UNIVERSITÀ DEGLI STUDI DI PALERMO

DOTTORATO DI RICERCA IN INGEGNERIA INFORMATICA

DIPARTIMENTO DI INGEGNERIA INFORMATICA

A COGNITIVE ARCHITECTURE FOR AMBIENT INTELLIGENCE

Tutor
Prof. Giuseppe Lo Re

Candidato
Dott.ssa Alessandra De Paola

Coordinatore
Ch.mo Prof. Salvatore Gaglio

Tesi di Dottorato di Ricerca in Ingegneria Informatica – XXII Ciclo
SSD: ING-INF/05

Ph.D. Thesis

Alessandra De Paola

February, 15 2011

This work is dedicated to my husband Adriano, for his love and his endless support, and to my daughter Aurora, for teaching me what it really matters in life.

Sommario

L'Ambient Intelligence (AmI) è caratterizzata dall'uso di sistemi pervasivi per monitorare l'ambiente e modificarlo secondo le esigenze degli utenti e rispettando vincoli definiti globalmente. Questi sistemi non possono prescindere da requisiti come la scalabilità e la trasparenza per l'utente. Una tecnologia che consente di raggiungere questi obiettivi è rappresentata dalle reti di sensori wireless (WSN), caratterizzate da bassi costi e bassa intrusività. Tuttavia, sebbene in grado di effettuare elaborazioni a bordo dei singoli nodi, le WSN non hanno da sole le capacità di elaborazione necessarie a supportare un sistema intelligente; d'altra parte senza questa attività di pre-elaborazione la mole di dati sensoriali può facilmente sopraffare un sistema centralizzato con un'eccessiva quantità di dettagli superflui.

Questo lavoro presenta un'architettura cognitiva in grado di percepire e controllare l'ambiente di cui fa parte, basata su un nuovo approccio per l'estrazione di conoscenza a partire dai dati grezzi, attraverso livelli crescenti di astrazione. Le WSN sono utilizzate come strumento sensoriale pervasivo, le cui capacità computazionali vengono utilizzate per pre-elaborare i dati rilevati, in modo da consentire ad un sistema centralizzato intelligente di effettuare ragionamenti di alto livello.

L'architettura proposta è stata utilizzata per sviluppare un testbed dotato degli strumenti hardware e software necessari allo sviluppo e alla gestione di applicazioni di AmI basate su WSN, il cui obiettivo principale sia il risparmio energetico. Per fare in modo che le applicazioni di AmI siano in grado di comunicare con il mondo esterno in maniera affidabile, per richiedere servizi ad agenti esterni, l'architettura è stata arricchita con un protocollo di gestione distribuita della reputazione.

È stata inoltre sviluppata un'applicazione di esempio che sfrutta le caratteristiche del testbed, con l'obiettivo di controllare la temperatura in un ambiente lavorativo. Quest'applicazione rileva la presenza dell'utente attraverso un modulo per la fusione di dati multi-sensoriali basato su reti bayesiane, e sfrutta questa informazione in un controllore fuzzy multi-obiettivo che controlla gli attuatori sulla base delle preferenze dell'utente e del risparmio energetico.

Abstract

Ambient Intelligence (AmI) systems are characterized by the use of pervasive equipments for monitoring and modifying the environment according to users' needs, and to globally defined constraints. Furthermore, such systems cannot ignore requirements about ubiquity, scalability, and transparency to the user. An enabling technology capable of accomplishing these goals is represented by Wireless Sensor Networks (WSNs), characterized by low-costs and unintrusiveness. However, although provided of in-network processing capabilities, WSNs do not exhibit processing features able to support comprehensive intelligent systems; on the other hand, without this pre-processing activities the wealth of sensory data may easily overwhelm a centralized AmI system, clogging it with superfluous details.

This work proposes a cognitive architecture able to perceive, decide upon, and control the environment of which the system is part, based on a new approach to knowledge extraction from raw data, that addresses this issue at different abstraction levels. WSNs are used as the pervasive sensory tool, and their computational capabilities are exploited to remotely perform preliminary data processing. A central intelligent unit subsequently extracts higher-level concepts in order to carry on symbolic reasoning. The aim of the reasoning is to plan a sequence of actions that will lead the environment to a state as close as possible to the users' desires, taking into account both implicit and explicit feedbacks from the users, while considering global system-driven goals, such as energy saving. The proposed conceptual architecture was exploited to develop a testbed providing the hardware and software tools for the development and management of AmI applications based on WSNs, whose main goal is energy saving for global sustainability. In order to make the AmI system able to communicate with the external world in a reliable way, when some services are required to external agents, the architecture was enriched with a distributed reputation management protocol.

A sample application exploiting the testbed features was implemented for addressing temperature control in a work environment. Knowledge about the user's presence is obtained through a multi-sensor data fusion module based on Bayesian networks, and this information is exploited by a multi-objective fuzzy controller that operates on actuators taking into account users' preference and energy consumption constraints.

Acknowledgments

I would like to express my sincere gratitude to my advisor Prof. Giuseppe Lo Re for his continuous encouragement and precious suggestions during my Ph.D study and research. His guidance helped me since the beginning of my study.

I am heartily thankful to Prof. Salvatore Gaglio, whose support enabled me to develop the main idea at the basis of my work.

Special thanks to Dr. Marco Ortolani, for his advise and for sharing with me his invaluable experience during our close collaboration.

I would like to thank all members of the Networking Research Group, Alfonso Farruggia, Orazio Farruggia and Fabrizio Milazzo, for the several useful discussions and their help with the experimental setup.

I am beholden to all my friends at the Department of Computer Engineering for their moral support.

Finally, I will forever be grateful to my parents, my sisters and my husband for their patience and encouragement, in the most crucial moments. Without their support and their sacrifices this research project would not have been possible.

Contents

List of Figures

List of Tables

Glossary

AmI	Ambient Intelligence
CN	Consumer Node
DB	Data Base
DHT	Distributed Hash Table
D-SOA	Distributed Service Oriented Architecture
ICT	Information and Communication Technology
IR	InfraRed
LG	Local Gateway
LIRC	Linux Infrared Remote Control
MSN	Manager Super Node
PET	Physiological Equivalent Temperature
PN	Provider Node
QoS	Quality of Service
RFId	Radio Frequency Identification
RSSI	Received Signal Strength Indicator
SOA	Service Oriented Architecture
SSN	Seeker Super Node
TLG	Top Level Gateway
WSN	Wireless Sensor Network

Chapter 1

Introduction

The main goal of Ambient Intelligence (AmI) is the development of systems aimed at adapting the surrounding environmental conditions so that they can match the users' needs, whether those are consciously expressed or not, while at the same time satisfying other system-driven goals, such as the minimization of global energy consumption. An implicit requirement is the use of pervasively deployed sensing and actuating devices, following the ubiquitous computing paradigm which states that technology must not intrude into human lives; hence, control and monitoring devices should be deployed so as to remain invisible to the users (Weiser, 1995; Ducatel et al., 2001). Wireless Sensor Networks (WSNs) fully meet these requirements, thanks to their intrinsic pervasiveness and low intrusiveness (Akyildiz et al., 2002; Cook et al., 2009; Benini et al., 2006), and may thus represent a suitable choice for the sensory layer of AmI systems.

This work presents SENSOR9k, a comprehensive architecture for designing and experimenting with WSN-based Ambient Intelligence applications whose main goal is to implement effective policies for energy saving in the context of indoor environments. The name of the architecture is meant to emphasize its pervasiveness, as it ideally recalls the fictional *HAL 9000* AI system, whose extremities pervaded the spaceship in "2001: A Space Odyssey'.

Moreover, the proposed system is able to communicate with the external world in order to access to services provided by other agents over the Internet, or by other intelligent buildings over the same "smart city". This communication is made secure and reliable thanks to the definition of a distributed reputation management system, that allows the network of intelligent buildings to construct a distributed opinion about available services.

1.1 Motivations and Goals

Ambient Intelligence is a novel design paradigm in Artificial Intelligence that introduces a shift in perspective as regards the role of the end user (Remagnino and Foresti, 2005; Aarts and Encarnação, 2006). Unlike other well established approaches, such as the human-in-the-loop design, where the contribution resulting from the exploitation of the human factor is limited to facilitate the system design process, or to infer more accurate models for the environment state, Ambient Intelligence aims to fully integrate the user's preference into the system. In this respect, the basic intrinsic requirement of any AmI system is the presence of pervasive and unobtrusive sensory devices (Ducatel et al., 2001), which is essential to ensure context-aware reasoning in order to act upon the environment, modify its state, and react to user-driven stimuli. Thus, the human user become the center of a pervasive digital intelligent environment, whose primary goal consists in satisfying users' requirements as regards controlling the conditions of their surroundings.

Today's advances in technology allow for cheap and unintrusive sensors that may be profitably employed as a distributed pervasive sensory means permeating the whole environment under observation. In this work, we discuss the use of Wireless Sensor Networks (Akyildiz et al., 2002; Estrin et al., 2001) to get precise and continuous monitoring of the physical quantities of interest; not only does this novel technology allow to perform remote sensing without causing disruption, but it may also perform basic in-network pre-processing of sensed data thanks to the limited computational capabilities of the nodes.

A WSN is made up of a potential large number of distributed computational units; those small sensor nodes are programmable, energetically autonomous, and able to wirelessly communicate with each other; moreover, they may be equipped with different sensors in order to measure several environmental characteristics, and ad-hoc sensors may be devised for specialized tasks; for instance, sensor nodes may be integrated with sensors for IR signals, sensors for monitoring polluting agents, or RFId readers. By exploiting the cooperation among nodes, a WSN allows for low-level pre-processing of the sensed data, in order to select, for instance, only relevant information from the huge amount of measurements. The possibility of a low-cost and low-intrusiveness, but pervasive deployment paves the way for the development of a ubiquitous and scalable system which is one of the primary requirements for Ambient Intelligence applications (Basten et al., 2003). The sensory subsystem permeates the environment almost transparently to the users; moreover, the possibility of tuning the execution of the program running in the sensor nodes on the fly, allows us to modify the behavior of the WSN without a direct intervention. However, despite their potential, WSNs cannot by themselves be a tool for collecting and, above all, *understanding* all of the sensed data.

WSNs are however just one part of a comprehensive architecture aimed at overcoming the difficulty of efficiently managing the enormous stream of sensed data without overwhelming the upper-level reasoner with irrelevant details.

This work proposes a novel cognitive system able to perceive, decide upon, and control the environment of which it is part. Such a system may be regarded as an intelligent entity embodied in the environment itself, and whose decisions are guided by goals related to the well-being both of the system, and of the other entities populating the environment. This intelligent organism employs WSNs as its sensory organ in order to perceive precise information on the environment. This technology enables the system to collect measurements at the preferred rate regardless of space constraints. The difficulty of managing the generated large amount of data requires the design of a new architectural scheme capable to make full use of the programmability of the sensor nodes. The idea underlying our work is inspired to the nervous system of complex biological organisms, that typically include some peripheral pre-processing mechanisms for extracting more meaningful information from the wealth of data. Striking examples may be found in some parts of the human nervous system, whose peripheral component deals with collecting sensory inputs, filtering them, and transferring them in an aggregated form to the central nervous system, where high-level processing will be performed. The cognitive architecture we propose here exploits its distributed sensory component in order to obtain necessary information to carry on cognitive, decision, and control activities.

In other words, the pervasive sensory infrastructure may be profitably used to gather information about the environmental conditions, into a centralized server, where artificial reasoning techniques may be implemented. Centralizing the reasoning activity preserves its consistency and unitarity (Amigoni et al., 2004), and allows to steer the behavior of the distributed actuators in order to bring the environment to the desired state.

The proposed architecture encompasses both hardware and software issues. Besides the pervasive sensory devices, SENSOR9k provides a minimal set of communication and processing devices, organized into a backbone of local gateways providing access to the remote WSNs; such intermediate infrastructure is designed in a hierarchical fashion in order to accommodate scalability and fault tolerance, and its main purpose is to act as a connection interface bridging the gap between the distributed sensors and the centralized AmI server. SENSOR9k's core is thus represented by a middleware, partly distributed on the remote sensory devices and on the backbone gateways, and partly residing on the central AmI server, which stores the library of modules implementing basic AmI functionalities.

This central component of the system is also inspired to the functional organization of human brain. Several studies in neurosciences (Tononi and Edelman, 1998; Kandel et al., 1995) proved that different brain areas are functionally specialized for well-defined tasks for sensory signal processing. Besides functional specialization, also functional integration is performed in the different areas, and at different spatial and temporal scale. This suggests the design of a hierarchical and modular architecture, whose components operate independently and in parallel on different environmental stimuli in order to provide a symbolic representation of them. The interconnection among the different modules lets lower-lever modules transfer their knowledge as input for higher-level ones that accept several simpler information streams and integrate them to provide a complex representation of the environment.

This modular organization further aims to boost the development of AmI applications, by providing an abstraction towards the physical layer through a composition of core services that will effectively let the AmI designer focus on higher-level issues; in this perspective, SENSOR9k's modules can be regarded as "building blocks" that implement basic intelligent functionalities on top of the underlying distributed sensory and actuating infrastructure.

Modules are organized according to a multi-tier cognitive scheme, similarly to what happens in the functional areas of the brain that are divided into functional clusters of neurons operating at increasing degrees of abstraction (Tononi and Edelman, 1998). This organization aims to gradually reduce the amount of data to be processed at each level, while increasing the information content of each information element. Our scheme comprises three tiers representing knowledge in a *subsymbolic*, *conceptual*, and *symbolic* way respectively.

In brief, the remote, distributed sensory device thus acts as the termination of a centralized sentient reasoner, where actual intelligent processing occurs; sensed data is processed in order to extract higher-level information, carrying on symbolic reasoning on the inferred concepts, and producing the necessary actions to adapt the environment to the users' requirements. Besides providing basic information about environmental conditions, the WSN allows to observe the interaction between the user and the surrounding environment, in order to model the users' actions, and to infer users' requirements about environmental conditions. According to the constructed model, the system will plan the sequence of actions to be performed in order to achieve the desired status. A set of actuators finally takes care of putting the planned modifications to the environment state into practice.

The architecture described here has been purposefully designed so as to be easily specialized in different application scenarios such as industrial, social, or home environments. In order to assess the validity of the proposed approach, this work presents a fully featured sample application, addressing temperature control in the context of a work environment, and involving conflicting goals, namely the satisfaction of the users' preferences in terms of pleasantness of the office environmental conditions, while minimizing the global energy consumption. It will be shown how SENSOR9k eases the development of such application; in particular a multi-objective fuzzy controller will be created by exploiting the basic

SENSOR9k's functionalities.

The importance of managing the energy consumption is confirmed by several recent studies showing that ICT technologies, and AmI systems in particular, may play a twofold role since their constituting elements are both significant consumers, and potential actors in steering a more clever overall usage of the available energy resources (Vastamaki et al., 2005; Hagras et al., 2008; Kastner et al., 2010; Mozer, 1998).

The architecture described so far allows to develop AmI systems able to fully understand its internal conditions, also those regarding its occupants, and to act in order to satisfy its goal. Nevertheless these introspective capabilities of monitoring, understanding and acting are not enough in order to meet user's requirements; the AmI system has to be able to communicate with the external world in order to obtain services that are useful for reaching its own goals. Just as an example, we can consider systems for automatic food supply, or for communicating with hospitals and first aid centres, or for communicating with energy providers in order to gather information about possible constraints on monthly average consumptions and scheduled blackouts; moreover, we can devise systems able to communicate with several provider for a give service select the most convenient economic conditions.

The most suitable architectural model in order to support the design and development of a intelligent building, as a node of the "smart city", and thus able to communicate with other agents in order to exchange products and services, is the Service Oriented Architecture (SOA) model. In the devised smart city the AmI system represents a consumer node; this agent needs to acquire distributed services from unknown service providers on the Internet and on the network covering the smart city.

Over the Internet, SOAs are typically implemented through the use of web services standards, and rely on a centralized approach, that requires the presence of a master node (trusting authority) maintaining relevant information about network services and the relative providers. This approach suffers from well-known limits of centralized systems, i.e. lack of scalability and presence of a single point of failure.

Distributed SOA architectures (D-SOA) Banaei-Kashani et al. (2004) represent an important evolution of classic SOAs and can overcome their limits using a hierarchical network structure, for distributing workload among several network nodes. This architectural paradigm is well-suited in those scenarios in which trusting authority is implicitly distributed.

The main goal of consumer nodes is the selection of the best services among the huge multitude provided by the network. As basic criteria for this choice, service cost and Quality-of-Service (QoS) can be considered, provided that the underlying SOA be augmented in order to support the declaration of this information.

However, in distributed environments where heterogeneous agents collaborate and interact in order to achieve their own goals, obtaining guarantees on their behavior from some reliable central authority is typically unfeasible. In such scenarios, a reputation management system capable of building a profile representing the reliability of each agent can be extremely useful. Knowing agent reputation is particularly helpful for detecting those agents that are deceitful or potentially dangerous for the community. In a non-centralized environment, interactions among agents are unpredictable and no central authority is present to carry out supervision and coordination activities. In this case, the lack of a centralized trusting authority may encourage antisocial behaviors, that in this scenario consist in declaring false QoS values. Thus, the correct behavior of such new SOA platforms, however, will depend on the presence of some mechanisms that allow consumer nodes to evaluate trustworthiness of service providers.

This work proposes a new methodology for discouraging antisocial behaviors, over an architecture fully distributed over the network, based on a reputation management schema. We present a general approach for reputation management in distributed environment and then we specialize it for the scenario of a distributed SOA, that represents a suitable model for a global AmI environment.

1.2 Contributions

The main contributions of the work presented in this dissertation are:

- The design of a cognitive architecture relying on a flexible and scalable paradigm for knowledge representation in order to efficiently extract environmental information from a pervasive sensory system and to turn it into a symbolic representation of the environment.

- The design and development of SENSOR9k, a comprehensive architecture for designing and experimenting with WSN-based Ambient Intelligence applications whose main goal is to implement effective policies for energy saving in the context of indoor environments. Besides the pervasive sensory devices, SENSOR9k provides a minimal set of communication and processing devices, organized into a backbone of local gateways providing access to the remote WSNs; such intermediate infrastructure is designed in a hierarchical fashion in order to accommodate scalability and fault tolerance, and its main purpose is to act as a connection interface bridging the gap between the distributed sensors and the centralized AmI server. SENSOR9k's core is thus represented by a middleware, partly distributed on the remote sensory devices and on the backbone gateways, and partly residing on the central

AmI server, which stores the library of modules implementing basic AmI functionalities.

- The design and development of a sample application showing the effectiveness of the proposed architecture. The sample application is composed by two modules and its main goal is the management of ambient temperature in order to meet the user's requirements and to minimize energy consumption. The first module is devoted to detect user's presence and relies on a Bayesian network for performing a multi-sensor data fusion. The context information produced by this module is provided as input to a planning module whose aim is to control actuators, based on a fuzzy controller. The rules of the fuzzy controllers are adaptively changed according to user's implicit feedbacks.

- The design and development of a hierarchical SOA that enables AmI applications to communicate with the external world in order to access services from external agents. This communication architecture is made reliable through a distributed reputation management system that allows the intelligent building to select the best services according to the declared QoS.

1.3 Dissertation Outline

The remainder of the dissertation is organized as follows. Chapter 2 describes the proposed architecture for simplifying the construction of complete AmI applications by providing basic hardware and software tools that can be composed and extended in order to build an intelligent comprehensive entity for controlling the environment. Chapter 3 presents a simple AmI application aimed at illustrating the use of the various hardware and software components of main architecture; the main goal is to provide a proof-of-concept of the complete sensing-reasoning-acting loop. Chapter 4 described a reliable communication architecture that enables AmI application to communicate with the external world in order to obtain necessary services from external agent. Through this contribution an AmI system can act as an intelligent node of a bigger "smart city". Finally, Chapter 4.2.4 states some conclusion about this work.

1.4 Publications

Parts of the work in this thesis have been published in several referred conference proceedings:

- A. De Paola, S. Gaglio, G. Lo Re, and M. Ortolani. Human-ambient interaction through Wireless Sensor Networks, *in Proceedings of the 2nd Conference on Human System Interactions (HSI)*, Catania, Italy, 2009.

- A. De Paola, A. Farruggia, S. Gaglio, G. Lo Re, and M. Ortolani. Exploiting the Human Factor in a WSN-based System for Ambient Intelligence, *in Proceedings of third International Conference on Complex, Intelligent and Software Intensive Systems (CISIS)*, Fukuoka, Japan, 2009.

- A. De Paola, S. Gaglio, G. Lo Re, M. Ortolani. An Ambient Intelligence Architecture for Extracting Knowledge from Distributed Sensors. *Proceedings of the International Conference on Interaction Sciences (ICIS)*, Seoul, Korea, 2009.

- A. De Paola, A. Tamburo. Reputation Management in Distributed Systems. *Proceedings of the 3rd International Symposium on Communications, Control and Signal Processing (ISCCSP 2008)*, St. Julians, Malta, 2008.

- C. Crapanzano, F. Milazzo, A. De Paola, and G. Lo Re. Reputation Management for Distributed Service-Oriented Architectures. *Workshop at the Fourth IEEE International Conference on Self-Adaptive and Self-Organizing Systems 2010 (TSOS 2010)*, Budapest, Hungary, 2010.

Chapter 2

System Architecture

The proposed architecture is aimed at simplifying the construction of complete AmI applications by providing basic hardware and software tools that can be composed and extended in order to build an intelligent comprehensive entity for controlling the environment.

We consider as case study a home automation application instantiated for a work environment, with the aim to provide constant monitoring of the environmental conditions in the rooms of the teaching staff of a University Department. After describing the designed WSN, representing the peripheral system that permeates the environment, and allows for distributed data pre-processing, this Chapter outlines the modular structure of the intelligent system.

2.1 Related Work

Complete testbeds for experimenting with AmI applications in the context of smart environments are reported in literature, as widely surveyed in (Cook et al., 2009; Cook and Das, 2007); each of them proposes an ad-hoc approach to some specific scenario, although it is possible to abstract some common functionalities that may be considered as the basic necessary tools for building the overall intelligent behavior. The MavHome (Cook and Das, 2004) has been designed following the agent-oriented approach, and is composed by a set of agents able to communicate through a hierarchical interconnection schema for control and information flow. The main research connected to this project concerns the prediction of users' activity exploiting past collected motion and lighting information, with the aim of performing an automating environmental control. The Aware Home (Kientz et al., 2008) is a living laboratory for researching how ubiquitous computing can support everyday home life for elderly people. The project focus is devoted to sensing users' interaction with the surrounding environment. This project includes systems for human position tracking through various hardware, such as ultrasonic, video, and floor sensors.

Finally, the iDorm research (Holmes et al., 2002) considers the scenario of a student bed-sitting room that provides the normal furniture arrangement found in a typical student study/bedroom environment, including bed, work desk and wardrobe, and allows the simulation of different activities like sleeping, working, and entertaining. The whole environment is purposely constructed in order to implement the designed testbed, given that a great number of wires and networked devices are hidden above the ceiling and behind the walls. The Gator Tech Smart House research (Helal et al., 2005) targets instead a single-family home, permeated by a wide set of sensors and actuators that can be automatically integrated through a generic middleware, that allows application programmers to assemble provided services in order to achieve their own goals, named programmable pervasive spaces. Both these works present flexible and expandable architectures thanks to the use of a middleware layer that gives a homogeneous representation of heterogeneous physical devices. However, in order to build an environment with the intended functionalities a preliminary design of the physical deployment must be carefully planned in advance, possibly requiring heavy modifications to some pre-existent premises, because of the non negligible level of intrusiveness of some specialized devices, such as for instance the *smart floor*.

2.2 Bio-Inspired Architecture

The architecture proposed in this work is inspired by the human nervous system, in which signals gathered by the peripheral system are filtered, aggregated and then sent to the central system for high-level processing.

2.2.1 Related Work

Several works on Ambient Intelligence are inspired to biological models for reasoning and learning. In some cases the biological model is taken as an example for the formalization of a logical architecture reflecting the logical structures that concur to the arising of consciousness, as described by cognitive science research; in other works, the starting point is the capability of learning through the interaction with the surrounding environment that is typical of complex biological systems.

In (Marchesotti et al., 2005), a logical structure for an Ambient Intelligence system is proposed that is inspired to the neuro-biological model of human brain, The authors focus on the use of contextual knowledge for the classification of events occurring in the considered environment, with the aim of facilitating intrusion detection. The classification step is based on an initial off-line training, based on a significant amount of training data, followed by an on-line phase, where a human operator provides explicit feedback about the quality of classification, so that the system may dynamically adapt its parameters. The same architectural scheme has been proposed in (Dore et al., 2007) and applied to the "classification of risk zones in a smart space"; the learning phase of the classifier is inspired here on the biological mechanisms that exploit memory of the past interactions between the intelligent entity and the other entities in the environment for learning.

In (Doctor et al., 2005), the authors propose an application of an unsupervised learning technique based on fuzzy logic to the intelligent agents constituting the Ambient Intelligence system. The fuzzy rules are learned by examining the users' behavior and are dynamically changed so that long-term goals may be satisfied. The inputs for the learning machine are gathered via the interactions between the user and the actuators allowing for manual environmental control.

With respect to the previously mentioned works, we also refer to a biological model to design our architecture. Unlike (Marchesotti et al., 2005) and (Dore et al., 2007), we do not delve into the deep mechanisms regulating the arising of consciousness, rather we propose an architecture inspired to the hierarchical model for processing sensory stimuli in the human nervous system. On the other hand, a higher similarity between our proposal and the previously cited works is represented by the idea of exploiting feedback from users in order to adapt the system behaviour. In particular, we devised a system based both on explicit feedback, similarly to what proposed in (Marchesotti et al., 2005; Dore et al., 2007), and to implicit feedback, similarly to (Doctor et al., 2005).

2.2.2 Peripheral Information Processing - WSN

We regard the aggregation and selection of environmental data as analogous to the processing of perceptual signals occurring in the human nervous system. Some components of the peripheral system filter perceptual information by means of distributed processing among several neurons. A remarkable example is the processing of visual information occurring in the retina (Kandel et al., 1995): in the human eye, photoreceptors convert light into electrical signals that are passed to a network of retinal neurons, and are modified before being transmitted to gangliar neurons; eventually, they are handed to the optic nerve that carries the information up to the brain. The retinal neuron network does not restrict itself to carrying signals from photoreceptors, but rather combines them to obtain an aggregate heavily dependent on the spatial and temporal features of the original light signal.

In the proposed architecture the terminal sensory component performing is represented by WSNs pervasively deployed in the environment. Figure 2.1, partially drawn from (y Cajal, 1911), highlights the similarity between the structures of the human visual organ and of the WSN employed here.

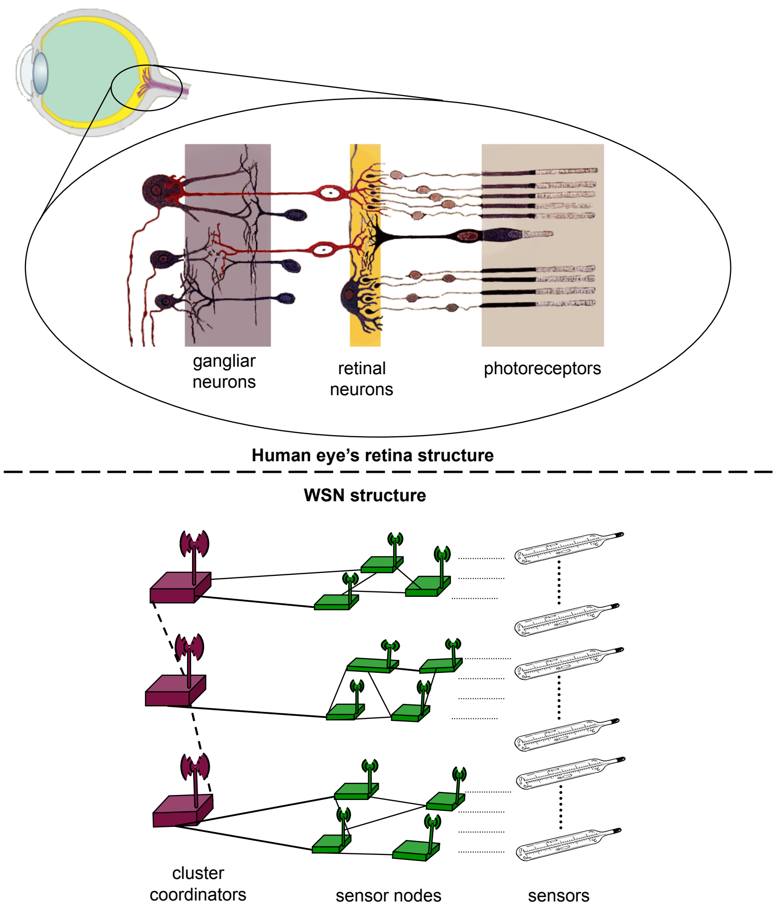


Figure 2.1: Comparison between the structures of the human retina and the proposed WSN.

This work proposes a clustered network structure in which each small cluster, constituted by heterogeneous nodes with different computational capabilities, distributedly processes homogeneous data. This pre-processing phase exploits spatio-temporal correlation of data, in order to compute a model that nodes will share, thanks to their cluster coordinator, similarly to the approach proposed in (Goel et al., 2006). This process serves the two-fold purpose of reducing the number of unnecessary transmissions (only data not fitting the model will be transmitted in order to update the model itself), and of performing a dimensionality reduction that is used to preserve only relevant features.

The implemented WSN is equipped with off-the-shelf sensors for measuring such quantities as temperature, relative humidity, noise, and ambient light exposure. Sensor nodes (in our implementation we have used MICAz nodes and

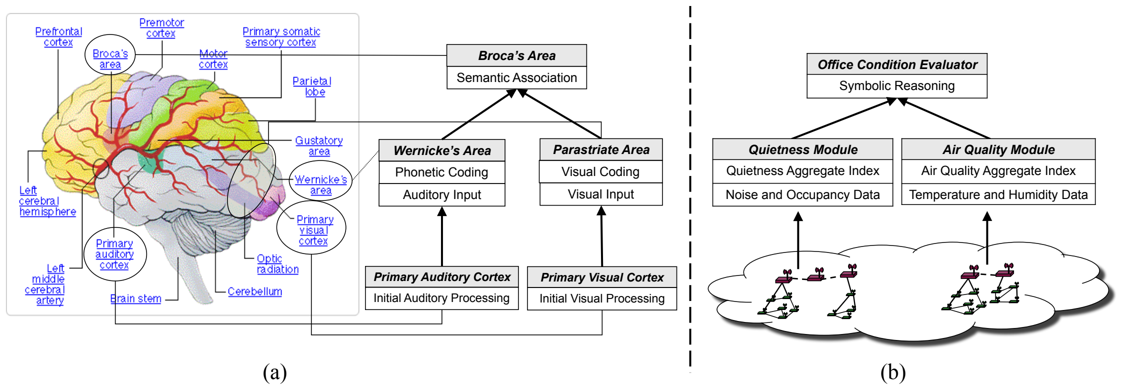


Figure 2.2: The human language comprehension model vs the proposed hierarchical reasoning model.

Stargate microservers) have been deployed in various rooms close to “sensitive” areas: by the door, by the window, and by the user's desk. Moreover, we are integrating this basic infrastructure with more specific sensors, i.e. RFId readers, that will provide information to be used for access control, and naive localization of people inside the premises.

2.2.3 Central Information Processing - Modular Architecture

The proposed system is organized according to a hierarchical structure whose modules are combined together in order to carry on specific reasoning on the environment at different levels of abstraction and on different kinds of perceptions. The overall behavior mimics that of the human brain, where the emerging complex behavior is the result of the interaction among smaller subsystems. From the design point of view, the modular organization allows for the realization of a scalable software architecture, able to effectively manage the huge amount of sensory data.

Figure 2.2, partially taken from (American Medical Association, 2010), draws a parallel between the human brain model and our system model. In our modular architecture, the outcome of lower-level reasoning is fed into the upper levels, that deal with the integration of information originated by multiple lower-level modules. Each module independently measures environmental quantities, conceptualizes them, and describes the extracted concepts linguistically, as will be detailed in the following sections where the multi-tier knowledge representation is presented. Moreover various modules process both direct and indirect measurements; the former occur at modules located at the lowest level in the hierarchy, while the latter are carried on at the upper layers, mediated by their lower-layer counterparts.

Considering a particular scenario, the human language comprehension model, described in (Kandel et al., 1995), provides a significant example of interaction patterns among specific areas of the brain, as schematically presented in the left side of Figure 2.2. Different anatomic structures are devoted to different phases of language processing: the primary auditory cortex initially processes the auditory signals while at the same time the primary visual cortex processes the visual signals. Pieces of information separately obtained by each low-level structure are sent to the areas devoted to phonetic and visual coding respectively. The outcome of the two intermediate modules are passed to the semantic association area, where they are merged.

In our architecture, an analogous example may be recognized in the modules devoted to assess whether environmental conditions are acceptable for a pleasant working activity, as shown in the right side of Figure 2.2. Low-level modules independently reason about air quality and room quietness, and the produced information is aggregated by a higher-level module; thanks to a broader knowledge of the environment, it may perform more complex reasoning, without being overwhelmed by the incoming information thanks to the previous filtering.

2.3 Architecture Overview

The architecture is logically designed according to a 3-tier model, as depicted in Figure 2.3: the *physical* layer is composed by all the sensory and actuation devices, including those necessary to implement the basic AmI functionalities, and possibly those required by the end user's application; the physical abstraction interface, in particular, will take care of exporting higher-level abstractions identifying the basic monitored units (e.g. each office room) besides dealing with basic connectivity issues among gateways, and will group together all the functionalities related to message relaying, monitoring and control of the physical infrastructure health, and reconfiguration due to changes in the underlying physical infrastructure. On top of it, the *middleware* defines a toolset of basic AmI functionalities in the form of building blocks for implementing intelligent services over the available hardware; finally, the actual AmI applications created by the final developer will be hosted at the *application* layer. The shadowed part of Figure 2.3 shows the logical boundaries of system-dependent own components, as opposed to the user-provided ones.

In the scenario of workplace monitoring, the basic monitored unit would be a single office room; however, the system architecture has been designed in order to be scalable with respect both to the number of monitored premises and to the potentially diverse employed technologies. According to this perspective, the logical architecture described has been designed as a partly centralized and partly

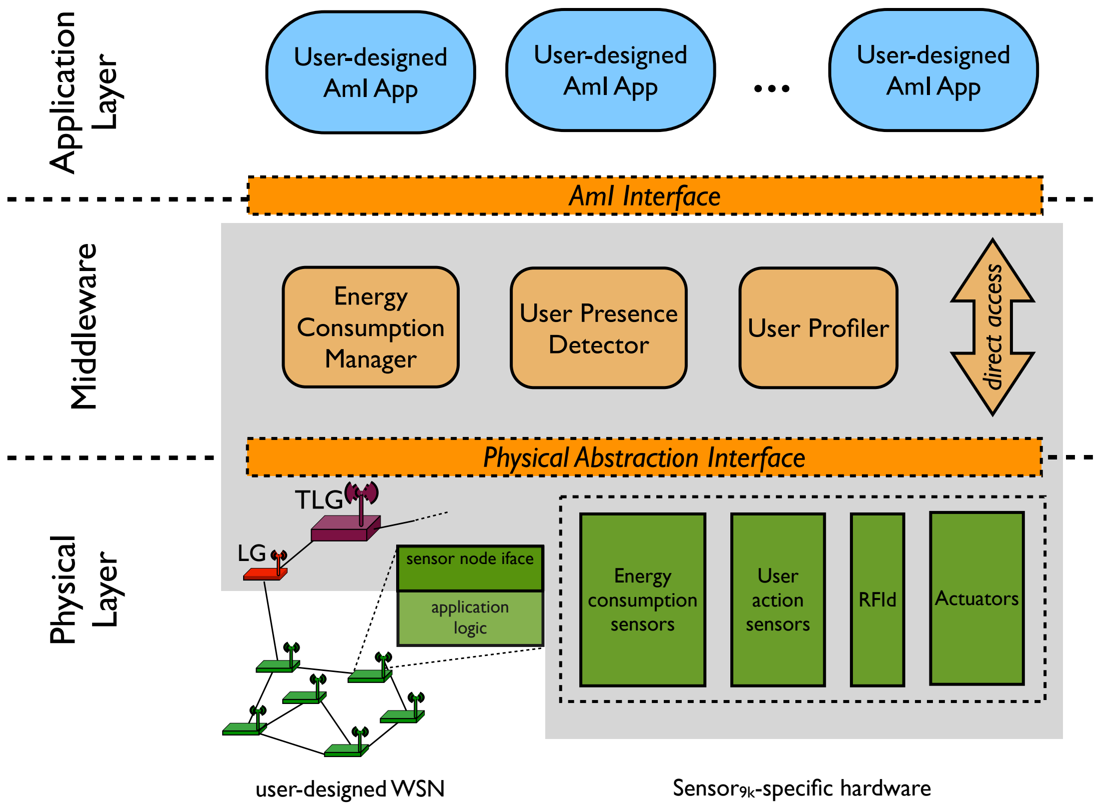


Figure 2.3: The logical 3-tier architecture.

distributed system; more specifically, high-level functionalities are implemented in a central AmI server, whereas the functionalities for managing low-level data gathering and command injection are pushed forward into the distributed nodes pervading the environment, as shown by the deployment diagram of Figure 2.4. In particular, the entire middleware is distributed over several components: part of it, namely the AmI modules and their interface with the applications, lies in the central AmI server, whereas most of the underlying services are provided by the remote gateways, and finally a tiny middleware layer is superimposed to the remote sensor nodes in order to have them respond to system's commands.

Each of the remote networks deployed into a specific room includes both wireless and wired sensor nodes and actuators. The system provides access to the room via a dedicated gateway node that implements the bridge between the physical devices, and the system itself. Such a component, named Local Gateway (LG) in the rightmost side of Figure 2.4, simplifies the connection among different network technologies and provides the higher layers with a homogenous representation of

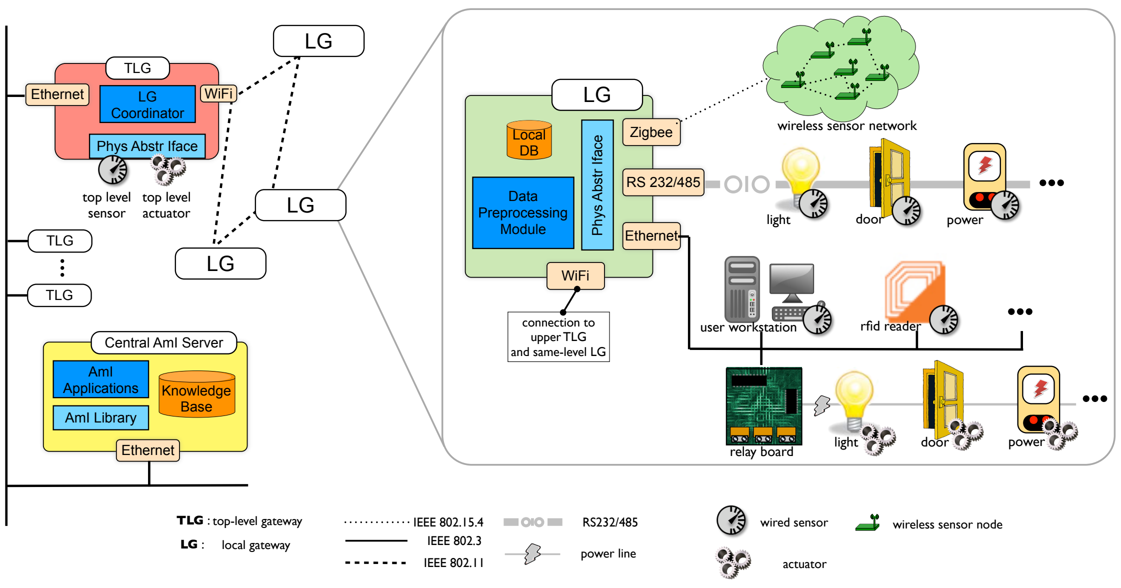


Figure 2.4: The hierarchical gateways system. The Central AmI Server represents the point of access for AmI applications; Top Level Gateways (TLGs) are connected to it, each managing a different environment, while fine-grained monitoring is guaranteed by Local Gateways (LGs).

data originated by the heterogenous sensory technologies.

The remote LGs may be installed on lower-performance computing devices, such as microserver nodes connected to the wireless sensor and actuator network. Those devices are more powerful than low-end wireless nodes, and may provide temporary storage trough their local DBs and an increasingly refined preliminary processing for data, before forwarding them to the remote processing units.

LGs are connected to each other in order to create a communication backbone in the controlled premises; this LG Network is coordinated by a Top-Level Gateway (TLG) which plays the role of collector of the information coming from all of the LGs. The TLG also supplies the programming interfaces towards some devices, related to general sensing and actuating functions, not specific for a single premises. For example, the TLG provides the interface for the sensors and actuators used to perform the access control for a unique floor. If the system architecture needs to be implemented for an entire building, individual rooms will be managed by a dedicated LG; one TLG will take care of coordinating them and of dealing with floor-wide functionalities; finally, the TLGs for different floors will all report to the Central AmI Server which will possess a comprehensive view over the physical system.

AmI applications access the functionalities of the physical layer only through the Central AmI Server; the intelligent software components of the system are not directly coupled to the hardware, thus making the development of ad-hoc applications both simpler and more generalizable. Figure 2.3 also shows the specific middleware modules provided by the system for implementing basic functionalities, such as monitoring the energy consumption of remote appliances and actuators, analyzing sensory data to infer the user's presence in a given area, and finally analyzing the users' interaction with the actuators in order to build a profile of their preferences. The outcome of such modules will provide the common ground over which general higher-level AmI applications may be developed.

2.4 The Physical Layer

The main sensory infrastructure of the proposed system is represented by WSNs; moreover, our system has been designed in order to be easily customized with additional sensors, thanks to the adoption of a standard abstraction layer. Such layer has been designed to comply with a reduced version of the specifications provided by the OpenGIS sensor model language (Botts and Robin, 2006), which indicates models and arrangement rules for device interfaces in order to obtain the maximum degree of interoperability among different technologies. We loosely adhere to their specifications for data types and for describing sensor characteristics, and adapt them to our case by including actuators. In particular, the proposed system provides a library of low-level common functionalities, analogous in some aspects to the SWE Commons of (Botts and Robin, 2006), in that it supplies basic software tools for data management, with additional functionalities for actively modifying the data gathering process in terms of sensing rate or precision. We also defined a protocol for the communications between the LG and the sensor/actuator nodes, as well as from the LG upward to the higher levels of the system hierarchy. The protocol allows for the injection of commands into the network and for sensory data retrieval. Typical commands include switching nodes on and off, triggering the sensing, varying the sensing rate, and even more radically changing nodes' behavior, e.g by having them aggregate data before transmitting them. Analogous commands exist for the actuators, ranging from simpler ones for on/off actuators, to more elaborate ones for the air conditioning, or light dimming systems. The interfaces of the LG towards the various connected elements rely on physically diverse media which may be wired or wireless, and the presence of the mentioned protocol allows the data management and communication software operation on board of the LG to be unaware of data formats and communication modalities specific to each of those technologies. Figure 2.5 shows a partial view of the DB schema contained in a LG, and showing the internal representation of the sensory

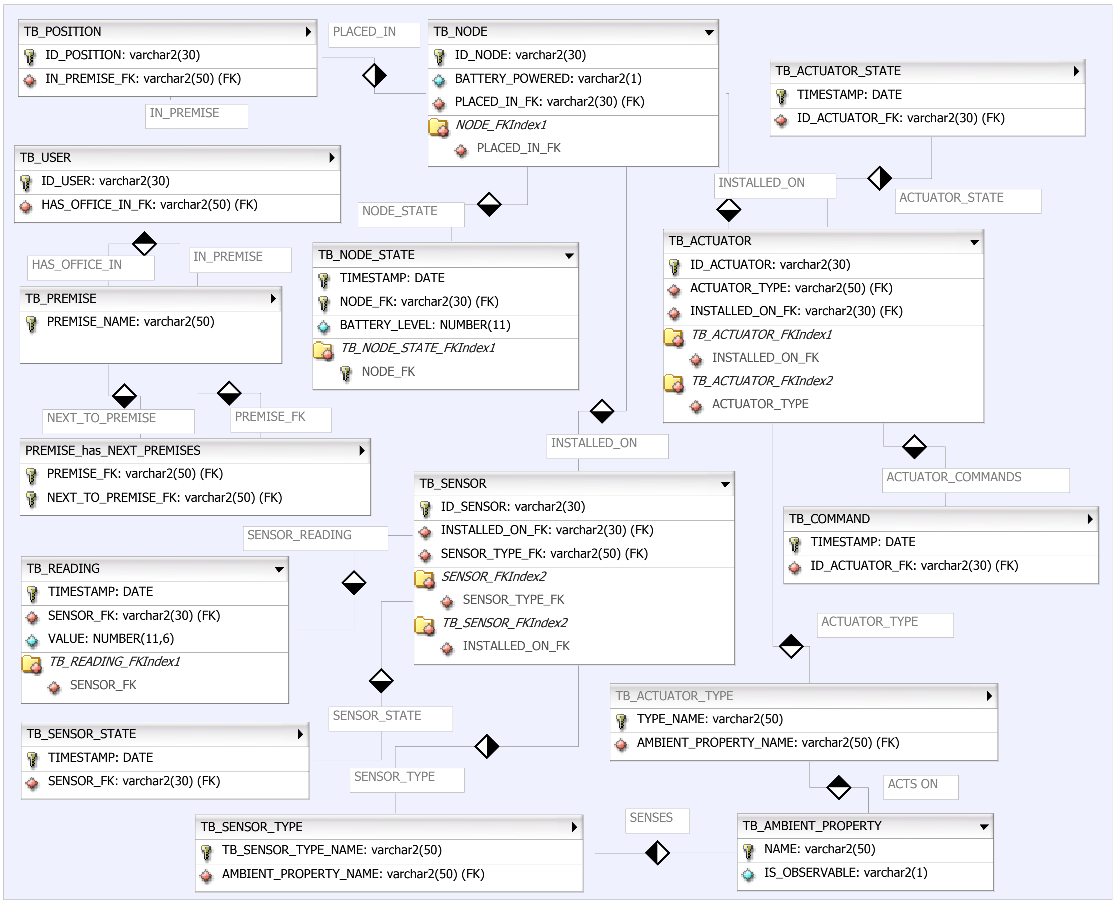


Figure 2.5: Part of the DB-schema showing the internal representation of the sensory and actuator devices for an indoor environment.

and actuator devices relative to office rooms, according to the physical abstraction interfaces exported by each node.

We assume that the choice of the sensors and of the sensor nodes platform for environmental monitoring is determined by developer of the AmI application under testing; however, our system shall use supplementary sensors for specific testing purposes, so as to provide AmI applications with additional information in order to allow them to implement proper policies of energy saving. For this reason, the system provides the tools for monitoring the electric energy usage of the building, for inferring the users' presence, and monitoring the interaction between users and actuators. Energy consumption may be monitored with varying resolution, and it is possible to observe the energy usage of entire premises, or specific devices. To monitor the global energy consumption of a specific room, we installed a multifunctional power analyzer; it was connected to the monophase line powering the room under observation and allowed us to collect information about voltage, current, and active and reactive power. Individual monitoring of specific devices is carried on by specialized "energy sensor nodes" that measure energy usage of any device connected to the power outlet of the sensor.

Besides mere measurement of instantaneous consumption, effective implementation of high-level policies for energy saving would also benefit from additional information. For instance, estimating the presence of specific users in the controlled areas might be interpreted as a trigger for the system to operate on the actuators only when actually needed. Our approach is to avoid deploying additional dedicated sensors, and rather exploit pre-existing ones; however, we do not assume that any of them, separately considered, is able to provide a sufficiently precise estimate of the user's presence, so we choose to merge multiple sensory information, coming from diverse sensors, through a data fusion process.

Monitoring users' access to the premises managed by the AmI system may provide rough indications on the presence of users in specific areas, and a suitable technology for implementing simple access control is represented by RFIds (Roussos and Kostakos, 2009). In the proposed system, the RFId readers have been coupled with sensor nodes installed close to the main entrance and to each office door, while RFId tags have been embedded into ID badges for the department personnel, so as to completely replace traditional keys.

Also the sensory infrastructure deployed for the AmI application purposes may be exploited beyond its "natural" purpose; in particular, the WSN may cooperate to the estimate of the user's presence in a given room; namely we extend the system with additional sensor nodes carried by the users, and we use their interaction with the deployed WSN infrastructure to provide naive localization. Portable nodes are not equipped with any specific sensor, as they are only used to communicate with fixed nodes and to estimate their relative distances to them. The basic idea consists in estimating the distance between a mobile node, carried by the user, and the other environmental nodes placed at know locations, that may act as "beacons"; a simple trilateration algorithm would theoretically be sufficient to provide an approximation of the position of the mobile node. This idea is in fact not new in the field of WSN research (Savvides et al., 2003; Langendoen and Reijers, 2003), and one of the easiest way to provide a rough estimate of distance is through RSSI measurements; however early research has already pointed out that this kind of signal is very noisy and that the relative measurement is highly error-prone especially because of the unpredictable signal attenuation model in unknown environments. Several works have thus been presented aimed at refining the distance estimate, either by using different, more reliable measures altogether, or via more advanced postprocessing (Wang et al., 2010; Ahn and Ko, 2009). Our approach is fundamentally different in that we do not intend to provide precise

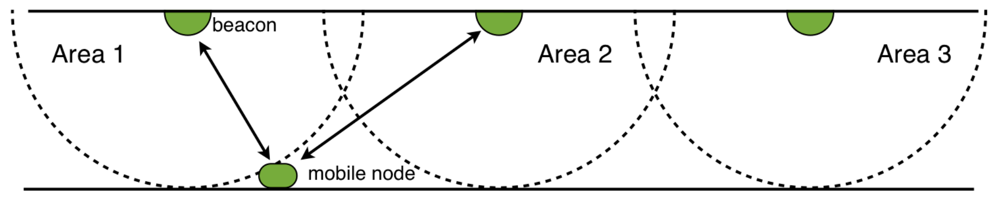


Figure 2.6: Example of coarse-grained localization: the mobile node is assigned to “Area1” by estimating the closest beacon.

localization of the mobile nodes, but we limit our system to determine the closest beacon, or in other words the macro-area through which the mobile node is currently moving. Whenever the system needs to find out the position of one of the mobile nodes, it injects a query into the part of the WSN that forms the localization subsystem; beacon nodes collect the RSSI signals and send those measurements back to the querying unit that may easily identify which beacon is closer to the mobile node; the interaction is depicted in Figure 2.6. This may be regarded as a coarse-grained localization, and its precision of course depends on the number and density of beacon nodes; in our context, though, we do not need to refine the estimate beyond a certain threshold as this piece of information will be used in conjunction to other ones, and will only be an additional uncertain input for the upper-level decision system.

In order to observe the interactions of the users with the actuators, and thus to provide the system with some indication about the users’ preferences, the physical tier of the proposed system includes a set of ad-hoc sensors. The simpler actuators provided by our system are remotely-controllable power relays, with the additional capability of providing information about their current state (e.g. the artificial lighting relay controller). More complex actuators are the domestic appliances controllable via IR remotes that may be typically found in homes or offices; within this category, we can mention air conditioning units, or automatic motorized electric blinds and curtains. Our hardware infrastructure thus includes the possibility of capturing the interactions between the users and such remotes, with the aim of providing the system with some indication about the users’ preferences. An ad-hoc sensor node was designed to this end, and equipped with an IR receiver through a suitable expansion board. We positioned one such node close to each appliance under observation, with the sensor next to its IR receiver. Each setting sent via the remote to the IR receiver is then captured by the sensor node, and sent to the LG along with other sensed data. No modification is thus required to existing devices, which makes the system highly adaptable to different scenarios, and suitable for capturing user-defined configurations of generic IR-based devices.

In order to develop the sensor node software for decoding the received IR information, a preliminary decoding of the pulse sequence sent by the IR remote needs to be performed. In our case, this has been accomplished by means of the utility library offered by the LIRC (Linux Infrared Remote Control) software package (Bartelmus et al., 2002), which allows to directly measure the duration of pulses got out from the IR receiver. The validation of these measurements has been achieved by comparing them with those obtained through a digital oscilloscope connected with the same IR received.

The hardware infrastructure is also meant to reproduce any possible user action, so it provides an IR remote control connected to the LG of each room. This device may be programmed to allow the system to remotely control most of the commonly available domestic appliances, as it just reproduces the same interface as traditional remote controls. The user action sensors, coupled with the USB remote control, contribute to the creation of an unintrusive and flexible system, easily deployable in most kinds of environments. The proposed solution also aims to keep the realization costs low since it is does not require to modify the devices to be controlled, or to buy specific versions of them; also it already solves the issue of collecting the relative measurements, that are just regarded as additional sensory inputs, and as such managed by the WSN infrastructure.

2.4.1 Related Work

Since their introduction, Wireless Sensor Networks have steadily evolved, especially with respect to the degree of complexity of the network configuration, as summarized in (Chatzigiannakis et al., 2008). The point of view has shifted from the use of one single WSN for the entire field, possibly composed by a very large number of nodes, towards a more structured approach involving several interconnected WSNs, each with a limited number of nodes, and up to a comprehensive strategy where the sensor nodes are enabled to interact with diverse devices and applications. Such progress has consequently widened and diversified the range of issues that the different middleware systems for WSNs, presented in literature, attempted to address.

While initial efforts were mainly focused on the optimization of the resources available to the nodes, in terms of energy or computational power, later research has also addressed the functionalities for enabling interoperability among heterogeneous devices and for providing a common interface to different applications. A survey of middleware tools for WSNs must thus consider such variety in the approaches, despite the fact that they may not always be directly comparable.

A traditional categorization of WSN middleware softwares (Molla and Ahamed, 2006; Hadim and Mohamed, 2006), mostly focused on the first type of proposals of single WSN deployments, specifically distinguishes them with respect to the adopted programming model; the common goal is always the provision of an intermediate layer decoupling the node application logic from the underlying operating system and hardware. The authors of Impala (Liu and Martonosi, 2003), for instance, adopts a modular design paradigm in order to improve the applications adaptability, and to provide a simple way to keep them up-to-date; in the authors' vision, the possibility of adapting the application interface on the fly is bound to improve the performance, energy-efficiency, and reliability of the overall system. The modular approach has also led to the adoption of an agent-based programming model, as for instance in (Fok et al., 2009); the latter work, in particular, employs mobile agents traversing several nodes, and carrying snippets of code with them; in this view, nodes are able, for instance, to host multiple applications at the same time. Another popular approach involves the use of virtual machines in order to provide the user with programming primitives in an assembly-like language, thus allowing developers to dynamically upload new "scripts" onto the network nodes; systems implementing this approach include Maté (Levis and Culler, 2002), and Magnet (Barr et al., 2002). Alternatively, the entire WSN has also been viewed as a single distributed database, so that the goal becomes to transparently provide access to sensed data according to the traditional relational model; one of the most relevant works in this context is TinyDB (Madden et al., 2005).

A limitation of such classification is that it just focuses on middleware running entirely *inside* the network, thus disregarding middleware that instead runs, partly or entirely, *on top* of the sensor network level. The specific context of Ambient Intelligence, on the other hand, has often stimulated researchers to exploit WSNs as a distributed sensory tool, and as a communication infrastructure, whereas the core of the intelligent services typically resides elsewhere, at a higher abstraction level. Our standpoint thus comprehensively considers that most of the AmI system is superimposed over the sensory equipment, and not entirely merged therein; the connection with the sensor nodes is provided by a minimal abstraction layer acting as a wrapper over the (user-defined) application logic, with the aim to hide possible future upgrades to the underlying hardware, as well as to provide a common interface towards the centralized Ambient Intelligence services.

A profitable comparison with other related approaches must thus analyze the functional aspects, besides the architectural paradigm; more specifically, a meaningful distinction for the classification of middleware in this perspective needs to consider the respective roles of the application logic at the sensor node level, and the higher-level application logic, that characterizes the behavior of the whole system, apart from the sensor network itself. While most middleware implementations discussed in literature are in fact an integral part of the sensor node software, the authors of (Chatzigiannakis et al., 2008) highlight the importance of both aspects, as functionalities implemented at the sensor node level may be used to provide novel services, whereas higher-level abstractions are useful to generalize the nodes' behavior so that it matches the upper layer requirements.

GSN (Aberer et al., 2006), for instance, use some abstractions to communicate with the software running on the sensor network gateways, and thus they do not depend on the use of some custom software of their own, although of course software drivers specific for each application have to be implemented.

Other works more explicitly separate intelligence from sensing; However, to our best knowledge, none of them fully exploits the potential computational capabilities of the sensor nodes; rather they are typically used as a mere data collection tool, with distributed sensors and communication capabilities.

In (Lee et al., 2006; Jimenez-Fernandez et al., 2006), systems for healthcare are proposed, especially targeted to monitoring chronic illness, of for assistance to the elderly. Such works employ WSNs as the support infrastructure for biometrical data collection toward a central server; sensor nodes are thus required to simply route data packets through multiple hops without operating any distributed processing on them.

In the work by Han, *et al.* (Han et al., 2007a), WSNs are used to provide inputs to an ambient robot system. Inside what the authors define a ubiquitous robotic space, a semantic representation is given to the information extracted from a WSN, but again this is used only as a sense-and-forward tool.

In (Stipanicev et al., 2007), a WSN-based infrastructure is described targeting the development of wildfire prevention system, whose architecture is based on three layers, the lowest of which relies on a sensor network for measurement gathering. Also the work presented in (Akhlaghinia et al., 2008) employs a WSN, but the goal is the collection of information about the occupancy of the monitored premises; collected data are aggregated in order to compute predictions about the occupant behaviour.

2.5 Knowledge Representation

The core of the entire architecture is represented by a centralized intelligent system that collects pre-processed data coming from the pervasive sensory system, carries on some reasoning in order to build an internal representation of the surrounding environment, and finally plans the proper actions taking into account both the internal representation and the goals derived from the users' requirements.

2.5.1 Multi-tier Knowledge Representation

The centralized part of the system is organized according to a layered architecture that allows to carry on specific reasoning on the environment at different levels of abstraction, and on different kinds of perceptions. From the designer's point

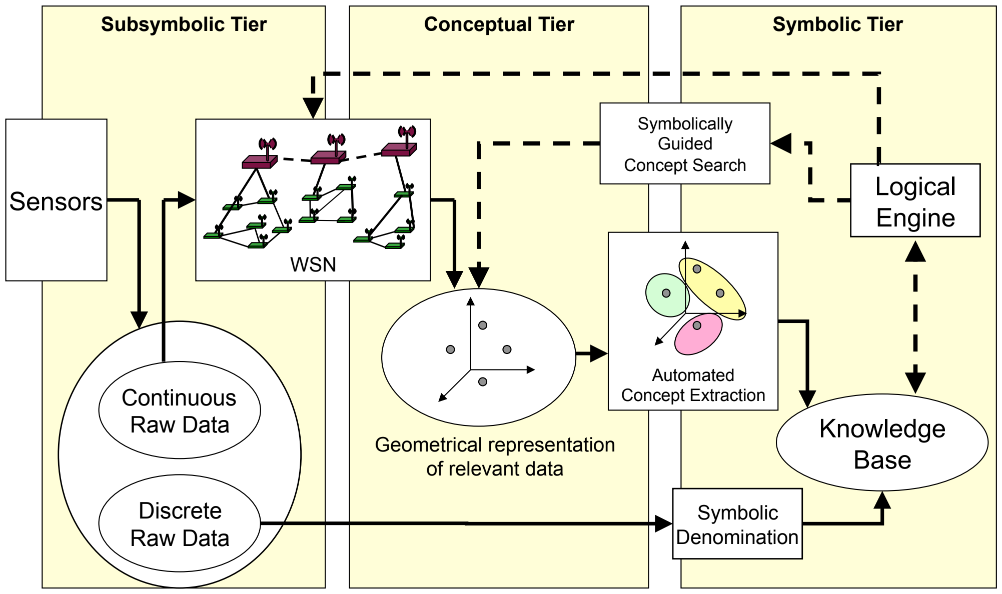


Figure 2.7: The three-tier structure of a low level module.

of view, the layered organization allows for the realization of a scalable sofware architecture, able to effectively manage the huge amount of sensory data.

The proposed system is based on a multi-tier paradigm for performing knowledge extraction starting from sensory data. As shown in Figure 2.7, this paradigm provides three tiers of knowledge representation, corresponding to different abstraction degrees. Starting from the rightmost block in the figure, knowledge is represented at *linguistic* level, where information is described symbolically via a high-level language, whose input is provided by a *conceptual* level where grounding of symbols occurs, and used to connect the system to the lower, *subsymbolic* tier, where sensory data is first acquired. This structure resembles the ideas presented in (Chella et al., 1997, 2000) that were applied to an artificial vision scenario; our system enhances this knowledge representation paradigm with the introduction of WSNs as the lowest-level pervasive data acquisition means, and by reproducing the same 3-tier schema so that the abstract information extracted by the low-level modules of the architecture may be used as input for higher-level modules, thus producing more and more abstracted vision of the world surrounding the system itself.

The subsymbolic tier processes the measurements collected by the pervasive sensory subsystem. As already mentioned, the purpose of the WSN-based infrastructure is not limited to the basic gathering of sensed data, but comprises also a preliminary processing aimed at the selection of the relevant information. Sensed measurements can be classified into two main categories, namely continuous or discrete; data belonging to the former class are fed to the intermediate conceptual tier, where they will be provided with a representation in terms of continuous quality dimensions. On the other hand, discrete data are outright handed over to the symbolic tier, where a linguistic representation will be given.

The conceptual tier is based on the idea of *conceptual spaces* introduced by Gärdenfors in (Gärdenfors, 2000); data are endowed with a geometrical representation that allows for a straightforward management of the notion of concept similarity, as long as a proper metric is chosen for the quality dimensions. Points populating the conceptual space, originally generated by the underlying measurement space, are represented as vectors, whose components are the quality measurements of interest. Concepts thus naturally arise from the geometric space as regions, identifiable through an automated classification process, that in our implementation occurs after a supervised training of the classifier.

Finally, the symbolic tier produces a concise description of the environment by means of a high-level logical language. At this level, regions individuated inside the conceptual space are associated to a linguistic construct, thus identifying basic concepts, while relations necessary to infer more complex concepts are described through an opportune ontology. The gap between a concept and its linguistic description is filled through two separate mechanisms inspired to the work of (Chella et al., 2000): an "automated concept extractor" deals with the translation of the regions in the conceptual space into symbolic elements, whereas a "symbolically guided concept search" identifies further points in the conceptual space as a consequence of the activation of some of the logical rules contained at the symbolic tier.

The created knowledge base is used to iterate the same knowledge extraction mechanisms at a higher abstraction level. In the considered case study, the concepts asserted at the symbolic tier are also employed for the activation of the control rules of the actuators, represented by the controllers of the heat, air conditioning, and lighting systems. Moreover, a subset of those rules is devoted to providing feedback to the WSN in order to guide its self-maintenance activity; for instance, under steady environmental conditions, the higher tier will opt for a reduction of the sensor sampling rate in order to reduce the overall energy consumption.

2.5.2 Recursive Multi-tier Schema

Form the knowledge management point of view, four main components may be identified in our architecture: a *sensory* component, implemented as a WSN in order to allow for precise and continuous environmental monitoring; a component for *understanding*, representing part of the system's "brain" and implemented by reproducing our multi-tier knowledge extraction architecture over multiple levels; a *planning* component, that completes our artificial "brain" and uses the extracted

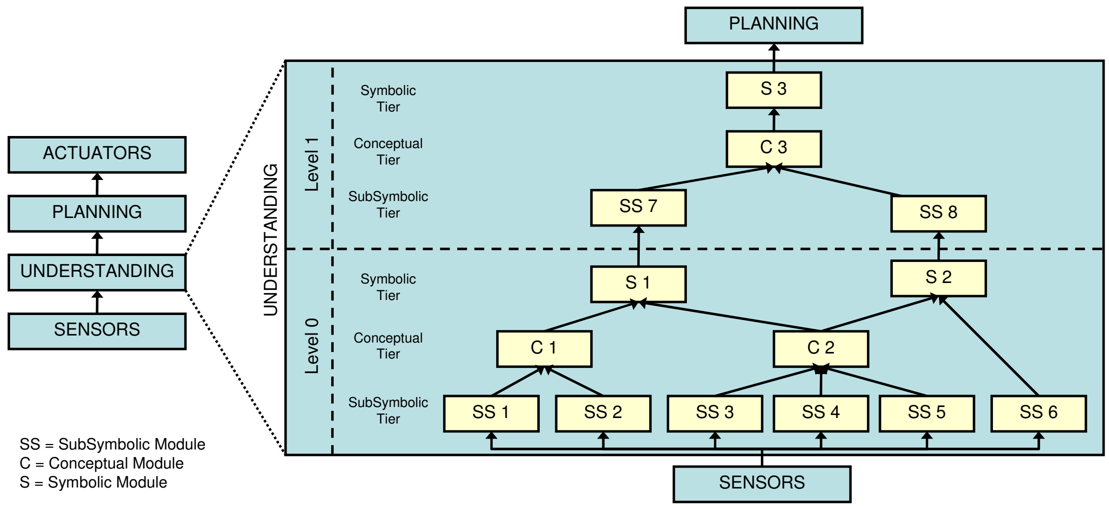


Figure 2.8: Layered architecture of the understanding component.

knowledge to plan the necessary actions to steer the environmental conditions towards a desired state; finally, an *actuation* component translating the high-level inference of the intelligent system into actions that modify the physical environment.

The *understanding* component processes the collected data so as to obtain a higher-level representation of the environment at different abstraction degrees used to merge different types of perceptual stimuli, in a way functionally resembling the organization of the human brain.

Several studies in neurosciences (Tononi and Edelman, 1998) proved that different brain areas are functionally specialized for well-defined tasks for sensory signal processing. Besides functional specialization, also functional integration is performed in the different areas, and at different spatial and temporal scale. This suggests the design of a hierarchical and modular architecture, whose parts operate independently and in parallel on different environmental stimuli in order to provide a symbolic representation of them.

Figure 2.8 shows how the *understanding* component is in fact split into multiple levels, each implemented according to a 3-tier structure of interconnected modules for representing knowledge, as described in Section 2.5.1. Each module in a level belongs to one of the tiers, and the connections specify how the information must flow from bottom to top, i.e. from the subsymbolic toward the symbolic tier, through the conceptual one.

Knowledge extracted from the symbolic tier of a given level provides the input for upper levels, where it is regarded as a kind of “higher-level sensory data”; information is actually originated at the sensor network only for the lowest-level modules, but this approach allows us to reproduce the same multi-tier knowledge management scheme at different levels, or in other words *the knowledge base created by a given level is used to iterate the same knowledge extraction mechanisms at higher abstraction levels.*

Besides constructing a faithful representation of the current state of the environment, our system is also able to plan a sequence of actions that will modify this state in order to bring it as close as possible to the users' desires, taking into account both the internal representation and the goals derived from the users' requirements. The *planning* component of the system needs to reconcile possibly opposed goals, through an accurate constrained planning system. In the case study we are proposing here, such goals are for instance related to maintaining pleasantness in room ventilation, or an adequate lighting, while at the same time minimizing the overall energy consumption, or guaranteeing that other user-defined constraints are satisfied, and that the WSN lifetime is maximized. The planning component exploits particular sensory information to adapt its internal representation of the user's requirements, such as information about the interaction of the user with the system actuators, through which an indirect indication of the user's preferences may be inferred. For instance, by switching on the light, the user is implicitly informing the system that the current lighting degree is inadequate; if this new piece of information is not consistent with the previous representation of the user's preferences, then it may be used to adapt the system's planning goals.

The output of the planning system consists of a sequence of actions to be executed in order to reach the goals, while fulfilling the constraints. A toy example may be the case when the internal representation of the environment produced by *understanding* component indicates an insufficient ambient lighting for the user's desires, and an external light index superior to the internal one. Knowing that the actions of "opening the curtains" and "switching on the light" may both restore an adequate ambient lighting for the user, and that the energy consumption derived from the former action is lower than for the latter, the *planning* component outcome will be the action of "opening the curtains", which will trigger the actuators for the relative automatic engine.

2.6 Middleware

This Section presents the middleware providing the core functionalities for the design of specialized modules on top of the hardware substrate.

These modules were developed according to the multi-tier knowledge representation schema and belong to the *Level 0*; some of them perform only a low-level data processing, and so they belong to the subsymbolic tier, and others perform a kind of high level processing, and consequently belong to the symbolic tier. These modules can be composed in order to support a specific AmI application.

From a logical standpoint, the main purpose of the middleware is to decouple the applications from a specific choice for the underlying hardware, so that the developer may focus on the issues concerning AmI aspects specifically. To this end, our system provides the modules implementing basic AmI functionalities that may be combined to form complete applications; moreover we also allow the possibility to directly access the lower level functionalities of the sensory and actuation equipment, although we still interpose an abstraction layer hiding the irrelevant details and providing a homogeneous view on the hardware.

In the following we provide some insight into the upper part of the middleware, by describing some of the specialized modules, currently available in our system, for monitoring energy consumption, interacting with the actuators, detecting the presence of users, and profiling their preferences.

2.6.1 Energy Consumption Management

In order to monitor energy consumption, the physical layer of the proposed system has been equipped with specific sensors, as described in Section 2.4. However, application developers might not be interested in detailed data about individual energy consumption of each device, but they might rather prefer to obtain only higher-level information. To comply with this need, our middleware includes a specialized module for governing energy consumption monitoring, continuous sensing, and for triggering notifications to the application layer only if some predefined thresholds are exceeded. AmI applications will be able to tune the behavior of such software component by acting on its parameters, by means of specific control messages. Besides implementing such basic functionalities, the middleware also provides support to planning in the context of energy saving by computing predictive models for energy consumption trends for selected devices. We assume that each monitored device is connected to one of the wireless nodes that gather data about energy consumption. The model will be synthetically represented by a table whose entries are pairs of the form $[actuator\_state, consumption]$; the energy consumption associated with each state of the actuator is estimated incrementally via an exponential moving average:

$$c(s) \leftarrow \alpha \cdot \bar{c} + (1 - \alpha) \cdot c(s), \tag{2.1}$$

where $c(s)$ represents the estimate of the energy consumption of the actuator with respect to its current state s, $\bar{c}$ is the latest reading of energy consumption, and α is a coefficient in the range $[0, 1]$. The cost model may then be queried by AmI applications in order to get information preliminary to planning.

Furthermore, the energy consumption module also includes the definition of the constraints to be taken into account by AmI applications in the planning phase.

Although not currently available, it is arguable that in the near future energy providers will be able to supply information about the current contractual offer, scheduled shortages and low-fare hours. We are considering here the possibility of dealing with a limited set of contractual options, so that the module controlling our intelligent energy meter may connect with the energy provider, gather information about possible constraints on monthly average consumptions and, by keeping track of past consumptions, provide an optimal estimate that meets the provided constraints.

Finally, a visionary but realistic scenario could include distributed energy production through the so called “smart grids” (Chen et al., 2009), whose main goal is to be autonomous by producing energy locally, via the exploitation of renewable energy sources. Such technology allows to feed possible overproduction of energy back into the distributed network in order to satisfy the demand coming from other network areas. In this context, the smart building managed by the AmI system represents a node of the electrical distribution network, and it is crucial to tune its energy consumption with respect to the amount of locally available energy, and to its cost.

2.6.2 User Preferences Profiling

SENSOR9k may perceive the actions performed by users via specifically designed sensors. The information obtained may be interpreted as *implicit* user feedbacks in order to learn their requirements; lacking any other kind of explicit feedback, the system resorts to analyzing the actions carried on by users, in order to extract implicit knowledge; for instance, the system is able to detect that a temperature decrease was requested and may use this piece of information to infer that the current environmental conditions are not satisfactory for the user; hence, a plan may be formulated on how to modify the system goals in order to better fit the intelligent environment inhabitants’ demands. The availability of these implicit feedbacks, obtainable by means of these special non-intrusive sensors that monitor user’s actions, is one of the innovative aspects of the proposed system.

The middleware includes a module that collects the detected feedbacks and produces a model for the user’s behavior. User-environment interaction modeling has been extensively studied in the field of Human-Computer Interaction; one of the most relevant issues in formalizing a method for user profiling is the creation of a unique profile for each user valid for all AmI applications possibly running on the testbed (González et al., 2005). Although such choice may appear as the most elegant, it also raises some issues; firstly, creating a unique user profile for all applications involves processing a wealth of data, originated from many diverse interactions of the users with the actuators; moreover, as the user profile is communicated to the applications, they risk to be overwhelmed by possibly irrelevant data about events they are not interested about.

We therefore opted for the creation of a user profile generation model that were general but at the same time parameterizable by the applications depending on which aspects of user-environment interactions they need to focus on. The users' profiles are built by regarding them as room occupant, and considering their interaction with the actuators. In particular, the software module may learn the utility value perceived by the user for each $[state, action]$ pair, by observing the environmental conditions and the interactions of the user with the actuators. The utility value is computed via an on-line mechanism based on a phase of action evaluation typical of reinforcement learning, so that the user profile may be built incrementally. Information related to implicit feedbacks is clearly filtered taking into account data about the user's presence obtained by the middleware modules devoted to this task.

Basically, whenever a user interacts with an actuator and changes its state, the correspondence between the current environment state and the current setting for the actuator is bound to a negative utility value; the greater the gap between the current setting and the user-imposed value, the greater the absolute value for the utility. When the user is present in the monitored area and they do not interact with the actuators, the correspondence between the environment state and the actuator setting gets a positive utility value. In order to automate this process, time is divided into slots and each of them is regarded as a discrete event.

According to the reinforcement learning formalization, we refer to the actuator setting at time t as a_t, to the environment state at time t as s_t. The environment state resulting at time $t+1$, as a consequence of the actuator setting, is referred to as s_{t+1}, while the reward obtained according to the users' feedback is r_{t+1}. After the evaluation of users' feedback, the current estimate of the average of the current actuator setting a_t in the current environment state s_t, referred to as $Q(s_t, a_t)$, is updated according to the following equation:

$$Q(s_t, a_t) \leftarrow (1-\beta)Q(s_t, a_t) + \beta[r_{t+1} + \gamma \underset{a}{max}\{Q(s_{t+1}, a)\}]. \tag{2.2}$$

The new utility estimate is obtained by merging the previous one with the information about the obtained reward and the future one; namely $\underset{a}{max}\{Q(s_{t+1}, a)\}$ is the maximum obtainable reward in the new state. The β and γ parameters, both ranging in $[0, 1]$, control the learning mechanism, and represent the learning rate and the discount factor, respectively. The former determines the weight of new information with respect to past history, and the latter determines the influence of future rewards. By setting the γ parameter to 0, utility is estimated according to a simple exponential moving average, similarly to what is done in the energy consumption module.

The information that may be gathered by AmI applications from the user's preference profile may regard synthetically the optimal action given a specific environmental condition, or broadly the entire utility table learnt so far.

Chapter 3

Sample Application

This Chapter presents a simple AmI application aimed at illustrating the use of the various hardware and software components of our testbed; the main goal here is to provide a proof-of-concept of the complete sensing-reasoning-acting loop. The testbed has been specialized for the context of temperature control in a work environment, and the physical layer was built by augmenting a classic office room with a set of non intrusive devices. The premises of our department were chosen as a convenient experimental platform, and we aimed to control the temperature conditions not just for improving the users' comfort, but also for optimizing the overall energy consumption.

The sample application presented in this chapter is composed by two AmI modules: an understanding module devoted to detect the user's presence and a planning module whose aim is to control the heating and air conditioning system. The module for detecting the user's presence exploits information coming from the sensor network provided by the testbed, through a Bayesian network devoted to perform the multi-sensor data fusion, in order to infer the user's presence; the output produced by the module for the user's presence detection is provided as input to the module for controlling the heating and air conditioning system; moreover, this module also obtain information directly from the testbed. The heating and air condition controller is composed by two sub-modules, a fuzzy controller that, as a function of its input, produces a set of commands for the actuators, and an autonomic manager whose goal is to optimize the fuzzy controller rules on the basis of the implicit feedback provided by the user. Figure 3.1 shows a block diagram of this sample application.

The rest of this Chapter is organized as follow: Section 3.1 describes the sensory infrastructure of SENSOR9k specialized for the sample application, Section 3.2 describes the module for detecting the user's presence and Section 3.3 describes the module for controlling the heating and air conditioning system. Finally, Section 3.4 shows results of the experimental evaluation.

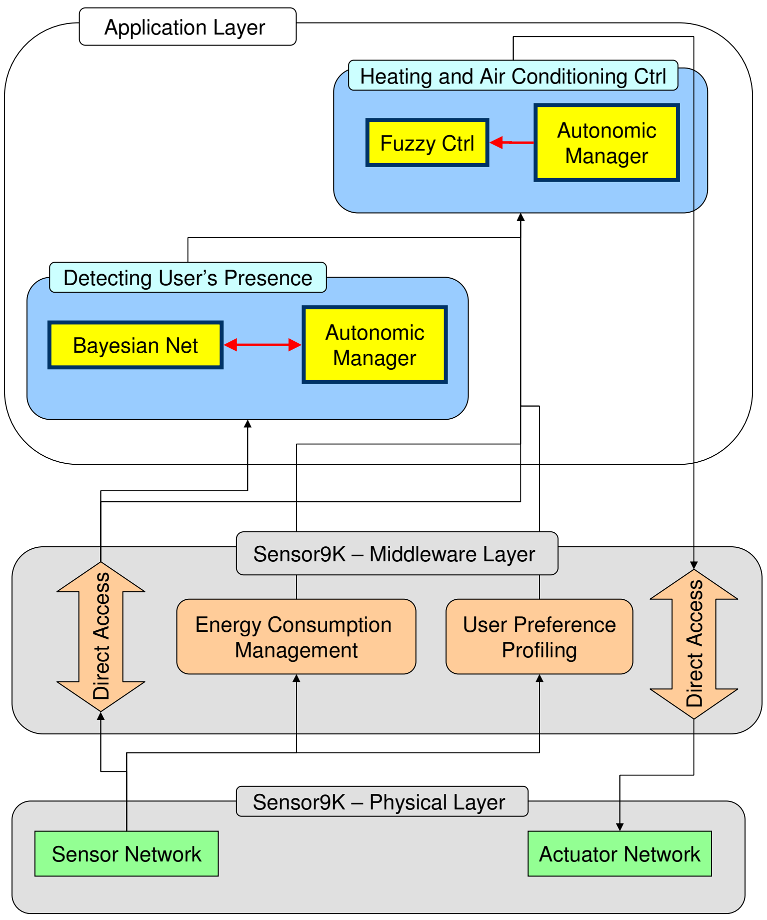

Application Layer
Heating and Air Conditioning Ctrl
Fuzzy Ctrl
Autonomic Manager
Detecting User's Presence
Bayesian Net
Autonomic Manager
Sensor9K – Middleware Layer
Direct Access
Energy Consumption Management
User Preference Profiling
Direct Access
Sensor9K – Physical Layer
Sensor Network
Actuator Network

Figure 3.1: Block Diagram of the Sample Application.

3.1 Exploiting the Testbed

3.1.1 The Sensory Infrastructure

The sensor nodes used for this sample application belong to the *mote* family (Memsic, 2010); they are particularly suitable for our purposes, also thanks to the possibility of extending their functionality by adding new types of sensors or actuators. TelosB Motes, in particular, are equipped with 10 kB RAM, 16 kB for configuration EEPROM, and 1024K bytes for data storage into Flash serial memory; the communications among them are based on the IEEE 802.15.4 protocol over a 250 kbps radio channel.

In order to assess the practical usability of our testbed, we compared its requirements in terms of memory occupancy with other widely used applications.

Table 3.1: Memory footprint comparison (data for Maté and Agilla are from (Costa et al., 2007)).

	App. code (ROM)	App. data (RAM)
Blink	2650 bytes	55 bytes
Maté	7.5 kB	600 bytes
Agilla	3.59 kB	41.6 kB
SENSOR9k middleware	4672 bytes	512 byte
Sample application	8.5 kB	3 kB

The two lowest rows of Table 3.1 show the separate memory footprints for the SENSOR9k middleware, and for the specific application considered here; for instance, our middleware requires roughly double memory for code, as compared to the basic Blink application for TinyOS. A more reliable comparison can be made with other popular middleware tools for WSNs, such as Maté (Levis and Culler, 2002), and Agilla (Fok et al., 2009), showing that the requirements for code and data are comparable.

Nodes have been placed at strategic points in the rooms (see Figure 3.2), in regions where sensed measurements may present oscillations and unexpected trends; specifically, we deployed nodes in different rooms, close to "sensitive" areas: by the door, by the window, and by the user's desk; additional nodes have been installed on the building facade, close to the office windows, for monitoring outdoor temperature, relative humidity, and light exposure. Besides the sensors provided by SENSOR9k, Table 3.2 shows the main sensors available to this sample application for environmental monitoring.

Based on the measurements of ambient temperature and relative humidity, the application computes the physiological equivalent temperature (PET) (Höppe, 1999), and uses this value to make a decision on the proper actuation policy. The equivalent temperature is computed through the index defined in (Bründl and Höppe, 1984), which adds the latent heat of condensation for the water vapor in the air to the actual temperature. The resulting empiric formula is the following:

$$PET = T + m_h \cdot (r - 2.326 \cdot T)/(c_p + m_h \cdot c_w). \tag{3.1}$$

For our purposes, we have considered a fixed value for the atmospheric pressure above sea level, and computed all other involved quantities as reported in Table 3.3, so that the value of the index only depends on the measured ambient temperature

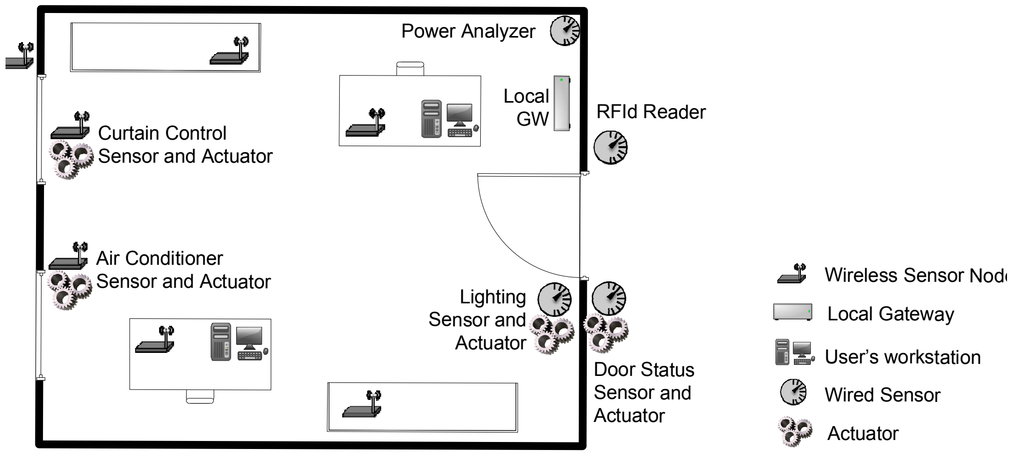

Power Analyzer
Local GW
RFId Reader
Curtain Control Sensor and Actuator
Air Conditioner Sensor and Actuator
Lighting Sensor and Actuator
Door Status Sensor and Actuator
Wireless Sensor Node
Local Gateway
User's workstation
Wired Sensor
Actuator

Figure 3.2: Location of wireless sensor nodes installed in user offices.

T (directly) and relative humidity (through the moisture content parameter m_h). The PET index, for atmospheric pressure in the 800–1100 mbar range, returns meaningful values when the measured ambient temperature is between 20°C and 45°C; since our test environment might also experience lower and higher temperature, we consider here the following aggregated index PT as representative of a subjective measure of human perceived temperature:

$$PT = \begin{cases} PET & if\ T > 20°C, \\ T & elsewhere. \end{cases} \tag{3.2}$$

The nodes of the WSN dedicated to environmental control are programmed to directly compute the PT value, by performing all computations and thresholding on board; in fact, the application layer may in all respects assume to deal with a virtual sensor, undistinguishable from other common sensors.

3.1.2 Middleware Modules

The sample application considered here shows how to profitably make use of the middleware modules described in Section 2.6. In particular, besides analyzing the perceived temperature, the application will get information about a profile of the user's preferences, computed thanks to the functionalities provided by the module described in Section 2.6.2; in this case, the module has been parameterized in order to collect the associations between the environment state and the users' actions, as represented by the following vectors:

$$\begin{aligned} state &= [PT, PT'], \\ action &= [\Delta T, Mode]. \end{aligned} \tag{3.3}$$

Table 3.2: The main sensors used for environmental monitoring, and their characteristics.

Measure	Sensor	Characteristics
Temperature and relative humidity	Sensirion SHT11	Temperature range: -40 °C to +123.8 °C Temp. accuracy: +/- 0.5 °C @ 25 °C Humidity range: 0 to 100% RH Absolute RH accuracy: +/- 3.5% RH Low power consumption (typically 30 μW)
Barometric pressure and temperature	Intersema MS5534	Pressure range: 300 to 110 mbar Pressure accuracy: +/- 3.5% Temperature range: -10°C to 60°C Temperature accuracy: +/- 2°C Operating range 3.6 to 2.2 volts
Outdoor Light	Hamamatsu S1087	Spectral response range λ: 320 – 730 nm Peak sensitivity wavelength λp 560 nm Photo sensitivity S (A/W) Infrared sensitivity ratio 10%
Ambient Light	Taos TSL2550	Range: 400 to 1000 nm Operating range 3.6 to 2.2 volts

This middleware module returns the user's profile expressed in terms of the utility function as perceived by the user for each $[state, action]$ pair. In our sample application we compute the reciprocal of this utility value, which may be interpreted as a metric representing the distance from the user's preference, and regard this as the first objective function to be minimized.

The sample application also uses the middleware module described in Section 2.6.1 for predicting the cost of each actuator setting; this cost value is the second objective function to be minimized.

It is worth mentioning that if the middleware modules for user profiling and cost prediction provided only static, off-line models, the resulting application policy would be characterized by overfitting, and the system would only correctly react to ambient conditions analogous to those experimented during the training phase. In our case this is avoided by allowing each module to incrementally update its model, based on on-line information.

3.2 Detecting User's Presence

The design approach for each understanding modules depends on what kind of environmental features is the subject of the reasoning. In some cases, where sensory information is not affected by noise, and the data fusion process can be easily

Table 3.3: Parameters for computing the *PET* index.

Quantity	**Meaning**	**Value**
m_h	moisture content	$0.620 \cdot \frac{p_v(hum)}{p_{atm} - p_v(hum)}$
p_{atm}	atmospheric pressure	$1013\, mbar$
$p_v(hum)$	vapor pressure	$610.78 \cdot e^{\frac{17.269 \cdot T}{T+237.30}} \cdot \frac{hum}{100}$
r	latent heat of vaporization	$585\, cal \cdot g^{-1}$
c_p	specific heat of air	$0.24\, cal\, °C\, g^{-1}$
c_w	specific heat of water	$1\, cal\, °C\, g^{-1}$

coded, it is possible to choose a rule-based approach. On the contrary, if the reasoning module has to cope with uncertainty, as is the case where the goal is to detect user's presence, it is desirable that the design rely on the Bayesian Network theory, which allows to infer knowledge through a probabilistic process, and offers an effective way to deal with unpredictable ambiguities from multiple sensors (Lu and Fu, 2009). This approach is unlike a rule-based approach, that is not suitable for dealing with environmental features characterized by a large uncertainty, as the set of logical rules constituting logical reasoning engine is exclusively deterministic; our domain, on the other hand, requires the integration of intrinsically noisy sensory information that, moreover, can only provide partial observations of the system state. Indeed, SENSOR9k offers various types of sensors capable of perceiving physical and environmental characteristics useful in order to detect users' presence. However, none of these sensors is sufficient, alone, for performing this kind of elaboration, because it is characterized from an excessive uncertainty or because it does not succeed in monitoring all the premises of interest.

Classical Bayesian networks (Pearl, 1988), however, may only provide a static model for the environment, which would not be suitable for the proposed scenario; we therefore chose dynamic Bayesian networks or, more specifically, Markov chains to implement our models which thus allow for probabilistic reasoning on dynamic scenarios, where the estimate of the current system state depends not only on the istantaneous observations, but also on past states. The validity of a Markovian approach for detecting users' presence by exploiting pervasive sensory information is confirmed by several works presented in AmI literature (Atallah and Yang, 2009).

Figure 3.3 (a) shows our proposed Markov chain used to infer probabilistic knowledge on a given state feature starting from a set of sensory data. Each state

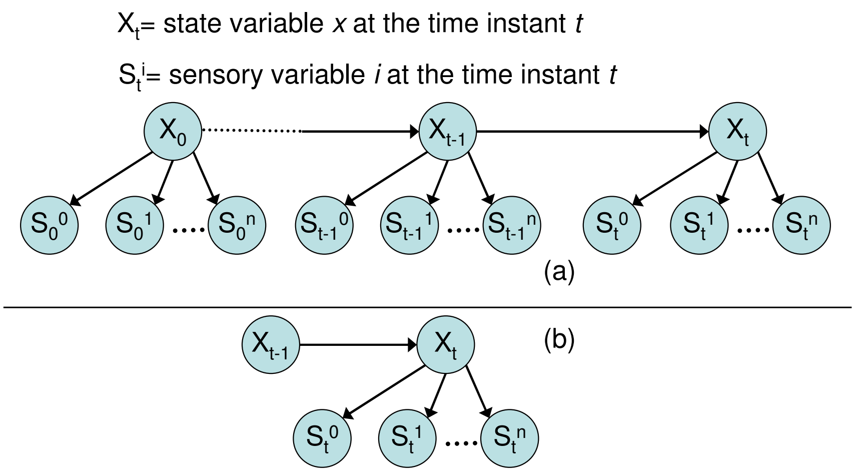

X_t= state variable x at the time instant t
S_t^i= sensory variable i at the time instant t
X_0
X_t-1
X_t
S_0^0
S_0^1
S_0^n
S_t-1^0
S_t-1^1
S_t-1^n
S_t^0
S_t^1
S_t^n
(a)
(b)

Figure 3.3: Structure of a Markov chain for inference a given state feature starting from a set of sensory data.

feature affects a set of sensory readings (we indicate each sensor node with s^i), that can be considered the perceivable manifestation of that state. The link among the current state and its sensory manifestation is given by the probabilist sensor model $P(s_t^i|x_t)$. Moreover the current state depends on past state according to a state transition probability $P(x_t|x_{t-1})$.

The belief about the value of a state variable is the conditional probability with respect to the whole set of observation from the initial time to the current time:

$$\begin{aligned} Bel(x_t) &= P(x_t|s_1^1, s_1^2, \ldots, s_1^n, \ldots s_t^1, s_t^2, \ldots, s_t^n) = \\ &= P(x_t|\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_t) = P(x_t|\mathbf{S}_{1:t}); \end{aligned} \tag{3.4}$$

Due to the simplifications introduced by the Markov assumption (eq. 3.6 $\rightarrow$ 3.7, 3.8 $\rightarrow$ 3.9), by the Bayes rules (eq. 3.5 $\rightarrow$ 3.6), by the probability laws (eq. 3.7 $\rightarrow$ 3.8) and by the conditional independence among sensory measurements given the state induced by the bayesian network (3.9 $\rightarrow$ 3.10), the belief about the current state can be inductive defined as follows:

$$\begin{aligned}
Bel(x_t) &= P(x_t|\mathbf{S}_{1:t}) && (3.5)\\
&= \eta P(\mathbf{S}_t|x_t, \mathbf{S}_{1:t-1}) \cdot P(x_t|\mathbf{S}_{1:t-1}) && (3.6)\\
&= \eta P(\mathbf{S}_t|x_t) \cdot P(x_t|\mathbf{S}_{1:t-1}) && (3.7)\\
&= \eta P(\mathbf{S}_t|x_t) \cdot \sum_{x_{t-1}} P(x_t|x_{t-1}, \mathbf{S}_{1:t-1}) P(x_{t-1}|\mathbf{S}_{1:t-1}) && (3.8)\\
&= \eta P(\mathbf{S}_t|x_t) \cdot \sum_{x_{t-1}} P(x_t|x_{t-1}) Bel(x_{t-1}) && (3.9)\\
&= \eta \prod_i P(s_t^i|x_t) \cdot \sum_{x_{t-1}} P(x_t|x_{t-1}) Bel(x_{t-1}). && (3.10)
\end{aligned}$$

Thanks to these simplifications, at each time step it is necessary to consider a reduced set of variables, as shown in Figure 3.3(b), which results in a reduced overall computation effort.

These principles had been followed in the design of the subsystem aimed at detecting the user's presence and therefore at reasoning on room occupancy. This subsystem only needs information directly obtainable from the sensory component, so it belongs to *Level 0* of our multi-level architecture. The outcome of this subsystem provides an estimate about the number of people present in the user's office room, and a probability for the user's presence as well; this information will form part of the input for subsystems at higher levels.

Since there are two interconnected state variables, that is two variables that are not probabilistically independent, the Bayesian network has been extended to manage two state variable, as shown in Figure 3.4. Sensory nodes are split into two sets, each of them is considered the measurable manifestation only of one hidden state variable. The two state variable are connected by dependency, that is the number of people in the user's office room (associated to the *PeopleInRoom* variable) is influenced also (there are some non measurable factor that influence this state variable) by the presence of the considered user in their own office room (*UserInRoom*)

The state is observable through sensory information associated to several virtual sensors. Variables modeling this sensory information are connected with state variables through sensor probabilistic models, expressed by conditional probability tables that were learned from an opportune training data set. In details the used sensory information are the following:

- *WSN-Localization*: the WSN-based subsystem for localization, as previously described;
- *Sound-Sensor*: is able to detect the average level of sound in the room;

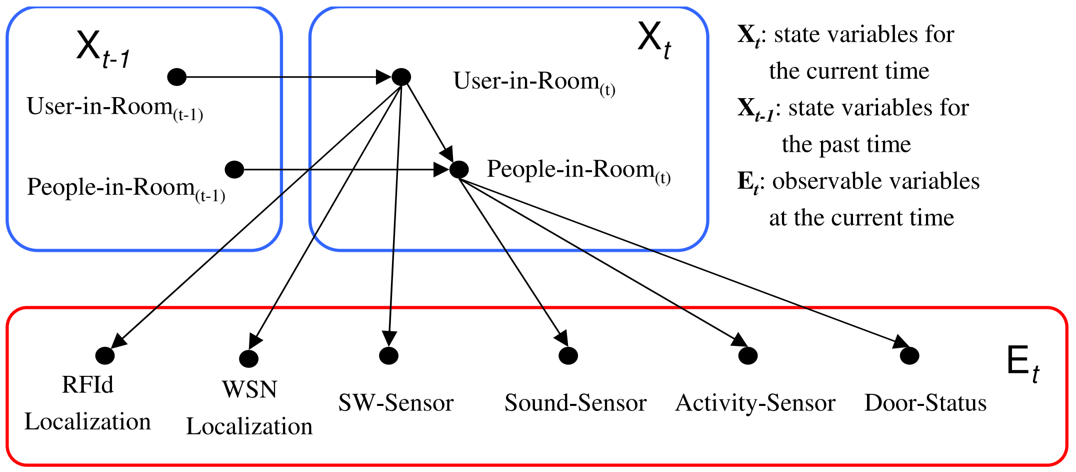


Figure 3.4: Markov chain for room occupancy evaluation. X_t and X_{t-1} are the sets of state variables at the current and at the past time respectively, while E_t is the set of observable variables at the current time.

- *Activity-Sensor*: is a virtual sensor installed in the LG with the job of detecting the interactions among the customer and the actuators and of merging these information in order to obtain the level of user's activity;
- *Door-Status*: a sensor for detecting the state of the office door (open / closed / locked);
- *RFId-Localization*: performing the RFId-based naïve user localization;
- *SW-Sensor*: a software sensor for detecting user's activity at his workstation.

These last two virtual sensors are a bit more complex than the others and so their require a deeper description.

The virtual sensor that performs the RFId-based naïve user localization exploits the information obtained through the few RFId sensors used for users' access control. Several works in literature ((Han et al., 2007b; Park and Hashimoto, 2009)) aim to perfect the localization process through the exploitation of a dense RFId grid. However in our work one of the most relevant goals is to maintain low costs and a low intrusiveness degree, so the proposed approach is to exploit several sensory information, each of which suffers from a non negligible uncertainty degree, but is obtainable via inexpensive processes.

Our virtual RFId-localization sensor employes Gaussian filters on the gathered data, according to the pseudocode reported in Figure 3.5. A badge reading gives timely information about the current presence of the user in the room where the

```
— Localization through RFId readings —
    Parameters: userID, targetRoom;
    Initialization:
 1: Graph ← graph(area topology);
 2: $\mu \leftarrow [x_{ID}, y_{ID}]^T$;
 3: $\Sigma \leftarrow \Sigma_{max}$;
 4: init $R$;
    Main Loop:
 5: loop
 6:     if new reading for userID in $[x, y]^T$ then
 7:         $\mu \leftarrow [x, y]^T$;
 8:         $\Sigma \leftarrow \Sigma_{min}$;
 9:     else $\Sigma \leftarrow \Sigma + R$;
10:     end if
11:     propagateBelief(Graph, $\mu$, $\Sigma$);    ▷ topology-aware belief propagation
12:     notify targetRoom.belief;    ▷ notify belief update to upper-layer modules
13: end loop
```

Figure 3.5: Pseudocode for the virtual sensor feeding the Bayesian network.

specific reader is placed. The underlying idea is that the *belief* about the presence of the user in a specific location may be represented by a normal probability distribution with mean μ, and covariance Σ, as expressed by the following equation:

$$p(x) = det\,(2\pi\Sigma)^{-\frac{1}{2}}\, exp\left\{-\frac{1}{2}(x-\mu)^T\,\Sigma^{-1}(x-\mu)\right\}. \tag{3.11}$$

Kalman filter (Kalman, 1960) is a suitable tool for managing linear Gaussian systems, because it represents the belief function through its moments; we modify the classical approach by specializing the measurement update step, and using a simplified transition state function. The RFId reading produces a precise information about the presence of the user in the same area where the badge is read, so the measurement update step of the Kalman filter may be simply formulated by centering a Gaussian in the reader location, and assigning a minimum variance to it (lines 7–8). Since we cannot predict the direction of the movements of the user, the control vector of the Kalman equation is assumed to be null, thus resulting in the following equations for the measurement update step:

$$\mu_t = \mu_{t-1}, \tag{3.12}$$

$$\Sigma_t = \Sigma_{t-1} + R; \tag{3.13}$$

where R represents the covariance of the Gaussian noise signal, with zero mean, assumed to affect the state transition (line 9). If no further readings occur for the same user, the uncertainty on the position estimate, represented by the Gaussian

— Topology-aware belief propagation —

```
    Parameters: $Graph$, $\mu$, $\Sigma$;
    Initialization:
 1: $updateList \leftarrow \emptyset$                      ▷ list of nodes whose belief is to be updated
 2: for all $n$ in $Graph$ do
 3:     $n.updated \leftarrow$ false;
 4: end for
 5: $root \leftarrow Graph.getNode(\mu)$;                  ▷ the location of the root node is the center of the belief distribution
 6: $root.belief \leftarrow gaussianValue(\mu, \Sigma, 0)$;  ▷ evaluate the Gaussian at its center
 7: $root.dist \leftarrow 0$;                              ▷ physical distance from root node along the graph
 8: $root.updated \leftarrow$ true;
 9: $neighborList \leftarrow neighbors(Graph, root)$;
10: for all $neighbor$ in $neighborList$ do
11:     $neighbor.parent \leftarrow root$;
12:     $updateList.append(neighbor)$;                     ▷ append the neighbor node to the update list
13: end for
    Belief Propagation Loop:
14: while $updateList \neq \emptyset$ do
15:     $n \leftarrow updateList.pop()$;
16:     $parentDist \leftarrow distance(n, n.parent)$;     ▷ Euclidean distance between the current node and its parent
17:     $n.dist \leftarrow parentDist + n.parent.dist$;    ▷ physical distance of node $n$ from root node along the graph
18:     $n.belief \leftarrow gaussianValue(\mu, \Sigma, n.dist)$;  ▷ evaluate the Gaussian at distance $n.dist$ from center
19:     $n.updated \leftarrow$ true;
20:     $neighborList \leftarrow neighbors(Graph, n)$;
21:     for all $neighbor$ in $neighborList$ do
22:         if $neighbor.updated$ is false then
23:             $neighbor.parent \leftarrow n$;
24:             $updateList.append(neighbor)$;
25:         end if
26:     end for
27: end while
```

Figure 3.6: Auxiliary pseudocode of the belief propagation function.

function centered on the location corresponding to the latest reading, increases with time; after some delay, the latest sensory reading does not provide relevant information any longer, so the user might be located anywhere in the building with the same probability.

The software implementing this module is fully aware of the complete topology

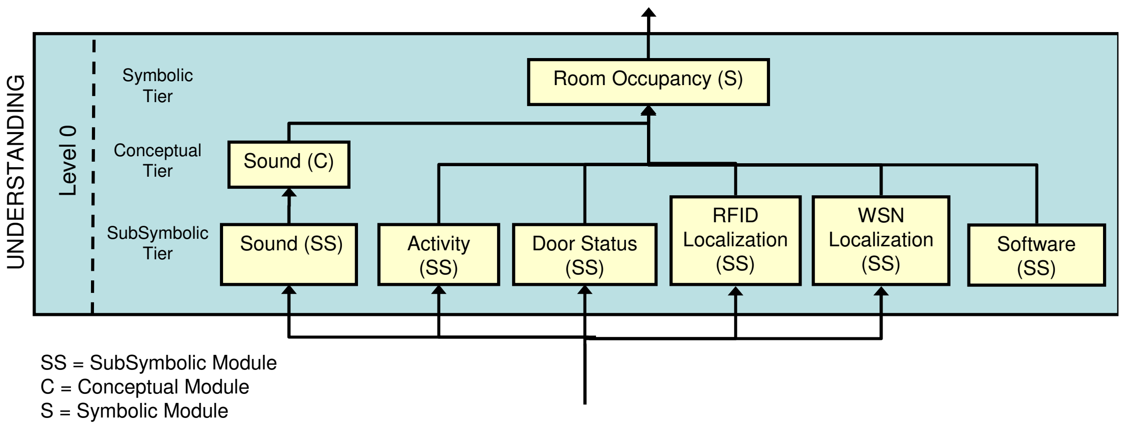


Figure 3.7: The subsystems for room occupancy.

of the building, represented as the graph of the connections between the building areas, in order to propagate the belief about the user's presence. A detailed pseudocode implementing this function is reported in Figure 3.6; it is worth pointing out that the belief value at a specific location does not depend on the *actual* Euclidean distance from the center of the distribution, but rather on the physical length of the complete path along the topology graph (lines 16–18).

The *SW-Sensor* detects the user's logins and logouts at their workstation, thus identifying time intervals when they are not logged in. When the user is performing some activities at the workstation using the peripheral devices, such as mouse and keyboard, the sensor software remains deactivated, and simply acts as a screen-saver; in this case the information on the user's activity is provided by the lack of data. If no activity is detected for a given time interval, it is possible that either the user is no longer sitting at their desk or that they are engaged in some activity that does not requires the use of the workstation; in order to discriminate between the two events, the software sensor activates a visual recognition process through which it tries to recognize the user's face in the images acquired through a webcam, through the approach described in (Ardizzone et al., 2009).

Almost all of the above mentioned sensory information is discrete and does not require conceptual modules for extracting factual information from qualitative data, with the exception of the noise level, whose attached conceptual module uses a statistical characterization of room noise to classify it as `Negligible Noise`, `LowNoise`, `MediumNoise`, or `HighNoise`.

Figure 3.7 shows those architectural modules. The information outcome of the *Activity (SS)*, *DoorStatus (SS)*, *RFId Localization(SS)*, *WSN Localization(SS)*, *Software (SS)* subsymbolic modules is directly handed over to the *Room Occupancy (S)* symbolic module that implements the previously described Bayesian network, while qualitative information produced by the subsymbolic module *Sound (SS)* needs preliminary classification through the *Sound (C)* conceptual module, before passing to the *Room Occupancy (S)* module.

3.3 Heating and Air Conditioning Control

The considered sample application, running on the central AmI server, deals with actuator management and enforces advanced AI algorithms for optimizing environmental control, also taking into account constraints about energy consumption. In particular, the management is performed through a simple fuzzy controller operating on the sensory inputs.

3.3.1 The Core Module

Fuzzy Logic allows to model uncertainty of sensory data and inaccuracy of human-based definitions; in our case, for instance, it allows to manage vague concepts such as *Cold* while speaking about perceived temperature. An AmI application whose core is a fuzzy controller is able to mimic human reasoning, and to simply model a non-linear mapping from inputs to outputs.

Considered input variables are the PT index, its variation rate PT' and the *userPresence* probability value, opportunely fuzzified, while output variables are the variation of the controlled temperature (ΔT) with respect to the current assigned value, and the operation mode of the air conditioner ($Mode$), which can be set to `Cold`, `Off` and `Hot`. A block diagram of this fuzzy controller is shown in Figure 3.8(a).

The fuzzy knowledge base comprises five gaussian membership functions for each input variable, seven gaussian membership functions for the output variable ΔT and three gaussian membership functions for the output variable $Mode$, as shown in Figure 3.8.

The fuzzy engine has been designed so that with a high probability that the user is present in his office the air conditioner is tuned by acting on its functioning mode and temperature, in order to found the best setting that fits with the actual perceived temperature and its variation rate. Rules designed for this case are in the following form:

if $(userPresence$ **is** $Probable) \wedge (PT$ **is** $val_{PT}) \wedge (PT'$ **is** $val_{PT'})$ **then** $(\Delta T$ **is** $val_{\Delta T}) \wedge (Mode$ **is** $val_M)$;

where val_{PT}, $val_{PT'}$, $val_{\Delta T}$ and val_M are linguistic values defined over the ranges of PT, PT', ΔT, and $Mode$ respectively. A sample of a complete mapping from

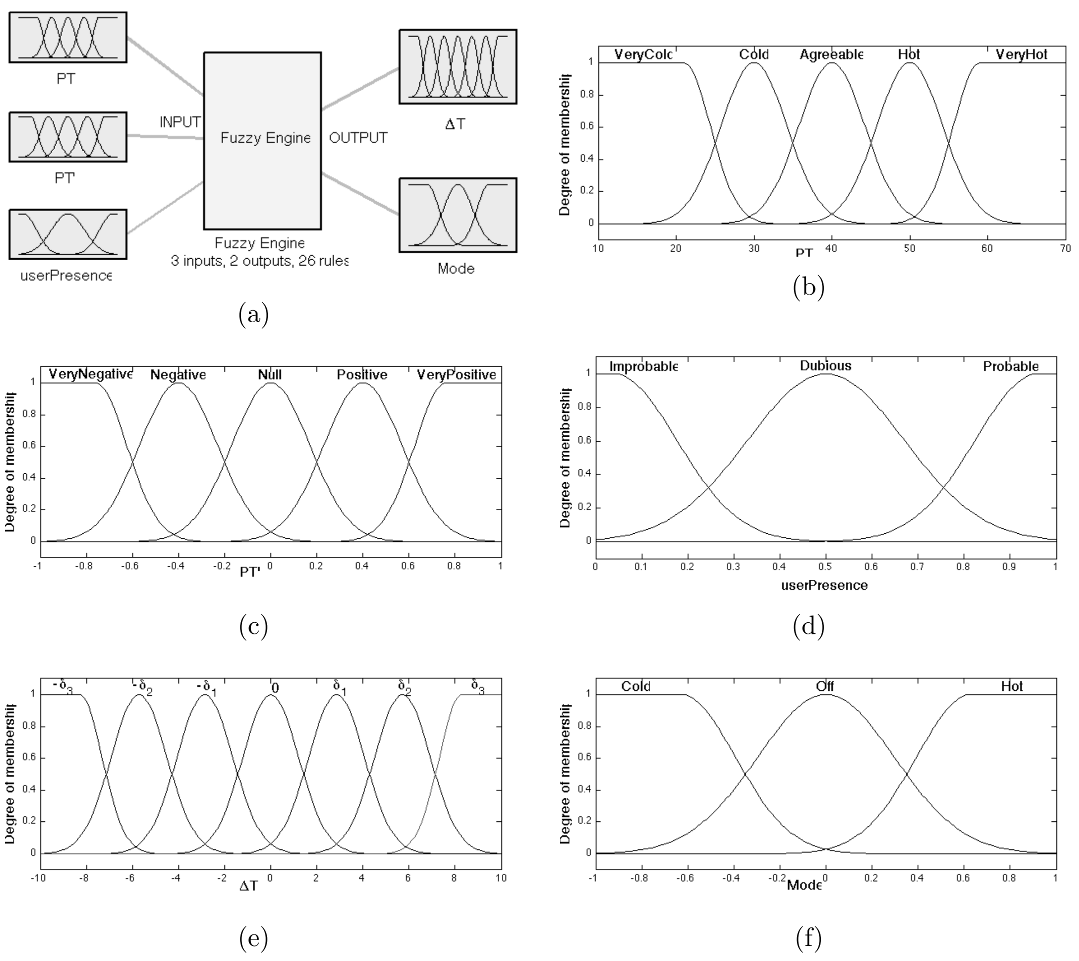


Figure 3.8: Block diagram of fuzzy controller (a) and membership functions for input and output variables: (b) input Perceived Temperature, (c) input Perceived Temperature Variation Rate, (d) input user's presence, (e) output Temperature Variation , (f) output Operating Mode.

$[PT, PT']$ to $[Mode, \Delta T]$, when *userPresence* is `Probable`, is summarized in Tables 3.4 and 3.5.

If *userPresence* is `Improbable` a unique rule is activated, that turns off the air condition in order to meet energy saving goal:

if (*userPresence* **is** *Improbable*) **then** (*Mode* **is** *Off*).

Table 3.4: Fuzzy rules for temperature variation.

$PT \setminus PT'$	**VNeg**	**Neg**	**Null**	**Pos**	**VPos**
VeryCold	δ_3	δ_3	δ_2	δ_1	0
Cold	δ_3	δ_2	δ_1	0	0
Agreeable	δ_2	δ_1	0	$-\delta_1$	$-\delta_2$
Hot	0	0	$-\delta_1$	$-\delta_2$	$-\delta_3$
VeryHot	0	$-\delta_1$	$-\delta_2$	$-\delta_3$	$-\delta_3$

Table 3.5: Fuzzy rules for the operating mode.

$PT \setminus PT'$	**VNeg**	**Neg**	**Null**	**Pos**	**VPos**
VeryCold	`Hot`	`Hot`	`Hot`	`Hot`	`Off`
Cold	`Hot`	`Hot`	`Hot`	`Off`	`Off`
Agreeable	`Hot`	`Hot`	`Off`	`Cold`	`Cold`
Hot	`Off`	`Off`	`Cold`	`Cold`	`Cold`
VeryHot	`Off`	`Cold`	`Cold`	`Cold`	`Cold`

3.3.2 The Autonomic Manager

When dealing with the conflicting goals of adjusting the perceived temperature according to the users' preferences while minimizing energy consumption, the traditional approach of merging them into a single objective function presents several limitations, mainly because it would require an accurate knowledge of the different objective functions, either in terms of relative priority or relevance. On the contrary, we chose to keep two independent objective functions, one expressing the dissimilarity from the users' preferences and the other expressing a degree of energy consumption, as previously mentioned.

The associations between $[state, action]$ pairs (as defined by Eq. 3.3), and the corresponding distance and cost values, are collected by the AmI application as shown in Table 3.6, where n is the number of discrete environment states and m

Table 3.6: Evaluation of $[state, action]$ pairs via two objective functions.

State	**Action**	**Dissimilarity**	**Cost**
s_1	a_1	$d_{s_1}(a_1)$	$c_{s_1}(a_1)$
s_1	a_2	$d_{s_1}(a_2)$	$c_{s_1}(a_2)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
s_n	a_m	$d_{s_n}(a_m)$	$c_{s_n}(a_m)$

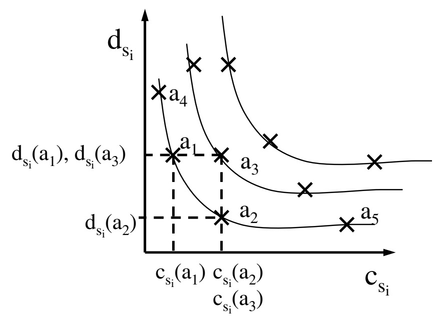


Figure 3.9: Graphical example of the Pareto-dominance analysis.

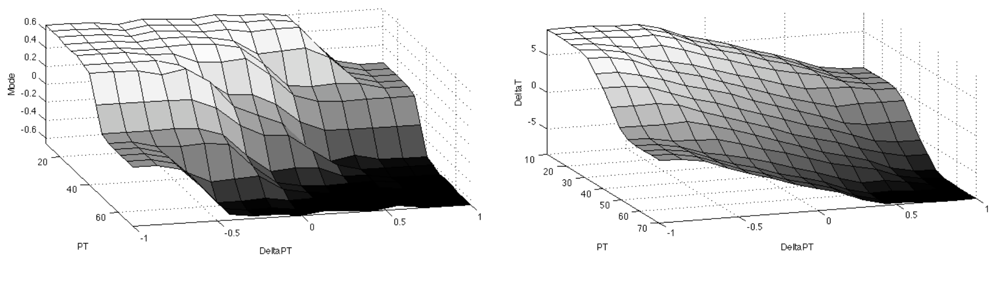
is the number of possible actions.

For each state s_i, optimizing only with respect to the user's preference would involve selecting the action a^* that minimizes the $d_{s_i}(a^*)$ value, whereas optimizing only with respect to the energy consumption would involve minimizing the $c_{s_i}(a^*)$ value.

Since we intend to consider both objective functions at the same time, we adopt a Pareto-dominance criterion for evaluating the actions; this implies the selection of multiple optimal actions.

For a given state s_i, an action a_j Pareto-dominates another action a_k if $d_{s_i}(a_j) \leq d_{s_i}(a_k) \wedge c_{s_i}(a_j) \leq c_{s_i}(a_k)$. An action a^* is Pareto-optimal if no other solution has better values for each objective function, that is if the following equation holds:

$$d_{s_i}(a^*) \leq d_{s_i}(a_j) \wedge c_{s_i}(a^*) \leq c_{s_i}(a_j), \quad \forall j = 1 \ldots m. \tag{3.14}$$



(a) (b)

Figure 3.10: Nonlinear surfaces for the proposed AmI fuzzy application concerning (a) the Functioning Mode and (b) the Temperature Variation.

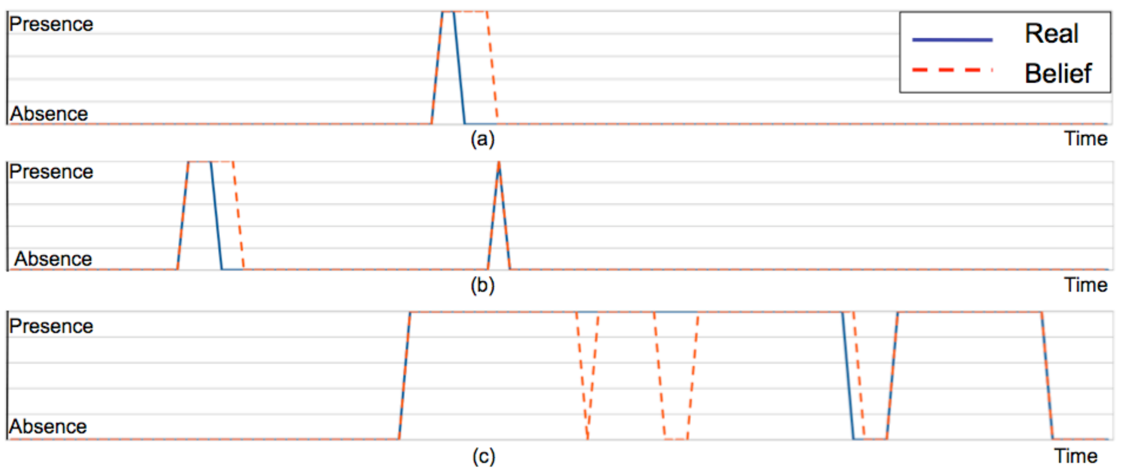


Figure 3.11: Results of the Bayesian multi-sensor data fusion process for user's presence detection (Test A).

Figure 3.9 represents an example of the Pareto-dominance analysis for a given ambient state s_i: actions a_1 and a_2 belong to the same non-dominated front because $d_{s_i}(a_2) \leq d_{s_i}(a_1)$ and $c_{s_i}(a_1) \leq c_{s_i}(a_2)$, while both actions a_1 and a_2 dominate action a_3; the set of optimal actions is $\{a_1, a_2, a_4, a_5\}$.

The action to perform in a given ambient state is selected inside the set of Pareto-optimal actions; namely, it is the action that represents the median point on the curve of the optimal front.

Finally, Figure 3.10 shows the nonlinear surfaces produced by the Pareto-optimal fuzzy controllers that models the mapping among inputs and outputs.

3.4 Experimental Evaluation

We assessed the performance of our AmI sample application as a whole, and of the individual modules of the testbed; we report here a representative subset of our tests conducted over a three days time period. We considered one target user, who was unaware of the ongoing experiment, and so did not modify his usual behavior. In order to validate the subsystem for detecting the user's presence, its outcome was compared with information obtained by manually analyzing the output of a video-surveillance system.

The plots reported in Figure 3.11 show how the system performs in detecting user's presence. Figures 3.11(a), 3.11(b), and 3.11(c) consider three different time intervals, and each of them shows two superimposed plots: the continuous line represents the actual trend of the user's presence signal; the dashed line represents the belief of the system about the user's presence, as obtained by the Bayesian

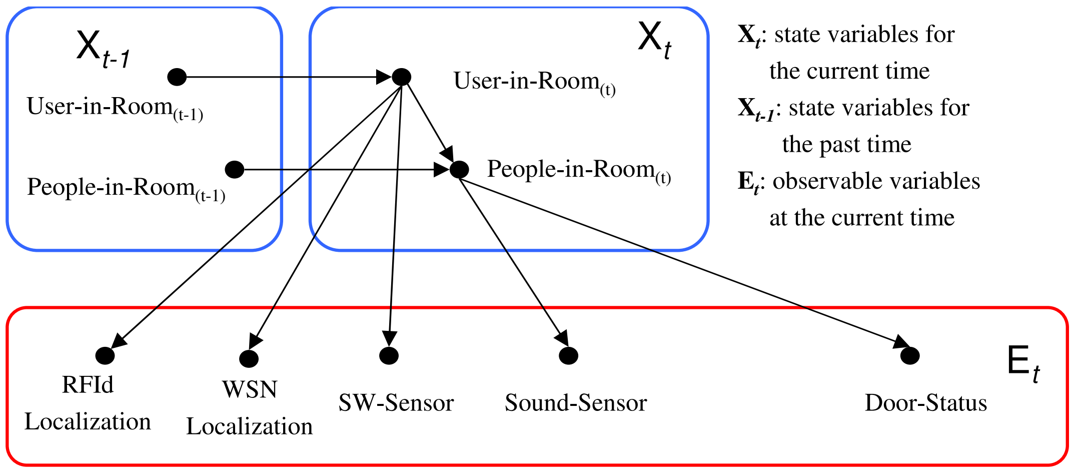


Figure 3.12: Structure of the Bayesian network without the ActivitySensor (Test B).

multi-sensor data fusion process, previously described (see Figure 3.4).

In order to get a deeper understanding of the behavior of this module in scenarios including a limited number of sensory devices, we carried on further experiments highlighting the impact of specific sensory information; namely, we iteratively excluded some of the sensory inputs from the probabilistic inference process, and run the tests on the same data in order to get comparable results. The new tests indicate that the user detection module is robust with respect to the exclusion of a limited number of sensor types. It is however clear that there exists a critical threshold for the number of excluded sensor types beyond which the results deteriorate intolerably. Figure 3.12 shows the Bayesian network obtained after excluding the *Activity-Sensor*; the corresponding test is labeled “Test B”, as opposed to the original test including all the available sensors, and labeled as “Test A”. As shown by the results in Figure 3.13, the presence detection system succeeds in compensating the lack of information by exploiting what is still available. The greatest difference appears at the beginning of time interval (b), when the software module detects the user only with some delay. Figure 3.14 reports the results from “Test C”, where the *Door-Status* sensor is missing in addition to the *Activity-Sensor*; the figure shows that additional errors are present as compared to “Test B”, both in time interval (b) and (c).

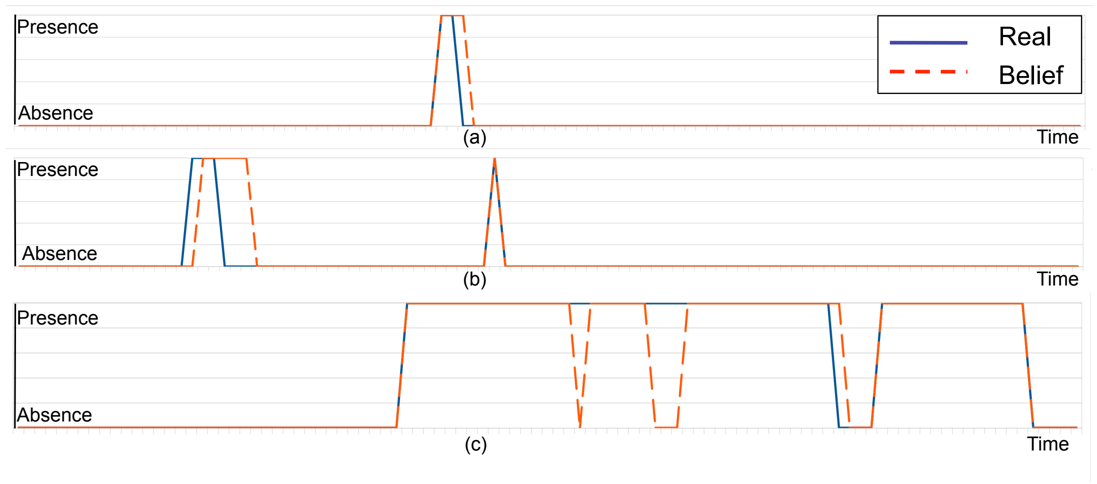


Figure 3.13: Results of the Bayesian multi-sensor data fusion process for user's presence detection without the ActivitySensor (Test B).

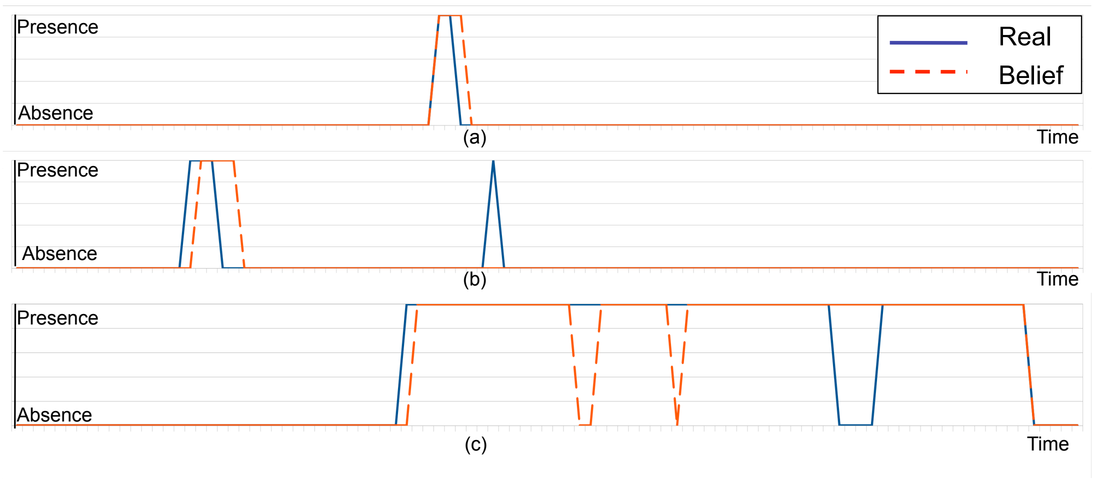


Figure 3.14: Results of the Bayesian multi-sensor data fusion process for user's presence detection without the ActivitySensor and the DoorStatus sensor (Test C).

Table 3.7: Specificity and sensitivity of the module for detecting the user's presence, in different sensor devices available.

Test	**Excluded Sensor Types**	**Specificity**	**Sensitivity**
Test A	none	97.50%	93.33%
Test B	*Activity-Sensor*	97.92%	91.67%
Test C	*Activity-Sensor*, *Door-Status*	97.08%	90.00%

In order to obtain a statistical evaluation of the system performance we discretized the time intervals and computed false positives and false negatives; the specificity and the sensitivity of the system are computed according to the following definitions:

$$\begin{aligned} specificity &= \frac{\#true\ negatives}{\#true\ negatives\ +\ \#false\ positives}; \\ sensitivity &= \frac{\#true\ positives}{\#true\ positives\ +\ \#false\ negatives}. \end{aligned} \tag{3.15}$$

If the gathered sensory information is not sufficient to infer the user presence, the software module opts for a default strategy that indicates the user's absence. Such choice implies that sensitivity is the most relevant factor for the assessment of the system performance. In order to get a deeper insight on that, we can imagine a scenario where the software module is constantly unable to detect the user's presence; in such scenario the module keeps indicating user absence, hence no false positives may arise and specificity amounts to a constant 100%; however the anomalous situation shows clearly by looking at the sensitivity parameter, which falls to 0. Both the specificity degree and the sensitivity degree are reported in Table 3.7 for all conducted tests, and the worsening performance of the system are shown by the lowering sensitivity degree throughout the tests. It is however worth noting that the system still produces good results, even when lacking groups of sensory devices.

In order to assess the effectiveness the system's decisions as regards the resulting environmental conditions, we measured the mismatch between the current ambient state, and the desired ambient state inferred by observing the interactions of the user with the actuators. We used the same time discretization as for the localization system, and labeled each time interval as "satisfactory" if the user was in fact present and no interaction was detected, whereas the opposite label is used if the user modified the actuators settings at least once in that interval.

In order to assess the performance of the AmI application, we need to take into account 6,6% of the cases where the user was present in the office, but the localization system failed in detecting them. This in fact prevented the triggering of the rules for environmental control. This testbed-dependent error must be added to the application-dependent error, so that overall the AmI application sets the environmental state coherently with the user's preferences in 84,5% of the cases, which is a remarkable behavior nonetheless.

Chapter 4

Communication with the External World

The architecture described so far allows to develop AmI systems able to fully understand its internal conditions, also those regarding its occupants, and to act in order to satisfy its goal. Nevertheless these introspective capabilities of monitoring, understanding and acting are not enough in order to meet user's requirements; the AmI system has to be able to communicate with the external world in order to obtain services that are useful for reaching its own goals. Just as an example, we can consider systems for automatic food supply, or for communicating with hospitals and first aid centres, or for communicating with energy providers in order to gather information about possible constraints on monthly average consumptions and scheduled blackouts; moreover, we can devise systems able to communicate with several provider for a give service select the most convenient economic conditions.

The most suitable architectural model in order to support the design and development of a intelligent building, as a node of the "smart city", and thus able to communicate with other agents in order to exchange products and services, is the Service Oriented Architecture (SOA) model. In the devised smart city the AmI system represents a consumer node; this agent needs to acquire distributed services from unknown service providers on the Internet and on the network covering the smart city.

Over the Internet, SOAs are typically implemented through the use of web services standards, and rely on a centralized approach, that requires the presence of a master node (trusting authority) maintaining relevant information about network services and the relative providers. This approach suffers from well-known limits of centralized systems, i.e. lack of scalability and presence of a single point of failure.

Distributed SOA architectures (D-SOA) (Banaei-Kashani et al., 2004) represent an important evolution of classic SOAs and can overcome their limits using a hierarchical network structure, for distributing workload among several network nodes. This architectural paradigm is well-suited in those scenarios in which trusting authority is implicitly distributed.

The main goal of consumer nodes is the selection of the best services among the huge multitude provided by the network. As basic criteria for this choice, service cost and Quality-of-Service (QoS) can be considered, provided that the underlying SOA be augmented in order to support the declaration of this information. However, in distributed environments where heterogeneous agents collaborate and interact in order to achieve their own goals, obtaining guarantees on their behavior from some reliable central authority is typically unfeasible. In such scenarios, a reputation management system capable of building a profile representing the reliability of each agent can be extremely useful. Knowing agent reputation is particularly helpful for detecting those agents that are deceitful or potentially dangerous for the community. In a non-centralized environment, interactions among agents are unpredictable and no central authority is present to carry out supervision and coordination activities. In this case, the lack of a centralized trusting authority may encourage antisocial behaviors, that in this scenario consist in declaring false QoS values. Thus, the correct behavior of such new SOA platforms, however, will depend on the presence of some mechanisms that allow consumer nodes to evaluate trustworthiness of service providers.

This work proposes a new methodology for discouraging antisocial behaviors, over an architecture fully distributed over the network, based on a reputation management schema. We present a general approach for reputation management in distributed environment and then we specialize it for the scenario of a distributed SOA, that represents a suitable model for a global AmI environment.

4.1 Reputation Management in Distributed Systems

This section presents a general solution for reputation management in distributed system. The proposed approach will be specialized for service oriented architecture in the following section.

A perspicuous example of distributed systems where such selfish behaviors may have a negative impact on the overall performance is represented by P2P networks; in those distributed systems, agents with equivalent functionalities interact without any centralized coordination. Unstructured P2P networks, in particular, achieve the highest decentralization degree: the network is self-organized without any predefined overlay structure. Unfortunately, the lack of a centralized control facilitates the occurrence of free riding phenomena, characterized by a resource exploitation by a few system agents. For instance, in file-sharing applications, free riders share few or scarcely appealing resources. Several recent works (Ripenau and Foster, 2002; Saroiu and Gummadi, 2002) have analyzed the relevance of this phenomenon in different models of P2P networks such, for instance, the Gnutella network. The general conclusion is its potential threat for the survival of the network itself. It modifies the network structure, that degenerates into a client/server architecture, with the consequent performances decrease. Free riders deliberately do not cooperate since collaboration implies some costs, and this behavior reduces the perceived utility. In P2P networks costs are related to the resources that a peer has to make available in order to actively connect into the system; among these, link bandwidth, CPU time of the hosts where P2P daemons run, and storage space. Such resources are used both when a peer directly answers in order to provide the requested resource, and also when a peer simply collaborates in order to find some resources, routing queries coming from other peers.

This Section proposes a distributed reputation management scheme, where participating agents interact in order to build a view of the network that is as homogeneous as possible with respect to agents reputation. The reputation management mechanism begins with a local assessment and then, through a reputation diffusion algorithm, propagates information among cluster of mutual reliable agents. The proposed system has been tested on an unstructured P2P network that adopts a flooding-based routing protocol for resource localization.

The remainder of the Section is organized as follows: in Section 4.1.1 other works presented in literature are described, both related to the general issue of reputation management in distributed systems, and to the particular problem of contrasting the free riding phenomenon in P2P networks; in Section 4.1.2 some basic principles of game theory are summarized and the choice of the evolutionary game theory, as the best choice for these complex systems, is motivated; Section 4.1.3 describes the proposed reputation management system and Section 4.1.4 reports the experimental results obtained by preliminary simulations.

4.1.1 Related Works

Reputation Management in Distributed Environments

In the past few years, several solutions have been proposed in order to provide reputation management in distributed environments. Most of the proposed approaches aim to evaluate agent reliability in a generic distributed system. Reputation management systems can be classified into *centralized* and *distributed* approaches.

In the *centralized* approach, a central reliable entity monitors transactions among agents in the community and computes a reputation value for each of them. Typical applications of centralized reputation systems are e-commerce systems, such as eBay, in which after any transaction, the buyer assigns a score in order to evaluate seller reliability. Epinions (Epinions, 2010) introduces the concept of double reputation: a reputation value for each agent providing a service, and a reputation value for each user expressing an opinion about a service. The idea is very interesting, since users with higher reliability have more influence during the overall reliability evaluation of a service or product; this way the reputation value is highly reliable. However, both approaches may prove unpractical in a fully distributed environment.

In the *distributed* approach no centralized control structure is present, rather each agent needs to collect data in order to calculate its own estimation about the reputation of all other agents, which it gets in contact with. This information could be shared among all the agents belonging to the same communities, using several information diffusion mechanisms. The EigenTrust project (Kamvar et al., 2003) adopts a distributed approach in order to calculate a global reputation value in a P2P network. It uses the satisfaction level expressed by peers, at the end of each interaction; this value constitutes a local reputation value. Local values are merged in order to provide a single, global reliability parameter. Each peer calculates a reputation value for its neighbor peers, averaging the reputation values provided by peers directly connected. Weights used in the average computation are the reputation values of evaluating peers. The global reputation value is computed by diffusing this evaluation to peers located more hops away from the evaluating peer, again averaging values obtained by information propagation. The proposed system does not explicitly take into account bad reputation: an unknown peer is simply considered a malicious peer.

In (Songand et al., 2005) a reputation management system based on fuzzy logic is proposed. It describes a system capable of successfully managing uncertain and partial information reported by other network peers. As in previous works, reputation values are at first collected locally and then propagated over the network; the information gathering mechanism is again based on the weighted average of reputation values, although enriched with a fuzzy logic system. Both approaches adopt Distributed Hash Tables (DHT) in order to maintain a unique global reputation. Using a DHT structure involves that a single peer is responsible for holding the reputation value of a given peer; every time this information is requested, the holding peer must be contacted; furthermore, each node failure on the network causes the reallocation of the managed identifiers. Previous considerations make it clear that DHT-based solutions are unpractical for highly dynamic systems such as P2P unstructured networks. In (Guha et al., 2004), in addiction to the good reputation value, the evaluation of a bad reputation value is proposed. This is a distinguishing feature in comparison to other works proposed in literature. However, authors assume the full collaboration of peers during the information diffusion, and this assumption appears rather unrealistic.

Reputation Management for Countering Free Riding

Most of the approaches proposed in literature aim to address free riding by using reputation management systems. These approaches try to estimate the goodness of a peer, as its cooperation degree. After estimating a peer's goodness, it is possible to employ a service differentiation, allowing more collaborative peers to obtain higher quality services. In (Ranganathan et al., 2003), the authors propose two different incentive systems based on reputation, in which a reputation score is assigned to each user. In the first case, each user is authorized to obtain resources only from users with lower or equal score; in the second one, the reputation index is used in order to give higher priority to requests coming from collaborative users. Both methods require a secure and reliable mechanism to maintain user reputation values. However, the overhead of maintaining and broadcasting this information becomes prohibitive for fully distributed networks. The authors of (Mortazavi and Kesidis, 2006) propose a distributed reputation system in which every peer maintains a reputation value about other peers which it gets in contact with; in this case, peer reputation depends on the provided upload bandwidth. To the best of our knowledge, this is one of the few works where some ideas borrowed from evolutionary game theory are adopted in the context of P2P networks. The total upload bandwidth shared by a peer is estimated through a mechanism based on Simulated Annealing algorithm, and the considered approach tries to estimate the reward associated to each upload bandwidth value.

Our paper relies on the reputation management system proposed in (Mortazavi and Kesidis, 2006), improved and modified in order to manage bad reputation in addition to good reputation, and to allow reputation diffusion.

4.1.2 Game Theory Background

Game theory aims to model systems composed of several agents interacting with each other while still trying to achieve their own particular goal, which typically consists in the maximization of a utility function representing the benefits received by each agent in a given situation. A situation is the result of all the actions accomplished by all other agents (called players) involved in the interaction (called game). Therefore, game theory is used to represent scenarios in which an agent cannot completely control the utility it perceives, which also depends on the choices made by other agents.

A particular category of games is represented by non-cooperative ones; in such games, an inefficient equilibrium is reached as a result of all agents making rational choices, i.e. selecting actions maximizing their own utilities. An equilibrium point is inefficient if there exists a different situation in which the utility perceived by each agent is higher. In non-cooperative games, the efficient equilibrium point cannot be reached; namely, in the absence of a full cooperation among players, each agent perceives that not cooperating is more convenient than doing it. In such games no reliable third party assumes the role of forcing all participating agents to cooperate. Unfortunately, this is the case in which most distributed systems fall.

Evolutionary Game Theory

Classic game theory studies strategy games, in which agents are considered perfectly rational: each agent knows exactly all actions that other agents are allowed to perform, as well as the utility value perceived by each of them in each situation. A perfectly rational agent is thus able to select the best action that maximizes its own utility, given the choice performed by other agents. Most real-life complex systems violate these theoretical conditions, so classical game theory results unsuitable. In particular, in several scenarios, agents do not know their own utility function, and least of all, the utility function of other agents. In P2P networks, for example, peer utility can be defined as the benefit perceived when obtaining resources from other peers, net of costs requested for cooperation. It is not convenient to express this function in a closed form, too heavily bound to system design choices, and the most appropriate approach consists in learning it during the continuous interaction with other agents.

In distributed environments, in which a wide set of not perfectly rational players repeatedly interact, classical game theory does not appear as the best tool for modeling the system. The most promising tool for modeling these complex systems appears to be a particular branch of game theory, known as evolutionary game theory (Kontogiannis and Spirakis, 2005). It analyzes the behavior of a population of players. The game is played several times between pairs of players, selected randomly. At the end of each match, the loser is able to change its penalizing strategy, allowing the evolution of the community strategy toward the most rewarding one. This approach allows considering more complex realms, in which agents are not perfectly rational, either because they do not have a complete and exact vision of the system, or even because they do not retain the computing capabilities required to find the best strategy among those possible. In evolutionary games, the best strategy is not typically computed as a closed-form expression; on the contrary, the best strategy emerges form a trial-and-error learning mechanism, thanks to which players can understand that, in a given situation, a given strategy is preferable to another one.

The way players modify their strategy, according to the results of previous matches, is called *evolutionary dynamic*, and several aspects may be considered, among which the most popular is the replicator dynamic. This assumes that the diffusion speed of a given strategy depends directly from the difference between the utility of this strategy and the population average utility. However, this approach does not focus on the learning mechanism of the single player that, as previously seen, does not possess the system global view. For this reason, it results suitable only for an external system analysis. The simplest replicator dynamic, among those focused on the individual choice method, is the imitation dynamic that consists of the trivial emulation of the winning strategy, after each match. A more robust approach may involve the exploitation of the past experience and of the current conditions in order to learn the best strategy. In this context, any learning method can be adopted, such, for instance, the reinforcement learning algorithm.

4.1.3 The Proposed Reputation Management System

This paper proposes a fully distributed reputation management system, in which autonomous agents interact in order to build a reputation value of all participating agents. The aim is to build a reputation estimate that will result as homogeneous as possible among all agents. As already mentioned, we adopted as case study the scenario offered by an unstructured P2P network. This class of networks represents a fitting example of distributed environments that are plagued by antisocial behaviors. Furthermore, because of their strong autonomy and dynamism, P2P networks are not suitable for the adoption of a structured approach, such as the systems based on DHT. The estimated reputation values are used within an incentive system whose aim is the penalization of not cooperating peers. The main feature of our system is its capability to adapt the utility perceived function during the interaction among agents. Through this modification, the cooperating behavior becomes a more appealing choice. In our system, this means that peers are motivated both to share more precious resources and to actively attending to the query forwarding process. In fact, the greater the cooperation level of a peer, the greater its reputation value maintained by agents with which it has interacted.

The incentive mechanism privileges peers with higher reputation, supplying them with more resources with respect to other system agents. Peers that are considered free riders are penalized, rejecting their requests with a probability proportional to their bad reputation.

In the reputation management mechanism two phases can be distinguished: a former for building a local reputation estimation and a latter where the reputation information is broadcast in all the community. The first phase (local estimation) exploits the information collected during the interaction among peers, integrating past values and information about the current context. During the reputation diffusion phase, pairs of mutually reliable peers exchange their local reputation values, so each peer can integrate the received information with its own local reputation value.

Local Reputation Estimation

The sub-system in charge for the local reputation estimation, requires that each peer estimates a cooperation degree of other peers with which it has interacted. The purpose of our incentive system is to achieve a stable equilibrium between resources that are provided and requested by an interacting peer. In particular, the proposed mechanism aims to maintain a balance among each pair of peers in the network. The local reputation mechanism uses the difference between supplied and obtained resources from a peer to another one, advantaging both peers in a balanced situation or the more generous peer otherwise. On the other hand, those peers, for which the number of obtained resources results greater than supplied ones, are penalized.

Two different values are used to represent trust and distrust about a peer; in the following these values are called *good reputation* and *bad reputation.* The design choice of adopting the different values was made according to what proposed in (Guha et al., 2004). In several works present in the literature, a single reputation value is adopted. However, using only this unique value, often, it may result hard to distinguish between new unknown peers and peers with malicious behaviors. To overcome this difficulty, in our system both the values are considered into a reinforcement-learning scheme that uses the difference between requested and supplied services.

Furthermore, most of the previous systems consider, as supplied services, only successfully completed downloads. The query forwarding mechanism, which indeed constitute the fundamental core of any file sharing P2P application, is not considered. At the best of our knowledge, paper (Li et al., 2004) is the only work considering the query routing as a supplied service, although a system based on token-exchange, rather than a reputation-based approach, is adopted. However, using tokens as a virtual coin requires a centralized trust authority. As extensively noted this kind of system is not suitable for unstructured P2P networks.

In our reputation mechanism, a peer maintains a reputation profile for each other peer, which has been previously contacted. The reputation profile maintained by peer i about peer j, contains the number of requests that have been satisfied for the peer j (req_i^j) and the number of services obtained from peer j ($serv_i^j$). More precisely, req_i^j represents the number of responses generated and the number of the query routed, by the evaluating peer for the peer j; $serv_i^j$ is the number of resources obtained and responses routed, by the peer j for the evaluating peer. The above values are used to calculate the local values for the instantaneous estimation of bad and good reputation for peer j ($\hat{gr}_i^j$ e $\hat{br}_i^j$), together with actual reputation values collected so far (gr_i^j e br_i^j), according to the equation 4.1. The actual reputation values are updated on the basis of the instantaneous reputation values, through a reinforcement learning approach. Indicating with $r_i^j(t)$ the generic reputation value at time t (good or bad) held by peer i about peer j, the update is carried out according the equation 4.2.

$$\begin{aligned} \hat{gr}_i^j &= \begin{cases} max(1 - \frac{req_i^j}{serv_i^j},\ gr_i^j), & if\ serv_i^j \geq req_i^j; \\ 0 & otherwise. \end{cases} \\ \hat{br}_i^j &= \begin{cases} max(1 - \frac{serv_i^j}{req_i^j},\ br_i^j), & if\ serv_i^j < req_i^j; \\ 0 & otherwise. \end{cases} \end{aligned} \tag{4.1}$$

$$r_i^j(t) = \alpha * \hat{r}_i^j(t) + (1 - \alpha) * r_i^j(t-1). \tag{4.2}$$

Reputation Diffusion

The protocol for the reputation diffusion aims to broadcast information about the peer behavior, among neighbors considered mutually reliable, so as to obtain a view of the network that is as homogeneous as possible, with respect to the reputation values. The diffusion mechanism to some extent mirrors human social interactions: when two agents consider themselves mutually reliable, they exchange information about other members of their community, in order to integrate the directly acquired information. Leveraging only the information coming from reliable agents avoids that malicious agents may attack the reputation system providing forged values. Furthermore, providing the reputation information only to reliable agents constitutes an additional obstacle for non-cooperative agents.

The diffusion protocol periodically refreshes reputation values maintained by each peer; this way, an extremely updated vision of other agent reputation is always maintained. Information obtained through the diffusion protocol contributes to determine the actual reputation value. The actual generic (good or bad) reputation value, held by peer i about peer j, $r_i^j(t)$, is computed adjusting the value collected so far, $r_i^j(t-1)$, with a weighted average of information obtained from other agents, $r_k^j(t-1)$. Weights used in this average are good reputation values of "gossipy" agents, $gr_i^k(t-1)$. This is a sort of distributed consensus algorithm (Johansson et al., 2007), with the difference that our approach takes in account the reputation of gossipy agents. The actual reputation value is computed according to equation (4.3).

$$r_i^j(t) = (1 - \beta) * r_i^j(t-1) + \beta * \frac{\sum\limits_{k \in K} gr_i^k(t-1) * r_k^j(t-1)}{\sum\limits_{k \in K} gr_i^k(t-1)}, \tag{4.3}$$

where K is the set of reliable gossipy agents for which the good reputation value exceeds an opportune threshold τ, experimentally determined:

$$K = \{k : gr_i^k(t-1) \geq \tau\}. \tag{4.4}$$

A change of the threshold τ will affect peer willingness to rely on neighbours.

4.1.4 Performance Analysis

In order to evaluate the proposed management reputation system we performed extensive testing in several simulation scenarios. A simple Java simulator has been developed and implemented to model unstructured P2P networks, with the aim of verifying the effectiveness of the proposed reputation management system. In our simulations we randomly generated several network topologies, constituted by 100 peers, each of them with a fixed number of neighbors, selected according to a uniform distribution. For each network we assume that the 20% of peers are free riders, which provide only the 10% of requested services. A flooding-based algorithm was adopted to forward queries; queries concerned resources uniformly selected from a given domain. The generation time between two successive queries follows a Pareto distribution. Using the same basic network scenario, we carried out simulations in three different configurations. In the first one, a very basic setting, any incentive mechanism is used, in the second one we introduced only the local reputation mechanism, while in the last one we adopted the full approach, constituted by the local reputation together with information diffusion.

The metrics used for comparisons were: the number of resources received by a free rider normalized with respect to the average number of resources received by a cooperative peer (Figure 4.1), and the mean value of actual good reputation for cooperative peers in comparison to good reputation of free riders (Figure 4.2). As shown in the Figure 4.1, without reputation management, a free rider has about the same resources than a cooperative peer. This is an expected result. The interesting result is that, using the local reputation, a free rider receives, on average, the 53% of resources with respect to those obtained by a cooperative peer. This percentage decrease up till the 44% when the reputation diffusion is enabled. In Figure 4.2 it is shown that the difference between good reputation for free riders and for cooperative peers is clear. Thanks to the reputation diffusion, this difference increases, allowing a more effective characterization of malicious peers.

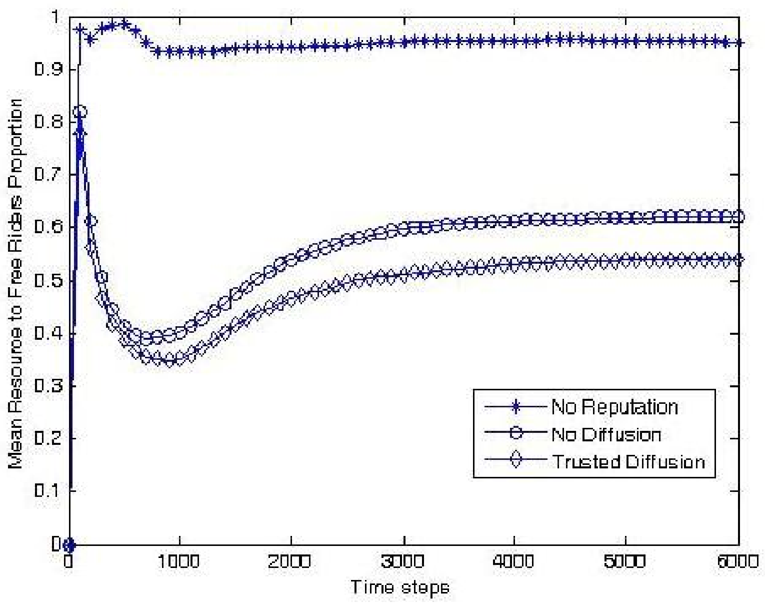


Figure 4.1: Resources obtained by free riders and by cooperative peers

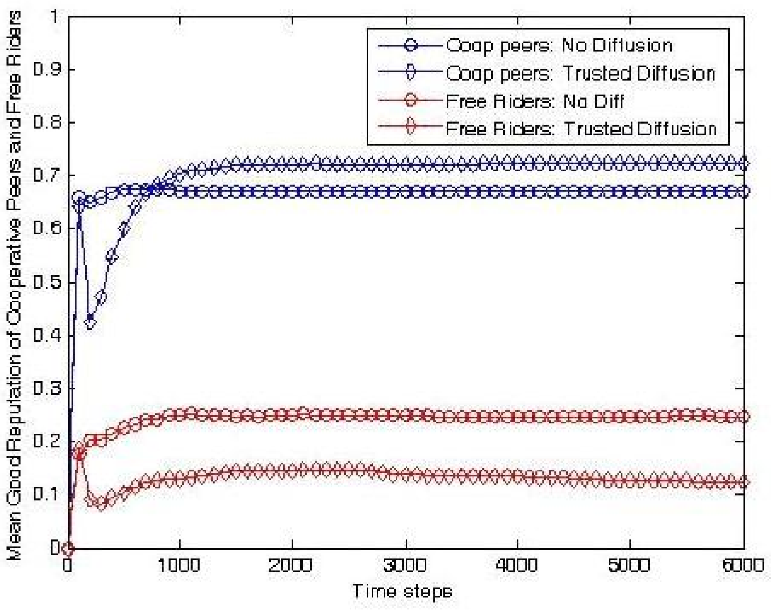


Figure 4.2: Good reputation for free riders and for cooperative peers

4.2 Reputation Management for Distributed Service-Oriented Architectures

A Service-Oriented Architecture is a software platform that describes the structure of service-oriented networks. Over the Internet, SOAs are typically implemented through the use of web services standards, such, for instance, WSDL (Web Service Description Language), an XML-based language that describes functionality and interfaces of web services, SOAP (Simple Object Access Protocol), a protocol specification that allows interaction between web services, and UDDI (Universal Description Discovery and Integration), an XML-based universal registry of web services. The typical SOA definition relies on a centralized approach, that requires the presence of a master node (trusting authority) maintaining relevant information about network services and the relative providers. Such information is stored in a centralized directory (UDDI) providing WSDL documents to the requesting applications, using SOAP protocol. This approach suffers from well-known limits of centralized systems, i.e. lack of scalability and presence of a single point of failure.

Distributed SOA architectures (D-SOA) (Banaei-Kashani et al., 2004) represent an important evolution of classic SOAs and can overcome their limits using a hierarchical network structure, for distributing workload among several network nodes. This architectural paradigm is well-suited in those scenarios in which trusting authority is implicitly distributed, for instance, as in Virtual Organizations. A Virtual Organization is a network of cooperating organizations that interact by purchasing and selling services. According to this representation, each organization is responsible for its own resources in the network, and it is not feasible to entrust the management of all network resources to a centralized trusting authority.

In order to overcome this problem, we propose a hierarchical structure in which each organization represents a trusting authority for services held by its providers. However, the lack of a centralized trusting authority may encourage antisocial behaviors, that in this scenario consist in declaring false QoS values. This behavior is fully explained by game theory, according to which, the analysis of agent interactions in a real complex scenario cannot take into account the quality of being honest. On the contrary, each agent selects its own actions in order to achieve its maximum advantage, to the best of its knowledge, even if they cause damages to other agents. In our scenario such an opportunistic behavior consists in the untruthful declaration of QoS values higher than the real ones, in order to guiltily promote inferior services. This consideration imposes that the traditional centralized trusting authority has to be replaced by a distributed one that offers equivalent functionalities.

According to the taxonomy presented in (Vassileva and Wang, 2007), our system can be defined as *personalized* and *decentralized.* It is decentralized because of the lack of a central entity managing information; rather, information on reputation and QoS are spread over the network. It is personalized because different nodes can have different reputation values for the same service provider; similarly, different nodes can maintain different QoS assessments for the same service. Provider reputation is managed by exploiting consumers feedbacks, released after service usage. The smaller the difference between declared and actual QoS, the greater the client satisfaction, with a consequent increase of the provider reputation. After an initial transitory phase, the system will converge toward an accurate estimate of the actual QoS values. Reputation values will be used in a mechanism of penalties and incentives, in order to allow consumer agents to identify malicious untruthful nodes.

The rest of the paper is structured as follows. In Section 4.2.1 other works presented in literature are described, as they contain some key concepts exploited in this work; Section 4.2.2 describes the proposed architecture, while details about the adopted policies for the management of QoS and reputation are provided in Section 4.2.3. Finally, Section 4.2.4 reports the experimental results.

4.2.1 Related Work

The approaches presented in the scientific literature related to the problem of separating malicious and truthful nodes in SOAs involve different techniques, such as exploitation of users feedbacks in order to produce single QoS estimate, single provider reputation estimate, or finally a hybrid QoS and reputation estimate.

In (Wang and Vassileva, 2003), the authors propose a model for reputation management in peer-to-peer networks. Information regarding the peers reputation is managed using ad hoc developed bayesian networks that periodically are exchanged among all peers. The reputation is updated by means of a reinforcement learning technique. This work, to the best of our knowledge, is the first one which proposes the adoption of reinforcement learning in order to model reputation; however, authors do not consider any trusting authority capable of penalizing malicious nodes.

In (Vu et al., 2005), a decentralized system for reputation management and service selection is proposed. The approach is based on the use of monitor nodes, that collect QoS values of the service providers. QoS values are compared with users' feedback in order to filter out deceitful providers through clustering methods. This work presents a distributed form of trusting authority. As far as the computational complexity is concerned, however, this system is extremely inefficient because of the complexity of the adopted filtering algorithm.

Authors of (Ran, 2004) propose the use of a Certification Authority (CA) in order to check that declared QoS match their real value. Network clients therefore can rely on CA for purchase network services. Such approach does not take advantage from users' feedback, and the CA represents a centralized bottleneck that prevents the full scalability of the system.

A system capable of integrating users feedbacks and reputation is proposed in (Xu et al., 2007). Reputation is computed by a weighted sum of users' feedbacks, as a function of the feedback age. The approach is fully centralized, since user's feedback are stored in a centralized database. Moreover, QoS is not evaluated for each service, rather it is assumed that truthful service providers update this information. This last assumption is fully unrealistic for the scenario considered in our work.

In (Menasce and Dubey, 2007) authors introduce the concept of service broker. The task of a broker is to seek those services in the network that better match user requirements, in order to maximize the customer utility function. The utility function for a service is computed under different conditions of load. Although a broker based approach may be useful in some architectures, the service broker computational load may be excessive in networks with a high density of providers.

Finally, authors of (Liu et al., 2009) propose a system that expands traditional SOA with three additional components: QoS registries, Universal QoS matching and Web Service Broker. Brokers monitor the invoked services and compute a QoS value. Universal QoS matching compute the service QoS by a weighted sum of declared QoS, feedback QoS and monitored QoS, in which weights are directly proportional to the age of the information. Such approach, however, does not adopt any reputation management mechanism capable of inducing service providers to declare real QoS values.

Works discussed here presented some key concepts, like trusting authority, QoS estimate, reputation management and network monitors; nevertheless none of them merges all these aspects in a comprehensive approach. One of the contributions of our work is thus the integration of QoS estimate and reputation management in a single distributed mechanism. The proposed system has the advantage of effectively detecting malicious behaviors, maintaining the computational load low thanks to the exploitation of a hierarchy of authoritative nodes.

4.2.2 The Proposed Architecture

Our work proposes a system capable both of managing the providers reputation and of estimating real QoS values in a distributed SOA. The proposed architecture is well-suited for those scenarios in which trusting authority is implicitly distributed, since it allows the achievement of high degrees of guarantee for the QoS, still maintaining a full autonomy for the local resource management. Several real scenarios fall within this description, for instance Virtual Organizations (Norman et al., 2004) and Cloud Computing (Buyya et al., 2009), whose main purpose

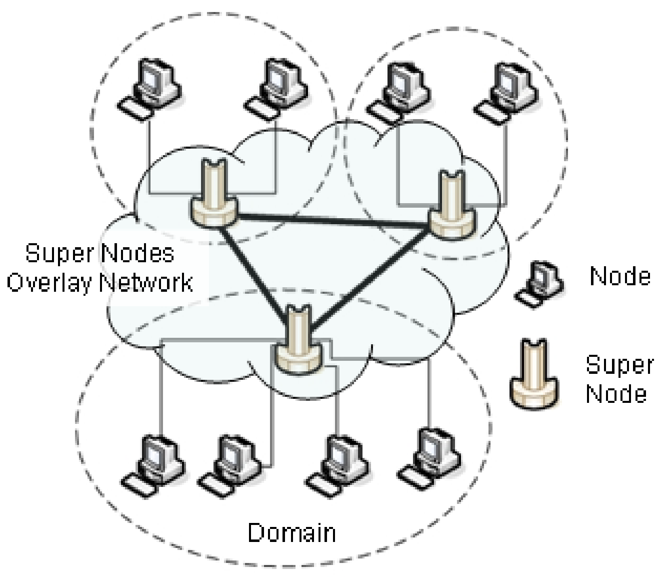


Figure 4.3: Clustered hierarchy organization of *Super Nodes* and *Nodes*.

is to dynamically coordinate different institutions in order to exchange services and advertise new ones. Adopting our QoS-based architecture, each institution can select the best services it needs on the basis of the QoS values estimates and discover malicious provider exploiting the provided reputation information.

System Architecture - Overview

From a logical point of view, the proposed D-SOA can be seen as a two-levels hierarchical network. Top-level subsystem is constituted by a set of *Super Nodes* forming an overlay network and acting as a distributed trusting authority. The low-level implements the service exchange subsystem and its components, called *Nodes*, are service consumers and providers. The main task of the distributed trusting authority is the monitoring of service exchange activity occurring at the low-level and the provision of updated information on both QoS and provider reputation for supporting service selection.

From a physical point of view, the whole network is partitioned into small clusters, called domains, as shown in Figure 4.3. Each cluster is supervised by a Super Node which is responsible for monitoring the activities of all Nodes belonging to its domain. More precisely, Super Nodes are actively involved in the initial service negotiation phase and in the final phase of users' feedback management, whilst the actual service exchanges occur through direct connections among Nodes.

System Architecture - Functional View

Super Nodes overlay network implements the distributed trusting authority by evaluating the reputation of service providers, penalizing malicious nodes and rewarding truthful ones. Each Super Node maintains a reputation value for each Node in its domain and also, in order to maintain links to different domains, a reputation value for any neighbor node in the overlay network. The neighborhood reputation values, owned by Super Nodes, allow to filter QoS information as function of the reputation of the node which provides it. This means that the information coming from reliable sources is associated with higher weights, or in other words, reputation values allow to estimate the degree of truth of QoS information. The result is that service consumers can reliably classify trustworthy and untrustworthy providers, thus to be able to correctly select the best services matching with their requirements of quality and cost. In order to provide a more detailed description of the architecture, the event flow generated by a service request is analyzed and shown in Figure 4.4.

We distinguish four roles:

- *Consumer Node*: the service consumer (`CN` in Figure 4.4);
- *Provider Node*: the service provider (`PN`);
- *Seeker Super Node*: the super node that is the cluster-head of the cluster hosting the consumer node (`SSN`);
- *Manager Super Node*: the super node that is the cluster-head of the cluster hosting the provider node (`MSN`).

When a CN looks for a given service, it sends a query to its SSN (1) that, in turn, forwards it to its Super Node neighbors in the overlay network (2). We assume, without loss of generality, that Super Nodes form a fully connected overlay network. Under this assumption, all Super Nodes in the network can reply to the query. For the sake of simplicity, the assumption also allows us to disregard problems related to the query routing that do not fall within the issues addressed by this work. When a Super Node receives a query, it performs a local search for the services provided by Nodes in its domain (3). Each Node replies to the local query declaring the updated QoS values for the requested service, QoS_{decl} (4). In this phase, these Nodes act as PNs. The Super Node replies to the the SSN with a list of services matching the query, enriched also by QoS information (5). In this phase, the queried Super Node acts as trusting authority for QoS information, thus playing the role of MSN. In its guarantor role, each Super Node has also the capability of modifying QoS values. Hence, the SSN receives from a certain number of MSN lists of services coupled with respective QoS values advertised by

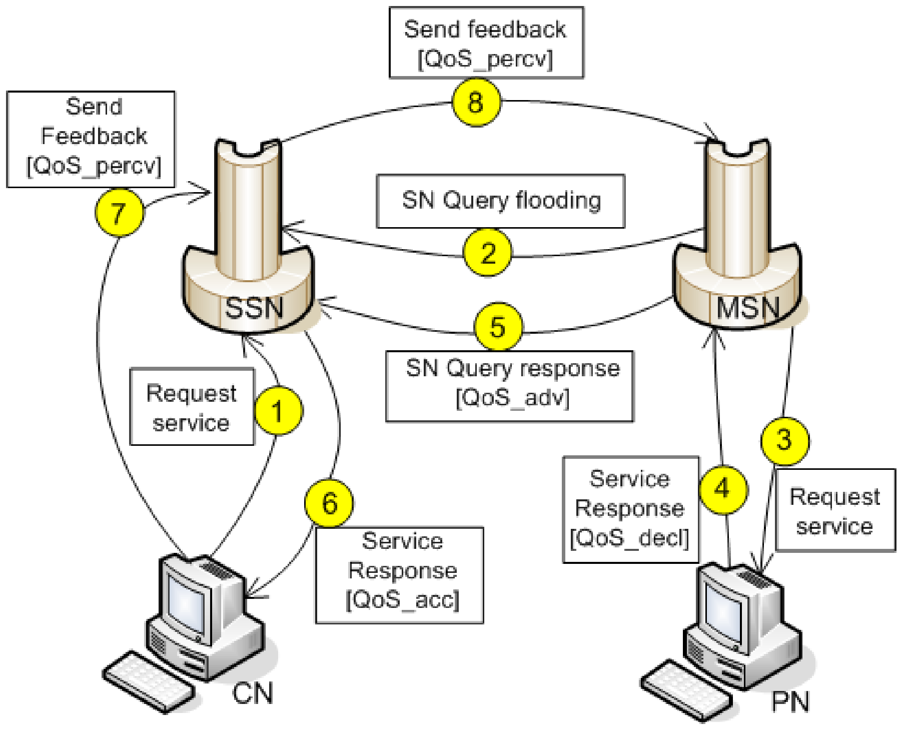

Send feedback [QoS_percv]
8
Send Feedback [QoS_percv]
7
SN Query flooding
2
SSN
MSN
5
SN Query response [QoS_adv]
Request service
1
6
Service Response [QoS_acc]
Service Response [QoS_decl]
4
3
Request service
CN
PN

Figure 4.4: Event flow generated by a service request.

respective MSNs, QoS_{adv}. A comprehensive merged list is then forwarded by SSN to the CN (6). Here, also the SSN has the capability of fixing the received QoS values thus producing the final accepted QoS values, QoS_{acc}. Received the service list, the CN selects a service out from the received ones, and as final step closing the loop it determines a feedback value by estimating the perceived QoS, QoS_{percv}, to be send back to the SSN (7). As a function of the received feedback, the SSN is thus able to update its reputation estimate of the MSN.This update is performed according to a function which takes into account the similarity between advertised QoS and perceived QoS. Finally, the SSN forwards the same feedback value to the MSN (8), in order to enable it to apply the same procedure to recalculate the reputation of the PN in its domain.

4.2.3 QoS and Reputation Management Policy

The system behavior heavily depends both on the policies adopted for the QoS and reputation management and on the service selection methods. For this reason, in order to provide a full specification of the system, we shall provide the description of the following functionalities:

- Service selection performed by CN after receiving the service list (step 6);

- MSN reputation management as performed by SSN after receiving userÕs feedbacks (step 7);
- PN reputation management as performed by the MSN after receiving userÕs feedbacks (step 8).

All the above policies exploit the reinforcement learning mechanism as their driving principle.

The Adopted Reinforcement Learning Model

Reinforcement Learning (RL) (Sutton and Barto, 1998) is a branch of Machine Learning, modeling how agents learn which actions to perform with the aim of maximize a score function, based on the results of past interactions with the environment. The RL model assumes that, after each interaction with the environment, the playing agent obtains a reward for the performed action. Such rewards constitute the input data of a trial-and-error learning mechanism whose goal is the generation of the best situation-action mapping to be considered for maximizing the average reward. In order to select the action to be performed, an optimal trade-off between the *exploitation* of the acquired knowledge and the *exploration* of not-yet-evaluated solutions must be achieved. The former criterion involves that the agent would choose the best action given its current state, whereas the latter implies that the agent would also choose sub-optimal actions in order to explore new outcomes.

We identify the learning agents with the network nodes, and for each agent its environment is represented by all the other nodes. In order to perform the reputation management, QoS estimate, and service selection, we adopted the *Q-learning* (Watkins and Dayan, 1992), among all the proposed RL techniques, because it considerably simplifies the formalization of the learning algorithm and it comes with a formal proof of its early convergence. In such method, the average utility of performing an action a in a state s, referred as $Q(a_t, s_t)$, is updated as a function of the past estimate and of the reward r_{t+1} obtained after the agent-environment interaction, according to the following equation:

$$Q(a_t, s_t) \leftarrow (1-\alpha)Q(a_t, s_t) + \alpha[r_{t+1} + \gamma \underset{a}{max} Q(s_{t+1}, a)], \qquad (4.5)$$

where $Q(a_t, s_t)$ is the current estimate of the utility obtained by performing the action a_t in the state s_t, s_{t+1} is the new state in which the agent transits after the action performance, r_{t+1} is the obtained reward, and $\underset{a}{max} Q(s_{t+1}, a)$ is the maximum reward obtainable in the new state. The α and γ parameters, both ranging in $[0, 1]$, control the learning mechanism, and represent respectively the learning rate and the discount factor. The former determines the weight of new information with respect to the past history, and the latter determines the influence of future rewards. Based on the $Q(a, s)$ values, the agent selects the action to be performed with a technique known as *reinforcement comparison*, according to which, each action can be selected with a probability π directly related to its estimated average reward, computed as follows:

$$\pi_t(a, s) = Pr\{a_t = a | s_t = s\} = \frac{e^{Q(a_t, s_t)/\tau}}{\sum_a e^{Q(a, s_t)/\tau}}. \tag{4.6}$$

Such a selection mostly stresses the choice of the best action thus enabling the exploitation; however, since the probability to select sub-optimal actions is never 0 , it also allows the exploration. High values for the τ parameter, *temperature* in the Boltzmann distribution, make the actions quite equiprobable, while low values make that a small difference in action utility correspond to a big difference in action selection probability.

QoS Filtering and Service Selection

In response to the query for a service, the Consumer Node receives a list of services matching the query parameters. These services are associated to some QoS information, advertised by MSNs and filtered by the SSN. QoS information filtering is performed by the SSN on the basis of its reputation value associated to the MSN that provided such information. If *rep* represents the reputation value of the MSN providing the QoS declaration, rep_{max} for the maximum reputation value of all neighbor Super Nodes, and QoS_{adv} for the QoS value advertised by the MSN, the filtering rule which determines the accepted QoS, QoS_{acc} can be written as:

$$QoS_{acc} = QoS_{adv} * \frac{rep}{rep_{max}}. \tag{4.7}$$

After its filtering activity, the SSN forwards the modified service list to the CN, in order to support it in the selection of best services. Selection is performed through reinforcement comparison method described in equation 4.6, where the selection of a service corresponds to an action, and the action reward corresponds to the QoS of the selected service.

Manager Super Nodes Reputation Management

After a CN uses a service, it replies its SSN, with a feedback value expressing the perceived QoS. The SSN exploits this information in order to update the MSN reputation. This update operation takes into account the gap between QoS filtered and accepted by the SSN, QoS_{acc}, and QoS perceived, QoS_{percv}. In such a phase the SSN may choose among three possible actions: it may increase, decrease, or confirm its MSN reputation. Intuitively, if the accepted QoS is similar enough to the QoS perceived by the CN, the current estimate of the MSN reputation value can be considered correct and then confirmed. Vice versa, if the filtered QoS value do not correspond to the perceived one, it is more appropriate to update the reputation estimate.

In order to select the best action to be performed, an ad-hoc reputation-learning subsystem was designed. The states of the subsystem represent the set of the possible reputation values for the MSNs in the neighborhood of the SSN; the subsystem goal is to learn the utility value of each action in all possible states. In this context, the utility value is function of the similarity between filtered and perceived QoS values.

For each possible action (in short: *incr*, *decr*, *conf*), the SSN evaluates which QoS value would have transferred to the CN, using equation 4.7; for each of these three hypothetical values, the SSN evaluates the difference between perceived QoS, QoS_{percv} and hypothetical filtered QoS, QoS_{acc_hyp}. Finally, this hypothetical error, err_{hyp}, is compared to the actual one, err_{act}, in order to obtain the reward r for all the possible actions, according to the following equations:

$$\begin{aligned} err_{act} &= |QoS_{percv} - QoS_{acc}|, \\ err_{hyp} &= |QoS_{percv} - QoS_{acc\_hyp}|, \\ r &= err_{act} - err_{hyp}. \end{aligned} \tag{4.8}$$

Obviously, the action of confirming the reputation of the MSN has a null reward.

The average utility for all actions is updated using the Q-Learning method as described in Section 4.2.3, as a function of the computed rewards. As regards the current state, represented by the current reputation of the MSN, the rule for updating actions' utility is the following:

$$\begin{aligned} Q(rep_t, incr) &\leftarrow (1-\alpha)Q(rep_t, incr) + \alpha[r_{incr} + \gamma \underset{a}{max} Q(rep_t + 1, a))], \\ Q(rep_t, decr) &\leftarrow (1-\alpha)Q(rep_t, decr) + \alpha[r_{decr} + \gamma \underset{a}{max} Q(rep_t - 1, a)], \\ Q(rep_t, conf) &\leftarrow (1-\alpha)Q(rep_t, conf) + \alpha[r_{conf} + \gamma \underset{a}{max} Q(rep_t, a)]. \end{aligned} \tag{4.9}$$

In summary, when a SSN receives a feedback from a CN, it performs the following actions:

1. Compute the reward values, r_{incr}, r_{decr}, r_{conf} (equation 4.8);
2. Evaluate the effects of possible actions for updating MSN reputation, by computing their utility values, $Q(rep, incr)$, $Q(rep, decr)$, $Q(rep, conf)$, (equation 4.9);

3. Select the action to be performed through the reinforcement comparison method (equation 4.6);

4. Update the reputation of the MSN according to the selected action.

Provider Nodes Reputation Management

The fact that SSN evaluates the reputation of MSN by estimating the reliability of the QoS information they advertise, represents, ultimately, the reason why a MSN that does not correctly certify the reputation of the PN in its own domain may experience a reduction of reputation, since it is not able to discover malicious behaviors. In order to avoid other Super Nodes discredit, each Super Node maintains the reputation values of all the PN belonging to its domain, with the aim to penalize them whenever they declare incorrect QoS values. The information necessary to manage the reputation of the PN is obtained from usersÕ feedbacks that are forwarded by the SSN. Such a management policy mirrors the management policy of MSN reputation carried out by the SSN, and described in Section 4.2.3. Reputation values of the PN are here used by MSN to filter at the origin advertised QoS values, with a mechanism equivalent to that described in Section 4.2.3.

4.2.4 Experimental Results

In order to evaluate the system behavior, we performed a wide set of simulations, through an ad-hoc developed simulator. We report the results of experiments devoted to highlight how the reputation management system motivates MSN to correctly advertise the reputation of their domain PNs, and how PNs are compelled to declare true QoS values.

Advantages from a correct reputation management

First experiment focuses on the usefulness deriving to MSNs from a correct management of the PN reputation. In our setting the simulated network is composed of 150 nodes spread over 11 domains, containing the same number of honest PNs, malicious PNs and CNs. Each SN correctly manages the reputation only for a part of the malicious PNs belonging to its domain, and masquerades for the other part. The experiment consists of 10 simulations of 1000 steps and the results were averaged over all simulations. Figure 4.5 shows that MSNs that correctly manage the reputation of greater percentages of PNs achieve a clear advantage, since they obtain high values of reputation, whereas the reputation of malicious MSNs decreases quickly over the time. Namely, since SSNs must provide the most accurate QoS estimates to CNs, they dramatically reduce the MSN reputation until

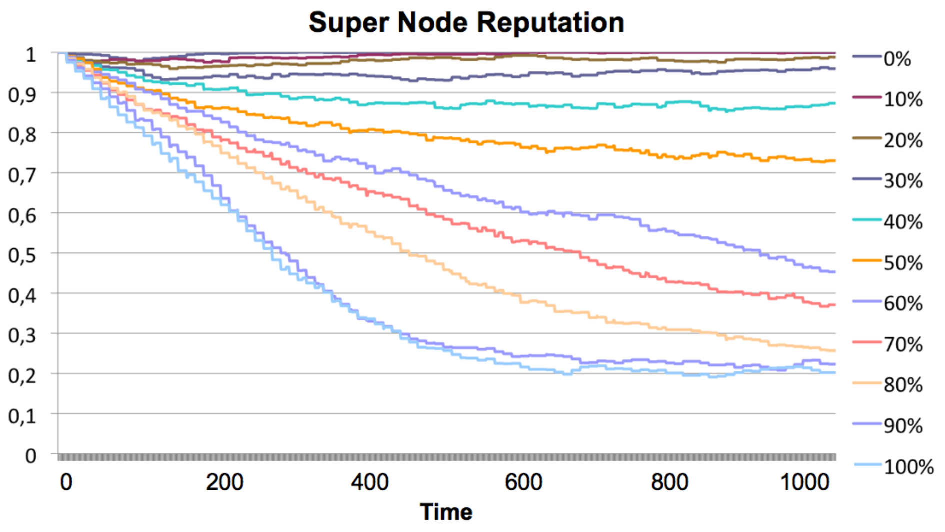


Figure 4.5: Comparison between the reputation values of MNSs that masquerade different percentages of malicious PNs.

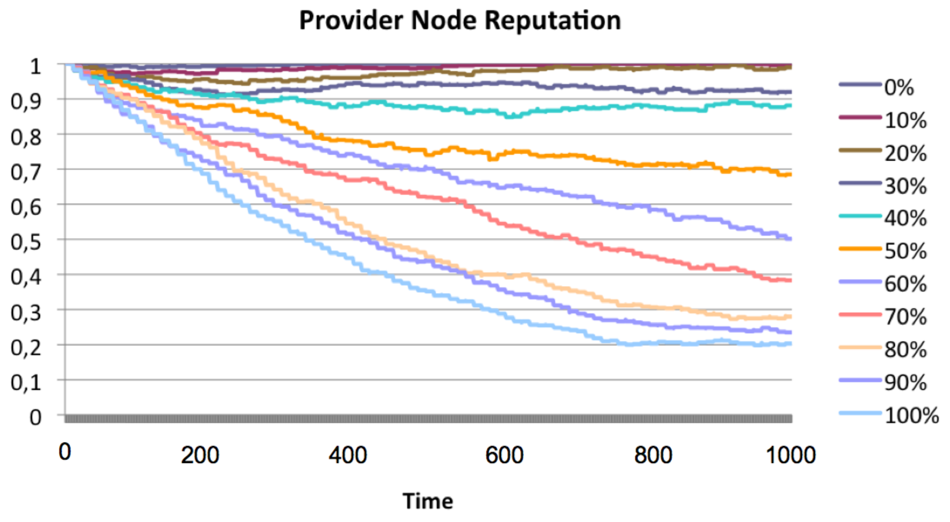


Figure 4.6: Comparison between the reputation values of PNs that provides false QoS values with different probabilities.

the accepted QoS values, filtered according to equation 4.7, match the perceived ones.

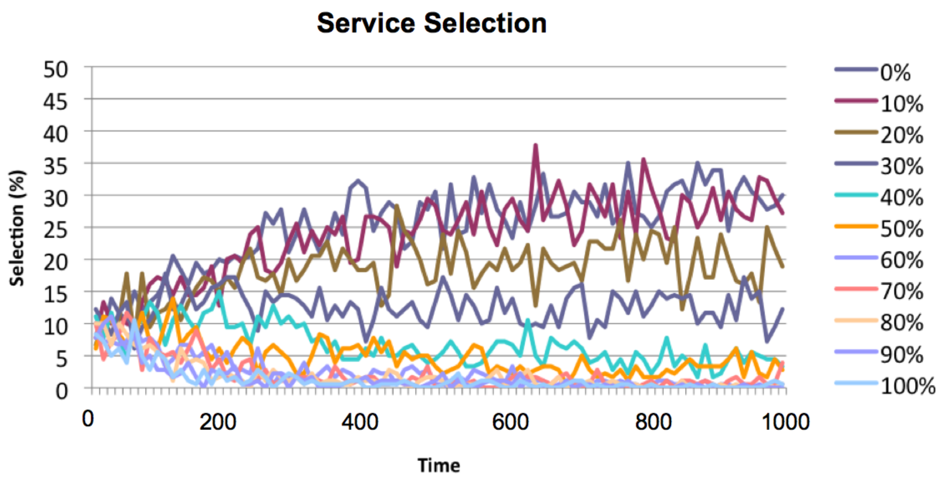


Figure 4.7: Comparison between the average percentage of selected services provided by PNs with different probability of being dishonest.

Detection of Malicious Provider Nodes

In order to effectively detect malicious PNs, a MSN should assign different reputation values to PNs as function of declared QoS values. The second experiment aims to prove that the reputation management policy provides MSNs with this capability. The setting differs from the previous experiment since all MSNs correctly manage the reputation of PNs. Figure 4.6 shows the average value of reputation PNs, aggregated by the probability of declaring false values. The reputation scores of malicious PNs decrease over the time, for the same reason adduced in the previous experiment. This leads to an important conclusion: malicious behaviors are always detected, either by the decrease of the PN reputation in the opinion of its MSN, or by the decrease of MSN reputation when it does not correctly advertise its PN reputations.

The Economic Drawback for Malicious Provider Nodes

A reputation management system able to detect malicious nodes can actually lead PNs to declare true QoS values only if this detection corresponds to some economic drawback for malicious providers. Such a deterrent can originate only from a reduction of sold services. In this experiment, we use the same basic setting of of the previous ones. Results shown in Figure 4.7 prove that, after a transitory phase, during which services provided both by malicious and truthful PNs are chosen with the same percentage, a drastic reduction of the percentage of selected services (that do not reach 5% for PN declaring false values more often than 40%) is determined by the decrease of reputation values for malicious PNs.

Conclusions

This work described the design and implementation of a comprehensive architecture for knowledge management in the context of Ambient Intelligence. A precise representation of the state of the environment can be obtained only through continuous and pervasive monitoring, but this makes the management of the stream of sensory readings very challenging, due to the huge amount of gathered data. The cognitive architecture presented here is an example of a flexible and scalable approach to knowledge extraction from the environment by means of the integration of a pervasive sensory framework and a central intelligent entity capable of symbolic reasoning.

The proposed modular framework relies on Wireless Sensor Networks as a pervasive sensory system. Here, WSNs are not merely used for data sensing and gathering purposes, rather their computational capabilities are effectively exploited in order to perform an initial preprocessing phase that constitutes the preliminary step for the overall reasoning. Besides providing basic information about environmental conditions, WSNs additionally allow to monitor the interaction between the users and their surrounding environment, in order to infer the users' requirements about environmental conditions.

In addition to providing the hardware substrate, our architecture is also equipped with a middleware that allows for decoupling the physical component from higher-level modules devoted to drive the operations of the entire system, thus improving the overall efficiency. The proposed architecture is also enriched with a communication module that supports a reliable interaction with external entities in order to obtain required services.

A proof-of-concept validating the proposed approach was provided by way of a sample application comprising a module for estimating the users' presence in an office environment, and a fuzzy controller for tuning the heating and air conditioning.

Bibliography

E.H.L. Aarts and J.L. Encarnação. *True Visions: The Emergence of Ambient Intelligence*. Springer, 2006.

K. Aberer, M. Hauswirth, and A. Salehi. A middleware for fast and flexible sensor network deployment. In *Proceedings of the 32nd International Conference on Very large data bases*, pages 1199–1202. VLDB Endowment, 2006.

H.S. Ahn and K.H. Ko. Simple pedestrian localization algorithms based on distributed wireless sensor networks. *IEEE Transactions on Industrial Electronics*, 56(10):4296–4302, 2009.

M.J. Akhlaghinia, A. Lotfi, C. Langensiepen, and N. Sherkat. Occupant behaviour prediction in ambient intelligence computing environment. *Journal of Uncertain Systems*, 2(2):85–100, May 2008.

I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communication Magazine*, 40(8):102–114, August 2002.

American Medical Association. Website, 2010. `http://braininfo.rprc.washington.edu`.

F. Amigoni, N. Gatti, C. Pinciroli, and M. Roveri. What planner for ambient intelligence applications? *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 35(1):7–21, 2004. ISSN 1083-4427.

E. Ardizzone, M. La Cascia, and M. Morana. Face Processing on Low-Power Devices. In *Proceedings of the 4th International Conference on Embedded and Multimedia Computing*, pages 1–6, Jeju, Korea, 2009.

L. Atallah and G.Z. Yang. The use of pervasive sensing for behaviour profiling - a survey. *Pervasive and Mobile Computing*, 5:447–464, 2009.

F. Banaei-Kashani, C.C. Chen, and C. Shahabi. WSPDS: Web Services Peer-to-peer Discovery Service. In *Proceedings of the International Conference on Internet Computing*, pages 733–743, 2004.

R. Barr, J. C. Bicket, D. S. Dantas, B. Du, T. W. Kim, B. Zhou, and E. G. Sirer. On the need for system-level support for ad hoc and sensor networks. *ACM SIGOPS Operating Systems Review*, 36(2):1–5, 2002.

C. Bartelmus, P. d'Angelo, H. Langos, T. Wheely, K. Scheibler, J. Paris, P. T. Jochym, and M. Pikula. LIRC - Linux Infrared Remote Control. Available online at `http://www.lirc.org/`, 2002.

T. Basten, L. Benini, A. Chandrakasan, M. Lindwer, J. Liu, R. Min, and F. Zhao. Scaling into ambient intelligence. In *Proc. of Design Automation and Test in Europe (DATE'03)*, pages 76–81, March 2003.

L. Benini, E. Farella, and C. Guiducci. Wireless sensor networks: Enabling technology for ambient intelligence. *Microelectronics Journal*, 37(12):1639–1649, 2006.

M. Botts and A. Robin. OpenGIS sensor model language (SensorML) implementation specification. *OpenGIS Best Practices Paper OGC*, pages 5–86, 2006.

W. Bründl and P. Höppe. Advantages and disadvantages of the urban heat island - an evaluation according to the hygro-thermic effects. *Meteorology and Atmospheric Physics*, 35(1):55–66, 1984.

R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, and I. Brandic. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, 25(6):599–616, 2009.

I. Chatzigiannakis, G. Mylonas, and S. Nikoletseas. 50 ways to build your application: A Survey of Middleware and Systems for Wireless Sensor Networks. In *Proceedings of the 12th IEEE Conference on Emerging Technologies and Factory Automation (ETFA 2007)*, pages 466–473. IEEE, 2008.

A. Chella, M. Frixione, and S. Gaglio. A cognitive architecture for artificial vision, artificial intelligence. *Artificial Intelligence*, 89(12):73–111, 1997.

A. Chella, M. Frixione, and S. Gaglio. Understanding dynamic scenes. *Artificial Intelligence*, 123(1-2):89–132, 2000.

S. Y. Chen, S. F. Song, Li L. X., and J. Shen. Survey on Smart Grid Technology. *Power System Technology*, 33(8):1–7, 2009.

D. J. Cook and S. K. Das. MavHome: Work in progress. *IEEE Pervasive Computing*, 2004.

D.J. Cook and S.K. Das. How smart are our environments? An updated look at the state of the art. *Pervasive and Mobile Computing*, 3(2):53–73, 2007.

D.J. Cook, J.C. Augusto, and V.R. Jakkula. Ambient intelligence: Technologies, applications, and opportunities. *Pervasive and Mobile Computing*, 5(4):277–298, Aug 2009.

N. Costa, A. Pereira, and C. Serodio. Virtual Machines Applied to WSN's: The state-of-the-art and classification. In *Systems and Networks Communications, 2007. ICSNC 2007. Second International Conference on*, page 50. IEEE, 2007.

F. Doctor, H. Hagras, and V. Callaghan. A fuzzy embedded agent-based approach for realizing ambient intelligence in intelligent inhabited environments. *IEEE Transactions on Systems, Man and Cybernetics, Part A*, 35(1):55–65, Jan 2005.

A. Dore, M. Pinasco, and C.S. Regazzoni. A bio-inspired learning approach for the classification of risk zones in a smart space. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '07)*, pages 1–8, June 2007.

K. Ducatel, M. Bogdanowicz, F. Scapolo, and J.-C. Burgelman. *Scenarios for Ambient Intelligence in 2010, Tech. Rep.* Information Soc. Technol., Advisory Group (ISTAG), Inst. Prospective Technol. Studies (IPTS), Seville, Feb 2001.

Epinions. Website, 2010. URL `http://www.epinions.com`.

D. Estrin, L. Girod, G. Pottie, and M. Srivastava. Instrumenting the world with wireless sensor networks. In *Proc. of Int. Conference on Acoustics, Speech, and Signal Processing (ICASSP 2001)*, Salt Lake City, Utah, May 2001.

C. L. Fok, G. C. Roman, and C. Lu. Agilla: A mobile agent middleware for self-adaptive wireless sensor networks. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 4(3):1–26, 2009.

P. Gärdenfors. *Conceptual Spaces The Geometry of Thought.* MIT Press, Cambridge MA, 2000.

S. Goel, T. Imielinski, and A. Passarella. Using buddies to live longer in a boring world. In *Proc. IEEE PerCom Workshop*, volume 422, pages 342–346, Pisa, Italy, 2006.

G. González, C. Angulo, B. López, and J. L. de la Rosa. Smart user models: Modelling the humans in ambient recommender systems. In *Proceedings of the workshop on decentralized, agent based and social approaches to user modelling (DASUM)*, pages 11–20, Edinburgh, Scotland, 2005.

R. Guha, R. Kumar, P. Raghavan, and A. Tomkins. Propagation of trust and distrust. In *13th International World Wide Web Conference*, 2004.

S. Hadim and N. Mohamed. Middleware: Middleware challenges and approaches for wireless sensor networks. *IEEE Distributed Systems Online*, 7(3):1–1, 2006.

H. Hagras, I. Packharn, Y. Vanderstockt, N. McNulty, A. Vadher, and F. Doctor. An intelligent agent based approach for energy management in commercial buildings. In *Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2008)*, pages 156–162. IEEE, 2008.

K. Han, J. Lee, S. Na, and W. You. An ambient robot system based on sensor network: Concept and contents of ubiquitous robotic space. In *Mobile Ubiquitous Computing, Systems, Services and Technologies*, pages 155–159, Nov. 2007a.

S. Han, H. S. Lim, and J. M. Lee. An efficient localization scheme for a differential-driving mobile robot based on rfid system. *IEEE Transactions on Industrial Electronics*, 54(6):3362–3369, 2007b.

S. Helal, W. Mann, H. El-Zabadani, J. King, Y. Kaddoura, and E. Jansen. The Gator Tech Smart House: A programmable pervasive space. *Computer*, 38(3): 50–60, 2005.

A. Holmes, H. Duman, and A. Pounds-Cornish. The iDorm: Gateway to heterogeneous networking environments. In *International ITEA Workshop on Virtual Home Environments, Paderborn, Germany (February 2002)*, pages 1–8, 2002.

P. Höppe. The physiological equivalent temperature – a universal index for the biometeorological assessment of the thermal environment. *International Journal of Biometeorology*, 43(2):71–75, 1999.

S. Jimenez-Fernandez, A. Araujo-Pinto, A. Cobo-Sanchez de Rojas, F. del Pozo-Guerrero, O. Nieto-Taladriz, P. de Toledo-Heras, and J.M. Moya-Fernandez. PERSEIA: a biomedical wsn to support healthcare delivery for the elderly and chronically ill. In *Proc. of IEEE Conf. on Eng. in Medicine and Biology Society (EMBS '06)*, pages 2064 – 2066, New York City, USA, Aug 2006.

B. Johansson, M. Rabi, and M. Johansson. A simple peer-to-peer algorithm for distributed optimization in sensor networks. In *46th IEEE Conference on Decision and Control*, 2007.

R.E. Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *12th International Conference on World Wide Web*, 2003.

E.R. Kandel, J.H Schwartz, and T.M. Jessell. *Essential of Neural Science and Behavior*. Appleton & Lange, New York, 1995.

W. Kastner, M.J. Kofler, and C. Reinisch. Using AI to realize energy efficient yet comfortable smart homes. In *Proceedings of the 8th IEEE International Workshop on Factory Communication Systems (WFCS 2010)*, pages 169–172. IEEE, 2010.

J. A. Kientz, S. N. Patel, B. Jones, E. Price, E. D. Mynatt, and G. D. Abowd. The Georgia Tech Aware Home. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 3675–3680. ACM, 2008.

S. Kontogiannis and P. Spirakis. The contribution of game theory to complex systems. In *10th Panhellenic Conference of Informatics (PCI)*, 2005.

K. Langendoen and N. Reijers. Distributed localization in wireless sensor networks: a quantitative comparison. *Computer Networks*, 43(4):499–518, 2003.

R.-G. Lee, C.-C. Lai, S.-S. Chiang, H.-S. Liu, C.-C. Chen, and G.-Y. Hsieh. Design and implementation of a mobile-care system over wireless sensor network for home healthcare applications. In *Proc. of IEEE Conference on Engineering in Medicine and Biology Society (EMBS '06)*, pages 6004–6007, New York City, USA, Aug 2006.

P. Levis and D. Culler. Maté: A tiny virtual machine for sensor networks. *ACM SIGARCH Computer Architecture News*, 30(5):85–95, 2002.

C. Li, B. Yu, and K. Sycara. An incentive mechanism for message relaying in peer-to-peer discovery. In *Second Workshop on the Economics of Peer-to-Peer Systems*, 2004.

G.-Q. Liu, Z.-L. Zhu, Y.-Q. Li, D.-C. Li, and J.-C. Cui. A New Web Service Model Based On QoS. In *International Symposium on Intelligent Ubiquitous Computing and Education*, 2009.

T. Liu and M. Martonosi. Impala: a middleware system for managing autonomic, parallel sensor systems. In *Proceedings of the ninth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pages 107–118. ACM, 2003.

C.-H. Lu and L.-C. Fu. Robust Location-Aware Activity Recognition Using Wireless Sensor Network in an Attentive Home. *IEEE Transactions on Automation Science and Engineering*, 6(4):598–609, 2009.

S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TinyDB: an acquisitional query processing system for sensor networks. *ACM Transactions on Database Systems (TODS)*, 30(1):122–173, 2005.

L. Marchesotti, S. Piva, and C. Regazzoni. Structured context-analysis techniques in biologically inspired ambient-intelligence systems. *IEEE Trans. on Systems, Man and Cybernetics, Part A*, 35(1):106–120, Jan 2005.

Memsic. TelosB datasheet. Available online at `http://www.memsic.com`, 2010.

D. A. Menasce and V. Dubey. Utility-based QoS Brokering in Service Oriented Architectures. In *IEEE International Conference on Web Services*, pages 422–430, 2007.

M.M. Molla and S.I. Ahamed. A survey of middleware for sensor network and challenges. In *Proceedings of the 2006 International Conference on Parallel Processing (CPP 2006) Workshops*, pages 6–228. IEEE, 2006.

B. Mortazavi and G. Kesidis. Model and simulation study of a peer-to-peer game with a peputation-based incentive mechanism. In *Information Theory and Applications Workshop (IEEE)*, 2006.

M.C. Mozer. The Neural Network House: An Environment hat Adapts to its Inhabitants. In *Proceedings of the Intelligent Environments AAAI Spring Symposium*, 1998.

T.J. Norman, A. Preece, S. Chalmers, N.R. Jennings, M. Luck, V.D. Dang, T.D. Nguyen, V. Deora, J. Shao, and W.A. Gray. Agent-based formation of virtual organisations. *Knowledge-Based Systems*, 17(2-4):103–111, 2004.

S. Park and S. Hashimoto. Autonomous Mobile Robot Navigation Using Passive RFID in Indoor Environment. *IEEE Transactions on Industrial Electronics*, 56 (7):2366–2373, 2009.

J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference.* Morgan Kaufmann, 1988.

S. Ran. A model for web services discovery with QoS. *ACM SIGecom Exchanges*, 4(1):1–10, 2004.

K. Ranganathan, M. Ripeanu, A. Sarin, and I. Foster. To share or not to share: An analysis of incentives to contribute in collaborative file sharing environments. In *Workshop on Economics of Peer-to-Peer Systems*, 2003.

P. Remagnino and G.L. Foresti. Ambient intelligence: A new multidisciplinary paradigm. *IEEE Trans. on Systems, Man, and Cybernetics—Part A: Systems and Humans*, 35(1):1–6, Jan 2005.

M. Ripenau and I. Foster. Mapping the gnutella network: Macroscopic properties of large-scale peer-to-peer systems. In *First International Workshop on Peer-to-Peer Systems*, 2002.

G. Roussos and V. Kostakos. RFId in pervasive computing: State-of-the-art and outlook. *Pervasive and Mobile Computing*, 5(1):110–131, 2009.

S. Saroiu and P.K. Gummadi. A measurement study of peer-to-peer file sharing systems. In *Multimedia Computing and Networking*, 2002.

A. Savvides, H. Park, and M.B. Srivastava. The n-hop multilateration primitive for node localization problems. *Mobile Networks and Applications*, 8(4):443–451, 2003.

S. Songand, K. Hwang, R. Zhou, and Y.K. Kwok. Trusted p2p transactions with fuzzy reputation aggregation. *IEEE Internet Computing*, 9:24–34, nov-dec 2005.

D. Stipanicev, L. Bodrozic, and M. Stula. Environmental intelligence based on advanced sensor networks. In *Workshop on Systems, Signals and Image Processing 2007 (in EURASIP'07)*, pages 209–212, June 2007.

R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction.* The MIT press, 1998.

G. Tononi and G.M. Edelman. Consciousness and complexity. *Science*, 282(5395): 1846–1851, 1998.

J. Vassileva and Y. Wang. A Review on Trust and Reputation for Web Service Selection. In *27th International Conference on Distributed Computing Systems Workshops (ICDCSW'07)*, 2007.

R. Vastamaki, I. Sinkkonen, and C. Leinonen. A behavioural model of temperature controller usage and energy saving. *Personal and Ubiquitous Computing*, 9(4): 250–259, 2005. ISSN 1617-4909.

L.-H. Vu, M. Hauswirth, and K. Aberer. QoS-based Service Selection and Ranking with Trust and Reputation Management. In *On the Move to Meaningful Internet Systems*, pages 466–483, 2005.

J. Wang, R. K. Ghosh, and S. K. Das. A survey on sensor localization. *Journal of Control Theory and Applications*, 8(1):2–11, 2010.

Y. Wang and J. Vassileva. Trust and Reputation Model in Peer-to-Peer Networks. In *IEEE Conference on P2P Computing*, 2003.

C.J. Watkins and P. Dayan. Q-Learning. *Machine Learning*, 8(3):279–292, 1992.

M. Weiser. The computer for the 21st century. *Scientific American*, 272(3):78–89, 1995.

Z. Xu, P. Martin, W. Powley, and F. Zulkernine. Reputation-enhanced QoS-based web services discovery. In *IEEE International Conference on Web Services, 2007. ICWS 2007*, pages 249–256, 2007.

S. Ramon y Cajal. Axial organization of the retina. *Histologie Du Système Nerveux de l'Homme et Des Vertébrés*, 1911. Available online at `http://en.wikipedia.org/wiki/Retina`.