SYLPH: An Ambient Intelligence Based Platform for Integrating Heterogeneous Wireless Sensor Networks

Dante I. Tapia, Ricardo S. Alonso, Fernando De la Prieta, Carolina Zato, Sara Rodríguez, Emilio Corchado, Javier Bajo, Juan M. Corchado

Abstract — **The significance that Ambient Intelligence (AmI) has acquired in recent years requires the development of innovative solutions. Nonetheless, the development of AmI-based systems requires the creation of increasingly complex and flexible applications. In this regard, the use of context-aware technologies is an essential aspect in these developments to perceive stimuli from the context and react upon it autonomously. This work presents a novel platform that defines a method for integrating dynamic and self-adaptable heterogeneous Wireless Sensor Networks (WSNs). This approach facilitates the inclusion of context-aware capabilities when developing intelligent ubiquitous systems, where functionalities can communicate in a distributed way. Furthermore, the information obtained must be managed by intelligent and self-adaptable technologies to provide an adequate interaction between the users and their environment. Agents and Multi-Agent Systems are one of these technologies. The agents have characteristics such as autonomy, reasoning, reactivity, social abilities and pro-activity which make them appropriate for developing dynamic and distributed systems based on AmI. This way, the integration of the platform with a Service-Oriented Multi-Agent architecture is proposed. Finally, conclusions and future work are presented.**

Manuscript received January 31, 2010. This research is funded through the *Junta de Castilla y León* (BU006A08) and the Spanish Ministry of Education and Innovation (CIT-020000-2008-2 and CIT-020000-2009-12). The authors would also want to thank the vehicle interiors manufacturer Grupo Antolin Ingeniería, S.A. for supporting the project through the MAGNO2008 – 1028 - CENIT Project funded by the Spanish Ministry of Science and Innovation.

D. I. Tapia is with the Department of Computer Science and Automatic. University of Salamanca. Plaza de la Merced, S/N, 37008, Salamanca, Spain. Phone: (+34) 923 294400 (Ext: 1525); fax: (+34) 923 294514; e-mail: dantetapia@usal.es.

R. S. Alonso is with the Department of Computer Science and Automatic. University of Salamanca. Plaza de la Merced, S/N, 37008, Salamanca, Spain. E-mail: ralorin@usal.es.

F. de la Prieta is with the Department of Computer Science and Automatic. University of Salamanca. Plaza de la Merced, S/N, 37008, Salamanca, Spain. E-mail: fer@usal.es.

C. Zato is with the Department of Computer Science and Automatic. University of Salamanca. Plaza de la Merced, S/N, 37008, Salamanca, Spain. E-mail: carol_zato@usal.es.

S. Rodríguez is with the Department of Computer Science and Automatic. University of Salamanca. Plaza de la Merced, S/N, 37008, Salamanca, Spain. E-mail: srg@usal.es.

E. Corchado is with the Department of Computer Science and Automatic. University of Salamanca. Plaza de la Merced, S/N, 37008, Salamanca, Spain. E-mail: escorchado@usal.es.

J. Bajo is with the Department of Computer Science and Automatic. University of Salamanca. Plaza de la Merced, S/N, 37008, Salamanca, Spain. E-mail: jbajope@usal.es.

J. M. Corchado is with the Department of Computer Science and Automatic. University of Salamanca. Plaza de la Merced, S/N, 37008, Salamanca, Spain. E-mail: corchado@usal.es.

I. Introduction

PEOPLE are becoming increasingly accustomed to living with more and more technology in the hopes of increasing their quality of life and facilitating their day-to-day living. However, there are situations where technology is difficult to handle or people lack the knowledge of how to use it. Ambient Intelligence (AmI) tries to adapt technology to people's needs by incorporating omnipresent computing elements that communicate ubiquitously amongst themselves [1], [2]. In addition, the continuous advancement in mobile computing makes it possible to obtain information about the context and to react physically to it in more innovative ways [3]. Therefore, it is necessary to develop new solutions capable of providing adaptable and compatible frameworks, allowing access to functionalities regardless of time and location restrictions.

One key aspect in any AmI-based system is the use of context-aware technologies. The context is defined as any information used to characterize the situation of an entity, which can be a person, a place or an object [4]. This information is important for defining the interaction between users and the technology that surrounds them. However, it is not enough to gather information about the context, but that information must be processed by self-adaptable and dynamic mechanisms and methods that can react independently of each particular situation that arises. In this sense, agents and Multi-Agent Systems (MAS) comprise one of the areas that can contribute expanding the possibilities of Ambient Intelligence [5]. Furthermore, most of the context information can be collected by distributed sensors throughout the environment and even by the users themselves. It is possible to distinguish between two types of sensor networks: wired and wireless. Wireless Sensor Networks (WSNs) are more flexible and require less infrastructural support than wired sensor networks [6]. Although there are plenty of technologies for implementing WSNs (e.g. ZigBee, Wi-Fi or Bluetooth), it is not easy to integrate devices from different technologies into a single network [7], [8]. The lack of a common architecture may lead to additional costs due to the necessity of deploying non-transparent interconnection elements amongst different networks [9]. Moreover, the developed elements are dependent on the application to which they belong, thus complicating their reutilization.

978-1-4244-8126-2/10/$26.00 ©2010 IEEE

This work describes the *Services laYers over Light PHysical devices* (SYLPH) platform. This platform is aimed at facilitating the development of AmI-based systems with context-aware capabilities by using dynamic and self-adaptable heterogeneous WSNs. Although there is currently a wide range of WSN technologies, most of them are not compatible with each other. SYLPH solves this problem by implementing a middleware that consists of additional layers added over the application layer of each WSN's stack. SYLPH implements an approach based on Service-Oriented Architectures (SOA). The platform provides a flexible distribution of resources and facilitates the inclusion of new functionalities in Ambient Intelligence environments. Unlike other SOA-WSNs approaches [10], [7], SYLPH allows both services and services directories to be embedded in devices with limited computational resources, regardless of the radio technology they use. Futhermore, SYLPH can be integrated with *Flexible User and ServIces Oriented multiageNt Architecture* (FUSION@) [11], an architecture that combines a SOA approach with intelligent agents for building AmI-based systems. Thus, context-aware information gathered by SYLPH WSNs can be used by intelligent applications based on agents that use reasoning mechanisms to adapt their behavior to the context.

Next, the problem description is introduced and it is explained why there is a need for defining a new platform. Later, the proposed platform and its integration with the FUSION@ architecture are described in Section III. Finally, Section IV depicts the conclusions and future lines of work.

II. Motivation and Problem Description

Ambient Intelligence proposes three essential concepts: ubiquitous computing, ubiquitous communication and intelligent user interfaces [1]. AmI-based systems must be dynamic, flexible, robust, adaptable to changes in context, scalable and easy to use and maintain. The development of AmI-based systems that integrate different subsystems demands the creation of complex and flexible applications. As the complexity of an application increases, it needs to be divided into modules with different functionalities. Since different applications could require similar functionalities, there is a trend towards the reutilization of resources that can be implemented as part of other systems. This trend is the best long-term solution and can be accomplished by using a common platform. However, it is difficult to carry out because the systems in which those functionalities are implemented are not always compatible with other systems.

An alternative to such an approach is the reimplementation of the required functionalities. Although it implies more development time, it is generally the easiest and safest solution. However, reimplementation can lead to duplicated functionalities and more difficult system migration. A distributed architecture provides more flexible ways to move functions to where actions are needed, thus obtaining better responses, autonomy, services continuity, and superior levels of flexibility and scalability than centralized architectures [2]. In addition, excessive centralization negatively affects system functionalities, overcharging or limiting their capabilities. For this reason, it is difficult for the system to dynamically adapt its behavior to changes in the infrastructure. Thus, distributed architectures look for the interoperability amongst different systems, the distribution of resources and the independence of programming languages [2].

One of the most prevalent alternatives in distributed architectures are agents and multi-agent systems. An agent can be defined as a computational system situated in an environment and is able to act autonomously in this environment to achieve its design goals [5]. Expanding this definition, we have that an agent is anything with the ability to perceive its environment through sensors and respond in the same environment through actuators, assuming that each agent may perceive its own actions and learn from the experience [12]. A multi-agent system is defined as any system composed of multiple autonomous agents with incomplete capabilities to solve a global problem, where there is no global control system, the data is decentralized and the computing is asynchronous [5]. As can be seen, the definition itself of an agent and a multi-agent system is closely related to Ambient Intelligence. There are several agent frameworks and platforms [13] which provide a wide range of tools for developing distributed multi-agent systems. The development of agents is an essential piece in the analysis of data from distributed sensors and gives those sensors the ability to work together and analyze complex situations, thus achieving high levels of interaction with humans [14]. Agent and multi-agent systems have been successfully applied to several Ambient Intelligence scenarios, such as education, culture, entertainment, medicine, robotics, etc. [15]. Furthermore, agents can use reasoning mechanisms and methods in order to learn from past experiences and to adapt their behavior according the context [11].

Nevertheless, multi-agent systems do not always cover the actual necessities of distributed systems. Thus, several developments consider the integration between agents and modern functional architectures, such as Service-Oriented Architectures (SOA) [16] [17]. These developments try to improve the distribution of the available resources, facilitate the reutilization of functionalities and optimize the compatibility amongst different platforms. In classic functional architectures the modularity and structure are oriented to the systems themselves. Modern functional architectures such as SOA allow functionalities to be created outside the system, as external services linked to it. The term “service” can be defined as a mechanism that facilitates the access to one or more functionalities (e.g. functions, network capabilities, etc.) [2]. Services are integrated through communication protocols that have to be used by applications to share resources in the network.

One of the key aspects for the construction of AmI-based systems is obtaining information from the context through sensor networks. The context includes information about the people and their environment. The information may consist of many different parameters such as location, the building status (e.g. temperature), vital signs (e.g. heart rhythm), etc. Sensor networks need to be fast and easy to install and maintain. Each element that forms part of a sensor network is called a node. In an AmI scenario, nodes must communicate directly with one another in a distributed way [9]. In a centralized architecture, most of the intelligence is located in a central node. That is, the central node is responsible for managing most of the functionalities and knowing the existence of all nodes in a specific WSN. That means that a node belonging to a certain WSN does not know about the existence of another node forming part of a different WSN, even though this WSN is also part of the system. Nonetheless, this model can be improved using a common distributed architecture where all nodes in the system can know about the existence of any other node in the same system regardless of the technology or interface they use or the sub-network to which they belong.

Fig. 1 shows a centralized versus a distributed model for integrating heterogeneous WSNs. A centralized model consists of a central component that gathers all the data forwarded by the nodes connected to it. The main component can be a computer with several wireless hardware interfaces (i.e. wireless network cards). Each of these interfaces is connected to one or more of the WSNs deployed in the system. One of the main problems in this model is that most of the intelligence of the system is centralized. That is, the central component is the responsible for knowing what nodes are in all WSNs. Thus, it gathers the required data from the nodes and, based on such data, it decides what commands will be sent to the each node. That means that a node belonging to a certain WSN does not know about the existence of another node forming part of a different WSN, although this WSN is also part of the system. This problem can be seen in Fig. 1a. For this reason, it is difficult for the system to dynamically adapt its behavior to the changes in the environment.

Nonetheless, this model can be improved using a common platform where all the nodes in the system can know about the existence of any other node in the same system no matter the technology they use. This is achieved by adding a middleware logical layer over the existing application layers on the nodes. Fig. 1b shows a distributed model for integrating heterogeneous WSNs. The code executing in a certain node can invoke functionalities (i.e. services) offered by any other node in the system, regardless the latter node is in the same WSN or not. This way, the central component now only has to act as a gateway amongst the distinct WSNs connected to it. Thus, it has not to keep track of either the nodes in the system or the functionalities they offer. Some nodes in the system can integrate services directories for distributing the registration and discovering of services. This way, a node can know about the existence of other nodes and services offered by them. Thus, it can directly communicate with other nodes in order to execute commands or gather data. Moreover, a node belonging to the system can react to a certain change in the environment with no need of having into account all the factors involved on the system. In addition, the node registration is done in the corresponding WSN and the service registration is maintained by the service directories. Therefore, the process of connecting a new node offering more functionalities to the system is performed in a dynamical way.

Some developments try to reach integration between WSNs by implementing some kind of middleware, which can be applied as reduced versions of virtual machines, middleware or agent approaches [18] [7] [19]. However, these developments require devices whose microcontrollers have large memory and high computational power, which increases costs and physical size. These drawbacks are very

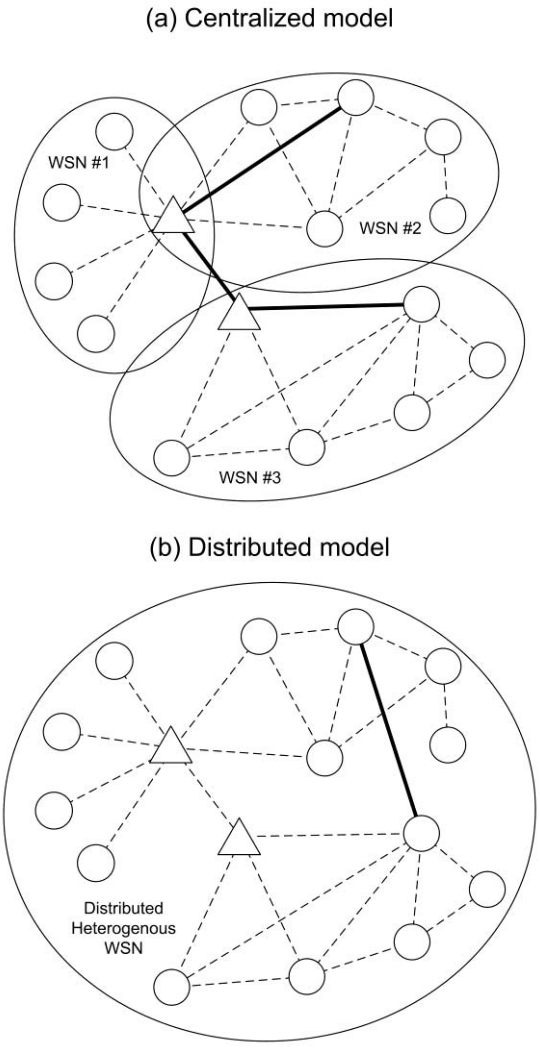

Fig. 1. A centralized versus a distributed approach for integrating heterogeneous WSNs.

important when it comes to WSNs, as it is essential to deploy applications with reduced resources and low infrastructural impact, especially in AmI scenarios. A service-oriented approach is adequate for implementing in WSNs since it allows the distribution of functionalities (i.e. services) into small modules that can be executed by devices with limited computational resources, such as wireless sensor nodes.

The SYLPH platform tackles some of these issues by enabling an extensive integration of WSNs and optimizing the distribution, management and reutilization of the available resources and functionalities in such networks. There are other developments that integrate WSNs and SOA. Reference [10] presents a SOA-based environment in which a WSN acts as service provider. Nevertheless, services are not implemented on each node, but offered by a centralized gateway. Reference [7] proposes a SOA-WSN approach that implements three additional layers. However, this approach needs special bridge nodes that implement the Universal Plug and Play standard for the interconnection of WSNs. Therefore, these developments do not consider the necessity of minimizing the overload of the services on the devices. In contrast, SYLPH allows the services to be directly embedded in the WSN nodes and invoked from other nodes either in the same network or another network connected to the former. It also focuses specifically on using devices with small resources to save CPU time, memory size and energy consumption. Furthermore, SYLPH can be integrated with FUSION@ [11], a multi-agent architecture, so that information from WSN nodes can be managed by intelligent agents running on an AmI-based multi-agent application. SYLPH is presented in detail in the following section.

III. Description of the SYLPH Platform

The *Services laYers over Light PHysical devices* (SYLPH) platform follows a SOA model [2] for integrating heterogeneous WSNs in AmI-based systems. The main objective is to distribute resources over multiple WSNs by modeling the functionalities as independent services. The SYLPH platform has been successfully applied in some Ambient Intelligence scenarios as a healthcare telemonitoring system [20]. The information gathered by SYLPH nodes is managed by intelligent agents by means of the integration of SYLPH with the FUSION@ multi-agent architecture. Thus, the agents running on FUSION@ can use reasoning mechanisms to adapt their behavior to the context information obtained through SYLPH nodes.

SYLPH covers aspects relative to services such as registration, discovering and addressing. Additionally, a node can invoke functionalities offered by any other node in the system, regardless of whether they are in the same WSN or not. Some nodes in the system can integrate services directories for distributing registration and discovering services. Node registration is done in the corresponding WSN (i.e. specific network) and service registration is maintained by multiple services directories. Thus, the process of connecting new nodes to the system is performed in a dynamic way. A node can know about the existence of other nodes and the services they offer. Therefore, it can directly communicate with other nodes to perform a specific service.

A SOA model was chosen because architectures based on this model are asynchronous and non-dependent on *context* (i.e. previous states of the system, which must not be confused with context-aware environments) [2]. Thus, devices working on them do not continuously take up processing time, consume less energy, and are free to perform other tasks.

SYLPH can be executed over multiple wireless devices independently of their microcontroller or the programming language they use. SYLPH works in a distributed way so that the application code does not have to reside almost completely on an only central node. SYLPH allows the interconnection of several networks from different wireless technologies, such as ZigBee or Bluetooth. Thus, a node designed over a specific technology can be connected to a node from a different technology. In this case, both WSNs are interconnected by means of a set of intermediate gateways connected to several wireless interfaces simultaneously. SYLPH allows applications to work in a distributed way and independently of the lower layers related to the WSNs formation (i.e. network layer) and the radio transmission amongst the nodes that conform them (i.e. data link and physical layers).

The services can be executed from multiple wireless devices. Given that neither developers nor users have to worry about what kind of technology each node in the system uses, the experience is transparent for everybody involved. This facilitates the inclusion of context-aware capabilities into AmI-based systems because developers can dynamically integrate and remove nodes on demand.

SYLPH implements an organization based on a stack of layers [6]. Each layer in one node communicates with its peer in another node through an established protocol. In addition, each layer offers specific functionalities to the immediately upper layer in the stack. These functionalities are usually called *interlayer services*, which must not be confused with the services invoked from node to node. These *interlayer services* are abstract functions and independent of the implementation of the platform. The SYLPH layers are added over the existent application layer of each WSN stack, allowing the platform to be reutilized over different technologies. The structure of SYLPH will now be described.

1) **SYLPH Message Layer (SML).** The SML offers the upper layers the possibility of sending asynchronous messages between two nodes through the *SYLPH Services Protocol* (SSP). These messages specify the source and destination nodes and the service invocation

in a *SYLPH Services Definition Language* (SSDL) format. The SSDL describes the service itself and the parameters to be invoked. The SML not only transports the services invocations over the network, but also the services registration and search functions.

2) **SYLPH Application Layer (SAL).** The SAL allows different nodes to directly communicate with each other using SSDL requests and responses that will be delivered in encapsulated SML messages following the SSP. The SAL implements the service code (i.e. firmware) from within each node, allowing each one to communicate with the SYLPH platform and invoke services located in other nodes. Moreover, there are other *interlayer services* for registering services or finding services offered by other nodes. In fact, these *interlayer services* for registering and searching services call other *interlayer services* offered by the *SYLPH Services Directory Sub-layer* (SSDS). Therefore, the SAL can use the *interlayer services* of the SML either directly or through the SSDS.
3) **SYLPH Services Protocol (SSP).** The SSP is the internetworking protocol of the SYLPH platform. SSP has functionalities similar to those of the Internet Protocol (IP). That is, it allows sending packets of data from one node to another node regardless of the WSN to which each one belongs. Every node has a unique SSP 32-bit address in the SYLPH network. Therefore, a SSP packet includes a header that describes the SSP addresses of the source node and the destination node, as well as information for managing transmissions that involve multiple SSP packets (i.e. number of SSP packet and remaining bytes).
4) **SYLPH Services Definition Language.** The SSDL is the IDL (*Interface Definition Language*) used by SYLPH. Unlike other IDLs such as WSDL (*Web Services Definition Language*) [21], SSDL does not use as many intermediate separating tags, and the order of its elements is fixed. SSDL has been specifically designed to work with limited computational resources nodes. Nodes can request the SSDS for the location of services and their specifications using SSDL.
5) **SYLPH Services Directory Sub-layer (SSDS).** The SSDS creates dynamical services tables to locate and register services in the network. A node that stores and maintains services tables is called *SYLPH Directory Node* (SDN). These tables are made up of a list of services entries, each of which includes the description of a service in SSDL format and the SSP address of the node that offers the service. In addition, each entry stores additional information about the service whose location and description is maintained in the network. Such information includes, for instance, a *Quality of Service* (QoS) rate and the last time the SDN checked if the service was available. A node in the network can make a request to the SDN to know the location (i.e. network address) of a certain service. Requests are packed in SML messages and must follow the SSP. The SSDS is also used by the SAL when registering a new service.

A. SYLPH Services

The behavior of SYLPH is in essence similar to the one of any other service oriented architecture. However, SYLPH has several characteristics and functionalities that make it different to other models.

First, a service registers itself on the SDN and informs its location in the network, the parameters it requires and the type of returned value after its execution. In order to do that, it is used SSDL which has been created to work with limited resources nodes. SSDL is the IDL (*Interface Definition Language*) used by SYLPH. Distributed architectures use an IDL in order to enable communication between software components regardless their programming language or hardware implementation. Unlike other IDLs as WSDL, based on XML and used on Web Services [21], SSDL does not use almost any intermediate separating tags and its services descriptions are short binary data sequences.

The reason for these constraints is to reduce processing in the devices microcontrollers. Using a simple IDL allows, consequently, utilizing nodes with fewer resources, less power consumption and lower cost. In most cases it is enough with a few float point data for informing the status of a sensor. Thus, most service definitions require only a few bytes. SSDL considers the basic types of data (e.g. integer, float or boolean), allowing more complex data structures as variable length arrays or character strings. In this way, SSDL is flexible enough to specify more complex services if required.

Once the service has been registered in the SDN, it can be invoked by any application by means of SYLPH. Both the SDN and the services can be stored in any node of the WSN or in other subsystem connected to the WSN. This system can be, for instance, a simple personal computer connected through a USB port to a wireless interface. Thus, developers decide which nodes or subsystems will implement each part of the distributed application. Any node in the network can ask the SDN for the location of a determined service and its specification using SSDL. This aspect is described in the following section.

With the aim of the architecture to be the more distributed as possible, it is allowed to be more than one SDN in the same network, so that can exist redundancy or services organized in different directories. The SDN can be stored in a node of the network, with a memory external to the microcontroller if necessary, or be contained on a computationally higher machine connected to the WSN, as is the case of a data server or a personal computer with wireless connection.

B. SYLPH Services Definition Language

The next example shows the use of SSDL to define a

SYLPH service. There is defined a simple service called *registerServiceOnFloodAlarm*. This service is stored in a flood sensor device that belongs to a WSN with the SYLPH architecture running over it. The service can be invoked by any other node in the SYLPH network to register another service that will act as a callback. Thus, when the flood sensor obtains a read over the specified threshold, the node where it is stored will invoke the service labeled as *callback* in the interface definition:

```
service registerServiceOnFloodAlarm {
   input {
      uint16_t threshold;
      servicepoint callback {
         output {
            boolean status; }; }; };
   output {
      boolean status; }; };
```

a SSDL header which specifies the SSDL data length and the kind of frame it is (registration, definition, invocation or response). As can be seen, there is an *outputs mark* for denote the input parameters end there and the output parameters follow it. In the example of the *registerServiceOnFloodAlarm* service showed, a *service callback* is described as an input parameter. For specifying that issue, there are some service *start* and *end marks*. As the service callback has no input parameters, the *outputs mark* is the first field inside its definition in the parameters description. Once the invoker node knows the service definition, it can call the service sending a SSP frame to the node which stores the service. This frame (Fig. 2b) does not need marks inside it, because the input parameters have to follow the specified order. Thus, the SSDL combines ease of parsing with flexibility in the type and size of the utilized parameters. The SSP header includes the destination node SSP address. In the response frame there is the only output parameter (Fig. 2c).

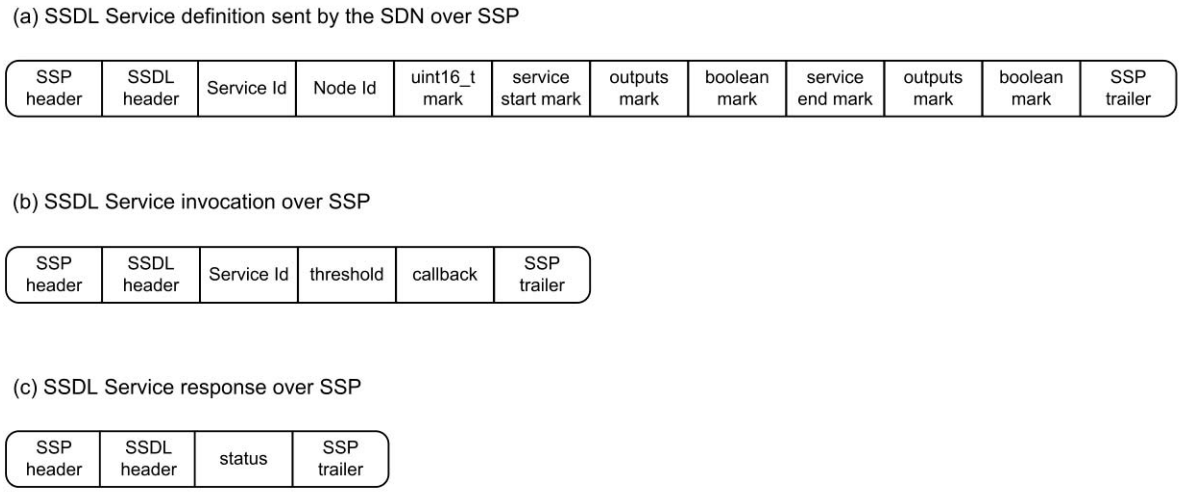

Fig. 2. Examples of SSDL frames over SSP.

In fact, this representation of the SSDL syntax is the one used by developers to specify the services in the firmware of the devices attached to SYLPH architecture. After specifying the service by means of SSDL human-readable syntax, developers translate definitions to specific code for the target language (e.g. C or nesC) and microcontroller where service will run. When the node registers its service in a SDN, SYLPH layers do not transmit the human-readable SSDL message, but a more compact array of bytes which describes the service and how to invoke it from other nodes. Fig. 2 shows the SSDL frames involved in the *registerServiceOnFloodAlarm* service definition (a), invocation (b) and response (c) when transmitted over SSP. When a node asks a SDN for the service definition, the SDN answers with a frame as showed in Fig. 2a. Such frame describes the service identification, the address of the node which stores the service, the definition of the input and output parameters and the QoS offered by the service. It has

C. SYLPH Gateways

As mentioned above, a node in a specific type of WSN (e.g. ZigBee) can directly communicate with a node in another type of WSN (e.g. Bluetooth). Therefore, several heterogeneous WSNs can be interconnected through a *SYLPH Gateway*. A SYLPH Gateway is a device with several hardware network interfaces (e.g. a Wi-Fi network card), each of which is connected to a distinct WSN. As an IP gateway, a SYLPH Gateway does not need to implement the layers over the SML. The SYLPH Gateway stores routing tables for forwarding SSP packets amongst the different WSNs with which it is interconnected. The information transported in the SSP header is enough to route the packets to the corresponding WSN. If several WSNs belong to the SYLPH network, there is no difference between invoking a service stored in a node in the same WSN or in a node from a different WSN. For example, if a

source node invokes a service stored in a destination node located in a different WSN, the source node looks for the service in a SDN present in the WSN to which it belongs. In fact, the entry stored in the services table of that SDN points to the SSP address of the SYLPH Gateway. When the source node invokes the service in the destination node, the SYLPH Gateway forwards the call message to the destination node through its hardware interface connected to the WSN where the destination node is located

D. Integration of SYLPH and FUSION@

In order to interact with a SYLPH network from a system that is not made up of WSNs, it is proposed to use FUSION@ [11], a Multi-Agent architecture for distributed services and applications. FUSION@ (*Flexible User and ServIces Oriented multiageNt Architecture*) proposes a new perspective, where Multi-Agent Systems and SOA-based services are integrated to provide ubiquitous computation, ubiquitous communication and intelligent interfaces facilities. The FUSION@ framework defines four basic blocks: *Applications*, *Services*, *Agents Platform* and *Communication Protocol*. This framework has been designed following the SOA model, but adding the applications block which represents a fundamental part in Ambient Intelligence: the interaction with users. These blocks provide all the functionalities of the architecture. There are predefined agents that provide the basic functionalities of the architecture: *CommApp Agent*, *CommServ Agent*, *Directory Agent*, *Supervisor Agent*, *Security Agent*, *Admin Agent* and *Interface Agent*. Interface Agents were designed to be embedded in users' applications and are simple enough to allow them to be executed on mobile devices, such as cell phones or PDAs. FUSION@ exploits the agents' characteristics to provide a robust, flexible, modular and adaptable solution that can cover most requirements of a wide diversity of Ambient Intelligence projects.

This way, we have designed two agents in SYLPH in order to interact with FUSION@: *SylphInterface Agent* and *SylphMonitor Agent*. The *SylphInterface Agent* allows the rest of agents to discover and invoke services offered by SYLPH WSN nodes. Moreover, the *SylphInterface Agent* can offer services to the wireless nodes. In order to do this, the WSN node in the FUSION@-SYLPH gateway stores services entries on its SSDS table. As can be seen in Fig. 3, the *SylphInterface Agent* performs as a broker between the SYLPH network and the FUSION@ architecture. The *SylphMonitor Agent* allows the agents platform to monitor the state and operation of the SYLPH network. Thus, *SylphMonitor Agent* monitors all the traffic (i.e. service invocations, responses, registrations or searches) in the SYLPH network. It is necessary for the nodes to operate in debug mode, so that every time a node invokes a service it also invokes a monitoring service on a node connected to the SYLPH WSN node in the gateway. The node gathers all the invocations and forwards them to the *SylphMonitor Agent* running on the agents platform. The same process is done for service responses, searches and registrations. The *SylphMonitor Agent* makes it possible to observe when a node is searching for a certain service in the network, the

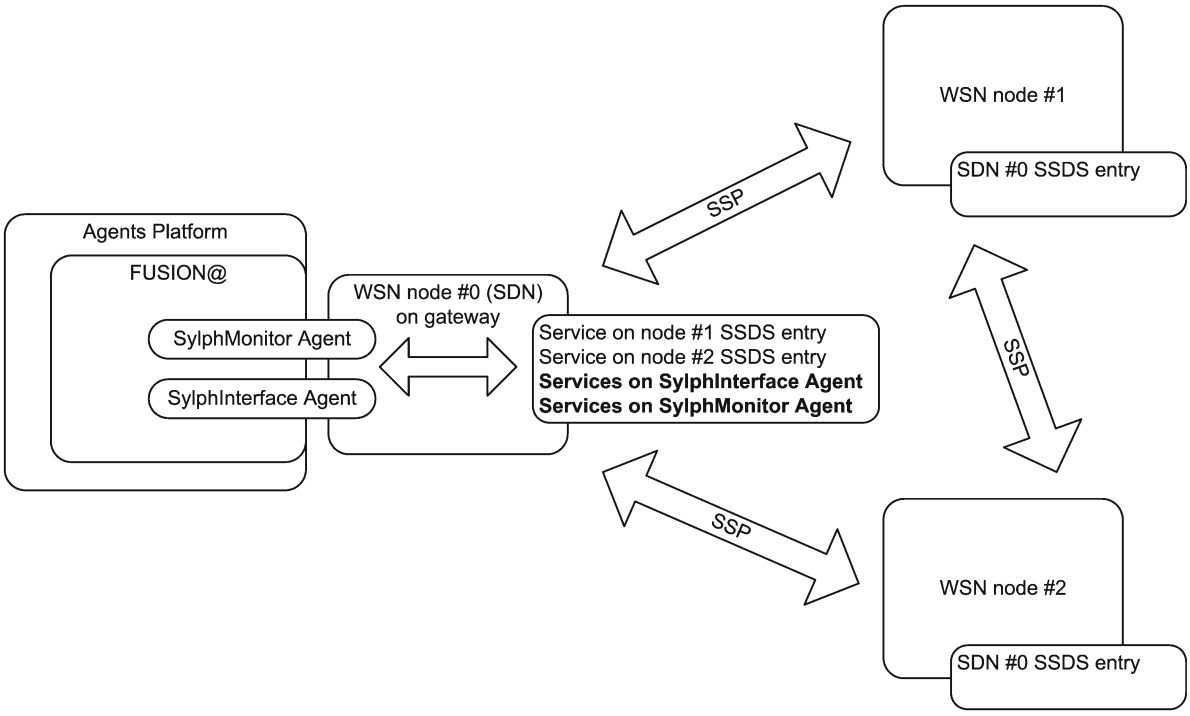

Fig. 3. Interaction between SYLPH and FUSION@.

services offered by the nodes, and the contents of the SSDS entries tables stored in the SDNs.

IV. Conclusions and Future Work

Ambient Intelligence is an emerging multidisciplinary area based on ubiquitous computing that proposes new ways of interaction between people and technology, adapting them to the needs of users and their environment. Ambient Intelligence improves the quality of life of the people providing them with easier and more efficient ways to communicate and interact with other people and systems. Ambient Intelligence based applications involve complex design and implementation stages. This is because such applications must be highly distributed, dynamic and scalable, as well as ubiquitous for users from both software and hardware point of views. One of the most successful approaches to Ambient Intelligence is agents and multi-agent systems.

The integration of SYLPH and the FUSION@ multi-agent architecture allows developing AmI-based applications where context information gathered by heterogeneous WSN nodes is managed by intelligent agents. These intelligent agents can use reasoning mechanisms and methods in order to learn from past experiences and to adapt their behavior according the context. The use of a SOA-based approach provides a flexible distribution of resources and facilitates the inclusion of new functionalities in highly dynamic environments. Furthermore, SYLPH allows integrating heterogeneous WSNs in a distributed way. Thus, functionalities are modeled as independent services offered by nodes (i.e. wireless devices) in the network. These services can be invoked by any node in the SYLPH infrastructure, regardless the physical WSN which they belong. In addition, SYLPH nodes do not need large memory chips or fast microprocessors. The easy deployment of SYLPH-based systems reduces the implementation costs in terms of development and infrastructure support.

Future work includes the development of new applications for Ambient Intelligence scenarios through the implementation of multi-agent systems based on the use of heterogeneous WSNs under the SYLPH platform. Surveillance and educational applications are our most immediate challenges. Furthermore, the next stage of the union between FUSION@ and SYLPH is being designed. This stage is aimed at considering wireless nodes as *hardware agents*. This way, there will be no distinction amongst software agents and hardware agents in the platform under development. Therefore, the platform will run software agents and hardware agents that offer services to other agents and applications, regardless if the agent is a piece of code or a wireless sensor. Consequently, it will be possible to build Ambient Intelligence scenarios where users cannot distinguish what part of the system is interacting with them, achieving a higher level of ubiquitous and pervasive computing.

References

[1] E. Aarts and R. Roovers, "Embedded system design issues in Ambient Intelligence," *Ambient Intelligence: impact on embedded system design*, Kluwer Academic Publishers, 2003, pp. 11–29.

[2] K. Lyytinen and Y. Yoo, "Introduction," *Commun. ACM*, vol. 45, 2002, pp. 62–65.

[3] G.T. Jayaputera, A. Zaslavsky, and S.W. Loke, "Enabling run-time composition and support for heterogeneous pervasive multi-agent systems," *The Journal of Systems & Software*, vol. 80, 2007, pp. 2039–2062.

[4] A.K. Dey and G.D. Abowd, *"Towards a better understanding of context and context-awareness,"* CHI 2000 workshop on the what, who, where, when, and how of context-awareness, 2000, pp. 304–307.

[5] M. Wooldridge, *An Introduction to MultiAgent Systems*, Wiley, 2009.

[6] J. Sarangapani, *Wireless Ad hoc and Sensor Networks: Protocols, Performance, and Control*, CRC, 2007.

[7] M. Marin-Perianu, N. Meratnia, P. Havinga, L. de Souza, J. Muller, P. Spiess, S. Haller, T. Riedel, C. Decker, and G. Stromberg, "Decentralized enterprise systems: a multiplatform wireless sensor network approach," *Wireless Communications, IEEE*, vol. 14, 2007, pp. 57-66.

[8] Jaekyu Cho, Yoonbo Shim, Taekyoung Kwon, and Yanghee Choi, "SARIF: A novel framework for integrating wireless sensor and RFID networks," *Wireless Communications, IEEE*, vol. 14, 2007, pp. 50–56.

[9] S. Mukherjee, E. Aarts, R. Roovers, F. Widdershoven, and M. Ouwerkerk, *Amiware: Hardware Technology Drivers of Ambient Intelligence*, Springer, 2006.

[10] A. Malatras, A. Asgari, and T. Bauge, "Web Enabled Wireless Sensor Networks for Facilities Management," *Systems Journal, IEEE*, vol. 2, 2008, pp. 500-512.

[11] J.M. Corchado, J. Bajo, Y. de Paz, and D.I. Tapia, "Intelligent environment for monitoring Alzheimer patients, agent technology for health care," *Decision Support Systems*, vol. 44, 2008, pp. 382–396.

[12] S.J. Russell, P. Norvig, J.F. Canny, J. Malik, and D.D. Edwards, *Artificial Intelligence: a modern approach*, Prentice Hall Englewood Cliffs, NJ, 1995.

[13] F. Bellifemine, A. Poggi, and G. Rimassa, "*JADE–A FIPA-compliant agent framework*," 1999, pp. 97–108.

[14] F. Pecora and A. Cesta, "DCOP FOR SMART HOMES: A CASE STUDY," *Computational Intelligence*, vol. 23, 2007, pp. 395–419.

[15] D. Tapia, J. Bajo, and J. Corchado, *"Distributing Functionalities in a SOA-Based Multi-agent Architecture,"* 7th International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS 2009), 2009, pp. 20–29.

[16] E. Cerami, *Web Services Essentials: Distributed Applications with XML-RPC, SOAP, UDDI & WSDL*, O'Reilly Media, Inc., 2002.

[17] L. Ardissono, G. Petrone, and M. Segnan, "A conversational approach to the interaction with Web Services," *Computational Intelligence*, vol. 20, 2004, pp. 693–709.

[18] Min Chen, S. Gonzalez, and V. Leung, "Applications and design issues for mobile agents in wireless sensor networks," *Wireless Communications, IEEE*, vol. 14, 2007, pp. 20–26.

[19] P. Schramm, E. Naroska, P. Resch, J. Platte, H. Linde, G. Stromberg, and T. Sturm, "A Service Gateway for Networked Sensor Systems," *IEEE Pervasive Computing*, vol. 3, 2004, pp. 66–74.

[20] J.M. Corchado, J. Bajo, D.I. Tapia and A. Abraham, "Using Heterogeneous Wireless Sensor Networks in a Telemonitoring System for Healthcare," *IEEE Transactions on Information Technology in Biomedicine*, vol. 14, Mar. 2010, pp. 234–240.

[21] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana, "Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI," *IEEE Internet Computing*, vol. 6, Apr. 2002, pp. 86–93.