A Context Middleware Based on Sensor and RFID information

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

Abstract

Pervasive computing environments have traditionally used distributed sensors to gather user context. EPC (Electronic Product Code) information extracted from RFID tags could additionally provide identification information associated with that context. However, the process of matching sensor and RFID information around the same context is not a trivial task. In our previous work, we proposed to move the sensors to the same objects that carry the RFID tags. By providing association capabilities among those objects using their EPCs, we can effectively build user and object context information while maintaining a global context identity. In this paper we analyze how the use of the EPC Network could greatly benefit our framework by providing the means for extracting, organizing and querying the context data. Additionally, we outline how services can be offered to the context data producers and how they can be built in a pervasive manner by linking them directly to the EPC Network infrastructure.

1. Introduction

Technology advances in embedded systems are, year after year, building the path towards the so called ubiquitous society. The design of small yet powerful devices and the provision of interoperability for a large variety of independent designed systems are certainly big challenges. Further difficulties arise when we consider the management of information originated by pervasive environments, such as sensor or RFID information. Large systems with potentially millions of information producers generate vast quantities of data. Furthermore, the data has to be organized to reflect the reality it represents and be properly indexed in order to facilitate query and update processes. Finally, a distributed data infrastructure offering standard interfaces is desirable in order to effectively make the information pervasively available.

In our previous work, we described mobile entities (users or objects) augmented with sensor information produced by Wireless Sensor Networks (WSN) and EPC information produced by RFID tags. Mobile entities interact and build shared context information that is transferred to a management agent in order to receive meaningful services from a Service Layer. In such a scenario, there is a need for a distributed engine able to capture and store dynamically the context, interpret it and retrieve associated services. In this paper we first analyze the challenges of such an engine and propose a middleware that aims to provide our clients with ubiquitous meaningful services. Secondly, we describe how the features of the EPC Network match the requirements of our framework and suggest how to combine both to create a pervasive infrastructure for Smart Environments. Finally we report our experience on the implementation of part of the infrastructure and discuss future directions.

The rest of the paper is organized as follows. Sections 2 and 3 outlines the system and relates it with other works. Section 4, 5 and 6 describe the key functionality of our middleware. Section 7 addresses the use of the EPC network both as a service and as an implementation asset and finally Sections 8 and 9 conclude with some implementation details and future work.

2. System Description

WISSE is a framework for service delivery to mobile entities carrying sensor nodes and RFID tags. We focus on the use of available and tested technology such as passive RFID tags (class 1 Gen2) and WSN, although we are also investigating the use of active tags which combine sensor and identification (class 2 and beyond). Figure 1 presents an overall view of WISSE through an example, showing the relationship between service producers and service consumers. Starting from the bottom, the Context Layer holds several entities such as clothes, furniture and personal identification cards. Each entity is identified uniquely by one EPC (Electronic Product Code) but it may carry several nodes with one or more sensors and actuators each. Entities sense not only the environment but also the presence of other entities, and so they may associate with them if they share the same context. In our example, three entities associate around a

user creating the abstract entity "businessman". Once the group is formed, the entities will immediately begin to produce context information about the businessman.

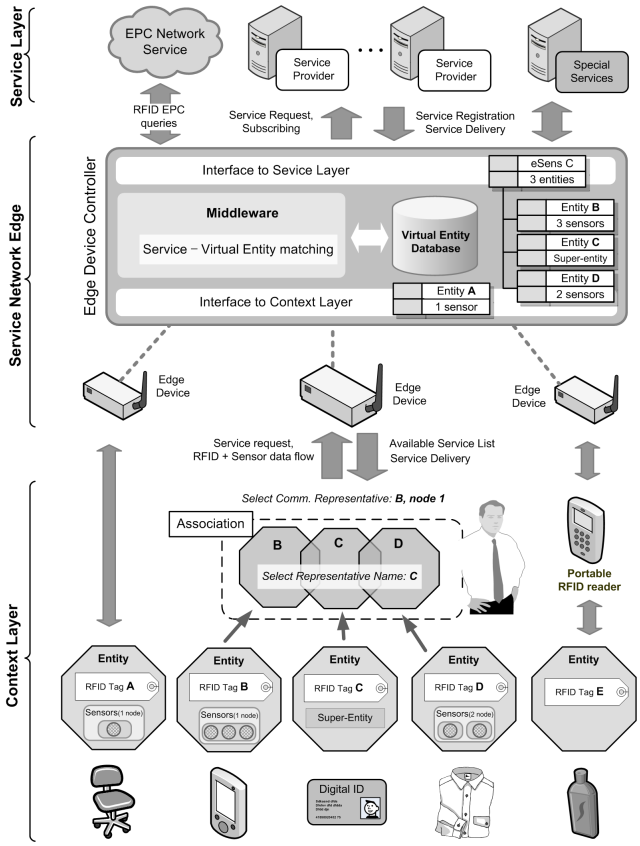

Figure 1. WISSE framework example

Service providers publish their service information in a Service Layer, which is nothing more than a pervasive accessible network such as the Internet. Services hold requirements that can only be satisfied by certain combinations of sensor, actuator and identification data. As an example scenario, consider the previous mentioned entity and user "businessman". While going back home after work, the user's shirt temperature and humidity sensor detects an increasing body heat resulting from an unusual warm day. Together with the location provided by his phone's GPS and identity from his ID card, this information is transmitted to the system by a CDMA link. The Service Network Edge recalls that this user also hired a home-network actuation service, and together with the user's context information, triggers a remote actuation command to turn on the businessman's home A.C. before he arrives home.

Many other scenarios can be constructed in a similar way by using the sensors, actuators and identity of the associated entities. In the end, the main goal of the framework is how to interpret the user's context information in order to offer meaningful services in an automatic, transparent manner.

3. Related work

Traditionally, 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. User's context is built entirely on the middleware from independent sensor readings distributed along the edge of the system. Context information is then interpreted to extract an overall meaning, and actions (services) according to that meaning are taken. This approach is clearly a limitation because the context about a mobile entity will always be limited to its external sensing and to make adequate decisions with this limited information. In some works such as [8], users are augmented with portable devices and sensors, but the service infrastructure is inexistent and all the middleware is included on the portable devices. Other works such as [3] build agents representing the sources of context and perform grouping of related agents. However, entities with various capabilities should communicate with the system independently, which makes middleware inflexible and hardly scalable. WISSE middleware (Figure 2) is based on context information obtained from the user's point of view, but still maintains a distributed infrastructure that analyzes the context and offers services. Furthermore, WISSE transfers part of the context reasoning to the users' level by allowing conditional associations of entities before the context is transmitted. We believe this framework balances the context processing tasks, providing an optimum combination of pre-processing and post-processing of information. In this way, WISSE simplifies the middleware decisions to a requirements matching, although increases the complexity on the user side that has to deal with entity associations and information updates.

Another key difference of the WISSE framework is the complete independence between users, services and middleware. Independent service providers publish their services with a standardized interface to a distributed repository. Users are offered a subset of services without a need to know anything about the service providers, and the middleware will just match the services from the repositories with the users' context at each moment in time. Projects such as [10] and [9] focus on augmenting everyday objects the same way we do. However, they lack a general infrastructure to provide spontaneous services which follow the independence patter that our work follows.

But maybe the most relevant difference and key contribution of our work is that WISSE provides a real integration of RFID and sensor networks in the sense that RFID data and sensor data refer to the same objects and users. On one hand, assigning unique identification to context allows an efficient and powerful middleware management. On the

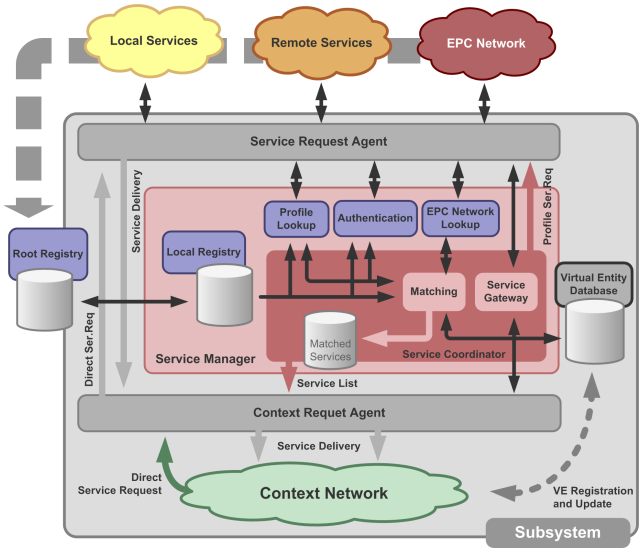

Figure 2. WISSE Middleware

other hand, the use of EPCs allows us to leverage the advantages of the EPC Network, offering potentially unlimited entity information beyond the WISSE infrastructure itself while inheriting RFID services such as *track-and-trace* and *business-to-business* relationships.

4. Context Capture and Storage

As explained before, the context information in WISSE is provided directly by the users and objects of the Context Layer. For each entity or group of associated entities (from now on referred just as "entities"), a dynamically elected representative plays the role of a bridge with upper layers transfering entity's context periodically. In order to do so, representatives route information through gateways that we call *edge devices* (Figure 1).

The information transferred by each entity is stored in a database for further processing. This information is organized around the entities' unique identifier (EPC), which assures a global data search key directly linked with the real entities. The piece of information that represents an entity and that is associated with a particular EPC is called Virtual Entity. Each Virtual Entity, thus, includes information of one or more associated entities, their RFID tags' EPCs, their sensor nodes and which particular sensors and actuators each node holds. Other data stored for each Virtual Entity includes a pointer to the edge device that most recently sensed the entity, the location information of all the edge devices, relationship history, observation history, etc. Note that the information stored in each Virtual Entity is dynamic, in the sense that it will reflect, at any point in time, the current situation of every entity under the governance of a particular middleware instance.

5. Context Interpretation

While the Virtual Entities are created and updated, on the other side of the system the service providers have registered their services through a standardized interface. The system holds a global repository where each WISSE service provider specifies the requirements for the services it offers. Additionally, every service should specify the means for its access. As an example, consider a pointer to a WSDL file for Web Services. Each middleware instance synchronizes periodically with the central repository. As mentioned before, service requirements are a mix of sensor/actuator types and identification information that service clients must fulfill.

The next step toward context interpretation is to find out which Virtual Entities satisfy the requirements of the registered services. This step is performed by the *Matching* module. The output of this process is a list of matched services, containing all the services whose requirements are fulfilled by all the Virtual Entities. Any addition of new services or any update in the Virtual Entity Database will trigger a new matching process. This assures a filtered and updated database of which possible services to offer to the clients. The matching process may be simple or grow in complexity depending on the service provider requirements. For example, let's consider the case of a service provider offering cell-phone game downloads, which grant an extended experience with the use of sensors. In this case, it is not only necessary to match sensors for the game, but also user authorization and compatible cell-phone models. One way to obtain enough information of whereas the service can be granted or not is by using the cell-phone's EPC. On one hand, device's EPC can be looked up in vendor databases to find out the model capabilities. On the other hand, device's EPC can be used to identify the user by contacting the cellular service provider databases. This authentication could also be verified, for example, if the user would carry some digital identification card that could take part in the entity group.

6. Service Delivery

Clients need to decide which services they wish to receive. The simplest way is to provide a list of possibilities and let clients choose (i.e. receiving the list of services through PDAs or cell phones). WISSE also supports the definition of profile sets where users define beforehand which behavior a certain client should have when services become available. Profile records are linked to the VED, maintaining references to which EPCs (entities) and which

services the profile refers to. Profile parameters are restricted to combinations of sensors, actuators and identities, to first order logic comparisons of their values, and to how a certain service should be executed (which is dependent on the service itself).

Once a service has been selected, the client needs to communicate with the service provider and define the service terms. Clients may consume services in a variety of ways, such as downloading the service, using RPC (Remote Procedure Call) or by just forwarding their context information. Furthermore, clients may want to perform a specific part of the service (for example, by providing some actuation) when a certain condition is met, such as reaching a defined sensor threshold, receiving some sensor event or arriving at a given location.

Finally, WISSE middleware should provide a gateway for those clients who are not able to contact the service providers directly. The *Service Gateway* module provides such functionality. Consider the following scenario: A bulb client is a simple entity formed by a single node with a light sensor and an adjustable power actuator. Additionally, a portable CD player entity is introduced in the room able to transmit the code of the CD that is currently playing. Both entities are grouped and registered in the Virtual Entity Database. When performing a service requirements matching, a new service is discovered that adjusts the intensity of a light source depending on the music played nearby. In order to keep the example simple, let's assume that the user (i.e. the owner of the house) has defined a profile allowing this service to be delivered. The service provider's duty is to analyze the CD code and determine the light intensity according to it. However, neither the CD player nor the bulb are capable of communicating directly with the provider (i.e. they don't have an Internet connection). The service gateway module bridges the entity and the service provider by forwarding the CD information to the server and transmitting back actuation information to the entity. Note that the service gateway is different from the edge devices in that the latest act as a hardware bridge (i.e. forwarding bytes among two communication technologies) and the former acts as a software bridge (i.e. forwarding sensor and actuation values).

7. WISSE Middleware and the EPC Network

As we stated before, WISSE entities are uniquely identified in the system using Electronic Product Codes. EPCs must also link entities with all the information stored about them, such as sensor and actuator types. Data is transferred from clients to the middleware, potentially filtered to discard non desired information and stored in databases, indexed by the entity's EPC. This data needs to be easily searched and updated, and is desirable to provide flexible interfaces for database access and functionality extensions.

The EPC Network infrastructure provides a similar functionality for the information gathered from RFID tags. Tag information is obtained though RFID readers, filtered at reader and middleware level (ALE [12]) and delivered to networked repositories (EPCIS [14]). These repositories behave as business entities and specify interfaces for insertion, update and query by either the ALE middleware or 3rd party applications. Data is also organized indexed by the EPCs obtained from the RFID tags. Using the flexibility of the exposed interfaces and performing small modifications on the system, it is possible to extend the EPC Network infrastructure to accommodate sensor and actuator readings and, what is more, to integrate it with the rest of the WISSE framework.

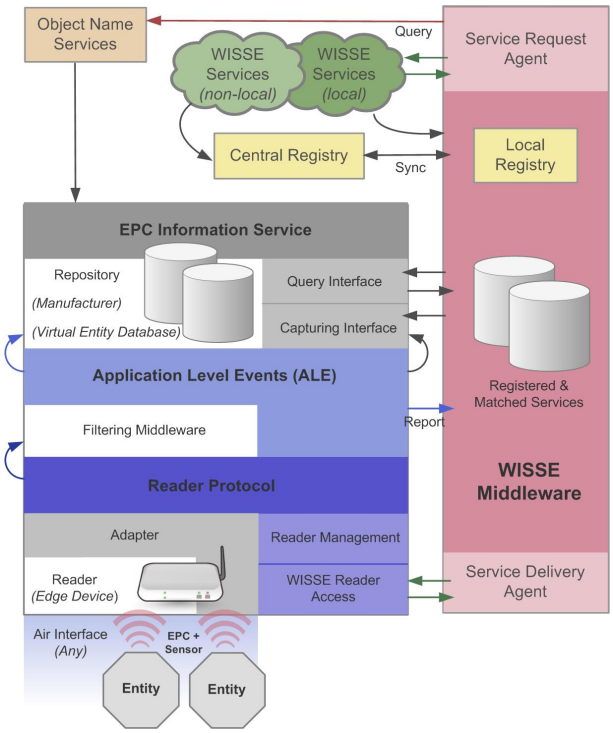

Figure 3. WISSE-EPC Network integration

Figure 3 shows our integration proposal of the WISSE framework and middleware with a modification of the EPC Network. Entities communicate through gateways sitting on the edge of the network (edge devices). In order to provide arbitrary communication technologies, adapters are used that control specific devices but expose standard interfaces. Information is then passed through the ALE middleware layer, filtered if necessary, and then forwarded to the databases in the Information Service. The EPCIS exposes query and capturing interfaces for the WISSE middleware, which behaves as an EPC Network accessing application. The Service Request Agent communicates the middleware

with the WISSE services, and the Service Delivery Agent links the accessing application with edge devices in order to provide the entities with the requested services. Additionally, Object Name Services (ONS [13]) can be queried to obtain pointers to other EPCIS that may contain additional information of the entities' EPCs.

There are two main advantages of the proposed integration. On one hand, by adopting the EPC Network infrastructure we can leverage many of its applications. On the other hand, by providing interface-level compatibility it is possible to communicate WISSE and other EPC Network implementations. Nevertheless, we are conscious that our modification on the infrastructure is not standardized. In this sense, we are also working on a full redesign of the EPC Network, called the EPC Sensor Network [11], which will be proposed as an international standard. WISSE framework could obtain great benefits from the adoption of the EPC Sensor Network, as all the functionality at capturing, processing and storing time will be provided without additional considerations. Our present work, thus, constitutes the first steps towards the standardization of the sensor integration with the EPC Network.

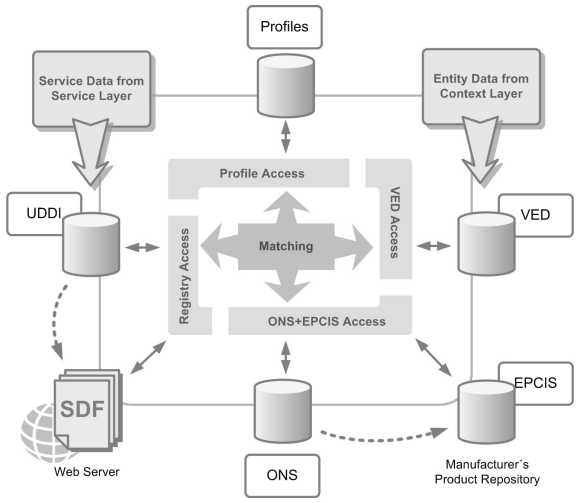

Figure 4. Middleware implementation elements

8. Implementation

In this section we explain the implementation of the concepts outlined during the previous sections and that constitute the WISSE middleware. Figure 4 depicts the elements involved. We implement the Service Manager and the Virtual Entity Database (VED). The former includes the Service Coordinator, local registry, profile lookup and EPC Network lookup modules.

For the VED and the manufacturer's EPCIS, we chose the databases provided as part of the Sun Microsystem's RFID Software. To implement the VED, which needs to support nodes, sensors and actuators, we added a few additional tables to the database, linking them to the main table containing EPCs as the primary key. To implement the association hierarchy of entity groups, we also used the EPCIS database to search parents and children in a recursive manner. Finally, we defined the edge devices as readers and store their IP information in order to reach them later with the entities' granted services. Another instance of an unmodified EPCIS was used to simulate the storage of manufacturers' EPC information. We also implemented an ONS server pointing to the manufacturer's EPCIS.

```
<service name="BulbClientService" version="0.1">
  <requirements type="mandatory">
    <sensors>
      <item type="TEMPERATURE" required="false"/>
      <item type="LIGHT" required="true"/>
      <item type="CD_CODE" required="true"/>
    </sensors>
    <actuators>
      <item type="BULB" required="true"/>
        <EPC>
          <product>
            <condition subject="FAMILY" operand="eq"
                object t="regulated"/>
          </product>
        </EPC>
      <item type="TV" required="false"/>
    </actuators>
  </requirements>
</service>
```

Figure 5. SDF file example

Web Services are good candidates for WISSE services due to their flexibility and increasing popularity. Moreover, the communication part of the current EPC Network standards is extensively based on Web Services. In order to provide the support for Web service discovery, we implemented the registry using a standard UDDI (Universal Description, Discovery and Integration [15]) database extended with a WISSE service taxonomy description (*tmodel*). This data structure provides a pointer to a Service Definition File (SDF - specified as a XML file) for each registered WISSE service. The SDF includes the requirement of the service in terms of sensors and actuators, and provides the possibility of specifying requirements in terms of identity. The current implementation allows indicating requirements about the serial number, the object class or the EPC manager part of the entities' EPCs that request the service. The SDF uses first order logic when comparing repository records. Figure 5 shows an example of a possible SDF file of the bulb client mentioned in Section VI. Here, the service specifies that the entity that holds the BULB actuator should belong to a product of the FAMILY "regulated", indicating that the

bulb actuation needs to be power-regulated to control the light intensity.

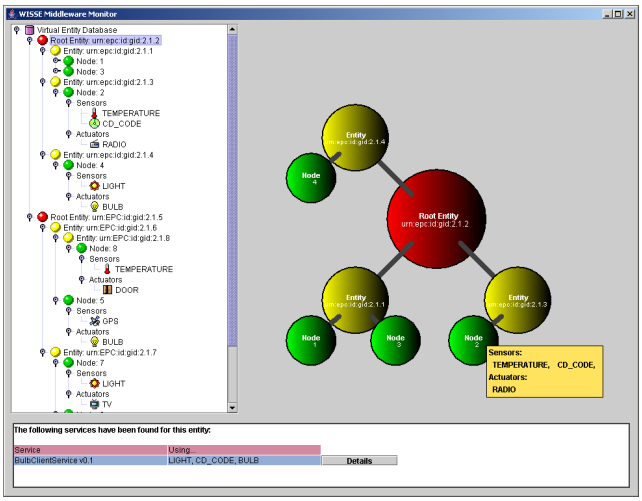

Figure 6. Middleware software screenshot

In order to monitor the middleware software, we also developed a GUI monitoring application. The application presents graphically the values from the various databases and also from the matching process. Figure 6 shows a screenshot which includes the entities for the bulb client example of section 5. The bulb itself is presented with the EPC `urn:epc:id:gid:2.1.4`, while the portable CD player is presented with the EPC `urn:epc:id:gid:2.1.3`. Other nodes and entities are also included to show the functionality of the GUI.

9. Conclusion

In this paper we show that merging RFID with sensor information not only provides more meaningful context but also enhances the context organization. By using these properties, we build a middleware that processes the context and retrieves services according to it. Additionally, we propose the use of the EPC Network for managing context information. We show how this approach provides a natural way of processing EPC-based context and gives added-value services originated in the EPC Network features. Finally our implementation demonstrates that it is possible to build WISSE infrastructure for Smart Spaces on top of the EPC Network and Web Services.

Acknowledgment

This work was supported in part by the "Development of Sensor tag and Sensor node technology for RFID/USN" project of ETRI, through the IT Leading R&D Support Programs of MIC, South 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] Hideyuki Takahashi, Takuo Suganuma and Norio Shiratori, *AMUSE: An Agent-based Middleware for Context-aware Ubiquitous Services*, ICPADS'05, July 20-22 2005

[4] *Sun JavaTM System RFID Software Documentation*, February 2006, http://sun.java.net/rfid-sensors

[5] Kiran Kumar, Salim Hariri, Nader V. Chalfoun, *Autonomous Middleware Framework for Sensor Networks*, ICPS 2005, 11-14 July 2005

[6] Anand Ranganathan and Roy H. Campbell, *A Middleware for Context-Aware Agents in Ubiquitous Computing Environments*, Middleware 2003, LNCS 2672, pp. 143161, 2003.

[7] Harry Lik Chen, *An Intelligent Broker Architecture for Pervasive Context-Aware Systems*, Thesis Disertation, University of Maryland, 2004

[8] Stephen S. Yau and Fariaz Karim, *Context-Sensitive Middleware for Real-time Software In Ubiquitous Computing Environments*, ISORC 2001

[9] Itiro Siio, Jim Rowan, Noyuri Mima, Elizabeth Mynatt, *Digital Decor: Augmented Everyday Things*, Graphics Interface 2003: 159-166

[10] F. Kawsar, Kaori Fujinami, Tatsuo Nakajima, *Experiences with Developing Context-Aware Applications with augmented artifacts*, in Proceedings of Ubicomp 2005, Tokyo, Japan

[11] Jongwoo Sung, Tomás Sánchez López, Daeyoung Kim, *The EPC Sensor Network for RFID and USN Integration Infrastructure*, to appear in the WiP session of Percom 2007, New York, March 19-21, 2007

[12] EPC Global Ratified Standard, *The Application Level Events (ALE) Specification*, Version 1.0, S. 2005

[13] EPC Global Ratified Standard, *Object Naming Service (ONS)*, Version 1.0, October 2005.

[14] EPC Global Working Draft, *EPC Information Services (EPCIS)*, Version 1.0, September 2005

[15] Claus von Riegen et all, *UDDI Version 2.03 Data Structure Reference*, UDDI Committee Specification