodes [81]. Contrarily to protocols for collection and dissemination, specialized protocols for peer-to-peer communication often are IP-based, thus not only sensor nodes can directly address other sensor nodes, but they can also exchange packets directly with other commodity clients connected to the Internet [74, 82].

In the following, we report the state of the art about WSN routing algorithms, summarized in Table 2.2. For collection, the most impor-

Table 2.2: Summary of communication protocols for WSNs.
C: Collection - D: Dissemination - P: Point-to-Point.

Protocol	Scenario	Basic mechanism	Support for bulk data	Underlying data link
BCP	C	Backpressure	✗	Any
CTP	C	Gradient	✗	Any
Deluge	D	Versioned timer	✓	Any
Drip	D	Versioned timer	✗	Any
RPL	C+D+P	Versioned graph	✗	IEEE 802.15.4
ZigBee	C+P	Routing tree	✗	IEEE 802.15.4

tant protocols are Collection Tree Protocol (CTP) and Backpressure Collection Protocol (BCP). CTP is the *de facto* standard reference for data collection protocols, because it provided by default as part of the widely used TinyOS operating system for sensor nodes [83]. It is based on periodic broadcasts that become exponentially slower and carry the network metrics necessary to build a routing tree routed on any gateway node; in case a node fails or a new node is added, the timers are reset, allowing the routing tree to adapt quickly to the new topology [75]. BCP is a collection protocol that for each packet computes a backpressure weight that is a function of the local packet queue and state of links to other nodes, thus establishing a gradient for routing without explicitly computing routing paths [76]. Both CTP and BCP are packet based, and support the collection of small packets only.

About dissemination, the most notable examples are Deluge and Drip. Deluge is a a network programming protocol that supports routing of bulk data. It is designed to broadcast firmware images in a WSN to update the code running on sensor nodes [84]. Drip supports reliable dissemination of small values, for example a new configuration setting, in a WSN [77]. Both Deluge and Drip are based on Trickle [85], a communication protocol based on periodic broadcasts that get exponentially slower as time goes on. The payload of each broadcast is the actual data to disseminate and a versioning number, which allows sensor nodes to know if they need to receive the disseminated value and re-broadcast it.

Point-to-point communication is technically the most difficult to realize, because it requires bi-directional routing path between any arbitrary pair of nodes. Notwithstanding the large number of works in this area, the most important proposal are the aforementioned ZigBee

[86] and Routing Protocol for Low Power and Lossy Networks (RPL) [87]. ZigBee manages routes point-to-point routes as an overlay over its basic single-rooted tree topology: whenever a sensor node receives a packet that its not addressed to itself or to one of its children, it starts a route discovery, based on the well known Ad Hoc Distance Vector (AODV) protocol [61]; if it has not enough resources to drive a route discovery, then it routes the packet along the tree [86]. RPL is a IPv6-based multi-hop routing protocol that could be, in principle, be applied to any physical protocol; however, in practice, it is currently considered mostly as routing layer for ZigBee (as part of the ZigBee IP effort). RPL organizes its nodes in Directed Acyclic Graphs (DAGs), which are partitioned in smaller DAGs called Destination Oriented Directed Acyclic Graphs (DODAGs), whose root is a destination node. This approach allows efficient collection of data and also supports dissemination of very small data. In addition, RPL supports point-to-point communications, because it is designed to be a full-fledged IPv6 stack. This flexibility comes at the cost of it being hard to use on very constrained devices with limited memory (less than 10KB), but it has the advantage of being able to integrate seamlessly WSN to the Internet [87].

MANETs and WSNs are key components and technological enablers of Future Networks and Pervasive Sensing. The overview of the issues and active researches sketched in this section shows that that there is no single routing protocol or network architecture that works for all MANET and WSN scenarios; rather, there are several protocols that provide good performances when used in the specific setting they were designed for.

Network Security

The security, confidentiality, and robustness of MANETs and WSNs are very important for a widespread adoption because they store and route critical environmental and personal data. The highly dynamic scenario of MANETs and the limited resources available to WSNs make network security an especially hard challenge. Due to its complexity and specialized scope, this thesis does not directly address network security, however we list here the most significant works that could be applied to the case studies that we will present throughout this work.

Security on MANETs focuses on three topics [88]: secure routing, authentication, access control. Most research efforts have studied the issues related to *secure routing*, exploring different solutions based on intensive usage of cryptographic primitives to validate MANET routing paths and to detect and isolate malicious nodes that try to disrupt routing. The seminal work by Marti *et al.* [89] extended the Dynamic Source Routing (DSR) protocol [62] to introduce special nodes called *watchdogs* and *pathrater* that monitor MANET nodes behavior and isolate misbehaving nodes. Similarly, Wang *et al.* describe in [90] a mechanism based on local observation of AODV [61] to identify selfish nodes that do not cooperate to the MANET routing. Soltanali *et al.* [91] combined the features of the two previous works by proposing a completely distributed mechanism that identifies selfish nodes using a reputation-based control and encourages cooperation via a currency-based scheme.

Authentication in MANETs tries to enforce control mechanisms that allow only trusted nodes to join the network. Davis [92] proposed a hierarchical model for trust management in MANETs based on public keys of all nodes being distributed to all other nodes. This approach makes the proposed scheme robust to malicious accusations, but poses scalability problems. Ngai and Lyu described in [93] a distributed authentication scheme based on cluster-local public key exchanges, that alleviates some the scalability issues of other solutions.

Access control refers to the techniques that enforce policies to control whether or not a node should have access to certain resources. Luo *et al.* describe in [94] a completely distributed access control framework that allows only well-behaving nodes to access MANET resources. This solution relies on threshold cryptography that is a technique that allows to decrypt an encrypted message if and only if a given number of parties exceeding a threshold cooperates in the decryption protocol. [94] exploits this system to globally trust a node if it is trusted by other k nodes, where k is a system-wide threshold.

WSNs have the same security challenges, but the techniques developed for MANETs can not be directly applied to WSNs because they rely on asymmetric cryptography that is hard to use on WSNs due to resource constraints. For this reason, equivalent solutions for WSNs exploit lightweight symmetric cryptography. For example secure routing has been tackled by Deng *et al.* [95] that proposed an intrusion-tolerant routing algorithm, that hinges on the idea that a subset of sensor nodes is more powerful than others and can manage the complete network topology, that is secured because every node provides a list of its neighbors with a proof of neighborhood. The work by Lee and Choi [96] improves on the centralized approach of the previous by delegating partial verifications to sensor nodes, that refer to more powerful nodes only when they detect a node misbehaving. An example of authentication algorithm for WSNs is the work by Perrig *et al.* [97] that proposes μTESLA, an authentication algorithm based on symmetric cryptographic primitives that exploit delayed key disclosure to avoid much more CPU- and memory-intensive asymmetric algorithms. In the same work, the authors propose *access control* in the form of data confidentiality that is achieved by exploiting a symmetric cryptographic that prevents eavesdropping from malicious nodes.

This brief review of the state of the art of network security for MANETs and WSNs is by no means intended to be a complete survey, however it shows that the challenges of network security, authentication, and access control are known to be of utmost importance and there is large research community that is actively tackling these challenges, providing solutions that can realistically be adopted in real-world deployments to make future networks secure, reliable, and trustworthy when dealing with sensitive data.

2.2.2 Opportunistic Behaviour

The reference scenario for Pervasive Sensing presented in the previous sections is highly dynamic at two levels: at the sensing level and at the networking level. At the sensing level, Pervasive Sensing systems, connected to commodity sensors and sensor networks, are expected to exploit also mobile sensor nodes, which are likely to be not directly controllable. At the networking level, Future Networks will comprise volatile networks, i.e., MANETs, and unreliable networks, i.e., WSNs. The dynamic and heterogeneous features of the reference scenario make in unsuitable for traditional system designs that are based on the assumption that network and system characteristics are well known at design time and do not change with time. We claim that the best approach to tackle this shape-shifting scenario is adopting novel design principles that assume that devices are unreliable and system conditions are uncertain. To still exploit as much as possible the available sensing and networking resources, we propose to adopt an opportunistic behavior at both levels.

Opportunistic Sensing

As explained in Section 2.1, Pervasive Sensing systems use smartphones and humans as sensors to better exploit all available sensing resources and increases the sensing coverage [7, 98], so to enable a wide range of novel sensing applications, that we already defined in Section 2.1.3 as human- or environment-centric, that rely on opportunistically exploiting resources to achieve their goals [17]. *Human-centric applications* focus on detecting and collecting data about human activities and understanding behavior. They can work by either explicitly asking users for input or by opportunistically accessing smartphone sensors, such as microphone and accelerometer, and run machine learning algorithms to infer current user activities. The metric that drives this category of applications is *time*: sensors should not be accessed when they are used for higher priority tasks and when they are in unfavourable conditions (e.g., sound sensing should be disabled during phone calls and when the phone is in a pocket).

Environment-centric applications strive to collect data about environmental parameters such as noise pollution, road and traffic conditions, and so on. They too use opportunistically smartphones as sensors, waiting for them to roam near the target area during the window of interest. Thus, the opportunistic approach of environment-centric applications is driven by *time* and *space*: sensors must be accessed, like in the previous case, when they are not being used and when they are in favourable conditions; in addition, their location must be close to the area targeted by the application. Moreover, environment-centric applications exploit humans, too, in an opportunistic fashion. In fact, they can assign sensing tasks to many people, asking them to go near the monitored area, or waiting for them to accidentally get there. The coordinating application has the burden of selecting mobile sensor nodes that may successfully execute the sensing task based on its requirements (e.g., the quantity to measure, its location, its validity window), to delegate the task to them and to coordinate its execution.

Opportunistic sensing poses several key challenges, that we present here along with some recent work aimed at overcoming them [17, 99, 100]:

1. filtering noisy data;
2. inferencing context;
3. preserving privacy;
4. detecting fraudulent data;
5. promoting user participation;
6. minimizing energy consumption.

First, opportunistic sensing must deal with noisy, incomplete sensing data, because sensors are not used exclusively. Moreover, when mobility is involved, the measurements can only come from areas where users are present, causing data to be randomly distributed in space and time. A possible solution to this challenge, proposed by Rana *et al.* is to use a technique called compressive sensing to recover missing data and improve the spatial-temporal coverage of sensing [14]. Second, advanced sensing requires raw-sensor data to be processed to infer user context and activities. This is typically achieved by using supervised machine learning algorithm, that learn from labelled data a model to classify unlabelled one [101]. However, it can be exceedingly hard to obtain a significant dataset of labelled data, hence approaches such as [102] use a pooling approach to share models among sensors. Third, opportunistic sensing must strive to preserve user privacy. In fact, if there are no controlling mechanism in place, smartphones can leak private information about users, for example by tracking their position, recording intimate discussions, and photographing private spaces. This possibility violates the reasonable expectations of individual privacy, and could discourage them from participating in sensing activities, thus reducing the impact of sensing campaigns. AnonySense proposes to sidestep this problem by using *k-anonimity*, namely by running sensing tasks over groups of k spatially co-located people and stripping sensed data by any unambiguously identifying feature, ensuring that users cannot be identified within the set of k users [103, 104]. PriSense protects user privacy by hiding raw harvested data and providing only aggregated results about it (e.g., average, median, min/max values) [105]. Fourth, the openness of sensing relies on user contribution, making it vulnerable to collection of erroneous data, either by mistake or by malicious contribution. For example, a user asked to take a picture of a building may mistakenly put his finger on the camera lens, or a leasing agent asked to use the accelerometer to monitor road condition data may alter the measurements to fabricate smoother readings so to promote properties in a particular neighbourhood. Huang *et al.* propose to use a reputation system that considers more trustworthy data coming from users that provided useful data, and quickly decreases their trustworthiness if they provide corrupted data, either by accident or by mischief [106]. Fifth, for large scale collections it is necessary to promote user participation [107]. Several studies have found that monetary incentives work for repetitive, low-effort tasks, but increasing the reward does not increase the quality of effort by people. Recent work suggest that modifying the task and proposing them as a game (a process called *gamification*) can actually improve the quality of people-provided data [108, 109]. Sixth, and final, opportunistic sensing strains smartphone resource, in particular reducing the battery lifetime. In particular, sensors such as GPS and microphone can easily halve battery duration. Rachuri *et al.* show in [110] that it is possible to learn from user behaviors to pro-actively shut down sensors when it is unlikely that they will collect useful data, thus saving energy.

Opportunistic Networking

At the networking level, Pervasive Sensing systems have to deal with the fragile connections of WSNs, due to the low-power lossy wireless protocol that they use, and of MANETs, due to node mobility. Referring to Future Networks presented in Section 2.2, there is an additional weak point, namely communications between mobile smartphones and sensor nodes over weak low-power links. These features make our reference scenario a kind of Delay Tolerant Network (DTN), namely a network where, given an arbitrary pair of nodes, there may not exist a complete routing path that connects them [111]; in this case, the message to delivery should be buffered in the network, until a new routing path emerges [112, 113]. This approach is called *opportunistic networking* and it finds a solution with increased delays in data delivery, but, compared to traditional networking protocols, it allows to exploit links that else would have never been used, and allows to exchange data between nodes that may have never been properly connected.

Opportunistic networking manages wireless devices carried by animals, people, and vehicles, that can form small mobile ad hoc networks when they are close to each other and store data and routing

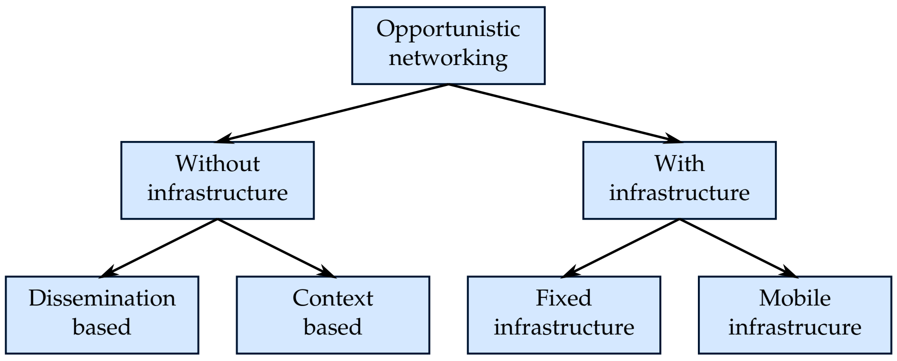


Figure 2.4: Taxonomy of opportunistic networking techniques.

information when they are disconnected. Applications that can benefit from opportunistic networking include mobile sensor networks [114], wild-animal tracking [115], "pocket switched networks" that base routing on pair-wise contacts between personal devices [116], and transportation networks [117].

Several routing algorithms have been developed to overcome the challenges of ephemeral links to realize opportunistic networking [112]. In first instance, they can be classified in algorithms designed for completely flat networks that can not rely on any kind of infrastructure (*without infrastructure*) and in algorithms designed for network that have some kind of tiered architecture (*with infrastructure*). Algorithm *without infrastructure* can be further divided in *dissemination-based* and *context-based*. Dissemination-based algorithm work by flooding the network with copies of the message to be delivered [118, 119]. This approach is very resource-hungry, and can lead to network congestions, typically reduced by limiting the number of copies of a packet that can be stored on the network at any given time. More refined algorithms are based on network coding, a technique that exploits information theory to allow nodes to reconstruct data packets using redundant bits carried by other packets, without receiving the full original packet [120]. Also *Context-based* algorithms view networks as a flat collection of nodes, but they limit the propagation of messages by choosing the next hop for each packet based on the context in which nodes are operating. Compared with dissemination-based algorithms, context-based algorithms greatly reduce the network load, at the cost of a generally higher latency. Algorithms such as Context-aware Adapting Routing (CAR) base routing decision on predictions of the probability that any given node will meet any other one. When a node must deliver a packet, it stores it in a buffer until it meets the destination node or when it meets another node that has a higher probability of having a *rendez-vous* with the destination node [121]. Algorithms based on networks *with infrastructure* do not need to disseminate packets, rather, they try to get packets as close as possible to the access points of the infrastructure, that is usually a less challenged network. The access point to the infrastructure can be either fixed or mobile. In case of *fixed infrastructure*, the nodes usually retain their packets until they are within the communication range of the infrastructure access point, then they rely their packet. This approach, called *Infostation*, causes high latencies [122]. The performances of the Infostation approach can be greatly improved by allowing packet routing between mobile nodes: in this way, a node can either forward a packet directly to the access point, or to another node that has a greater chance than itself to reach the access point [123]. Finally, the infrastructure can be formed by mobile nodes that act as data collectors, called *data MULEs* or *ferries* [8, 124]. These agent move around the network to gather data, and can be the only ones enabled for data collection, or, if node-to-node communication is allowed, they roam to enhance connectivity in areas that have severely limited bandwidth or are completely disconnected. All these approaches to opportunistic networking, summarized in Figure 2.4, focus on specific scenarios; however, there are still no proposal of multi-tier opportunistic networks that exploit mesh networks, mobile networks, fixed and mobile infrastructures. Among the other result, the present work tries to overcome this challenge.

2.3 CHAPTER CONCLUSIONS

This chapter has presented the reference scenario of Pervasive Sensing systems in Future Networks. It has shown that the sensing capabilities derive from the large diffusion of sensors, sensor nodes and mobile sensor nodes (i.e., smartphones), which, in addition to raw sensed data, can also provide high level inferences, and possibly by asking users to input data themselves. From the point of view networking, the reference scenario relies on MANETs and WSNs, which we have described presenting their features and challenges. We claim that, due to the characteristics of the reference scenario of Future Networks, opportunistic sensing and opportunistic networking are key mechanisms for the successful realization of Pervasive Sensing systems. In the next chapter we will describe a logical model that provides a robust structure to design multi-tier Pervasive Sensing systems.

3 LOGICAL MODEL

The overview about sensing technologies and networks presented in the previous chapter has shown the complex and multi-layered scenario of Future Networks in which Pervasive Sensing systems should be integrated. This chapter outlines general design principles for Pervasive Sensing systems and analyzes the main challenges of working on this heterogeneous scenario. Then, it presents a reference logical model for Pervasive Sensing systems and compares it to the current state of the art. Finally, it proposes five case studies based on the proposed logical model.

3.1 DESIGN PRINCIPLES

Before describing the main challenges of Pervasive Systems, let us recall the reference scenario of Future Networks. In Section 2.2 we have defined and described a three-tier scenario. At the lowest tier there is the physical sensing infrastructure, that comprises WSNs and sensor nodes, that communicate using low-power protocols such as IEEE 802.15.4. At the middle tier there is the mobile infrastructure, composed of smartphones and other mobile devices, possibly organized in MANETs, that can connect to nearby sensor nodes. Both sensor nodes and mobile devices can communicate with the top tier, namely the fixed infrastructure, via wired networks, via wireless networks such as 2G/3G/4G cellular protocols or IEEE 802.11. The three tiers can be used together to design a full-fledged Pervasive Sensing system, however let us stress once again that none of them is essential: combinations of just two tiers still allow to design significant working systems with different use cases, as we will show when describing the case studies at the end of this chapter. The analysis of Pervasive Sensing systems and Future Networks highlighted recurring challenges that we used to identify some design principles that underpin the logical model proposed in this chapter.

- *Opportunistic behavior*. The mobile infrastructure and the physical sensing infrastructure have very dynamic network behaviors, as explained in Section 2.2.1. To manage them, the reference model should support opportunistic networking, thus exploiting network resource that would be unusable for traditional networking architectures. In addition, we have presented in Section 2.2.2, the advantages of exploiting opportunistic sensing to collect data that is out of reach of WSNs. For these reasons, the reference logical model should emphasize the opportunistic approach to networking and sensing, that allow to successfully collect data even from challenged networks.

- *Adaptability*. Nodes participating in a Pervasive Sensing system are connected over time to networks that have different resources (e.g., a smartphone that is connected to a MANET composed of other powerful devices and to a low-power WSN), or whose resources change over time (e.g.: available network bandwidth that changes during peak traffic times). Thus, the reference logical model should support adaptability, namely the capability of dynamically tuning resource consumption based on current use conditions.
- *Power conservation*. Energy consumption is not a very relevant issue at the fixed-infrastructure tier, but it is very important at the mobile infrastructure and physical sensing infrastructure tiers, for different reasons: on mobile devices such as smartphones, Pervasive Sensing systems exploit their resource to either collect sensing data or to opportunistically connect to WSNs; both tasks can be battery-intensive, to the point that they could suggest the user not to lend her phone resource because the perceived benefit brought by the sensing system does not compensate the detrimental effects to the user experience. On sensor nodes running at the physical sensing infrastructure, battery lifetime is a very precious resource due to the sheer size of WSNs: replacing exhausted batteries is an arduous, if not impossible, task. The sensing and routing tasks opportunistically run by Pervasive Sensing systems may reduce the battery lifetime; however, this reduction should be minimized to extend as long as possible the working period of WSNs and to not penalize users that voluntarily offer resources for Pervasive Sensing by using techniques such as duty cycling and offloading of tasks to upper tiers should be employed to minimize energy requirements.
- *Resource awareness*. Our reference scenario clearly shows that Future Networks are composed of sub-networks composed by devices with very different hardware and software capabilities. Roughly, computational resources increase two order of magnitudes (or more) from a tier to the upper one; for example, a very common sensor nodes such as the MEMSIC TelosB is powered by a 10 MHz microcontroller, with 10 kB of RAM and 48 kB of flash storage, and can communicate via a IEEE 802.15.4 compliant transceiver, that has a nominal data rate of 250 kbps [73, 125]. Sensor nodes like this are expected to run on two AA batteries for a couple of years. Going up of one tier, a modern smartphone such as the LG Nexus 4 has a quad-core 1.5 GHz

ARM processor, 2 GB of RAM and 16 GB of flash storage, it can connect to 42 Mbps cellular networks and to IEEE 802.11n wireless networks (from 54 Mbps to 600 Mbps), and are expected to run for about 24-48 hours on a fully charged battery. A typical server at the fixed-infrastructure tier, such as these offered by Amazon EC2, has a quad-core 1.2 GHz Xeon Central Processing Unit (CPU), 15 GB of RAM and 2 TB of local storage and is connected to the Internet via a wired 100 Mbps Ethernet link; however, exploiting cloud technologies, all these resources are virtually infinite for many practical purposes. It is apparent that tiers are heavily asymmetric, and each tier should be managed with specialized sub-architectures, aimed at exploiting in the best possible way the available resources, thus the reference model should provide specialized components for devices running at different tiers of the reference scenario, tailored to perform optimally on the available resources. Moreover, special care has to be taken in the design of the communication interfaces between the different tiers: in fact, due to the difference of performance, data can be freely routed from any tier to the upper ones, even in bulk, but the reverse would easily cause network congestion and even overloading of CPUs; hence data should trickle from upper tiers to lower ones.

- *Scalability*. Pervasive Sensing system can scale to different sizes, from an apartment, to a building, to a campus, to a whole city. As the size of the system increases, so does the number of involved devices. Scaling up systems is a classical problem of distributed systems; however, for Pervasive Sensing systems, the challenges of scaling up are made worse by the limited resources available on WSNs and MANETs and by theoretical limits [58]. The key to enable scaling of the system is the *"divide et impera"* approach: breaking down devices in each tier in smaller, more manageable groups, possibly with the help of devices at the upper tier, and exploiting *local interactions* to ease the coordination load on upper tiers.
- *Resiliency*. WSNs and MANETs are inherently unreliable networks, due to mobility, faulty hardware, and use of lossy wireless links. However, a Pervasive Sensing system is expected to have reasonable reliability and tolerate these disruptive events. Hence, Pervasive Sensing systems should implement appropriate mechanisms to provide a resilient service despite these expected failures.
- *Low disruption*. Due to their opportunistic behavior, Pervasive Sensing system are expected to exploit resources on mobile devices, that are not directly interested in the final goal of the sensing (e.g., exploiting a mobile node for routing data). Users can agree to provide these resources as long as they are appro-

priately incentivized and the resource consumption is not detrimental to the user experience of their own devices, otherwise they may not cooperate and participate in sensing tasks, harming the quality and quantity of sensed data. Thus, the software implementation should safeguard the normal workflow on devices that it opportunistically exploits.

- *Modularity*. Pervasive Sensing systems present a very high degree of variability: they can involve only one tier of the reference architecture, or two, or all three; they can be centered on a single person, or on a building, or a whole city, they may require continuous sensing or just a periodic one, it may be necessary to harvest data as soon as possible or it can be safely stored on sensing nodes and harvested only once in a while, and so on. All these possible variations, and many other, impose the strict need for the reference logical model to be highly modular, to make it possible to include only necessary components in the specific Pervasive Sensing system developed atop the logical model.

These design principles summarize the most important features, requirements and challenges of Pervasive Sensing systems. We used them as directive to develop a reference logical model, that is described in the next section.

3.2 SYSTEM LOGICAL MODEL

In this section we present a general logical model for Pervasive Sensing systems based on the overview of sensing and networking technologies outlined in the previous section. This logical model is not intended to be a full featured architecture, rather it is aimed at highlighting the architectural components that overcome the challenges of Pervasive Sensing systems while following the design principles listed in Section 3.1.

Before delving into the details of the components that compose our logical model, let us recall the reference scenario. Pervasive Sensing system can span up to three different interconnected network tiers: the physical sensing infrastructure, the mobile infrastructure, and the fixed infrastructure (Figure 3.1). At a very high level, the physical sensing infrastructure focuses on sensing physical data using sensor nodes, possibly organizing them in WSNs; the mobile infrastructure manages sensing on mobile nodes (i.e., mainly smartphones) and opportunistic connections with sensor nodes; finally, the fixed infrastructure provides coordinates the lower tiers, collects data from them and provides powerful processing and storage services for sensed data. According to the *modularity* principle, none of these tiers is strictly necessary. Devices in each tier have to perform the following tasks:

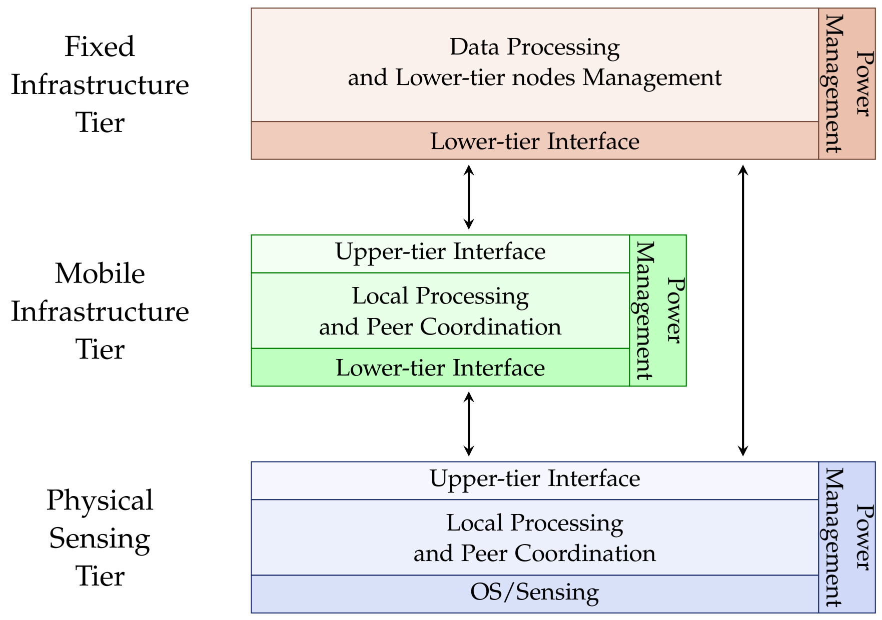

Fixed Infrastructure Tier
Data Processing and Lower-tier nodes Management
Lower-tier Interface
Power Management
Mobile Infrastructure Tier
Upper-tier Interface
Local Processing and Peer Coordination
Lower-tier Interface
Power Management
Physical Sensing Tier
Upper-tier Interface
Local Processing and Peer Coordination
OS/Sensing
Power Management

Figure 3.1: Pervasive Sensing system tiers.

- sense data (if provided with sensor);
- process data;
- communicate with peers in the same tier;
- communicate with devices in different tier;
- manage power consumption (if it is a concern).

The following sections analyze in detail how these functions are organized in logical components running on devices of each tier, and how they enable inter-tier communication for deep integration of heterogeneous networks.

3.2.1 Physical Sensing Infrastructure Tier

The physical sensing infrastructure is composed of sensor nodes, that can be deployed in a very sparse fashion, resulting in disconnected sensor nodes that have to wait for another node to roam nearby to forward their sensed data, or close enough to allow them to communicate with each other, thus forming a WSN. The components running on sensor nodes need to provide the following functionalities: manage sensor node hardware and provide access to sensors, process sensed data, coordinate with other sensor nodes, manage communication with upper tier devices (either smartphones or fixed infrastructure servers), and dynamically manage power consumption. The logical blocks that realize these functionalities are, in the same order, the components *OS/Sensing*, *Data Processing*, *Peer Coordination*, *Upper-tier Interface*, and *Power Management*, that are organized in a layered structure, as shown in Figure 3.2.

OS/sensing

The *OS/Sensing* component is a basic building block of the components running on sensor nodes. It wraps any Operating System (OS) specialized for sensor nodes, such as TinyOS and Contiki [83, 126], that provides core functionalities such has a hardware abstraction layer, a filesystem to store data on the local storage, process management, and primitives for communication via network interfaces. We explicitly wrap in a component the OS because, due to the very limited resource available on sensor nodes, it can not provide its service in a transparent manner and processes running inside its context are strongly coupled with it; hence, we want to stress its relevant role on sensor nodes. Among the hardware abstractions provided by the OS, there are the interfaces to access the sensors available on the node (e.g., sensors for light, humidity, and vibrations): they allow to collect physical data, that can be passed to the upper layer component, *Data Processing*.

Data Processing

The *Data Processing* component receives sensed data and processes it, producing more refined data. Examples of data processing are discarding too noisy or irrelevant samples, averaging or calculating other aggregated results over time, and detecting physical conditions that should trigger a reaction. This capability is one of the main advantages of sensor nodes over simple sensors, because it allows to easily remove redundancy and minimize network traffic by transmitting aggregated data, that are more compact than raw values. The data processing component is not restricted to process locally collected data: in fact, if the device is part of a WSN, it can also process data transmitted by other sensor nodes to fuse it with local information, thus producing higher level data. To receive data from other sensor nodes, the data processing component must cooperate with the *peer coordination* component, that manages communications with other sensor nodes.

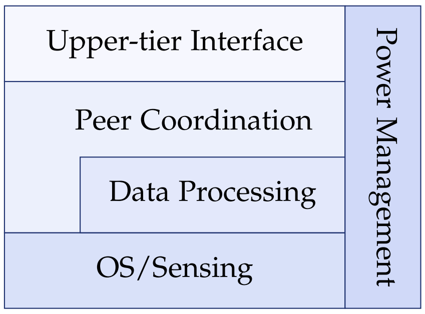


Figure 3.2: Components running on sensor nodes (physical sensing infrastructure).

Peer Coordination

If sensor nodes are densely deployed, thus making peer communication possible via low-power wireless interfaces, then the *peer coordination* component manages the collective behavior of sensor nodes, providing routing and high level network services, realized on top of the basic functionalities implemented by the OS/sensing component. It is a wrapper of routing algorithms presented in Section 2.2.1; thus, depending on the desired usage scenario of the WSN, it can provide dissemination, collection, or point-to-point communication functionalities. *Peer coordination* is the key component that enables scalability of WSNs, because it makes possible to quickly deploy a large number of sensor nodes that self-organize to sense the environment. Peer coordination uses lossy, low-power links, such as IEEE 802.15.4, hence this component, according to the *resiliency* principle, has to implement suitable mechanisms to reliably manage communications even when poor wireless link conditions cause high packet losses.

Upper-tier Interface

From the point of view of Pervasive Sensing systems in Future Networks, sensor nodes and WSNs are not stand alone: they are part of a larger system that spans different tiers. The *Upper-tier Interface* manages the connection of sensor nodes with the fixed and mobile infrastructure, usually to upload sensed data. Connections to fixed infrastructure can be wired or wireless: in the former case the sensor node is physically connected to a fixed network, for example via a Universal Serial Bus (USB) or ethernet cable, in the latter case, the sensor node is provided, in addition to the low-power wireless transceiver, with a long-range wireless interface, such as a 2G/3G/IEEE 802.11 interface. These sensor nodes are sometimes called *gateways* or *border routers* because they connect low-power WSNs with commodity, infrastructured networks. It is important to stress that having an additional special-purpose wireless interface increases significantly the cost of a sensor node, hence most WSN deployments limit the number of sensor nodes with two interfaces to minimize costs [127]. Sensor nodes can also communicate opportunistically with mobile devices, such as smartphones. In this case, the communication can be established on traditional, energy-hungry links, such as IEEE 802.11, or on low-power links, such as IEEE 802.15.4, depending on the hardware available on sensor nodes and on mobile devices. According to the *resiliency* principle, in the latter case the *Upper-tier Interface* may need to implement a mechanism for data transmission that minimizes packet drops over lossy wireless links. In addition to being a network interface, the *Upper-tier Interface* is also an access interface that allows mobile and fixed infrastructure to coordinate sensing tasks of sensor nodes and issue re-configuration commands. This open access makes

WSN integrated in Pervasive Sensing systems more dynamic and capable of being reconfigured after the deployment, thus they are an improvement compared to traditional, stand-alone networks, that are very hard to configure or re-program after their deployment.

Power Management

The *power conservation* principle that we adopted in the design of Pervasive Sensing systems clearly states the utter importance of a careful management of energy-intensive processes, because they can quickly exhaust sensor nodes batteries. The *Power Management* component oversees all activities on sensor nodes, tuning them to minimize battery consumption. Power management is a cross-layer functionality, because almost any functionality of a sensor node can be tuned and controlled to minimize its power consumption. For example, the power management component can modify the parameters of the wireless Media Access Control (MAC) protocol to shut down the radio transceiver more often, it can postpone processing of non-urgent data, and periodically shut down sensors and the microcontroller to save energy.

3.2.2 Mobile-Infrastructure Tier

The mobile-infrastructure tier comprises mobile devices, that can autonomously run sensing task, and can connect to devices in the physical infrastructure tier. By definition any mobile device could belong to this tier, such as laptops and PDAs, but in practice the most common ones are smartphones. The components running on mobile devices are very similar to those running on sensor nodes, as shown in Figure 3.3; most of the differences in the model derive from the intrinsic hardware and networking differences of the devices, analyzed in the following sections.

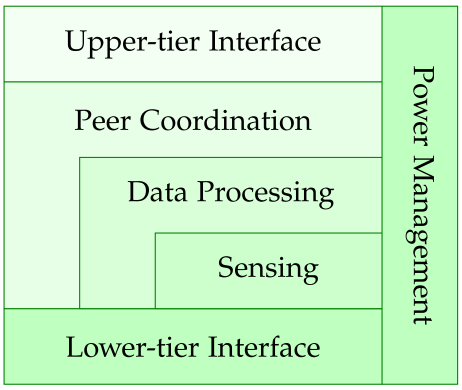


Figure 3.3: Components running on mobile devices (mobile infrastructure).

Lower-tier Interface

The *Lower-tier Interface* component manages the opportunistic interactions between mobile nodes and sensor nodes, both to exchange sensed data, and to issue re-configuration commands. As explained when describing its symmetrical counterpart, namely the *upper tier component* on sensor nodes, sensor node – mobile node communications links can be established on either energy hungry links or on low-power links [48–50]. The latter case is of major interest, because opportunistic networking over low-power links makes mobile infrastructure a cost-effective way of accessing physical sensing infrastructure in Pervasive Sensing systems.

Sensing

Smartphones are sensors from several points of view: they are mobile sensor nodes, hence they are capable of measuring physical properties, they are also social sensors, because they have access to social data generated by their owners, and they are also enabling devices for crowdsensing (Section 2.1). The *sensing* component wraps these functionalities, by collecting data and forwarding it to the *data processor* for further analysis and refinement. The role of the sensing component is especially relevant when accessing hardware sensors available on phones. In fact, most smartphone sensors are exclusive resources, namely they can not be accessed by more than one process at time, and they need to be accessed by user-level applications; for example the accelerometer is used to rotate the screen when the phone is horizontal, and the microphone is used during vocal searches and phone calls. Hence, they can not be acquired exclusively and for long periods of time. According to the *low disruption* principle, the *sensing* component arbitrates the accesses to sensors, serializing them and making sure that opportunistic sensing does not interfere with the normal usage of mobile devices.

Data Processing

The *Data Processing* component receives data from the *sensing* component, to process it and infer higher level data. It can also receive data by other mobile devices, via the *peer coordination component*, and from sensor nodes, via the *Lower-tier Interface*. The main difference of this component, compared with the *Data Processing* component on sensor nodes, is its much higher computational power. Thanks to the fast CPUs available on smartphones, the *Data Processing* component can run very complex analysis on sensed data, that would be impossible on sensor nodes, such as signal processing and machine learning algorithms.

Peer Coordination

Like its counterpart on sensor nodes, the *peer coordination* component manages communication with other mobile devices, by implementing routing algorithms such as those presented in Section 2.2.1, and their group behavior, such as cooperatively assigning sensing tasks. Peer coordination is a key element to enable scaling of Pervasive Sensing systems, because it encourages "*divide et impera*" solutions that distribute workload over many different nodes. Due to their higher communication capabilities compared to sensor nodes, that allow mobile devices to be always connected to the fixed infrastructure, peer coordination can be driven locally (i.e., via ad hoc networking with nearby mobile nodes) and globally (i.e., delegating organization decision to the infrastructure): in the former case *peer coordination* manages communication with other nodes, runs distributed algorithms that drive group behavior, and enforces resulting decision, in the latter case it receives configurations from the fixed infrastructure via *Upper-tier Interface* and enforces them.

Upper-tier Interface

The *Upper-tier Interface* manages connection of smartphones to the fixed-infrastructure tier. It allows to connect to the fixed infrastructure via 2G/3G/4G/IEEE 802.11; and it models the possibility of using very advanced communication protocols such as Session Initiation Protocol (SIP) [128], publish-subscribe framework, and web services, for more reliable data transfers. In addition, it receives configuration commands from the fixed-infrastructure tier, that may provide policies for sensing, for coordination with other mobile devices, or even for sensor nodes. In the last case, the configuration data is passed to the *Lower-tier Interface*.

Power Management

The *power management* component is a cross-layer mechanism that watches the Pervasive Sensing components running on smartphones and can manage their behavior to reduce power consumption. Simple examples of possible actions carried out by *power management* component are turning off unused wireless interfaces, pausing sensing activities, and temporarily disabling access to hardware sensors. Generally, aggressive power saving techniques reduce the impact of Pervasive Sensing on smartphones, usually at the cost of reducing the quantity of harvested sensed data, whereas lazy power saving policies allows to collect very detailed sensing data, at the cost of a very reduced battery lifetime. Hence, the challenge for power management for Pervasive Sensing systems on smartphones is to strike the right balance between high quality data collection and acceptable battery lifetime.

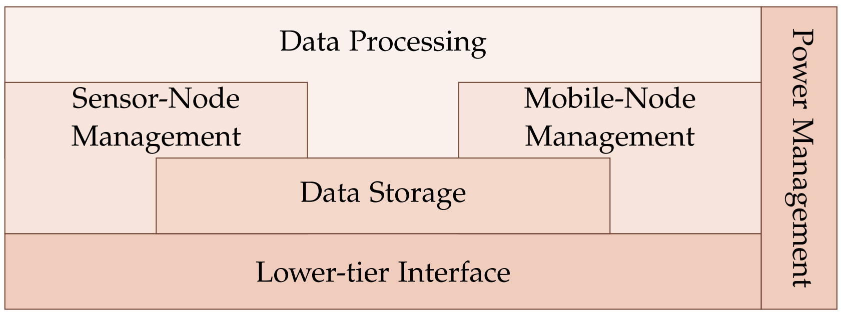


Figure 3.4: Components running on fixed infrastructure.

3.2.3 Fixed-Infrastructure Tier

The fixed infrastructure is the last tier in our reference scenario. In addition to traditional servers, it includes servers running in the cloud, namely virtual servers that can be easily allocated on-demand and destroyed when they are no longer in use. The reference logical model for devices in this tier is reported in Figure 3.4.

Lower-tier Interface

The *Lower-tier Interface* is the access end-point for mobile devices and sensor nodes to fixed infrastructure. In addition to supporting access over standard wireless links, such as 2G/3G/IEEE 802.11, the *Lower-tier Interface* also supports high level protocols such as SIP [128], that enable a robust communications between the the infrastructure and nodes from other tiers. It receives sensed data and forwards it to *Data Storage* component to archive it. Moreover, the *Lower-tier Interface* is the exit point for re-configuration commands targeted at mobile devices and sensor nodes, driven by *Sensor-Node Management* and *Mobile-node management* components.

Data Storage

Depending on sampling rate and number of devices participating, a Pervasive Sensing system can produce a huge amount of data, that is not manageable by sensor nodes and smartphones. For this reason fixed infrastructure includes a *Data Storage* component, either local or distributed, that stores all sensed data. *Data Storage* can be a relational database, a key/value data store or any other suitable persistent storage service. Stored data is accessed by the *Data Processing* component, that can analyze it and produce additional, higher level data, and by the *Sensor-node Management* and *Mobile-node Management* components, that use data about nodes in the lower tiers to take decisions about their management.

Sensors-Node and Mobile-node Management

The privileged position of fixed infrastructure in the multi-tier stack allows it to have a broad view of the status of devices in lower tiers. To exploit this information in the best possible way, our logical model proposes to run *Sensor-Node Management* and *Mobile-node Management* components on fixed infrastructure, that manage devices respectively in the physical infrastructure tier and in the mobile-infrastructure tier, issuing configuration commands that are dispatched to devices via the *Lower-tier Interface*. Examples of configurations generated by the *Sensor-Node Management* component are scheduling sensing activities and tuning parameters of the MAC protocol. Examples of configurations generated by the *Mobile-node Management* component assigning sensing tasks to mobile nodes in a specific area and scheduling data harvesting tasks on mobile nodes near a WSN. Let us note that the two management components can cooperate to manage sensor and mobile nodes; for example, if sensor nodes are not directly connected to infrastructure, then the *Sensor-node Management* component can generate a configuration and upload it on sensor nodes using mobile nodes, configured by the *Mobile-node Management* component, as opportunistic relays.

Data Processing

Notwithstanding the importance of computation run on mobile and sensor nodes by the *Data Processing* component, the fixed infrastructure is in an even better position for *Data Processing*. In fact, freshly collected data and historical values are stored in the *Data Storage*, thus it is possible to further process sensed data to get high level inferences that would be infeasible to get as result of distributed processing on sensor nodes and smartphones. Data resulting from data processing is not necessarily just passive data to be stored back in the *Data Storage*, because it can be fed back to the system as basis to manage sensor and mobile nodes. For example, data history about location of mobile devices can be processed to predict their movements; this knowledge can be exploited to assign sensing task in a specific area to nodes that are likely to pass nearby it, improving the overall performance of the Pervasive Sensing system [35, 129].

Power Management

The devices in the fixed-infrastructure tier do not have the same strict energetic constraints of sensor and mobile nodes, because they are directly connected to the electrical grid, often backed up by uninterruptible power supplies. Nonetheless, a *Power Management* component in fixed infastructure is of utter importance because energy consumption is the most important cost in data centers due to the power absorbed by servers and by air conditioning systems that keep them cool. Software power management can contribute to minimize energy consumption, complementing hardware approaches such as using low-wattage CPUs and optimizing the air-flow in data centers. More in detail, architectures based on cloud computing provide two main techniques to manage power consumption: live migration and on-demand resource allocation. Live migration allows to migrate virtualized systems from a physical server to the other, hence it is possible to consolidate running virtual machines on few servers, turning off those that are not currently used. On-demand system allocation is a generic term that highlights the possibility for scalable services running on the cloud to dynamically instantiate resources only when they are needed. For example if a web server on the cloud is overloaded a new one can be quickly started. The *Power Management* component supervises live migration and on-demand resource allocation, driving them based on cross-layer knowledge of the load on the system and possibly of other virtual systems in the same cloud.

3.3 RELATED WORKS

In the following we present the most relevant proposals that are close to the logical model presented in this chapter, while we will present other works more closely related to the specific applications of the logical model in following chapters. The logical model that we proposed for Pervasive Sensing systems in Future Networks is a recent research topic, that overlaps several different areas in the areas of urban scale sensing, tiered sensor networks, and delay tolerant sensor networks.

A seminal work about urban scale sensing is the MetroSense framework proposed by S. B. Eisenman [98]. MetroSense relies on a physical architecture that has three tiers, called Server Tier, Sensor Access Point (SAP) tier, and Sensor Tier. The Server Tier provides practically unbounded storage and computational resources and provides services such as data mining, data storage, and support for queries to select sensors. The SAP tier comprises all SAP nodes, namely nodes that offer gateway access to server tier for sensor tier elements and provide services such as managing rendezvous with other nodes and uploading tasks and configurations. The sensor tier is formed by sensor nodes and mobile sensor nodes as defined in Section 2.1.1; some of these nodes can double as SAP nodes, but they are not required to do so. The G-Sense architecture [130] defines autonomous systems that are based on four components: sensing devices that run data collection tasks, first-level integrators that receive and process data from sensing devices, a data transport network, and servers, that perform additional processing on sensed data. Each system can be federated to other systems for information sharing. Compared to MetroSense and

G-Sense, our proposal has a more modular architecture, that makes it suitable for people-centric sensing like the existing proposals, but can also be tailored for very different scenarios, such as using mobile nodes just as passive network bridge to other nodes and deploying completely distributed autonomous sensing systems that do not rely on fixed infrastructure.

Tiered sensor networks are similar to our proposal for the use of heterogeneous nodes for sensing. The ExScale project [127] uses heterogeneous nodes in WSN to improve network performances as the number of participating nodes increases, in particular, it requires WSNs to have a subset of highly powered, multi-homed nodes that can be exploited for faster networking and to improve connectivity. Siphon proposes to use special nodes provided with two radio interfaces: a low-power short-range radio to interact with sensor nodes, and a long range, high bandwidth radio. These special nodes form a network parallel to the WSN, that can be used to reduce network congestion by rerouting excessive traffic [131]. Gnawali *et al.* proposed the Tenet architecture in [132], which divides sensors in two tiers: a lower tier of cheap, low-powered sensor nodes, and an upper tier of high-powered "master" devices that have completely different hardware and software stacks. Tenet forces sensor nodes to collect data only and master devices to process it. This approach facilitates operations such as data aggregation, but also limits scalability of the whole system, making it economically unsuitable for large scale sensing. Several other proposal have leveraged tiered architectures in WSNs to realize improvements such as better connectivity coverage, longer battery lifetime, support for Quality of Service (QoS), and faster data processing [133–137]. However, all these works focus on static WSN, and have very little or no support for mobile devices and sensing task management.

Our system shares also some similarities with delay tolerant sensor networks that are characterized by the usage of opportunistic networking to route data in sparse sensor networks. The data MULE system focuses on exploiting mobile nodes as harvester of data collected by fixed sensors [8]. Its architecture is divided in three tiers: the lower level performs sensing, the middle level consists of mobile nodes named *MULEs* that roam over the area serviced by the lower tier to collect sensed data then forwarded to the upper layer that consists of gateways connected to a data storage facility. The message ferrying approach proposed by Zhao *et al.* in [124] implements a delay tolerant routing based on mobile nodes named *message ferries* that move around on predefined known paths: mobile sensor nodes can approach a message ferry to relay to it a packet, that is buffered by the ferry and forwarded to the destination node as soon as the ferry happens to encounter it. The Delay/Fault-Tolerant Mobile Sensor Network (DFT-MSN) architecture by Wang and Wu analyzes a scenario where all sensors are mobile and have a short radio transmission range; while roaming, they can run into fixed access points to the backbone network, that allow them to upload collected data [138]. The authors show that using flooding and network coding it is possible to achieve a good data delivery ratio while minimizing overhead. Khouzani *et al.* investigate in [139] the trade off between epidemic routing and energy consumption in DTNs, showing that simple threshold-based policies based on the remaining energy of each sensor are optimal for dynamic forwarding decisions. All these works are very focused on routing of sensed data while minimizing network overhead, delivery latency and energy consumption. Our proposal supports the implementation of these techniques, in additions it manages sensing tasks and enables integration with the fixed infrastructure, that can tune the parameters of data collection process.

3.4 OVERVIEW OF CASE STUDIES

The overview of related works in the previous section shows that there is an active interest in Pervasive Sensing systems. So far, research efforts have been focused on vertical architectures for urban sensing that delegate to the backend coordination of sensing on mobile nodes, or on optimized algorithm for data collection in WSNs. Our reference logical model has a modular design that spans these different scenarios and actually allows to integrate them. To test the validity of the proposed logical model, we used it as reference to develop several large scale Pervasive Sensing systems, hinged on different tiers of the reference scenario presented in Section 2.2 and focused on different sensing activities.

We defined three categories to classify Pervasive Sensing systems based on their goals to better present the implemented systems, ordered by increasing scope of their goals. The categories are: *data-centric*, *person-centric*, and *social-centric* systems. *Data-centric systems* focus on the collection, routing, and processing of sensed data; *person-centric systems* enhance the previous ones adding advanced processing and inference capabilities that provide personal services, possibly based on personal data; finally, *social-centric systems* extends the services provided on a personal basis to serve a whole community. In the next sections, we give an overview of the implemented systems, that are analyzed in detail in the following chapters.

3.4.1 Sensor Network and Smartphone Integration

The first category of Pervasive Sensing system is data-centric, with its goal of exploiting smartphones as opportunistic mobile infrastructure to provide high QoS routing to WSN (Figure 3.5). More in detail,

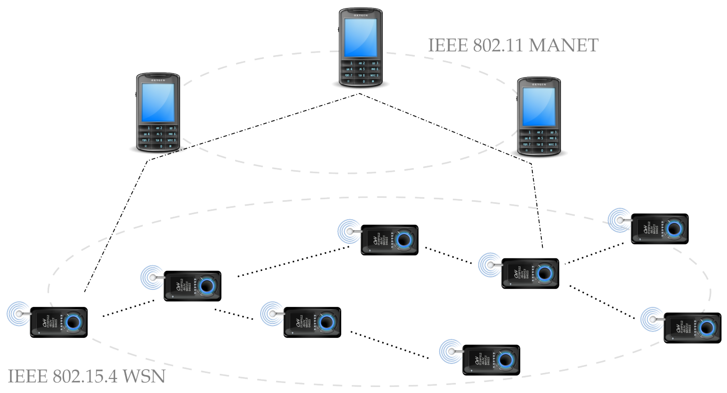


Figure 3.5: Integration of MANET and WSN.

this system, called WSN Hybrid rOuting prOtocol (WHOO) [140], derives from the reference logical model a novel WSN/MANET integration based on the primary design guideline of opportunistically exploiting MANET overlays impromptu formed over the WSN to improve and boost the data collection task of a typical WSN, in a completely distributed fashion that does not require a fixed infrastructure. On the one hand, it adopts a cross-layer approach that exploits MANET connections to differentiate and speed up the delivery of sensed urgent data by pushing them over low-latency MANET paths. On the other hand, it takes advantage of local cross-layer visibility of the WSN data collection procedures and protocols to carefully control and limit WSN–MANET coordination overhead. We prove that our proposed solution can obtain significant quality of service improvements via differentiation, by granting faster delivery times to urgent data with a very limited cost in most common execution scenarios.

3.4.2 Centralized Sensor Network Management via Smartphone Integration

The second Pervasive Sensing system that we present is a data-centric three tiered system for the management of sparse sensor nodes (Figure 3.6). This system uses the support session control and interoperability in future networks made available by the IP Multimedia Subsystem (IMS) to opportunistically use smartphones, that roam in geographically sparse environments (such as a smart city), as WSN data harvesters and as relays to upload on sensor nodes configuration parameters decided by the fixed infrastructure. The primary design guideline is to exploit IMS and the recently released IMS Presence Service to effectively coordinate mobile nodes, thus saving energy on sensor nodes via reduction of unnecessary communications. Our logical model allows to effectively design an architecture tailored for this scenario, unambiguously assigning responsibilities to each de-

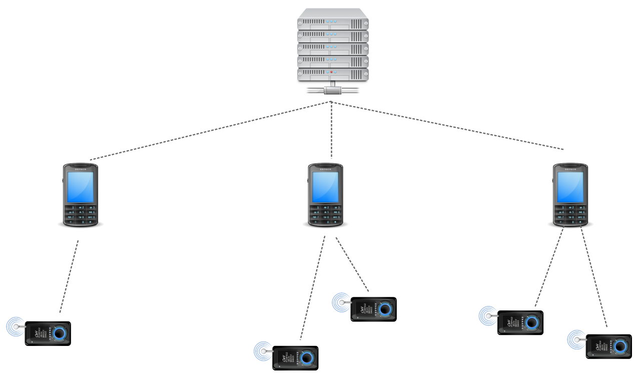


Figure 3.6: Integration of MANET and WSN.

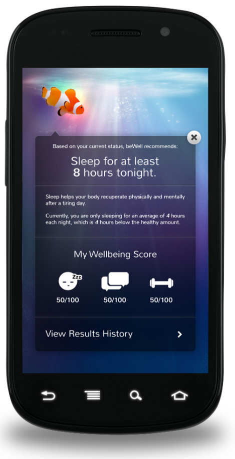
vice, while promoting opportunistic interaction to minimize energy consumption on battery-operated devices.

3.4.3 Smartphones as Sensors: Healthcare with BeWell

We now shift the focus from data-centric to person-centric Pervasive Sensing systems, presenting the architecture of BeWell, a joint project with Dartmouth College and Cornell University. BeWell is a mobile app that uses sensors available on smartphones to automatically monitor user everyday lifestyle, and uploads collected data to the fixed infrastructure, that provides advanced data mining functionalities [141] (Figure 3.7). This chapter analyzes the core component of BeWell that enables continuous, low-power sensing and provide high level inferences from raw sensor data. The core component design is derived from the components defined by our reference model for mobile nodes, and it is actually a completely reusable and flexible platform, that eases the development of mobile sensing applications through the definition of a common set of facilities that masks all low-level technical details in reading and processing raw sensor data.

3.4.4 Smartphones as Social Sensors: Recommendation and Participative Sensing

Finally, we present two social-centric systems, that exploit user mobile devices and users themselves as sensor to provide socially relevant services; they both opportunistically exploit devices in the mobile-infrastructure tier for sensing and the fixed infrastructure for coordination of the devices. The first project, Social-Aware IMS-enabled Recommender (SAIR), is a recommendation system: it measures users engagement when using smartphone applications and correlates it with the context of the usage (e.g., location, physical activities, recent interests posted on social networks); this knowledge is then exploited



Based on your current status, beWell recommends:

Sleep for at least
8 hours tonight.

Sleep helps your body recuperate physically and mentally after a tiring day.

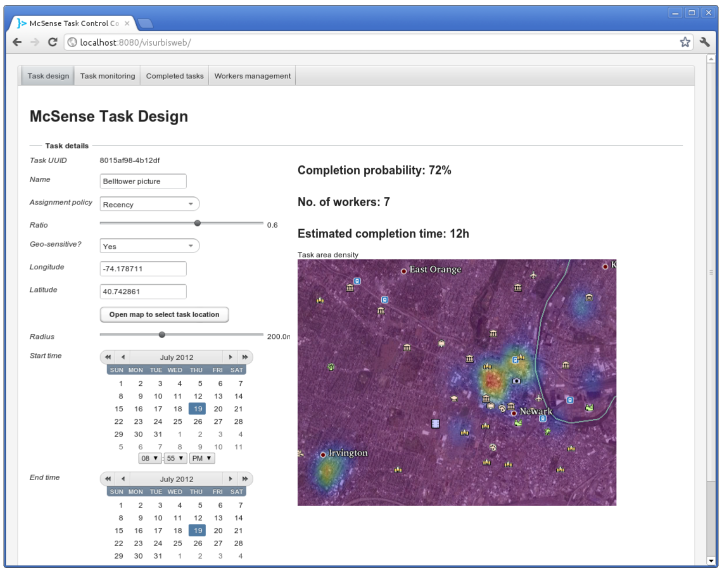
Currently, you are only sleeping for an average of 4 hours each night, which is 4 hours below the healthy amount.

Figure 3.7: Smartphone running BeWell, a healthcare app based on mobile sensing.

to suggests new applications to users that are in a similar context. The second project, named McSense, is a joint project with the New Jersey Institute of Technology (NJIT) focused on study of incentives to drive crowdsensing (Figure 3.8). Here we will analyze the issues related to using smartphones as sensors to optimize task assignment to users for crowdsensing, by collecting data that allows characterization of regions to monitor, evaluation of a good balance between sensing accuracy and resource usage (e.g., number of people involved, network load, battery level), and profiling users to know which sensing tasks are more likely to execute successfully.

3.5 CHAPTER CONCLUSIONS

In this chapter we have presented the main issues of Pervasive Sensing systems in Future Networks, namely the management of tier imbalance, the necessity of using a software design that eases scaling up as the number of users increases, and the importance of reducing the energy consumption on smartphones and WSN to avoid detrimental effects on user experience and on WSN battery lifetime. To address these issues we presented eight design principles (i.e., *modularity*, *opportunistic behavior*, *adaptability*, *power conservation*, *resource awareness*, *scalability*, *resiliency*, and *low disruption*), that we translated to a general multi-tier logical model, that overlays the structure of Future Networks. The logical model ensures that every sensor and mobile node is provided with components to collect and process data, and to communicate with peer nodes and, possibly, with devices in other tiers, while guaranteeing that the fixed infrastructure can receive and process collected data. After describing the details of the logical model,



McSense Task Design

Task details

Task UUID 8015af98-4b12df

Name: Belltower picture

Assignment policy: Recency

Ratio: 0.6

Geo-sensitive?: Yes

Longitude: -74.178711

Latitude: 40.742861

Radius: 200.0m

Start time: July 2012, 19; 08 : 55 PM

End time: July 2012, 19

Completion probability: 72%

No. of workers: 7

Estimated completion time: 12h

Figure 3.8: Control console of McSense, an application for management of crowdsensing tasks.

we compared it to existing solutions. Finally, we presented five Pervasive Sensing applications with different goals and requirements, whose architecture is based on the our logical model. The next chapters will analyze in detail the case studies, highlighting how the reference logical model drives the design and implementation of robust and scalable Pervasive Sensing systems.

4 SENSOR NETWORK AND SMARTPHONE INTEGRATION

THE first case study that we present as implementation of the reference model presented in Chapter 3 is a data-centric application that exploits MANETs in the mobile infrastructure tier to improve data collection on WSNs in the physical sensing tier. In fact, the possibility of integrating WSNs and MANETs paves the way to brand new cross-network routing opportunities to overcome the typical limitations of WSN data collection solutions. For many WSN applications, it is possible to identify specific classes of traffic that are urgent (for example, they may be measurements that are connected to possible future critical condition or may be part of a premium service) and would benefit from a specific, low-latency delivery. Current WSN data collection solutions are usually unable to grant timely delivery of urgent data with low-latency requirements [75, 142]. In fact, data must traverse a large number of potentially congested WSN nodes before reaching the ultimate gateway toward the fixed infrastructure; hence, data delivery time could unacceptably increase, especially if low-power communication mechanisms can impose radio switched-off times. In addition, existing data collection solutions typically tend to interfere and mix together in-band urgent data with normal non-urgent ones.

To overcome all the above problems, research about tiered sensor networks, described in Section 3.3, have started considering the possibility of using special nodes, immersed in the WSN and equipped with both low-power and powerful ad hoc wireless radios, that can play as relays to facilitate WSN data collections [127, 131]. These nodes, dynamically organized in MANET overlays, could accelerate WSN data collection toward collection points. However, notwithstanding their potential, those seminal proposals have not been widely deployed because they required special-purpose sensor node hardware, that is more expensive, requires more energy and, consequently, more maintenance. Due to the possibility for mobile phone to host on board low-power communication technologies and directly interact with sensor nodes [48–50], we propose to exploit these new mobile devices to help in WSN data collection. Let us stress that, differently from other approaches in the literature, we do not consider these nodes neither as mobile harvesters that relay sensed data to the Internet, nor as mobile users of WSN, rather as efficient relays taking part in the WSN data collection to speed it up with the help of other MANET nodes.

This chapter presents an architecture based on the logical model of Chapter 3 for cross-network opportunistic data collection for WSN in mobile environments with several contributions:

1. we propose an opportunistic solution that exploits the network layering created by a MANET within a WSN, without assuming any previous knowledge about node mobility, to differentiate and speed up data collection, especially tailored for most valuable urgent data;
2. we adopt a cross-network and cross-layer design that limits the communication over the WSN by activating interactions between MANET and WSN only when necessary, typically for urgent data delivery, by integrating and fine-tuning local execution of the data collection protocol to eliminate any possible energy waste;
3. we dynamically adjust the proposed protocols to MANETs and WSNs to find the best balance between the benefit of enabling a higher level routing layer and the cost of coordinating both MANETs and WSN.

Our urgent data collection solution is fully compliant with the Collection Tree Protocol (CTP) standard, integrates with existing open-source WSN implementation platforms and tools such as TinyOS operating system and components [83], and is publicly available for WSN practitioners. Our experimental setups show that our solution can effectively differentiate and speed up urgent data delivery with controlled, predictable and very limited cross-network coordination overhead.

This chapter is organized as follows. Section 4.1 presents the motivating scenario for our work and the design guidelines that drive our study. Section 4.2 describes the distributed architecture of our proposal and the facilities that enable the opportunistic integration of its main components. Section 4.3 derives the software architecture of our protocol from the logical model of Chapter 3. Section 4.4 presents the MANET–WSN integration protocol and the theoretical analysis to illustrate its optimal low cost. Section 4.5 discusses the implementation detail of our protocol. Section 4.6 that presents analytical and experimental results that support our proposal. The chapter finally presents a brief survey of related works (Section 4.7), and the chapter conclusions in Section 4.8.

4.1 BACKGROUND AND REFERENCE SCENARIO

This section details the motivating scenario for urgent data delivery and the design guidelines that stem from that environment and from the peculiarities of involved WSN and MANET networks.

4.1.1 Urgent Data Delivery Motivating Scenario

To better clarify the importance of urgent data delivery for WSN, let us introduce a real-world example. Our main motivating scenario is the realization of quality-enabled networks for environmental monitoring in emergency prevention and response scenarios such as underground mines. That example is typical for all those environments that should take into account known and avoidable dangerous situations in operating conditions.

Monitoring of mine tunnels is a crucial task to ensure workers' safety, by sensing several factors, such as wall stability and percentage of air components, mainly oxygen, methane, water and dust. The monitoring of the structural stability of mine walls is a task of paramount importance: of the 480 coal mine fatalities in the past 10 years in the USA, 50% where caused by the fall of ribs or high walls [143]. WSN technology could greatly help in such scenarios: it does not require wiring, its deployment is simple notwithstanding poor working conditions and high maintenance costs underground, and it scales well in long and narrow tunnels. In particular, the stress of walls can be monitored by sensor nodes equipped with strain gauges and fiber optic strain sensors, that can not only detect catastrophic collapses of walls, but also forewarn changes such as cracks and flexures [144, 145]. Moreover, wireless sensor nodes can spontaneously form impromptu WSNs to collect and forward measurements to a data sink that can work as a gateway toward the fixed Internet.

During monitoring operations, measured data can be both in either normal range values (e.g. normal levels of oxygen) or it can be a warning that associates abnormal values with some environmental parameters that, if not cared about, may worsen and cause major failures (e.g. detection of slight deformation of a rib, which signals that it requires urgent maintenance): warning-level data should be marked as urgent and delivered very fast to data sinks, possibly overcoming non-urgent data. Since miners and machinery are already equipped with portable ad hoc devices able to form MANETs for system control and team communication, it is possible to use MANETs also to create low-latency paths for urgent data delivery.

Of course, most deployment scenarios that exploit WSN for environment monitoring share the same essential trait of producing both normal and urgent data, that would greatly benefit from differentiated routing. Examples of such deployments are WSNs for structural monitoring to prevent collapses of old buildings, for detection of pollution in urban and industrial areas, for measurement of road conditions and so forth.

Stemming from the above application scenarios, we can directly derive our main requirements. First of all, we want the WSN urgent data relay and flow over more powerful MANET network trunks as soon as the WSN can communicate with a mobile node. That goal is crucial because integrating MANET with WSN can alleviate both fundamental WSN communication problems: scarcity of energy and low communication bit rate [74]. Moreover, even if MANET devices suffer constraints on computational power, communication bandwidth, and available energy, MANET constraints are of orders of magnitude weaker than the WSN ones; therefore, whenever possible, MANETs should take over the urgent WSN communication load. In addition, since communication between MANET and WSN could drain precious WSN node energy resources, it is of utter importance to design solutions and protocols that minimize MANET–WSN interaction by employing it only when necessary, such as for urgent data delivery, and to automatically adapt system configuration depending on the current execution environment, especially based on the density of WSN and MANET nodes in reciprocal visibility. Finally, the resulting integrated data collection system should not only grant fast delivery of urgent data, but also carefully monitor MANET–WSN reciprocal visibility to prevent possible packet losses due to MANET node movements.

4.1.2 Desing Guidelines for Mobile WSN Data Collection

The realization of a real-world system to support cross-network data collection of normal and urgent WSN data is a challenging task that requires a deep understanding of several management issues that span different networks and different protocol stack layers. To enable fast and low-cost data delivery in mixed MANET–WSN environments, we distilled the following five main design guidelines, derived from the principles presented in Section 3.1.

First, the system should have an *opportunistic behavior*. It should be able to take advantage of any communication opportunity between the MANET and the WSN. MANET nodes are mobile, and thus the time frame when one WSN node can communicate with one MANET node may be at any time and connections can have an unknown duration; thus, it is crucial to exploit them as soon as they are available, but be ready to react when no longer available. Some forms of initial discovery and continuous advertisement between the MANET and the WSN are necessary to integrate MANET node mobility.

Secondly, MANET–WSN integration should be as much as possible *power conservative*. The MANET should not blindly integrate with the WSN when there are no urgent data to route, thus wasting sensor node battery; instead, the integration should be reactive: The MANET should typically stay in a dormant state (with no MANET–WSN traffic exchange) and react to urgent data arrival, by waking up and starting to route urgent data. To sense urgent data arrival without draining precious energy resources for additional communication, the MANET should exploit traffic snooping techniques. Finally, the management of MANET mobility should be carefully designed to control and limit the overhead of all protocols used for monitoring node mobility.

Thirdly, all cross-network protocols should be *adaptive*. Because the dynamics of the MANET–WSN interaction depend on several parameters, data collection solutions should have cross-network and cross-layer visibility of all system parameters (mainly WSN and MANET size, average number of WSN nodes visible by each MANET node and MANET nodes speed) and exploit that awareness to adapt integration policies and time configurations at different protocol stack layers.

Fourthly, to be resilient the integration should be *localized*. It is well known in the literature that MANETs have capacity and bandwidth problems when their size and routing Path Length (PL) increase [146, 147]; hence, MANET–WSN integration should be enabled only when useful, by avoiding the fragility of large mobile networks, and hence by limiting the number of involved MANET nodes. In other words, the MANET should organize itself into impromptu formed, independent, limited-size clusters as soon as there are urgent data to route and be freed quickly when it is no more needed.

Finally, it should be *off the shelf*, to facilitate the acceptance and penetration of the proposed solution so as to ease integration with existing data collection proposals and seamless inter-working with them. In particular, we claim the importance of accepted tree-based data collection protocols, such as Hyper, CTP and ZigBee [75, 86, 142]; multi-homed MANET nodes functioning as a bridge between the MANET (ad hoc IEEE 802.11) and the WSN (IEEE 802.15.4) should support most widely diffused tree-based data collection protocols to make cross-network communications viable and easy.

4.2 OPPORTUNISTIC LOW-COST NETWORK INTEGRATION

According to the above design guidelines, we have realized our cross-network opportunistic data collection solution; this section describes the distributed architecture of our proposal and its main facilities; then it details our protocol to guarantee cross-network integration with a low communication cost by focusing both on the WSN layer and on the MANET layer.

4.2.1 Background and Reference Architecture

The abstract reference model of our distributed architecture spans different network layers and includes several distributed entities; this section gives some needed background material about the data collection protocol.

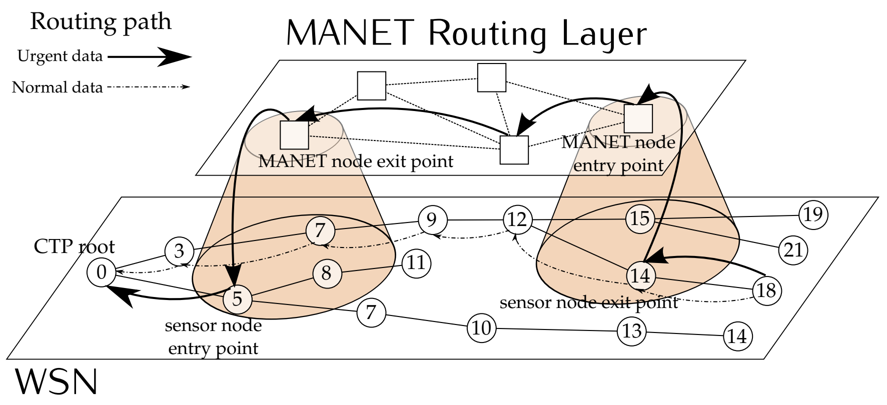


Figure 4.1: Overview of the scenario showing the WSN and MANET routing layer. The numbers reported for each WSN node are their costs. Dashed and normal arrows show the routing path for normal data and the desired routing path for urgent data.

Our model includes two tiers: the lower tier is the physical sensing infrastructure tier, i.e., the WSN network, sensor nodes form an autonomous routing layer that routes normal and urgent data to one or more data roots, while the higher tier is the mobile-infrastructure infrastructure tier, where multi-homed mobile MANET nodes roam across the WSN-equipped environment (Figure 4.1). We assume to use tree-based data collection for WSN, that leads to organize the WSN in a tree-like topology and to exploit a very general tree formation method based on a gradient function [75, 86, 142]. Data roots start advertising a zero cost, while each internal node advertises a total incremental cost, equal to the cost of its father node plus the cost of the link to the next hop: data packets flow along paths to lower cost nodes (dashed arrows in Fig. 4.1). The MANET level opportunistically exploits its nodes in visibility with WSN sensor nodes to create an additional low-latency high-bandwidth layer to route urgent data. To glue together WSN and MANET networks, MANET nodes exploit their WSN interface to reactively participate in urgent data routing by dynamically discovering sensor nodes at the WSN layer as they move by, and by advertising their presence to them. To overcome possible mobility and scalability issues typical of large and dense MANET deployments, we organize mobile nodes in small local clusters.

To make our system easier to understand, let us define here some useful terms (at Figure 4.1). *Roots* are sensor nodes that advertise themselves as collection tree roots: they are typically gateways to the Internet and offer interfaces, such as wired, satellite and wide area cellular ones (in Figure 4.1 there is only one root advertising a zero cost); all other *sensor nodes* build routing trees to forward collected data toward roots at the WSN layer. The WSN *exit point* is any WSN node able to forward packets to the MANET because it can directly communicate with at least one MANET node (in Figure 4.1 the WSN exit point is the sensor node advertising a cost of 14; it is the one that routes urgent data), while the WSN *entry point* is the WSN node with the lowest gradient cost that the MANET cluster can reach (in Fig. 1 the sensor node advertising a cost of 5). Similarly, MANET *entry/exit points* are MANET nodes that can, respectively, receive/forward data from/to the WSN.

Our solution is general enough to work with most tree-based collection protocols; however, we preferably adopt the CTP [75] because it is representative of a wide set of tree-based collection protocols and it is robust and thoroughly assessed, apart from being widely available as the default collection protocol on TinyOS [83]. The CTP uses the estimated number of transmissions as the cost metric [148], builds minimum cost data collection trees by using control beacons exchanged by WSN nodes and keeps the topology coherent by exploiting both control beacons and data piggybacking to disseminate control information about sensor node gradient costs. The low-overhead algorithms and protocols used by the CTP allow quickly building and reconfiguring the routing topology, while keeping low the control traffic overhead in stable conditions; for more details about the CTP, refer to [75, 149]

4.2.2 WSN-side Facilities for MANET-WSN Integration

This subsection overviews our facilities for MANET–WSN integration by addressing specifically the interactions within the WSN. Our proposal aims to enable inter-network urgent data forwarding over the MANET by seamlessly integrating with and by extending CTP; the main goal is to let MANET nodes locate and reach those WSN nodes more convenient for data collection, and to advertise the upper MANET routing layer to the lower WSN network at the same time.

To acquire and create reciprocal knowledge of WSN and MANET nodes, we need two facilities: *discovery*, to permit MANET nodes to explore the tree collection topology to select the best MANET node exit point, and *advertising*, to make sensor nodes aware of the available MANET node entry points. In particular, the *discovery facility* exploits local MANET clusters, created impromptu when an urgent data routing situation occurs, to scan the WSN to select sensor nodes with low gradient costs as WSN entry points. It should be noted that a functionally equivalent of the discovery facility would be to have sensor nodes periodically advertise their gradient value. However, unsolicited WSN-sent advertisement would be blindly transmitted without knowing if there is a MANET node that could receive it, thus unnecessarily draining precious battery power. For this reason MANET-sent discovery messages are energetically more efficient. After the cluster has obtained the best WSN entry point cost, it starts to *advertise* it to all reachable WSN exit points, by forming urgent data routing paths from entry to exit MANET nodes (see details in the next subsection). Regardless of the inner workings of MANET–WSN integration, it comes without saying that continuous interactions with sensor nodes, if they were always on, would significantly shorten the WSN battery lifetime. Hence, it is crucial to activate integration protocols reactively, only when there are urgent data to route and to minimize the number of message exchanges so as to lower communication energy costs.

Therefore, according to our power-conservative design principle, the MANET cluster is on (*running* state) only as long as it receives urgent data from the WSN, and it switches off as soon as these data stop (*dormant* state). The scheduling of the switch-off time is difficult: going to the dormant state too early means swinging back to running state for additional single urgent packet, while switching too late leads to higher energy consumption. We propose a novel power management strategy that models MANET–WSN integration analytically and evaluates the threshold traffic level to sustain it; our strategy monitors and controls several configuration parameters, such as WSN size, MANET cluster size and discovery/advertising periods, and adaptively chooses the intervals for the running and the dormant state, as detailed in Section 4.4.

To further lower energy consumption, we have also carefully designed our *discovery* and *advertising* facility protocols. In the discovery protocol, each MANET cluster node obtains CTP gradients by periodically emitting a request and expecting replies from WSN nodes in visibility (we call it *full discovery*). To limit the energy cost of continuous requests/responses, we have defined also an asymmetric discovery protocol, where sensor nodes have to reply only to the requests coming from not recently listened nodes, by using *lazy discovery*. Let us stress that the discovery protocol enables low-cost monitoring of nearby WSN nodes: as long as a MANET node does not change its radio neighborhood, which typically occurs either because of interfering events (such as suddenly interposed obstacles) or mobility, lazy discoveries reach the same WSN nodes that will not reply, thus saving energy. When discovery messages will reach new WSN nodes, they will send a reply, thus letting the MANET node know that the radio neighborhood has changed. About the *advertising* protocol, it enables data routing from the WSN to the MANET: each MANET node periodically beacons to the WSN a message with the gradient of the farthest WSN node the MANET cluster can reach; to limit message exchanges, only MANET nodes that overhear urgent data transmissions participate in advertising. WSN nodes, once having received the advertisement, compare advertised gradients with the ones they see at the WSN layer and decide whether they have to intervene and route urgent data over the MANET.

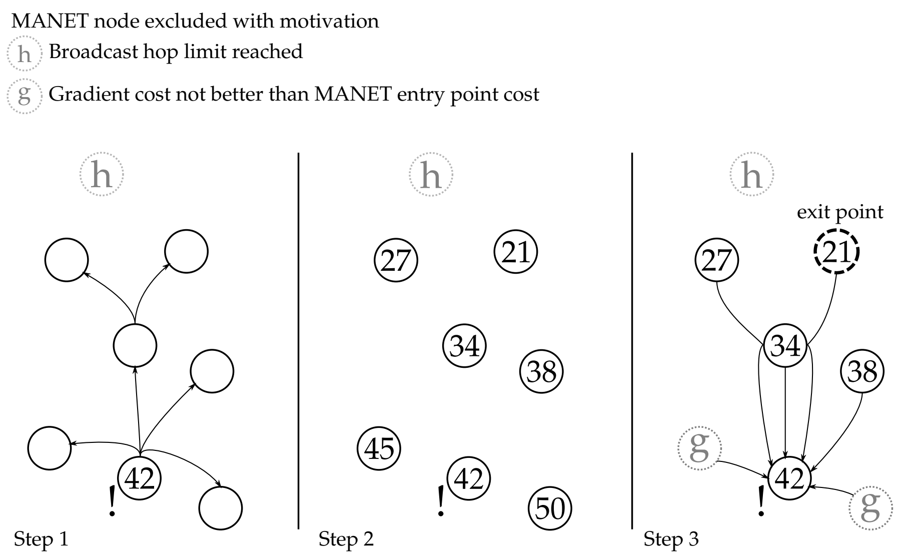

MANET node excluded with motivation
h Broadcast hop limit reached
g Gradient cost not better than MANET entry point cost
h
42
!
Step 1
h
27
21
34
38
45
42
!
50
Step 2
h
exit point
27
21
34
38
g
42
!
g
Step 3

Figure 4.2: Phases of the MANET cluster formation. Phase 1: a MANET node snoops an urgent data packet and broadcasts a 2-hop limited broadcast request. Phase 2: MANET nodes hit by the request send a full discovery request to the WSN, obtain the gradient cost of reachable sensor nodes and choose the best one. Phase 3: all sensor nodes reply to the MANET node that started the process. MANET nodes marked with a "g" will not take part in the cluster because their gradient cost is not better than the one broadcasted in phase 1.

4.2.3 MANET-side Facilities and Protocols

Our solution exploits MANET clusters opportunistically formed and localized on restricted areas centered where urgent message transmission is needed; the proposed cluster formation protocol is simple and robust with its own packet routing at the application level, and hence it does not require any underlying network layer ad hoc routing protocol: it uses only 1-hop communications and some limited broadcasts. Even if our proposal does not intrinsically impose hop limits on broadcasts to form and maintain clusters, we have found that a radius of 2-hops is a good trade-off between MANET size and achieved robustness. That is the reason why in the following we assume that all MANET broadcasts are propagated in a 2-hop neighborhood.

The cluster formation protocol is followed reactively by any MANET node that overhears a data urgent transmission at the WSN layer and consists of three phases (Figure 4.2). In the first phase, the MANET node that snoops the urgent data (acting as the MANET entry point) extracts from its header the sensor node gradient, propagates it to its 2-hops neighborhood and, with the same message, asks other MANET nodes for the best reachable gradient. In the second phase, every MANET node in the 2-hop neighborhood sends a full discovery request to sensor nodes and, based on received replies from the WSN layer, determines whether it is part of the MANET cluster. The cluster membership of MANET nodes too close to the potential entry point would engage resources while providing no relevant benefit to the system: hence a MANET node does not participate in the cluster if it is unable to provide access to a WSN node with a cost strictly better than the one declared in the received broadcast request. In the third phase, every MANET node reached by the broadcast replies to the MANET entry point with its bid of being member of the cluster (or not) and the gradient cost of its reachable sensor nodes. Finally, the MANET entry point that started the clustering formation protocol and plays the cluster-head role chooses the lowest gradient node as the MANET exit point, while collecting important information about the current deployment, such as the number of messages exchanged with the WSN and the number of MANET cluster nodes to be used to better schedule MANET switch-off time.

Before delving into internal cluster maintenance details, it is worth remarking that when CTP routes the first urgent message on the WSN, more MANET nodes along CTP data delivery path could snoop it and initiate, as cluster-heads, new concurrent cluster formation phases. During this initial transitory phase, there may be several overlapping clusters in the same 2-hop neighborhood, but this behavior does not jeopardize cluster organization because when urgent data start to be routed by a preceding cluster-head, following cluster-heads, along the same CTP path and in the same 2-hop neighborhood, can detect no urgent data on the WSN (already routed over the MANET) and switch-off to the dormant state.

Within each cluster it is the cluster-head that manages routing and coordinates between other MANET nodes of the same cluster; MANET entry points (potentially many in the cluster) forward urgent data to the cluster-head to be routed to the MANET exit point. In particular, the cluster-head coordinates also the energy-saving strategy by emitting periodically 2-hop cluster keep-alive broadcast messages confirmed by cluster members so as to refresh cluster information; each cluster member automatically switches back to the dormant state when it does not receive such messages, because either the cluster-head has switched-off the cluster (stopping keep-alive transmissions) or the MANET node has moved far away from the cluster.

Finally, let us focus on the mobility of either the cluster-head or one cluster member. When the cluster-head moves, it can no longer intercept urgent traffic; in this case, it stops sending keep-alive messages and the cluster will switch-off eventually. When the MANET cluster member moves, on the other hand, it may either reach sensor nodes with a different gradient cost or be too far from the cluster to establish a successful communication. Both these situations are handled by using our keep-alive cluster maintenance protocol: cluster members reply to the 2-hop keep-alive broadcast by sending back to the cluster-head the best reachable gradient and, when a cluster member moves so far that it is not able to send replies to the cluster-head, the cluster-head considers the member unreachable and cancels it from the cluster.

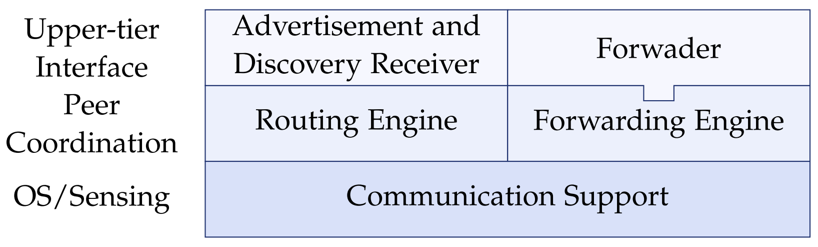
4.3 WSN HYBRID ROUTING PROTOCOL SYSTEM ARCHITECTURE AND PROOTOCOLS

We have designed and realized a working prototype called WSN Hybrid rOuting prOtocol (WHOO) that implements the distributed architecture, facilities and protocols, described in the previous sections, based on the logical model described in Chapter 3. In the following, we sketch the layered architecture of main distributed component and we provide a detailed description of our power management strategy to schedule the MANET switch-off time and of all low-cost integration protocols at the WSN layer.

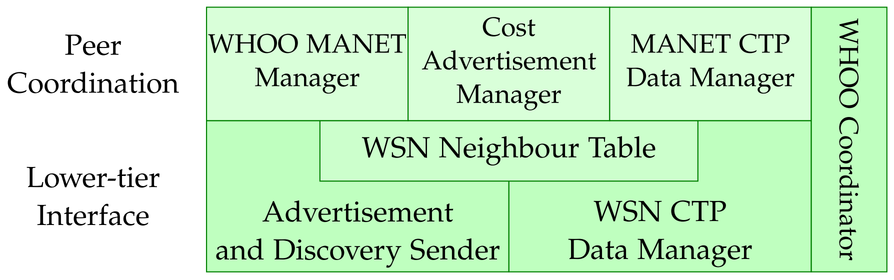
4.3.1 WHOO Layered Software Architecture

An important goal of our layered architecture is to hide complicated underlying sensor node communication mechanisms and to neatly separate our cross-network routing support from WSN data collection protocol (CTP in the current WHOO implementation). We focus first on WHOO sensor node internal architecture, which is organized in three levels (Figure 4.3a): the *OS/Sensing* level offers low-level support for sensor node hardware access; the *Peer Coordination* level realizes the tree-based collection protocol [75] and the highest *Upper-tier Interface* realizes our implementation of the MANET–WSN integration. WHOO does not explicitly mandate any data processing or additional power management functions, hence these components of the logical model are not present in this software architecture.

The operating system provides a *Communication Support* that enables reliable and unreliable 1-hop transmission functions; The *Peer Coordination* level implemented by CTP is developed atop and consists of two components: the *Routing Engine* manages the routing table and the beaconing of CTP routing packets and the *Forwarding Engine* is in charge of forwarding collected data at the WSN layer. The *Upper-tier Interface* is the upmost level and includes two components: the *Advertisement and Discovery Receiver* accepts incoming advertisements/discoveries from MANET nodes, while the *Forwarder* intercepts traversing urgent data and exploits cross-layer visibility of the *Forwarding Engine* at CTP level (as pointed out in Fig. 4.3b) to decide whether to forward them over the MANET or via the CTP data collection path.



(a) Sensor node.



(b) MANET node.

Figure 4.3: WHOO software architecture.

The MANET nodes internal architecture is also layered with two main levels: the *Lower-tier Interface* manages the integration with the WSN layer, while the *Peer Coordination* level handles coordination among MANET nodes. The *Lower-tier Interface* consists of two main components: *Advertisement and Discovery Sender* realizes our WHOO advertisement and discovery protocols by interacting with its counterpart running at the sensor node *Upper-tier Interface*; the WSN CTP *Data Manager*, on the other hand, glues together WSN data collection and MANET data routing by enabling data reception/transmission functions. At the MANET entry point, the WSN CTP *Data Manager* receives urgent data from the sensor node *Forwarder* component and at the MANET exit point it sends urgent data back to the WSN as normal CTP messages, received and managed by the sensor node *Routing Engine* component. At the same *Lower-tier Interface*, the WSN *Neighbour Table* aggregates the information extracted by CTP routing and data traffic to create and maintain, for each MANET node, a routing table toward all reachable WSN sensor nodes.

The higher *Peer Coordination* level coordinates MANET nodes and includes three main components: the WHOO MANET *Manager* deals with initial cluster formation and lifecycle management, until the cluster goes back to the dormant state; the *Cost Advertisement Manager* runs at the cluster-head node, it receives gradient updates from other MANET nodes and keeps the MANET cluster up to date with the latest best reachable gradient, so as to enable consistent advertising; finally, the MANET CTP *Data Manager* forwards CTP data from entry to exit MANET points by using source routing techniques to reach the local cluster-head, acting as a centralized routing entity and to deliver the urgent data to the MANET exit point. The *Whoo Coordinator* includes all the decision logic that enables power management and coordinates *Peer coordination* and *Lower-tier Interface* levels; for instance, at the cluster-head node it schedules the MANET switch-off time, while at cluster members it receives cluster keep-alive messages and de/activates discovery and advertisement protocols conveniently, as detailed in the next section.

4.4 ADAPTIVE SWITCH-OFF MANAGEMENT

The scheduling and the evaluation of the MANET switch-off time stems from a balancing activity between the running state traffic related to (continuous) periodic lazy discovery and advertising, and the cost of initial full discovery for each dormant-to-running-state switching. This balance dilemma belongs to the class of problems to help in choosing between periodic cost, paid repeatedly (rent a pair of skis) and a determined price paid once (buying price) also known as the "ski rental problem" [150]. It can be demonstrated that the optimal off-line deterministic strategy to minimize losses for this class of problems is to pay the repeated rent cost until it is equal to the buying price; after that it is better paying the buying cost. In our scenario, it means that the optimal strategy is to keep the MANET cluster in running state until the sum of messages periodically exchanged for lazy discovery and for advertising equals the number of messages exchanged for initial full discovery, namely to switch from dormant to running state. Karlin *et al*. demonstrated in [151] that this strategy is two-competitive, namely never sends more than twice the number of messages of an ideal online strategy that could forecast urgent packet arrival time and apply the best possible strategy to switch the MANET back to the dormant state. Even though there are probabilistic strategies that offer better cost guarantees, we chose to use the two-competitive strategy because it is the simplest and achieves easily reproducible results [152].

More formally, the proposed strategy is based on two metrics: MANET *pressure* (MP) is the repeating cost of renting and MANET *wakeup* (MW) *cost* is the one-time buying cost to switch on the MANET cluster. MP accounts for all messages exchanged between the WSN and the MANET while in a stable running state situation due to advertising and lazy discovery. Since sending and receiving messages usually requires comparable amounts of energy when not employing specific power transmission tuning [73], the MP is a good estimation of the energy cost of using the MANET for routing. MW cost, on the other hand, accounts for messages exchanged in the initial full discovery load by all MANET cluster nodes to obtain sensor nodes gradient during cluster formation, when the MANET switches from the dormant to the running state plus the cost of urgent data delivery at the WSN layer (until the inter-network routing has been set up).

To obtain analytical results about our power management strategy, we analytically model WHOO integration protocols by using some simple formulas. First of all, let F be the number of 2-hop neighborhood MANET nodes around the cluster-head, D be the number of MANET nodes selected by the cluster formation protocol and A be the subset of the cluster MANET nodes that participate in advertisement, with $A \subseteq D \subseteq F$. In addition, the delays between two consecutive advertisements/lazy discoveries are for the ith and jth member of A and D, τ_i^A and τ_j^D seconds, respectively, while each lazy discovery/advertisement transmission reaches, respectively, an average of S_i^A and S_j^D WSN sensor nodes. Consequently, we define MP as:

$$MP = \sum_{i=1}^{|A|} \frac{1}{\tau_i^A} S_i^A + \sum_{j=1}^{|D|} \frac{1}{\tau_j^D} S_j^D$$

where the first term accounts for advertisement packets received by sensor nodes, and the second term accounts for lazy discovery packets, assuming that MANET nodes are stationary and all discoveries are lazy.

MW cost, on the other hand, includes the number of full discovery messages initially exchanged by the F MANET nodes to obtain gradient data from the WSN, with each message from the kth node reaching an average of S_k^F WSN sensor nodes, plus the urgent data messages exchanged at the WSN layer until inter-network routing is active. Let us remark that typically $D \subseteq F$, i.e. the set of MANET nodes involved in the initial full discovery is a superset of the set of nodes that send lazy discovery while the MANET cluster is in the running state because MANET nodes unable to reach sensor nodes with a good gradient are evicted during cluster formation. Moreover, we define PL the number of sensor nodes that urgent data would traverse at the WSN layer without our cross-network routing support, and *lost occurrences* (LO) the number of lost inter-routing occurrences, namely the number of urgent packets that the MANET layer failed to route because of not being in the running state. Hence, we define *MW cost* as:

$$\sum_{k=1}^{|F|} 2S_k^F + 2 \cdot LO \cdot PL - 2 \cdot LO$$

The first term accounts for the full discovery packets sent by MANET nodes and related replies by sensor nodes and the second term accounts for the lost chance of routing an urgent data packet using the MANET; because each WSN sensor node sends and receives the urgent data packet we count them twice. Finally, we subtract two times LO because, whether the cross-network routing is already active or not, at least the WSN exit and entry points, respectively, send and receive the urgent message.

Then, we apply the two-competitive strategy to compute the optimal timeout. We recall that the optimal strategy is to pay the repeating cost (MP) until its cumulative cost equals the one-time cost (MW). The time that satisfies this condition is expressed as the ratio between the initial one-time *MW cost* and the continuous *MP cost*:

$$t = \frac{MW}{MP} = \frac{\sum_{k=1}^{|F|} 2S_k^F + 2 \cdot LO \cdot PL - 2 \cdot LO}{\sum_{i=1}^{|A|} \frac{1}{\tau_i^A} S_i^A + \sum_{j=1}^{|D|} \frac{1}{\tau_j^D} S_j^D}$$

Here S_i^A, S_j^D, S_k^F are known at runtime and all other values (i.e., A, D, F, τ_i^A, τ_j^D) are known and can be controlled by the cluster-head. The value of LO is unknown and typically depends on the specific deployment; it can be estimated by dividing the urgent data frequency by the time that it takes to form a cluster; after the identification it can be considered a constant. PL is unknown too, but even for mid-sized MANET clusters and WSNs, it is easy to see that it quickly becomes not so significant as the other parameters grow; hence, it can be considered a constant value. Thus, we can control the integration cost independently of the urgent data traffic pattern on the WSN, by properly configuring MP and MW parameters, in other words, by limiting the MANET cluster size and by modifying discoveries/advertisements time intervals.

4.5 DISCOVERY AND ADVERTISEMENT PROTOCOL IMPLEMENTATION INSIGHTS

The implementation of discovery and advertisement protocols required modifying existing CTP packets and implementation modules [132]; this section provides some more details about our WHOO extensions to CTP.

WHOO discovery relies tightly on the standard CTP routing mechanism but it adds some adjustments to avoid excessive and useless additional load. In CTP, when sensor nodes receive a gradient cost query, they start to broadcast repeatedly their routing information with an exponentially decaying frequency: this technique allows CTP to build a collection tree, and to take advantage of new and better routes quickly by using the same mechanism in WHOO would have caused unnecessary control overhead and would have wasted battery energy of sensor nodes; hence, we use a reserved bit from the option field of CTP routing header, called `DNR` ("do not reset"), to signal the sensor node to broadcast routing information (gradient cost) only once. In addition, we exploit another reserved bit from the option field of CTP routing header, called L ("lazy") to realize lazy discoveries: a discovery request with L bit zero is full and requires a response, otherwise it is a lazy request. Let us also remark that a MANET node discovery request potentially triggers immediate responses from many sensor nodes simultaneously, thus inducing possible MAC layer failures; to obviate to this problem, especially for highly dense WSNs, each sensor node schedules its response randomly in a given period to lower the probability of data collisions.

As far as advertisement is concerned, WHOO adds a new control packet to CTP with two main fields: the ValidTime field that reports interval of node reachability, after which a sensor node receiving the advertisement can consider the advertising MANET node unreachable, and the Cost field that contains the advertised gradient cost. The delay between two consecutive advertisements should be shorter than ValidTime to avoid undesirable bouncing effects of the MANET–WSN integration: we set the delay to one-third of ValidTime to allow a robust advertisement mechanism, even in the presence of losses of single control packets. About Cost, when it is lower than the one of its parent in the data collection tree, the sensor node forwards the urgent data to the advertising MANET node. Another relevant aspect is the management of urgent data forwarding during the transition from the running to the dormant state. In fact, when MANET–WSN integration has been switched-off, MANET nodes are no longer willing to accept urgent packets; hence, we add a bit called POSITIVE in the advertisement packet header that, when 0, signals a negative advertisement that immediately stops urgent data forwarding from the WSN to the MANET layers.

4.6 EXPERIMENTAL RESULTS

This section presents results coming from different sources and methods; we evaluated WHOO partly analytically, partly on a large scale simulated deployment, and partly on a small scale real deployment. In the first part of this section, we describe performance figures about MANET–WSN integration energy cost in terms of switch-off time evaluation collected with our analytical model. However, since our analytic model has been designed to evaluate the MANET–WSN integration cost, it does not cover other aspects such as routing latency, protocol overhead and node mobility impact. Hence, the second part of this section is devoted to performance figures collected on a large scale the field by running a prorotype of WHOO on a network simulator. The last part includes results from a real testbed.

4.6.1 Analytical Evaluation of Adaptive Switch-off Performance

We show two main sets of analytical results: first, we explore how the MANET switch-off time depends on the number of MANET and WSN nodes, and secondly we analyze the switch-off time in dependency from the lazy discovery delay. To have a better idea of how the optimal switch-off time varies, we simplify the general formula so as to be more easily evaluated, with the goal of deriving general considerations about sensitivity to various parameters.

$$t = \frac{MW}{MP} = \frac{\sum_{k=1}^{|F|} 2S_k^F + 2 \cdot LO \cdot PL - 2 \cdot LO}{\sum_{i=1}^{|A|} \frac{1}{\tau_i^A} S_i^A + \sum_{j=1}^{|D|} \frac{1}{\tau_j^D} S_j^D}$$

With the assumption that the MANET node that starts cluster formation is homogeneously surrounded by N MANET nodes in its 2-hop broadcast neighborhood (i.e., $|F| = N$), half of them become members of the MANET cluster (i.e. $|D| = \frac{N}{2}$). In addition, we assume that both the interval between discoveries and the number of sensor nodes that every MANET node can communicate with (i.e., τ_D and S previously defined) have almost the same values for all MANET nodes. Let us also assume that only one node sends advertisement packets ($|A| = 1$): this assumption reflects the typical situation where one MANET node intercepts urgent data traffic from a sensor node that forwards it on behalf of a whole collection tree branch. Finally, we assume that the MANET cluster misses the opportunity to route just one urgent data packet ($|LO| = 1$). That can simplify the timeout formula:

$$t = \frac{2S \cdot N + 2PL - 2}{\frac{1}{\tau_A} S + \frac{1}{\tau_D} S \frac{N}{2}}$$

We intend to evaluate how the MANET cluster size influences the optimal switch-off time: Figure 4.4 reports the values of the switch-off time, given $\tau_A = 5s$, $\tau_D = 10s$, $PL = 10$ for a growing number of MANET nodes and for different values of S. Figure 4.4 shows that if the MANET cluster size grows, the timeout goes toward its limit:

$$\lim_{n \to +\infty} \frac{2S \cdot N + 2PL - 2}{\frac{1}{\tau_A} S + \frac{1}{\tau_D} S \frac{N}{2}} = 4\tau_D$$

Thus, increasing the MANET cluster size does not necessarily shrink the optimal switch-off time. Let us remark that obviously the timeout is guaranteed to be positive because $PL > 0$, but by using very aggressive parameters (e.g. extremely high discovery frequency), it becomes arbitrarily small, thus the MANET–WSN integration ceases to be beneficial. Nonetheless, such values are unviable for most deployments, but can easily be detected so as to exclude MANET–WSN integration.

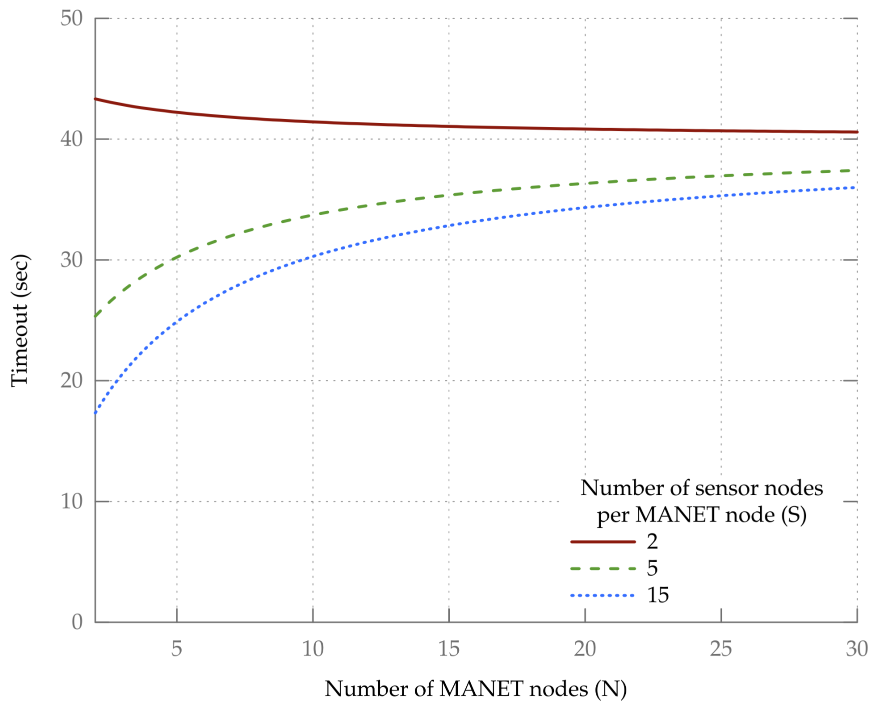


Figure 4.4: MANET state switch timeout as function of MANET size (N) and the number of reachable sensor nodes per MANET node (S).

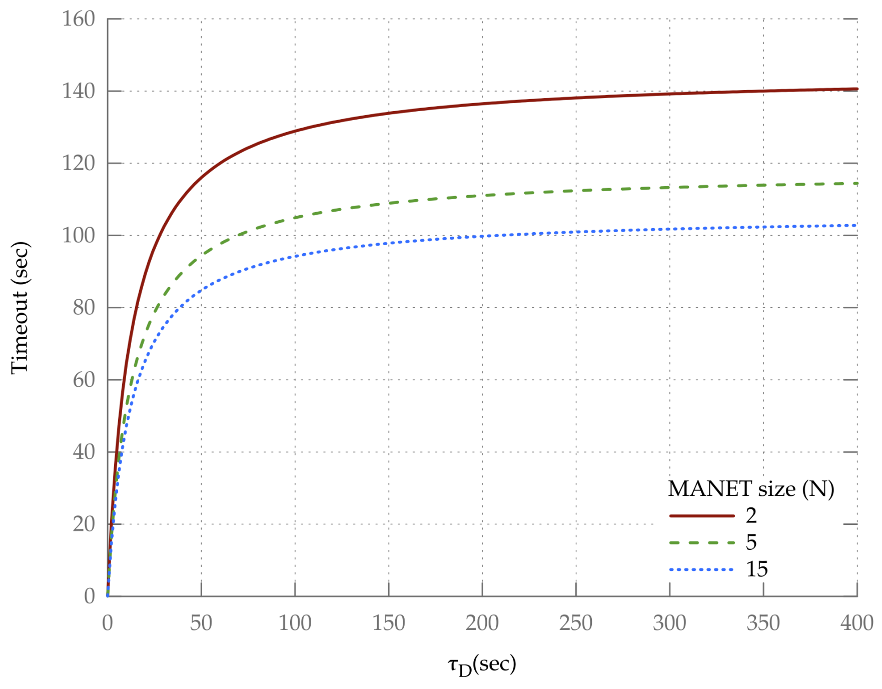


Figure 4.5: MANET state switch timeout as function of lazy discovery delay and MANET size (N).

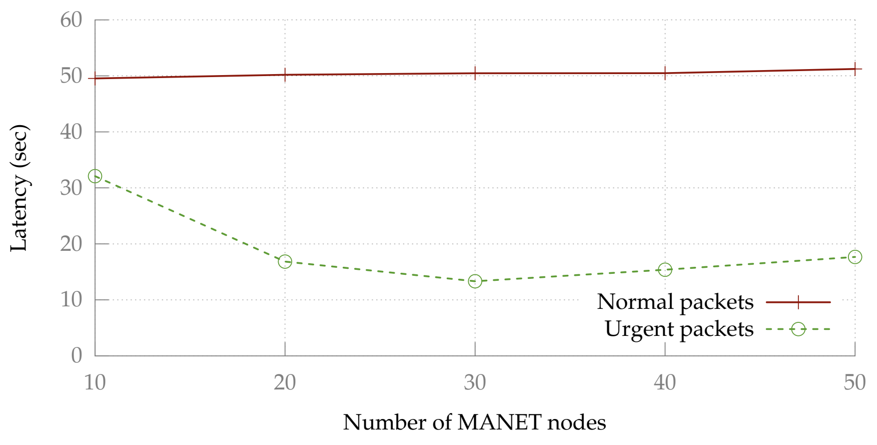
Our first results show the influence of τ_D over optimal switch-off; hence, it is interesting to detail how τ_D influences the switch-off timeout for different values of N and for a WHOO system with $\tau_A = 5s$, $S = 5$ and $PL = 10$. Figure 4.5 shows that the optimal switch-off time grows rapidly as τ_D grows for low values of τ_D and that it is upper bounded by:

$$\lim_{\tau_D \to +\inf} \frac{2S \cdot N + 2PL - 2}{\frac{1}{\tau_A} S + \frac{1}{\tau_D} S \frac{N}{2}} = \frac{\tau_A (S \cdot N + PL - 1)}{S}$$

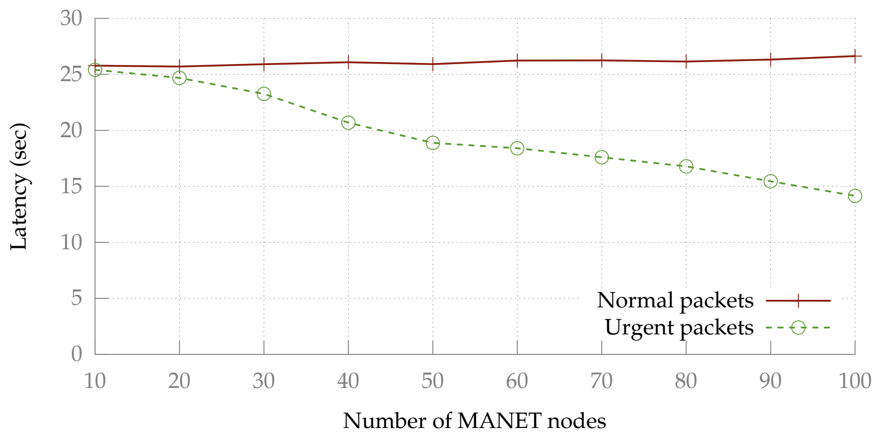
The figure shows that the timeout value reaches a plateau for high values of τ_D , while for lower values, especially before the 50s mark, a small increment of τ_D causes a relevant increment of the timeout. That clearly suggests that using the highest possible value of τ_D to maximize the time duration of the MANET–WSN interconnection, especially when τ_D is too small to achieve the plateau range. Moreover, a MANET node that moves at high speed would typically use low τ_D values to discover fast new potential sensor nodes; so, Figure 4.5 shows also the sensitivity of WHOO to MANET node speed and justifies more formally the intuitive notion that MANET–WSN integration would cost less when MANETs are relatively static.

4.6.2 Simulated Evaluation of Network Performance

To validate our proposal on a wide scale, instead, we developed and run extensive simulations: we have originally ported CTP to the QualNet network simulator and we have implemented our protocol on top of it. In the adopted simulation environment, we made the sensor nodes use the IEEE 802.15.4 physical layer and a Carrier Sense Multiple Access (CSMA) MAC protocol, thus simulating a realistic communication testbed, similar to what used by many widely adopted real-world sensor nodes, such as TelosB and MICAz . Moreover, we stress another point about this performance result: real WSN deployments often duty cycle the radio, to save energy by modifying the ratio between the time intervals with the radio on and off, because duty cycle techniques can grant higher energy savings at the cost of higher data delivery latencies [153]. We have tested WHOO with a duty-cycled version of CTP, simulating the duty-cycling described in [154] (that is representative of a large class of asynchronous MAC protocols very suitable for opportunistic integration WSN communication such as B-MAC and X-MAC [155, 156]), simulating the delays experienced by a TelosB sensor node running on a 2.5% duty cycle (i.e., that keeps the radio interface active for the 2.5% of its running time). This duty cycling, much less aggressive than what usually employed (1% or less [157]), has been chosen as a worst-case scenario not to favor too much our MANET-enabled urgent data delivery. Finally, MANET nodes use the IEEE 802.11b physical and MAC layers.



(a) Street



(b) Square

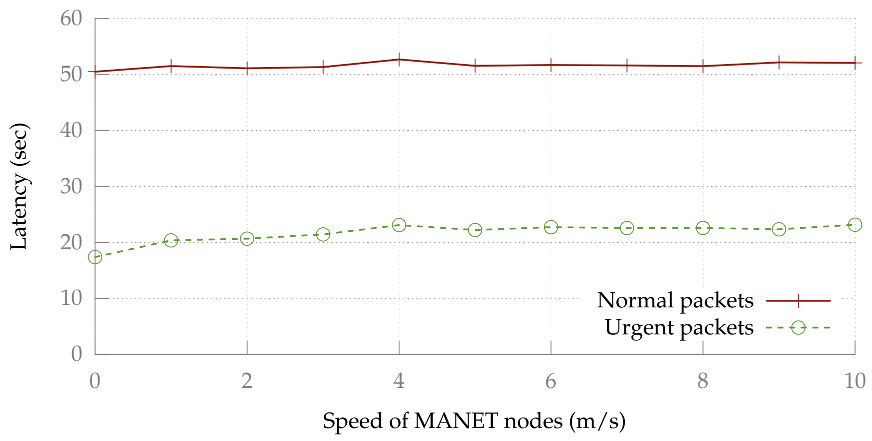
Figure 4.6: WHOO packet latency vs. MANET nodes density.

To evaluate the performance of our system in a smart city environment, in QualNet we have modeled two scenarios. The first one is a 1 km-long and 10 m-wide street, monitored by 50 sensor nodes, 20m apart from each other; the sensor node at the beginning of the street acts as the tree root, while the one at the end of the street alternately generates one normal data packet and one urgent data packet, with a period of 3s. The second one is a 1km-long and 1km-wide square, monitored by 200 nodes positioned in a star-like shape with eight branches, each one composed by 25 sensor nodes 20m apart from each other; the sensor node at the center of the star is at the center of the square and acts as the tree root, while the ones at the end of the branches alternately generate one normal data packet and one urgent data packet, with a period of 3s.

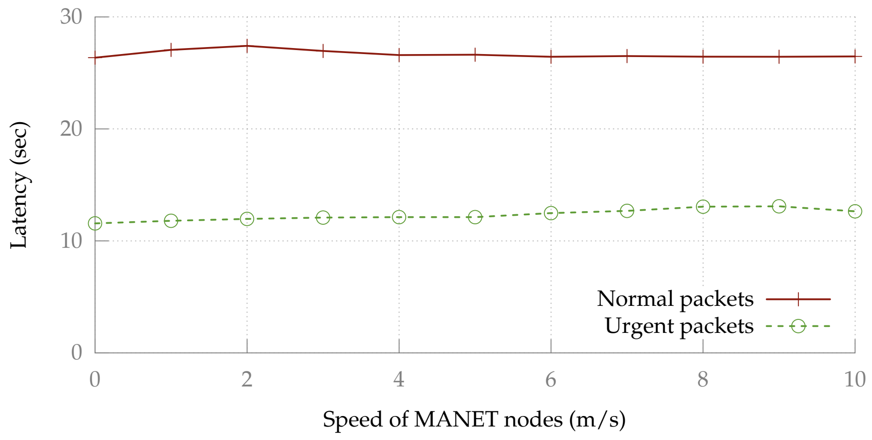
Our first evaluation focus on packet latencies. In particular, we have observed the impact of MANET node density on packet delivery e latency. We have simulated the reference scenarios by constantly increasing the number of randomly placed MANET nodes and we have repeated each test for 30 runs; the collected error margins were always under 5% of the reported average. Figure 4.6a shows obtained experimental results for the street scenario: as expected, our solution vastly reduces delivery latency of urgent packets in every tested case; in addition, results show that there is a minimum for MANET node density that gives the best latency improvements (in our scenario, for 30 MANET nodes). For lower MANET densities, latencies are not so good because MANET clusters do not cover all the WSN areas, thus forcing data packets to hop over slower WSN hops. When there are more than 30 nodes, latency goes up again due to the increased traffic induced by advertising and discovery packets, which cause more packet collisions. Experimental results for the square scenario (Figure 4.6b) show that it is necessary a higher number of MANET nodes to have a relevant decrease of latencies for urgent packets: that is expected, because MANET nodes are scattered in a much larger area and can easily be placed in areas that are disconnected from the WSN; however, let us stress that 100 MANET nodes per km^2 is a very low density, hence the performances of our proposed approach are good even with a low MANET density. Finally, it is worth noting that, since the basic CTP protocol does not include any form of traffic differentiation, the reported results for normal packets also exemplify the performance of urgent packets in a WSN-only scenario where MANET nodes are not available to help in speeding up the routing function.

Our second set of simulated experimental results assesses the impact of MANET node mobility on packet latency. We used the same scenarios of the previous evaluation and kept the number of MANET nodes fixed at 40 for the street scenario and 100 for the square scenario. We made MANET nodes move randomly over the simulated areas adopting the random waypoint mobility model at various speeds (from 0 m/s to 10 m/s). Figure 4.7 reports the related evaluation results. Numerical simulation shows that for both scenarios, as MANET node speed increases, latency remains substantially constant, with a small growth for lower speeds. This result shows that speed per se has a limited impact on packet latency, because a successful routing over a MANET cluster takes much less time than the time needed by speed to break apart the cluster itself.

Another important indicator is the packet delivery ratio, namely the number of packets successfully dispatched from data source to root. Figure 4.8 reports packet delivery ratio under the same conditions of the previous evaluation and with MANET nodes moving at growing speeds. As expected, CTP is very reliable and always delivers more than 98% of packets. Our solution achieves a packet delivery ratio comparable to CTP when MANET nodes are not moving, while the ratio sensibly decreases as MANET nodes become more and more mobile. In particular in the street scenario, it delivers more than 80% of urgent packets when MANET nodes move at 1 m/s, and drops to about

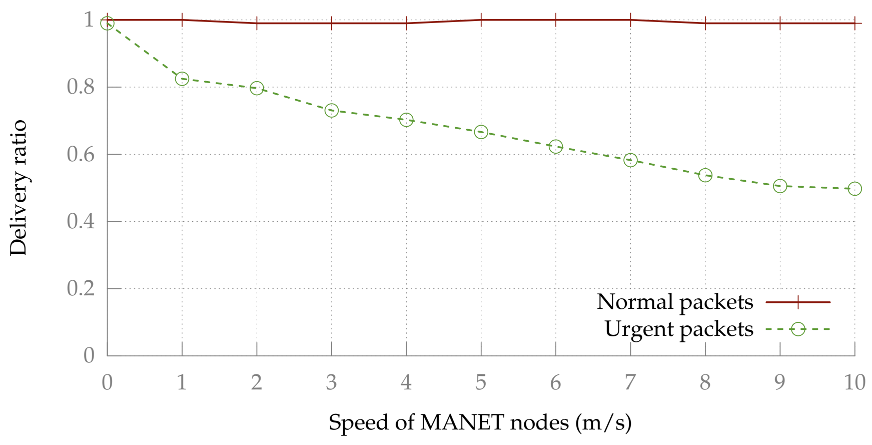


(a) Street

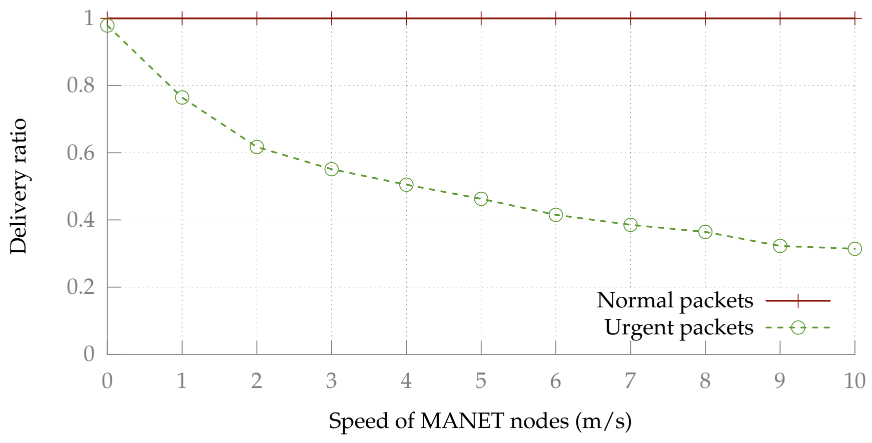


(b) Square

Figure 4.7: WHOO packet latency vs. mobile nodes speed.



(a) Street



(b) Square

Figure 4.8: WHOO Delivery ratio vs. mobile nodes speed.

50% when MANET e nodes move at 10 m/s, whereas in the square scenario it delivers slightly less than 80% packets at 1 m/s and 30% at 10 m/s.

That trend is expected, because in the i square scenario MANET nodes have much more movement freedom, thus MANET clusters can break apart more easily compared to the street scenario. However, let us note that the delivery ratio of our solution can be boosted by repeated sending of urgent packets. For example, when MANET nodes move at 1 m/s, a urgent packet has about 80% probability to be successfully routed to the collection tree root. If the node sends the packet twice, the probability of successful routing raises to 96%; another repeated send operation achieves the success probability of 99%. Thus, we claim that our solution can achieve a good trade-off between routing speed and delivery ratio, by significantly improving data collection in scenarios that do not rely on a fixed infrastructure.

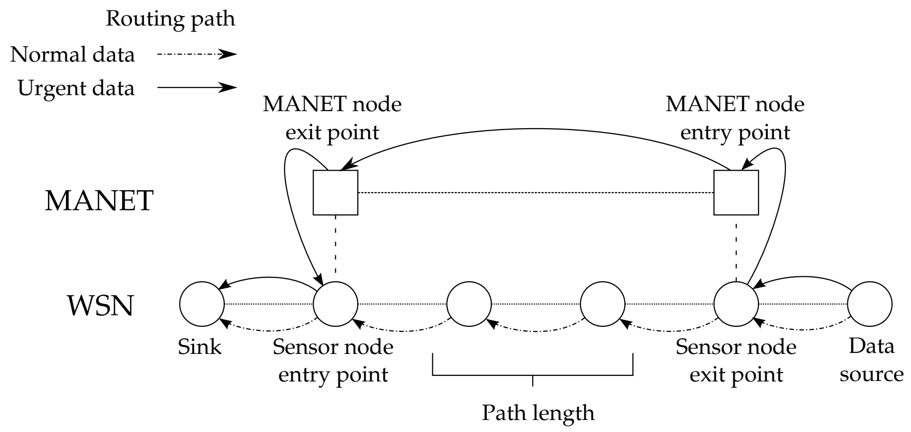


Figure 4.9: Setup of the experimental testbed to evaluate WHOO performance.

4.6.3 Real World Deployment Evaluation

To thoroughly evaluate our proposal, we conclude the evaluation of our proposal by presenting three three crucial performance results collected on our real WHOO prototype: the first set of results shows urgent data packet latency with and without our cross-network routing support to evaluate the speed improvement granted by WHOO; the second one, under the same working conditions, assesses WHOO protocol overhead and the third one evaluates the packet delivery ratio and WHOO resiliency to MANET node mobility. To collect these performance measurements, we used our WHOO implementation by deploying it in the heterogeneous wireless network at our campus. On the WSN side, we developed WHOO components for TinyOS 2.1.1 and installed the software on our WSN testbed, composed of TelosB sensor nodes manufactured by MEMSIC [73, 125]. On the MANET side, we developed WHOO in Java, and deployed it on Linux laptops, each one connected to a TelosB node that enables communications with WSN nodes.

From now on, we refer to Figure 4.9 for the reference deployment of our practical tests. We set up a chain of sensor nodes of variable length, the first node acting as root and the last one as data generator and used two MANET nodes as the entry and exit points. In the following, we refer to the total number of sensor nodes in the chain as the *total chain length*, while the number of sensor nodes between the WSN entry point and the WSN exit point is the already defined PL parameter (Section 4.4).

Because we intend to get faster delivery for urgent data, our first set of experiments assesses how much faster the network can route urgent data while exploiting opportunistically a MANET compared with the plain use of the WSN. In particular, we measure the time for a data packet generated by the last node to reach the sink by using either CTP only or CTP with WHOO. The data source sensor node generates 200 messages alternately: one normal and one urgent CTP data

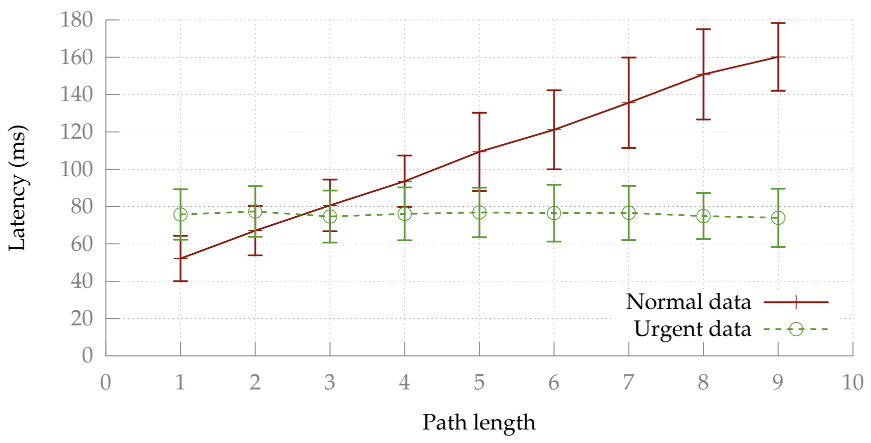


Figure 4.10: Latency for normal and urgent data when using WHOO for different PLs.

packet, each one with a 20-byte long payload and sent every 2s. The radio interface of sensor nodes runs on a 15% duty cycle. Experimental results measure latencies by using different total chain lengths and by making the MANET cover each time the whole WSN except for the root and data generator nodes; Figure 4.10 shows collected results, as average values over 50 runs and their 90% confidence interval (vertical bars), as a function of the PL. We can observe that WHOO becomes convenient when the PL is either 3 or more, i.e. when WHOO allows to avoid CTP routing on at least three sensor nodes.

Our second set of experimental results on a real testbed estimates the WSN–MANET integration cost by measuring the overhead due to WHOO integration protocols (discovery, lazy discovery and advertising ones). We use the same chain topology of previous tests with the total chain length fixed to 12 nodes and by progressively increasing the PL. The data generation rate is the same as that of our previous experiment, but with 400 messages instead of 200. For each run, we reset the WSN, wait for 90 s to let CTP routes stabilize and then we again start the data source node and wait for it to send all messages. Figure 8 shows collected results: we derive the total traffic as the sum of sent and received messages on each node by classifying and dividing messages as follows: CTP data traffic, CTP routing traffic, WHOO advertisements, WHOO discoveries and WHOO discovery replies.

We expect a low overhead and, in fact, WHOO introduces a small and fixed overhead cost independently of the PL. Even when the PL is only 1 WHOO slightly relieves sensor nodes from traffic handling (columns 0 and 1), but as soon the PL increases and WHOO skips more and more hops on the WSN (and related CTP data packet transmissions), the benefits of inter-routing become stronger and much more evident. These experimental results can also evaluate the ratio between the number of saved CTP messages and of additional WHOO (overhead) messages; let us call O the WHOO overhead, T the total

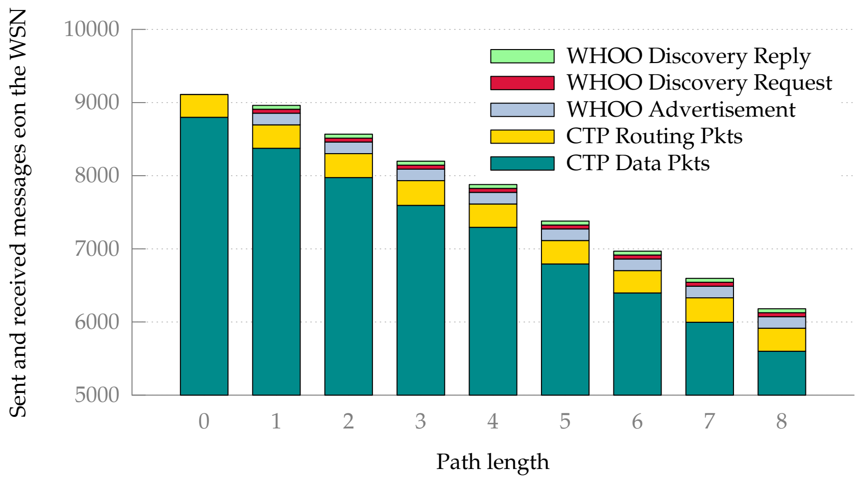


Figure 4.11: Overhead when using WHOO for different PLs.

traffic without using WHOO and W the total traffic using WHOO (subtracting the WHOO overhead): the ratio is $O/(T-W)$. For example, when the value of PL is 1, this ratio is $266/(9104-8696) \approx 1/1.5$; in other words, for each WHOO overhead message, the system saves 1.5 messages as whole. When the PL is 8, on the other hand, the ratio is $266/(9104-5915) \approx 1/12$, and thus in this case WHOO reduces the WSN traffic load by more than an order of magnitude.

Finally, our last set of performance results measures how mobility influences the packet delivery ratio between a fixed WSN sensor node and a mobile MANET node. As shown by some previous works in the field, the general problem is complex due to the presence of asymmetries in communication links of WSNs both under static and dynamic conditions [158–160]; our goal here is to assess the reliability of WHOO protocols, even with MANET node mobility, and to give a qualitative estimation of WHOO behavior under those dynamic conditions.

We observe that in our reference scenario, namely sparse MANET nodes roaming and immersed in a dense WSN, communication between the WSN and the MANET is essentially asymmetric; in fact, it is probable that a WSN exit point can communicate only with one MANET entry point; in contrast, a MANET exit point can dispatch urgent data to a specific WSN node (the one that is advertising the lowest cost), but if it fails, since the WSN is dense, it is probable that there are other WSN nodes with a near-optimal cost nearby that the MANET exit point can communicate with; hence, we are more interested in the results about the WSN-to-MANET scenario than in the MANET-to-WSN ones. To evaluate performances about communication from the WSN to the MANET, we use a fixed WSN sensor node and a mobile MANET node. We have programmed the fixed sensor node so that when it detects a MANET node advertisement packet, it start generating one urgent data packet per second and tries to forward it to the MANET node; if it fails, it resends it again after a 30ms delay, and then it gives up. Due to the

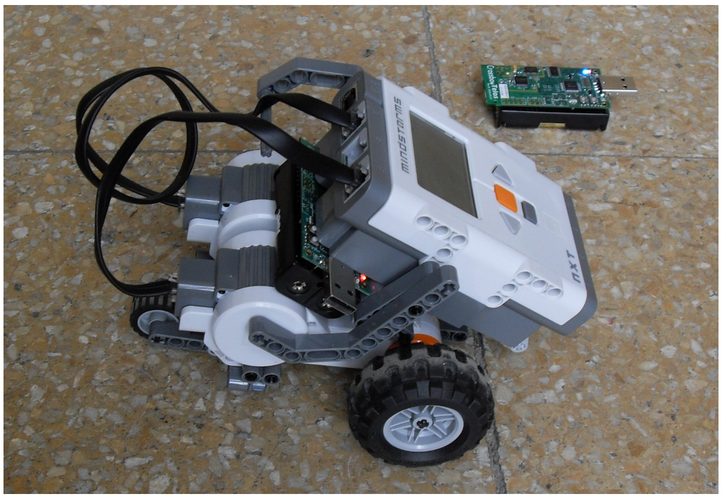


Figure 4.12: Robot carrying a TelosB node. On the background, a fixed sensor node.

intrinsic characteristic of the wireless medium, this communication could fail if the link is too weak or noisy. To minimize the packet losses, we added a threshold mechanism that allows the fixed node to send a packet if and only if the Link Quality Indicator (LQI), that is a synthetic indicator of the reliability of a IEEE 802.15.4 link, is higher than 95, a value that we chose empirically [161]. We deployed the sensor node at the center of a 90m long corridor at our campus; we put the mobile node on a small robot capable of moving at different speeds, from 0.2 up to 1m/s, which is a good approximation of the speed of a walking person (Figure 4.12) [162], and we made it to send one advertisement packet per second, each one with `ValidTime` of 3s. Let us stress that this specific setup makes communication a hard nut, mainly for two reasons: first, because the robot moves continuously and traverses areas with both high and low packet reception rate; secondly, both the WSN and the MANET being extremely sparse, because we have only one node for both networks, the capacity and possibility of the WSN node to route urgent data into the MANET layer are strongly reduced with respect to a more dense MANET scenario.

Figure 4.13 shows the number of packet deliveries successfully completed and failed at the mobile node using our LQI-enabled threshold-based sending function plotted for different speeds, from 0.2 to 1m/s. When the speed increases, the number of successfully delivered packets decreases because the mobile node spends less time in the communication range of the fixed node.

To further evidence the relevance of the above result, Figure 4.14 plots Packet Reception Ratio (PRR), i.e., the ratio of packet successfully received over total sent packets, of packets successfully delivered at the first and second try, and the percentage of failed deliveries. A first important finding is that PRR is almost constant regardless of mobile node speed. The average PRR (averaged over all considered

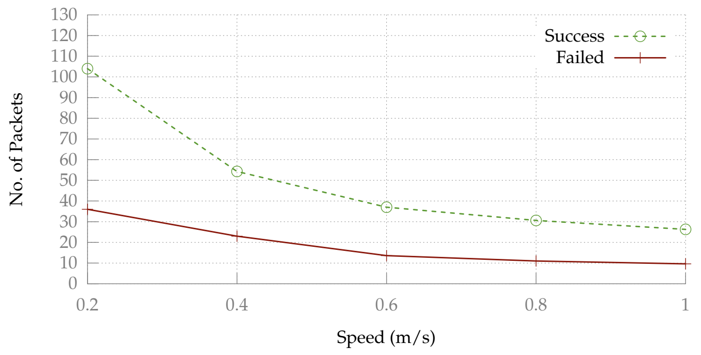


Figure 4.13: Delivery performance (absolute values) for a mobile node at different speeds.

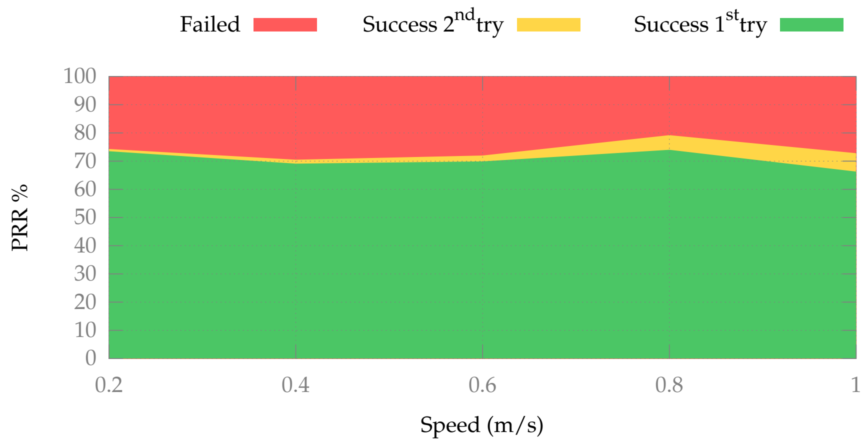


Figure 4.14: PRR for a mobile node, moving at different speeds, that tries to communicate with a fixed node, using incoming LQI threshold.

speeds) is 70% and reaches 80%. In addition, focusing on particular PRR values obtained for each speed, as mobile node speed increases, the portion of packets delivered at the second try increases too. This is a consequence of the fact that LQI filtering reduces communication in grey areas that need to repeat a send for a successful delivery, but such second tries cannot be completely eliminated; hence as speed increases and the total number of sent packets decreases, second tries become proportionally more relevant.

4.7 RELATED WORKS

The use of multi-radio devices in a WSN scenario and the integration of WSN and mobile nodes, have already been explored with reference to specific target techniques, without considering the MANET–WSN as an integrated system. In the following, we sketch a selection of research efforts in three main technical areas, presented in order of growing similarity with our research: tiered radio-heterogeneous sensor networks, data harvesting and WSN coordination with mobile nodes and MANET–WSN integration. Some of these works have been already presented in Section 3.3; here, we describe them with a focused comparison to WHOO, rather than to the logical model of Chapter 3.

The usage of heterogeneous radio interfaces to optimize and improve WSN has been already proposed in the literature: Yarvis *et al.* [163] demonstrated that using a modest number of reliable long-range link vastly improves sensor network delivery ratio and battery lifetime. A practical evidence of that effect is the ExScal project that deployed more than a thousand sensor nodes with about 200 high-powered dual-radio sensor nodes to provide an always-on high-speed network overlay usable by all other resource-constrained sensor nodes [127]. Siphon is similar to ExScal but exploits multi-radio sensor nodes to provide an on-demand traffic management service that relieves congested traffic [131]. In general, these works assume that a subset, but relevant, number of sensor nodes provides both a low-power radio and an IEEE 802.11 interface: this assumption is reasonable in some scnearios, but it is usually not viable for real large sensor nodes deployments, given the higher cost of dual-radio nodes adtheir short battery lifetime.

Another hot research area is the one that explored the use of mobile nodes as data harvesters and as WSN gateways toward the fixed Internet. Chakrabarti *et al.* [164] proposes usage of mobile nodes whose path can be predicted as data sinks to lower the power consumption at the expense of higher latencies, and Wang *et al.* [165] give interesting theoretical results about the use of mobile relays by showing that one mobile relay that stores and forwards gathered data to a data sink can improve a sensor network lifetime up to a factor of 4. The mWSN architecture proposes a tiered architecture in which sensor nodes form a cluster around the expected position of mobile nodes that act as mobile data sinks able to store and forward sensed data to well-known Internet gateways [166], while a survey of hierarchical multi-tiered architectures for WSN enhanced by mobile nodes can be found in [167]. All these works stress energy-saving aspects, but energy savings come at the cost of higher data delivery latencies, which becomes critical when dealing with urgent data and MANET node mobility.

4.8 CHAPTER CONCLUSIONS

This chapter proposes a novel solution for opportunistic cross-network data collection for urgent data, based on the integration of MANETs and WSNs outlined in Chapter 3. Through analytical and practical experimental results, we fully assessed the qualities of our proposal by evaluating both its benefits and its costs. Analytical results show that our system can give guarantees about the total integration cost, according to strong energy-saving principles that have driven WHOO development, while the performance results collected on the field demonstrate that WHOO significantly boosts data collection performances and can be easily integrated into existing WSN deployments to lower the urgent data delivery time. All results show the viability of our proposal in terms of caused overhead and confirm that the opportunity of employing mobile MANET nodes equipped with low power interfaces can valuably contribute to increase data collection performance.

5 CENTRALIZED SENSOR NETWORK MANAGEMENT VIA SMARTPHONE INTEGRATION

The previous chapter showed an instance of completely distributed Pervasive Sensing system that does not rely on any fixed infrastructure. However, if a fixed infrastructure is available then it can be exploited to coordinate access of mobile nodes to WSN, with the final goal of minimizing WSN communications and energy consumption. The main advantage of a centralized approach is that knowledge about the whole network status is available, making it easier to control in a optimal way interactions between mobile nodes and sensor nodes. Controlling devices in the mobile-infrastructure tier from the fixed-infrastructure tier is a challenge *per se*. A large group of standardization bodies has recently defined the IP Multimedia Subsystem (IMS) [128]. IMS defines an overlay architecture for session control in all-IP next generation networks to obtain openness and interoperability with an application-layer approach, based on the Session Initiation Protocol (SIP). A few research activities have already proposed to integrate IMS and WSN, for instance to disseminate data collected from a WSN but, to the best of our knowledge, none of them considered the possibility to use IMS to deliver highly innovative, interoperable, and ready-to-market energy-efficient solutions for WSN data harvesting.

This chapter fills that gap by proposing a novel solution that integrates IMS in the logical model proposed in Chapter 3 with three original core properties. First, it exploits mobile devices to enable decentralized sensed data harvesting and dynamic control/change of sensor node communication strategies (such as changing radio wake-up periods). To save further energy, second, it centrally coordinates mobile devices to avoid unnecessary communications with already harvested sensors by disseminating lists of harvested sensor nodes and harvester locations. Third, it is fully compliant with IMS: the diffusion of sensor status and device location data effectively exploits, also from the energy point of view, the recently standardized IMS-based Presence Service (PS). Our solution prototype integrates with widely accepted, internationally recognized, and open-source IMS/WSN implementation platforms and tools, such as OpenIMSCore, OpenSIPS, UCT Client, and TinyOS components [83, 168–170].

5.1 MOTIVATIONS AND BACKGROUND

This section first defines the main requirements and guidelines for energy-efficient sensor data harvesting driven by fixed infrastructure. Next, it briefly introduces the IMS architecture, the IMS-based PS, and main elements of low-power WSN media access, which are the crucial technical elements to fully understand our original proposal.

5.1.1 Application Scenario and Design Guidelines

Our target scenario is an environmental monitoring application involving several sensor nodes sparsely disseminated in a smart city. Sparse WSN deployments are very interesting because single sensors can usually provide with measurements that are valid for an area much wider than their communication range. For this reason, in normal (dense) WSNs, there is the need to add wireless (sensor) nodes just to route collected data to a gateway, thus increasing WSN deployment/maintenance costs. We claim the relevance of novel opportunistic solutions where user mobile devices, such as smartphones, could act as sensor data harvesters by (i) directly querying sensor nodes, (ii) collecting recorded data, and (iii) especially publishing them to the Internet via their usually available long-range wireless interfaces. However, to effectively save energy at sensor nodes, it is overwhelming important to intelligently coordinate and to adaptively tune data harvesting operations.

Let us introduce some of the benefits of our energy-efficient approach with a simple and practical example. Consider a monitoring service targeting air pollution, with sensor nodes deployed over different city areas, namely busy roads, boulevards, and parks. In addition, suppose that mobile devices have Internet access via a 4G network composed by WiFi hotspots, deployed along the roads and in parks, and by a Universal Mobile Telecommunications System (UMTS) infrastructure deployed over all city districts. The main goal of our energy-efficient harvesting solution is to save communication energy by automatically adapting to the specific conditions a sensor node operates. In fact, depending on the type of urban area, a sensor node is exposed to harvesters with different mobility patterns. First, sensor nodes deployed on busy roads and boulevards would be easily reachable from the early morning until evening, and almost completely isolated during the night. Second, depending on the area, data harvester speed may vary significantly. Third, in areas such as parks, mobility patterns could be time-dependent: for instance, in early morning the park is populated by fast moving joggers equipped with their audio-enabled smartphones, while during lunch times and in the evening periods potential harvesters are (rather static) users equipped with their laptops and PDAs. Another important aspect is sensor querying frequency. Each sensor node periodically samples and logs air pollution data; hence, querying it before it has collected an adequate number of samples is inefficient, especially during rush hours when there may be a high number of contacts between sensors and harvesters. Finally, it might happen that a sensor node remains isolated from any harvester for relatively long time spans; an effective solution for data harvesting should be aware of this and should switch off the sensor node radio for some time to save energy.

Existing works have shown the feasibility of distributed access to WSNs via mobile nodes [8, 124, 171]; however harvesting solutions managed in the mobile ecosystem by means of open session control standards, such as IMS, allows to support the execution of all distributed and possibly complex management operations to save energy at resource-limited sensor nodes (integrated communication management, harvester coordination, energy-saving strategy control, etc.).

Let us note that, from the sketched application scenario, we naturally adapt the design principles described in Section 3.1, followed in our energy-efficient data harvesting infrastructure: *adaptivity*, *location/communication awareness*, and *resource awareness* implemented as *proactivity*.

First, and most important, the data harvesting infrastructure should *adapt* to dynamically changing sensor communication strategies and to switch off wireless interfaces whenever possible. In particular, it should support a long-term power strategy to (de-)activate sensor communications for a long time span, such as during the night. It should also implement a short-term power strategy to control sensor node duty cycle, namely the ratio between the time interval of radio on and the time interval between two consecutive wake-ups (for instance, when harvesters move relatively fast), duty cycle should be raised to increase the probability of communication establishment. Figure 5.1 shows an example of combined long-/short-term power strategies.

Second, the data harvesting infrastructure should be aware of *location and communication status* to avoid useless transmissions. Mobile devices should periodically update their locations and attempt low-power communications for data harvesting only if there are sensors in their proximity. At the same time, the infrastructure should coor-

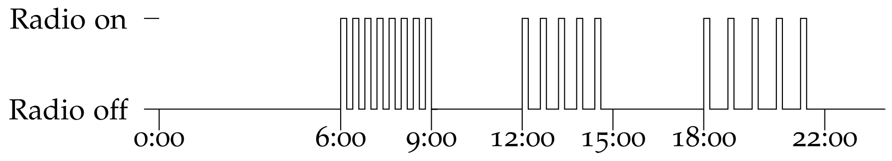


Figure 5.1: Example of differentiated duty cycles for a sensor node in a park: high when people jogging, low during lunch time and evening.

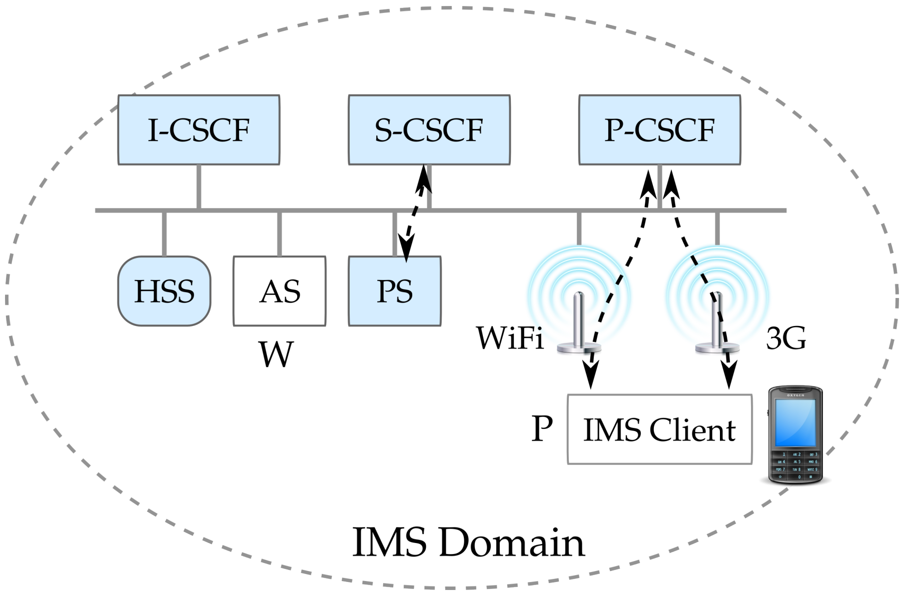


Figure 5.2: IMS distributed architecture. The core IMS components are:Home Subscriber Server (HSS); Proxy-Call Session Control Function (P-CSCF);Serving-Call Session Control Function (S-CSCF); Interrogating-Call Session Control Function (I-CSCF); and Application Server (AS) such as our energy-saving management function. The figure shows also IMS-based presence service components: Presence Service (PS); Watcher (we use the label W to tag an entity acting as watcher); and Presentity (we use the label P to tag an entity acting as presentity).

dinate data collection to avoid hammering already harvested sensors, with positive effects on sensor node lifetime. Third, the harvesting infrastructure should be *proactive*, to predict harvester movements and mobility patterns, thus properly anticipating power strategy changes. For instance, it should switch off sensors that will be isolated for a given time, and finely tune short-term power strategy by changing it before harvester mobility modifications.

5.2 BACKGROUND ABOUT IMS

The provisioning of interoperable session control over heterogeneous wireless infrastructures, including converged Future Network scenarios with WSNs for environmental monitoring, is still a challenging issue. To tackle this problem, the large group of standardization entities that has defined IMS and related services, has recently proposed IMS PS [128, 172].

Very briefly, the core IMS functional entities are as follows. The IMS Client controls session setup and media transport via SIP extensions specified by the IETF and 3GPP IMS-related standards; a unique HTTP-like Uniform Resource Identifier (URI), such as `sip:user@domain`, identifies any IMS Client. HSS stores authentication data and profiles for clients by using standard Data Base Management System (DBMS). AS allows the introduction of new IMS-based services; the IMS-based PS and our energy-saving management functions are realized as specific ASs. P-CSCF, I-CSCF, and S-CSCF are the core entities of IMS. They realize several main functions including: localizing mobile clients within IMS, establishing secure associations with Mobile Nodes (MNs), routing out/ingoing SIP messages, associating a client with its S-CSCF (as indicated within the client profile), and modifying the routing of specific types of SIP messages to ASs depending on filters/triggers specified by client profiles (IMS filter criteria). Figure 5.2 shows the deployment of all main IMS components presented in this and in the following subsections.

By specifically focusing on IMS PS, presence is a well-known service in the traditional Internet, widely used in applications such as Instant Messaging or multiparty games [173]. PS allows users and hardware/software components, called *presentities*, to communicate presence-related data to interested *watchers*. In particular, to receive PS publish/update messages from presentities, i.e., presence notifications, watchers subscribe to PS servers that act as intermediaries in any communication between presentities and watchers. The main IMS PS components are:

- IMS Clients, which may act as either presentities or watchers. More precisely, the PS user agent is the entity that provides PS information about a presentity (for the sake of presentation simplicity, in the following we use presentity to refer both).
- Presence Servers, which facilitate PS interactions. They accept and store PS subscriptions from watchers, and notify messages from presentities to registered watchers.

In addition, PS specifies protocols, such as the XML Configuration Access Protocol (XCAP), to tailor notification distribution; additional detail about IMS and its PS are out of the scope ofthis chapter, interested readers may refer to [128].

5.3 BACKGROUND ABOUT WSN LOW-POWER MEDIA ACCESS

In this section, we briefly analyze the impact of network communications on sensor nodes battery lifetime and techniques to minimize their detrimental effects.

Energy is the most important resource for sensor nodes and wireless interfaces are widely recognized to be one of the most relevant components that contribute to drain battery power. For example a widely adopted microcontroller installed on several WSN nodes draws:

1.1µA in idle state and up to 1320µA (when running at 4MHz, 2.2V) [174]. For the sake of comparison the Texas Instruments CC2420, one of the most common Radio Frequency (RF) transceiver for WSN nodes draws: less than 1µA when turned off, 426µA when in idle state, and around 20mA when active (receiving/transmitting) [175]. The relevant difference between those values show that a sensible management of wireless transmissions can vastly improve the sensor lifetime.

Techniques at the MAC level that aim at lowering power consumption of RF transceivers are called Low-Power Listening (LPL). Their main goal is to periodically turn on the radio receiver just long enough to detect whether there is a transmitter nearby and, if there is none, to turn the radio off as fast as possible. The ratio between the time spent listening for packets and the interval between two consecutive checks is called *duty cycle*. Reducing the duty cycle is the simplest way to reduce power consumption at the expenses of packet latencies. In fact, nodes using LPL infrequently check for incoming packet, thus a point-to-point transmission will take more time compared to the same transmission between two nodes that are not duty cycling their radio interface. In addition, a node that wants to send a packet to a node needs to make sure to send it while the receiver radio is on to wake it up. Each MAC protocol employs a different technique for waking up receivers, for example repeatedly sending the data packet, sending a train of specially crafted wake up packets, announce transmission with a long preamble, and so on.

Besides the use of duty cycling there are other techniques that MAC protocol may adopt to reduce the time the radio is switched on and achieve a more efficient LPL; among them some of the most prominent are:

- invalid packet shutdown: a receiver may turn off the radio as soon as detects that the destination address of the packet being sent is different from its own;
- early transmission completion: the receiver may acknowledge the correct reception of a packet, allowing the transmitter to terminate earlier the transmission;
- auto shutdown: if the radio does not send or receive packets for a certain amount of time the sensor node goes back to duty cycling mode.

Each MAC protocol may employ one or more of these and other techniques, based on the transceiver and traffic conditions that it targets, giving therefore different guarantees in terms of energy consumption and network performances, additional details are available in [154].

5.4 ENERGY-EFFICIENT IMS-BASED HARVESTING

This section describes our distributed architecture for energy-efficient data harvesting derived from the logical model proposed in Chapter 3, and its main components. Then, it shows how our solution exploits IMS-based session control protocols to coordinate and adapt the distributed data harvesting task. Finally, it sketches how data harvesters interact with sensor nodes to finely and locally tune sensor node energy-saving strategies.

5.4.1 Distributed architecture

To understand our solution for data harvesting, let us overview the main entities of our proposal (Figure 5.3). Apart from the IMS infrastructure and the PS, there are three main entities: *sensor nodes*, mobile nodes as *harvesters*, and the *coordinator*. Sensor nodes, at the physical sensing infrastructure tier, log their measurements and provide them on request. Harvesters, part of the mobile-infrastructure tier, are the mobile devices that act as presentities; they query sensor nodes, can elaborate collected data, publish sensed data to the IMS infrastructure, and receive notifications about data collected by other harvesters. Finally the coordinator, that is part of the fixed-infrastructure tier, is our core component: it acts both as a watcher that is notified about all the data published by the harvesters and as a presentity that uses publications to coordinate the harvesting task, by also notifying harvesters about data they are interested in.

With a closer view to details, *sensor nodes* have four components, derived from the logical model described in Section 3.2.1: *Sensor*, *Data Processor*, *Wireless Interface*, and *Low-power Strategy Manager*. *Sensor* is derived from the *OS/Sensing* component and wraps the access to hardware sensors. *Data Processor* receives data from Sensor and runs local processing algorithms to clean it or extract higher level data, then forwards it to the *Wireless Interface*, that realizes the *Upper-tier interface*, that sends it on request to harvesters. Finally, the *Low-power Strategy Manager* receives power management instructions from harvesters and enforces them on the whole sensor-node system, for example by accordingly throttling MAC duty cycling.

The *harvester* consists of four main components, derived from those described in Section 3.2.2: WSN *Access Interface*, IMS *Presentity/Watcher*, *Data Processor*, and WSN *Low-power Strategy Manager*. The WSN *Access Interface* realizes part of the *Lower-tier interface* connecting to sensor nodes to harvest data and to configure their communication strategy. The *Data Processor* has two main functions: it aggregates data harvested from different sparse sensor nodes so to possibly limit transmissions between the mobile node and the fixed infrastructure; in addition, it can locally elaborate sensed data by executing algo-

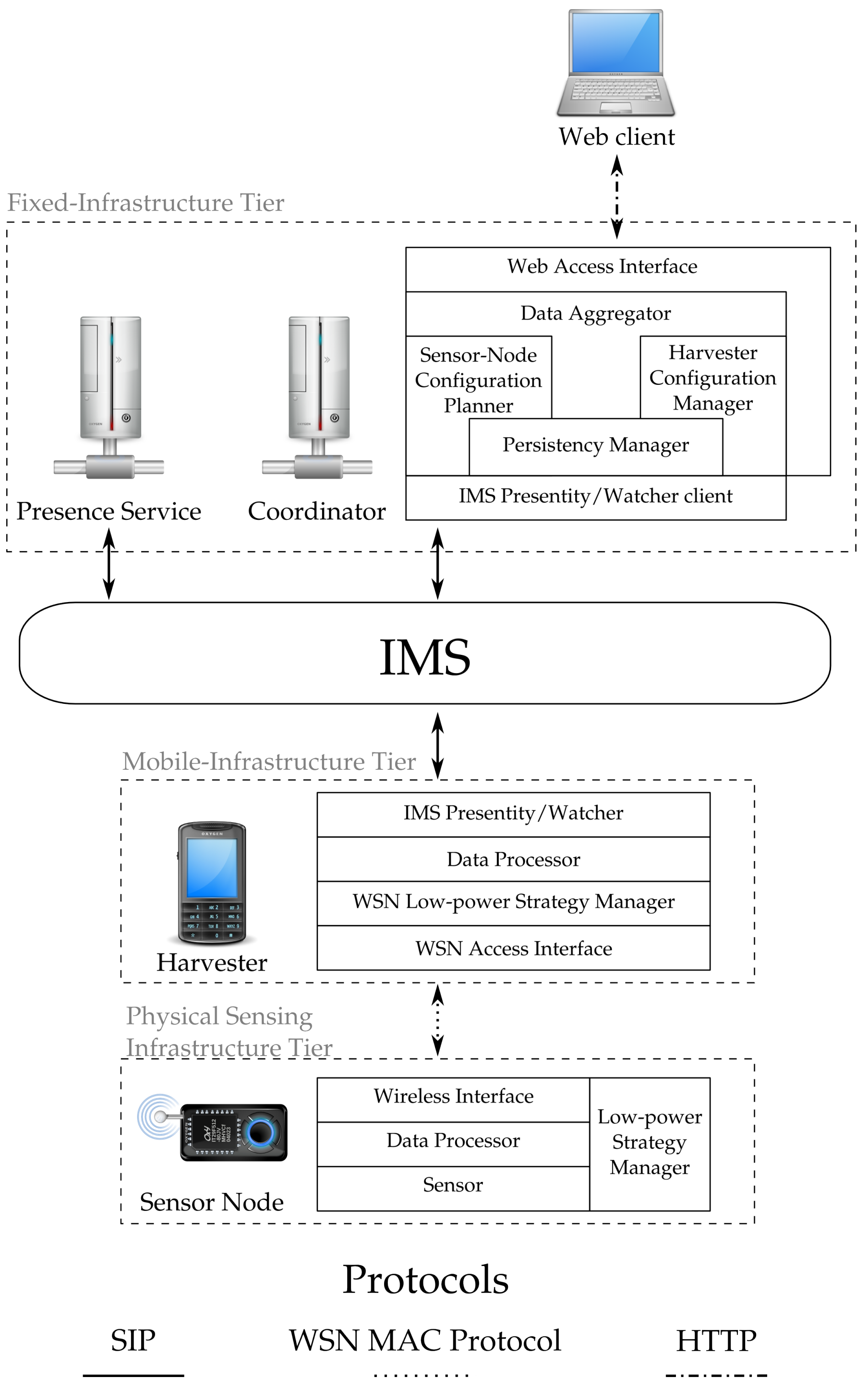


Figure 5.3: General architecture of our IMS-based architecture for data harvesting. The picture does not represent the various components of IMS for the sake of simplicity.

rithms, such as harmonic analysis to filter noisy measurements, that cannot execute directly on sensor nodes due to computing and memory resource limitations. The IMS *Presentity/Watcher* effectively realizes the *Upper-tier interface* acting as both presentity and watcher; it publishes data harvested from sensor nodes, gets notifications about data harvested from other nodes, updates its own location, and receives power management configuration to upload on sensor nodes. Finally, the WSN *Low-power Strategy Manager* completes the *Lower-tier interface*: it translates long-/short-term power strategy control instructions, received by the coordinator as IMS PS notifications, into low level WSN control messages; in addition, if needed, it inhibits useless sensor node querying, for example by avoiding querying a sensor node whose data was just harvested by another mobile node.

The *Coordinator* consists of several components, derived from the architecture described in Section 3.2.3: an IMS *Presentity/Watcher client*, a *Data Aggregator*, a *Persistency Manager*, a *Web Access Interface*, a *Sensor Node Configuration Planner*, and a *Harvester Configuration Manager*. The IMS *Presentity/Watcher client* is the *Lower-tier interface*; it collects mobile nodes location updates and sensed data notifications by all harvester nodes, and publishes harvesting updates to coordinate and control the harvesting task. The *Data Aggregator* realizes the *Data Processing* component: it processes and puts together sensed data using mechanisms more general than the ones available to harvesters, due to its broader view of current and past sensed data from many sources, such as evaluating longer temporal span average values for a specific sensed data type (temperature, pollution, etc.). The *Persistency Manager* is the implementation of the *Data Storage* component: it stores all harvested data on a database that in turn permits to provide a wide range of additional services. For instance, the *Web Access Interface* enables HTTP-based access to sensed data by full-fledged clients and provides also easy-to-use data analysis and data visualization facilities. *Sensor-Node Management and Mobile-Node Management* are implemented by the *Sensor-Node Confguration Planner* and the *Harvester Configuration Manager*. The *Sensor-Node Configuration Planner* collects statistics about sensor nodes querying frequency and selects adequate long- and short-term power management strategies to regulate radio off periods and duty cycling. Finally, the *Harvester Configuration Manager* duty is twofold: on one hand, it computes the length of the wake up signal that the harvester must emit to switch on the target sensor node radio; on the other hand, it inhibits the querying of sensor nodes that have already been queried in the recent past by other harvesters.

Due to its role as core integration component, it is important that the *Coordinator* is able to serve large deployments with good scalability, such as real urban monitoring scenarios with a large number of harvesters and frequent data publications and harvesting synchronizations. About scalability, we claim that that all main Coordinator tasks, namely data harvesting and harvesting synchronization, are intrinsically highly parallelizable because they are spatially correlated and it is possible to apply a strong locality principle to divide different geographical areas and to neatly separate and manage their internal state, apart from the limited sets of data that characterize border zones between different areas.

5.4.2 IMS Network Protocols

Two main events trigger messages exchanges in our infrastructure: harvested data publication and location updates sent by harvesters.

IMS-based data harvesting publication and management requires five steps. First, after querying a sensor node, the harvester (H) publishes the data on the PS (step 1 in Figure 5.4), which in its turn notifies the coordinator (C) with the sensed data (step 2). Then, the coordinator stores the data in its database (DB) and synchronizes all harvesters with a harvesting status update notified by means of IMS PS (steps 3–5). To represent harvesting status, we defined a new XML-based SIP event package including the following fields [176]:

- addresses of the sensor nodes in that area;
- next time to query the sensor node;
- current communication strategy of each node;
- any needed communication strategy update; and
- whether to instruct the sensor node to switch off the radio after successful querying and, if so, the time span of the additional sleep time.

About harvester location updates, harvesters can gather location information by using various methods (GPS, decentralized wireless-based localization, and any available IMS-based localization service), and publish them as a presence notification encoded according the GEOPRIV standard data format [177]. The coordinator node uses location updates to update system harvesting status. Moreover, in large system including several harvesters and sensor nodes, location information allows to divide the notification load according to a physical locality principle. In brief, by using XCAP it is possible to instruct the IMS PS to send harvesting status updates triggered by the harvesting of specific sensor, only to harvesters that are moving in its vicinity.

5.4.3 Sensor Node Energy-saving Strategies Tuning

As detailed in the previous sections, we propose to improve sensor lifetime by deploying a communication strategy on a per-sensor node basis, so that each sensor node is configured based on its specific location and the mobility characteristic of the harvesters that typically may acquire its data.

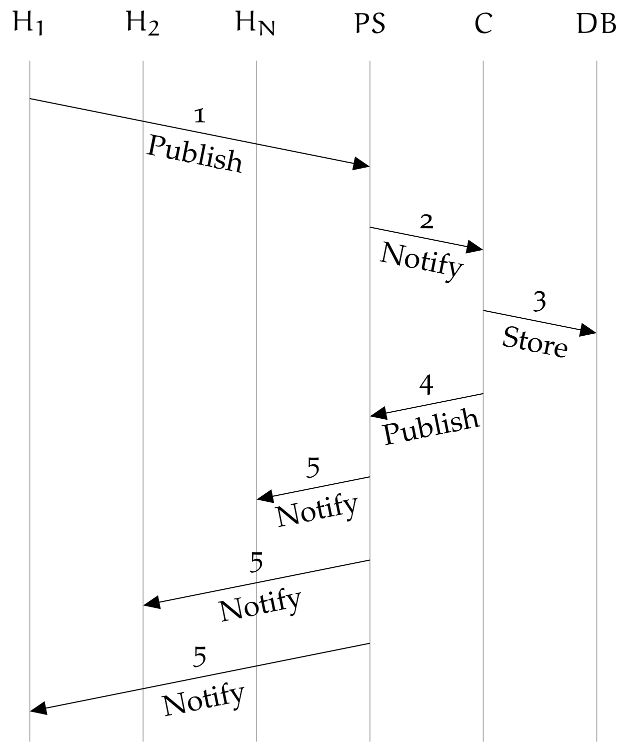
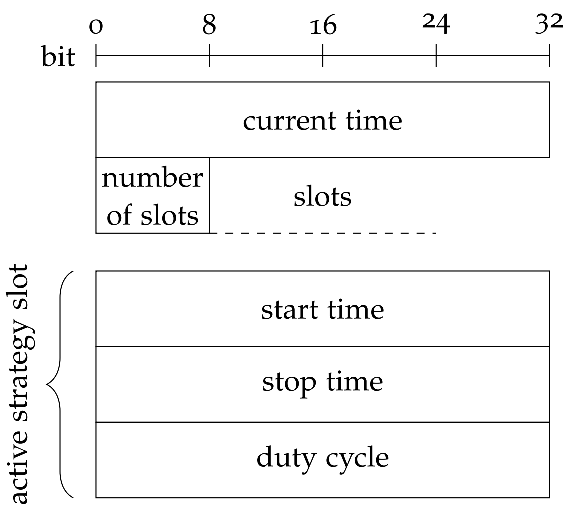


Figure 5.4: Protocol for data publishing, data notification and harvesting synchronization.



```
//start time, stop time
//and duty cycle for
//a single active time slot

typedef nx_struct {
  nx_uint32_t start_tm;
  nx_uint32_t stop_tm;
  nx_uint16_t duty_cycle;
} energy_slot_t;

// current time, number of
// active slots and array of
// slots
typedef nx_struct {
  nx_unit32_t curr_tm;
  nx_uint8_t slots_count;
  energy_slot_t slots[0];
} energy_strtg_t;
```

Figure 5.5: Structure of data packets that carry sensor node energy saving policies.

According to our solution guidelines, our infrastructure supports both long- and short-term power management strategies. The communication protocol between harvesters and sensor nodes is based on compact packets that easily fit in the limited-length frames supported by many sensor nodes. Figure 5.5 shows the header of the packet that uploads a communication strategy on a sensor node, and a description of it. The first `current_tm` field contains the current time represented as standard Unix time and is necessary to compensate the drift between sensor nodes internal clock and real time, which is often non negligible because sensor node clocks are usually cheap and imprecise to minimize their cost. The second field, `slots_count`, represent the number of slots contained in the data packet that define an *active period* that is a period of time during which the radio transceiver of a sensor node is active (possibly duty-cycled). The last field, `slots`, is an array of structures that define *active periods*. Each active period is described by its start time (`start_tm`), its end time (`end_tm`), and the duty cycle, expressed as a 32 bit fixed point value (`duty_cycle`).

For each *active period* slot, the sensor node uses the above configuration parameters to set up two asynchronous timers: one to switch on the radio at the slot start time and the other to switch it off at its end. Then, it uses the duty cycle specified by the strategy to check for transmitters during the on periods.

5.5 EXPERIMENTAL RESULTS

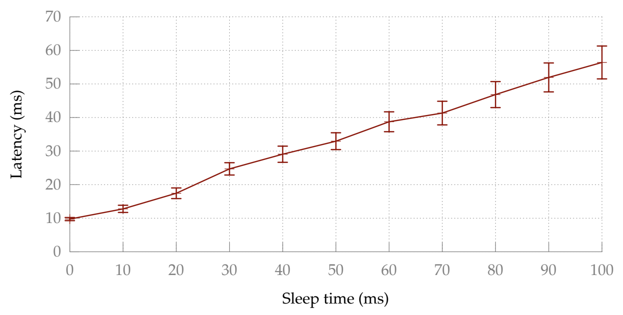
We have tested and evaluated the performance of our solution by deploying it in the heterogeneous wireless network at our campus that includes both WSN nodes and Cisco WiFi access points. Sensor nodes are TelosB nodes running TinyOS 2.1.1 [73, 83]. Harvesters are Linux laptops equipped with an OrinocoGold WiFi card and a TelosB node that enables communications with WSN sensor nodes. Finally, the IMS infrastructure components and the coordinator run on Linux boxes, each one equipped with two 1.8 GHz CPUs and 2048 MB RAM.

As for software components, we have based the development of our energy-efficient harvesting solution on the currently available standard technologies for next generation IMS-based mobile multimedia services. Hence, for session signaling we employed the OpenIMSCore, an IMS platform that is fully compliant with 3GPP IMS specifications [128, 168]. OpenIMSCore provides all the basic components of the IMS infrastructure, e.g., P-CSCF, I-CSCF, S-CSCF, and HSS and a framework to support and manage new ASs. Harvester has been implemented by using standard Linux tools to integrate the sensor node available onboard and designed to integrate with the open-source University of Cape Town (UCT) IMS Client that we significantly modified to support our energy-efficient data harvesting [170].The power management strategies implemented at sensor nodes use the default LPL implementation of TinyOS for TelosB (Box-MAC-2) [154]. Coordinator implementation is based on the eXosip stack [178]. Finally, we employed OpenSIPS for the PS server [169].

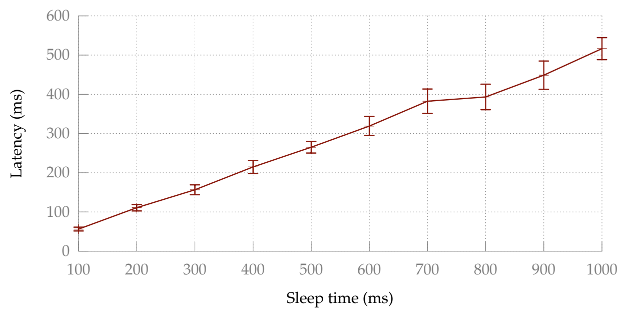
The reported experimental results aim to evaluate three different and crucial aspects of our proposal: (i) the performances of our IMS-based support and of the coordinator component; (ii) how the proposed short-term power management strategy can still grant low data harvesting latency times; and (iii) the significant battery lifetime improvements achievable at sensor node via our long-term power management strategies.

About IMS-based support performances, we refer to Figure 5.3 deployment scenario; reported experimental results, including total response time, IMS PS CPU usage, and coordinator CPU usage, are average values over 100 runs. We concentrate on CPU utilization because, from our experience, CPU is the main bottleneck in IMS infrastructures due to the costs of message parsing/forwarding. In addition, we assume that the number of harvesters that the coordinator has to notify in the same location (step 5 in Figure 5.4) is 20; in other words, we have a watcher/presentity (w/p) factor of 20 subscriptions (from harvesters) for presentity (coordinator). Under those conditions, we stressed the system by injecting PUBLISH messages over harvester WiFi interface (step 1) with an increasing rate, from 10 to 120 call per second (cps) and increasing steps of 10 cps. The system response time, evaluated from PUBLISH message sending (step 1) to final NOTIFY message reception (step 5), always lasts below $20ms$ with a standard deviation of $1ms$ in the $[10, 110]cps$ traffic interval; $120cps$ is IMS PS scalability threshold that provokes a sudden response time increase to $800ms$. CPU utilization follows a similar trend: IMS PS CPU load increases linearly from about 2% (for $10cps$) to 60% (for $110cps$), then it goes up to 100%. The coordinator instead, does not show any overload issue: before IMS PS overload, its CPU load is always below 55%. Let us note that those values, obtainable in a centralized easy-to-manage system, are largely worse in load than in a highly dense metropolitan monitoring scenarios; for instance, a recent study about sensor data harvesting frequencies in central London, Taipei, and Beijing, shows that obtained frequency values are always under 72 cps [179], well below the 120 cps load that our system achieves; for higher loads, it is straightforward to split the incoming load accordingly by applying the geographical locality principle as detailed in Section 5.4.1.

The second experimental result assesses our short-term power management strategy by measuring the latency time to wake up a sensor node using LPL for different duty cycles: this measurement assesses the possibility for a mobile node to wake up a fixed sensor node



(a) Packet latency for sleep time from 0ms to 100ms



(b) Packet latency for sleep time from 100ms to 1000ms

Figure 5.6: Latency time for different sleep times. The radio checks for senders for an 11 ms long period. The vertical bars show the 95% confidence interval.

even if the mobile node moves quickly and the sensor node runs an aggressive duty cycling policy that keeps the radio off most of the time; data harvesting requests are uniformly distributed in a $[0, 2]s$. LPL keeps the radio on for a fixed amount of time ($11ms$) and allows changing the sleep time; Figure 5.6 reports obtained results (a sleep time of 0 ms means that the radio is always on): each result is an average value over one hundred runs, while error bars represent the 95% confidence interval. The average latency time can be considered as half the sleep time, with a better precision as the sleep time grows, and is a linear function of the sleep time. Let us stress that when a mobile node successfully sends a first wake up packet to a sensor node, the sensor node can switch off radio duty cycling to get better network performances. The results in Figures 5.6a and 5.6b show that the time necessary for a first successful communication is predictable and reasonable short even for very aggressive duty cycling policies (e.g., the sleep time of $1000ms$ is a 1% duty cycle).

The third result is an analytical evaluation of the benefits of our long-term power management strategies; for our analysis, we exploit Crossbow TelosB energy consumption characterization model and parameters [73, 125]. In particular, we considered constant energy consumption values for all sensor node components (namely, microcontroller load, flash memory reads and writes, sensors sampling, battery

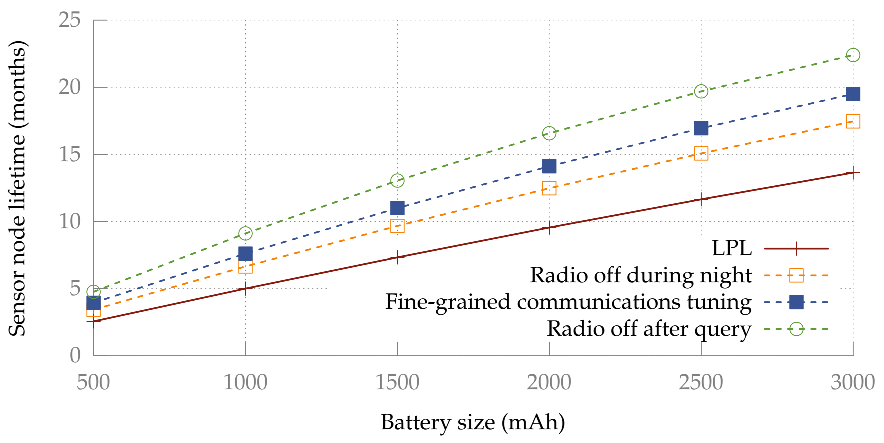


Figure 5.7: Estimate of battery lifetime for different radio management strategies and battery sizes.

self-discharge rate) except for the radio. Focusing on the radio duty cycle, instead, we considered four different strategies:

- LPL with sleep time 1s ("LPL" in Figure 5.7);
- a naïve strategy with sleep time 1s and a radio switching off period of 8 hours per day ("Radio off during night" in 5.7);
- a more complex strategy with two different radio duty cycles, respectively for morning and lunch/evening periods with sleep times 500ms and 1s, such as the one shown in Figure 5.1 ("Fine-grained communications tuning" in 5.7); and
- an enhancement of the previous strategy with radio switching off after data harvesting and until next morning, assuming that sensor node harvesting occurs before noon ("Radio off after query" in 5.7).

As shown in Figure 5.7 the energy-saving technique enabled by our solution enhances noticeably the battery lifetime of the sensor node and permits to obtain an average improvement of 76% in the best case, i.e., comparing "Radio off after query" and "LPL" graphs, thus proving the significant contribute of centralized sensor node management to feasibility of large scale distributed Pervasive Sensing systems.

5.6 RELATED WORKS

Even if a few good papers have started to discuss about the general benefits and issues of energy-efficient data harvesting in Future Network application scenarios, the research work in IMS-WSN integration is still in its infancy. We refer readers interested in works about mobile nodes and WSN integration to Section 4.7.

By specifically focusing on the few seminal research studies about IMS-WSN integration, several works mainly discuss the issues of dispatching sensed data. El Barachi *et al.* [180] analyze the technical challenges to integrate WSNs in the IMS architecture, by proposing a gateway and a common representation of sensor data. iSSEE is a search engine that allows IMS applications to get information about sensor availability, location, and type of sensed data [181]. Finally, iRide is an IMS application that notifies drivers about hazardous situations on the road [182]. Differently from above proposals, that typically focus on IMS-based data dispatching only, our solution explores the possibility to use IMS to actively coordinate and control WSN management tasks, such as data harvesting. We foresee this as a very fruitful research area for the next decade.

5.7 CHAPTER CONCLUSIONS

The project presented in this chapter demonstrates the adaptability of the logical model presented in Chapter 3, by allowing to effectively design a energy-efficient solutions for WSNs data harvesting based on the standard IMS infrastructure. Our tests practically show that a centralized IMS-based system can manage the coordination of data harvesting and the adaptive configuration of each sensor node with minimal impact on data freshness and with a relevant increase in battery lifetime. We stress that our proposal is fully compliant with IMS standards and, thus, easily deployable over already existing IMS-conformant networks. It is important to note that even if they both tackle the problem of dynamic integration of mobile nodes in WSNs, the project described in this chapter and WHOO, described in Chapter 4 are not competitors, rather they are complementary: the project presented in this Chapter leverages a centralized control component to upload power-efficient strategies on sparse sensor nodes and collect data, whereas WHOO dynamically exploits in a completely distributed fashion mobile nodes to route urgent data.

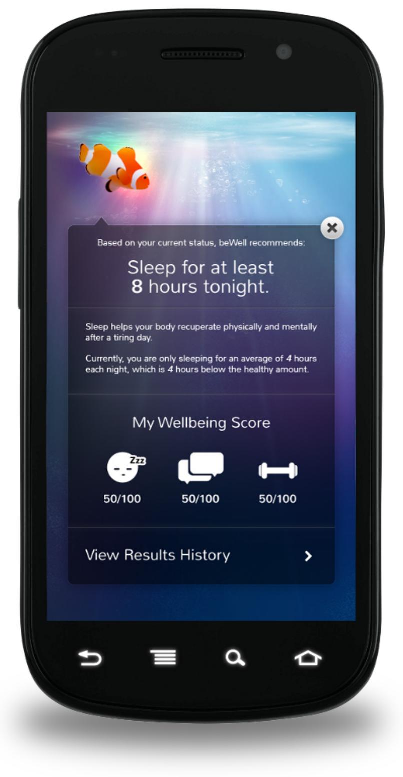
6 SMARTPHONES AS SENSORS

We presented in Chapters 4 and 5 two implementations of our logical model to data-centric Pervasive Sensing systems focused on environment monitoring that opportunistically exploit mobile devices as infrastructure to support sensing tasks. However, mobile devices, especially smartphones, are candidates not just as support for WSN, but also as sensors themselves, due to the large set of sensing hardware that they host on-board, including accelerometers, gyroscopes, GPSs, microphones and cameras. These new features make mobile devices powerful and complete sensing platforms to continuously watch and monitor the behavior of users who move and act in the physical world bringing with them their mobile devices. Moreover, it is possible to process on the mobile device large sets of locally collected raw data and to distill meaningful views of the activity currently done by the user, such as running, cycling, talking, and sitting, by exploiting signal processing and machine learning algorithms; in brief, we call the whole continuous sensing process as *inferring user current activity*. Many mobile applications can benefit these brand-new mobile sensing capacities and span different areas, from healthcare to homecare, from safety to smart grids and environmental monitoring, and many more.

In this chapter we briefly present a healthcare mobile sensing application (Section 6.1) that is supported by a continuous sensing library introduced in Section 6.2. Section 6.3 describes previous works useful to understand main issues in mobile sensing, Section 6.4 details the design guidelines stemmed from the analysis of previous work and the resulting architecture of our sensing library. Section 6.5 delves into the details of a classifier supported by the sensing library. Finally, Section 6.6 describes implementation details and assesses the library performance by comparing it to an existing framework and to a bare-bone implementation. Section 6.7 summarizes the contributions of this chapter.

6.1 BEWELL: A HEALTHCARE MOBILE APPLICATION

A leading example of a Pervasive Sensing system that exploits smartphones as sensors is BeWell that was created at Dartmouth college and has been further developed in joint collaboration with Cornell University and University of Bologna [141]. BeWell is an application


Based on your current status, beWell recommends:

Sleep for at least 8 hours tonight.

Sleep helps your body recuperate physically and mentally after a tiring day.

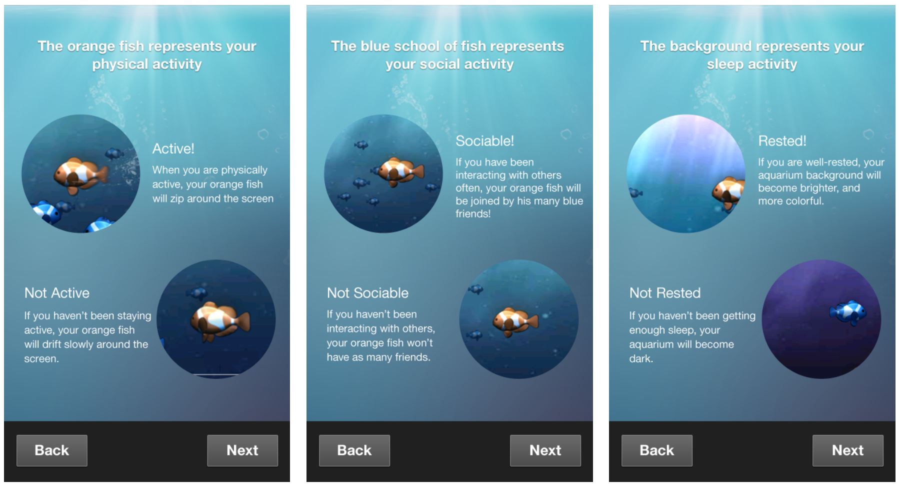
Currently, you are only sleeping for an average of *4* hours each night, which is *4* hours below the healthy amount.

Figure 6.1: BeWell showing user score summary.

for Android smartphones that exploits sensors to automatically estimate three wellness metrics: physical activity, social activity, and sleep. These metrics are normalized in a 0 - 100 score, 0 being the worst score and 100 being the best, which is shown on request to users as a popup (Figure 6.1). In addition, BeWell implements a live wallpaper that allows users to understand what are their scores by simply glancing the screen (Figure 6.2).

The live wallpaper shows an animated marine environment with an orange clownfish that represents the user himself. The fish swimming style reflects the level of physical activity: if the score is high the fish will zip across the screen, if it is low it will slowly drift (Figure 6.2a). The clownfish is surrounded by a school of blue fish that represent social activity: if the user has many social interactions and talks to several people then there will be a lot of blue fish that become fewer and fewer as the score decreases, until the clownfish swims alone if the social score is very low (Figure 6.2b). Finally, the background light represents sleep: if the user sleeps enough time then the marine environment is bright and colorful, otherwise it will become progressively darker and gloomier (Figure 6.2c).

Physical activity is classified as stationary, walking, or running by continuously processing accelerometers values. These inferences are then used to estimate Metabolic Equivalent of Task (MET) value


The orange fish represents your physical activity

Active!

When you are physically active, your orange fish will zip around the screen

Not Active

If you haven't been staying active, your orange fish will drift slowly around the screen.

The blue school of fish represents your social activity

Sociable!

If you have been interacting with others often, your orange fish will be joined by his many blue friends!

Not Sociable

If you haven't been interacting with others, your orange fish won't have as many friends.

The background represents your sleep activity

Rested!

If you are well-rested, your aquarium background will become brighter, and more colorful.

Not Rested

If you haven't been getting enough sleep, your aquarium will become dark.

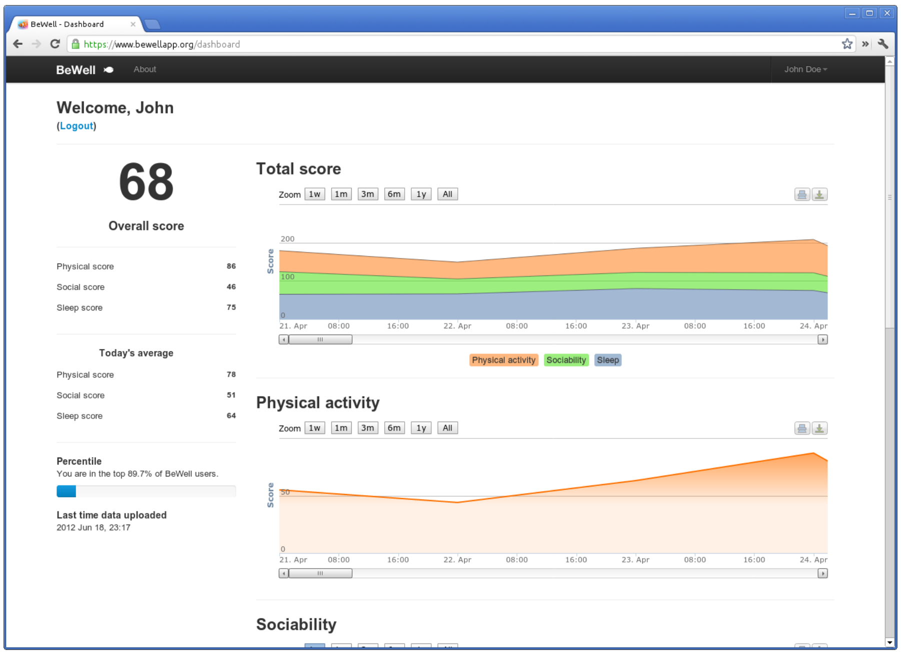
(a) Physical activity. (b) Social activity. (c) Sleep.

Figure 6.2: Tutorial of the BeWell mobile application that explains to users how their scores influences the live wallpaper.

[183]. MET is a physiological measure that estimates the energy cost of physical activities, from sleeping (0.9 MET) to running (23 MET). Estimated daily MET is then compared with healthy ranges suggested by Centers for Disease Control and Prevention (CDCP). Social activity is estimated as a function of the duration of ambient speech during a day that is detected using a classifier that continuously receives data from the smartphone microphone. Currently the score is estimated by comparing ambient speech duration to the typical daily quantities of speech encountered by people in a field trial, as described in [141]. Finally, sleep duration is currently estimated by a best effort classifier that uses regression over several inputs (time since last phone usage, ambient light, physical activity), and achieves a ± 1.5 hours accuracy compared to ground truth. Sleep duration is then converted to a 0-100 score by mapping it to a normal bell curve that has its peak for 8 hours of sleep, decreasing for much longer or shorter sleep durations. As part of the research on BeWell, we developed a more accurate classifier, described in Section 6.5.

Users can optionally upload their data on our web site that allows to see the historical trend of their wellness scores (Figure 6.3). Moreover, the web site geolocalizes scores, thus allowing users to understand whether there is a correlation between their scores fluctuations and the places that they attend (Figure 6.4). From the point of view of researchers, data collected on the fixed infrastructure allows to build a "mood map" that shows the influences of places on average wellness of contributing users.

A key challenge for the realization of BeWell is supporting continuous sensing on smartphones without reducing too much battery lifetime and avoiding bad impacts on the quality of smartphone user



Welcome, John
(Logout)

68
Overall score

Physical score	86
Social score	46
Sleep score	75

Today's average

Physical score	78
Social score	51
Sleep score	64

Percentile
You are in the top 89.7% of BeWell users.

Last time data uploaded
2012 Jun 18, 23:17

Figure 6.3: Web interface of BeWell that shows user historical scores.

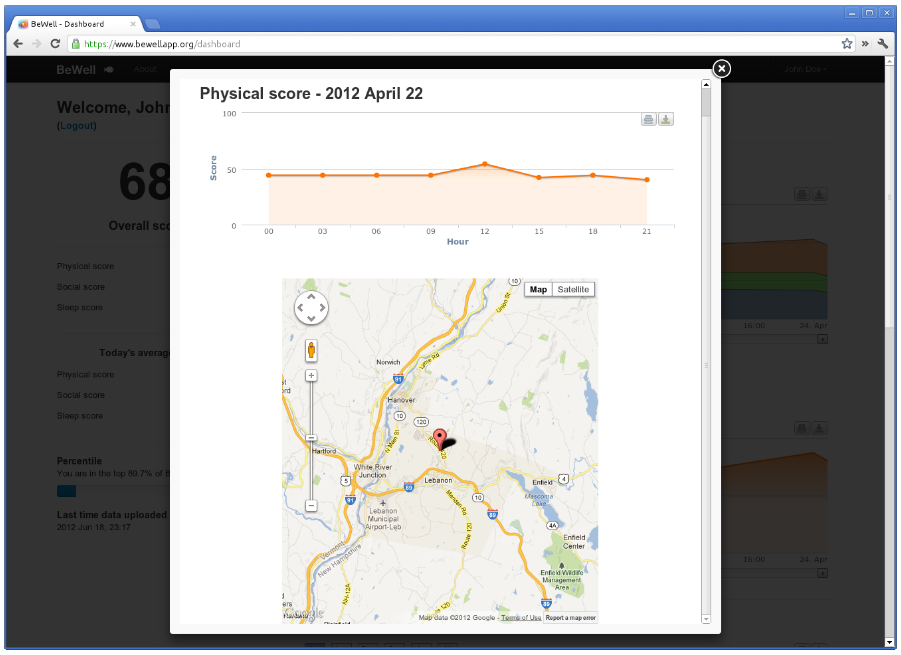

BeWell - Dashboard
https://www.bewellapp.org/dashboard
Physical score - 2012 April 22
Score
Hour
Map
Satellite
Norwich
Hanover
Hartford
White River Junction
Lebanon
Enfield
Lebanon Municipal Airport-Leb
Enfield Center
Enfield Wildlife Management Area
Map data ©2012 Google - Terms of Use Report a map error

Figure 6.4: Web interface of BeWell showing relationship between physical activity scores and locations.

experience. We derived from the experience of BeWell development an open source library that tackles and solves these challenges, and can be re-used as basic building block for novel applications based on smartphone sensing.

6.2 BACKGROUND TO SMARTPHONE SENSING

There are several technical challenges to be solved in designing mobile sensing applications and supports viable and valuable to the mobile market, mainly because there are still several open technical issues that affect the mobile sensing practice. First, most of the currently available solutions are considered vertical ones and make it difficult to reuse specific components, such as data gathering and energy management just to cite two typical horizontal facilities. Second, inferring user activity is a CPU-intensive task that requires retrieving raw data from sensors, preprocessing them to extract some synthetic characterizations of sampled signal periods (or features) and using these features to evaluate and infer actual activity [25]. In other words, mobile sensing is intrusive and risks to disrupt the overall user experience, especially because several mobile apps include multimedia services with strict soft real-time constraints. Third, monitoring tasks require intensive use of hardware sensors, computing resources, and storage to continuously gather, process, and save data; those activities can drastically reduce battery lifetime and should be carefully managed to control and minimize the mobile sensing energy footprint. Fourth, although early projects used to off-load sensing and processing to external devices [10], modern sensing applications run both monitoring and processing directly on the smartphone, thus requiring a more careful management of concurrent access to both sensors and computing resources [184].

To address all above issues, we propose the Mobile Sensing Technology (MoST) solution, a novel Android framework for mobile sensing that aims at offering app developers a set of attractive facilities and functions to quickly and easily design their own mobile sensing services. Faithful to the principles described in Section 3.1, MoST exhibits several original characteristics. First, MoST is *general-purpose*: its architecture includes some horizontal services, such as sensor control functions, raw data gathering, power management to allow developers to focus on sensing logic and to easily wire their own data processing code into existing MoST skeletons, without having to deal with repetitive control and management tasks, demanded to the framework. Second, MoST is *non-intrusive on user experience*: it has been optimized for managing large streams of raw sensing data by carefully tuning and controlling concurrent access to system resources so to avoid any useless resource bind and to minimize additional processing overhead. Third, MoST is *energy-aware*: it implements an efficient and flexible power management component that minimizes the impact of sensing applications on battery lifetime and includes several policies, both pre-defined and configurable by developers, to automatically control the duty-cycle of each sensing task. Finally, MoST is *performant*: to better underline this original aspect of MoST, we presents a thorough quantitative comparison with a selection of very close mobile sensing solutions to show that MoST outperforms other benchmarked systems in terms of performances and scalability.

6.3 RELATED WORKS

The mobile sensing trend has spurred many research projects that have focused on a variety of different problems, spanning from power-efficient signal processing to social sharing of sensed data. We believe that understanding the goals, the needs and techniques of existing research efforts is the key to develop a lightweight mobile sensing framework that is useful and easy to use. Hence, in the following we report some of the major works in mobile sensing literature presented, by starting with more simple single ones, such as applications based on a single sensor, and then moving on to multi-sensor applications, to conclude with general-purpose sensing frameworks.

The first type of mobile sensing applications includes seminal works that leverage a single sensor, among the many available on the mobile device (accelerometer, microphone, light, etc.), to gather monitoring data and to use them to infer user current activity, such as walking, running, and standing. Usually, these are typically vertical "silos" applications that start from raw sensor data, go through a pre-processing stage, and end with a classification stage. A commonly used sensor is the accelerometer exploited to infer the current physical activity of the user [185, 186]; then, recognized activities can be used in multiple ways, to include promoting green behavior, monitoring fitness, and emergency detection. For instance, UbiFit measures the physical activity of users to nudge them towards using more environmentally sustainable means of transportation (e.g.: walking vs. driving) [10]. GymSkill is another example of a fitness mobile sensing application that monitors and assesses the quantity and quality of some physical activities that use standard fitness equipment [187]. About healthcare, PerFallID uses accelerometer data to detect user falls via a simple classification stage based on a threshold whose value is dynamically adjusted by using data collected from real users and occurred falls [188]. However, the accelerometer is not the only sensor used in many people-centric sensing applications: microphone is also a good source of information to make accurate inferences about people and environment. For example, the SoundSense realizes a high-level activity inference component that recognizes music, speech, and different ambient sound [11]. SpiroSmart, instead, is an iPhone healthcare app from an analysis user exhaling sound estimates breathing parameters, usually obtained via spirometer [13]. The second type of mobile sensing applications explores the possibility to fuse data coming from multiple sensors toward different possible goals, such as either increasing classification accuracy or providing additional features. These proposals are still vertical systems, but start to adopt more complex architectural solutions that may include horizontal services, such as in power management. Prominent examples of these multi-sensor applications, in addition to BeWell itself, are CenceMe and BALANCE [101, 189]. CenceMe is a personal sensing system that allows users to share their activities with friends on social networks: it gets data from accelerometer, camera, and microphone, and infers different socio/physical dimensions such as user activity (e.g., walking, biking, and running), disposition (e.g., happy and sad), and environmental conditions (e.g. noisy, hot, and bright) that can be automatically shared on popular social network such as Facebook and Twitter [102]. BALANCE, instead, based on input from accelerometer, barometer, GPS, light sensor, humidity and microphone, aims at automatically estimating calories burst by users [189].

All these research efforts, together with many ones we cannot cover here for space limitations, have generated enough momentum to push the development of the third generation of mobile sensing applications represented by more comprehensive mobile sensing frameworks. A seminal work in this direction is the Funf Open Sensing Framework (in short Funf) [190]. Funf is an extensible sensing and data processing framework for Android providing a minimal set of reusable facilities for collecting, locally configuring, and uploading to remote servers a wide range of sensing activities. First of all, Funf divides sensors in hardware ones (e.g., accelerometer, microphone, GPS, etc.) and logical ones (e.g., recorded sound streams, application usage data, etc.) and defines data sources abstractions, namely "probes", and Application Programming Interface (API) to gather raw sensing data. However, it presents several limitations: it provides only low-level sensing data access functions and does not support inferring higher-level activities; its horizontal facilities are very static, such as power management that only allows for the definition of configuration files indicating the query period of each probe; and it does not include those micro-/macro-optimizations, such as object reuse, lightweight resource binding, and so forth, that are extremely important to make a framework acceptable in terms of overhead, responsiveness, and resource consumption for the final user.

6.4 DESIGN GUIDELINES AND LOGICAL ARCHITECTURE

As in the previous section, designing a general-purpose mobile sensing is still a complex task that requires a deep knowledge about all common traits and issues of mobile sensing applications. From our knowledge and experience of this area, we have identified some main design guidelines for a logical architecture of our full-fledged sensing framework that includes not only generalized sensing capabilities but also horizontal ancillary service, to ease the design of novel, even complex, activity recognition components, by efficiently taking care of all low-level system-/resource-management issues and by hiding unnecessary details.

6.4.1 Design Guidelines

We apply the generic principles of Section 3.1 to MoST. First of all, mobile sensing applies to several application domains, each one with its own specific characteristics: sensing can be either continuous or sparse, classification can be either lightweight or complex, data can be processed either locally or not. Thus, the framework architecture should be modular and easy-to-use at two levels: application developers should be able to quickly create new applications based on raw data and/or already computed high-level inferences; library developers should be able to easily plug-in new components, such as support for new sensors and activity classifiers. To achieve both goals, it is crucial to adopt the *modular approach* via a *layered architecture* that clearly divides the main framework levels: accessing sensors and system resources; inferring activities by processing sensed data; and providing high-level abstractions to services to query and to register for specific activity recognition events.

Second, mobile devices must always be responsive to user input and should not cause any unexpected behavior, such as errors due to hardware resource locking. Therefore, the framework should be able to *manage and control itself*, namely to adaptively tailor all sensing operations that might undermine and degrade user experience, by resolving all possible conflicts. For example, because the microphone resource can be acquired exclusively by one process sensing at a time, the framework should automatically switch-off all audio sampling sensing tasks whenever the user receives a call not to interfere with user expected phone-call behavior. Moreover, user transparency requires to hide the take-over to library developers without requiring a deep knowledge of low-level system issues.

Third, the framework should be *adaptive*, allowing library and application developers to *flexibly and dynamically change the framework behavior* through easy-to-use configuration primitives and directives.

Along that direction, the framework should provide a set of configurable management components, especially for sensor and power management. The sensor management should make it possible to dynamically reconfigure all ongoing activity recognition tasks, when applications/users decide activities on specific sensors. Similarly, since mobile devices have a limited battery capacity and continuous sensing can significantly reduce battery life time, the framework should support multiple energy saving strategies and adaptive duty-cycling approaches, and it should be possible to switch from one approach to another at runtime, without stopping ongoing sensing tasks.

Fourth, and finally, because mobile sensing applications rely heavily on CPU and memory hungry algorithms that can deteriorate smartphone performances, the sensing framework should be *resource-aware*, hence *carefully designed and tailored to include any possible low-level optimization* to reduce overhead and to limit the impact on local resource usage (CPU, memory, communications, bandwidth, etc.). In particular, used resources should be carefully bounded and reuse of resources should be fostered, so to avoid the execution of frequent and heavy garbage collection operations; hence, whenever possible, resources should be preferred reusable in pools and employed for both passive and active entities involved in the whole sensing process.

6.4.2 MoST Logical Architecture

The MoST goal is to provide a high level framework for the development of sensing applications that provide out-of-the-box classification algorithms capable of inferring high-level activities (e.g., walking, running, and cycling) from raw sensor data.

Let us all start by introducing some basic MoST concepts and abstractions. First of all, we name *Input* any source of sensing data (e.g., accelerometer, microphone, GPS), while a *Pipeline* is the component that encapsulates the application logic to gather, process, and meld together sensed data. Pipelines process data collected from one or more Inputs, so to evaluate – typically by exploiting specific classification algorithms – high-level views of current user activity, such as sleeping, walking, and standing. Conceptually, Inputs and Pipelines are the basic building blocks to realize new MoST mobile sensing chains that continuously run and evaluate activity via inferences. According to our sensing framework resource-aware self-control principle, there must be a way to interact with and, when necessary, to interrupt parts of these sensing chains. The Management System is in charge of mediating MoST interactions with the external world, by handling two main types of event: system events, triggered either by the system of by other apps (e.g., incoming call, battery running out, etc.), and user events, triggered by the user (e.g., pause all sensing tasks, disable

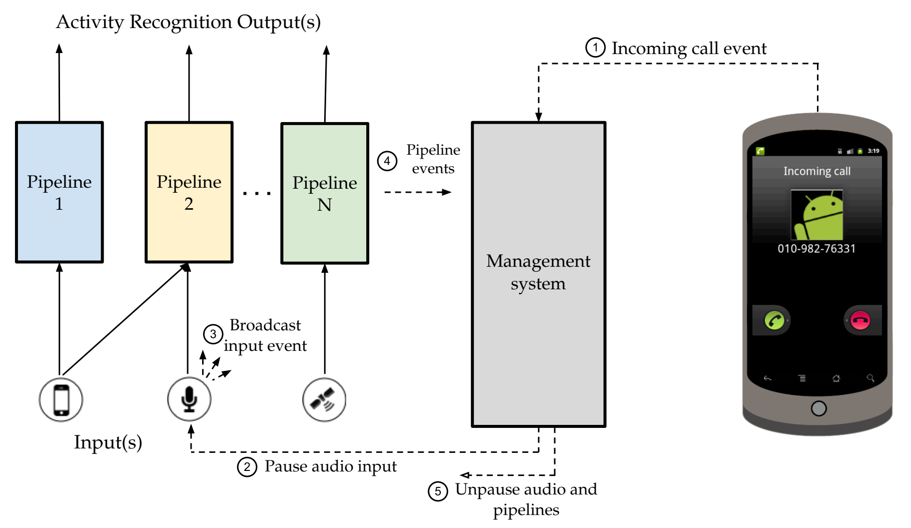


Figure 6.5: MoST use case for system/user events handling.

an input, etc.), and controls and coordinates the interactions between Inputs and Pipelines.

To better ground these concepts and to illustrate the complexity of the management issues involved in realizing non-intrusive mobile sensing tasks, let us introduce a simple but real activity recognition scenario. Suppose that we want to develop a sensing application that uses a microphone and an accelerometer to recognize the following activities: walking, running, stationary silent, and stationary speaking user. In order to realize it, we need two Input objects to read, respectively, the microphone and the accelerometer, and a Pipeline that encapsulates activity inference application logic. When the application is running, Input components read and deliver sensed data to the Pipeline, which returns the result of the activity recognition algorithm to the registered application. Now, let us assume that an incoming call, signaled as an internal system event, arrives while the sensing application is executing: the framework must immediately release the microphone to allow user answering the call.

To be more precise, Figure 6.5 details all main interactions between framework components in the above example. When the incoming call arrives, the audio-call app broadcasts an event to other interested apps; MoST, that handles all available system/user events (step 1 in Figure 6.5), dispatches it to our Management System that must stop sensing processes that require the microphone, release the microphone, and then reacquire it, and restart stopped processes when the call completes. In particular, the Management System triggers the following chain of actions: it sends a control message to the audio Input asking it to release the microphone resource (step 2); the Input pauses, releases the microphone and notifies this state by broadcasting an internal Input event (within the MoST framework) that can be caught by all Pipelines using the microphone Input that can either pause

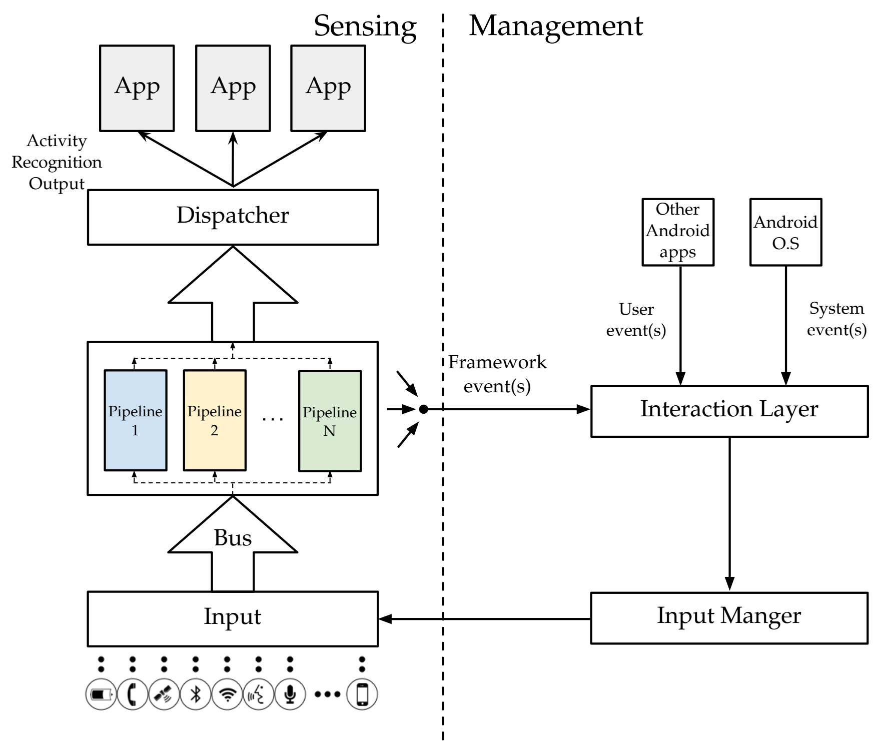


Figure 6.6: MoST basic components and main management operations.

themselves until the microphone is available again or keep running without that data (step 3); Pipelines that decide to stop notify their decision to the Management System that keeps track of the whole MoST internal state (step 4). When the call ends, the Management System wakes up the microphone Input and notifies all stopped Pipelines that microphone is available again (step 5). It is important to stress the relevance of this design that allows continuous access to sensors and dynamically pausing them on external events, thus minimizing the impact on user perceived responsiveness of their smartphone, because it minimizes the possibility of users uninstalling applications based on MSF due to a detrimental effect of user experience.

After presenting the main MoST components, in the following, we detail the logical architecture and all entities of our framework that consists of two main subsystems: Sensing and Management (Figure 6.6).

The Sensing subsystem adopts a three-layered architecture and deals with all data gathering and data processing aspects: at the bottom layer, we find Inputs to gather sensing data and to wrap them in easy-to-manage local objects sent by a Bus to enable internal delivery of sensed data to all interested Pipelines; Pipelines are the core components of the middle layer to work on a global and efficient view; and finally, at the top layer, the Dispatcher provides to interested apps the APIs to register to MoST to receive high-level inferences evaluated by

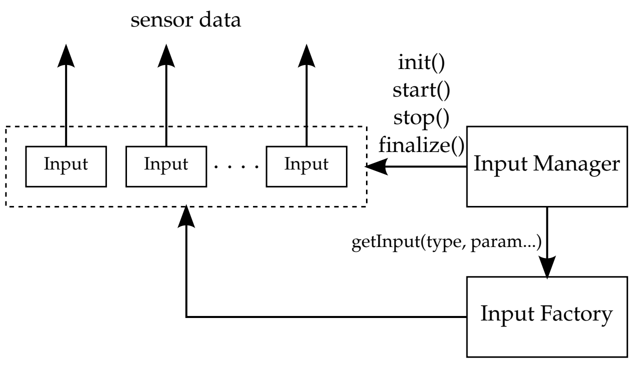
Pipelines. The Management subsystem, on its turn, coordinates and controls the whole interaction of the Sensing subsystem with the environment: the Interaction Layer is the listener that receives internal (i.e., framework) and external (i.e., system and user) events, while the Input Manager is the core management component that, according to Pipeline needs and currently monitored system/user situation, coordinates Input and Pipeline execution by propagating Input state change to Pipelines that can temporarily pause until the needed Inputs are available again. From this logical architecture, applicable to any mobile platform, we realized our MoST for the Android platform that is the most widely adopted one in mobile sensing. In addition, Android allows sensor access even when the system is in standby, a key feature needed by continuous sensing systems and is not available on other mobile platforms (e.g., Apple iOS) [141, 191].

6.4.3 Sensing Components

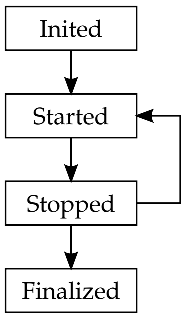
This section describes some finer-level details of Sensing subsystem components. The Input component is the data source to enable sensing from local hardware and logical sensors: it gathers data and makes them available to Pipelines. All Inputs share the same interface, thus MoST can instantiate them and manage their lifecycle by abstracting from their internal details: this allows third-party developers to easily develop and integrate new Inputs. Input instantiation is dynamically managed by Input Manager according to Input needs expressed by Pipelines through the Input Factory; then, Input Manager manages the whole Input lifecycle (Figure 6.7a).

To facilitate programmers with a well-known pattern, the Input lifecycle is a simplified flow derived from the one of Android components [192]. As soon as it is created, Input switches to the *Inited state* (Figure 6.7b) in which it is configured and initialized before start reading sensor data; more precisely, this state makes the Input get all long-term resources to use for its whole lifetime span, such as internal buffers to store sensed data. *Started state* is the actual execution state in which Input gathers sensing data and pushes them up to the Bus and eventually to Pipelines; then, when Input has to temporarily stop, it makes a transition from *Started* to *Stopped*. Following the behavior suggested by Android, in Stopped state, Input must release lightweight resources that can be easily re-acquired afterwards, including used sensors, such as the microphone released during a phone call. Finally, in the Stopped state Input can be either destroyed (to *Finalized state* that frees all long-term resources) or reactivated (to Started state again).

The second sensing component is the *Bus*: it realizes many-to-many distribution of sensed data samples from each Input to all Pipelines subscribed to that Input type; to correctly dispatch sensed data to



(a) Input instantiation



(b) Input lifecycle

Figure 6.7: Input-related components and lifecycle management.

Pipelines, it keeps track of both active Pipelines/Inputs and all dependencies between Pipelines and Inputs.

Focusing on the resource usage aspects, we reason on *sample*, the minimum chunk of sensed data returned by hardware/logical sensor, and encapsulate each sample in a *DataBundle* realized as a container object wrapping that tags the raw sensed data sample with additional details useful both for the Pipeline and for the internal resource management: typical information are Input type, sensing timestamp, and a reference counter. In particular, let us stress that Input sampling rate may cause a high rate creation of DataBundle objects, such as in common cases of using microphone and accelerometer with high sampling frequency; hence, Pipelines use them, DataBundle objects could uselessly waste memory resources until they are collected by the garbage collector, thus wasting CPU cycles too.

Following our main design guidelines, MoST avoids that waste by using a DataBundle object pool to reduce the average memory footprint of the framework, by recycling already used DataBundle objects, and by using explicit reference counting: when the Input obtains a new sample from a sensor, it gets a free DataBundle from DataBundle pool and passes it to the Bus; then, the Bus initializes the reference counter to the number of Pipelines subscribed to that Input and delivers the DataBundle; afterwards, each Pipelines, once completed the sampling process, decrements the DataBundle counter; when the

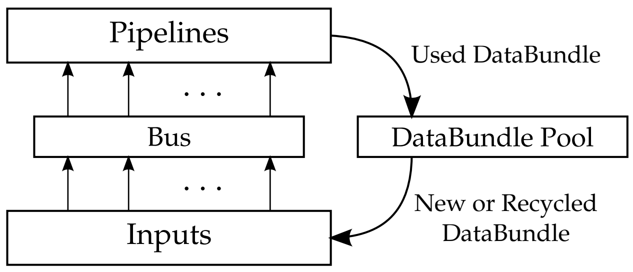


Figure 6.8: Bus and DataBundle management.

DataBundle has been used by all Pipelines it is released back to the pool (Figure 6.8).

We have to better describe *Pipelines*, the general-purpose skeletons to be filled with specific activity inference application logic. Pipelines are designed to be self-contained and easy to dynamically instantiate, activate, and deactivate at runtime. Each Pipeline, identified via a unique global identifier, has to explicitly declare the Inputs it wants to subscribe to, and has to follow the Pipeline lifecycle defined by MoST. The unique identifier, currently represented as the fully-qualified package name, allows applications to unambiguously choose the Pipelines to use; about Input subscriptions, instead, each Pipeline has to statically declare them and then, at runtime, Bus and Input Manager uses those subscriptions, respectively, to deliver DataBundles and to signal the Pipeline of Input state changes. Finally, Pipeline lifecycle is similar to Inputs one: Pipelines are first created and inited; then, while started they can receive raw data from Inputs and can output inferences; Pipelines can also stop, by temporarily releasing resources; if they are not needed anymore, they are finalized. At the same time, it is important to note that while Input lifecycle is mainly driven by external user and system events via the Input Manager, Pipeline lifecycle is self-contained in the private Pipeline code that controls and decides needed strategies for state transitions autonomously, to allow different policy coexistence. For instance, a Pipeline that takes data from two different Inputs may be able to work even if one of them is stopped by generating approximate inferences; another Pipeline may not work without the data from all Inputs and hence should switch to stopped state as soon as any of its Inputs stops.

The very loose constraints imposed by MoST allow the development of arbitrarily complex Pipelines, that run in separate threads; however, it is up to developers not to abuse by designing Pipelines CPU- and memory-hungry that, in their turn, could cause excessive battery usage and worsen final user experience.

The last Sensing component is the *Dispatcher*. It realizes a many-to-many distribution model to deliver activity inferences from Pipelines to interested apps. New apps present to the Dispatcher their interest in receiving activity inferences from a specific Pipeline, and the Dispatcher calls them back, as soon as a new relevant inference is available. Thanks to MoST dynamic management of Pipelines and Input, apps can also register for Pipelines that have not been instantiated because there are not interested clients yet: in that case, Dispatcher bootstraps the required sensing chain by creating a new Pipeline; symmetrically, if all apps de-registered from a Pipeline, the Dispatcher frees systems resources by shutting it down. Pipeline creation and destruction triggers the creation/destruction of Inputs, managed by Input Manager.

6.4.4 Management Components

The *Input Manager* is the key Management subsystem component that manages Input lifecycle from instantiation to shut-down thus, indirectly, influencing also Pipelines lifecycles. When the Dispatcher creates a new Pipeline, the Input Manager starts all needed Inputs; correspondingly, when a Dispatcher is destroyed, the Input Manager also frees all Inputs that are no longer used. In addition to Inputs de-/allocation, the Input Manager is also responsible for driving the whole Input lifecycle by triggering their pause, resume, stop, and restart operations, based on event notification received from the Interaction Layer.

The *Interaction Layer* receives system- and user-events and reports them to MoST so to avoid their interference with the expected behavior of the mobile phone, in the sense of non-intrusiveness. The Interaction level also supports other events, such as low-level battery warning, screen going on or off, and other user-defined events (that allow users to stop sensing whenever they want). The Interaction Layer receives all these events and passes them to the Input Manager, which then accordingly manages the life cycle of inputs. The Interaction Layer acts on a strict event-action basis and its support for arbitrary events provides the building block to develop new power management policies that drive the duty cycling of Inputs. The current MoST power management policy leverages the Interaction Layer to implement a simple duty-cycle policy that periodically pauses and, after a while, resumes all active sensors; however, the same architecture allows to easily integrate more complex policies that selectively pause Inputs by adapting their decision to the current usage context [193–195].

6.5 SLEEP CLASSIFIER

According to the long-term plan to make MoST a complete sensing library that provides high level inferences to developer, and to refine the capabilities of the BeWell app, we developed a Pipeline that exploits the accelerometer to estimate sleep duration.

6.5.1 Motivation

From the medical point of view, sleeping is a complex activity whose impact on overall wellbeing depends on its quantity and quality. Quality of sleep can be evaluated partially via physiological measurements, such as respiration rhythm, alternation between deep and Rapid Eye Movement (REM) sleep phases, and limb movement, and partially via personal evaluations [196]. The proper way to monitor sleep quantity and quality are polysomnographic studies that use a polysomnogram to monitor sleep. A polysomnogram monitors at the same time brain waves via Electroencephalography (EEG), eye movements via Electrooculography (EOG), muscle contraptions via Electrocardiography (ECG), blood oxygen levels via pulse oximetry, snoring via microphone, and restlessness via camera. Due to their complexity and costs polysomnograms are impractical for long term sleep monitoring. Currently there are no devices that can cheaply run full-fledged polysomnographic studies on a large scale. Due to this technological restriction, we focus on the simpler task of measuring sleep duration. Sleep duration is a coarse indicator of sleep quality, however it is well correlated to wellbeing [197], and can be deeply influenced by a wide range of physical and physiological conditions such as affective disorders, depression, anxiety, hypertension, heart diseases, diabetes. For example, Schlosberg and Benjamin report in [198] that heavily stressed people suffer of reduction total sleep time (an average of 3.8 hours), of longer sleep latency (the amount of time taken to fall asleep initially), and of more awakenings during the night.

There exists tools that approximate a polisomnographic study using additional hardware such as accelerometers strapped to patient limbs and simplified EEGs [199–201], but they constitute an additional cost, thus they may not be a viable solution for large scale sleep monitoring. There exist apps that help users keeping track of their sleep patterns, but they require users to actively input their sleep time, making them less appealing to users who are not enough keen on actively tracking their wellbeing. Using a smartphone for estimating sleep duration has two advantages: first, it does not require any additional hardware, second, it is based on a device that users often interact with, thus easing the task of detecting whether the user is asleep or not.

Inspired by other accelerometer-based devices such as Actigraph and Fitbit [199, 201], the core idea of the pipeline for sleep duration estimation is to ask users to keep their phone face-down in bed with them when they go to sleep: this action can be automatically detected by a pipeline by processing accelerometer data. Let us state early that even if we were able to detect whether a phone is on a bed (or not), that does not guarantee that the user is sleeping (or not); however if the user complies to the very simple protocol, we can get accurate estimations of the time spent in bed, which is reasonably linked with the actual sleep time.

6.5.2 Pipeline Implementation

Realizing a classifier like the sleep classifier is a three step process: first, build a dataset of labelled raw data, second, process the raw data to extract relevant features that summarize their properties, third, feed it to a machine learning algorithm that trains a classifier that can label unknown data.

To build the dataset of raw data we wrote a simple accelerometer-logging application that records at the highest possible sampling rate data from accelerometers. This application has been used for one week by three different people, using three different smartphones, sleeping on three different mattresses, for a total of more than five hundred hours worth of data. Users manually took note of the time they put the phone on the bed and the time they took it off, allowing to correctly label collected data. Figures 6.9 and 6.10 show examples of accelerometer readings respectively collected when the user is awake and when she has gone to sleep and has put her phone face down under her pillow, showing that they are qualitatively different (in the figures the Z axis points towards the outside of the front face of the screen). It is important to notice that even if the phone is left face-down on a flat surface, as in the first 10 minutes inf Figure 6.9, acceleration values are much less noisy than values collected when the phone is in a bed, that gives us confidence that it should be possible to train an accurate classifier.

We then extracted characterizing features from accelerometer data. Feature extraction requires deciding the time window in which extract features. A small time windows allows to be more *precise* when running the classifier by reducing the granularity of classification results, for example a one hour long window allows to infer at what time the user went to bed at most within a one hour precision, whereas a one minute window potentially allows to be much more precise. However, a too small time window may make the classifier less *accurate*, in other words smaller time windows are easier to misclassify. We empirically decided to divide sensed data in 5-minutes long windows that are long enough to present significant features but not too short to boost misclassification. The dataset samples were extracted using a 5-minutes long window with a 3-minute step (i.e., the first dataset sample goes from minute 1 to minute 5, the second sample goes from minute 3 to minute 8, the third sample goes from minute 6 to minute 11, and so on), resulting in more than twelve thousand samples that were then used for feature extraction. For the sake of simplicity, in the following we will refer to samples collected while a phone is face-down on a mattress as *sleeping* samples and to other

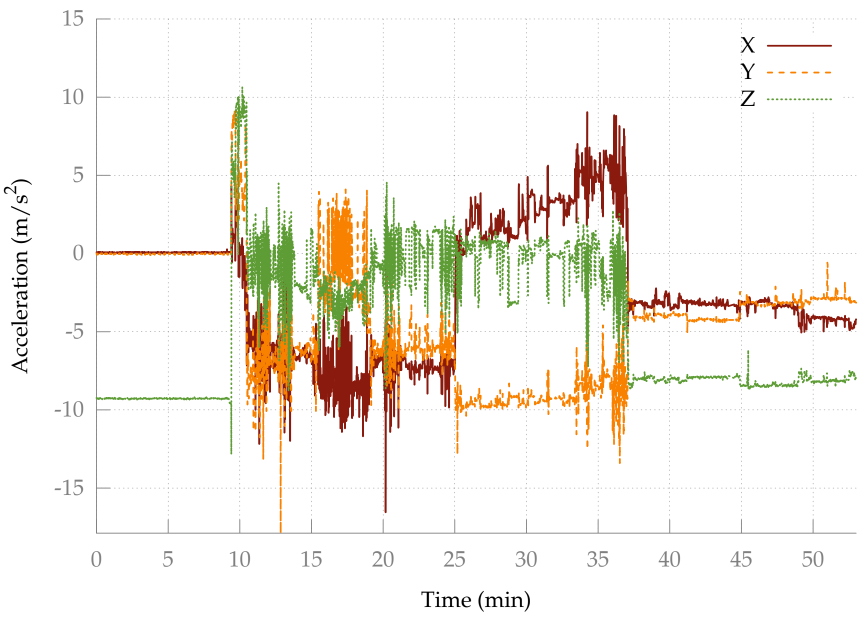


Figure 6.9: Accelerometer readings over one hour when the user is awake. First the phone is resting face-down on a table, then the user picks it up and walks while keeping it in a pocket until the 25 minutes mark, then he rides a bicycle for about fifteen minutes and then sits at a table.

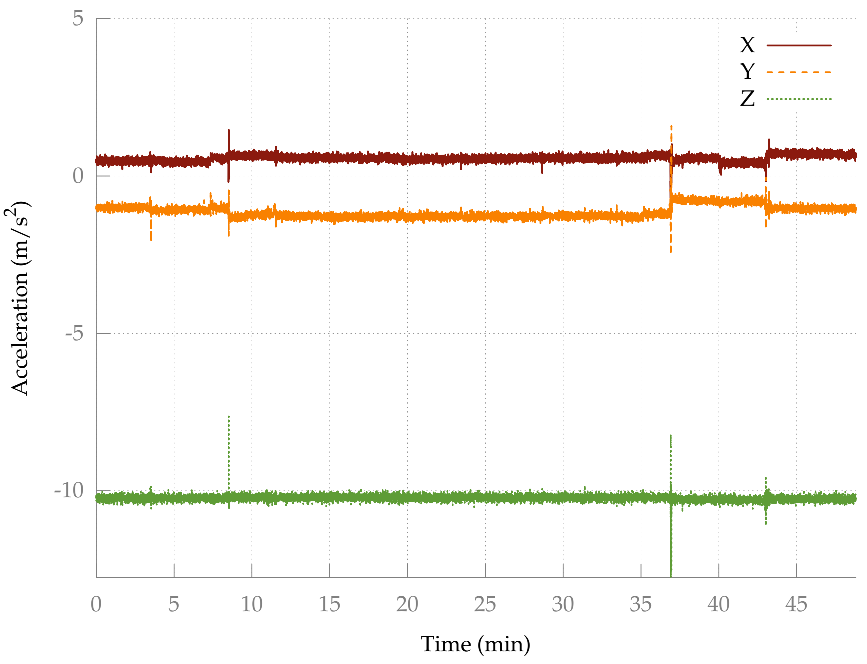


Figure 6.10: Accelerometer readings over one hour while the user is in her bed. Sporadic peaks are caused by the user rolling over.

samples as *awake* samples, even if neither of them conclusively proves that the user is asleep or awake.

Features to extract from a signal in a time windows can be time-domain or frequency domain; however, since our collected data about sleep does not seem to contain periodic components that we are interested in, we chose to use time-domain features only because they are not computationally intensive. In particular, for each five minutes window we extract for each accelerometer axis:

- average;
- maximum;
- minimum;
- standard deviation;
- Root Mean Square (RMS).

The strong accelerations experienced by a smartphone when the user is awake, as shown in Figure 6.9, can be easily detected by average, maximum and minimum values for each time window. Telling apart when the phone is face-down on a flat surface (e.g., on a table, as shown in the first ten minutes in Figure 6.9) from when it is in a bed is more difficult; but it can be picked up by the offset between accelerations over X and Y axes and by the more noisy values, as measured by standard deviation and RMS.

Finally, we used the Weka machine learning software [202] to train the actual model for sleep detection. We tested several industry-standard algorithms: C4.5 decision tree [203], Support Vector Machine (SVM) [204], Bayesian Network Classifier [205], and random forests [206]. All the resulting models had similar performances; we decided to use the C4.5 decision tree because it is simpler to implement without depending on the Weka library. The resulting decision, notwithstanding the large size of the dataset, is rather compact having an height of 4, 11 nodes, and 6, which suggests that it does not overfit the training data. This intuition is proved correct in Section 6.6.2 that reports the performances of the sleep classifier.

The resulting classification model has been wrapped in a MoST pipeline. The sleep pipeline receives accelerometer data from the Bus and processes it to extract on-line the features that we used to train the classifier. After five minutes of data processing it passes resulting features to the decision tree that classifies them as either *awake* or *sleeping*. The output of the decision tree is then sent to the Dispatcher, that notifies all components interested in sleep classification. The resulting pipeline is used by BeWell, but as part of MoST it can be easily reused by any other application (e.g., sleep trackers).

6.6 EXPERIMENTAL RESULTS

This section presents MoST implementation and shows an interesting selection of experimental results that compare MoST with the Funf framework that is the closest one available in state-of-the-art literature [190]. In addition, it evaluates inference performances of the sleep classifier described in Section 6.5.

6.6.1 MoST performances

MoST has been realized as a self-contained app that runs on Android platform and is compatible with version 2.2 up to version 4.2. The current MoST implementation includes Input objects for the following sensors: accelerometer, microphone, magnetometer, gyroscope, and light sensor; in addition, it provides two Pipelines, one based on accelerometer data and one based on audio data. The accelerometer Pipeline is the one described in the previous section that detects if the user is actively carrying the phone or if she is resting and has put the phone under her pillow, by running a classification algorithm that analyzes maximum, minimum, average, standard deviation, and root mean square over the three accelerometer axes. The audio Pipeline recognizes human voice based on some time-domain and frequency-domain features typically considered in the related literature, namely, L1-norm, L2-norm, L-inf norm, Fast Fourier Transform, power spectral density across five different band ranges, and Mel-frequency cepstral coefficients [25, 101, 141]. These pipelines are representative of real world workloads, because similar functionalities have been used by existing works based on continuous mobile sensing.

We compared MoST performances for these two Pipelines, with other two implementations of the same Pipelines: a *native* Android and a *Funf* solution. The native solution does not rely on any external library and runs the barebone minimum code to perform sensor sampling and the feature extraction of audio: this solution represents the reference to evaluate memory and CPU overhead of other approaches, namely MoST and Funf. Funf-based solution, instead, uses two probes (data sources), namely, *AccelerometerSensorProbe* and *AudioFeaturesProbe*, that implement the same MoST sensing chain for the two considered Pipelines. In order to make the test fair and comparable, we disabled the Funf default feature that dumps all data to a local database because it slows down and worsen system performances, especially for high sensing frequencies. Let us also stress that the audio feature extraction code that is the most CPU intensive task has been coded exactly in the same way for all the compared solutions.

We tested the three implementations (native, Funf, and MoST) using different sensor sampling frequencies. Audio was tested at 8kHz and 44kHz sampling frequency, while accelerometer was tested us-

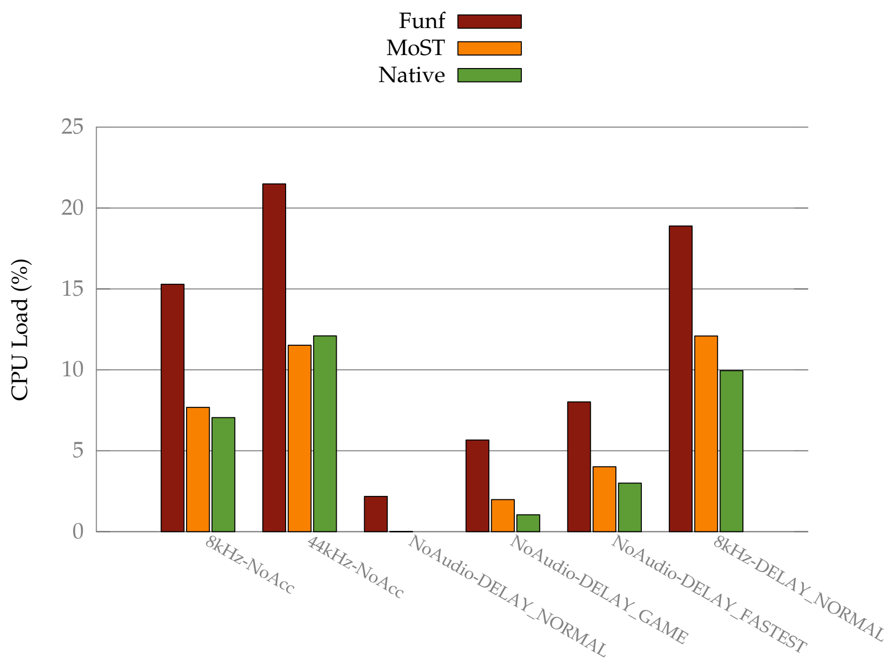


Figure 6.11: CPU usage for Funf, MoST, and native implementations.

ing the three sampling frequencies made available by standard Android APIs, from lowest to highest frequency: `SENSOR_DELAY_NORMAL`, `SENSOR_DELAY_GAME`, and `SENSOR_DELAY_FASTEST`. All test have been run on a Samsung I9100 Android device featuring a dual core ARM Cortex-A9 processor running at 1.2 GHz and 1 GB RAM.

Our first set of experiments measures the introduced overhead in terms of CPU. Figure 6.11 shows obtained average CPU usage for each test setting, for different sampling frequencies and with different active Pipelines (only audio, only accelerometer, both); each experiment has been repeated 33 times and black vertical error bars report 95% confidence intervals. In general, the results show that MoST has a very little overhead compared to native, barebone, implementation, thanks to its careful management of resources. Focusing on the second test that processes audio at 44kHz, the MoST CPU usage is even smaller than that of the native solution one: we believe that is because the thrifty resource management of MoST recycles many of the internal objects and especially the byte arrays that store audio samples wrapped in DataBundles; whereas, the simple solution creates a lot of new objects for each sensing cycle that have to be routinely freed by the Garbage Collector, thus causing increased CPU usage. That effect is more noticeable when audio is sampled at 44 kHz because that high frequency stresses more the sensing code. The last and sixth test, instead, that realistically simulates the accelerometer and microphone being sampled at the same time, shows the very low overhead of MoST: the barebone native solution causes 9.9% CPU load on average,

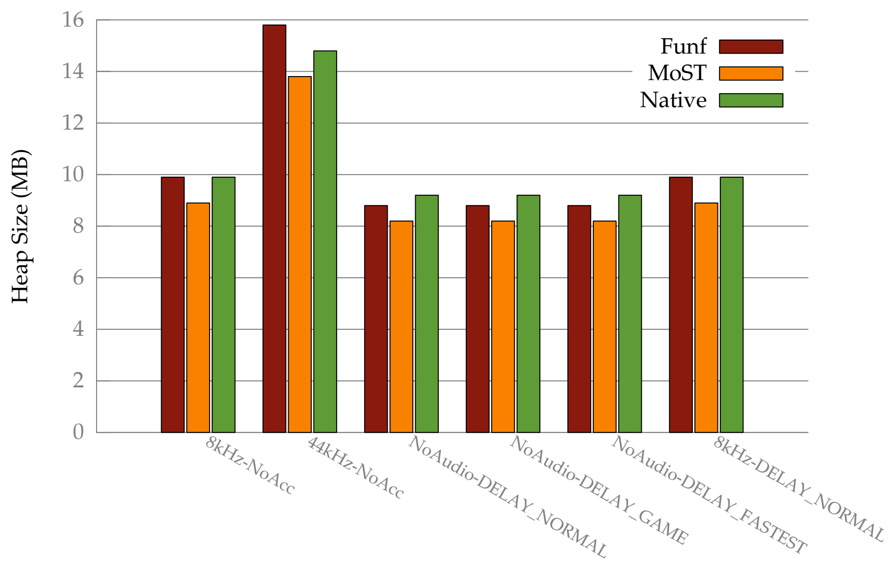


Figure 6.12: Heap memory usage for Funf, MoST, and native implementations.

while MoST takes 11.8%, only 1.9% more. On the other hand, Funf CPU load is 18.9%, almost the double compared to the cheapest solution: that big difference mainly is due to internal Funf architecture that does not pool objects by completely relying on Android Bundle objects that are easy to use, but introduce very high overhead [207].

Our second set of experimental results assesses, under the same experimental conditions, memory usage for MoST, Funf, and native Android; in particular, we have used the heap dump feature of the Dalvik Debug Monitor Server (DDMS) provided by the Android Software Development Kit (SDK) that takes a snapshot of current heap status for a running application by describing the live set of objects allocated at the moment the snapshot was triggered. Figure 6.12 shows the heap size in each test settings and highlights the very good performance of MoST: Funf and native Android solution have a very similar approach to memory management based on creating new objects for each sensor sampling and letting the garbage collector remove them, thus they have very similar heap usages; MoST optimized object pooling, instead, significantly reduces the average heap size by up to 2 MB compared to them. This good improvement over other solutions is even more important considered the strict heap size constraints that Android enforces by default: on lower-end smartphones the maximum allowed heap size is as low as 16MB; hence, MoST frees valuable memory resources that can be more fruitfully exploited to increase activity inference tasks. We also compared MoST heap footprint with the one of an empty Android application (not shown in Figure 6.12) that, on our test device, was 8.1 MB; as Figure 6.12 shows, apart when dealing with high quality audio, MoST footprint is always very close

Table 6.1: Details of the accuracy of the sleep classifier.

	TP rate	FP rate	Precision	Recall	Class
	0.964	0.076	0.965	0.964	Awake
	0.924	0.036	0.922	0.924	Sleeping
Weighted average	0.951	0.063	0.951	0.951	

to that value thus confirming the limited overhead introduced by our framework.

6.6.2 Sleep Classifier Performances

We trained and tested the sleep classifier using ten-fold cross-validation as implemented by Weka. The classifier correctly identified 12018 samples out of 12633, i.e., it correctly classified 95.13% of the samples. Table 6.1 shows more in depth details about the accuracy of the classifier, reporting them for the two classes that identifies and averaging them based on the relative size of samples labelled as *awake* and as *sleeping*. The True Positive (TP) *rate* column shows the rate of TP samples (i.e., correctly classified samples), while the False Positive (FP) *rate* reports the rate of FP samples (i.e., incorrectly classified samples); the high TP rate (95.1%) and low FP rate (6.3%) prove that the classifier can reliably infer whether the phone is face-down on a bed or not. It is interesting to note that the FP rate of the *awake* class is higher than the FP rate of the *sleeping* class, in other words it is easier to misclassify a time window as *awake* even if actually the phone on a bed (for example due to strong acceleration) than the other way around. The misclassificated *sleeping* samples have strong acceleration, caused by users vigorously rolling over in their bed, that make the classifier assume that the phone was picked up.

An additional way to assess the performance of a binary classifier, such as our sleep classifier, is ROC curve. ROC curve shows how TP rate and FP rate are affected by varying the probability threshold of the classifier above which a sample is assigned to that class. Figure 6.13 shows the ROC curve for the sleep classifier for the *awake* class. The ROC curve of a perfect classifier is represented by a single point at coordinate $(0, 1)$ that means that the classifier has zero false positives, i.e., it does not classify any sample of the *sleeping* class as *awake*, and has 100% true positives, i.e., all *awake* samples are correctly classified. The diagonal line that goes from $(0, 0)$ to $(1, 1)$, called *line of no-discrimination*, represents the behavior of a classifier that randomly guesses the class of each instance (e.g., by throwing a fair coin). The colored line in Figure 6.13 is the ROC curve of our classifier. When the line is dark red the probability threshold to classify a sample as *awake*

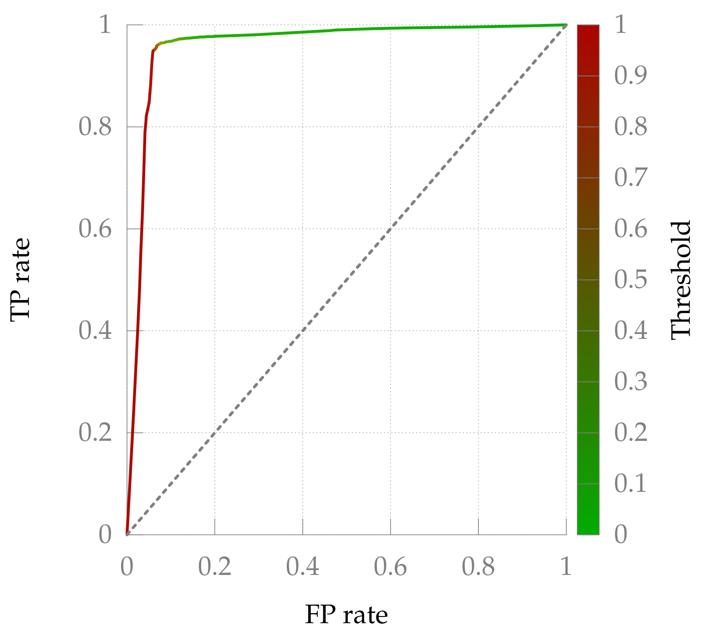


Figure 6.13: ROC curve of the sleep classifier for the *awake* class. The gray dashed line is the line of no-discrimination.

is close to 1, hence a sample is classified as *awake* only if the classifier is very confident of its output, causing low FP rate and TP rate, because most samples are rejected. As the line hue becomes green, the probability threshold goes to 0, hence every sample is classified as *awake* because most of them are accepted regardless of the classification confidence. The evaluation of the ROC curve confirms that the sleep classifier performs much better than chance and it is actually close to an ideal classifier.

From the practical point of view, the sleep time estimated by the classifier has an precision of ±40 minutes compared to the actual sleep time per day, thus it is a good improvement compared to the ±1.5 hours error of the best effort classifier originally implemented in BeWell.

6.7 CHAPTER CONCLUSIONS

This chapter presented BeWell, a wellness app for Android smartphones. In particular, we focused on its core sensing library MoST, a general-purpose high-performance framework for mobile sensing. The modular design of MoST presents two main architectural advantages over existing solutions: it makes it easy for application-level developers to exploit the intelligent MoST inferences and allows signal processing and machine-learning experts to quickly add new Pipelines without having to deal with all additional technical complexities and details of mobile sensing. We also described a pipeline based on accelerometer that exploits simple time-domain features to accurately and precisely estimate whether the user is laying on his bed or not. The inferences that it outputs are being used by BeWell, but they are also available to any other application based on MoST.

In conclusion, MoST design choices have allowed very efficient memory and CPU management, as stressed by experimental performances, that make MoST a powerful, easy-to-use, and flexible mobile sensing platform.

7 SMARTPHONES AS SOCIAL SENSORS: RECOMMENDATION AND PARTICIPATIVE SENSING

So far we have shown how the logical model proposed in Chapter 3 allows to model *data-centric systems* that are focused on data collection for environmental monitoring, and *person-centric systems* that run inferencing algorithms on collected data, such as the MoST library, to provide to users accurate high-level data that is relevant to their everyday activities. We call *social-centric systems* those Pervasive Sensing system that further scale up the usage of sensing from single persons to social communities.

In the scenario of socially relevant sensing applications, human communities can participate as simple recipients of services based on collected data and as collectors themselves. In the former case social communities are exploited as collective intelligence agents to provide users with services based on crowdsensed data. In the latter case people voluntarily participate in data collection campaigns, encouraged with monetary or symbolical goods [18, 19]. In this Chapter we present two projects that represent the two cases, called SAIR and McSense: SAIR focuses on providing mobile software recommendations based on social preferences and crowdsensed usage context, while McSense on managing crowdsensing tasks.

7.1 SAIR: SOCIO-TECHNICAL AWARE RECOMMENDATION

The sensing capabilities available on smartphones (Section 2.1) represent the convergence of computing, social, technical and physical worlds: they can access typical Internet resources such as web sites and web services, they can retrieve user provided data from social networks, they can gather technical measurements about service usage and they can infer user activities. What is still missing is a general support platform able to truly enrich the whole mobile service management cycle to fully exploit the power of the collective (though imprecise) socio-technical monitoring information deposit to enhance service delivery from both social (recommendation of services of interest) and technical (resource and data delivery optimization) points of view so to improve the Quality of Experience (QoE) for final users.

IP Multimedia Subsystem (IMS) (Section 5.2), based on the Session Initiation Protocol (SIP), has recently gained success as the session control protocol for application delivery over all-IP next generation networks. IMS specifies a simple and powerful service delivery platform and service composing model, and defines the Service Capabilities Interaction Management (SCIM) component to manage integration and composition of IMS service blocks [208]. At its current stage, however, IMS still exhibits limitations in the support of integrated monitoring and service composition/recommendation management. On the one hand, the possibility to automatically maintain lists of more useful and widely used mobile services built around user needs, social behaviors, and current activities is still widely unexplored in IMS. On the other hand, SCIM is still a raw composer component without much intelligence: in other words, SCIM is unable to dynamically exploit socio-technical monitoring information about people, service, and network usage dynamically (re-)adapt the delivery of composed mobile multimedia services.

This section addresses all the above issues by proposing a novel solution that exhibits several original characteristics. First, it exploits a very lightweight component mobile that nodes can deploy to monitor socio-technical information in three main areas: user physical location and activity (running, driving, sit down in the office, . . .), user social context (friends, common social interests, application used by friends, . . .), and service usage (frequency of use, time from last use, messages exchanged, . . .). Second, it proposes a novel service recommendation system that applies scoring and social matching techniques to distill user common interests and service usage pattern similarities from monitored data. Third, it presents a new IMS-based SCIM implementation that uses workflow management abstractions and tools to ease the mobile service composition process and to enable dynamic adaption and tuning of service delivery at runtime. Finally, our proposal, called Social-Aware IMS-enabled Recommender (SAIR), is fully compliant with the IMS standard and does not require any change on already installed IMS equipment. SAIR greatly advances the related work in the field of major flexibility and leveraging service usage information [209, 210].

In the following we overview background and related work about the main research areas needed for a full comprehension of our proposal. Then, we motivate the need for the SAIR platform and detail its distributed architecture and main components in Section 7.1.2. Section 7.1.3 presents SAIR platform implementation and experimental results.

7.1.1 Background and State of the Art

SAIR operates at the intersection among socio-technical monitoring, service recommendation, and IMS-based service management platforms for Future Networks. This section gives some technical background and briefly surveys the current state-of-the-art in these fields.

Socio-technical monitoring of service usage and social context

Traditional monitoring metrics include technical information to evaluate service provisioning quality at the system level, such as bandwidth usage, latency, CPU and memory load. More recently, various research efforts on service recommendation are trying to capture and measure also service usage through higher-level, but still rather social-agnostic, user engagement indicators. A widely diffused and effective metric is Recency, Frequency, Duration (RFD) that is a variation of the Recency, Frequency, Monetary (RFM) marketing model widely used to evaluate loyalty of customers [211]. The RFD score monitors and quantifies service usage as a weighted sum of: *Recency*, measuring the time elapsed since the last time the service was used; *Duration*, namely the fraction of time spent using the service; and *Frequency*, the number of times that the service was accessed in a given period.

Focusing on *social information*, the analysis of user social context permits to infer interesting data about user interests via information provided spontaneously by users, and analyzing behavior and habits of his social network. Therefore, several research efforts in this area proposed and studied novel methods and algorithms that analyze social networks and contents to understand and to extract automatically from that social information deposit users relationships, interests, and even their mood [212]; the main problem of those approaches approach is that they are prone to errors and imprecision. Therefore, some social network market players, such as Twitter and Google+, have recently started to explore more pragmatic social context characterization solutions; they encourage their users to classify their input using the so called hashtags, namely, keywords marked in user messages by the # symbol (e.g., #IEEE). The goal is to improve the accuracy of searches on past messages and to ease the aggregation of similar messages; in fact, relationships between hashtags are simpler to infer compared to raw text, and they have already been successfully exploited for content classification and recommendation [213].

The monitoring of *activities* in which users are involved while accessing a service, including both user physical location and current activity, is another good indicator of service usefulness. For the background on activity detection we refer readers to Sections 2.1 and 6.3.

Recommendation Systems

The explosive growth of mobile services and applications numbers makes almost unfeasible for users to browse all related resources tailored to their specific needs: recommendation systems improve that situation by collecting information about application usage from previous users and by using that context data to automatically select and recommend applications with minimal user intervention. Previous work in this area has shown that mixing together monitoring data about service usage, user activities and social interests, allows recommendations very valuable for final users. The two most important trends in recommendation are collaborative and social filtering: *collaborative filtering* is more social-agnostic and based on previous service usage and activity indicators; *social filtering*, instead, privileges and looks for social information about users, services, and their social interconnections.

A seminal research effort in collaborative filtering for services is AppJoy, that automatically rates user interest for a specific application based RFD score and suggests to users applications used by other people with a similar application usage pattern [214]. Woerndl *et al.* proposed to use location information to recommend applications, by observing that the same applications will be used in the same place (e.g., a bus timetable application will be used near bus stops); thus, on the basis of geographical location and proximity to points of interest ranked by user-provided score, the system scores application usefulness and recommendations [215]. Focusing on social filtering, the key observation of this research trend is that social networks link the user with people that are like-minded or that she shares some interests with; thus, the content contributed by people linked to her indirectly provide information about the user herself. For example interests, activities and tags of users in the social network of a user can be integrated in recommendation systems by modeling them as a graph and using well known techniques, such as random walks, to predict common interests [216].

Let us stress that while application recommendation systems, both collaborative and social filtering based, have already been tackled in the literature, the possibility to leverage socio-technical monitoring information not only for the benefit of final users, but of the service provider and network efficiency too is still widely unexplored. In particular, we are convinced that knowing when, where, how, and by whom the services are used will lead to improve the efficiency of service provisioning. For example, a video streaming service could take advantage of mobile nodes to relay the stream other nodes in their vicinity, knowing in advance from common socio-technical profiles that those users are interested in the same service, content distribution could be partially off loaded on the client side thus saving bandwidth and guaranteeing overall better performances such as [217].

These services, that we refer in the following as Mobile P2P (MP2P) because they are able to take advantage of more decentralized and mixed service operation modes, are likely to alleviate cellular network usage, especially in highly dense and overcrowded situations such as exhibitions, Olympics games, and so forth, through the usage of local impromptu wireless ad-hoc interactions. Hence, we claim the importance to evolve recommendation systems to suggest those classes of services whenever possible according to current socio-technical context.

IMS Platform

The deployment, provisioning, and management of services over heterogeneous wireless infrastructures is still a challenging issue. To tackle the problem, a large group of standardization entities has defined the IP Multimedia Subsystem (IMS) framework, that provides a homogeneous access to multimedia and voice applications, without imposing a standard structure to services that use it [128]. The main components of IMS have already been presented in Section 5.2; in this section we present additional details about integrating additional applications running in IMS.

IMS services are run and hosted on the Application Server (AS) and they can be orchestrated to provide more complex, feature-rich services. IMS composite services integrate multiple services that act on a single user session and that are organized in a workflow without the need to modify their own service interfaces [128, 218]. In particular, IMS specifications say that service composition and orchestration is delegated to the Service Capabilities Interaction Management (SCIM), however there are no guidelines on what its responsibilities are and how it should be structured, thus so far SCIM has been open to various implementation and researches [208].

We conclude this section with an overview of seminal research efforts done on SCIM and dynamic composition of services in IMS-based environments, mostly focusing on easing management of composite services and on providing added value to service providers. Komorita *et al.* proposed in [209] a framework that allows users to ask to service providers a specific sequence of loosely coupled services (e.g.: to transcode and compress a video stream), that are dynamically chained to provide the desired service. Jianxin *et al.* designed a multi-tier architecture for next generation networks that supports service composition and can exploit it to optimize content delivery [210]. However, those works design solely infrastructure-side architectures and are targeted at optimizing technological aspects only and do not consider socio-technical recommendation aspects of service management, such as performance improvements and increased user satisfaction.

7.1.2 Social-driven data elaboration for service composition and recommendation

SAIR is a complete framework that leverages environmental and social context to suggest services to users, and allows them to easily build new services based on existing ones to better suit their needs. In fact, advanced users could be willing to customize a service composing basic IMS AS building blocks to create new composite service. Just to give a very simple example, a user could personalize her phone calling service as follows: she can receive both audio and and video streaming when she is at home, as audio streams only when walking and even downsized as text messages when working and in a "busy" state. The possibility to introduce new composite services paves the way to new social-aware mobile service scenarios where the service ecosystem can autonomously evolve and recommend to people with similar lifestyles and social/activity patterns most popular and valued services. In general, this services composition and social sharing requires: access to context information (e.g., current activity) for both service composition and suggestion; a mechanism for service recommendation; and an interface for the simple definition of new composite services. In the following, we describe how these three core functionalities are distributed between mobile devices and backend infrastructure, and how context information and service usage data is processed for service recommendation.

Distributed Architecture

SAIR architecture is organized in three layers: Service Configuration and Usage Monitoring layer, Infrastructure ayer, and Service Recommendation layer (see Figure 7.1).

The Service Configuration and Usage Monitoring layer consists of a lightweight software stub (SAIR stub) running on mobile user devices that associates context to service usage; compared to the logical architecture presented in Chapter 3 it realizes the Mobile Infrastructure Tier by implementing the Sensing, Data Processing, and Peer Coordination components (Section 3.2.2). The Service Configuration and Usage Monitoring layer also includes a web interface that allows to compose services, that can be considered part of the Data Processing component of the Fixed Infrastructure Tier. The Infrastructure layer of SAIR contains IMS and its satellite services, such as the IMS Presence Service (PS), that run IMS services. They provide the cross-tier communication interface between mobile nodes and the fixed infrastructure, that we called *Lower-tier interface*. The Infrastructure layer also contains UniboSCIM, our custom component that enables service composition, that logically implements Data Processing mechanisms of the Fixed Infrastructure Tier (Section 3.2.3). Finally, the Service Layer contains all the SAIR components that manage

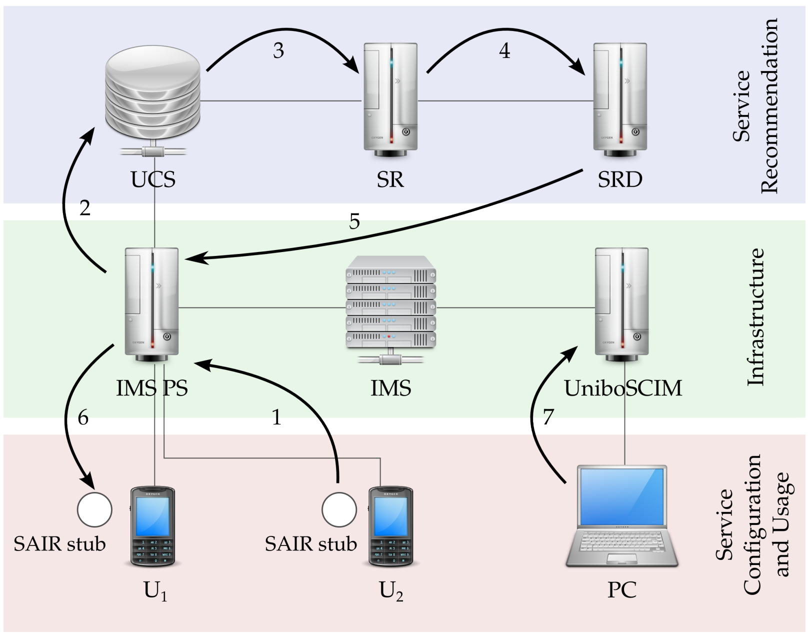


Figure 7.1: SAIR distributed architecture and data flow.

and process application context and are part of the Fixed Infrastructure Tier: the User Context Storage (UCS) maintains user and service socio-technical context (Data Storage component), the Service Ranker (SR) sorts services by different socio-technical monitoring dimensions (Data Processing component), and the Service Recommendation Dispatcher (SRD) builds the final recommendations and notifies them to users (Data Processing component). In the following, for sake of understanding, we describe the data flow that spans these three layers to sustain SAIR core functionalities in Figure 7.1.

Service Usage and Physical and Social Context Data Collection

SAIR stub is the core monitoring component: it logs service usage and detects applications that are in the foreground and are actively used. Then, it links application usage with physical and social context and dispatches those monitoring data to the infrastructure components (Figure 7.1, step 1).

Activity context includes both physical location and user activity and is obtained in completely decentralized way by exploiting currently available local smart phone sensors. GPS provides the precise location of the user; when GPS is not available, the SAIR stub switches to other wireless-based localization techniques with lower precision, such as Bluetooth co-location and WiFi triangulation techniques. Raw coordinates are useful to evaluate user spatial vicinity and are also exploited to determine whether the user is close to a specific place (e.g., a restaurant, a stadium) by using specialized APIs, such as those provided by foursquare.com. Accelerometer and microphone can provide very accurate information about physical activities; in particular, SAIR uses the same library that enables activity detection in BeWell (Chapter 6).

To get social context, SAIR requests user permission to access user personal data on social networks and considers hashtags as privileged social context information, thus avoiding complicate text processing. In the current implementation, SAIR can collect and process hashtags (called tags from now on) from Google+ and Twitter, via their respective APIs. The basic idea is to obtain a broad indication of the links between service usage and social tagging, by associating each running application to all the tags generated by users within a time window that contains the service usage period. The significance of this simple association, although initially imprecise, increases when the number of contributing people grows: the more tags associated to a service pour in, the higher the probability that more frequent tags are likely to be the ones to consider.

Transmission of service usage and context is managed by the IMS PS: SAIR stub acts as a presentity and pushes collected monitoring data to PS that dispatches them to interested receivers via the UCS (see Figure 7.1, steps 1 and 2).

Service Recommendation

For service recommendation SAIR follows a novel hybrid approach that aims to take the best of both collaborative and social filtering methods, but taking into account also some additional technical information. The proposed recommendation method consists of three main phases as shown in Figure 7.2: the first one collects socio-technical context data at the mobile side; the second one ranks the applications according to socio-technical dimensions; and the third phase melds those ranks together to obtain the final recommendation for final users.

Looking closer to more technical details, SAIR stub, SR, and SRD work in a pipeline where each stage exploits output from previous one to output data on its turn over the next stage. In the first phase, SAIR stub collects and pushes monitoring data to UCS as described in the previous section. In the second one, SR processes incoming data: it clusters services usage by geographical area, by tags, and by physical activity, and then it uses RFD to determine which services are most relevant within each cluster. SR processes all data of all users and, for each user, processes all data of his related friends only; in other words, that approach builds two pools of service usage data, one based on all users suitable for default recommendations for general population, and another one based only on data collected from people in a social relationship with the user (the higher the sociality of the user the better social-aware recommendations and obtained QoE).

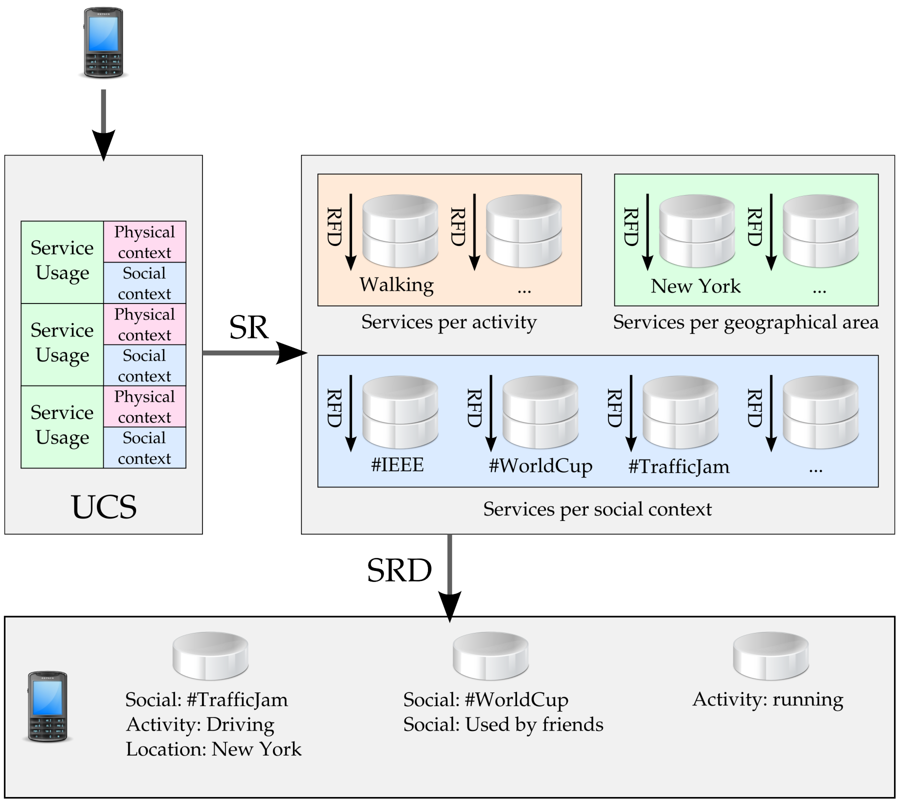


Figure 7.2: Recommendation process phases: gathering of context associated to service usage, clustering and ranking of services, and dispatching of recommendations.

We define the RFD score as the sum of three scores (s_R, s_F, and s_D), each one normalized in a $[0, 1]$ interval: $RFD = s_R + s_F + s_D$. The s_R score measures how recently the service has been used, by weighting less services that have not been used lately; more formally, s_R is the multiplicative inverse of the time elapsed since the last time the service was used, i.e., $s_R = \frac{1}{time\_interval\_since\_last\_service\_use}$. The s_F score, instead, fixes a time window and an upper threshold of service use frequency in that window (number of times a Highly Frequently Used Service (HFUS) is launched in the time period), scores the service 1 when it has been used more frequently than that threshold, and $s_F = \frac{number\_of\_service\_launches}{number\_of\_HFUS\_launches\_upper\_bound}$ otherwise. The s_D definition is similar to the s_F one, but while s_F focuses on frequency, s_D considers the total time sent using a service in the time window; in brief, a fixed time window and an upper threshold for the total service use duration of a Highly Used Service (HUS); $s_D = 1$ when the total service usage time is more than the upper duration time threshold of HUS, and $s_D = \frac{service\_usage\_total\_time}{HUS\_usage_total\_time\_upper\_threshold}$ otherwise. Then SR weights RFD scores with the popularity of the service: a service used by more people will rank higher than a less popular service with the same RFD score. Thus, the score of service s_1 will be $RFD_{s_1} = \frac{\#users\_of\_s_1}{\#users}\left(s_{R_{S_1}} + s_{F_{S_1}} + s_{R_{D_1}}\right)$; finally, for ease of comparison for the final user, SR normalized the obtained scores in the $[0, 100]$ interval.

The third and last phase is executed by SRD and it is the most articulated and value-added one. SRD rank cycle consists of four steps: getting user socio-technical context; matching user context with services clusters ranked by the SR; identifying services whose usage is favorable for the service provider and boost their scores; and finally notifying results to the user. In the first step, the SRD filters from the UCS the clusters corresponding to current social and activity context of the user: for instance, let us assume that a user is walking in New York City and recently wrote a tweet about an "#IEEE" (see step 1 in Figure 7.3). In particular, for each context dimension, SRD looks up in the SR outputs which are N top ranked applications (in Figure 7.3 $N = 2$), as scored by RFD. Then, SRD chooses as topmost services to recommend first those ones that appear more frequently across different clusters and, for those that appear with the same frequency, it ranks higher the ones with higher RFD scores. Hence, in the example shown in Figure 7.3, SRD would rank services in the following order (not shown in Figure 7.3): A, I, H, B and K starting with A, the one most frequent one across different clusters, and assuming $RFD_I > RFD_H > RFD_B > RFD_K$.

When SRD has associated to the user a ranked list of services of potential interest, in the third step, it adjusts the RFD score of those services that the service provider is interested in promoting, such as MP2P services for an overcrowded provisioning area that present a

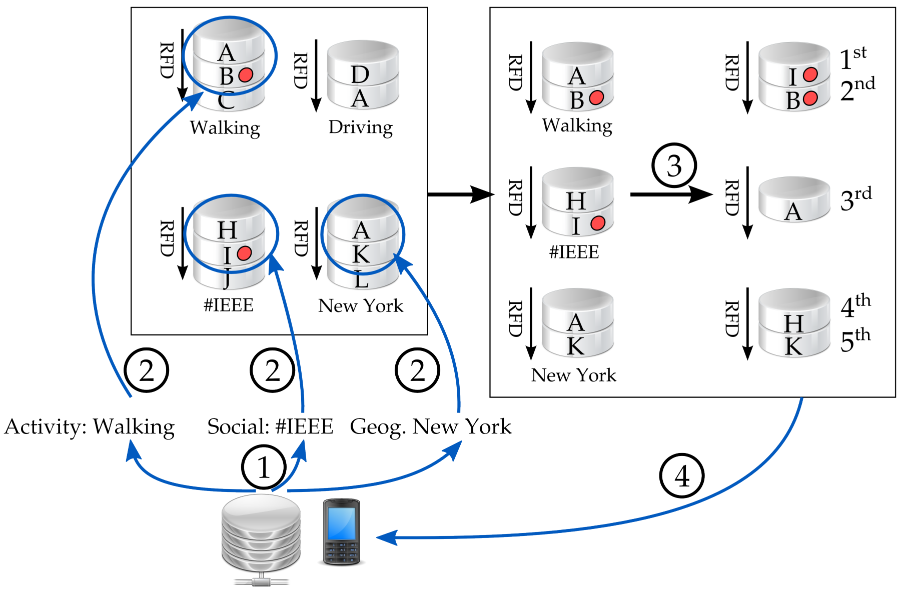


Figure 7.3: Service recommendation steps: MP2P services are marked with a red dot. To reduce figure complexity, SR output from user friends is not shown.

sudden technical quality indicators drop. In that case, SRD looks for MP2P services in the temporary recommendation list, and looks how many people, close to each other in the same location, have recently used it. If there is a critical enough number, the SRD boosts the score of MP2P services to further increase their use, namely, services I and B that would be ranked lower otherwise, as shown in the final right-most rank in Figure 7.3. Finally, in the fourth and last step, IMS PS pushes the recommendations to users by concluding the work cycle.

User-driven Service Composition

IMS supports and provides already several services; however users may have sophisticated requests unlikely to be addressed by available service: let us recall the example of a user who wants to receive phone calls as audio stream only when her status is “at work”, as both audio and video stream when she is at home, and just a text message when her status is “busy”. Conceptually, this service is still a phone call service, but it is composed by simpler services that can be expressed by a simple workflow with a few branches and decision points.

To maximize usefulness of available services and to ease their composition, SAIR provides an intuitive user friendly Web application, that displays services as building blocks to be connected in a workflow by using a simple Graphical User Interfce (GUI) to build services based on existing ones, and so to make them immediately available. SAIR manages composed services in the same way as simple ones, thus they can get in the loop of recommendation system and can be automatically suggested to interested users. In addition, more expert

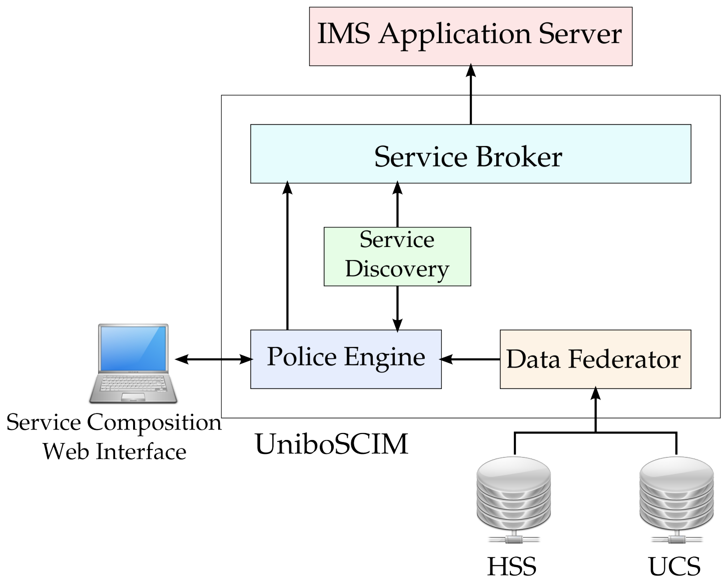


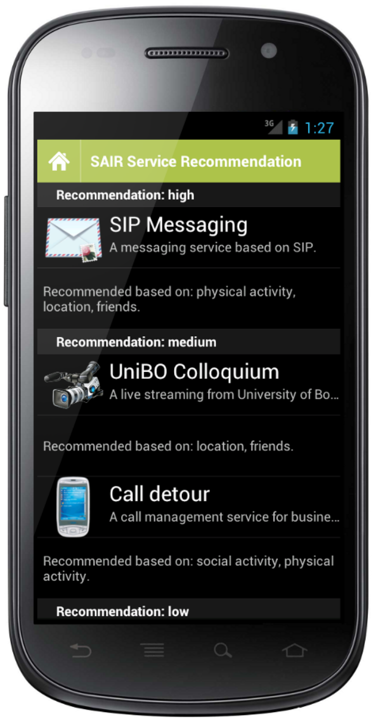
Figure 7.4: Internal architecture of UniboSCIM, black arrows represent main data flows.

users can also mark composite services as MP2P, further enhancing he possibilities of SAIR of leveraging social information to improve service provisioning and QoE.

Service composition is managed by our custom SCIM, called UniboSCIM, that does all the heavy lifting to provide an easy way to express composed services and to run them on IMS. UniboSCIM consists of four working blocks of Figure 7.5: the Data Federator, the Service Discovery, the Policy Engine and the Service Broker. *Data Federator* is the interface that allows context data retrieval (e.g., user status, user location, data in the HSS, and especially a console showing all socio-technical data store by UCS). *Service Discovery* finds available IMS services and makes them available for composition. *Policy Engine* translates the graphical description of composite services in a business process management description, retrieves context data from the *Data Federator* to correctly chain services and pushes the description of the service to run to the *Service Broker*. The *Service Broker* receives the description of the service, retrieves the details of the service via the Service Discovery and runs them using IMS session management mechanism, such as the IFC! (IFC!) [218].

7.1.3 Implementation and experimental Results

To evaluate the advantages and the challenging aspects of SAIR, we fully implemented several prototypes. The user layer software is based on Android 2.3 (Figure 7.5 shows SAIR Android-based user-friendly and intuitive interface used by final users) and leverages MoST for activity recognition; the infrastructure layer exploits the OpenIMSCore



1:27

SAIR Service Recommendation

Recommendation: high

SIP Messaging
A messaging service based on SIP.

Recommended based on: physical activity, location, friends.

Recommendation: medium

UniBO Colloquium
A live streaming from University of Bo...

Recommended based on: location, friends.

Call detour
A call management service for busine...

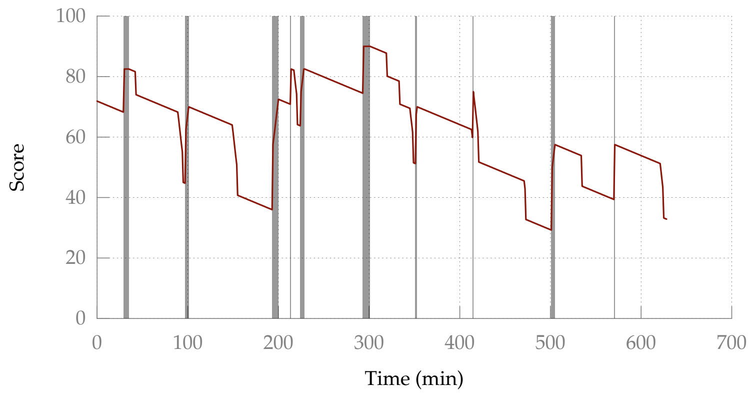
Recommended based on: social activity, physical activity.

Recommendation: low

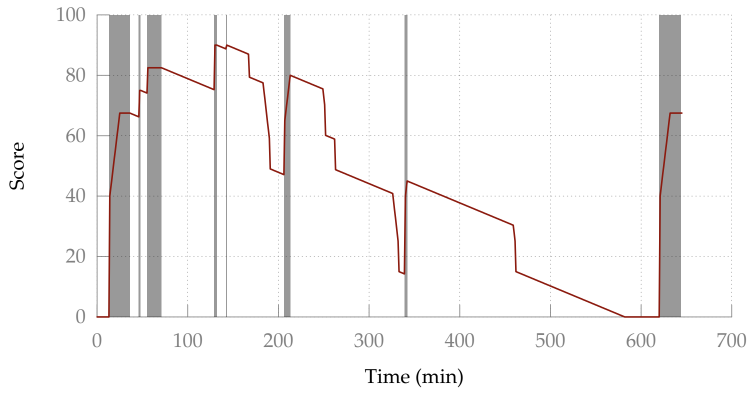
Figure 7.5: Google Nexus S phone running SAIR service recommendation application.

[168]; UniboSCIM has been developed using the Mobicents JAIN SLEE container [219] and uses the jBPM business process and workflow management suite [220]; finally, SR and SRD are custom Java-based servers. To assess SAIR quality, we evaluate mainly the user experience, such as the relevance of suggested services and the overall responsiveness of services. Thus, we have chosen three performance metrics: accuracy of the RFD scoring for service ranking, system load when the UniboSCIM aggregates services, and impact of running SAIR on mobile devices.

Figure shows representative RFD scores of a messaging (Figure 7.6a) and of a video streaming application (Figure 7.6b) during a working day, from 8 a.m. to 6:30 p.m. and the gray bars indicate service usage time intervals. First of all, we experimentally evaluated on the field the duration of the time window and of the frequency and the service usage duration upper thresholds defined in Subsection 7.1.2, and we found that 2 hours time span, 4 times, and 12 minutes (over that 2 hours period) are practical values that can be used for common, uniformly distributed, services. To ease the presentation, the graphs show the RFD score for the two applications used by a student. The video streaming application is used less frequently but for longer periods, namely, in the early morning when he is commuting, at lunchtime, and in the evening when he is back home; during those period the RFD score increases accordingly and that confirms the effectiveness of our approach since other people in similar situations

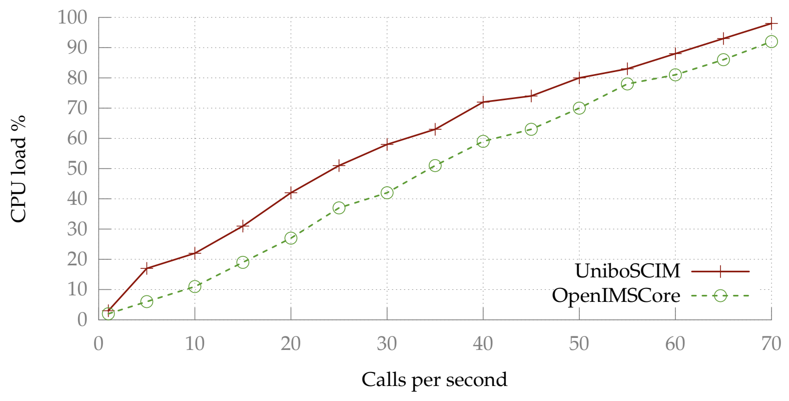


(a) Messaging service

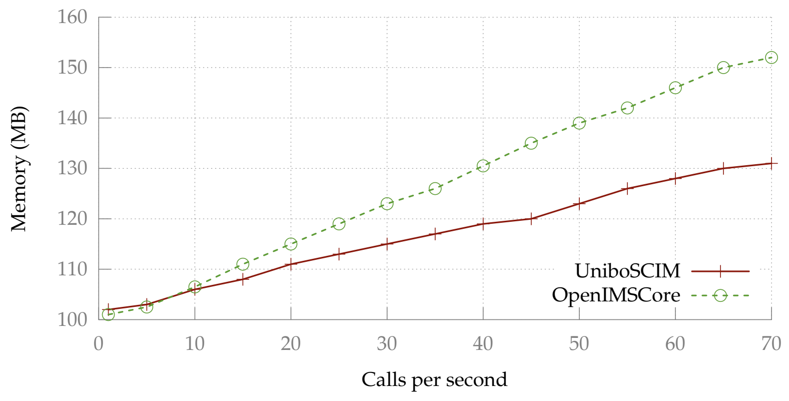


(b) Video streaming service

Figure 7.6: RFD score for two services over an interval of 10.5 hours (630 minutes) starting in the morning, when users go to work at 8 a.m., until evening, when they go back home at 6.30 p.m..



(a) CPU load.



(b) Memory load.

Figure 7.7: UniboSCIM load.

can take great advantage of this recommendation. Similarly, the messaging application is more recommended during the working hours, due to the higher frequently application use, although service usage is for shorter time intervals.

IMS services often have soft real-time requirements (e.g., phone calls), thus it is very important that the SAIR service composition does neither lower service performance nor reduce user QoE. To assess this aspect, we ran a stress test on UniboSCIM and IMS, to evaluate how many clients can be served and the latency in service provisioning caused by service composition. Figure 7.7a shows that the CPU load on an average server grows linearly with the number of calls per second, reaching the saturation at 70 calls per second: considered that the UniboSCIM in our deployment executes both the workflow engine and IMS session signaling processing on the same host, and according to performance evaluations about vertical handoff signaling handling in IMS [221], that is a realistic and satisfactory value. We also measured the time to set up a sample composited service: our test showed that OpenIMSCore takes about 610 ms, to which UniboSCIM adds a delay that, depending on the complexity of the composition, varies in the

300-600 ms range for a composite service consisting of three branches and starting up to three services at the same time (according to our experience, three branches is the usual complexity of real practical composite services). We can thus claim that the performance impact of UniboSCIM is acceptable because it introduces a delay that is definitely compatible with the usual total session set-up time [221].

We also tested the impact of off loading service provisioning on the mobile side of MP2P services. In particular, we tested bandwidth requirements of a simple video streaming service. We set up a video streaming service that streams a medium quality video requiring 220kbps and used concurrently by 5 mobile devices over a 3G network. A bandwidth measurement of the devices shows that this service takes 223kbps per device, for a total of 1.11Mbps on the cellular network corresponding to the bandwidth required by the video plus a control stream. Then, we set up one phone as WiFi access point (this function is natively supported by Android phones) to receive and re-broadcast the same video stream to other devices in the same location: using WiFi it is possible to offload the cellular network of a very significant 4.7 factor that can be improved further in more densely populated ares. This simple experiment shows that leveraging social knowledge for technical tuning of services can be good both for service provider and users.

We are aware that due to the complexity of SAIR, fully assessing its potentialities, accuracy, limits and performances requires further thorough testing, possibly over a demographically homogeneous group of people to encourage creation and sharing of composed services. We plan to test our system by asking a group of university students attending the same department, about fifty people aged between twenty and twenty-two years, to use SAIR for three months that will allow us to collecti performance statistics, user feedbacks, and raw context data. About performance statistics we plan to measure the computational, memory, and network loads caused by the deployment and, if we find bottlenecks in our system, using appropriate load-balancing techniques to improve the scalability of SAIR. User feedbacks will be used to evaluate the accuracy of suggestion, thus allowing to tune how context features are weighted to suggest applications. Finally, raw context data will be used to build a valuable dataset that, merged with users feedback, can be exploited to train and test alternative suggestion algorithm based on different approaches, such as pattern recognitions algorithms and logic-based knowledge basis.

7.2 MCSENSE: CROWDSENSING MANAGEMENT

SAIR shows that *social-centric* sensing applications that opportunistically gather data from users can provide valuable services by leveraging social sensing. However, users are not necessarily passive players in these category of applications, because they can willingly and actively collaborate toward continuous data harvesting process (*crowdsensing*). From a social perspective, there is the need to identify people willing to participate in urban sensing tasks and to find good incentives for participation, not only monetary rewards but also social ones (e.g. cleaner and safer cities). Once such people are identified, they have to be involved and kept in the crowdsensing loop, thus fostering people *participAction*. From a more technical perspective, one of the main challenges is finding, through a careful management of all involved resources, a good balance between system scalability and sensing accuracy for city-wide deployment environments.

At the current stage, although a few seminal works have started to consider how to facilitate the delivery of crowdsensing tasks and the collection of their results [38, 39, 222], much work is still to be done to characterize and manage the urban crowdsensing process itself due to several open issues. First of all, human behavior is inherently difficult to predict and calls for novel approaches based on probabilistic models. In addition, to design meaningful models, there is the need to run beforehand extensive experiments to collect large datasets by involving a high number of socio-technical resources for a long time duration. Finally, while some efforts have started to appear in the crowdsourcing research community to mathematically model and quantify socio-technical resources and the performance of accomplished crowdsourcing tasks (completion duration, ratio, etc.) in fixed Internet settings [223, 224], to the best of our knowledge this relevant effort is still missing in the mobile crowdsensing area, which is becoming of greater and greater importance due to the tremendous opportunities offered by people carrying today's smartphones.

We expand the proposal of SAIR, that does no elicit active user participation, with another project, called *McSense*. McSense is an ongoing project in collaboration with the New Jersey Institute of Technology (NJIT) that aims at studying the social and technical dynamics of crowdsensing. In this section, we focus more specifically on task management issues and propose a novel technical solution that exhibits several original characteristics. First, it proposes a novel geo-social model that statistically quantifies and describes, in a compact and easy-to-use way, how socio-technical resource availability varies in space and time, e.g., people density and vitality. The core idea is to build time-variant resource maps that could be used as a starting point for the design of crowdsensing participActions. Second, it studies and benchmarks different matching algorithms aimed to find, according to specific urban crowdsensing goals geo-localized in the urban environments, the "best" set of people to include in the collective participAction. Here the technical challenge is to find, for the specific geo-socially modeled region, the good dimensioning of number/profile of involved people and sensing accuracy. Third, it presents a new crowdsensing platform that consists of, on the one hand, an Android mobile app to ease crowdsensing tasks delivery/execution and collected results upload, and, on the other hand, an infrastructure-side server to manage crowdsensing tasks and associated incentives. We have validated the proposed solution over a large set of data collected for 2 months over a population of 44 people; we present both technical results about the accuracy/overhead of the employed matching algorithms and the results of a survey to assess user satisfaction about our crowdsensing mobile app usage and the received incentives. In short, the collected results show that, with relatively low socio-technical resource overhead, it is possible not only to achieve good sensing accuracy, but also to minimize monetary incentives necessary to drive user participation.

7.2.1 Background and State of the Art

ParticipAction services typically cross-cut and work at the intersection of two main research areas: mobile sensing and crowdsourcing techniques. The current state of the art about mobile sensing has been already presented in Chapters 2 and 6. In this subsection we give some technical background and overview mobile crowdsourcing models and tools for collaborative sensing.

One way to collect sensing data across large cities in a scalable way is to exploit the full potential of crowdsensing: while crowdsourcing aims to leverage collective intelligence to solve complex problems by splitting them in smaller tasks executed by the crowd, crowdsensing splits the responsibility of harvesting information (typically urban monitoring) to the crowd. Traditional crowdsourcing platforms, such as Amazon Mechanical Turk (AMT), act as mediators between task clients, usually called workers, and providers: the introduction of such platforms has driven specific research efforts that aim at optimizing crowdsourcing resources. For example, Bernstein *et al.* propose a statistical model and an algorithm to pre-recruit, pool, and pay workers to minimize task execution time on AMT [223]. Another important efficiency goal is to minimize the number of workers to get a sufficiently reliable result; for instance, CrowdSense aims at sampling subsets of workers and weighting their contributions to efficiently obtain an output that approximates the opinion of the whole crowd [224].

Although very useful and effective in fixed Internet settings, these crowdsourcing models, tools, and platforms are typically not suitable for mobile crowds because they are unable to exploit the sensors available onboard of mobile devices and do not include context-aware mechanisms which are necessary for effective mobile sensing (e.g., task replication to mitigate client mobility or impossibility to connect). Therefore, mobile crowdsensing is recently emerging as a new research area to propose original solutions that enhance and extend traditional crowdsourcing platforms with the goal of facilitating the management, delivery, and execution of potentially global massive tasks of sensing information to mobile clients. In the following, we briefly report a selection of these projects, from the first seminal works in crowdsensing to the research activities that are closer to our proposal.

Personal Environmental Impact Report (PEIR) is a smart application that exploits mobile phones to evaluate if users have been exposed to airborne pollution, enables data sharing to encourage community participation, and estimates the impact of individual user/-community behaviors on the surrounding urban environment [38]. mCrowd is an iPhone app that enables users to post and work on sensor-related tasks, such as requesting photos of a specific location, asking to tag photos, and monitoring traffic [39]. mCrowd can accommodate other mCrowd users, as well as ChaCha and AMT subscribers, as workers to complete the posted tasks. Medusa is the project closest to our effort: it uses a high-level domain-specific programming language, called MedScript, to define sensing tasks and workflows [222]. Medusa tasks can be deployed on both mobile phones, for instance to take pictures or videos, and the AMT platform; it also supports incentives to encourage user participation. By using MedScript it is possible to organize incentives, sensing tasks, and processing tasks in high-level workflows while the underlying Medusa framework hides the resulting complexities and takes care of task coordination, workers management, incentives assignment, and result collection. PEIR, mCrowd, and Medusa are important milestones of crowdsensing, but still do not include any specific model to ease the decisions of mobile sensing task assignment to people and to quantify urban sensing participActions. Our platform aims to fill that gap, as detailed in the following.

7.2.2 McSense: a Geo-social Mobile Crowdsensing Platform

In its simplest formulation, crowdsensing is a two-step process: assign sensing tasks to users and wait for them to complete the assignments. A more refined approach, taken by McSense is to exploit information about potential workers and their mobile execution context (processing power of their phones, battery level, people geographical location, etc.) to better and more effectively tailor the task assignment process. In particular, McSense puts task description, task assignment, and mobile sensing in a closed loop that allows a more efficient and effective usage of all involved socio-technical resources.

Figure 7.8 shows the McSense sensing lifecycle that consists of three main modules: mobile sensing, user/region profiling, and task as-

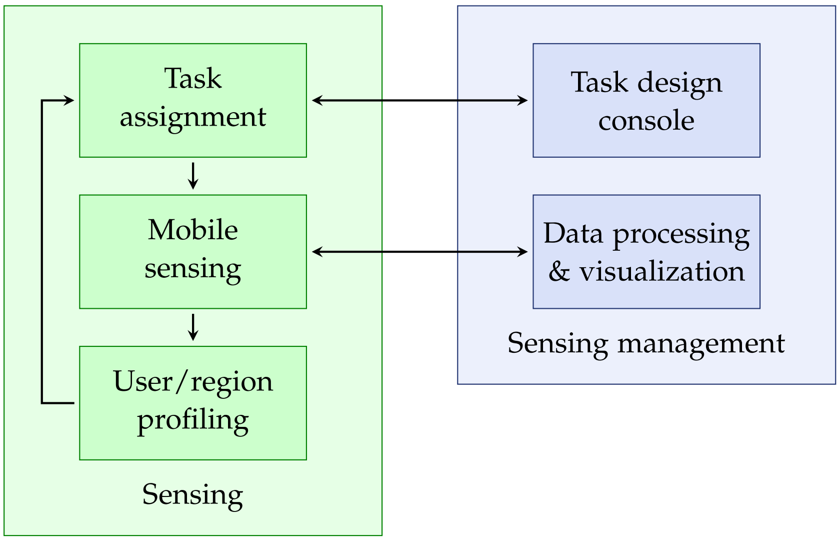


Figure 7.8: The feedback loop of McSense: users sense surrounding environment; McSense profiles their performance and helps designing/assigning new tasks which feed new sensing activities.

signment. The Sensing Management system interacts with the Sensing modules to provide a full-fledged framework for data sensing including functions to create and describe new sensing tasks and to analyze sensed data. *Mobile sensing* is a comprehensive term used to describe all types of sensing activities carried out by users via their mobile devices, such as recording noise pollution and taking pictures. Data processing analyzes the output of mobile sensing to reap the relevant data for the active sensing task. *User profiling* processes the collected data and aims to profile user participAction by drawing accurate estimations about their potential involvement in future mobile sensing tasks; moreover, to avoid long processing operations and to improve user experience, McSense includes *region profiling* to cache already evaluated user profiles, stored by geo-localized regions, thus exploiting also the physical locality principle (higher probabilities of similar profiles in the same region). *Task assignment* automatically assigns tasks to users as function of their characteristics derived by the profiling module. The *task design console* is a web-based user interface that enables the creation and publication of new crowdsensing tasks and includes all tools and support components to ease and assist the design of these tasks. *Data processing & visualization* provides active mapping features and exploits over-imposed informative visualization layers to show statistics about completed sensing tasks and collected data.

Before going on providing more details about these core components, let us briefly introduce and define more formally three core McSense entities: *worker*, *task*, and *task app*. *Worker* is the user running the McSense app and specific sensing tasks on her smartphone; in particular, in McSense, we assume that worker participAction is incentivized with a monetary compensation notified to users with the request to participate to a crowdsensing action. However, we could think of that monetary compensation as virtual currency, which could be translated into other types of incentives in the future (e.g., increased social recognition/visibility and task-related awards). *Tasks*, instead, are geo-dependent units of sensing work and are either long-running, such as recording location via GPS or noise-level via the microphone for six hours, or one-shot, such as taking a picture of a specific location. For task localization, we adopt a simplified model that associates each task with three properties, namely location, area, and duration: location is the set of physical GPS coordinates; area is a circle-like region determined by a radius, and a task is successfully executed if the worker executes it within the area; duration is the time a task remains valid and waiting for possible completion before its deadline. Finally, to ease the sensing action for workers, we provide each sensing task with a ready-to-use and intuitive mobile app, called *task app*, which includes everything needed to complete the required task. For instance, for a geo-localized photo, the McSense task app interacts directly with GPS and WiFi localization functions and only requires the user to snapshot the picture by pressing a single button.

7.2.3 The McSense Distributed Architecture

The McSense architecture includes three main distributed components that map the three phases of data collection and management into the mobile app, the data backend, and the task control console, as shown in Figure 7.9. The *McSense mobile app* acts as an active stub deployed on the worker smartphone: it receives task offers, allows users to accept them, and provides the tools to complete them, possibly with a very simple interaction and user GUI, by accepting the corresponding task app that seamlessly configures all needed sensors available onboard. The task apps report their data to the McSense mobile app. In addition, the McSense mobile app collects all data useful to profile users, devices, and, eventually, regions. When tasks are completed, it also uploads both sensed data and locally profiling results to the data backend. Functionally, the McSense mobile app implements the Sensing, Data Processing, and Upper-tier Interface components of the Mobile Infrastructure Tier described in Chapter 3.

The *McSense data backend* is the infrastructure component that receives data from the McSense mobile app via a public interface, reliably stores and analyzes sensed data to evaluate task-related statistics, such as the number of workers receiving it, the number of tasks successfully completed, and the task completion time. In the proposed reference logical model it is part of the Fixed-Infrastructure Tier, because it stores and processes sensed data. The data backend exploits global visibility to proceed with user profiling by using both tech-

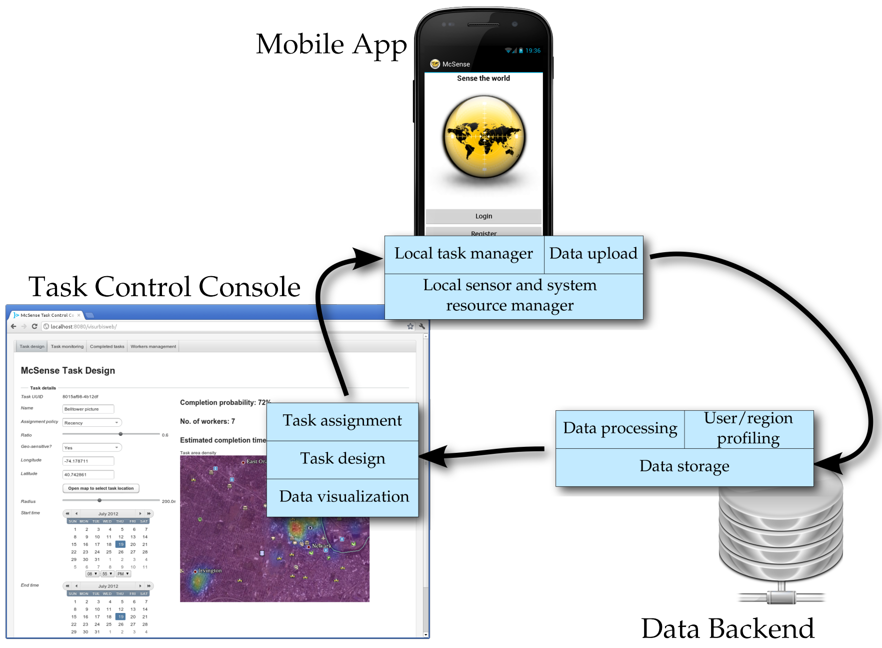


Figure 7.9: The McSense distributed architecture.

nical and social dimensions. Technical dimensions are system-level resources, such as number and type of sensors available on the smartphone, type of available network interfaces, and available battery. Social ones, instead, focus more on geo-social behavior of workers in terms of location, such as visited areas and current position, and of social relationships, such as co-location and social ties. Worker and region profiles, incrementally refined as McSense collects more and more data about workers crossing an area, are very useful to tailor future task assignment and making it more effective.

The last component of the distributed architecture of McSense is the *task control console*, used by sensing managers, namely the human operators who create and assign tasks. Like the previous component, it is part of the Fixed-Infrastructure Tier, implementing Data Processing and Mobile-Node Management functionalities. The console, apart from data visualization, offers two main functions: task design and task assignment. The task design component gives immediate feedback to the operators by estimating expected performance for a geo-localized sensing task. In particular, by using statistics/profiles stored in the data backend and task-related data (location, area, and duration), the task design component evaluates the time required and the number of workers needed to complete the task with a desired probability. The task assignment component goal, instead, is to define the optimal set of workers to carry out the sensing task with the targeted objectives and success level. For example, assigning a task to a user typically spending much time in an area increases its probability of

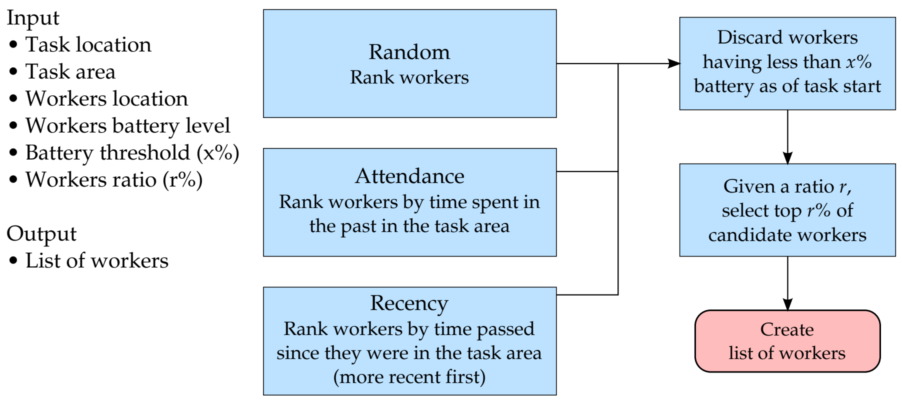

Input
• Task location
• Task area
• Workers location
• Workers battery level
• Battery threshold (x%)
• Workers ratio (r%)
Output
• List of workers
Random
Rank workers
Attendance
Rank workers by time spent in the past in the task area
Recency
Rank workers by time passed since they were in the task area (more recent first)
Discard workers having less than x% battery as of task start
Given a ratio r, select top r% of candidate workers
Create list of workers

Figure 7.10: The McSense task assignment.

success, whereas assigning it to a user that has been there recently tends to decrease the time needed for task completion.

In McSense, we prototyped several different task assignment algorithms and thoroughly evaluated their performance based on collected user profile data and real-world user participAction. The Sensing Manager subsystem uses these valuable user/region profiles to assist sensing managers in the definition and selection of the assignment algorithm that is expected to maximize the performance metrics they care most, with minimum consumption of associated socio/technical resources.

7.2.4 Task Assignment Policies

McSense deals with a continuously changing landscape: tasks are dynamically described and assigned, while workers roam free and by following paths that unpredictably change dynamically, making it difficult to predict the best task-worker assignment schema. McSense evaluates sensing task performance in terms of three main goal parameters: success ratio (i.e., ratio of successfully completed tasks over created ones), completion time (i.e., the interval between task start time and its successful conclusion), and number of required workers (i.e., number of workers to activate for task execution). At sensing task creation time, the sensing manager can either rely on default goal parameter values proposed by McSense or configure them depending on the experience stemming from previous task execution runs and specific application requirements. For instance, for a mission-critical sensing task, she will choose higher completion probability with higher number of workers, whereas if the budget is low the she may be primarily interested in minimizing the number of workers, so to reduce the associated incentive costs. Moreover, let us stress that these parameters also represent variables that are mutually dependent and deeply related to the assignment algorithm used to select workers.

McSense provides three different policies that, taking as inputs task properties (e.g., location, area, and duration) and user/region profiles, assign tasks to workers (see Figure 7.10): random policy, attendance policy, and recency policy. The *random policy* does not exploit any context awareness and selects the set of workers to employ as a random group of available people in the whole city. The *attendance policy* exploits knowledge about the time previously spent by people in the task area; based on that indicator, it chooses and ranks potentially good workers. The *recency policy*, instead, favors and selects as workers the people who have more recently (with respect to the creation time of sensing task) traversed the sensing task area. For each policy, McSense calculates ranked lists of candidate workers; let us note that profiling information about people attendance and recency permits to keep their respective lists much smaller than the one for the random policy, which will include any user in the whole city area. In addition, all the implemented policies do not consider workers whose battery level is below a certain threshold, called *battery threshold*, at task starting time because we assume that these workers will unlikely run the sensing task to avoid battery exhaustion. Finally, the last input parameter, called workers ratio, is used to decide the percentage of candidate workers that will receive the task assignment, expressed as a percentage value in the $[0.0, 1.0]$ range; in fact, since a task can be completed by any one worker, the sensing manager can set this parameter to replicate the task to multiple workers so to increase the completion probability before the deadline.

Given the above inputs and assignment policies our McSense task assignment component evaluates sensing task performance (success ratio, completion time, and number of required workers). Moreover, because completed tasks are typically much fewer than all possible crowdsensing tasks of potential interest (e.g., some of them could relate to scarcely traversed areas), the task assignment component also runs prediction algorithms to decide how to effectively self-manage its behavior based on the sensing&profiling data harvested so far. The primary goal is to exploit the already collected sensing&profiling information to calculate reasonably accurate performance forecasts and estimations about future potential tasks in a relatively lightweight way. The prediction process consists of two phases and exploits the collected real-world dataset, by dividing it temporally into two parts. The first phase considers the first part of the dataset as the past history and builds user/region profiles, such as ranked candidate worker lists. The second phase, instead, as better detailed in the following, creates synthetic ("virtual") future tasks and assigns them to candidate workers selected according to the prediction algorithms; then it evaluates the performance of the predicted situations. Task designers take advantage of those forecasts, evaluated offline by our prediction process, to get fast feedbacks online about the expected performance of the sensing tasks they are defining.

With a closer view to technical details, McSense task prediction automatically and stochastically generates virtual tasks with different locations, areas, and durations. It emulates their execution based on the user profile stored in the data backend, in particular based on location traces: a virtual task is considered to be successfully completed if the location trace of a user comes in its range. To make the model more realistic, we also assume that workers whose device battery level is very low (e.g., less than 20%) will never execute any task, while if the battery level is high (e.g., 80% or more), they will always execute any task they can; the probability of executing a task increases linearly between 20% and 80%. In particular, given a task, its duration, and the set of workers it has been assigned to, the emulator looks for a worker within the task area by iterating over worker position records in the task duration period. When it finds one, it stochastically evaluates whether the worker will be able to complete the task based on its battery level, and then updates the statistics about the policy under prediction by moving to the next worker location record. In the future, additional user profile parameters can be added, such as task completion rate or quality of data provided. We run those predicted situations for each assignment policy implemented in McSense; thus, sensing managers can exploit these additional data to compare possible assignment policies and to choose the one which better suits their needs, being it having a high chance of successful completion, minimizing the success time, or minimizing the number of workers involved.

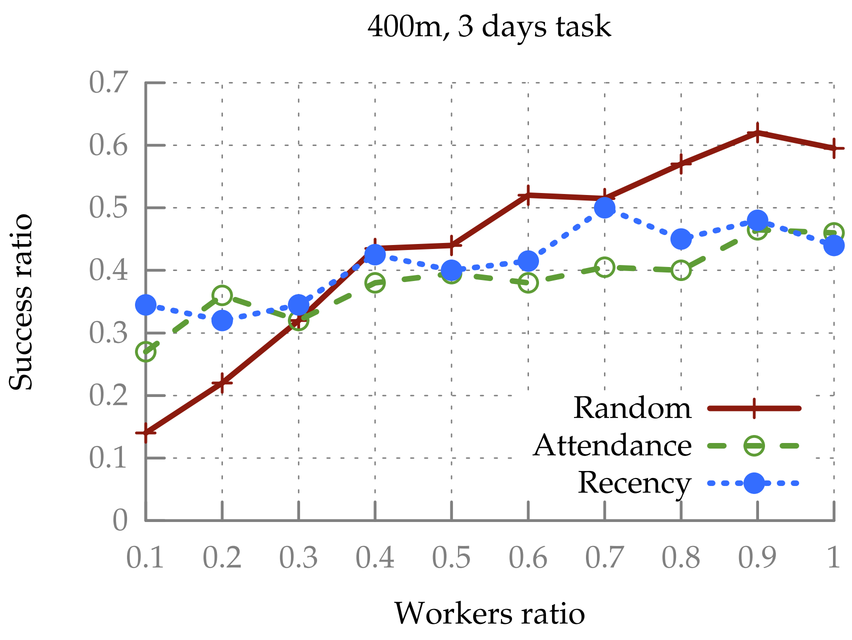
Implementation Insights and Experimental Results

We developed a prototype of the McSense platform to assess its architecture and the validity of its assumptions. The McSense mobile app has been implemented as an Android app that runs on the smartphones of available workers. The McSense data backend interface that receives data from the mobile app is a Java servlet, running on Apache Tomcat 7.0; it uses a PostgreSQL 9.1 database for persistent data storage. The database has been optimized with PostGIS 1.5, a package to support geographic object storage and indexing: this upgrade makes geographical queries much faster if compared with a naïve schema that simply stores coordinates as longitude-latitude tuples. Finally, the Task control console is an Ajax Web application based on the Google Web Toolkit at the client side and Apache Spring at the server side.

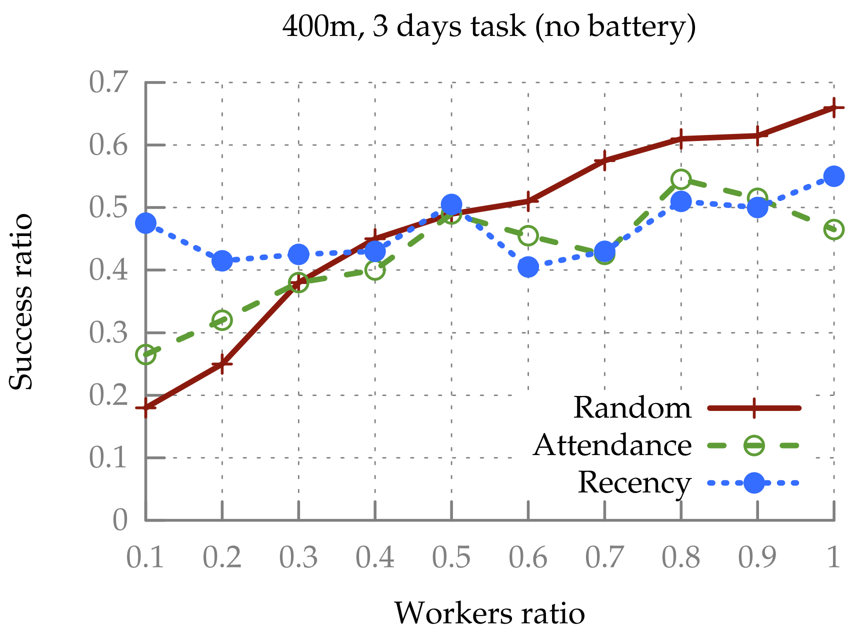
As part of our experiment, the McSense mobile app has been installed by 44 people who often visit the NJIT campus in Newark, New Jersey, and who volunteered as potential workers. The experiment has run for two months in the middle of the spring semester, from February 2012 to April 2012. During that period, McSense has executed many tasks that collected the following data: location, accelerometer readings, MAC addresses of nearby Bluetooth devices, Basic Service Set Identification (BSSID), capabilities and signal power of WiFi hotspots, application usage and associated bandwidth usage, battery level, and photos. During the two-month study, we posted daily sensing tasks requests to all users; workers had competes to get them, without enforcing any specific assignment policy, such as in the random policy.

The collected data from real scenarios of urban sensing have been used to quantitatively evaluate the emulated performance of our McSense assignment policies. In particular, we assigned tasks randomly placed on a $4km \times 4km$ area centered on NJIT campus. Tasks had an area with radius in the $[100m, 500m]$ range and duration between 1 and 7 days; according to our prediction process, we used the first part of the dataset (data sensed in the first month) to evaluate user/region profiles, while virtual tasks were temporally placed in the second month. Fixing an area with radius $400m$ and task duration equal to 3 days, Figure 7.11a shows how selected candidate workers ratio influences the success ratio of executed tasks: when the workers ratio is 1, the random policy has the highest success ratio, but employs all available workers (see Figure 7.12). Let us preliminary anticipate that in all our experiments we consider a worst case scenario in which tasks are randomly generated in the whole target $4km \times 4km$ area; hence, because some tasks may be placed in locations very rarely visited by workers, none of the considered policies can achieve a 100% completion rate. If we consider highly-visited locations only, completion rate would definitely be higher, but the associated results would represent unrealistically optimistic situations, while we want to derive forecasts valid for the whole area, including less visited ones. Of course, the same prediction process may be modified and applied to smaller and more densely populated areas only. Attendance and recency policies (with slightly better performance results by recency) have exhibited better results when the workers ratio is low because they carefully target task assignment only to users who have higher probability to accept executing the assignments. In addition, we collected analogous experimental results by ignoring battery level: as shown in Figure 7.11b, those predictions overestimate the success ratio because they are based on the assumption that tasks will be executed by workers located within the task area independently of their battery levels, which typically does not hold in real-world scenarios. That result confirms the importance to use technical profiles to forecast workers and device behavior so to improve task assignment performance.

Relating the task success ratio reported in Figure 7.11a to the number of workers (Figure 7.12), we remark that the recency and attendance policies allow for very good performance even using fairly low



(a) Battery monitored.



(b) Battery not monitored

Figure 7.11: Knowledge of battery level impact on task success ratio as function of workers ratio.

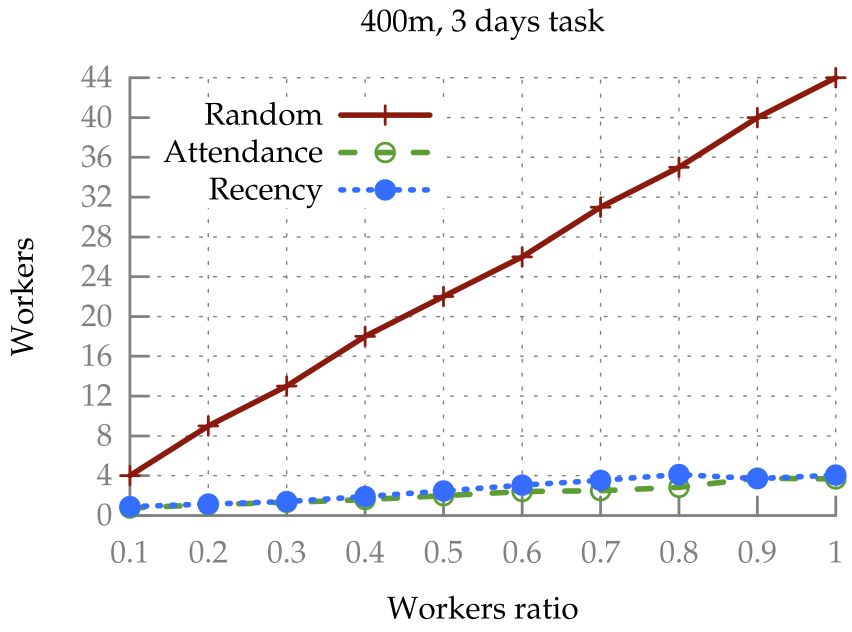


Figure 7.12: Number of workers that receive a task assignment as function of the worker ratio parameter.

numbers of workers. Moreover, they allow to carefully control the number of workers to involve in the participAction, even for high worker ratio values: indeed, these two policies always use less than 4 workers because they are able to filter and keep in only good candidates; in other words, their ranked lists of workers are much shorter than the one for the random policy.

Figure 7.13 shows the impact of task radius on success time: as the radius increases, the success time decreases, because there is a larger number of workers who are likely to execute the task; however, the radius is more relevant for the random policy, mainly because the attendance and recency policies have already good performance also for small radius values.

We also collected statistics about task that were actually assigned to users. *First, by observing the user behavior regarding task completion vs. monetary incentive, McSense could perform a more intelligent task assignment that finds the right balance between budget and data quality*. In our study, users prefer long-term automatic tasks (e.g., collect GPS and accelerometer readings for a day) to short-term manual tasks (e.g., take a photo at a given location), but once tasks were accepted, the completion rate was higher for manual tasks: that was caused in large part by certain power-hungry sensing applications that induced users to abort the tasks when the battery was below a threshold. However many were willing to recharge their phone during the day due to the monetary incentive. Similarly, many users seem willing to trade-off privacy (e.g., location privacy) for money. *Second, fault-tolerance mech-*

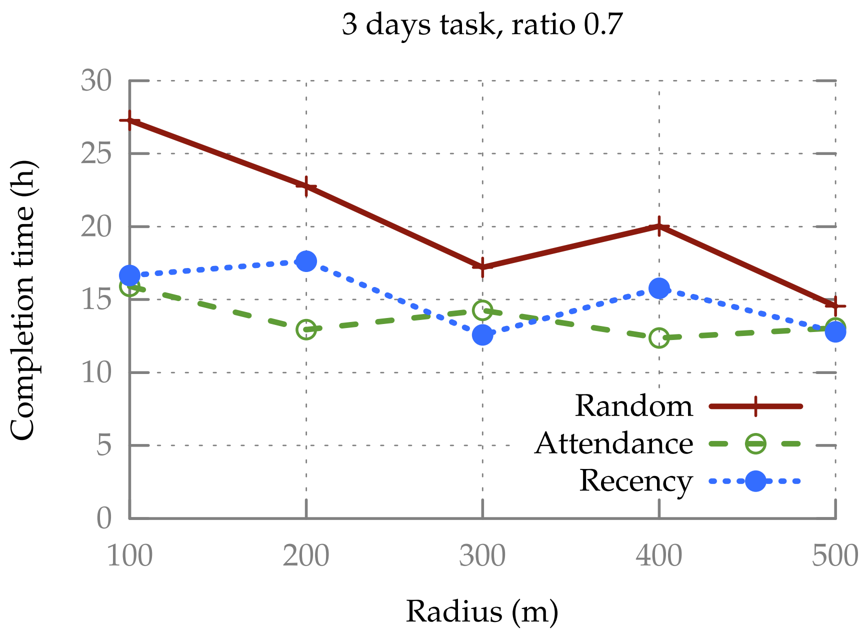


Figure 7.13: Task completion time as function of the task radius.

anisms are needed to cope with poor quality data or not receiving data at all from users who accepted a task. Assigning more tasks than necessary (i.e., using high workers ratio) could be a potential solution, although budget constraints could create impediments. In terms of the quality of the data collected in our study, we learned two interesting facts: (i) as expected, a non-negligible percentage of users attempted to "fool" the system by providing fake data (i.e., for photo tasks); (ii) users are tempted to accept high-priced short-term tasks with short duration, but quite often they were not able to complete them. *Third, user and region profiling can help with task decomposition decisions*. For example, the decision how to split long term tasks into several shorter tasks given to many users or keep the longer tasks and assign them to a few users is important. Based on our results, the completion rate is higher for longer tasks, but mobility traces and other parameters should also be considered when making decomposition decisions. *Forth, the places most frequented by users are a good indicator of task acceptance*. In fact, our results show that users are less willing to take tasks that require to change their daily routine (e.g., to take a picture on a road that they do not usually go through for their normal commute) and prefer those located at highly frequented points of interest in the campus.

7.3 CHAPTER CONCLUSIONS

In this chapter we conclude the scaling up of Pervasive Sensing systems bringing them from *data-centric* collection applications to full fledged systems that exploit sensed data and social communities to provide novel, useful services. We have shown that the role of social communities is two-fold: they can be passively involved for data collection and they can actively and willingly participate to provide data that is impossible to harvest opportunistically.

For the passive data collection aspect we have presented SAIR, a system that exhibits the capacity of providing a great user experience, keep the user engaged over time and build personalized services. This work paves the way to a new generation of socio-technical IMS-based mobile service platforms able not only to keep users in the service creation and provisioning loop through both ease service composition tools and automatic recommendation.

We investigated the possibilities of actively involving users in data collection as part of the McSense project, a full-featured geo-social crowdsensing platform. Its distributed architecture and data analysis capabilities make it a flexible and reliable framework for leveraging sensing crowds in city-wide deployment environments with expected high density of workers. The reported experimental results show that McSense assignment policies allow easily tuning the preferred performance tradeoff depending on specific task properties.

8 CONCLUSIONS

In the previous chapters we presented our work on Pervasive Sensing systems in Future Networks. Our case studies showed the applicability of our logical model that opportunistically exploits sensing and networking resources on *data-centric*, *person-centric*, and *social-centric* systems in five different significant scenarios, that have been throughly evaluated on simulators and on real-world testbeds. In this chapter we overview our technical contributions and the future research directions highlighted by this thesis. In Section 8.1 we summarize our main findings, then in Section 8.2 we describe the most interesting and promising future research directions that the work presented in this thesis has uncovered. Section 8.3 concludes this chapter and the thesis work.

8.1 MAJOR CONTRIBUTIONS

The analysis of previous researches presented in Chapter 2 shows that the bulk of sensing tasks (i.e.: actually measuring physical and social data) is a very well investigated field that can be reliably exploited. What is still lacking is a deep integration of sensing activities in Future Networks that are expected to comprise low-power embedded devices and mobile personal devices, backed up by powerful servers, both physical and virtualized in the Cloud (Chapter 2).

The challenges of integrating sensing systems in Future Networks are twofold: technical and social. On the one hand, there are many technical issues to address to realize Pervasive Sensing systems mainly caused by the heterogeneous scenario of Future Networks, on the other hand, there is the social challenge to provide useful services to engage users and encourage them to use and participate in sensing systems. In the following we present our main technical and social findings. We start with *data-centric* systems that deal with raw data collection, then we focus on *person-centric* systems that process raw data to provide services to individuals, and finally we describe *social-centric* systems that exploit sensing in human communities.

In Chapter 3 we presented a generic, robust module for the development of Pervasive Sensing systems, that can be used as reference for the development and deployment of *data-*, *person-*, and *social-centric* systems. Our model implements a tiered architecture that supports sensor nodes, mobile nodes, and fixed infrastructure that makes it easy to design scalable systems, and at the same time adopts cross-tier visibility that allows devices to be aware of the resources and limitations of other devices. In other words we claim that the key to realize Pervasive Sensing systems is not to conceal device intrinsic features and to let devices be aware of the reciprocal heterogeneity and status: this approach sacrifices the simplicity of assuming that all devices have similar resources to gain the capability of tailoring data collection and processing to the actual available resources, and dynamically adapting these tasks as they cross the boundary from a tier to another. Differently from existing works [8, 98, 124, 127, 131–139], our proposal does not mandate all the tiers to be present and all the tier components to be implemented; rather, it is a highly modular framework that can be easily adapted to very different sensing scenarios, ranging from traditional WSN-based data collection systems to cloud-backed large-scale sensing deployments.

We tested the logical model by using it as framework for two data-centric systems in Chapters 4 and 5: the first one exploits mobile devices that roam in a dense WSN to opportunistically organize routing bridges that improves data collection performances by reducing routing latencies for urgent data, while the second one leverages mobile devices as intermediaries that receive configuration data from the Cloud and upload it on sparse sensor nodes, enabling large battery savings. The analysis of these systems prove two major claims of this dissertation: firstly, that the engineering cost of distributing Pervasive Sensing systems over heterogeneous devices is compensated by the performance boost caused by properly shifting workloads on more powerful devices; secondly, that the detrimental dynamic effects of Future Networks, caused by node mobility and high packet loss probability, can be mitigated with opportunistic networking.

Moving on to person-centric systems, Chapter 6 shows that mobile devices, in addition to being a support to WSN, can directly sense surrounding environment and provide high level inferences about user activities. In particular, we designed a high-performance reusable sensing component for smartphones called MoST that provides high-level inferences about user, that can be easily exploited by programmers to quickly develop novel applications based on user activities, for example by simply tracking them or by adapting application behavior based on them. The work presented in this chapter proves that the guideline of an opportunistic design, already used for dynamic network integration, is useful also for sensing itself; in fact, even if smartphones are not primarily designed to be sensors, our work has shown that robust machine learning techniques can filter out noisy data and produce meaningful measurements. In addition, we have identified several techniques and software designs, such as object pooling to reduce the overhead of Java garbage collection and multiplexing of sensed data to minimize data processing latencies, that maximize the efficiency of mobile sensing software with minimal impact on the quality of user experience when using smartphones, significantly improving sensing performance compared to current state-of-the art mobile sensing software.

We have built upon the techniques and findings of the previous chapters to design two Pervasive Sensing systems, called SAIR and McSense, that provide services to social communities (Chapter 7). SAIR uses MoST to build a recommendation systems that suggests apps to users based on their activity, location and interests. We have shown that RFD scoring is a simple metric that allows to accurately estimate over time user interest in a given application and that it can be exploited to rank by relevance applications to real world contexts, such as location and physical activity. In addition, the experimental assessment of SAIR has shown that the industry-standard IMS platform can provide a scalable support for the development of collaborative social-centric sensing applications. McSense explores the possibility of exploiting users as providers of sensed data (i.e., *crowdsourcing*) incentivizing them with monetary reward, focusing on predictive techniques to drive sensing task to optimize metrics such as task success probability and cost. The real-world data collected during the McSense experiment has highlighted the need for design social-centric sensing systems to be socio-technical aware to maximize opportunistic sensing performances. Both SAIR and McSense are systems that keep humans in the loop of sensing, allowing to provide innovative, useful services and even to collect data that would be very hard to get in an automatized fashion, thus enhancing Pervasive Sensing systems and allowing them to thrive. Our logical model has shown to be a robust framework to develop these systems despite their radical differences compared to more traditional sensing systems such as WSNs.

To summarize, in this dissertation we provide design guidelines and a robust, reusable model for Pervasive Sensing systems. Experimental results, obtained via simulation and real world deployments show that our design guidelines and logical model, based on *opportunistic approaches*, *cross-layer visibility*, *localized interactions*, and *tiered architectures*, are a successful framework for the design and development of Pervasive Sensing systems. Opportunistic networking and sensing have been proved to be useful tools to exploit resources in the scenario of Future Networks, whose dynamic behavior makes resources availability very volatile, enabling the development of sensing systems in an otherwise very hostile scenario. However, opportunistic networking and sensing can not be simply implemented as blindly and aggressively starting communications and sensing whenever resources are available, due to the heterogeneity of involved devices, some of which could soon run out of energy: cross-layer visibility promotes explicitly sharing node status, and in our case studies has proven to be a key technique to tune networking and sensing tasks according to the resources actually available on very different devices, that are expected to co-exist in Future Networks. Finally, Pervasive Sensing systems naturally aim to large-scale deployments, either to monitor large areas or to provide services to large human communities; we proved that such large-scale deployments are made possible by the tiered architecture of our logical model and the localized interactions recommended by our guidelines, that enable scaling by respectively promoting hierarchical organization of devices and avoiding costly long-range communications.

To conclude, in this thesis we strove to strike a balance between Pervasive Sensing system requirements and their applicability to real-world scenarios in Future Networks, with their limits and features. With the extensive experimental evaluation of our work we have proven the validity of of theoretical background, hoping that it will be useful to the research community to further develop Pervasive Sensing systems and foster their widespread adoption.

8.2 FUTURE RESEARCH DIRECTIONS

In this dissertation we pursued several important research directions, however there are still aspects related to the presented work that deserve further investigation. In the following we present the principal research directions that we intend to pursue.

Data-centric systems - While our work showed the valuable contribution of mobile nodes to data collection on sensor nodes, we feel that there is still room to optimize data collection, by reducing power consumption and data collection latencies. In particular, we plan to further exploit dynamic MANET-WSN integration even for non-urgent data, by supporting delay-tolerant routing. In particular, we aim at developing novel algorithms with power consumption guarantees similar to those described in Section 4.4 that dynamically decide whether to route data or to store it and forward it to a more powerful mobile node as soon as it is possible. About the opportunistic exploitation of smartphones to re-configure sensor nodes, described in Chapter 5, we have so far explored the potentialities of IMS for coordination and the possible power savings on sensor nodes; we plan to refine the reconfiguration task assignment, currently based on simple geographical considerations, by profiling users and selectively assigning to them the reconfiguration of sensor nodes that they are more likely to roam by, possibly using techniques similar to those presented in Chapter 7. Let us stress once again that prolonging battery lifetime, both on sensor nodes and mobile nodes, is a very important step to make Pervasive Sensing systems viable in real-world scenarios, hence additional research in this direction is needed.

Person-centric systems - We claim that the MoST component described in Chapter 6 for mobile sensing is an important step to promote the development of novel applications based on smartphone sensing, allowing programmers not familiar with signal processing and machine learning techniques to directly get high-level inferences. We have two main goals for the development of MoST: the first one is transforming it in a full-fledged system for data collection, the second one is to make it usable even to non-technical savvy researchers. We will achieve the first goal in three steps: first, we will integrate more inputs and high-level inference algorithms (e.g.: user stress level from voice processing and environment context via geo-localization). The second step is implementing several power management policies to reduce the impact on smartphone battery lifetime, for example by adapting sensing rate to the time of the day and to current user activity. In the final step we will complement MoST with a server-side support, capable of automatically harvesting high level inferences from smartphones. The second goal, i.e. making MoST available to non-technical researchers, is motivated by the feeling that MoST could be a great instrument to help psychologist and sociologist tracking human behavior in a non-intrusive fashion, collecting information that has always been unavailable until now. Hence, we plan to develop a simple applications that asks researchers what kind of data they want to track (e.g.: location and stress level via voice processing), configures accordingly MoST and automatically compiles it as an Android application ready for the deployment, that will automatically upload sensed data to a server. We feel that these extensions to MoST will make it an important building block for both the development of Pervasive Sensing systems and a powerful aiding tool for psychological studies and tracking physical and mental behavior. It is possible that some of signal processing and machine learning algorithm will have CPU or memory requirements unavailable in the foreseeable future on smartphone. Even though we do not currently plan to directly investigate this problem, we feel that the integration of MoST with cloudlets, i.e. resource-rich computer clusters available for use by nearby mobile devices [52], would be an interesting research topic showcasing the potentialities of both mobile sensing and dynamically-instantiated cloud services.

Social-centric systems - Our research, reported in Chapter 7, and existing and ongoing works on crowdsensing have just started scratching the surface of the potentialities of including people in the sensing loop. A very interesting topic is how to best incentive user participation to sensing. We believe that the capabilities of MoST for sensing and the already presented work put us in a good position to further study what influences willingness of participating in sensing, how much users are willing to go out of their way to provide requested data, how rewards can be tuned for data can be passively collected vs. data that needs active user participation, and what kind of rewards (e.g.: monetary, virtual badges) most improve or reduce sensing task success. On a larger scale, we believe that social-centric systems based on MoST, SAIR, and McSense can greatly contribute to research on smart cities. In fact, the high-level inferences collected by MoST, together with the collaborative filtering techniques of SAIR and crowdsensing capabilities of McSense, deployed in a city-wide scenario allow to run social evaluations, (e.g., analyze people commute habits and derive their carbon footprint, identify places that promote social interactions and physical activities) and develop several novel services (e.g., suggest city sightseeing tours based on automatically built user profiles, realize collaborative journalism services, provide navigation services that relieve traffic by exploiting global knowledge of people location and direction).

8.3 FINAL REMARKS

The recent technological advances are making the vision by Mark D. Weiser of a "calm technology" that disappears while still providing its services a reality [1]. This profound change in technology and society is being pushed by several concurrent trends: the pervasiveness of sensing devices, the availability of powerful personal mobile devices, the possibility of dynamically integrating heterogeneous network, the consistent accessibility of computing and data storage resources on the Cloud in wired and wireless scenarios.

We believe that Pervasive Sensing systems will play a pivotal role in this change, because they are the technological enabler that truly make computer systems aware of real world environments, allows to interact with it and let them perceive the activities and intentions of users, thus making possible to seamlessly provide targeted services to users with minimal effort. Due to the findings reported before, supported by the publication record obtained from this dissertation, and to the promising future research directions, we are convinced that this thesis can foster future research activities and have an impact on the design of Pervasive Sensing systems in Future Networks.

ACRONYMS

AODV	Ad Hoc Distance Vector	21
AS	Application Server	
AMT	Amazon Mechanical Turk	138
API	Application Programming Interface	101
BAN	Body Area Network	15
BCP	Backpressure Collection Protocol	20
BSSID	Basic Service Set Identification	146
CAR	Context-aware Adapting Routing	26
CDCP	Centers for Disease Control and Prevention	97
CPU	Central Processing Unit	31
CSMA	Carrier Sense Multiple Access	67
CTP	Collection Tree Protocol	20
DAG	Directed Acyclic Graph	21
DBMS	Data Base Management System	83
DDMS	Dalvik Debug Monitor Server	116
DFT-MSN	Delay/Fault-Tolerant Mobile Sensor Network	42
DTN	Delay Tolerant Network	25
DODAG	Destination Oriented Directed Acyclic Graph	21
DSR	Dynamic Source Routing	21
EDGE	Enhanced Data Rates for GSM Evolution	16
EEG	Electroencephalography	110
EOG	Electrooculography	110
ECG	Electrocardiography	110
FP	False Positive	117
GPRS	General Packet Radio Service	16
GPS	Global Positioning System	11
GUI	Graphical User Interfce	131
HSS	Home Subscriber Server	
HFUS	Highly Frequently Used Service	130
HUS	Highly Used Service	130
IaaS	Infrastructure as a Service	15
I-CSCF	Interrogating-Call Session Control Function	

IMS	IP Multimedia Subsystem	44
IoT	Internet of Things	15
LQI	Link Quality Indicator	75
LPL	Low-Power Listening	84
MAC	Media Access Control	36
MANET	Mobile Ad-hoc NETwork	15
MP2P	Mobile P2P	125
MET	Metabolic Equivalent of Task	96
MN	Mobile Node	83
MoST	Mobile Sensing Technology	99
NJIT	New Jersey Institute of Technology	46
OS	Operating System	34
PDA	Personal Digital Assistant	17
PaaS	Platform as a Service	15
PEIR	Personal Environmental Impact Report	139
P-CSCF	Proxy-Call Session Control Function	
PL	Path Length	53
PRR	Packet Reception Ratio	75
PS	Presence Service	79
QoS	Quality of Service	42
QoE	Quality of Experience	121
REM	Rapid Eye Movement	110
ROC	Receiver Operating Characteristic	
RF	Radio Frequency	84
RMS	Root Mean Square	113
RFD	Recency, Frequency, Duration	123
RFM	Recency, Frequency, Monetary	123
RPL	Routing Protocol for Low Power and Lossy Networks	21
SaaS	Software as a Service	15
S-CSCF	Serving-Call Session Control Function	
SAIR	Social-Aware IMS-enabled Recommender	45
SCIM	Service Capabilities Interaction Management	122
SDK	Software Development Kit	116
SIP	Session Initiation Protocol	38
SR	Service Ranker	127
SRD	Service Recommendation Dispatcher	127

SVM	Support Vector Machine	113
TP	True Positive	117
UMTS	Universal Mobile Telecommunications System	80
UCS	User Context Storage	127
URI	Uniform Resource Identifier	82
USB	Universal Serial Bus	35
WHOO	WSN Hybrid rOuting prOtocol	44
WSN	Wireless Sensor Network	10
WSAN	Wireless Sensor and Actor Network	19
XCAP	XML Configuration Access Protocol	83

LIST OF FIGURES

LIST OF TABLES

BIBLIOGRAPHY

[1] M. Weiser, "The Computer for the 21st Century," *Scientific American*, vol. 3, no. 3, pp. 94–104, 1991.

[2] K. V. Laerhoven, "The Pervasive Sensor," *Ubiquitous Computing Systems*, vol. 3598, pp. 1–9, 2005.

[3] J. M. Kahn, R. H. Katz, and K. S. J. Pister, "Next Century Challenges: Mobile Networking for "Smart Dust"," in *MobiCom '99: Proceedings of the Fifth ACM/IEEE Annual International Conference on Mobile Computing and Networking*, pp. 271–278, 1999.

[4] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless Sensor Networks: a Survey," *Computer Networks*, vol. 38, pp. 393–422, Mar. 2002.

[5] J. Lue, M. Chu, J. Liu, J. Reich, and F. Zhao, "State-centric Programming for Sensor-Actuator Network Systems," *IEEE Pervasive Computing*, vol. 2, pp. 50–62, Oct. 2003.

[6] G. J. Pottie and W. J. Kaiser, "Wireless Integrated Network Sensors," *Communications of the ACM*, vol. 43, no. 5, pp. 51–58, 2000.

[7] B. Liu, P. Brass, O. Dousse, P. Nain, and D. Towsley, "Mobility Improves Coverage of Sensor Networks," in *MobiHoc '05: Proceedings of the Sixth ACM International Symposium on Mobile Ad Hoc Networking & Computing*, pp. 300–308, 2005.

[8] R. C. Shah, S. Roy, S. Jain, and W. Brunette, "Data MULEs: Modeling and Analysis of a Three-tier Architecture for Sparse Sensor Networks," *Ad Hoc Networks*, vol. 1, pp. 215–233, Sept. 2003.

[9] H. W. Stone and G. Edmonds, "HAZBOT: a Hazardous Materials Emergency Response Mobile Robot," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 67–73, 1992.

[10] S. Consolvo, D. W. Mcdonald, T. Toscos, M. Y. Chen, J. Froehlich, B. Harrison, P. Klasnja, A. Lamarca, L. Legrand, R. Libby, I. Smith, and J. A. Landay, "Activity Sensing in the Wild: A Field Trial of UbiFit Garden," in *CHI '08 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 1797–1806, 2008.

[11] H. Lu, W. Pan, N. D. Lane, T. Choudhury, and A. T. Campbell, "SoundSense: Scalable Sound Sensing for People-Centric Applications on Mobile Phones," in *MobiSys '09: Proceedings of the Seventh ACM International Conference on Mobile Systems, Applications, and Services*, pp. 165–178, 2009.

[12] P. Pelegris, K. Banitsas, T. Orbach, and K. Marias, "A novel method to detect heart beat rate using a mobile phone.," *EMBC '10: Proceedings of the Thirty-second IEEE Conference of Engineering in Medicine and Biology Society*, pp. 5488–5491, Jan. 2010.

[13] E. C. Larson, M. Goel, G. Borriello, S. Heltshe, M. Rosenfeld, and S. N. Patel, "SpiroSmart: Using a Microphone to Measure Lung Function on a Mobile Phone," in *UbiComp '12: Proceedings of the Fourteenth ACM International Conference on Ubiquitous Computing*, pp. 280–289, 2012.

[14] R. K. Rana, C. T. Chou, S. S. Kanhere, N. Bulusu, and W. Hu, "Ear-Phone : An End-to-End Participatory Urban Noise Mapping System," in *IPSN '10: Proceedings of the Ninth ACM/IEEE International Conference on Information Processing in Sensor Networks*, pp. 105–116, 2010.

[15] P. Mohan, V. N. Padmanabhan, and R. Ramjee, "Nericell: Rich Monitoring of Road and Traffic Conditions using Mobile Smartphones," in *SenSys '08: Proceedings of the Sixth ACM Conference on Embedded Networked Sensor Systems*, pp. 323–336, 2008.

[16] Y. Chon, N. D. Lane, F. Li, H. Cha, and F. Zhao, "Automatically Characterizing Places with Opportunistic CrowdSensing Using Smartphones," *UbiComp '12: Proceedings of the Fourteenth ACM International Conference on Ubiquitous Computing*, pp. 481–490, 2012.

[17] A. T. Campbell, N. D. Lane, E. Miluzzo, R. A. Peterson, H. Lu, X. Zheng, M. Musolesi, Kristóf Fodor, S. B. Eisenman, and G.-S. Ahn, "The Rise of People-Centric Sensing," *IEEE Internet Computing*, vol. 12, pp. 12–21, July 2008.

[18] F. C. Pereira, A. Vaccari, F. Giardin, C. Chiu, and C. Ratti, "Crowdsensing in the Web: Analyzing the Citizen Experience in the Urban Space," in *From Social Butterfly to Engaged Citizen* (M. Foth, L. Forlano, C. Satchell, and M. Gibbs, eds.), pp. 353–374, Cambridge, MA, USA: MIT Press, 2011.

[19] R. K. Ganti, F. Ye, and H. Lei, "Mobile Crowdsensing: Current State and Future Challenges," *IEEE Communications Magazine*, vol. 49, no. 11, pp. 32–39, 2011.

[20] N. Kourtellis, J. Finnis, P. Anderson, J. Blackburn, C. Borcea, and A. Iamnitchi, "Prometheus: User-Controlled P2P Social Data Management for Socially-Aware Applications," in *Middleware '10: Proceedings of the ACM/IFIP/USENIX Eleventh International Conference on Middleware*, pp. 212–231, 2010.

[21] A. Iamnitchi, J. Blackburn, and N. Kourtellis, "The Social Hourglass," *IEEE Internet Computing*, vol. 16, no. 3, pp. 13–23, 2012.

[22] R. Montoliu and D. Gatica-Perez, "Discovering Human Places of Interest from Multimodal Mobile Phone Data," in *MUM '10: Proceedings of the Ninth International Conference on Mobile and Ubiquitous Multimedia*, pp. 1–10, 2010.

[23] T. M. T. Do and D. Gatica-Perez, "Contextual Grouping: Discovering Real-Life Interaction Types from Longitudinal Bluetooth Data," in *MDM '11: Proceedings of the Twelfth IEEE International Conference on Mobile Data Management*, pp. 256–265, June 2011.

[24] S. Mardenfeld, D. Boston, S. J. Pan, Q. Jones, A. Iamntichi, and C. Borcea, "GDC: Group Discovery Using Co-location Traces," in *SocialCom '10: Proceedings of the Second IEEE International Conference on Social Computing*, pp. 641–648, Aug. 2010.

[25] H. Lu, J. Yang, Z. Liu, N. D. Lane, T. Choudhury, and A. T. Campbell, "The Jigsaw continuous sensing engine for mobile phone applications," in *SenSys '10: Proceedings of the Eighth ACM Conference on Embedded Networked Sensor Systems*, pp. 71–84, 2010.

[26] A. Agneessens, I. Bisio, F. Lavagetto, and M. Marchese, "Design and Implementation of Smartphone Applications for Speaker Count and Gender Recognition," *The Internet of Things*, pp. 187–194, 2010.

[27] K. K. Rachuri, M. Musolesi, C. Mascolo, P. J. Rentfrow, C. Longworth, and A. Aucina, "EmotionSense: A Mobile Phones based Adaptive Platform for Experimental Social Psychology Research," in *UbiComp '10: Proceedings of the Twelfth ACM International Conference on Ubiquitous Computing*, pp. 281–290, 2010.

[28] H. Lu, M. M. Rabbi, G. T. Chittaranjan, D. Frauendorfer, M. Schmid Mast, A. T. Campbell, D. Gatica-Perez, and T. Choudhury, "StressSense: Detecting Stress in Unconstrained Acoustic Environments using Smartphones," in *UbiComp '12: Proceedings of the Fourteenth ACM International Conference on Ubiquitous Computing*, pp. 351–360, 2012.

[29] M. Raento, A. Oulasvirta, and N. Eagle, "Smartphones: An Emerging Tool for Social Scientists," *Sociological Methods & Research*, vol. 37, pp. 426–454, Feb. 2009.

[30] J.-K. Min, S.-H. Jang, and S.-B. Cho, "Mining and Visualizing Mobile Social Network," in *UIC '09: Proceedings of the Sixth International Conference on Ubiquitous Intelligence and Computing*, pp. 111–120, 2009.

[31] M. Raento, A. Oulasvirta, R. Petit, and H. Toivonen, "ContextPhone: A Prototyping Platform for Context-Aware Mobile Applications," *IEEE Pervasive Computing*, vol. 4, pp. 51–59, Apr. 2005.

[32] M. A. Russel, *Mining the Social Web*. Sebastopol, CA, USA: O'Reilly Media, 2011.

[33] J. Staiano, B. Lepri, N. Aharony, F. Pianesi, N. Sebe, and A. Pentland, "Friends don't Lie - Inferring Personality Traits from Social Network Structure," in *UbiComp '12: Proceedings of the Fourteenth ACM International Conference on Ubiquitous Computing*, pp. 321–330, 2012.

[34] A. Gaggioli, G. Pioggia, G. Tartarisco, G. Baldus, D. Corda, P. Cipresso, and G. Riva, "A Mobile Data Collection Platform for Mental Health Research," *Personal and Ubiquitous Computing*, Sept. 2011.

[35] M. De Domenico, A. Lima, and M. Musolesi, "Interdependence and Predictability of Human Mobility and Social Interactions," in *Proceedings of the Nokia Mobile Data Challenge Workshop*, 2012.

[36] S. Scellato, C. Mascolo, M. Musolesi, and J. Crowcroft, "Track Globally , Deliver Locally: Improving Content Delivery Networks by Tracking Geographic Social Cascades Categories and Subject Descriptors," in *WWW '11: Proceedings of the Twentieth World Wide Web Conference*, pp. 457–466, 2011.

[37] N. D. Lane, S. B. Eisenman, M. Musolesi, E. Miluzzo, and A. T. Campbell, "Urban Sensing Systems: Opportunistic or Participatory?," in *HotMobile '08: Proceedings of the Ninth Workshop on Mobile Computing Systems & Applications*, pp. 11–16, 2008.

[38] M. Mun, S. Reddy, K. Shilton, N. Yau, J. Burke, D. Estrin, M. Hansen, E. Howard, R. West, and P. Boda, "PEIR, the Personal Environmental Impact Report, as a platform for Participatory Sensing Systems Research," in *MobiSys '09: Proceedings of the Seventh ACM International Conference on Mobile Systems, Applications, and Services*, pp. 55–68, 2009.

[39] T. Yan, M. Marzilli, R. Holmes, D. Ganesan, and M. Corner, "mCrowd: A Platform for Mobile Crowdsourcing," in *SenSys '09: Proceedings of the Seventh ACM Conference on Embedded Networked Sensor Systems*, no. iii, pp. 347–348, 2009.

[40] S. Reddy, A. Parker, J. Hyman, J. Burke, D. Estrin, and M. Hansen, "Image Browsing, Processing, and Clustering for Participatory Sensing: Lessons From a DietSense Prototype," in *EmNets '07: Proceedings of the Fourth Workshop on Embedded Networked Sensors*, pp. 13–17, 2007.

[41] D. Cook, A. T. Campbell, and R. Want, "NSF Workshop on Pervasive Computing at Scale," tech. rep., 2012.

[42] ABI Research, “802.15.4 Wireless Sensor Network,” tech. rep., 2012.

[43] M. Hatler, D. Gurganious, and C. Chi, “Industrial Wireless Sensor Networks - A Market Dynamics Report,” tech. rep., ON World, Inc., 2012.

[44] H. LeHong and J. Fenn, “Hype Cycle for Emerging Technologies, 2012,” tech. rep., Gartner, Inc., 2012.

[45] H. LeHong, “Hype Cycle for the Internet of Things, 2012,” tech. rep., Gartner, Inc., 2012.

[46] A. Gupta, C. Milanesi, R. Cozza, and C. Lu, “Market Share Analysis: Mobile Phones, Worldwide, 3Q12,” tech. rep., Gartner, Inc., 2012.

[47] M. Hung, J. Erensen, and R. Sheng, “Market Trends: The Emergence of the $50 Smartphone,” tech. rep., Gartner, Inc., 2012.

[48] A. Ranalli, “Body Sensor Network and SPINE Framework,” in *COST2010/CONET/NEWCOM++ Training School and Workshop on Cooperating Objects and Wireless Sensor Networks*, 2010.

[49] ABI Research, “Short Range Wireless ICs: Bluetooth, NFC, UWB, 802.15.4, and Wi-Fi Market Forecasts,” tech. rep., 2010.

[50] P. Trevor, Z. Pei, C. Rohit, A. Yaw, and W. Roy, “The PSI Board: Realizing a Phonecentric Body Sensor Network,” in *BSN '07: Proceedings of the Fourth International Workshop on Wearable and Implantable Body Sensor Networks*, 2007.

[51] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, “A View of Cloud Computing,” *Communications of the ACM*, vol. 53, pp. 50–58, Apr. 2010.

[52] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, “The Case for VM-based Cloudlets in Mobile Computing,” *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, 2009.

[53] T. Halonen, J. Romero, and J. Melero, eds., *GSM, GPRS and EDGE Performance: Evolution Towards 3G/UMTS*. Wiley, 2003.

[54] G. Barrenetxea, F. Ingelrest, G. Schaefer, and M. Vetterli, “The Hitchhiker's Guide to Successful Wireless Sensor Network Deployments,” in *SenSys '08: Proceedings of the Sixth ACM Conference on Embedded Networked Sensor Systems*, p. 43, 2008.

[55] I. Chlamtac, M. Conti, and J. J.-N. Liu, “Mobile Ad Hoc Networking: Imperatives and Challenges,” *Ad Hoc Networks*, vol. 1, pp. 13–64, July 2003.

[56] A. Boukerche, B. Turgut, N. Aydin, M. Z. Ahmad, L. Bölöni, and D. Turgut, "Routing Protocols in Ad Hoc Networks: A Survey," *Computer Networks*, vol. 55, pp. 3032–3080, Sept. 2011.

[57] M. S. Corson, J. P. Macker, and G. H. Cirincione, "Internet-Based Mobile Ad Hoc Networking," *IEEE Internet Computing*, vol. 3, no. 4, pp. 63–70, 1999.

[58] P. Gupta and P. R. Kumar, "The Capacity of Wireless Networks," *IEEE Transactions on Information Theory*, vol. 46, no. 2, pp. 388–404, 2000.

[59] A. Shrestha and F. Tekiner, "On MANET Routing Protocols for Mobility and Scalability," in *PDCAT '09: Proceedings of the IEEE International Conference on Parallel and Distributed Computing, Applications and Technologies*, pp. 451–456, Dec. 2009.

[60] V. D. Park and M. S. Corson, "A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks," in *INFOCOM '97: International Conference on Computer Communications*, pp. 1405–1413, 1997.

[61] C. E. Perkins and E. M. Royer, "Ad-hoc On-Demand Distance Vector Routing," in *WMCSA '99: Proceedings of the Second IEEE Workshop on Mobile Computing Systems and Applications*, pp. 90–100, 1999.

[62] D. B. Johnson, D. A. Maltz, and J. Broch, "DSR: the Dynamic source Routing Protocol for Multi-hop Wireless Ad Hoc Networks," in *Ad Hoc Networking* (C. E. Perkins, ed.), pp. 139–172, Addison-Wesley Longman Publishing Co., Inc., 2001.

[63] C. Perkins and P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers," in *SIGCOMM '94: Proceedings of the Conference on Communications Architectures, Protocols and Applications*, pp. 234–244, 1994.

[64] T.-W. Chen and M. Gerla, "Global State Routing: a New Routing Scheme for Ad-hoc Wireless Networks," in *ICC '98: Proceedings of the IEEE International Conference on Communications*, pp. 171–175, 1998.

[65] T. Clausen, P. Jacquet, A. Laouiti, P. Muhlethaler, A. Qayyum, and L. Viennot, "Optimized Link State Routing Protocol for Ad Hoc Networks," in *INMIC '01: Proceedings of the Fourth IEEE International Multitopic Conference*, pp. 62–68, 2001.

[66] G. Pei, M. Gerla, and T.-W. Chen, "Fisheye State Routing in Mobile Ad Hoc Networks," in *ICDCS '00: Proceedings of IEEE ICDCS Workshop on Wireless NEtworks and Mobile Computing*, pp. 71–78, 2000.

[67] G. Pei, M. Gerla, and X. Hong, "LANMAR: Landmark Routing for Large Scale Wireless Ad Hoc Networks with Group Mobility," in *MobiHoc '00: Proceedings of the First ACM International Symposium on Mobile Ad Hoc Networking & Computing*, pp. 11–18, 2000.

[68] P. Samar, M. Pearlman, and S. Haas, "Independent Zone Routing: an Adaptive Hybrid Routing Framework for Ad Hoc Wireless Networks," *IEEE/ACM Transactions on Networking*, vol. 12, no. 4, pp. 595–608, 2004.

[69] Y. Jo and N. Vaidya, "Location-Aided Routing (LAR) in Mobile Ad Hoc Networks," in *MobiCom '98: Proceedings of the Fourth ACM/IEEE Annual International Conference on Mobile Computing and Networking*, pp. 66–75, 1998.

[70] B. Karp and H. Kung, "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks," in *MobiCom '00: Proceedings of the Sixth ACM/IEEE Annual International Conference on Mobile Computing and Networking*, pp. 243–254, 2000.

[71] C. Davis, Z. J. Haas, and S. D. Milnerv, "On How to Circumvent The Manet Scalability Curse," in *MILCOM '06: Proceedings of the IEEE Military Communications Conference*, pp. 23–25, 2006.

[72] C. E. Perkins, *Ad Hoc Networking*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2001.

[73] J. Polastre, R. Szewczyk, and D. Culler, "Telos: Enabling Ultra-low Power Wireless Research," in *IPSN '05: Proceedings of the Fourth ACM/IEEE International Conference on Information Processing in Sensor Networks*, pp. 364–369, 2005.

[74] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless Sensor Network Survey," *Computer Networks*, vol. 52, pp. 2292–2330, Aug. 2008.

[75] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection tree protocol," in *SenSys '09: Proceedings of the Seventh ACM Conference on Embedded Networked Sensor Systems*, p. 1, 2009.

[76] S. Moeller, A. Sridharan, B. Krishnamachari, and O. Gnawali, "Routing Without Routes: The Backpressure Collection Protocol," in *IPSN '10: Proceedings of the Ninth ACM/IEEE International Conference on Information Processing in Sensor Networks*, pp. 279–290, 2010.

[77] G. Tolle and D. Culler, "Design of an Application-cooperative Management System for Wireless Sensor Networks," in *Proceeedings of the Second European Workshop on Wireless Sensor Networks*, pp. 121–132, 2005.

[78] P. K. Dutta, J. W. Hui, D. C. Chu, and D. E. Culler, "Securing the Deluge Network Programming System," in *IPSN '06: Proceedings of the Fifth ACM/IEEE International Conference on Information Processing in Sensor Networks*, pp. 326–333, 2006.

[79] K. Lin and P. Levis, "Data Discovery and Dissemination with DIP," in *IPSN '08: Proceedings of the Seventh ACM/IEEE International Conference on Information Processing in Sensor Networks*, pp. 433–444, Apr. 2008.

[80] T. Dang, N. Bulusu, W.-c. Feng, and S. Park, "DHV: A Code Consistency Maintenance Protocol for Multi-hop Wireless Sensor," *Wireless Sensor Networks*, pp. 327–342, 2009.

[81] I. F. Akyildiz and I. H. Kasimoglu, "Wireless Sensor and Actor Networks: Research Challenges," *Ad Hoc Networks*, vol. 2, pp. 351–367, Oct. 2004.

[82] J. J. P. C. Rodrigues and P. A. C. S. Neves, "A Survey on IP-based Wireless Sensor Network Solutions," *International Journal of Communication Systems*, vol. 23, no. 8, pp. 963–981, 2010.

[83] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. E. Culler, "TinyOS: an Operating System for Wireless Sensor Networks," in *Ambient Intelligence* (W. Weber, J. M. Rabaey, and E. Aarts, eds.), pp. 115–148, Springer-Verlag, 2004.

[84] J. W. Hui and D. E. Culler, "The Dynamic Behavior of a Data Dissemination Protocol for Network Programming at Scale Categories and Subject Descriptors," in *SenSys '04: Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems*, pp. 81–94, 2004.

[85] P. Levis, N. Patel, D. Culler, and S. Shenker, "Trickle : A Self-Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks," in *NSDI '04: Proceedings of the First USENIX/ACM Symposium on Network System Design and Implementation*, 2004.

[86] P. Baronti, P. Pillai, V. W. C. Chook, S. Chessa, A. Gotta, and Y. F. Hu, "Wireless Sensor Networks: a Survey on the State of the Art and the 802.15.4 and ZigBee Standards," *Computer Communications*, vol. 30, pp. 1655–1695, May 2007.

[87] P. Thubert, A. Brandt, J. W. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander, "RFC 6550: RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," 2012.

[88] J.-H. Cho, A. Swami, and I.-R. Chen, "A Survey on Trust Management for Mobile Ad Hoc Networks," *IEEE Communications Surveys & Tutorials*, vol. 13, no. 4, pp. 562–583, 2011.

[89] S. Marti, T. Giuli, K. Lai, and M. Baker, "Mitigating routing Misbehavior in Mobile Ad Hoc Networks," in *Proceedings of the Sixth ACM/IEEE Conference on Mobile Computing And Networking*, pp. 255–265, 2000.

[90] B. Wang, S. Soltani, J. Shapiro, and P. Tab, "Local detection of Selfish Routing Behavior in Ad Hoc Networks," in *Proceedings of the Eighth International Symposium on Parallel Architectures, Algorithms and Networks*, pp. 392–399, 2005.

[91] S. Soltanali, S. Pirahesh, S. Niksefat, and M. Sabaei, "An Efficient scheme to Motivate cooperation in Mobile Ad Hoc Networks," in *ICNS '07: Proceedings of the Third International Conference on Networking and Services*, p. 98, 2007.

[92] C. R. Davis, "A Localized Trust Management Scheme for Ad Hoc Networks," in *ICN '04: Proceedings of the Third International Conference on Networking*, pp. 671–675, 2004.

[93] E. C. H. Ngai and M. R. Lyu, "Trust and Clustering-based Authentication Services in Mobile Ad Hoc Networks," in *Proceedings of the Twenty-fourth International Conference on Distributed Computing Systems Workshops*, pp. 23–24, 2004.

[94] H. Luo, J. Kong, P. Zerfos, S. Lu, and L. Zhang, "URSA: Ubiquitous and Robust Access Control for Ad Hoc Applications," *IEEE/ACM Transactions on Networking*, vol. 12, no. 6, pp. 1039–1063, 2004.

[95] J. Deng, R. Han, and S. Mishra, "INSENS: Intrusion-tolerant routing for wireless sensor networks," *Computer Communications*, vol. 29, pp. 216–230, Jan. 2006.

[96] S.-B. Lee and Y.-H. Choi, "A Secure Alternate Path Routing in Sensor Networks," *Computer Communications*, vol. 30, pp. 153–165, Dec. 2006.

[97] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler, "SPINS: Security Protocols for Sensor Networks," *Wireless Networks*, vol. 8, no. 5, pp. 521–534, 2002.

[98] S. B. Eisenman, *People-Centric Mobile Sensing Networks*. PhD thesis, Columbia University, 2008.

[99] S. S. Kanhere, "Participatory Sensing: Crowdsourcing Data from Mobile Smartphones in Urban Spaces," in *MDM '11: Proceedings of the Twelfth IEEE International Conference on Mobile Data Management*, pp. 3–6, June 2011.

[100] M. Srivastava, T. Abdelzaher, and B. Szymanski, "Human-centric Sensing," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 370, no. 1958, pp. 176–197, 2012.

[101] E. Miluzzo, N. D. Lane, K. Fodor, R. Peterson, H. Lu, M. Musolesi, S. B. Eisenman, X. Zheng, and A. T. Campbell, "Sensing Meets Mobile Social Networks: the Design, Implementation and Evaluation of the CenceMe Application," in *SenSys '08: Proceedings of the Sixth ACM Conference on Embedded Networked Sensor Systems*, pp. 337–350, 2008.

[102] E. Miluzzo, C. T. Cornelius, A. Ramaswamy, T. Choudhury, Z. Liu, and A. T. Campbell, "Darwin Phones: the Evolution of Sensing and Inference on Mobile Phones," in *MobiSys '10: Proceedings of the Eighth ACM International Conference on Mobile Systems, Applications, and Services*, pp. 5–20, 2010.

[103] A. Kapadia, N. Triandopoulos, C. Cornelius, D. Peebles, and D. Kotz, "AnonySense: Opportunistic and Privacy-Preserving Context Collection," in *Pervasive '08: Proceedings of the Sixth International Conference on Pervasive Computing*, pp. 1–18, 2008.

[104] C. Cornelius, A. Kapadia, D. Kotz, D. Peebles, M. Shin, and N. Triandopoulos, "AnonySense: Privacy-aware People-centric Sensing," in *MobiSys '08: Proceedings of the Sixth ACM International Conference on Mobile Systems, Applications, and Services*, pp. 211–224, 2008.

[105] J. Shi, R. Zhang, Y. Liu, and Y. Zhang, "PriSense: Privacy-Preserving Data Aggregation in People-Centric Urban Sensing Systems," in *INFOCOM '10: Proceedings of the Twenty-ninth IEEE International Conference on Computer Communications*, pp. 1–9, Mar. 2010.

[106] K. L. Huang, S. S. Kanhere, and Wen Hu, "Are You Contributing Trustworthy Data? The Case for a Reputation System in Participatory Sensing," in *MSWiM '10: Proceedings of the Tirtheenth ACM International Conference on Modeling, Analysis, and Simulation of Wireless and Mobile Systems*, pp. 14–22, 2010.

[107] S. Reddy, D. Estrin, and M. Srivastava, "Recruitment Framework for Participatory Sensing Data Collections," in *Pervasive '10: Proceedings of the Eighth International Conference on Pervasive Computing*, pp. 138–155, 2010.

[108] W. Mason and D. J. Watts, "Financial Incentives and the "Performance of Crowds"," in *HCOMP '09: Proceedings of the ACM SIGKDD Workshop on Human Computation*, vol. 11, pp. 77–85, 2009.

[109] C. Eickhoff, C. G. Harris, A. P. de Vries, and P. Srinivasan, "Quality through Flow and Immersion: Gamifying Crowdsourced Relevance Assessments," in *Proceedings of the Thirty-fifth International ACM SIGIR Conference on Research and Development in Infromation Retrieval*, pp. 871–880, 2012.

[110] K. K. Rachuri, C. Mascolo, and M. Musolesi, "Energy-Accuracy Trade-offs of Sensor Sampling in Smart Phone Based Sensing Systems," in *Mobile Context Awareness* (T. Lovett and E. O'Neill, eds.), pp. 65–76, London, UK: Springer-Verlag, 2012.

[111] K. Fall, "A Delay-Tolerant Network Architecture for Challenged Internets," in *SIGCOMM '03: Proceedings of the 2003 ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pp. 27–34, 2003.

[112] L. Pelusi, A. Passarella, and M. Conti, "Opportunistic Networking: Data Forwarding in Disconnected Mobile Ad Hoc Networks," *IEEE Communications Magazine*, vol. 44, no. 11, pp. 134–141, 2006.

[113] H. Liu, B. Zhang, H. T. Mouftah, X. Shen, and J. Ma, "Opportunistic Routing for Wireless Ad Hoc and Sensor Networks: Present and Future Directions," *IEEE Communications Magazine*, vol. 47, no. 12, pp. 103–109, 2009.

[114] Y. Wang and H. Wu, "DFT-MSN: the Delay Fault Tolerant Mobile Sensor Network for PErvasive Information Gathering," in *INFOCOM '06: Proceedings of the Twenty-fifth IEEE International Conference on Computer Communications*, 2006.

[115] P. Juang, H. Oki, Y. Wang, M. Martonosi, L.-s. Peh, and D. Rubenstein, "Energy-Efficient Computing for Wildlife Tracking : Design Tradeoffs and Early Experiences with ZebraNet," in *ASPLOS-X '02: Proceedings of the Tenth International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 96–107, 2002.

[116] P. Hui, A. Chaintreau, J. Scott, R. Gass, J. Crowcroft, and C. Diot, "Pocket Switched Networks and Human Mobility in Conference Environments," in *WDTN '05: Proceedings of the 2005 ACM SIGCOMM Workshop on Delay-tolerant Networking*, pp. 244–251, 2005.

[117] S. B. Eisenman, E. Miluzzo, N. D. Lane, R. A. Peterson, G.-S. Ahn, and A. T. Campbell, "BikeNet: a Mobile Sensing System for Cyclist Experience Mapping," *ACM Transactions on Sensor Networks*, vol. 6, pp. 1–39, Dec. 2009.

[118] A. Vahdat and D. Becker, "Epidemic Routing for Partailly Connected Ad Hoc Networks," tech. rep., Department of Computer Science, Duke University, 2000.

[119] B. Burns, O. Brock, and B. N. Levine, “MV Routing and Capacity Building in Disruption Tolerant Networks,” in *INFOCOM ’05: Proceedings of the Twenty-fourth IEEE International Conference on Computer Communications*, 2005.

[120] J. Widmer and J.-Y. Le Boudec, “Network Coding for Efficient Communication in Extreme Networks,” in *WDTN ’05: Proceedings of the 2005 ACM SIGCOMM Workshop on Delay-tolerant Networking*, pp. 22–26, 2005.

[121] M. Musolesi, S. Hailes, and C. Mascolo, “Adaptive Routing for Intermittently Connected Mobile Ad Hoc Networks,” in *WoWMoM ’05: Proceedings of the Sixth IEEE International Symposium on World of Wireless, Mobile and Multimedia Networks*, 2005.

[122] D. Goodman, J. Borras, N. B. Mandayam, and R. D. Yates, “INFOSTATIONS: a new system model for data and messaging services,” in *VTC ’97: Proceedings of the Fourty-Seventh IEEE Vehicular Technology Conference*, pp. 969–973, 1997.

[123] T. Small and Z. J. Haas, “The Shared Wireless Infostation Model: a New Ad Hoc Networking Paradigm (or Where there is a Whale, there is a Way),” in *MobiHoc ’03: Proceedings of the Fourth ACM International Symposium on Mobile Ad Hoc Networking & Computing*, pp. 233–244, 2003.

[124] W. Zhao, M. Ammar, and E. Zegura, “A Message Ferrying Approach for Data Delivery in Sparse Mobile Ad Hoc Networks,” in *MobiHoc ’04: Proceedings of the Fifth ACM International Symposium on Mobile Ad Hoc Networking & Computing*, pp. 187–198, 2004.

[125] MEMSIC, “TelosB Mote Platform Datasheet,” 2011.

[126] A. Dunkels, B. Gronvall, and T. Voigt, “Contiki: a Lightweight and Flexible Operating System for Tiny Networked Sensors,” in *LCN ’04: Proceedings of the Twenty-Ninth Annual IEEE International Conference on Local Computer Networks*, pp. 455–462, 2004.

[127] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y. Choi, T. Herman, S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita, “A Line in the Sand: a Wireless Sensor Network for Target Detection, Classification, and Tracking,” *Computer Networks*, vol. 46, pp. 605–634, Dec. 2004.

[128] G. Camarillo and M. A. García-Martín, *The 3G IP Multimedia Subsystem (IMS)*. Wiley, 2006.

[129] H. Lu, N. D. Lane, S. B. Eisenman, and A. T. Campbell, “Bubble-sensing: Binding Sensing Tasks to the Physical World,” *Pervasive and Mobile Computing*, vol. 6, pp. 58–71, Feb. 2010.

[130] A. J. Perez, M. A. Labrador, S. J. Barbeau, and S. Florida, "G-Sense: A Scalable Architecture for Global Sensing and Monitoring," *IEEE Network*, vol. 24, no. 4, pp. 57–64, 2010.

[131] C.-Y. Wan, S. B. Eisenman, A. T. Campbell, and J. Crowcroft, "Siphon: Overload Traffic Management using Multi-Radio Virtual Sinks in Sensor Networks," in *SenSys '05: Proceedings of the Third ACM Conference on Embedded Networked Sensor Systems*, pp. 116–129, 2005.

[132] O. Gnawali, K.-Y. Jang, J. Paek, M. Vieira, R. Govindan, B. Greenstein, A. Joki, D. Estrin, and E. Kohler, "The Tenet architecture for tiered sensor networks," in *SenSys '06: Proceedings of the Fourth ACM Conference on Embedded Networked Sensor Systems*, pp. 153–166, July 2006.

[133] W. Kang, S. H. Son, and J. A. Stankovic, "Quality-aware data abstraction layer for collaborative 2-tier sensor network applications," *Real-Time Systems*, vol. 48, pp. 463–498, May 2012.

[134] W. Hu, N. Bulusu, C. T. Chou, S. Jha, A. Taylor, and V. N. Tran, "Design and Evaluation of a Hybrid Sensor Network for Cane Toad Monitoring," *ACM Transactions on Sensor Networks*, vol. 5, pp. 1–28, Feb. 2009.

[135] A. Bari, A. Jaekel, and S. Bandyopadhyay, "Clustering Strategies for Improving the Lifetime of Two-tiered Sensor Networks," *Computer Communications*, vol. 31, pp. 3451–3459, Sept. 2008.

[136] C. Hartung, R. Han, C. Seielstad, and S. Holbrook, "FireWxNet: A Multi-Tiered Portable Wireless System for Monitoring Weather Conditions in Wildland Fire Environments Categories and Subject Descriptors," in *MobiSys '06: Proceedings of the Fourth ACM International Conference on Mobile Systems, Applications, and Services*, pp. 28–41, 2006.

[137] M. Vemula, M. F. Bugallo, and P. M. Djuric, "Target Tracking in a Two-Tiered Hierarchical Sensor Network," in *ICASSP '06: Proceedings of the 2006 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 1–5, 2006.

[138] Y. Wang and H. Wu, "Delay/Fault-Tolerant Mobile Sensor Network (DFT-MSN): A New Paradigm for Pervasive Information Gathering," *IEEE Transactions on Mobile Computing*, vol. 6, pp. 1021–1034, Sept. 2007.

[139] M. H. R. Khouzani, S. Eshghi, S. Sarkar, N. B. Shroff, and S. S. Venkatesh, "Optimal Energy-aware Epidemic Routing in DTNs," in *MobiHoc '12: Proceedings of the Tirteenth ACM International Symposium on Mobile Ad Hoc Networking & Computing*, pp. 175–182, 2012.

[140] G. Cardone, A. Corradi, and L. Foschini, “Cross-Network Opportunistic Collection of Urgent Data in Wireless Sensor Networks,” *The Computer Journal*, 2011.

[141] M. Lin, N. D. Lane, M. M. Rabbi, X. Yang, L. Hong, G. Cardone, S. Ali, A. Doryab, E. Berke, T. Choudhury, and A. T. Campbell, “A Scalable Approach for Multidimensional Wellbeing Monitoring: Community and Energy Based Adaptation of Mobile Sensing and Feedback,” in *WH '12: Proceedings of the Third Conference on Wireless Health*, 2012.

[142] T. Schoellhammer, B. Greenstein, and D. Estrin, “Hyper: A Routing Protocol To Support Mobile Users of Sensor Networks,” tech. rep., Center for Embedded Network Sensing (CENS), 2006.

[143] M. Li and Y. Liu, “Underground Coal Mine Monitoring with Wireless Sensor Networks,” *ACM Transactions on Sensor Networks*, vol. 5, no. 2, pp. 1–29, 2009.

[144] M. Ceriotti, L. Mottola, G. P. Picco, A. L. Murphy, S. Guna, M. Corra, M. Pozzi, D. Zonta, and P. Zanon, “Monitoring Heritage Buildings with Wireless Sensor Networks: the Torre Aquila Deployment,” in *IPSN '09: Proceedings of the Eighth ACM/IEEE International Conference on Information Processing in Sensor Networks*, pp. 277–288, 2009.

[145] F. Stajano, N. Hoult, I. Wassell, P. Bennett, C. Middleton, and K. Soga, “Smart Bridges, Smart Tunnels: Transforming Wireless Sensor Networks from Research Prototypes into Robust Engineering Infrastructure,” *Ad Hoc Networks*, vol. 8, no. 8, pp. 872–888, 2010.

[146] J. Li, C. Blake, D. S. J. D. Couto, H. I. Lee, and R. Morris, “Capacity of Ad Hoc Wireless Networks,” in *MobiCom '01: Proceedings of the Seventh ACM/IEEE Annual International Conference on Mobile Computing and Networking*, pp. 61–69, 2001.

[147] R. Ramanathan and J. Redi, “A Brief Overview of Ad Hoc Networks: Challenges and Directions,” *IEEE Communications Magazine*, vol. 40, pp. 20–22, 2002.

[148] D. S. J. D. Couto, D. Aguayo, J. Bicket, and R. Morris, “A High-throughput Path Metric for Multi-hop Wireless Routing,” in *MobiCom '03: Proceedings of the Ninth ACM/IEEE Annual International Conference on Mobile Computing and Networking*, pp. 134–146, 2003.

[149] R. Fonseca, O. Gnawali, K. Jamieson, S. Kim, P. Levis, and A. Woo, “TinyOS Extension Proposal 123 - Collection Tree Protocol,” tech. rep., TinyOS Network Working Group, 2006.

[150] R. M. Karp, “How Much is it Worth to Know the Future?,” in *Proceedings of the IFIP Twelfth World Computer Congress on Algorithms, Software, Architecture*, pp. 416–429, 1992.

[151] A. R. Karlin, M. S. Manasse, L. Rudolph, and D. Sleator, “Competitive Snoopy Caching,” *Algorithmica*, vol. 3, pp. 79–119, 1988.

[152] A. R. Karlin, M. S. Manasse, L. A. McGeoch, and S. Owicki, “Competitive Randmized Algorithms for Non-uniform Problems,” in *Proceedings of the First Annual ACM/SIAM Symposium on Discrete Algorithms*, pp. 301–309, 1990.

[153] L. Sha, F. Kai-Wei, and P. Sinha, “CMAC: an Energy Efficient Mac Layer Protocol Using Convergent Packet Forwarding for Wireless Sensor Networks,” in *SECON '07: Proceedings of the Fourth IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, pp. 11–20, 2007.

[154] D. Moss and P. Levis, “Box-MACs: Exploiting Physical and Link Layer Boundaries in Low-power Networking,” tech. rep., Stanford Information Networks Group, 2008.

[155] J. Polastre, J. Hill, and D. E. Culler, “Versatile Low Power Media Access for Wireless Sensor Networks,” in *SenSys '04: Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems*, pp. 95–107, 2004.

[156] M. Buettner, G. V. Yee, E. Anderson, and R. Han, “X-MAC: a Short Preamble MAC Protocol for Duty-cycled Wireless Sensor Networks,” in *SenSys '06: Proceedings of the Fourth ACM Conference on Embedded Networked Sensor Systems*, pp. 307–320, 2006.

[157] P. K. Dutta, D. E. Culler, and S. Shenker, “Procrastination Might Lead to a Longer and More Useful Life,” in *HotNets VI: Proceedings of the Sixth Workshop on Hot Topics in Networks*, 2007.

[158] G. Zhou, T. He, S. Krishnamurthy, and J. A. Stankovic, “Models and Solutions for Radio Irregularity in Wireless Sensor Networks,” *ACM Transactions on Sensor Networks*, vol. 2, pp. 221–262, 2006.

[159] M. Zúñiga and B. Krishnamachari, “An Analysis of Unreliability and Asymmetry in Low-power Wireless Links,” *ACM Transactions on Sensor Networks*, vol. 3, no. 7, 2007.

[160] K. Srinivasan, P. K. Dutta, A. Tavakoli, and P. Levis, “An Empirical Study of Low-power Wireless,” *ACM Transactions on Sensor Networks*, vol. 6, pp. 1–49, 2010.

[161] G. Cardone, A. Corradi, and L. Foschini, “Reliable Communication for Mobile MANET-WSN Scenarios,” in *ISCC '11: Proceedings*

of the Sixteenth IEEE Symposium on Computers and Communications, ISCC '11, pp. 1085–1091, 2011.

[162] R. L. Knoblauch, M. Petrucha, and M. Nitzburg, "Transportation Research Board Records No. 1538: Studies of Pedestrian Walking Speed and Start-up Time," tech. rep., Transportation Research Board, 1996.

[163] M. Yarvis, N. Kushalnagar, H. Singh, A. Rangarajan, Y. Liu, and S. Singh, "Exploiting Heterogeneity in Sensor Networks," in *INFOCOM '05: Proceedings of the Twenty-fourth IEEE International Conference on Computer Communications*, pp. 878–890, 2005.

[164] A. Chakrabarti, A. Sabharwal, and B. Aazhang, "Observer Mobility for Power Efficient Design of Sensor NEtworks," in *Information Processing in Sensor Networks* (F. Zhao and L. Guibas, eds.), Springer Berlin Heidelberg, 2003.

[165] W. Wang, V. Srinivasan, and K.-C. Chua, "Using Mobile Relays to Prolong the Lifetime of Power Efficient Design of Wireless Sensor Networks," in *MobiCom '05: Proceedings of the Eleventh ACM/IEEE Annual International Conference on Mobile Computing and Networking*, pp. 270–183, 2005.

[166] J. Ma, C. Chen, and J. Salomaa, "mWSN for Large Scale Mobile Sensing," *Journal of Signal Processing Systems*, vol. 51, pp. 195–206, 2008.

[167] S. A. Munir, R. Biao, J. Weiwei, W. Bin, X. Dongliang, and M. Man, "Mobile Wireless Sensor Network: Architecture and Enabling Technologies for Ubiquitous Computing," in *AINAW '07: Proceedings of the Twenty-first International Conference on Advanced Information Networking and Applications Workshops*, pp. 113–120, 2007.

[168] Open IMS team, "The Open Source IMS Core Project." http://www.openimscore.org/.

[169] OpenSIPS Project Team, "The OpenSIPS Project." http://opensips.org/.

[170] University of Cape Town Communications Research Group, "UCT IMS Client." http://uctimsclient.berlios.de/.

[171] U. Lee, E. Magistretti, B. Zhou, M. Gerla, P. Bellavista, and A. Corradi, "Efficient Data Harvesting in Mobile Sensor Platfrms," in *Proceedings of the Fourth Annual IEEE International Conference on Pervasive Compiting and Communications Workshops*, 2006.

[172] 3GPP, "Presence Service: Architecture and Functional Description - TS 23.141 v9.0.0," tech. rep., 3rd Generation Partnership Project, 2009.

[173] R. Shacham, W. Lellerer, H. Schulzrinne, and S. Thakolsri, "Composition for Enhanced SIP Presence," in *ISCC '07: Proceedings of the Twlefth IEEE Symposium on Computers and Communications*, pp. 203–210, 2007.

[174] Texas Instruments, "MSP430F16x datasheet." http://focus.ti.com/lit/ds/symlink/msp430f1611.pdf.

[175] Texas Instruments, "CC2420 Datasheet." http://www.ti.com/lit/gpn/cc2420.

[176] A. B. Roach, "RFC 3265: Session Initiation Protocol (SIP)-specific Event Notification," 2002.

[177] J. Peterson, "RFC 4119: A Presence-based GEOPRIV Location Object Format," 2005.

[178] eXosip Project Team, "eXtended osip library." http://savannah.nongnu.org/projects/exosip.

[179] S. T. Sheu, S. Lu, and Y. T. Lee, "Load Analysis for MTC Devices in Idle Mode or Detached State," tech. rep., Institute for Information Industry (III), Coiler Corporation. 3GPP RAN 2 Meeting Document, 2010.

[180] M. El Barachi, A. Kadiwal, R. Glitho, F. Khendek, and R. A. Dssouli, "Presence-based Architecture for the Integration of the Sensing Capabilities of Wireless Sensor Networks in the IP Multimedia Subsystem," in *WCNC '08: IEEE Wireless Communications and Networking Conference*, pp. 3116–3121, 2008.

[181] A. Outtgarts and O. Martinot, "iSSEE: IMS Sensors Search Engine Enabler for Sensors Mash-ups Convergent Application," *International Journal of Computer Science Issues*, vol. 6, 2009.

[182] M. Elkotob and E. Osipov, "iRide: A Cooperative Sensor and IP Multimedia Subsystem Based Architecture and Application for ITS Road Safety," *Communications Infrastructure. Systems and Applications in Europe*, vol. 16, pp. 153–162, 2009.

[183] B. E. Ainsworth, W. L. Haskell, S. D. Herrmann, N. Meckes, D. R. Bassett, C. Tudor-Locke, J. L. Greer, J. Vezina, M. C. Whitt-Glover, and A. S. Leon, "2011 Compendium of Physical Activities: a second update of codes and MET values.," *Medicine and Science in Sports and Exercise*, vol. 43, pp. 1575–1581, Aug. 2011.

[184] N. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. T. Campbell, "A Survey of Mobile Phone Sensing," *IEEE Communications Magazine*, vol. 48, pp. 140–150, Sept. 2010.

[185] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Activity Recognition Using Cell Phone Accelerometers," *ACM SIGKDD Explorations Newsletter*, vol. 12, no. 2, pp. 74–82, 2010.

[186] T. Brezmes, J.-L. Gorricho, and J. Cotrina, "Activity Recognition from Accelerometer Data on a Mobile Phone," in *Proceedings of the Tenth International Work-Conference on Artificial Neural Networks: Part II: Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living*, pp. 796–799, 2009.

[187] A. Möller, L. Roalter, S. Diewald, J. Scherr, M. Kranz, N. Y. Hammerla, P. Olivier, and T. Plötz, "GymSkill: A Personal Trainer for Physical Exercises," in *Percom '12: Proceedings of the Tenth IEEE International Conference on Pervasive Computing and Communications*, pp. 213–220, 2012.

[188] J. Dai, X. Bai, Z. Yang, Z. Shen, and D. Xuan, "PerFallID: A Pervasive Fall Detection System Using Mobile Phones," in *PerHealth '10: Proceedings of IEEE PerCom Workshop on Pervasive Healthcare*, pp. 292–297, 2010.

[189] T. Denning, A. Andrew, R. Chaudhri, C. Hartung, J. Lester, G. Borriello, and G. Duncan, "BALANCE: Towards a Usable Pervasive Wellness Application with Accurate Activity Inference," in *HotMobile '10: Proceedings of the Tenth Workshop on Mobile Computing Systems & Applications*, pp. 1–6, 2010.

[190] N. Aharony and W. Gardner, "Funf Developer Site." http://www.funf.org, 2012.

[191] N. Ramanathan, F. Alquaddoomi, H. Falaki, D. George, C.-K. Hsieh, J. Jenkins, C. Ketcham, B. Longstaff, J. Ooms, J. Selsky, H. Tangmunarunkit, and D. Estrin, "Ohmage: An Open Mobile System for Activity and Experience Sampling," in *PervasiveHealth '12: Proceedings of the Sixth International Conference on Pervasive Computing Technologies for Healthcare*, 2012.

[192] Google, "Android Developers Site." http://developer.android.com/index.html, 2012.

[193] K. K. Rachuri, C. Mascolo, M. Musolesi, and P. J. Rentfrow, "SociableSense: Exploring the Trade-offs of Adaptive Sampling and Computation Offloading for Social Sensing Categories and Subject Descriptors," in *MobiCom '11: Proceedings of the Seventeenth ACM/IEEE Annual International Conference on Mobile Computing and Networking*, pp. 73–84, 2011.

[194] P. Bellavista, G. Cardone, A. Corradi, and L. Foschini, "The Future Internet Convergence of IMS and Ubiquitous Smart Environments: an IMS-based Solution for Energy Efficiency," *Journal of*

Network and Computer Applications, vol. 35, no. 4, pp. 1203–1209, 2012.

[195] G. Cardone, A. Corradi, L. Foschini, and R. Montanari, "Socio-technical Awareness to Support Recommendation and Efficient Delivery of IMS-enabled Mobile Services," *IEEE Communications Magazine*, vol. 50, pp. 82–90, June 2012.

[196] D. Buysse, C. Reynolds, T. Monk, S. Berman, and D. Kupfer, "The Pittsburgh Sleep Quality Index (PSQI): a New Instrument for Psychiatric Research and Practice," *Psychiatry Research*, vol. 28, no. 2, pp. 193–213, 1989.

[197] G. G. Alvarez and N. T. Ayas, "The Impact of Daily Sleep Duration on Health: a Review of the Literature," *Progress in Cardiovascular Nursing*, vol. 19, no. 2, pp. 56–59, 2004.

[198] A. Schlosberg and M. Benjamin, "Sleep patterns in Three Acute Combat Fatigue Cases," *Journal of Clinical Psychiatry*, vol. 39, pp. 546–548, 1978.

[199] G. Jean-Louis, D. F. Kripke, R. J. Cole, J. D. Assmus, and R. D. Langer, "Sleep Detection with an Accelerometer Actigraph: Comparisons with Polysomnography.," *Physiology & Behavior*, vol. 72, pp. 21–28, Jan. 2001.

[200] Zeo, "MyZeo." http://www.myzeo.com.

[201] Philips, "Fitbit." http://www.fitbit.com.

[202] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA Data Mining Software: an Update," *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[203] J. R. Quinlan, *C4.5: Programs for Machine Learning. Vol. 1.* Morgan Kaufmann, 1993.

[204] C. Cortes and V. Vapnik, "Support-Vector Networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[205] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian Network Classifiers," *Machine Learning*, vol. 29, no. 2-3, pp. 131–163, 1997.

[206] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[207] C.-K. Hsieh, H. Falaki, N. Ramathan, H. Tangmunarunkit, and D. Estrin, "Performance Evaluation of Android IPC for Continuous Sensing Applications," in *HotMobile '12: Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications*, 2012.

[208] H. Cuiting and N. Crespi, "Enriched SCIM for Service Composition within IMS Environment," in *MASS '09: Proceedings of the International Conference on Management and Service Science*, pp. 1–4, 2009.

[209] S. Komorita, M. Ito, H. Yokota, C. Makaya, B. Falchuk, D. Chee, and S. Das, "Loosely Coupled Service Composition for Deployment of Next Generation Service Overlay Networks," *IEEE Communications Magazine*, vol. 50, no. 1, pp. 62–72, 2012.

[210] L. Jianxin, W. Jingyu, W. Bin, and W. Wei, "Toward a Multiplane Framework of NGSON: a Required Guideline to Achieve Pervasive Services and Efficient Resource Utilization," *IEEE Communications Magazine*, vol. 50, no. 1, pp. 90–97, 2012.

[211] J. R. Miglautsch, "Thoughts on RFM Scoring," *The Journal of Database Marketing*, vol. 8, pp. 67–72, 2000.

[212] X. Qi and B. D. Davison, "Web Page Classification: Features and Algorithms," *ACM Computing Surveys*, vol. 41, pp. 1–31, 2009.

[213] M. Efron, "Hashtag Retrieval in Microblogging Environment," in *Proceedings of the Thirty-third International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 787–788, 2010.

[214] B. Yan and G. Chen, "AppJoy: Personalized Mobile Application Discovery," in *MobiSys '11: Proceedings of the Ninth ACM International Conference on Mobile Systems, Applications, and Services*, pp. 113–126, 2011.

[215] W. Woerndl, C. Schueller, and R. Wojtech, "Hybrid Recommender System for Context-aware Recommendations of Mobile Applications," in *ICDE '07: Proceedings of the IEEE Twenty-third International Conference on Data Engineering Workshops*, pp. 871–878, 2007.

[216] I. Konstas, V. Stathopoulos, and J. M. Jose, "On Social Networks and Collaborative Recommendation,," in *Proceedings of the Thirty-second International ACM SIGIR Conference on Research and Development in Infromation Retrieval*, pp. 195–202, 2009.

[217] P. Bellavista, A. Corradi, and L. Foschini, "Self-organizing Seamless Multimedia Streaming in Dense MANETs," *IEEE Pervasive Computing*, 2013.

[218] 3GPP, "IP Multimedia Subsystem (IMS); Stage 2 - TS 23.228 V11.3.0," tech. rep., 3rd Generation Partnership Project, 2011.

[219] Mobicents Project Team, "Mobicents JAIN SLEE." http://www.mobicents.org/slee/intro.html.

[220] Red Hat inc., "jBPM." http://www.jboss.org/jbpm/.

[221] P. Bellavista, A. Corradi, and L. Foschini, "IMS-compliant Management of Vertical Handoffs for Mobile Multimedia Session Continuity," *IEEE Communications Magazine*, vol. 48, no. 4, pp. 114–121, 2010.

[222] M.-R. Ra, B. Liu, T. La Porta, and R. Govindan, "Medusa: a Programming Framework for Crowd-Sensing Applications," in *MobiSys '12: Proceedings of the Tenth ACM International Conference on Mobile Systems, Applications, and Services*, pp. 337–350, 2012.

[223] M. S. Bernstein, D. R. Karger, R. C. Miller, and J. Brandt, "Analytic Methods for Optimizing Realtime Crowdsourcing," in *CI '12: Proceedings of Collective Intelligence*, 2012.

[224] S. Ertekin, H. Hirsh, and C. Rudin, "Learning to Predict the Wisdom of Crowds," in *CI '12: Proceedings of Collective Intelligence*, 2012.

ACKNOWLEDGEMENTS

«The most exciting phrase to hear in science, the one that heralds new discoveries, is not "Eureka!" but "That's funny..."»

— Isaac Asimov

First, I would like to thank my advisor prof. Antonio Corradi and Dr. Luca Foschini for their guide during my Ph.D. studies. They followed my work with patience and providing endless encouragements. I also would like to thank prof. Andrew Campbell and prof. Tanzeem Choudhury for accepting me as visitor student at Dartmouth College and inviting me to work on his projects. I also want to thank the people that I met at Dartmouth College: Emiliano Miluzzo, Hong Lu, Tianyu Wang, Mu Lin, Mashfiqui Rabbi. Great thanks also go to prof. Cristian Borcea and Manoop Talasila for having me as collaborator to the McSense project. Finally, I want to thank my thesis reviewers for their precious feedback on this dissertation.

Thanks to my family for letting me play with chemicals, electricity, and the very first home computers and for putting up with the contraptions that I scattered in our home. Not being afraid of taking things apart to see how they work has been the most formative and funny experience ever.

Special thanks to my girlfriend Giuliana for being the adorable epitome of beauty and nerdiness: roses are #FF0000 violets are #0000FF all my base are belong to you.

Thanks to my friends Massimo, Massimiliano, Jo, and Ciro for being like brothers to me. Bonus points to Ciro for being my roommate for five years and making University years unforgettable and for waking me up in unorthodox ways. A shootout to Alice e Iole: you're awesome! Thanks also to all my high school friends, especially (in rigorous random order) Emanuele, Federica, Roberto, Eleonora, and Fabiana: you're among the nicest people that I know and I owe you a lot of beautiful memories. Thanks to my fellow Ph.D. students Mario, Andrea, and Primiano and our partners in crime Ghedo and Alessio: I'll miss our lunches (and sometimes dinners and nights at the pub) talking about protocol stacks, CPU designs, and software news; you make me feel like we're almost normal. Thanks to the lovely people I met while studying in the U.S.A.: Doug, Yan, Chris, and Dhana. CTCP THANKS to the fellow geeks on IRC.

Thanks to everyone I forgot to mention here: I owe you a beer.