A Self-Organized Multiagent Approach for Distributed Management of Contextual Data

Gabriella Castelli
DISMI
University of Modena and Reggio Emilia
42100 Reggio Emilia, Italy
gabriella.castelli@unimore.it

Franco Zambonelli
DISMI
University of Modena and Reggio Emilia
42100 Reggio Emilia, Italy
franco.zambonelli@unimore.it

ABSTRACT

Pervasive computing devices are able to generate enormous amounts of data, from which knowledge about situations and facts occurring in the world should be inferred for the use of pervasive services. However accessing and managing effectively such a huge amount of distributed information is challenging for services. In this paper we propose a self-organized approach to autonomously organize distributed contextual data items into sorts of knowledge networks. Knowledge networks are conceived as an alive self-organized layer in charge of managing data, that can facilitate services in extracting useful information out of a large amount of distributed items. We motivate our approach and present the W4 Data Model that we used to represent contextual data. On these basis, we introduce the general idea of W4 Knowledge Networks and we detail the specific bio-inspired approach for self-organized networking of data items. A case study is introduced to clarify the concepts expressed, and experimental results are reported to support our arguments and proposal.

Categories and Subject Descriptors

J.9 [**Mobile Applications**]: Pervasive Computing; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent Systems*

Keywords

Pervasive Computing, Self-Organization, Context-Awareness, Knowledge Engineering

1. INTRODUCTION

Pervasive and mobile computing scenarios consider the possibility of providing users with ubiquitous and on the move access to digital services, and of supporting users interactions with their surrounding environments. For this possibility to become practical, services should be able to understand situations occurring in the surrounding physical context and autonomously adapt their behavior to the context from which they are requested. This requires both the technology to capture contextual data and the capability of services to exploit such data at the best [10, 7] .

Pervasive devices are already able to generate an overwhelming amount of data, from which knowledge about situations and facts occurring in the world should be inferred for the use of pervasive services. Accordingly, the real challenge for future pervasive services is the investigation of principles, algorithms, and tools, via which this growing amount of distributed information can be properly represented, organized, aggregated, and made more meaningful, so as to facilitate the successful exploitation by pervasive services [16].

Traditional (e.g., centralized and/or deterministic) approaches to data organization and aggregation are not practical in this scenario because of the enormous amount of generated data and the decentralized, dynamic, and unpredictable nature of pervasive systems. Accordingly, in this paper, we propose a self-organized approach to organize, link, and aggregated, related items of contextual information. In the proposed approach a multitude of simple agents, each in charge of a specific relationship among data, continuously analyze and scan the data space in order to link together isolated pieces of data. The overall result is the self-organization of the data space in networks of linked data, called Knowledge networks. Such knowledge networks can eventually be browsed by other agents that are in charge of inferring new knowledge from the existing one. We show that knowledge networks can be easily accessed by services that are in need of contextual knowledge, and that accessing such an organized and distributed data space is beneficial for services because knowledge management costs and knowledge access costs are reduced. We also show with evaluation results that knowledge networks may be beneficial for services that are in need of solving complex queries over contextual data.

The remainder of this paper is organized as follow: Section 2 motivates the paper and present the scenario that will be used in the following of the paper. Section 3 presents the W4 Contextual Model that is used as underlying data model, Section 4 introduce the Knowledge Networks approach and Section 5 discuss evaluation results. Section 6 presents Related Work and finally Section 7 concludes.

2. MOTIVATIONS AND SCENARIO

Ubiquitous computing considers the possibility for users to access general digital services from everywhere and on the move. Pervasive computing additionally considers exploiting pervasive networks, made up of both sensing and data-consumer devices, and possibly actuating infrastructures for the provisioning of innovative services for on-line monitoring of surrounding world and interacting with it, as well as services for enhancing our social experiences in an environment by enabling novel models of localized social interactions. In both cases, it is clear that pervasive services have to collect information about situations around and acting accordingly, i.e. they should be context-aware. Moreover, given the intrinsic dynamics and decentralization of pervasive scenarios, autonomic behavior is necessary to ensure servicesŠ continuity without forcing costly and hard to be managed human intervention.

A number of technologies that contribute producing large amounts of contextual information already exists. The produced items of contextual information (i.e., "data atom"), contribute populating a large cloud of data atoms and at making it available to services (see Fig. 1). A service in need of understanding what is happening around can access (i.e., internalize) the needed data atoms and analyze them to understand what is the current situation of its context. Unfortunately, such description is far too simplistic and does not emphasize a number of complexities inherent in it: the process of data internalization can lead to high communication and computational costs for a service, in that it may require accessing large amounts of data atoms, and the process of analyzing retrieved contextual data atoms and turning them into useful knowledge may be non-trivial.

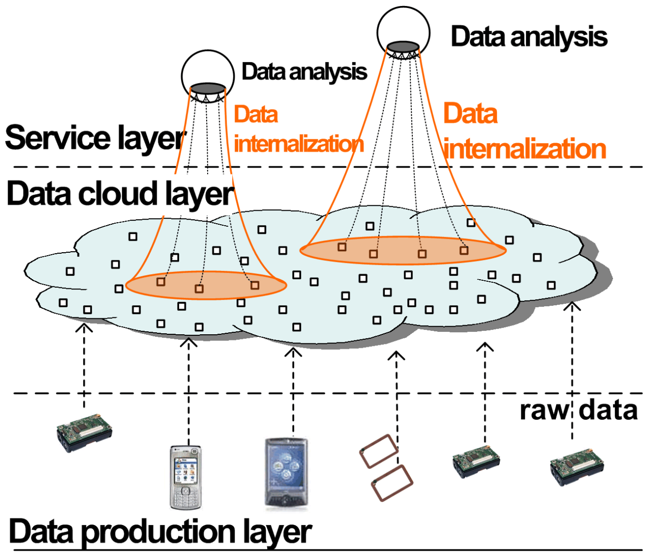

Figure 1: Pervasive devices and sensors make available to services a sort of "data cloud layer", fed with large amounts of heterogeneous data atoms. their own goals.

In this paper we claim that there must be an evolution from a model of simple context-awareness, in which services access isolated pieces of contextual data and are directly in charge of digesting them, towards a model of situation-awareness, in which services access properly structured and organized information reflecting comprehensive knowledge that is related to a "situation" of interest. With reference to Figure 2, we envision that the access by services to contextual information does no longer occur directly, but rather via a "knowledge network" layer. Such layer should encapsulate mechanisms and tools to analyze and self-organize contextual information into sorts of structured collections of related data items, i.e. knowledge networks. From the software engineering viewpoint, an approach based on knowledge networks has the advantage of providing a clear separation of concerns between data analysis and data exploitation. While data analysis and organization is delegated to the knowledge network layer, services are left with the only duty of exploiting such data to reach specific functionalities.

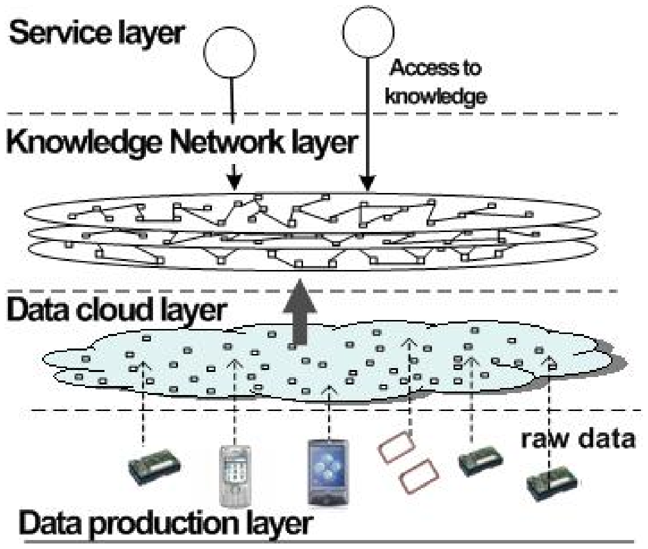

Figure 2: By exploiting a knowledge network layer, services are no longer forced to access the raw data cloud layer. Knowledge organization and analysis is externalized in the middleware, and services are given access to pre-digested information, with a notable complexity reduction.

University campuses are excellent test-bed scenarios to large scale pervasive systems. They are densely populated by individuals carrying a variety of pervasive devices. Moreover, existing network and security infrastructures facilitate people to be always connected and able to get additional information about what is happening in their surroundings. Such information can be fruitfully exploited by context-aware services as the ones discussed above. The illustrated scenario will be used as a running example in later sections of this paper, however we point out that this case study is effective for generalization and the related issues are recurrent in every pervasive system that deals with huge amounts of data.

Consider the scenario in which Gabriella, a university student, has a Wi-Fi enabled PDA that collects information about about herself, share it with other users on campus, and can also access other usersŠ information. The campus infrastructures themselves can collect data about Gabriella and make it available. In general, the user exploits classical contextual services and takes advantages of the knowledge management and generation features in the system to retrieve the desired information. In fact, the system itself can continuously analyze data coming from pervasive devices in the attempt to infer and extract new knowledge, and inject it again in the system. Just thinking to the tens of thousand students that a medium campus may have, and understanding that data is also coming form pervasive sensor networks densely distributed in the campus, it is quite clear that the overall system may lead to the creation of an overwhelming amount of data and inferred knowledge. Therefore, tools for knowledge generation from raw data and data organization are necessary to make the system usable.

For the knowledge networks to be attainable and become a useful tool, both in the case study and in general pervasive scenarios, a number of challenges have to be faced. In particular:

- *Data Model.* There is the need for a simple, general-purpose and uniform model to represent contextual information as individual data atoms as well as their aggregates.
- *Access to data.* It is necessary to identify a suitable API by which services can be given access to the knowledge network layer and the information within.
- *General approaches for data aggregation.* The knowledge networks should be a Şlive layerŤ continuously and autonomously analyzing information to aggregate data atoms, relate existing knowledge atoms with each other, and extract meaningful knowledge from the available data.
- *Application specific views.* Specific services may require the dynamic instantiation within the knowledge networks of application-specific algorithms for knowledge analysis.

In the following Section the W4 Data Model and the related API will be introduced.

3. THE W4 APPROACH

Our proposal for a novel, simple yet effective, data model for expressing contextual knowledge about the world starts from the consideration that any elementary data atoms as well as any higher-level piece of contextual knowledge, in the end, represents a “fact” which has occurred. Such facts can be expressed by means of a simple yet expressive four-fields tuples *Who, What, Where, When*: “someone or something (Who) does/did some activity (What) in a certain place (Where) at a specific time (When)”.

Their four-fields structure is flexible and general enough to uniformly deal with information coming from diverse sources and can account for adaptation to context and incomplete information (i.e., some of the four fields being unspecified).

In addition, the simple W4 structure supports general distributed algorithms for data aggregation and manipulation, and facilitates the building of semantic knowledge networks and of multiple, application-specific views as it will be described in Section 4.

3.1 Data Representation

The four-fields (*Who*, *What*, *Where*, *When*) of the W4 data model each describes a different aspect of a contextual fact.

- The *Who* field associates a subject to a fact. The *Who* field is represented by a type-value pair, in the form of a string, with an associated namespace that defines the *type* of the entity that is represented. E.g., *student: Patricia*.
- The *What* field describes the activity performed by the subject. This field is represented as a string containing a predicate:complement statement. For example, valid entries for the *What* field are: *attending:Computer Foundation class*, *read:temperature = 23*.
- The *Where* field associates a location to the fact. In our model the location may be a physical point represented by its coordinates (longitude, latitude), a geographic region (described as a bounding box), or it can also be a logical place. In addition, context-dependent spatial expressions can be used for context-aware querying.
- The *When* field associates a time or a time range to a fact. This may be an exact time/time range (e.g., *2008/ 07/19:09.00 am–2008/07/19:10.00 am*) or a context-dependent expression.

The way it structures and organizes information makes the W4 data model general enough to represent data coming from very heterogeneous sources and simple enough to promote ease of management and processing (although we are perfectly aware that it cannot capture each and every aspect of context, as freshness of data, reliability, access control, etc).

3.2 Data Access

It is fundamental to define a simple API for services to access to contextual knowledge and enabling data sources and services to inject new data in the knowledge network layer.

Knowledge atoms are stored in the form of W4 tuples in a shared data space (or in multiple data spaces), we took inspiration from tuple-space approaches [1] to define the following API:

```
void inject(KnowledgeAtom a);
KnowledgeAtom[] read(KnowledgeAtom a);
```

The inject operation is equivalent to a tuple space ŚŚoutŠŠ operation: an agent accesses the shared data space to store a W4 tuple there.

The read operation is used to retrieve tuples from the data space via querying. A query is represented in its turn as a W4 tuple with some unspecified or only partly specified values (i.e., a template tuple). Upon invocation, the read operation triggers a pattern-matching procedure between the template and the W4 tuples that already populate the data space. In any case, pattern-matching operations work rather differently from the traditional tuple space model and may exploit differentiated mechanisms for the various W4 fields.

3.3 Data Generation

In the W4 model, we rely on the reasonable assumption that software drivers (or, more in general, software agents) are associated with data sources and are in charge of creating W4 tuples and inserting them in some sorts of shared data spaces. In the end, any data source must be somehow associated with some software agent/driver to gather and store data items, W4 agents have the additional goal of collecting all the necessary information to produce a W4 tuple which is as accurate and complete as possible. This occurs by sensing and inferring information from all the devices and sources available (e.g., RFID tags, GPS devices, Web services), and by combining them in a W4 tuple.

The following simple examples may clarify this concept. Let us assume Gabriella is walking in the campus park. Agents running on her GPS-equipped PDA, can periodically create the following tuple:

```
WHO     user:Gabriella
WHAT    walk:4km/h
WHERE   lonY, latX
WHEN    2006/10/17:10.59am
```

Where the *who* is entered implicitly by the user at the login, *what* and *where* can be derived by the GPS (e.g., the speed of Gabriella as measured by the GPS can be used to deduce that she is walking), *when* can be provided both by the PDA or by the GPS. Viewing this from a different, more fine-grained perspective, we can imagine that one agent controlling the user profile can create a raw W4 tuple in which only the who and where are specified; another agent controlling the GPS agent create a tuple in which only where and what (i.e., the speed) are specified. Accordingly, the merging of these two raw W4 atoms into the complete one represented below can be considered as an action of "knowledge networking" that produces a more complete and expressive information. Merging and inferencing from W4 data atoms in order to generate higher-level Knowledge is the general idea that is behind the concept Knowledge networks, indeed this data generation process is generalized in the knowledge networks approach described in Section 4. It as to be point out that both W4 data atom describing raw data and W4 atom describing high-level knowledge generated by some reasoning process (as result of the self-organizing process we describe in Section 4) are accessed in the same way as knowledge networking processes are completely transparent to services that exploit such data.

4. THE W4 KNOWLEDGE NETWORKS

Although pattern-matching techniques proved rather flexible to retrieve context information, our idea is to exploit the W4 structure to access the context repository in a more sophisticated and flexible way. More specifically, we propose general-purpose mechanisms and policies to link together knowledge atoms, and thus form W4 knowledge networks in which it will be possible to navigate from a W4 tuple to the others. Our idea is that, by querying such a knowledge network instead of a flat W4 tuple space, one can obtain much higher knowledge, as resulting from the analysis, manipulation and inference upon the link structure of the knowledge network.

In particular, new information could be produced by navigating the knowledge network and combining and aggregating existing information into new knowledge atoms likewise to the knowledge generation process introduced in Section 3.3. Such new knowledge could also arise from the analysis of the historical context (e.g., the location where a person spends 8h every day could be his workplace) or from a wide analysis over the wholeW4 repository (e.g., If nobody go to work on 2007/12/25, it could be an holiday).

4.1 W4 Relations

The W4 knowledge networks approach is based on the consideration that a relationship between knowledge atoms can be detected by a relationship (a pattern-matching) between the information contained in the atoms fields. In particular, for the W4 model, we can identify two types of pattern matching correlations between knowledge atoms:

- *Same value – same field*: We can link together W4 tuples belonging to the same user, about the same place, activity or time (or, more in general, those W4 tuples in which the values in the same field match according to some pattern-matching function). Matching two or more *same value – same field* relationships, we can render complex concepts related to groups of W4 tuples, e.g. *All students (same subject) who are attending a class (same activity) at the same room (same location)*.g
- *Same value – different field*: We can link atoms in which the same information appears in different fields. This kind of pattern matching can be used for augmenting the expressive level of the information contained in the W4 tuples. For example, a knowledge atom having *When: 18/09/2008* can be linked with another atom like *Who: Fall Class Begin* , *When: 18/09/2008* to add semantic information to that date.

Table 1 summarizes the basic relationships between knowledge atoms. On the principal diagonal, it is represented the "same value-same field" pattern matching. By reading the table by columns, it is possible to find all relationships between one particular atom with all other atoms in a knowledge network. For example, looking at the first column on the left, we are comparing all atoms with the same subject. The first cell is on the diagonal, so it is a "same value-same field" pattern matching. The 2nd row, 1st column cell identifies all atoms containing the different activities performed by the same subject. Then we have all atoms containing the different locations where the same subject has been, the last cell is a particular case: all atoms generated for the same user.

This network of correlation between atoms may be used as the basis for more elaborated inference and reasoning upon knowledge network, i.e., for identifying and creating links between W4 atoms and for eventually creating new W4 knowledge atoms.

The following example illustrates the the process of discovering new knowledge. Let's suppose that Gabriella's PDA at a certain time generates the following tuple:

```
WHO     user:Gabriella
```

	who	**what**	**where**	**when**
who	Same subject	All subject who performed a particular activity	Atom describing an indoor location	Atom describing a logical time
what	Different activity performed by the same subject	Same activity	All activities performed in the same location	All activity performed at the same time
where	All locations in which a subject has been	All location in which an activity has been performed	Same location	All location occupied at the same time
when	Same subject-different time: a living diary	All times in which an activity has been performed	All times in which an activity has been performed	Same time

Table 1: Relations between the fields of the W4 Knowledge atoms.

```
WHAT    *
WHERE   room:room_35
WHEN    2009/10/17 10.05am
```

Algorithms in the system continuously analyze the data spaces, find a lot of correlation and organize them in multiple knowledge networks. For instance there is an agent that is in charge of building a knowledge network about Gabriella, i.e. a knowledge network whose W4 representation is the following one:

```
WHO     user:Gabriella
WHAT    *
WHERE   *
WHEN    *
```

This specific knowledge network can be considered as a knowledge view, the view about all information generated by the user Gabriella. Another agent is in charge of building a knowledge network about room 35, i.e. a knowledge network whose W4 representation is the following one:

```
WHO     *
WHAT    *
WHERE   room:room35
WHEN    *
```

This represents the knowledge view related to the room 35. The previous two knowledge views intersects whenever a tuple generated by user Gabriella and placed in room 35 is retrieved.

Let's now assume an agent finds a correlation with an atom describing what is happening in the current time in room 35, where Gabriella is at the moment. A new atom carrying higher-level logical information may be created, such atom states that Gabriella is attending the Computer Science class (see Figure 3).

This example gathers that two distinct processes are required in order to have the knowledge networks working: a knowledge linking process, in which knowledge views are built and maintained, and a knowledge generation process which exploits the knowledge views to generate new W4 atoms carrying higher-level information. In the next Section, the self-organized approach that drives the generation of the knowledge networks is presented.

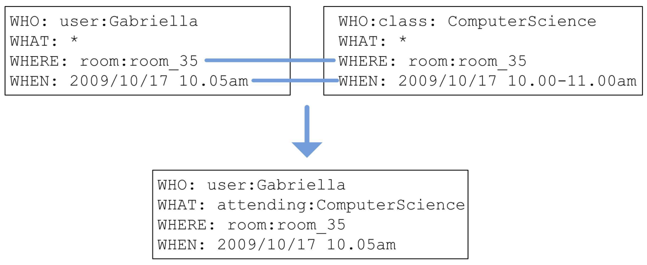

Figure 3: W4 knowledge network data inference

4.2 The Self-Organizing Approach

A self-organizing approach to generate and maintain the knowledge networks' layer is clearly required by the decentralized nature of pervasive computing systems and the overwhelming amount of generated data, which prevent the use of a centralized process for data management. The proposed approach is based on self-organizing multi-agents system to adaptively build and maintain the knowledge networks.

To this end, we adopt an algorithmic approach which relies on a two-phase process: a knowledge linking generation phase in which related knowledge atoms are linked together, and a knowledge generation phase in which new w4 atoms are generated.

The first phase is the identification of all possible correlations between knowledge atoms (according to Table 1), and the creation of link between W4 atoms. This can be done by a number of agents (we call spider as they weave their webs between correlated tuples), in charge of identifying relationships between tuples. Each spider is responsible for a single relationship over W fields (E.g., the agent A1 is in charge of linking together all the tuples with the *who* field equals to "user:Gabriella", the agent A100 is in charge of linking together all the tuples in which the *where* field fulfill the relation "in building:CS_building", etc.). Spiders continuously surf W4 Tuple Spaces in order to retrieve tuples that fulfill the specific relationship, those tuples are virtually linked together thus creating a W4 knowledge network for the given relationship. Obviously spiders must be capable of analyzing W4 Tuples stored in different tuple spaces and building correlation networks that extends over distributed tuple spaces. The spiders' algorithm follows:

```
define:
    spider_rel;
    knet;

Main:
    Do forever:
        TS = choose_Random_TS();
        last_Knet_Ref = Analize_TS(TS, last_Knet_Ref);
    Done;

Analize_TS(current_TS, last_Knet_Ref){
    tuples_inRel[] = current_TS.read[spider_rel];
    /*non destructive read
    if (tuples_inRel NOT null){
        Add_to_Knet (knet, tuples_inRel);
        return current_TS;
        }
    else
        return last_Knet_Ref;
}
```

The spider chooses a random Tuple Space and checks if any tuple in the Tuple Space fulfills the *spider_rel* relationship. If it is positive the tuples are added to the knowledge network *knet* by adding a reference to the last Tuple Space that was found earlier, i.e. drawing a link between the last TS added to the knowledge network *spider_rel* and the current one. This process continuously repeats. In this way, a single knowledge network of links between correlated tuples is generated. More spiders can work concurrently, building the knowledge networks layer in a self-organizing fashion.

The second step is the generation of new knowledge atoms, by analyzing which of the identified link can lead to a new W4 atom as a process of merging related atoms. This is performed by another type of agents, called browser because they are capable of browsing the knowledge networks trying to generate new W4 atoms. Each browser is capable of inferencing a specific type of relationship. The browsers' algorithm follows:

```
define:
    browser_rel;
    /* the relation that the browser is cable to infer

Main:
    Do forever:
        TS = choose_a_random_TS;
        t = choose_a_ramdom_tuple (TS);
        Generate_New_Knowledge(t);
    Done;

Generate_New_Knowledge(t){
    For each (Knet(t)):
        ti = get_Next_Tuple (Kneti, t);
        if (isInfereable (t, ti) IS true){
        new_tuple = inference (t, ti);
        add (new_tuple);
        }
}
```

The browser chooses a random tuple t in the system, and locates all the knowledge networks in which the tuple t is involved. Then the browser start to browse each of the found knowledge networks. For each tuple $\hat{\text{ti}}$ found in a related knowledge networks, the browser checks if he is able of generate a new w4 atom carrying higher knowledge. If positive, the new atom is generated and added to the current tuple space. This generation process is not a trivial issue as the W4 Knowledge Network approachs can produce an overwhelming amount of inferred data, as discussed in [5], [6]. Indeed in [6] we proposed a self-organizing approach to let the system self-organize the knowledge generation in response to knowledge queries being made.

Nevertheless, even when new data are not generated, the Knowledge networks of links between atoms can be fruitfully used during the query resolution phase in order to retrieve tuples more efficiently. When a query is submitted to the W4 Tuple Space System, a query-solving agent capable of browsing knowledge networks, i.e. a browser, analyze the query template and determine one or more knowledge networks to which the matching tuples should belong. Then the browser agent choose a random W4 tuple space in the System and scans it until he finds an entry point for one of the identified knowledge networks, i.e. a tuple belonging to one of those knowledge networks. When the entry point is found, the agent starts to jump from the entry point tuple to the other tuples in the identified knowledge network, checking if they matches the template and finally returns the retrieved tuples. This is beneficial for services because fewer read operations have to be performed when exploiting knowledge networks instead of a flat data space.

In the following Section we discuss simulation results that validate the Knowledge Networks approach when used to answer complex queries.

5. EVALUATION

In order to evaluate the proposed approach, we conducted some experiments to determine how the services improve their data access costs exploiting the w4 knowledge networks infrastructure instead of accessing isolated pieces of information.

We considered the scenario introduced in Section 2 and simulated a distributed campus where W4 tuples are generated by users and inserted in multiple tuple spaces. While the tuple generation is simulated, a prototype of the w4 systems has been realized, there a multitude of spiders explore the systems building networks of correlations as described in Section 4, while browsers are in charge of browsing the correlations' networks to answer queries submitted to the system.

We developed a W4 tuple space implementation on top of LighTS Tuple Space [2], a lightweight tuple space framework highly customizable and particularly suitable for context-aware operation over data. W4 tuple spaces contain W4 Tuples formatted accordingly to the W4 Data Model and implement the various context-aware matching operations already introduced. The W4 tuple spaces also include the implementation of spiders and browsers able to build and exploit the w4 knowledge networks.

Data stored in the W4 tuple spaces' system come from the simulated environment. We built a virtual environment based on the Repast framework [http://repast.sourceforge.net/], an agent based simulation toolkit allowing flexible control over model parameters. We represented a location with 2D coordinates and a number of users each moving in the environment. At the beginning of the simulation the position of each user is randomly determined, a random destination is also set for each user in order to make the simulation more realistic. Periodically a W4 tuple for each user is generated based on the current position and the current time (the what field is randomly chosen in a pool of pre-defined activities). The virtual environment is split in 100 zones, each of this zone holds a private W4 tuple space that stores all the tuples generated in its reference zone. When a user moves between the zones in the virtual environment, it always knows its position and the corresponding reference zone in this way generated tuples are injected in the correct tuple space.

In this scenario many tuples are stored in the w4 tuple spaces, and services many find hard to access those data. We performed some experiments to measure how services may have benefits by accessing w4 knowledge networks. We submitted a complex query to the w4 system, and made it solved by browser agents that are capable of browsing w4 knowledge networks as explained in Section 4.

We submitted to the system the following complex query: "Retrieve all the users that were near agent A5 was, on time 500". For a W4 System this means the following two queries should be subsequently resolved:

```
WHO     user:A5
WHAT    *
WHERE   *
WHEN    500
```

Please note that the *when* field is automatically transformed in a time interval. Let's now suppose we find that agent A5 is in "125, 300" position, so the second query looks as follow:

```
WHO     *
WHAT    *
WHERE   125, 300
WHEN    500
```

Here both the *where* is automatically considered as a bounding box and the *when* field is automatically transformed in a time interval.

On this simulated environment, we compared the W4 Tuple Space System with the Exhaustive Search in Tuple Spaces and with Hash based Tuple Spaces based on the performance of the above system when the non destructive query that we have already introduced is submitted to the system.

The Exhaustive Search is performed on a Tuple Space that embeds the W4 Data Model facilities but not the W4 knowledge networks mechanisms. When a query is submitted to this simplified W4 tuple space system, a query agent chose a random tuple space in the system and scans it seeking for the w4 tuples that fulfill the query template. Then a random Tuple Space is chosen again, until the whole system is scanned.

Hash based Tuple Space is a well known and popular technique for data indexing in distributed environment. Here we follow an approach similar to [13] in which a single field of the tuple structure is used for the hashing operation and indexing purpose. When a tuple is injected in the distributed system, the hashing operation is performed over the designated field and the tuple is then stored on the resulting tuple space. Then when a query is submitted to the system, the same hashing operation is performed on the query template and the result indicates the tuple space to scan for results. However in the W4 Data Model one or more fields of a w4 tuple may be left empty, if the information is not known. In the case the field to be used for the hashing operation is empty, the tuple will be stored in a specific tuple space. In this simulations we considered two cases: hash performed on the who fields and hash performed on the where fields. In the first case, the hash function works on the who field considering the type:value string. In the latter case the hashing function works as a proximity function and the current coordinates submitted are aggregated in 100 zones.

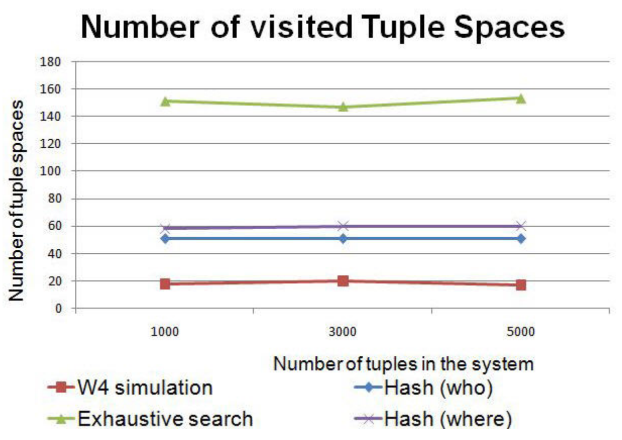

Figure 4: Number of view operations done by query-solving agents.

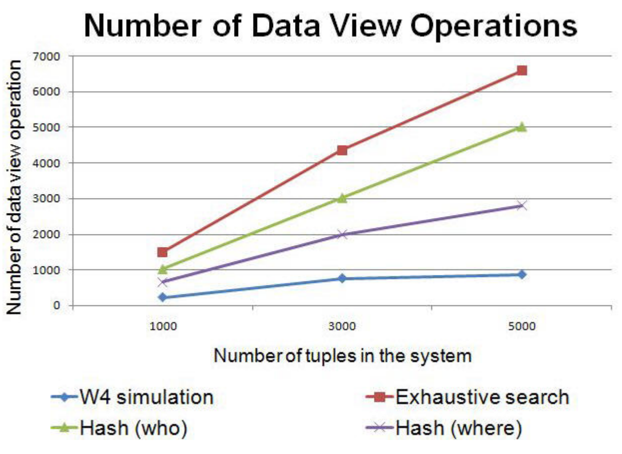

Figure 5: Number of view operations done by query agents

We run the simulations 10 times and depicted the average values. Figure 4 shows the number of tuple spaces visited by the query-solving agent in the considered systems. The W4 Tuple Space Systems performs further better the the Exhaustive Search and better enough than the Hash based System. Indeed, in the medium case, the Exhaustive Search has to query half the number of Tuple Spaces in the system to solve the first sub-query and the whole number of the Tuple Spaces to solve the second ones. The Hash approaches work better than the Exhaustive query because one of the sub-query is solved thanks to the hashing operation, nevertheless the other sub-query have to be solved traditional as in the case of the Exhaustive Search. However exploiting the W4 Knowledge Networks is even better because, one the browser finds the knowledge networks that should be browsed in order to solve the query, the number of tuple spaces accessed is determined by the number of tuple spaces involved in the knowledge networks of interest.

Figure 5 shows the number of read operation performed. Also in this case the W4 Tuple Space System performs better then the other systems. As in the previous case, the Exhaustive Search have to access half the number of tuples in the systems ro solve the first sub-query, and all the tuples in the system to solve the second sub-query. Here we can see that the performance in the Hash based systems are significantly different depending if the hashing is performed on the who field or on the where field. The W4 System performs however better because all the fields are considered important the same when building knowledge networks.

Experimental results show that accessing pre-organized W4 Knowledge Networks instead of a flat Tuple Space System or a hash-based Tuple Space System greatly improves the system performance in terms of number of data view operations and number of Tuple Spaces accessed. In order to test the

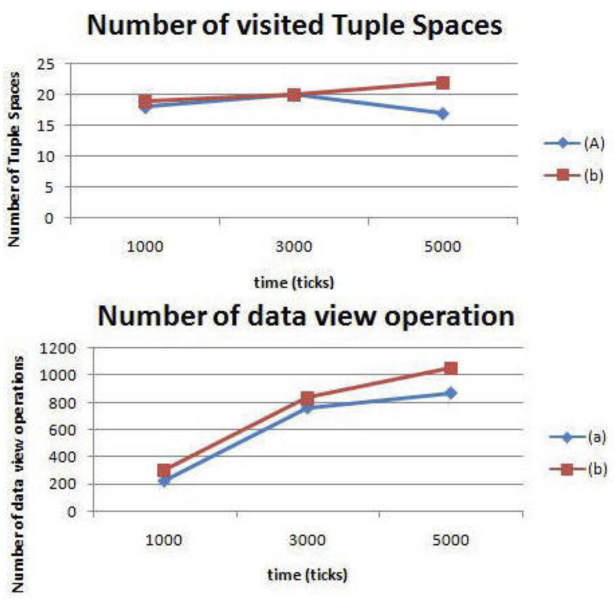

Figure 6: Scalability test. In (b) the simulated area is doubled than in (a), while the population density is the same.

scalability of our approach we performed some preliminary test (see Figure 6), simulating a wider environment (extension doubled than the previous one) with the same population density of the previous one. The results are shown in Figure 6. (a) is the first environment we simulated, (b) is a second environment with doubled extension. We run both the simulation for a while and measured how the systems perform when submitting the query presented at the beginning of this section. The figures shows that the system scale quite well. One should note that in (b) the number of tuple spaces in the system and the number of generated tuples increases. In particular (a) accounts for 100 Tuple Spaces and (b) for 150. However, given that the number of Tuple Spaces in the system increases by 0,5 factor, we have that the number of accessed Tuple Space does not increase significantly, this is due to the fact the number of tuple spaces involved in the knowledge networks related to the query performed almost does not increase. Instead when considering the number of accessed tuple, we must note that the number of tuple in the system greatly increase in (b) and this motivate why the number of tuple accessed in (b) respect to (a), this implies that also knowledge networks account for more tuples.

However, the generation of Overlay Networks never comes without overhead costs. The idea behind W4 Knowledge Networks, as already stated in Section 4, is to pre-organize data in a fashion that will be useful for a number of services and will decrease delay experiences by services. In order to keep Knowledge Networks feasible and manageable one should find a good trade-off between the number of knowledge networks to be build in the system and overhead costs associated, in general only knowledge networks that might be useful (i.e., accessed by agents) should be built and maintained. An approach that goes in this direction is the one presented in [6]. Another factor to be considered is the number of agents that are involved in Knowledge Networks' generation and maintenance. Having multiple agents collaborating for a single Knowledge Network, this would decrease the knowledge networks' generation time and would allow the knowledge network to be promptly updated but, once again, this increases the overhead.

6. RELATED WORK

Context is a very fluid notion, although several researchers claim that it is very hard to abstract it in terms of variables and data models [19], it is also a widespread opinion that a more pragmatic perspective should be adopted. Early works in this area, as from Schmidt et al. [24] and Dey et al. [8], concentrates on the issue of acquiring context data from sensors and of the processing such data but they generally miss in identifying a uniform model to describe the data. A different thread of researchers [11] focuses more on the issue of providing rich data models for contextual information and of facilitating querying by services. However, this does not go toward simplicity and generality, which we instead feel should both be goals. Some recent proposals, such as [25, 3] focus on providing models for contextual data that adopt a uniform well-defined structure. Indeed, our W4 proposal accounts for a very similar structuring for contextual information, and enriches it further with a well-defined API, and with the possibility of linking data atoms and of providing application-specific views to services.

An increasing number of research works get inspiration from tuple space data models [1] and propose representing contextual information in the form of tuples. Egospaces [14] adopts this perspective, without committing to a specific pre-defined structure for context tuple, which can make it difficult for services to uniformly deal with tuples represented in different formats. Other proposal, such as The Context Fabric model [12] rely on well-structured context tuples. Recent proposals focusing on sensor networks, suggest exploiting a tuple-based approach to provide application-specific views on sensorial data [18]. In general we consider tuple-based approaches very suitable for organizing and accessing contextual information, but we also think that there is need of more structuring and flexibility than those exhibited by the existing approaches.

However, in the previous work, the issue of relating contextual data atoms with each other and of providing different views to different applications is not generally addressed. More recently, other proposals have adopted a similar endeavor but have considered the issue of adopting specific ontologies to model context information and enable Ű other than efficient querying Ű also efficient context-reasoning [22, 15]. Although such approaches tend to be too application-specific, they attribute the importance of linking independent atoms of contextual information (with ontological relations) and of reasoning not only on individual data items but also on their relations, an idea which is fully shared by our knowledge network vision.

Some recent proposals go beyond context modeling and representing and start consider also context reasoning, i.e. considering related piece of information for generating new knowledge as w4 knowledge networks do. Many works, such as and [21], are focused on situation learning and situation relationships in smart environment. Other works, such as [20] propose predicate logic as an effective language for context-aware reasoning. The Knowledge Networks approach we propose aim to be more general and propose an approach different from traditional ones, considering self-organizing agents. [4] considers the possibility of extracting higher-level knowledge from raw sensed data merging feature vectors in an opportunistic fashion for people-centric application. The idea of merging and considering data coming from diverse sources is shared with the w4 knowledge networks approach. However in the W4 Approach we go further considering multiple knowledge views that can be accessed by multiple services. In [23] the context management processes is performed by BDI agents that generate and administrate the context artifacts at run time. We share with [23] the idea of a middleware responsible for contextual data, but we also argue overwhelming amounts of data require novel approaches for data management, such as bio-inspired and self-organizing ones.

Also other areas of research contributed towards the realization of our knowledge networks vision, in particular data mining and pattern discovery and granular computing. Data mining [9] always come to play when dealing with an overwhelming amount of data to be analyzed, indeed the analysis task performed by the knowledge networks layer can be considered as a sort of data mining process. Also Granular Computing [26] is of interest, indeed w4 knowledge atoms can be seen as information granules, providing knowledge at different scales.

7. CONCLUSIONS AND FUTURE WORK

Pervasive computing devices can generate an overwhelming amount of data related to facts occurring in the physical and social world. Making such data effectively available to services requires proper solution to represent data, pruning and organizing into application-specific views on contextual knowledge. In this paper, after having discussed the above issues, we presented the W4 Approach as an effective solution to represent contextual data and organize them in distributed knowledge networks. We proposed a self-organized mechanism capable of analyze the data and organize them in multiple views usable by pervasive services. We have shown experimental results to support our proposal.

Despite the promising results achieved so far in the study of the W4 model, several research issues still have to be faced. The Knowledge Networks approved showed useful, however more experiments should be done to evaluate properly the generation and maintenance overhead, as introduced in Section 5. Moreover, in the current implementation of the W4 System, the number of tuples stored in the system is constantly increasing as new data are injected in the system. The more knowledge atoms are stored, the more will be the overhead on the the knowledge network level. There is the need for a "garbage collection" solution, we plan to experiment with a concept of knowledge tuple fading as introduced in [17].

8. ACKNOLEDGEMENTS

Work supported by MIUR under the project PRIN 2007 "Analisi e controllo di fenomeni franosi attraverso sistemi di monitoraggio wireless e airborne" (Wireless and Airborne Monitoring of Landslide phenomena).

9. REFERENCES

[1] S. Ahuja, N. Carriero, and D. Gelernter. Linda and friends. *Computer*, 19(8-9):26–34, 1986.

[2] D. Balzarotti, P. Costa, and G. P. Picco. The lights tuple space framework and its customization for context-aware applications. *International Journal on Web Intelligence and Agent Systems*, 50(1-2):36–50, 2007.

[3] J. Bravo, R. Hervás, I. Sánchez, G. Chavira, and S. Nava. Visualization services in a conference context: An approach by rfid technology. *Journal of Universal Computer Science*, 12(3):270–283, 2006.

[4] A. Campbell, S. Eisenman, N. Lane, E. Miluzzo, R. Peterson, H. Lu, X. Zheng, M. Musolesi, K. Fodor, and G.-S. Ahn. The rise of people-centric sensing. *Internet Computing, IEEE*, 12(4):12–21, July-Aug. 2008.

[5] G. Castelli, M. Mamei, and F. Zambonelli. Engineering contextual knowledge for autonomic pervasive services. *International Journal of Information and Software Technology*, 52(8-9):443–460, 2008.

[6] G. Castelli, R. Menezes, and F. Zambonelli. Self-organized control of knowledge generation in

pervasive computing systems. *ACM Symposium on Applaied Computing, 2009*, 8-12 March 2009.

[7] G. Castelli, A. Rosi, M. Mamei, and F. Zambonelli. A simple model and infrastructure for context-aware browsing of the world. *Pervasive Computing and Communications, 2007.*, pages 229–238, 19-23 March 2007.

[8] A. K. Dey, G. D. Abowd, and D. Salber. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *HumanŰComputer Interaction*, 16(2):97–166, 2001.

[9] J. Galloway and S. J. Simoff. Network data mining: methods and techniques for discovering deep linkage between attributes. In *APCCM '06: Proceedings of the 3rd Asia-Pacific conference on Conceptual modelling*, pages 21–32, Darlinghurst, Australia, Australia, 2006. Australian Computer Society, Inc.

[10] K. Henricksen and J. Indulska. Developing context-aware pervasive computing applications: models and approach. *Pervasive and Mobile Computing, In*, 2:2005, 2005.

[11] K. Henricksen and J. Indulska. Developing context-aware pervasive computing applications: models and approach. *Pervasive and Mobile Computing*, 2, 2005.

[12] J. I. Hong. The context fabric: an infrastructure for context-aware computing. *CHI '02 extended abstracts on Human factors in computing systems*, pages 554–555, 2002.

[13] Y. Jiang, G. Xue, Z. Jia, and J. You. Dtuples: A distributed hash table based tuple space service for distributed coordination. *Grid and Cooperative Computing, 2006*, pages 101–106, October 2006.

[14] C. Julien and G.-C. Roman. Egospaces: facilitating rapid development of context-aware mobile applications. *Software Engineering, IEEE Transactions on*, 32(5):281–298, 2006.

[15] D. Lee and R. Meier. Primary-context model and ontology: A combined approach for pervasive transportation services. *Pervasive Computing and Communications Workshops, 2007. PerCom Workshops '07. Fifth Annual IEEE International Conference on*, pages 419–424, 2007.

[16] F. Z. M. Baumgarten N. Bicocchi, M. Mulvenna. Selforganizing knowledge networks for smart world infrastructures. In *International Conference on Self-organization in Multiagent Systems*, 2006.

[17] R. Menezes and A. Wood. The *fading* concept in tuple-space systems. In *Proceedings of the 2006 ACM Symposium on Applied Computing*, pages 440–444, Dijon, France, 2006. ACM, ACM Press.

[18] L. Mottola and G. P. Picco. Logical neighborhoods: A programming abstraction for wireless sensor networks. *In Proc. of the the 2 nd Int. Conf. on Distributed Computing on Sensor Systems (DCOSS)*, 2006.

[19] D. Paul. What we talk when we talk about context. *Personal and Ubiquitous Computing*, 8, 2004.

[20] A. Ranganathan and R. H. Campbell. An infrastructure for context-awareness based on first order logic. *Personal Ubiquitous Comput.*, 7(6):353–364, December 2003.

[21] P. Reignier, O. Brdiczka, D. Vaufreydaz, J. L. Crowley, and J. Maisonnasse. Context-aware environments: from specification to implementation, 2007. IN PRESS.

[22] I. Roussaki, M. Strimpakou, N. Kalatzis, M. Anagnostou, and C. Pils. Hybrid context modeling: A location-based scheme using ontologies. *Pervasive Computing and Communications Workshops, IEEE International Conference on*, (1):2–7, 2006.

[23] I. Salomie, T. Cioara, I. Anghel, and M. Dinsoreanu. Rap - a basic context awareness model. *Intelligent Computer Communication and Processing, 2008. ICCP 2008. 4th International Conference on*, pages 315–318, 2008.

[24] A. Schmidt, K. A. Aidoo, A. Takaluoma, U. Tuomela, K. V. Laerhoven, and W. V. d. Velde. Advanced interaction in context. *Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, 8, 1999.

[25] C. Xu and S. C. Cheung. Inconsistency detection and resolution for context-aware middleware support. *Proceedings of the 10th European software engineering conference held jointly with 13th ACM SIGSOFT international symposium on Foundations of software engineering*, pages 336–345, 2005.

[26] Y. Yao. Three perspectives of granular computing. *Journal of Nanchang Institute of Technology*, 25(2):16–21, 2006.