9
Conclusion

In this thesis we proposed a decentralized reasoning approach for performing rule-based reasoning about context data targeting AmI systems, according to the characteristics of a specific model, where we considered that there are two main interacting parties, the *user side* and the *ambient side*, and each side has access to different context information, which is not shared with the other side.

The provision of rule-based inference mechanisms is a fundamental requirement of middleware systems that aim at supporting the development and deployment of AmI services and applications, as rules provide a formal model for describing and detecting situations, which are relevant for AmI applications. Moreover, using rules the application developer can define the relevant situations separated from the application code, achieving a higher degree of flexibility: he may easily modify existing rules to adapt applications to different domains or reuse available rules to describe new situations. Furthermore, the use of free variables in rules give even more flexibility in the description of situations, as the developer can refer generically to the elements of a domain, rather than mention them specifically.

The complexity of context reasoning in AmI systems is enhanced by the fact that applications, services, rules and context information may be partially or fully distributed among the different elements involved. Thus in some circumstances a centralized approach may be inefficient and even infeasible. In such environments, distributed reasoning is necessary to address the complexity that arises from the coexistence of different elements that collect, store, process, exchange and reason about context data. Approaches for distributed reasoning that try to overcome this limitation, such as Gaia [18], OWL-SF [34], DRAGO [93], P2P-DR [28] and P2PIS [94], provide solutions that either are not completely distributed, or are not capable of evaluating complex rules with variables, indicating that there must be a trade-off between these features.

As such, we proposed a simplified model for our system, where we considered two main interacting parties in the reasoning process: the *user*

side and the *ambient side*, both comprised by the services, applications and knowledge base that are available at each side, in a two-tier approach. In our model, not all context information is available both at the users' mobile devices and at the ambient infrastructure. Some information may be available only at the *user side*, while some other information may be available only at the *ambient side*.

Based on this model, we propose a strategy in which two entities — a reasoner running on the user side, the *device reasoner*, and another one running on the ambient side, the *ambient reasoner* — interact to infer situations described by rules involving context variables that refer to data collected at both sides, performing what we defined as *cooperative reasoning*. After identifying a set of general design strategies for implementing a distributed reasoning service tailored to the model we proposed, and formalizing the *cooperative reasoning* operation, we defined a complete process — comprising a protocol and the corresponding distributed algorithms — to execute the *cooperative reasoning*. Finally, we implemented the Decentralized Reasoning Service (DRS), a prototype middleware service for performing the *cooperative reasoning process*.

	Type	**Inference**	**Variables**	**Asynchr.**
Gaia	Rule-based	Local	Yes	Yes
OWL-SF	Rule-based	Local	Yes	Yes
DRAGO	Classification	Distributed	No	No
P2P-DR	Rule-based	Distributed	No	No
P2PIS	Rule-based	Distributed	No	No
DRS	Rule-based	Two-tier	Yes	Yes

Table 9.1: Comparison of DRS with the related work.

In Table 9.1, the main features of DRS are compared with those of the related systems discussed in Chapter 3. We can notice that — differently from DRAGO, P2P-DR and P2PIS — DRS is capable of executing rule-based reasoning using variables and providing asynchronous communications (pub/sub). And compared to Gaia and OWL-SF, DRS is capable of executing inference based on context data distributed in a two-tier scenario, i.e, involving the *user side* and the *ambient side*.

In the *cooperative reasoning process*, an important design strategy is to provide asynchronous communication (publish/subscribe). To achieve this goal in the *cooperative interaction*, a *local reasoner* has to constantly update the information forwarded to the *remote reasoner*. For that reason, if there are frequent context changes at the *local reasoner*, the reasoning operation may never converge. Besides that, the frequent update messages exchanged between

the reasoners may cause a significant communication overhead, making this strategy inadequate for reasoning with highly variable context data.

One could be concerned if this data exchange between the *local reasoner* and the *remote reasoner* would compromise the user's privacy. However, as the *local reasoner* forwards no complete RDF tuple to the *remote reasoner* — but only tuples containing ontology individuals representing values of context variables — there is not a knowledge sharing between those reasoners. As such, as far as the local part of the rule is not diclosed to the *remote reasoner*, the *local reasoner* is able to keep privacy about its context data.

Besides DRS, we implemented also the Context Model Service (CMS), a prototype service responsible for collecting context data from context providers available in a specific domain, keeping an updated representation of the assembled data according to a valid context model (an ontology), and providing access to up-to-date context information. Both CMS and DRS were implemented using the KAON2 reasoning API to access ontology data and perform reasoning operations. Since KAON2 was available only for J2SE environment, it was not possible to port our services also to mobile devices that execute only J2ME based applications. Moreover, to be used in real-world AmI scenarios, dealing with the dynamic and heterogeneous characteristics of such environments, these services must be executed on top of a more complex middleware architecture, capable of providing complementary functionalities such as service discovery or support to semantic interoperability.

DRS was submitted to a battery of tests. The functional tests indicated that the service worked as expected. In the performance tests we tried to compare the *cooperative reasoning* with a simulated centralized configuration. When measuring the response times for the cooperative approach, we got values that — in spite of being satisfactory — were greater than the values observed for the centralized approach. This result can be explained by the fact that in our simulation of the centralized configuration we did not account for the communication overhead caused by the interaction with a large number context providers. Corroborating this expectations, the measurements of the communication traffic showed that it was much higher in the centralized configuration than in the *cooperative reasoning*. Besides that, the reasoning service in the cooperative approach presented a smaller memory footprint.

Nevertheless, the present implementation is not scalable, i.e., is not ready for use in scenarios where a huge number of clients request the reasoning service. The scalability was hindered both by the increasing use of memory and communication overhead, when the number of subscriptions grows. In our tests, memory overflow limited by approximately 700 the number of subscriptions

we could simulate, and for more than 300 subscriptions the performance was greatly affected.

9.1 Contributions

Despite the discussed limitations, we believe that this thesis paved the way towards a strategy and protocol for decentralized reasoning and made the following contributions:

1. 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.

2. 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.

3. The enumeration of a set of design strategies for implementing a distributed reasoning service tailored for AmI environments that follow the model defined before.

4. The formalization of the *cooperative reasoning* operation, in terms of a split inference of facts involving data distributed in two tiers.

5. 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 services cooperate to perform decentralized rule-based reasoning.

6. The implementation and evaluation of the Decentralized Reasoning Service (DRS), a prototype middleware service using KAON2 and MoCA's publish/subscribe service for performing the *cooperative reasoning process*.

Moreover, as a minor contribution we detach:

7. The implementation of the Context Model Service (CMS), a prototype service responsible for collecting context data from context providers available in a specific domain, keeping an updated representation of the assembled data according to a valid context model (an ontology), and providing access to up-to-date context information.

In particular, we highlight the fact that not only our two-tier proposal for modeling context data in AmI environments (Contribution 2), but also the corresponding process for performing split inference of facts (Contribution 5) are original contributions for solving the problem of distributed reasoning.

9.2 Why Two-Tier?

There is a question that could easily come to the mind of the reader of this work. "Why adopting a two-tier approach instead of a multi-tier approach for modelling the AmI system?". In principle, the two-tier model is a novel approach that may be extended to fit a multi-tier model, as we discuss in the next section, but this entails several issues that were considered beyond the scope of this work. Moreover, the two-tier approach meets our initial goal of considering the access to ambient services for the perspective of a single user, and not for multi-user ubiquitous applications.

There are other reasons for focusing only on user and ambient side reasoners, instead of investigating a generic multi-tier approach. Even if the ambient infrastructure contains several devices, it is not reasonable to have the inference of rules split among several devices. Instead, the inference should be performed on a central server, because (i) many of the available devices may have severe resource constraints and (ii) usually there is a stable network link interconnecting these devices. Besides, a multi-tier reasoning would have great impact on the stability and scalability of the system. The stability would be affected due to the need of having n parts of a rule simultaneously satisfied. Considering the communication latency between reasoners and the fact that some context values may change frequently, as we discussed in Section 5.1.3, a great number of *Update* messages (Steps S8 and S9 of our process, Section 5.1.2) would be necessarily exchaged among the n peers, having high impact on the stability of the system. As a consequence, also the scalability would be limited by the huge increase in the number of messages exchanged among the reasoners.

9.3 Future Work

We identify several topics that could be possibly tackled in future work. As a first issue, we think that the scalability of the DRS can be considerably improved. In our current prototype, we used the KAON2 API — which is relatively light, when compared to other available implementations — to implement the reasoning functionalities, but we did not try to optimize the

use of memory, keeping large data structures (e.g., a Query object of the KAON2 API) stored in memory, as we focused on improving the execution time performance. However, in our tests we concluded that the time consumed with the reasoning operation was not critical, indicating that the implementation may be revised, prioritizing to reduce the use of memory.

The communication overhead is another factor that affects the scalability of our implementation, as it increases the response time when there is a great number of subscriptions. As such, the protocol may be improved to avoid the constant message exchange between the local an the remote reasoner. Instead of providing updates every time a single context variable changes, the update messages could be sent less frequently, aggregating changes on several context variables over a period of time. Of course this solution will affect the reasoning time of the system.

Some rules — or parts of rules — that are submitted to the reasoning service may be recurrent, i.e., the same or different applications may want to submit these same rules in several occasions. As such, providing persistence mechanisms for recurrent rules and their inferred results — for a specific AmI system and its applications — is a way of improving our service's performance. This mechanism could not only make a faster response time possible, but also extend the system's knowledge by creating a data base containing usual inference rules.

In fact, a thorough discussion about the service's expressiveness, which is currently bounded by the characteristics of the reasoning API, is still a task to be fulfilled. We intend to define the necessary conditions for the rules to be processed by the service to be DL-safe, so as to guarantee the decidability of the decentralized inference.

A direct way to improve our work is by making our protocol more robust so as to deal with communication problems, i.e., loss of messages, disconnections, etc. In our system model we assumed that the communication was reliable, i.e., there would be no loss of messages. As such, we did not include confirmation messages in our protocol, and hence, the loss of a message can cause an inference operation to be discontinued, with no warning being sent to the clients.

Another aspect to be approached in the future is the use of a concrete and real world ubiquitous application to test our service. Deploying our reasoner in such a real world scenario could bring great advantage, indicating problems to be corrected in our implementation or new functionalities to be added. The accomplishment of such task would enable a practical analysis of the non-functional attributes of the service, yielding their improvement and favouring

a discussion about QoS aspects in AmI, which is still an open problem.

In our present work, the implemented service can not be executed in mobile devices with J2ME-compatible virtual machines, such as most smartphones and many PDA's. As the use of these devices is fundamental in ubiquitous systems, porting our implementation to this execution environment is a task with great priority.

In our scenario, privacy was mentioned as the main reason not to have all context information available in a central repository. However, in the present implementation of the cooperative DRS reasoner, even while the *local reasoner* keeps safely its local context data, other context information can be easily obtained from the *remote reasoner* as a result of some particular query. Hence, the addition of access control mechanisms to avoid the disclosure of specific information is a possible improvement to DRS.

Finally, it is important to give a new step towards a more general distributed reasoning scenario. Therefore, we intend to study how the proposed strategy could be extended to allow the inference of rules in scenarios where the context data is divided in more than two tiers.