1
Introduction

Ubiquitous computing features the seamless integration of computer systems into the everyday lives of users to provide information and functionalities anytime and anywhere [1]. It involves the interaction of diverse computational devices aiming at providing services to support users in a transparent and continuous way. For this purpose, ubiquitous computing environments encompass different kinds of sensors and mobile devices (e.g., PDAs, notebooks, smartphones), interconnected through a combination of several wireless network technologies.

Compared to traditional distributed systems, ubiquitous computing systems are characterized by an increased dynamism and heterogeneity of devices, applications and services [2]. The underlying ubiquitous computing infrastructures are more complex and bring into the foreground issues such as user mobility, device disconnections, join and leave of devices, heterogeneous networks, as well as the need to integrate the physical environment with the computing infrastructure [3]. A software infrastructure to support the execution of ubiquitous applications has to be flexible and reactive to deal with the unpredictable, heterogeneous and dynamic nature of the computing environments encountered when users move around different locations [4].

In ubiquitous computing environments, a large number of autonomous software entities, i.e., ubiquitous applications and services, work together to transform physical spaces into smart and interactive environments [5]. In such scenarios, new issues are beginning to arise, such as how to enable users to continuously perform their computational tasks while moving through different locations [6], how to enable users to cooperate while located in different places [7], or how to promote ad-hoc cooperation in situations when a group of users sharing common interests unexpectedly meet at a same location and need to engage into collective activities [8].

The approaches and technologies for supporting these new ways of working are still under investigation. Nevertheless, a particularly interesting trend in ubiquitous computing is exploring the Ambient Intelligence (AmI) paradigm, a multidisciplinary approach that aims at the integration of innova-

tive sensing, communication and actuation technologies to create computer-mediated environments that support user activities through specific services of the environment, provisioned with minimal user intervention [9].

A prominent example of an environment enriched with AmI is "smart home", a house equipped to bring advanced services to its users [10]. In such environment, several domestic artifacts and items can be enriched with sensors to gather information about their use and in some cases even to act independently without human intervention [11]. For instance, door locks in a smart home could be capable of identifying persons and permitting hands-free opening, which could be a useful functionality for elderly, children and disabled people, as well as parents entering home with both hands carrying shopping bags and guiding several children along [12].

Another class of applications for AmI focus on providing health monitoring and support to individuals with cognitive or physical impairments, aiming at helping these people to live independently by improving their access to a wide range of services and facilities [13]. For example, in the case of people at early stages of senile dementia — the most frequent case being elderly people suffering from Alzheimer's disease — such a system could be tailored to ensure appropriate care at critical times by monitoring activities, diagnosing risk situations, such as a stroke or heart attack, and advising a carer or doctor in case of need [14].

There has been also a lot of significant work on the use of AmI technologies for facilitating interaction in academic environments, such as universities [15] [16], research centers [17], conference facilities [18] [19], i.e., environments where researchers, teachers and students gather to engage in learning activities or participate in technical meetings and presentations. In such a scenario, for instance, after detecting that all listed key participants in a pre-arranged meeting have already arrived in the respective room, an AmI service could dim the lights, stop the background music and turn on the projector in that room [20].

Essentially, an AmI system should be aware of the presence of persons in the geographical space, perceive their needs and be able to autonomously personalize and make available services that help users to perform their tasks [21]. For that sake, as a fundamental requirement, AmI applications must be capable of automatically responding to dynamic changes in the environment — such as a person entering or leaving a room, a steep increase in the temperature indicated by a sensor or a new device connecting to the local network, for example — with none or minimal human interference. Hence, applications executing in AmI systems are intrinsically context-aware, i.e., they

have to strongly rely on context data collected from sensors embedded into the environment and in the user's devices to trigger adaptations at different levels, such as in the wireless communication links, middleware and application services, or the user interfaces.

AmI requires middleware support for software development — and deployment — capable of integrating large quantities of different devices and sensors and building a programmable and auto-configurable infrastructure [5]. Several projects, e.g. Gaia [18], CoBrA [19], CHIL [22], etc., have developed prototypes of such environments, usually focusing at specific use cases, user tasks or application domains. In general, these middleware systems provide not only services to store, distribute and process context data collected from different sources, but also services for reasoning about context information, which may be based on logical inference using some type of derivation rule mechanism, e.g., first-order logic, descriptive logic, case-based reasoning or fuzzy logic, or others specialized techniques, such as neural networks, Bayesian networks, etc [23].

In AmI systems, as in most ubiquitous systems, reasoning is necessary for several purposes. First, it is required for dealing with the intrinsic imperfection and uncertainty of context data [24]. In this way, reasoning allows to detect possible errors, to estimate missing values, to determine the quality, precision and validity of the context information. It could be used, for instance, for resolving conflicts when different positioning methods inform different locations for a same entity. For example, when a user forgets his GPS-enabled smart phone in a meeting room and logs in to his office's workstation [25], both pieces of information are provided to the middleware. Second, reasoning may also be used for determining higher-level context information, i.e., inferring new, implicit pieces of context information derived from raw context data. For example, reasoning could be used to determine that a seminar has started based on information about the location of a speaker in a conference room and on the volume of noise proceeding from the pulpit area [2]. Third, reasoning is fundamental for triggering actions or adaptations according to specific situations that may be meaningful and relevant to some applications [26]. For example, reasoning could be used to identify that a user is busy when he is in a meeting room with at least one other colleague, his coffee cup and at least one of his colleagues' coffee cups are co-located in the same room and are warm [27].

In most cases, context reasoning in AmI systems is very complex due to the dynamic, imprecise and ambiguous nature of context information and the need to process large volumes of data collected from a large number of context providers. This complexity is increased by the fact that in some conditions rea-

soning needs to be performed in a decentralized way involving several entities of the system [28]. Decentralization takes the form of physical distribution of computing and sensor devices, context providers and consumers, entities responsible for brokering and reasoning, and ambient services, applications and users that may potentially interact.

1.1 Problem Setting

In the design of ubiquitous applications, the *situation abstraction* is regarded as a powerful feature [26]. Situations are defined as particular combinations of aggregated context data [29], which are relevant for triggering actions or adaptations in applications or services [30]. Rules provide a formal model — based on some type of logic — for describing these situations, which hence may be identified by reasoning operations [31]. Moreover, using rules the application developer can define the relevant situations apart from the application code, achieving great flexibility: he may easily modify previously defined rules to adapt applications to different domains or reuse available rules to describe new situations. Therefore, rule-based inference mechanisms are a fundamental requirement for middleware systems that support the development and deployment of AmI services and applications.

Most middleware systems adopt a centralized approach for their reasoning mechanisms. For example, CoBrA [19], CHIL [22] and Semantic Space [32] are middleware systems that aim at the deployment of small scale smart rooms, where a central server collects all context data available in such ambients — provided either by sensors or user's devices — and performs all reasoning operations. In AmI environments, however, these reasoning operations may need to evaluate context data collected from distributed sources and stored in different devices, as usually not all context data is readily available to the reasoners within a ubiquitous system. For reasons ranging from privacy to performance issues, a fully distributed reasoning scheme is regarded as a necessity [33]. In fact, distributed reasoners that work with small partitions of large context data sets, such as in OWL-SF [34], may have a better performance locally. Nevertheless, if the result of the inference depends on context data distributed among different partitions, the communication overhead necessary to produce a combined result may be excessively high, reducing the scalability of the system.

In our work, we model an AmI system considering that there are two main interacting tiers in the reasoning process: the *user side*, comprised by the user's devices and client applications, and the *ambient side*, represented

by the ambient infrastructure and its services, each managing different context information. We assume that, for reasons ranging from privacy to performance, neither side is prone to disclose its context information, and as such, neither side has access to all context data. In this case, context reasoning has to be executed in a decentralized way, involving reasoners at both sides, performing what we define as "cooperative reasoning".

1.2 Goals

The goal of this thesis is to propose a decentralized reasoning approach for performing rule-based reasoning about context data targeting AmI systems, according to the characteristics of our model, i.e., considering that there are two main interacting parties in the reasoning process: the *user side* and the *ambient side*. We assume that each side has access to different context information, which is not shared with the other side. As such, we propose a novel *cooperative reasoning* approach, in which two entities cooperate to perform a split inference of facts involving data distributed between the two tiers. To show the feasibility of this approach, we designed, implemented and evaluated a middleware service supporting decentralized reasoning based on the cooperation of reasoning services located at the each side. The service is expected to be scalable to support a large number of clients, robust to deal with message failures, portable for mobile devices and expressive to represent a wide variety of rules.

1.3 Contributions

The main contributions of this thesis are the following:

- The identification of a trade-off between completely distributed reasoning systems and systems that are capable of evaluating complex rules with variables, offering greater expressiveness.
- The definition of a context model for AmI environments assuming that context data is distributed over two sides, the *user side*, represented by the users and their mobile devices, and the *ambient side*, represented by the fixed computational infrastructure and ambient services.
- The enumeration and discussion of a set of design strategies for a distributed reasoning service tailored for AmI environments that follow the model defined before.

– The formalization of the *cooperative reasoning* operation, in terms of a split inference of facts involving data distributed in two tiers.
– The definition of a complete process — comprising a strategy, a protocol and the corresponding algorithms — to perform the *cooperative reasoning*, i.e., in which two entities cooperate to perform decentralized rule-based reasoning.
– The implementation and evaluation of a prototype middleware service using KAON2 and MoCA's publish/subscribe service for performing cooperative rule-based reasoning.

It is important to emphasize that, despite some works have presented different approaches for distributed reasoning, our two-tier proposal for modeling context data in AmI environments — and the respective process for split inference of facts — are original contributions for this subject.

1.4 Organization

In the next chapter, we present an AmI scenario and explain the fundamental concepts involved in this work: Ambient Intelligence, context-awareness, ontology-based context model, context reasoning and rule-based reasoning, giving clear definitions about all these concepts and examples involving our scenario. In Chapter 3, we describe some research projects related to the topic of this thesis, namely Gaia, OWL-SF, DRAGO, P2P-DR, and P2PIS, explaining how they tackle the distributed reasoning problem. In Chapter 4, we describe our approach for enabling decentralized reasoning, discussing the design strategies for implementing a service capable of executing the approach. In Chapter 5, we define a strategy, protocols and algorithms for decentralized reasoning. In Chapter 6, we explain our strategy by means of a case study. In Chapter 7, we present the Decentralized Reasoning Service (DRS), a prototype middleware service that implements the proposed approach, and a prototype application that exemplifies how this service may be used. In Chapter 8, we describe the functional and performance tests that were made on the Decentralized Reasoning Service (DRS) and evaluate the results. Finally, in Chapter 9, we discuss the contributions and limitations of the proposed approach, presenting also some topics of future work.