Dynamic Context Networks of Wireless Sensors and RFID tags

Tomás Sánchez López, Daeyoung Kim
AutoID Lab Korea, ICU, Daejeon, South Korea.
Email: tomas, kimd@icu.ac.kr

Kyungseon Min and Joonho Lee
Korea Telecom, Daejeon, South Korea.
Email: minks, joon@kt.co.kr

Abstract— **Current research on pervasive systems envisions a world where computation is not only everywhere, but is also attached to objects and users moving and interacting with the environment. For these mobile computing *entities* to communicate, wireless links should be evaluated, built and released in a dynamic, intelligent way. In this paper we analyze the specific case of mobile entities augmented with sensor and Radio Frequency Identification (RFID) information. We propose a set of protocols for intelligent building of dynamic context networks, where node interactions are conditionally granted and a shared context is built with the information provided by collaborating entities.**

I. Introduction

RFID tags hold Electronic Product Codes (EPC) that can be transmitted over the air and serve as universal identifiers. Wireless Sensor Networks (WSN) are small devices able to transmit sensor data through the radio channel and perform limited amounts of computation. In our previous work, we proposed WISSE [1] as a service framework for mobile Wireless Sensor Networks and RFID. In WISSE, mobile entities interact and build groups with shared context information in order to receive meaningful services from a Service Layer (Figure 1). These entities, ranging from any kind of objects to users, are equipped with WSN and RFID tags as the means for gathering their context information.

In this paper, we analyze in detail a part of the WISSE framework, the Context Layer, where all the interactions among mobile entities occurs. In the Context Layer, entities associate and form groups that provide information about a common context. This context is relevant to users that are related to the entities, so the the more accurate the provided information is, the better the services the users will receive. In order to perform appropriate associations, entities must discern which groups they should join and assign association priorities according to their own preferences. Furthermore, groups of entities and the protocols that tie them together need to be flexible enough to support dynamic joins and leaves. Figure 1 shows the WISSE architecture emphasizing the Context Layer. In order to help the understanding of which functionality is necessary for this layer, we use an example with several entities and how they decide to group together to provide context for a user “businessman”.

This paper is organized as follows. Section II compares our work with other related work in the area. Section III introduces the Context Layer architecture. Section IV details the main grouping algorithms while Section V introduces some extensions. Finally Section VI concludes the paper.

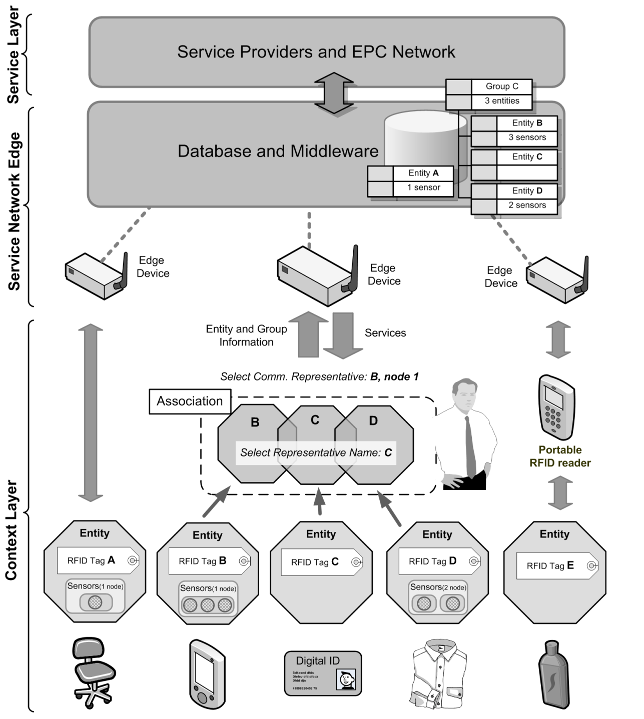


Fig. 1. WISSE framework through an example. Three entities (PDA, Shirt and ID card) team up to provide context information about a user ”businessman”.

II. Related Work

WISSE architecture could be considered inside the Smart Space concept, where users’ context is automatically interpreted and actions according to that interpretation are taken. In traditional Smart Spaces, context information is usually obtained through sensors located in the edge of the system, observing the user [10] [8] [11] [9]. In this approach, the user and the context are decoupled in the sense that the main source of information is not from the user’s point of view but from the system’s point of view. This means that all the decision power is put on the system’s side, as the context information about a user is limited to external sensing and to make adequate decisions with this limited information. In WISSE, part of the decision power is placed on the user’s side, whose responsibility is to gather the context information only from relevant sources. This simplifies the service provider's task, but increases the complexity on the user side which has to deal with entity associations and information updates. Most importantly, this new concept of *active users* entails great benefits by providing first-hand context data which is very hard to obtain with traditional approaches.

There exist some works in the area that attempt to provide first hand information in a similar way our work does. In [12], users are augmented with portable devices and sensors, but the service infrastructure is inexistent, making impossible a truly distributed system. Other works such as [13] build agents representing the sources of context and perform groupings of related agents. However, entities with various capabilities should communicate with the system independently, which makes the architecture inflexible and unscalable. Projects such as [3] and [4] and focus on augmenting everyday objects the same way we do. However, they lack a general infrastructure to provide spontaneous services which follow the independence pattern that our work follows.

To our knowledge, there are very few attempts to integrate RFID and sensor networks, and none of them embed both technologies inside mobile entities and use the EPC Network to enhance the Smart Spaces services. In [5], sensor nodes and RFID readers are merged to provide the user the capability of reading tags in the environment. However, among other differences, tags don't belong to user's context and there is no infrastructure to offer services according to sensor or RFID information. [6] also integrates RFID readers in sensor nodes, but only intends to, from a user point of view, extend the range of the RFID system. In [7], RFID tags are used into objects and users, while sensors are also distributed in the environment. However, tags are only read to deduce the user's context in terms of spatio-temporal constraints.

III. Entities

WISSE entities are not physically different from any object that may carry a RFID tag. The difference is functional: while normal RFID tagged objects are just individually read by some RFID reader, WISSE entities may associate and then transmit their information jointly. In order to achieve this, entities use the wireless communication capabilities of their sensor nodes.

From the broad concept of Radio Frequency Identification (RFID), we choose the EPC Global Tag Data standards [15]. The Electronic Product Code (EPC) is an identification scheme designed to support the needs of various industries by accommodating both existing and new coding schemes. The EPC Global standards specify a whole architecture framework in a collection of interrelated standards for hardware, software and core services [2]. This architecture enables numerous advantages when managing information related to EPCs.

The EPC Global standards consider various kinds of tags distributed in classes. A total of five classes are defined, from the well known passive tags to the battery powered active tags. In December 2004, EPC Global ratified the second generation of part of the tag standards, known as the Class-1 Generation 2 UHF RFID, or air interface. In order to emphasize a practical approach, we focus our research in current available technology, such as Class-1 Gen2 RFID tags and existing wireless sensor networks. However our current research doesn't exclude the use of future technology and part of of our present considerations regarding issues such as mediation and information capture will be solved with the evolution of the standards themselves.

According to what we argued before, WISSE entities should carry one (and only one) passive RFID tag. The tag's ECP will uniquely identify the entity, the same way today's tagged objects are identified by one EPC. In addition, each entity may carry several sensor nodes, which may contain any combination of sensors and actuators. Ideally, each sensor node has a map of the RFID tag's memory of the entity it belongs. Having so, each node has a sense of identity by holding an EPC and has also access to other parts of the memory needed by the WISSE protocols. Unfortunately we can't assume WSN to be able to read RFID tags. We believe is a reasonable assumption for manufactures integrating both RFID tags and WSN in their products to inject RFID information into the nodes by reader/SN gates installed in factory facilities such as conveyor belts. This way, while RFID tags would still maintain their functionality if read by regular RFID readers, the sensor nodes could participate in the WISSE network on behalf of the same objects. Nevertheless, Section V considers how to overcome the situations where this mapping was not done at manufacturing time.

IV. Grouping

Single entity data is normally poor in the sense that can not offer enough context information for obtaining relevant services. Providing as much data as possible for the same client may extend its context information and hence the possible available services.

WISSE entities sense not only the environment but also the presence of other entities, and so may associate with them if they have the same interests. Grouping is the process by which two or more entities decide to collaborate by sharing their context information. Entities periodically advertise their presence by sending advertisement packets and listen to other entities in periodic, unsynchronized intervals. To provide entities with the ability to decide which other entities they should associate with, we divide the grouping process in two phases. The first phase involves the information stored in the RFID tags to make basic decisions and prioritize the association process. The second phase involves choosing representatives, invoking the addressing process and distributing the results to all the group members. Only when the grouping process is finished, the entities will be aware of their new membership and results will be communicated to the rest of the WISSE infrastructure.

In order to deal properly with dynamic wireless networks, WISSE organizes any combination of associated entities in a double clustered architecture. On one hand, each individual entity chooses a cluster head, which will communicate with other cluster heads from other entities. On the other hand, a group of associated entities chooses a correspondent cluster head to communicate with the service network. Finally, each entity group chooses a meaningful identifier (EPC) among all entities, which will represent the group for the rest of the system. An entity will report its interactions dynamically, getting services according to the context gained through them. The following sections detail all the process happening in the Context Layer, from choosing which entities shall form a group to which identifier should be chosen.

A. Representative Election

1) Entity Cluster Head Election: Entity Cluster Heads (ECH) are used by single entities to communicate with other entities. Thus, the first task of the entity's sensor nodes is to undertake an ECH election process. This process is as follows: When a node marked as "ECH capable" is unable to find its entity cluster head, it sends a proposal for being ECH for a time T_{entity}, which is a function of the node's remaining energy. Only nodes which compute higher T_{entity} will respond to the proposal. In order to avoid collisions when more than one node tries to answer, a delay in the response is introduce, also function of the remaining power plus a random component [14]. We can compute the proposed time and delay as follows:

$$T_{entity} = C_1 \times RemainingEnergy$$

$$Delay = \frac{C_2}{RemainingEnergy} + Random$$

where $0 \leq Random \leq \frac{C_2}{2 \times MaxEnergy}$ and C_1, C_2 are constants

Once the process is finished, the ECH election result will be broadcasted to the rest of the entity's nodes, which from that moment will use the ECH as a relay node to communicate with the outside.

2) Correspondent Election: WISSE *correspondents* are defined as ECHs that are elected to communicate the entity with the Service Network Edge. Only one *correspondent* can exist per group of associated entities. If an entity doesn't belong to any group, its ECH is automatically promoted to *correspondent*.

Not any ECH is eligible at any time to become a *correspondent*. It is possible for certain nodes of an entity to be in range with some sensor network gateway (edge device) while some other remain "hidden" or out of range. Apart from an efficient energy use, the *correspondent* election procedure should not choose a *correspondent* which is hidden while some other ECH from the group is in range with an edge device. To address this issue, edge devices send advertisement packets to announce their presence. Only ECHs that receive an advertisement (ADV packet) will start the *correspondent* election process:

The *Correspondent* election procedure follows the same algorithm exposed before for the ECH election, using $T_{corresp}$ as the proposed time. The *Correspondent* election procedure starts when:

1) No *correspondent* is currently selected in the group,
2) When $T_{corresp}$ expires
3) When current *correspondent* looses range with all the edge devices
4) An ECH who didn't participate in the previous election and has more remaining power than current *correspondent* now receives an ADV packet
5) An ECH can not communicate with current *correspondent* before $T_{corresp}$ expires
6) A new entity is added to the *correspondent*'s group.

According to this algorithm, if an entity looses its *correspondent* and no ECH receives an ADV packet, the election process won't start again to choose a new *correspondent*. This situation is undesirable because the context information of the grouped entities may still be useful locally. Moreover, we can not allow new groupings to be discarded due to temporal disconnections. To avoid this problem, the *correspondent* selection procedure will be started by any node which runs more than a certain time T_{No-ADV} without being able to communicate with its *correspondent*. When connection with the service network is reestablished, entities holding a *correspondent* selected in this way will start a regular *correspondent* election procedure again.

In our research we focus on groups of entities that can associate dynamically and spontaneously. For this reason, we should avoid storing vital grouping information in a centralized manner, but rather distribute this information. This is specially true for the elected *correspondent*, as the entity is contained in may un-group without prior notice. Unlike *correspondents*, however, EHCs are unlikely to die without warning as they remain attached to their entities. WISSE defines that each ECH shall store a table with a list of its entity's nodes and the sensors and actuators they hold (Node Table). Additionally, the ECH will store its address and may store some other additional information such as its level in the association hierarchy. When an ECH finishes its representation period or is about to exhaust its battery, its node table will be transferred to the next elected ECH.

B. Grouping Procedure

The grouping procedure concept behind WISSE associations is rather simple: entity *correspondents* send periodic broadcast of `grouping_request` packets looking for other entities to associate with. Entities that receive `grouping_request` packets may process the packet information and decide to associate sending back a `grouping_response`. The responding entity makes decisions on which will be the new representative EPC of the resulting entity. Results are communicated to all entity members and a new *correspondent* election process begins. This process may involve several entities at the same time.

However, a too simplistic approach may lead to an unrealistic design. In general, we need to monitor the grouping procedure to avoid uncontrollable chain associations that will exhaust the nodes' batteries. In particular, if the objective is to provide shared context relevant to the clients, association nature and priorities should be consider carefully before using the devices' scarce resources. In our work, we try to meet these requirements by proposing a two-phase grouping procedure. The first phase, or pre-grouping, aims to organize and filter the entities that should undertake a full association procedure, which takes place in the second phase. The following sections describe the concepts and algorithms behind this two-phase grouping.

1) Phase 1: Pregrouping: In general, the main purpose of the RFID tags is to provide unique identification. In the EPC Global Class-1 Gen2, that unique identification is given by the tag's Electronic Product Code. The standard also specifies other kind of information that may be stored in the tag's memory. There are four logically separated memory banks in a Gen2 tag. Bank 1 contains the EPC information, while banks 0 and 3 contain other data for security and compatibility reasons. Gen2 also specifies a fourth memory bank called the *user* memory bank. Its organization, size and purpose is said to be user-specific. WISSE uses this additional data storage to keep a minimum set of logical information that will help in the pre-grouping phase. We justify the need for this additional data based on the observation that:

1) The properties of the EPC Global core services and infrastructure makes it possible to retrieve information from remote databases using any EPC as a search key. This information could help to make decisions on the association procedure. However, even if we could make relevant information available for any EPC and build the logic to process that information, we would still have a connectivity problem: the inherently unreliable wireless communications may occasionally break the bridge to upper layers, making logical information unavailable. Furthermore, decision time could become unacceptably long.
2) We should consider priorities when grouping multiple entities at the same time. Even if the "businessman" from Fig. 1 could associate with his office chairs to receive, let's say, chair locations for tracking purposes, he would definitely prefer to associate his PDA and his ID card first to find out his own location.
3) Security mechanisms are necessary that will allow private groups and prevent unauthorized associations.

WISSE specifies the user memory bank containing two different ids, a *user ID* and a *group ID*. Unlike the EPCs, these 16 bit identification numbers don't refer to the physical product, but rather to its logical use. The *group ID* is compulsory and shall contain a non zero value for any RFID tag compatible with the WISSE architecture. Its use is to define general object classes such as furniture, human, vehicle, food, books, clothes, etc. The *user ID* is optional and is intended for user-defined object classes. Additionally, a user password may be specified for preventing unauthorized association to user-defined groups. A tag not belonging to any user-defined class should set its *user ID* to 0. In this case, the user password bit shall be also set to 0. Figure 2 shows the logical memory map of the user memory bank of tags participating in the WISSE architecture.

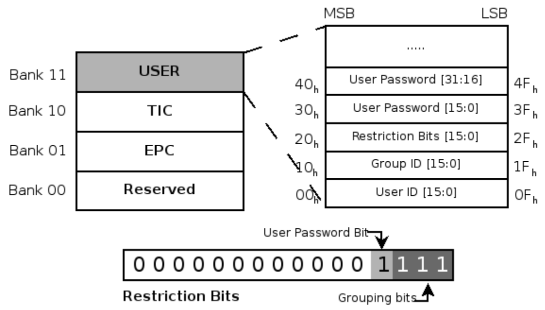


Fig. 2. RFID tag user memory bank layout

Entities send their user memory bank information together with their `grouping_request` packets once every Grouping Period (GP). GPs have a fixed size, and they always start with an also fixed sending interval in which the entities actually send their `grouping_request` packets. During the rest of the GP, entities listen for other `grouping_request` packets. Figure 3 shows how the grouping periods are distributed. During the fixed interval time T2-T1 entities listen for request packets. If any request is received, they process them producing a `grouping_response`. The processing time is variable depending on the number of requests received. If any request is received from T2 to T3, they are queued and will be processed at next GP's processing period, starting at T5.

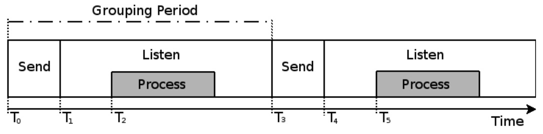
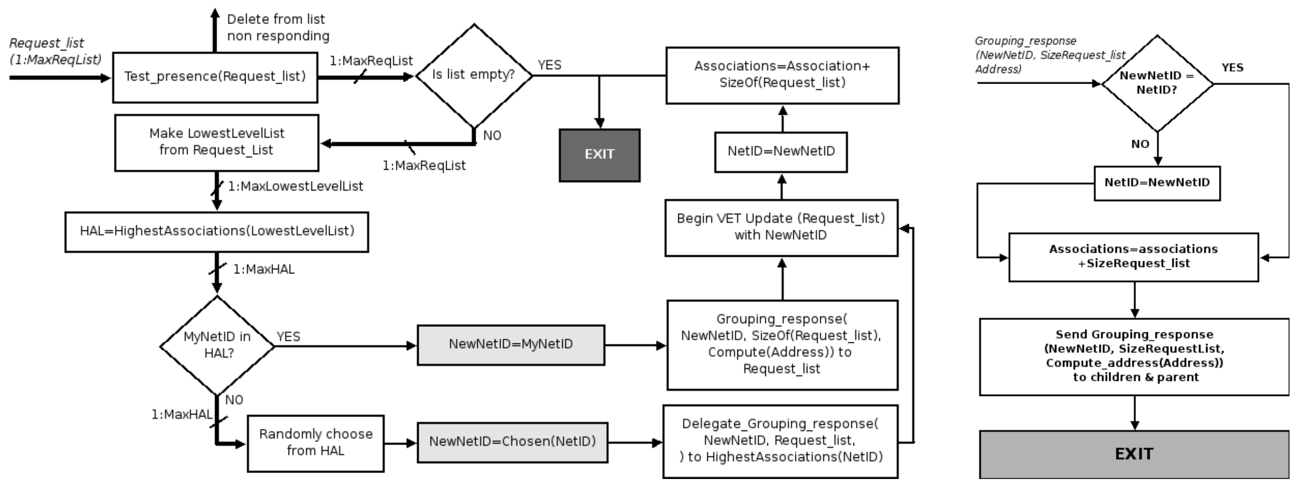


Fig. 3. Distribution of Grouping Periods.

As mentioned before, entities send their user memory bank information together with their `grouping_request` packets. According to this information, WISSE pre-grouping distinguishes 3 classes of entities at association time: *User*, *Group* and *Other*. The *User* class is granted to those `grouping_request` packets that share the same user ID as the entity that receives that packet. The *Group* class is granted in a similar way but considering the group id values. Packets not matching the *User* id nor the *Group* id values are granted the *Other* class.

It is also possible to specify which kind of combination of classes this entity is allowed to group with. The grouping bits provide this function by marking "1" when a certain class is allowed for association. The default value is "111". meaning that this entity is willing to associate with any kind of entities. Grouping requests are hence classified upon reception in the


Request_list (1:MaxReqList)
Test_presence(Request_list)
Delete from list non responding
1:MaxReqList
Is list empty?
YES
NO
EXIT
Associations=Association+ SizeOf(Request_list)
NetID=NewNetID
Begin VET Update (Request_list) with NewNetID
Make LowestLevelList from Request_List
1:MaxReqList
1:MaxLowestLevelList
HAL=HighestAssociations(LowestLevelList)
1:MaxHAL
MyNetID in HAL?
YES
NO
NewNetID=MyNetID
Grouping_response(NewNetID, SizeOf(Request_list), Compute(Address)) to Request_list
1:MaxHAL
Randomly choose from HAL
NewNetID=Chosen(NetID)
Delegate_Grouping_response(NewNetID, Request_list,) to HighestAssociations(NetID)
Grouping_response (NewNetID, SizeRequest_list, Address)
NewNetID = NetID?
YES
NO
NetID=NewNetID
Associations=associations +SizeRequest_list
Send Grouping_response (NewNetID, SizeRequestList, Compute_address(Address)) to children & parent
EXIT

(a) Grouping Request (b) Grouping Response

Fig. 4. Block diagrams related to the `grouping_request` and `grouping_response` packets. The algorithm in a) will be executed when a list with accepted requests is received from Phase 1. The algorithm in b) will be executed when the a `grouping_response` packet is received

three mentioned groups, and are processed by class starting from the ones in the *User* class and finishing with those in the *Other* class. This procedure assures not only that associations will be granted in a controllable way, but also that there will be a priority in those associations, giving preference to the user-defined classes. Figure 5 depicts the algorithm that classifies the requests in classes according to their user memory bank information.

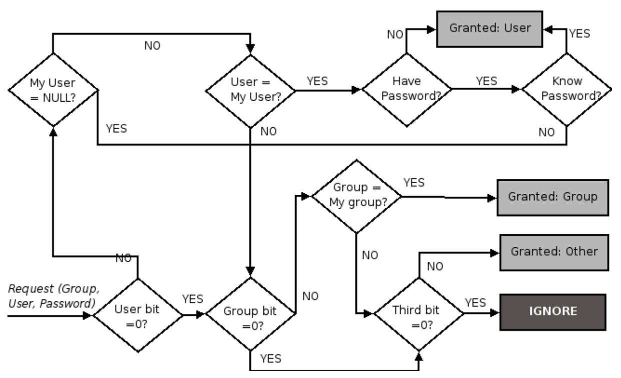


Fig. 5. Classification of request in the pre-grouping phase

The pre-grouping phase ends by providing the next phase with a list of received requests with the highest priority. Only one class will be processed in one grouping period, while the rest of requests will be queued in a FCFS manner for next processing time.

2) Phase 2: Grouping: One of the most important principles of WISSE is that entities are represented in the system by one EPC. When entities associate, one of their EPCs is chosen as representative of the group. We could say that the group of associated entities form a wireless sensor network, and that the EPC is its network ID (or NetID, for short). This design allows the addressing of meaningful groups of entities globally under a unique identifier. Nonetheless, WISSE still maintains the possibility of accessing every individual EPC through a database of associations called the Virtual Entity Database (VED). The VED design is described on [1].

Although the purpose of having a unique address for every group of associated entities is fulfilled by choosing any of their EPCs as representative, is obvious that the more meaningful this choice is, the best it will represent the group. For example, our "businessman" group would be better identified by the EPC from its digital ID card than the EPC of its shirt, even if both EPCs may be valid unique network identifiers. Better choices allow to easily recognize the nature of a group without having to dig in all its entities. In order to make optimum choices and better organize the associations, the VED is also arranged in a hierarchical manner, where lower levels hold more representative entities.

The main goal of Phase 2 is to process the `grouping_requests` coming from Phase 1 and choose a NetID among all the entities that will associate. The chosen NetID must be notified to all the members of the group. Another important task of this phase is assigning local addresses for communication and routing. Explanation of the addressing scheme (called Sequence Chain) is out of the scope of this paper.

Figures 4(a), and 4(b) depict the algorithms involved in this second phase of grouping. Since entities are mobile, the first task of the algorithm is to check if all the entities from which we have a request are still there. Next task is to elect the NetID that will represent the entity. For this, we first choose according to the level of the entity in the hierarchy, as we mentioned earlier. For the same level, we choose the entity which is already grouped with more entities. This has to do with efficiency reasons, as the cost of updating databse records is directly related with how many entities are associated around the same NetID. Finally, the algorithm generates the `grouping_response`, sends the updates to the service network and updates also its number of associated entities.

The previous algorithm, executed by ECHs receiving `grouping_request` packets, produces a `grouping_response` packet that is sent back to the sender. Upon reception, the node receiving the `grouping_response` should, first, proceed with the changes indicated in the packet. These changes involve updating its NetID information, its address and the number of entities that now form part of the group. Finally, those changes should be transmitted to the rest of the entities of this group. The algorithm achieves that by sending itself in a recursive manner until all the members are notified. Note that the part of the algorithm that deals with addressing issues has been simplified to make the diagrams clearer.

V. Grouping Extensions

As we mentioned in3 section III, the ideal scenario for our framework is the RFID tag's memory to be mapped into the RFID node's memory. However, it is possible for this assumption to be invalidated by scenarios where manufacturers don't provide the mapping or where new nodes are added to the entities by the users. To try to consider all possible scenarios, we also propose an extension to the regular grouping where the tag's memory is not initially mapped into any sensor node.

In order to read RFID tags inside the mobile entities without using sensor nodes, we should consider portable RFID readers. Readers keep a list with the RFID tags they read. This way we can avoid repetitive groupings, not only with the tags mediated through the RFID reader, but also with tags that belong to regular entities. We call this list Registered Tag List (RTL).

There are two cases where the memory of a RFID tag might not be mapped into a sensor node's memory:

1) The entity where the tag is located doesn't contain any sensor node.
2) The entity where the tag is located does contain sensor nodes but one or more of them don't have a valid tag map into their memory

Algorithm 1 shows the procedure that a reader belonging to a WISSE group will undertake when it reads a tag. In lines 1-3, the reader checks if that tag was already read in the past. In line 4, the reader sends a MAC level broadcast looking for nodes without a valid network address (nodes without a valid tag memory map). If it receives any response (lines 6-7), the reader will inject the memory map from the tag to those nodes and let them undertake the regular grouping procedures. Otherwise (lines 8-10), the reader will register the tag as a new entity and will begin the periodic grouping procedure by sending `grouping_request` on behalf of the tag.

Algorithm 1 ReadTag(*tagMemoryMap*,*RTL*)

```
1:  if (tagMemoryMap.EPC ϵ RTL) then
2:      break
3:  else
4:      send responseList=broadcast(MACbroadcastAddress)
5:      add tagMemoryMap to RTL
6:      if (responseList ≠ null) then
7:          send mapInjection(tagMemoryMap, responseList)
8:      else
9:          send ve_register(tagMemoryMap.EPC)
10:         send grouping_request(gpLength)
11:     end if
12: end if
```

VI. Conclusion

In this paper we analyze in detail the Context Layer protocols of the WISSE framework. These protocols allow to transfer part of the smartness from context-based systems directly to the context producers. Furthermore, we exploit the use of the RFID technology by including additional information in the tags that will augment the context data and help the Context Layer protocols. Merging this work with the rest of the WISSE infrastructure provides a novel approach to Smart Spaces, where *active clients* provide dynamically their context information through arbitrary gateways, simplifying the needed infrastructure and providing the means for true pervasive environments.

Acknowledgment

This research has been sponsored by the M-BcN project of KT(Korea Telecom) BcN Business Unit in Korea.

References

[1] Tomás Sánchez López, Daeyoung Kim and Taesoo Park, *A service Framework for Mobile Ubiquitous Sensor Networks and RFID*, ISWPC'06
[2] Ken Traub et all, *The EPC Global Architecture Framework*, EPCglobal, July 2005
[3] F. Kawsar, Kaori Fujinami, Tatsuo Nakajima, *Experiences with Developing Context-Aware Applications with augmented artifacts*, Ubicomp 2005, Tokyo, Japan
[4] H.W. Gellersen, Albrecht Schmidt and Michael Beigl, *Adding some smartness to devices and everyday things*, WMCSA'00, 2000
[5] Waylon Brunette, Jonathan Lester, Adam Rea and Gaetano Borriello, *Some sensor network elements for ubiquitous computing*, IPSN 2005, 388-392, April 2005
[6] Christer Englund, Henrik Walling, *RIFD in wireless sensor networks*, Technical Report, Chalmers University of Technology, April 2004
[7] Yoshinori ISODA et al., *Ubiquitous Sensors based Human Behavior Modeling and Recognition using a Spatio-Temporal Representation of User States*, AINA04, Valume 1, p. 512, 2004
[8] Kiran Kumar, Salim Hariri, Nader V. Chalfoun, *Autonomous Middleware Framework for Sensor Networks*, ICPS 2005, 11-14 July 2005
[9] Saad Liaquat Kiani, Maria Riaz, Sungyoung Lee, Young-Koo Lee,*Context Awareness Scale Ubiquitous Environments with a Service Oriented Distributed Middleware*, ICIS 2005
[10] Anand Ranganathan and Roy H. Campbell, *A Middleware for Context-Aware Agents in Ubiquitous Computing Environments*, Middleware 2003, LNCS 2672, pp. 143161, 2003.
[11] Harry Lik Chen, *An Intelligent Broker Architecture for Pervasive Context-Aware Systems*, Thesis Disertation, University of Maryland, 2004
[12] Stephen S. Yau and Fariaz Karim, *Context-Sensitive Middleware for Real-time Software In Ubiquitous Computing Environments*, ISORC 2001
[13] Hideyuki Takahashi, Takuo Suganuma and Norio Shiratori, *AMUSE: An Agent-based Middleware for Context-aware Ubiquitous Services*, ICPADS'05, July 20-22 2005
[14] Srikanth Kandula, Jennifer Hou, Lui Sha, *A case for Resource Heterogeneity in Large Sensor Networks*, in Proceedings of MilCom 2004
[15] EPCglobal Inc, *Class 1 Generation 2 UHF Air Interface Protocol Standard Version 1.0.9: "Gen 2"*, Ratified Standard, January 2005.