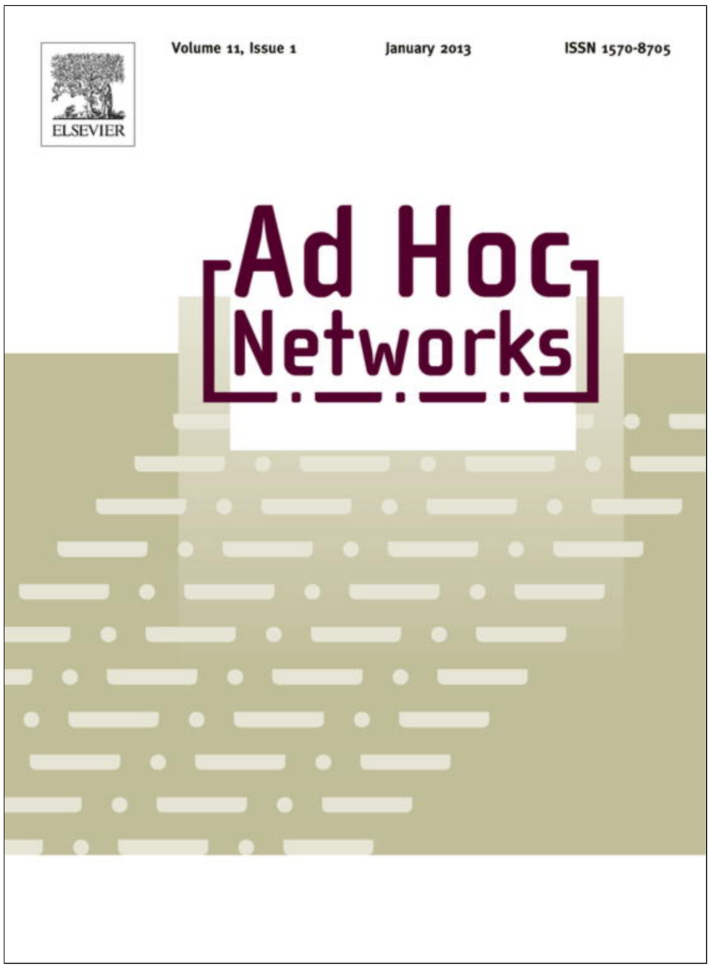
Volume 11, Issue 1
January 2013
ISSN 1570-8705
ELSEVIER
Ad Hoc Networks

Ad Hoc Networks 11 (2013) 151–166

Contents lists available at SciVerse ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

Implementing a hardware-embedded reactive agents platform based on a service-oriented architecture over heterogeneous wireless sensor networks

Ricardo S. Alonso *, Dante I. Tapia, Javier Bajo, Óscar García, Juan F. de Paz, Juan M. Corchado

Department of Computer Science and Automation, University of Salamanca, Plaza de la Merced s/n, 37008 Salamanca, Spain

ARTICLE INFO

Article history:
Received 6 November 2011
Received in revised form 27 March 2012
Accepted 29 April 2012
Available online 30 May 2012

Keywords:
Distributed architectures
Multi-agent systems
Heterogeneous wireless sensor networks
Embedded agents
Case-Based Planning

ABSTRACT

Wireless Sensor Networks (WSNs) represent a key technology for collecting important information from different sources in context-aware environments. Unfortunately, integrating devices from different architectures or wireless technologies into a single sensor network is not an easy task for designers and developers. In this sense, distributed architectures, such as service-oriented architectures and multi-agent systems, can facilitate the integration of heterogeneous sensor networks. In addition, the sensors' capabilities can be expanded by means of intelligent agents that change their behavior dynamically. This paper presents the *Hardware-Embedded Reactive Agents* (HERA) platform. HERA is based on *Services laYers over Light PHysical devices* (SYLPH), a distributed platform which integrates a service-oriented approach into heterogeneous WSNs. As SYLPH, HERA can be executed over multiple devices independently of their wireless technology, their architecture or the programming language they use. However, HERA goes one step ahead of SYLPH and adds reactive agents to the platform and also a reasoning mechanism that provides HERA Agents with Case-Based Planning features that allow solving problems considering past experiences. Unlike other approaches, HERA allows developing applications where reactive agents are directly embedded into heterogeneous wireless sensor nodes with reduced computational resources.

1. Introduction

Nowadays, there is a wide range of devices for gathering context information about both the environment and the users [1]. In this sense, Wireless Sensor Networks (WSNs) are used for collecting the information needed by intelligent environments, whether in home automation, industrial applications or even farming, among many others [2]. There are plenty of technologies for implementing WSNs, such as ZigBee, Wi-Fi or Bluetooth. Nonetheless, it is not easy to integrate devices from different technologies into a single network [1]. The lack of a common architecture may lead to additional costs due to the necessity of deploying non-transparent interconnection elements among the different networks [3]. Moreover, the developed elements can be dependent on the application to which they belong, thus complicating their reutilization.

Therefore, it is necessary to develop innovative solutions that integrate different approaches to create flexible and adaptable systems. In this sense, the deployment of distributed architectures is presented as a solution to such problems [4]. One of the most prevalent alternatives in distributed architectures is Multi-Agent Systems (MASs) [5]. A distributed agent-based architecture provides more flexible ways to move functions to where actions are needed, thus obtaining better responses at execution time,

* Corresponding author. Tel.: +34 923 294400x1525; fax: +34 923 294514.

E-mail addresses: ralorin@usal.es (R.S. Alonso), dantetapia@usal.es (D.I. Tapia), jbajope@usal.es (J. Bajo), oscgar@usal.es (Ó. García), fcofds@usal.es (J.F. de Paz), corchado@usal.es (J.M. Corchado).

1570-8705/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.adhoc.2012.04.013

autonomy, services continuity and superior levels of flexibility and scalability than centralized architectures [6]. Furthermore, the sensors' capabilities can be enhanced by means of intelligent agents, changing dynamically their behavior and personalizing their reactions [7].

The main objective of the work presented in this paper is to design and build a new platform that allows developers to take advantage of the use of intelligent agents directly embedded into nodes belonging to heterogeneous wireless sensor networks. In this sense, this paper describes the *Hardware-Embedded Reactive Agents* (HERA) platform. HERA is an evolution of the *Services laYers over Light PHysical devices* (SYLPH) platform [8–10], which allows developers to use dynamic and self-adaptable heterogeneous wireless sensor networks following a Service-Oriented Architecture (SOA) approach [4]. Unlike other approaches, SYLPH allows the interconnection of wireless sensor networks based on different radio and link technologies [10]. As SYLPH, HERA focuses specially on heterogeneous devices with reduced resources to save CPU time, memory size and power consumption. However, HERA goes one step ahead of SYLPH and adds reactive agents [11] and a reasoning mechanism to the platform, extending its context-aware features. In HERA, unlike other approaches, agents are directly embedded into the WSN nodes and their services can be invoked from other nodes in the same WSN or other WSN connected to the former one, no matter the radio technology they use. Furthermore, HERA incorporates a reasoning mechanism based on the Case-Bases Planning model [29] that allow solving problems by using solutions to similar past problems. Solutions are stored into a case memory, which the mechanism can consult to find better solutions for new problems. HERA Agents use this mechanism to learn from past experiences and to adapt their behavior according to the context information.

The remainder of the paper is organized as follows. In Section 2 we present the problem description that essentially motivated the development of SYLPH and HERA. In Section 3 the main characteristics and components of SYLPH are briefly depicted, while Section 4 presents the principal features that HERA adds over the SYLPH platform, including the HERA Agents and the Case-Based Planning mechanism. After that, some experiments aimed at testing the HERA performance are described in Section 5, including the implementation of HERA in a real scenario. Finally, the conclusions and the future lines of work are presented in Section 6.

2. Problem description and related work

Smart environments must take into account the information about the context, which can be collected by sensor networks [12]. This context information may consist of many different parameters about the people and their environment, such as the users' location, their heart rhythm, or the ambient temperature, among many others. In a real scenario, all these sensors can belong to wireless sensor nodes from different architectures or wireless technologies, forming together which is usually known as a heterogeneous wireless sensor network. In a centralized heterogeneous WSN architecture, most of the intelligence is located in a central node. That is, the central node is responsible for managing most of the features and knowing the existence of all nodes in each WSN in the system. 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. Distributed architectures such as Service-Oriented Architectures (SOAs) [13] or Multi-Agent Systems (MASs) [5] improve the distribution of the available resources, facilitate the reutilization of functionalities and optimize the compatibility among different platforms. In this sense, the SYLPH platform [9,10,14] was designed to address these challenges as an innovative platform for integrating heterogeneous WSNs in Ambient Intelligence (AmI) systems [15], implementing an approach based on service-oriented architectures [4,13,16,17].

Once SYLPH solves the problem of distributing resources over heterogeneous wireless sensor networks, the next challenge is to embed intelligent agents into the same heterogeneous wireless sensor nodes. At this point, it is worth mentioning again that SYLPH does not include by itself the support of agents or reasoning mechanisms. An agent can be defined as a computational system situated in an environment and able to act autonomously in this environment to achieve its design goals [5]. Expanding this definition, an agent is anything with the ability to perceive its environment through sensors, and to respond in the same environment through actuators, assuming that each agent may perceive its own actions and learn from the experience [18]. There are several agent frameworks and platforms [19–21] that provide a wide range of tools for developing distributed multi-agent systems. The development of agents is an essential component 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 [22–27]. Furthermore, agents can use reasoning mechanisms and methods in order to learn from past experiences and to adapt their behavior according to the context, such as Case-Based Reasoning (CBR) and Case-Based Planning (CBP) [28,29]. CBR and CBP mechanisms solve new problems by adapting solutions that have been used to solve similar problems in the past, and learn from each new experience [28, 29]. In CBP, the proposed solution for solving a given problem is a plan, which is generated by taking into account the plans applied for solving similar past problems [29].

Unfortunately, the fusion of the multi-agent technology and wireless sensor networks is not easy due to the difficulty in developing, debugging and testing distributed applications for devices with limited resources [30]. The interfaces developed for these distributed applications are either too simple or, in some cases, do not even exist, which even further complicates their maintenance. Even so, there are other works that try to integrate multi-agent systems and wireless sensor networks [31–33].

ActorNet [31] is a study that describes a mobile agent platform for WSNs. ActorNet provides an abstract environment for mobile code oriented to light objects over WSNs. ActorNet platform defines as its top layer an actor language interpreter. Likewise, the platform provides services such as virtual memory management and blocking input–output operations. Thus, ActorNet allows a wide range of dynamic applications, including customized queries and aggregation functions, in the sensor network platform.

Baker et al. [32] present the integration of an agent-based WSN within an existing MAS focused on condition monitoring. In this research, it is used SubSense, a multi-agent middleware platform developed to allow condition monitoring agents to be deployed onto a WSN. The architecture of the SubSense platform is based on the model defined by FIPA (Foundation for Intelligent Physical Agents), but customized so that agents are embedded into sensor nodes. SubSense platform is implemented over 512 KB RAM SunSPOT sensor nodes using the Java Mobile Edition (J2ME).

Other works that relate multi-agent systems and WSNs talk about *Mobile Agents based on WSN* (MAWSN). Zboril et al. [33] proposes WSageNt, a platform that is implemented through mobile agents running on wireless sensor nodes. One key feature of this platform is a module for an agent control language interpretation. This language is presented as an original low-level control language known as Agent Low Level Language (ALLL). This research poses that in WSN-based agent platforms the resources limitations of sensor motes do not allow affording its development as an ordinary agent platform that should accomplish the FIPA specifications.

However, these studies [31–33] have not been designed to work with heterogeneous wireless sensor networks. These approaches do not take into account the use of such heterogeneous WSNs and they are focused on working with sensor nodes that use just an only radio technology. Because HERA is based on SYLPH, it allows devices from different radio and networks technologies to coexist in the same distributed network. In addition, HERA platform can run on lightweight sensor nodes with just 8 KB RAM, while other approaches as SubSense [32] require nodes with 512 KB RAM. Besides, in the design of HERA, it has been mainly aimed to address context-awareness and ubiquitous computing, while other existing approaches are not specially centered on dealing with these requirements. Furthermore, HERA includes a Case-Based Planning (CBP) mechanism which allows the agents to make use of past experiences to create better plans and achieve their goals, while SYLPH does not, as it is not based on agents, but only services.

In the next section, the main components and the basic operation of the SYLPH platform, on which HERA is based, is briefly described. After that, Section 4 presents the novelties that HERA offers with regard to SYLPH.

3. The SYLPH platform

HERA is an evolution of *Services laYers over Light PHysical devices* (SYLPH) [9]. In this section, the SYLPH platform is only briefly depicted, as the objective of this paper is to describe the HERA platform. For a more extended description of the SYLPH platform, please consult previous publications [9,10,14]. The SYLPH platform follows a SOA model [4] for integrating heterogeneous WSNs in AmI-based systems. SYLPH focuses specifically on devices with small resources in order to save CPU time, memory size and energy consumption. There have been other attempts to integrate WSNs and a SOA approach [34]. In SYLPH, services are directly offered by the wireless sensor nodes that are part of the platform. In the same way, any node in the platform can directly invoke a SYLPH service offered by other node in the platform, no matter if both nodes are in the same physical wireless network or not. SYLPH provides the possibility of connecting wireless sensor networks based on different radio and link technologies, whereas other approaches do not. 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 a set of intermediate devices, called SYLPH Gateways and described in Section 3.4, which are simultaneously connected to several wireless interfaces.

3.1. Main components of the SYLPH platform

SYLPH implements an organization based on a stack of layers [2]. 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. The SYLPH layers are added over the existent application layer of each WSN stack, allowing the platform to be reutilized over different technologies. Fig. 1 shows the different layers of SYLPH and the different protocols that communicate each layer on two different ZigBee nodes (homogeneous WSN), as well as over a ZigBee node and a Bluetooth node by means of a SYLPH Gateway (heterogeneous WSN). The structure of SYLPH will now be briefly described.

– *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.
– *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 SYLPH Service Protocol. 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.
– *SYLPH Services Protocol (SSP)*. The SSP is the internetworking protocol of the SYLPH platform. SSP allows sending packets of data from one node to another node regardless of the WSN to which each one belongs.
– *SYLPH Services Definition Language (SSDL)*. The SSDL is the IDL (Interface Definition Language) used by SYLPH. Nodes can request the SSDS for the location of services and their specifications using SSDL.

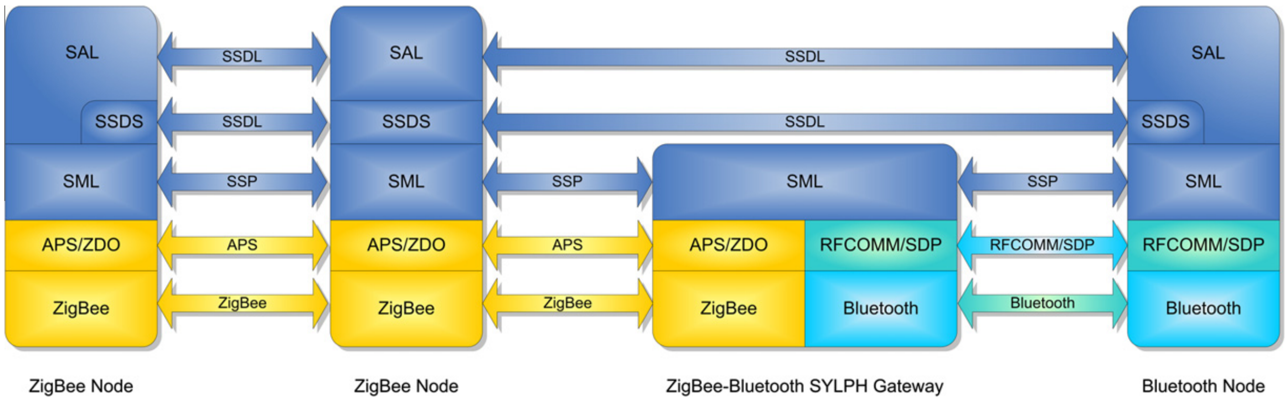

Fig. 1. Layers and protocols of the SYLPH platform.

– *SYLPH Services Directory Sub-layer (SSDS)*. The SSDS creates dynamic services tables to locate and register services in the network. A node that stores and maintains services tables is called SYLPH Directory Node (SDN). A node in the network can make a request to the SDN to know the location (i.e., network address) of a certain service.

3.2. SYLPH basic operation and SYLPH Directory Nodes (SDNs)

The behavior of SYLPH is essentially similar to that of any other service oriented architecture. First, a service registers itself on the SDN and informs the network of its location, the parameters it requires, and the type of returned value after its execution. In order to do this, the service uses SSDL, described in Section 3.3. Once the service has been registered in the SDN, it can be invoked by any application using SYLPH. Any node in the network (or other subsystem connected to the WSN) cannot only offer or invoke SYLPH services, but also include SDN functionalities to provide services descriptions to other network nodes.

The UML sequence diagram depicted in Fig. 2 shows an example of the basic operation of SYLPH platform when registering and discovering services using the SYLPH Directory Nodes. For example, SYLPH node #1, belonging to WSN “A” registers itself in the SYLPH platform. For this, it sends a broadcast message searching for existing SDNs in the network. At this moment, only SDN #0 is active, so after receiving the broadcast message it sends a message to node #1 informing of its SSP address and its setup parameters. After this, node #1 is able to communicate with SDN #0 to obtain information about the possible services existing in the network. Later, node #2 registers itself on the platform. It belongs to a different WSN, called WSN “B” and perhaps uses a radio technology different than that used by WSN “A”. As node #2 has SDN functionalities, it in-

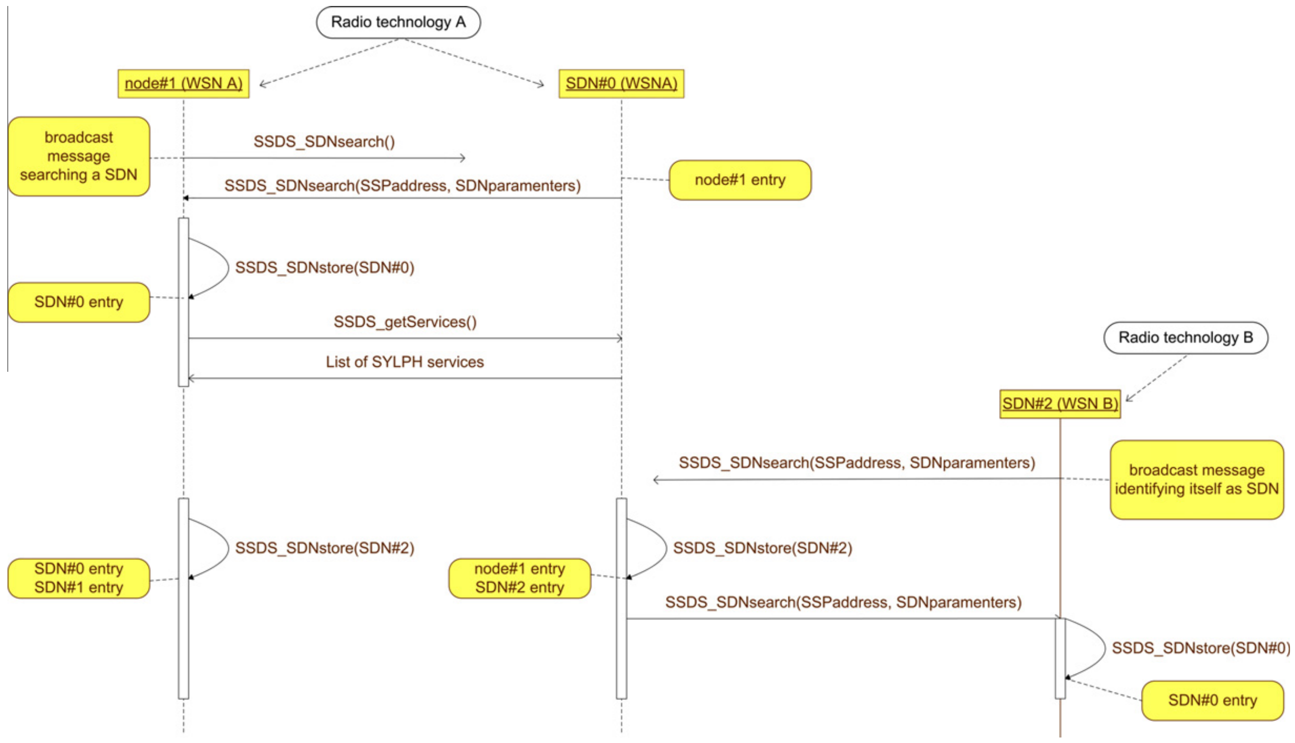

Fig. 2. Basic operation of SYLPH platform and SYLPH Directory Nodes.

forms the rest of the nodes with a broadcast message. SDN #1 stores this information on its SSDS entry list and informs node #2 about its role as SDN.

3.3. The SYLPH Services Definition Language (SSDL)

SSDL has been specifically designed to work with limited computational resource nodes [10,14]. Unlike other IDLs such as WSDL (*Web Services Definition Language*) [16], SSDL does not use as many intermediate separating tags, and the order of its elements is fixed. The reason for these constraints is to reduce processing in the devices microcontrollers. Consequently, using a simple IDL makes it possible to use nodes with fewer resources, less power consumption and at a lower cost.

The next example defines a simple service called `getLuminosity` to show the use of SSDL to define a SYLPH service. The following text represents the SSDL syntax used by developers to define this service in the node's firmware, not the version actually transmitted:

```
service getLuminosity {
  input {};
  output {
    status_t status;
    string units;
    uint16_t luminosity;};};
```

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 the microcontroller where the 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 that describe the service and how to invoke it from other nodes. Fig. 3 shows the SSDL frames involved in the `getLuminosity` 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 shown in Fig. 3a. This frame describes the service identification, the address of the node that stores the service and the definition of the input and output parameters. There are *marks* to denote the input and the output parameters. Once the invoker node knows the service definition, it can make a request to the service by sending a SSP frame to the node that stores the service (Fig. 3b). Finally, response frame (Fig. 3c) does not need *marks* to separate the parameters because the output parameters must follow a specific order. However, a *string end mark* must be used to know where the string-type data ends.

3.4. Operation of SYLPH over heterogeneous WSNs using SYLPH gateways

As previously mentioned, with SYLPH, 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. A SYLPH Gateway does not need to implement the layers over the SML. This fact can be seen in Fig. 1. The SYLPH Gateway stores routing tables for forwarding SSP packets among the different WSNs with which it is interconnected. 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.

4. The HERA platform

HERA (*Hardware-Embedded Reactive Agents*) facilitates communication amongst agents, applications and services through the use of dynamic and self-adaptable heterogeneous WSNs. Unlike other approaches [31–33], the agents

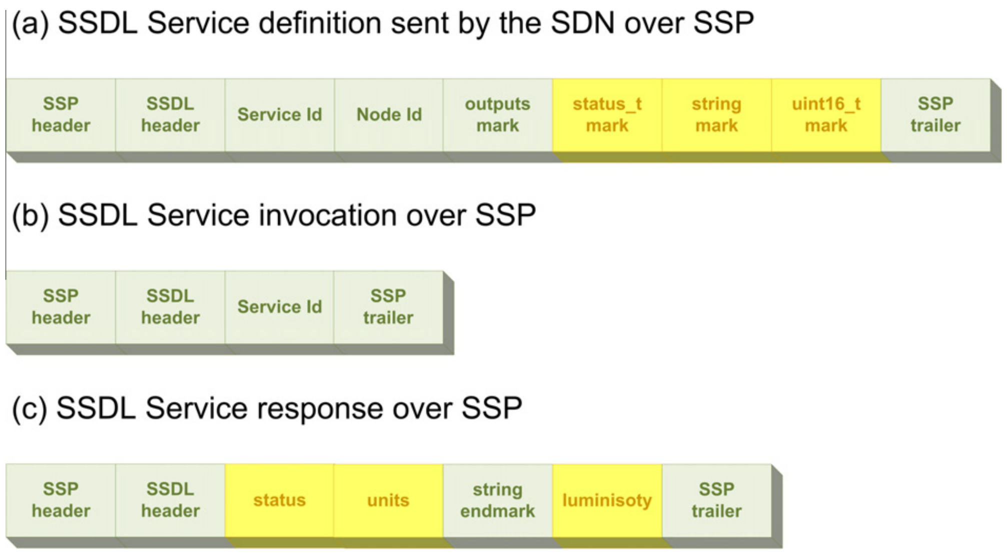

Fig. 3. Examples of SYLPH's SSDL frames over SSP.

in HERA are directly embedded on the WSN nodes and, as a result of SYLPH, HERA provides the possibility of connecting wireless sensor networks based on different radio and link technologies, whereas other approaches do not. That is, HERA allows the agents embedded into nodes to work in a distributed way and does not depend on the lower stack layers related to the WSN formation (i.e., network layer) or the radio transmission among the nodes that form part of the network (i.e., data link and physical layers). Likewise, HERA can be executed over multiple wireless devices independently of their microcontroller or the programming language they use.

Therefore, the main contributions of HERA over SYLPH are that HERA incorporates a new layer of reactive agents over the existing layers provided by SYLPH, as well as a Case-Based Planning mechanism that provides reasoning features to HERA Agents. These two main additions allow HERA to be a more powerful platform than SYLPH. One the one hand, HERA takes advantage of the main SYLPH features, that is, the distribution of functionalities over nodes with reduced computational and memory resources and using different wireless technologies. On the other hand, the HERA Agents and the HERA Case-Based Planning mechanism allow HERA to execute reactive agents that can make use of reasoning features, while SYLPH does not.

This way, developers can deploy intelligent context-aware applications by implementing HERA Agents embedded in each node. For example, a developer can design a home automation application in which is needed to collect context information from the environment using sensor nodes coming from different wireless technologies (e.g., ZigBee and Wi-Fi). In order to do this, it is necessary to have the SYLPH and HERA layers implemented for the target microcontroller and transceiver of each wireless sensor node. This way, it is easy to implement SYLPH Gateways that interconnect two or more wireless technologies through SYLPH layers for forwarding SML messages among the distinct WSNs. However, these previous developments must be done only once for each target microcontroller and transceiver, and then developers have only to implement the code of each HERA Agent they need in each node for accessing a certain set of sensors (e.g., luminosity, presence, temperature, smoke) and actuators (e.g., alarms, blinds, locks). Finally, developers can deploy a HERA Planning Agent in a central node so that HERA Agents can make use of the planning mechanisms, thus building powerful context-aware applications that learn from past experiences and adapt dynamically to new situations.

4.1. Adding the components of the HERA platform over SYLPH

The HERA agent platform adds its own agent layer over the SYLPH stack of layers, as shown in Fig. 4. This figure shows the schema of SYLPH and HERA over a ZigBee and a Bluetooth network through a SYLPH Gateway. As can be seen, HERA Agents on nodes from different radio technologies communicate each other in a transparent way thanks to SYLPH. As the HERA platform is based on the existing layers of SYLPH platform, HERA agents running on WSNs with different radio technology can communicate among themselves through one or more SYLPH Gateways, as explained in Section 4.4. Therefore, the main components added by HERA to the SYLPH's stack of layers are:

- *HERA Agents Layer (or just HERA)*. HERA agents are specifically intended to run on devices with reduced resources, precisely what SYLPH was designed for. To communicate with each other, HERA agents use HERACLES, the agent communication language designed for being used under the HERA platform. Each HERA agent is an intelligent piece of code running over the SYLPH Application Layer. As explained in Section 4.2, there must be at least one *facilitator agent* in every agent platform. This agent is the first created in the platform and acts as a directory for searching agents. In HERA, the equivalent of these agents is the HERA-SDN (*HERA Spanned Directory Node*).
- *HERA Communication Language Emphasized to Simplicity (HERACLES)*. The HERACLES language is directly based on the SSDL language. As with SSDL, HERACLES does not use intermediate tags and the order of its elements is fixed to constrain the resource necessities of the nodes. This makes its human-readable representation, used by developers for coding, very similar to SSDL. When HERACLES is translated to HERACLES frames, the actual data transmitted among nodes, they are encapsulated into simple SSDL frames using "HERA" as their *service id* field.

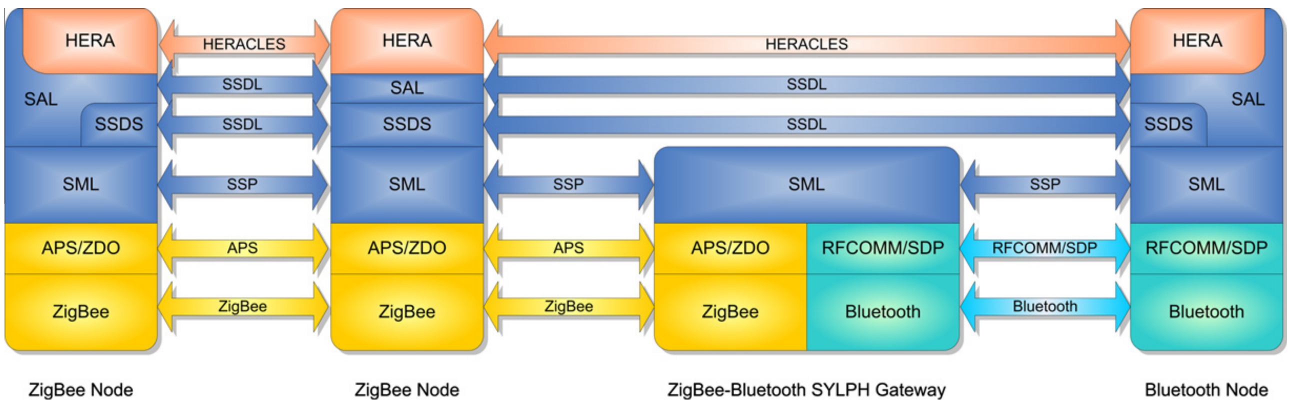

Fig. 4. Layers and protocols of the HERA platform.

4.2. HERA basic operation and HERA Spanned Directory Nodes (HERA-SDNs)

Every agent platform needs some kind of *facilitator agent* that needs to be created before other agents are instantiated in the platform [19–21]. Facilitator agents act as agent directories. This way, every time an agent is created, it is registered on one of the existing facilitator agents. This allows other agents to request one of the facilitator agents in order to know where an agent with certain functionalities is and how to invoke such functionalities. As HERA is intended to run on machines that are not more complex than sensor nodes themselves are, it was necessary to design some hardware facilitator agents that do not need more CPU complexity and memory size than what a regular sensor node has. In order to do this, HERA's facilitator agents, called *HERA Spanned Directory Nodes* (HERA-SDNs) are based on the SYLPH Directory Nodes (SDNs), described above. This way, any HERA node can perform as a HERA-SDN, just as SDNs do in the SYLPH platform. However, a HERA-SDN does not also have to be a SDN. HERA-SDN instances itself and starts the HERA platform by registering a special SYLPH service called "HERA" on a SDN stored on any node of the SYLPH network. When a new HERA Agent wants to instantiate itself through a HERA-SDN, it looks for the "HERA" service on the SYLPH network, using a primitive service of the SSDL/SSP layers. When a HERA Agent is correctly instantiated, the HERA layer also registers a "HERA" service for the agent in a SDN. In this way HERA Agents can send HERACLES messages to each other over SYLPH, referring to the service of each node with HERA Agents, including HERA-SDNs, as "HERA".

The UML sequence diagram in Fig. 5 shows an example of the basic operation of SYLPH and HERA platforms when registering services or agents on the HERA Spanned Directory Nodes. In order to start the HERA platform, an initial HERA-SDN must be created. This will be HERA-SDN #0 running on SDN #0. At that moment, other SYLPH nodes with HERA running on them can instantiate more HERA agents or even more HERA-SDNs. Because HERA is designed to run on devices with low resources that are usually connected wirelessly, it is very important that the platform does not have to depend on only one HERA-SDN (i.e., one facilitator agent). This way, if the HERA-SDN crashes (e.g., power failure or problems with radio trans-

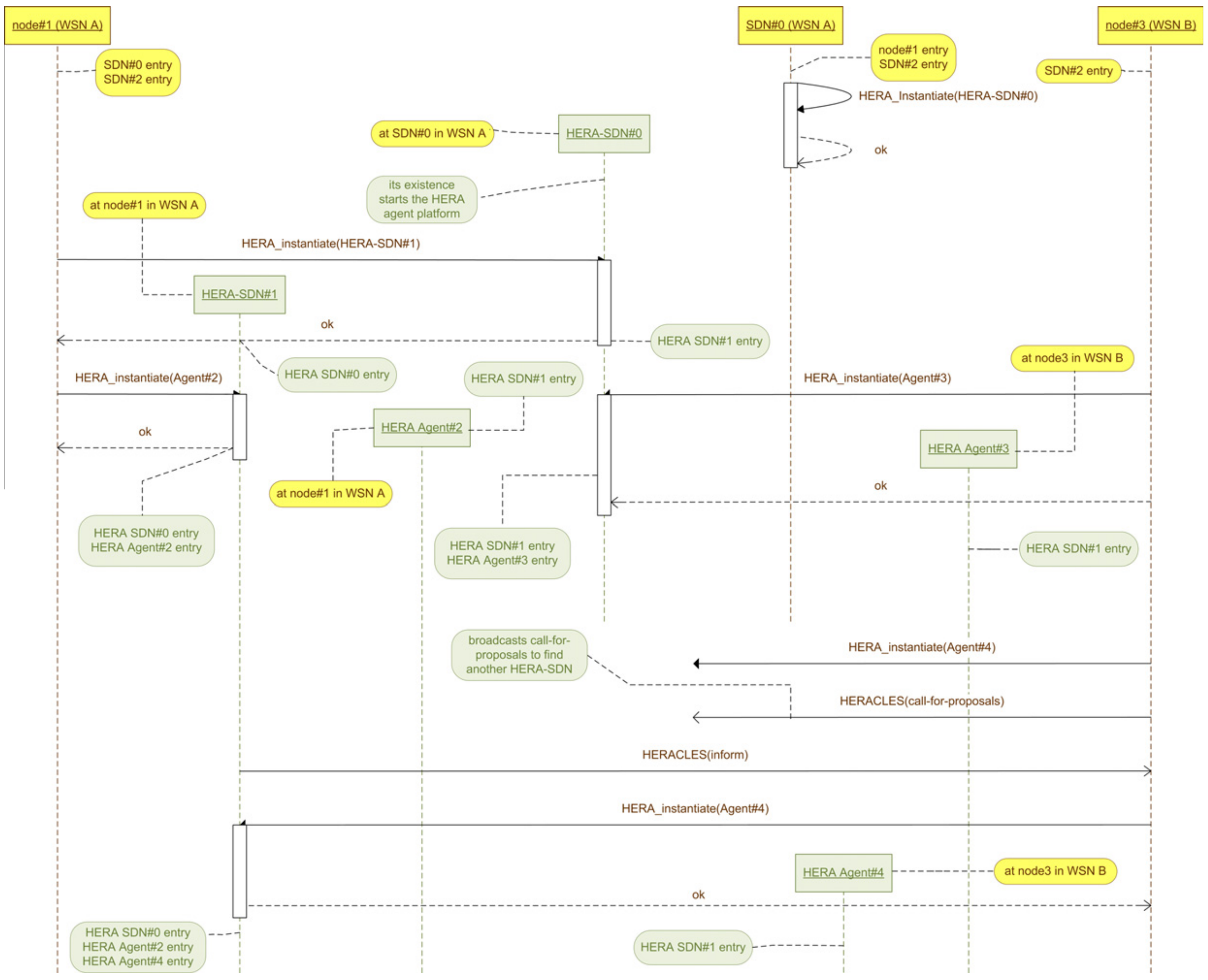

Fig. 5. Basic operation of HERA platform and HERA Spanned Directory Nodes.

mission), the HERA platform will not fail and will not need to be started again. After the creation of the HERA-SDN #0, the SYLPH node #1 uses the HERA-SDN #0 to instantiate a new HERA-SDN, the HERA-SDN #1, thus increasing the redundancy of the HERA-SDNs and the robustness of the platform. The SYLPH node #1 also instantiates the HERA Agent #2, this time through the HERA-SDN #1. SYLPH node #3, in the other WSN, instantiates HERA Agent #3 through the HERA-SDN #0, even if they are in distinct WSNs. With SYLPH, this is no longer a problem. At a specific moment, SDN #0 is powered off. After that, SYLPH node #3 looks for HERA-SDN #0. As HERA-SDN #0 does not reply, SYLPH node #3 sends a broadcast *call-for-proposal* HERACLES frame in order to find a live HERA-SDN. As HERA-SDN #1 replies, HERA Agent #4 is created through HERA-SDN #1. As shown in Fig. 5, there can be several HERA Agents in a single SYLPH node. Moreover, there can be SYLPH nodes with no HERA implementation. A SYLPH Gateway is a clear example of this, as explained below. If, at a certain moment, HERA Agent #2 wants to look for an agent, but HERA-SDN #0 is not alive again, then HERA Agent #2 also has to look for an existing HERA-SDN in the platform, thus storing entries for the two HERA-SDNs. When HERA-SDN #0 is alive again, it will be useful for HERA Agent #2 to have this redundancy on HERA-SDNs entries.

4.3. The HERA Communication Language Emphasized to Simplicity (HERACLES)

In HERA, the hardware agents communicate with each other through the *HERA Communication Language Emphasized to Simplicity* (HERACLES). This language is an extension of the SSDL used in SYLPH. As explained above, SSDL has two distinct representations [9]: one that is human-readable, similar to C language and used for services development proposals, and one embedded on frames that SYLPH nodes understand. This is done in this way because in nodes with reduced resources (memory and CPU time) it is not convenient to overload the microcontroller and the memory space with a heavy parsing method. When developing a program, programmers use the human-readable representation to define agents' functionalities, similar to that shown as follows.

```
request {
  sender agent1;
  receiver agent2;
  content {
    action(agent2) {
      inform-if {
        sender agent1;
        receiver agent2;
      content {
        message {
        state result;};};
      language HERACLES;
        ontology HERA_ONTOLOGY;};};};
      LANGUAGE HERACLES;
      ontology HERA_ONTOLOGY;};
    inform {
    sender agent2;
  receiver agent1;
    content {
      message {
        state result;};};
    language HERACLES;
    ontology HERA_ONTOLOGY;};
```

However, similar to SYLPH, HERA agents transmit the more compact representation of HERACLES as frames. The compact frames corresponding to the previous example are also represented in Fig. 6. These kinds of compact frames are what HERA agents transmit in a heterogeneous WSN based on HERA-SYLPH over the SSDL/SSP protocols. Fig. 6a shows the frame corresponding to the HERACLES *request*, whereas Fig. 6b depicts the frame corresponding to the HERACLES *inform* of the previous example.

4.4. Operation of HERA over heterogeneous WSNs using SYLPH gateways

Because of HERA is implemented over SYLPH through the addition of new layers and protocols (HERA Agents and HERACLES), it can be used over several heterogeneous WSNs in a transparent way. HERA Agents are implemented

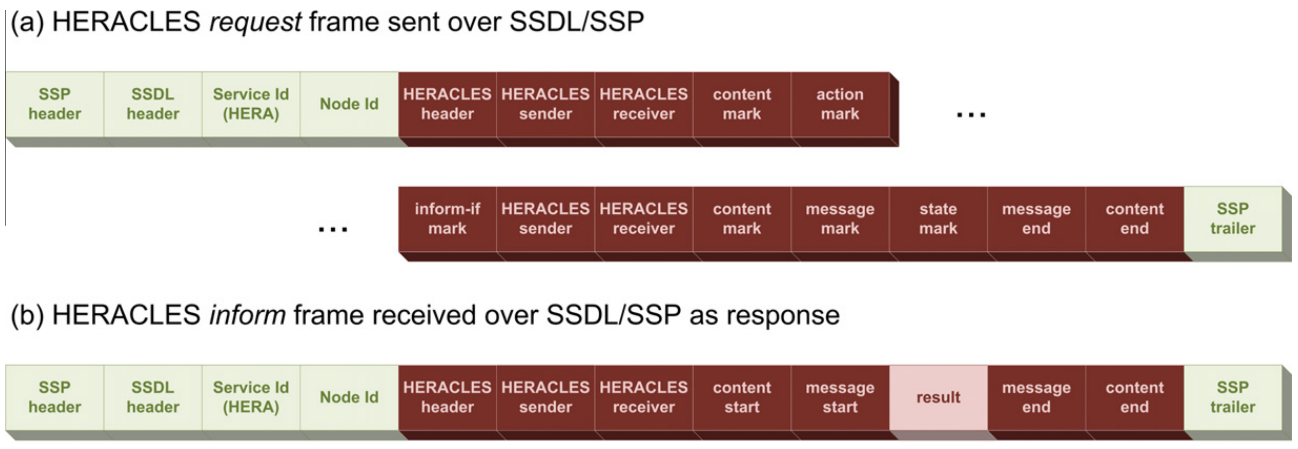

Fig. 6. Examples of HERA's HERACLES frames over SSP/SSDL.

over the SAL layer, so HERA does not mind how many intermediate SYLPH Gateways and different WSNs there are between the location of one HERA Agent and another. This is demonstrated in Fig. 4. Both HERA Agents and HERA-SDNs communicate with each other directly through HERACLES. HERA Agents use SAL's service points to deliver HERACLES frames between agents. Since HERACLES frames are transported as other SSDL frames over SSP between SYLPH nodes, HERA Agents do not need to know which nodes other HERA Agents are stored on, or if such nodes are in remote WSNs.

4.5. HERA Case-Based Planning mechanism

As previously mentioned, some agents in HERA integrate a Case-Based Planning (CBP) mechanism. The CBP mechanism provides the agents with greater adaptation capabilities. As it is a complex and resources demanding task, the CBP mechanism has been modeled as a service provided by a special HERA Agent, known as *HERA Planning Agent*, which runs on a central node (i.e., a computer or a wireless device with moderate computational resources). The main characteristics of this mechanism are described in the remainder of this section.

CBP comes from CBR, but is specially designed to generate plans (sequence of actions) [29]. The problems and their corresponding plans are stored in a plans memory. The reasoning mechanism generates plans using past experiences and planning strategies, which is how the concept of Case-Based Planning is obtained [29]. CBP consists of four sequential stages similar to CBR stages: retrieve, reuse, revise and retain. Problem description (initial state) and solution (situation when final state is achieved) are represented as beliefs, the final state as a goal (or set of goals), and the sequences of actions as plans. The CBP cycle is implemented through goals and plans. When the goal corresponding to one of the stages is triggered, different plans (algorithms) can be executed concurrently to achieve the goal or objective. Each plan can trigger new sub-goals and, consequently, cause the execution of new plans.

The HERA CBP mechanism needs a set of HERA Agents running on a set of nodes (i.e., wireless devices). Each of the nodes is connected physically to different sensors and actuators. This way, each node in the system can transmit commands to the actuators according to the sensors measurements. Each device d_i is defined by the sensors and actuators to which it has access, as expressed in the following equation:

$$d_i = (S_i, A_i) \tag{1}$$

where S_i is the set of sensors and A_i is the set of actuators.According to the values read from the sensors, each HERA Agent running in the devices makes use of the actuators in order to achieve the required goal (e.g., stop a heater when a target temperature has been achieved). Thus, the behavior of the HERA CBP mechanism is established by a database generated from the information of the sensors and actuators. This way, when some event produces an interaction with the sensors in the devices (e.g., a user action or a variation in the environment), these devices forward the values from the sensors and actuators to a central node that runs a special HERA Agent that stores this information. This agent is known as *HERA Planning Agent*. Depending on the size of the whole system (i.e., the number of nodes in the network, as well as the number of sensors and actuators associated to them), this central node can be implemented as a computer with a database stored in a physical disk or as a wireless sensor node with a smaller database stored in an EEPROM or Flash memory. Each device d_j has its own cases memory. Each case of the device d_j follows the structure indicated in the following equation.

$$c_i^{d_j} = \left(V_i^{S_j}, V_i^{A_j}\right) \tag{2}$$

where $V_i^{S_j}$ is the set of values from the sensors associated with the device j, and $V_i^{A_j}$ the values associated to the actuators.

The reactive behavior of the HERA Agents is defined as a set of rules that determine the relation among the sensors and the actuators. The rules are generated by the HERA CBP mechanism. There are two kinds of rules: static rules and dynamic rules. Static rules are pre-defined rules that have priority over dynamic rules. Static rules determine the default behavior of each node and act also as a backup of these behaviors (i.e., they are stored directly in the nodes). Dynamic rules are automatically generated from the defined cases for each of the devices. The dynamic rules are periodically updated on every run of the HERA Agents (i.e., each time they are requested to execute a task). Fig. 7 shows the functioning of the HERA CBP mechanism when a device must execute a new task.

Both static and dynamic rules are defined using a defined grammar that facilitates the error detection and also the generation of native code that is actually running in the HERA Agents. One the one hand, static rules are stored in the database by the HERA Planning Agent following this grammar. This way, rules are formed by sensors, literals, comparison and logical operators, as well as the final action to be performed by the actuator. On the other hand, dynamic rules are defined by the CBP mechanism by means of the information of the cases shown in Eq. (2). During the recovery stage the CBP mechanism recovers the information with the sensors and actuators associated to a certain device. During the reutilization stage, automatic rules are generated from this information. For the generation of automatic rules, different algorithms based on decision rules and decision trees can be used. An example of algorithm based on decision rules is M5 [35], while J48 [36] is based on decision trees.

In HERA, the J48 algorithm is used for the generation of rules [36]. The inputs of the classifier are the value of the sensors, and the output belonging to the actuator. Using this configuration the J48 is trained, thus obtaining a decision tree that represents the behaviors of the actuators. In this sense, the different actuators are chosen as leaf nodes in the decision trees, while the sensor values are placed in the intermediate nodes. There is a decision tree according to the sensors situated in the device for each actuator. The devices can select a value of the actuators according to the value of the sensors through the decision tree, and the system only has to follow the conditions in the intermediate nodes until it arrives at a leaf node.

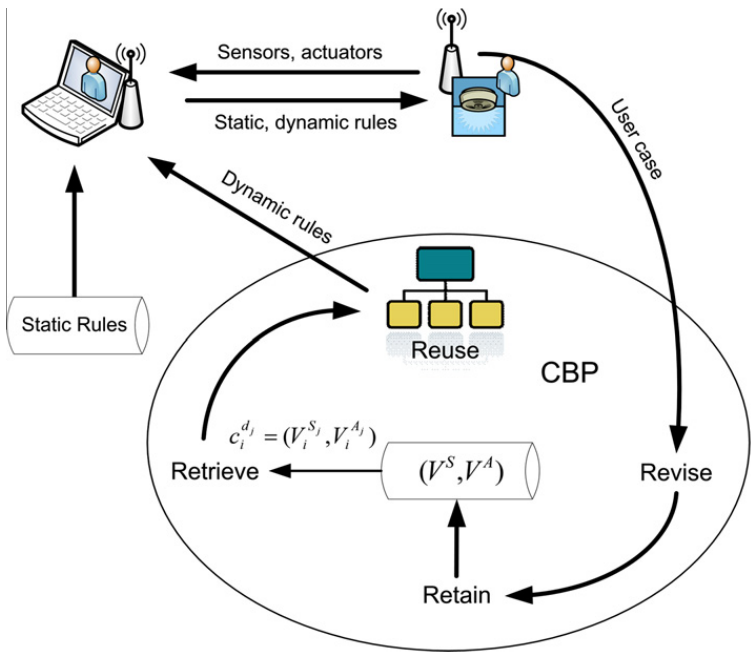

Fig. 7. Functioning of the HERA CBP mechanism.

The final schema of the decision trees would be similar to that shown in Fig. 8, and rules would be generated as indicated in the schema. The generated rules follow the grammar indicated above. The decision tree generated contains all the necessary information to generate the rules according to the grammar. The system only has to select each leaf node and move up toward the root node, introducing a new condition for each movement throughout the tree.

During the revision and learning stages, the system only stores the values of the actuators and sensors when the actuators are established manually by the users. The automatic decision of the agents are not stored in the system. The system only stores the interactions of the users as it tries to adapt to their behaviors. The system does not store generated cases as it is able to predict the behaviors, and only stores a new case when a user modifies the automatic configuration.

For the development of the rules generator system the Weka libraries are used for the implementation of J48 algorithm. JFlex and Cup are used as lexical and syntactic analyzers. By means of Cup it is generated the native code that is then transferred and executed on chips.

5. Experiments and results

This section describes distinct experiments performed to test the HERA platform. On the one hand, Section 5.1 describes a test battery performed to evaluate the instantiation of HERA Agents and transmission of HERACLES

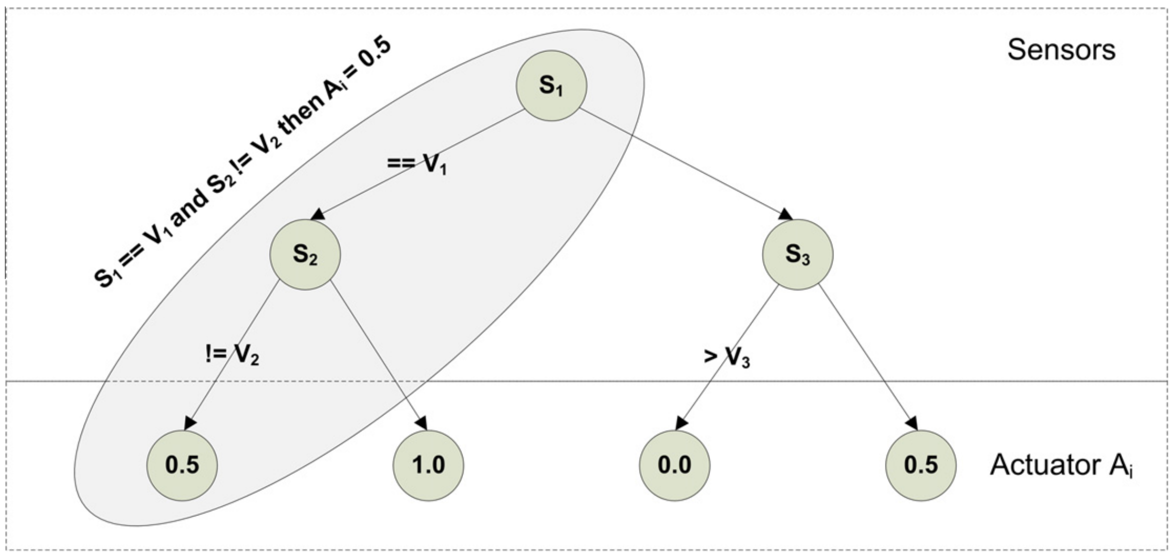

Fig. 8. Example of decision tree for the dynamic rules.

frames about them, both in homogeneous and in heterogeneous HERA WSNs. On the other hand, Section 5.2 describes an implementation of the HERA platform in order to develop some context-aware applications in a real scenario.

5.1. HERA performance tests

Several experiments were carried out to evaluate the performance of the HERA platform, mainly to test how it handled the instances of HERA-SDNs and HERA Agents and the exchange of HERACLES frames, both in homogeneous and heterogeneous HERA WSNs. In this sense, we deployed two distributed WSN infrastructures with HERA running over it, the first one formed by only ZigBee nodes and the second one by both ZigBee and Bluetooth nodes.

We have also developed an application for monitoring the state and operation of the HERA network. This application is based on a previous one we had developed for monitoring SYLPH networks [14]. As with the previous one, this application 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 a computer (e.g., via a USB port). The node gathers all the invocations and forwards them to the monitoring application running on the computer. The same process is done for service responses, searches and registrations. The monitoring application makes it possible to observe when a node is searching for a certain service in the network, the services offered by the nodes, and the contents of the SSDS entry tables stored in the SDNs. In addition, with the newly developed application it is possible to monitor the HERA agent instances in the HERA-SDNs of the HERA platform and also the HERACLES *request*, *inform* and other frames.

The first sensor infrastructure consisted of a ZigBee network with 50 devices, one acting as coordinator and the rest as routers. The sensor infrastructure was formed by n-Core Sirius A devices belonging to the novel n-Core platform (http://www.n-core.info). Each n-Core Sirius A 2.4 GHz device includes an ATmega1281 microcontroller with 8 KB RAM, 128 KB Flash memory, an AT86RF231 transceiver and several communication ports (GPIO, ADC, I^{2}C and USB/RS-232 UART) to connect to a wide range of sensors and actuators [37]. ZigBee is based on the IEEE 802.15.4 standard and operates in the 868/915 MHz and 2.4 GHz unlicensed bands [38,39]. Unlike Wi-Fi or Bluetooth, ZigBee is designed to work with low-power nodes and allows up to 65,534 nodes to be connected in a star, tree or mesh topology network. The ZigBee nodes were distributed in a short-range simple mesh, with less than 10 m between any router and the coordinator. Each time the ZigBee network was formed, the nodes were powered on different random times, so that the mesh topology was different each time. However, they were some constraints: the maximum depth of the network (i.e., the maximum number of hops between the coordinator and any node in the network) was 5, the maximum number of neighbors of any node was 8 and the maximum number of children of any node in the network was also 8.

The experiments consisted of trying to start a platform with HERA over the ZigBee network. In the network, the ZigBee coordinator and a ZigBee router acted as SDNs and the other 48 ZigBee routers acted as SYLPH nodes. After the entire network was correctly created, the SDN in the coordinator tried to instance a HERA-SDN. The HERA-SDN instanced itself and started the HERA platform registering a special service called "HERA" on the SDN stored on the same ZigBee coordinator node. Then, 10 nodes tried to instance one HERA Agent in the HERA platform. Once the HERA-SDN and the 10 HERA Agents were successfully instantiated, the HERA-SDN started to "ping" every of the ten HERA Agents with a *request* HERACLES frame including an *inform-if* command and waiting for an *inform* frame as a "pong" response. Each HERA Agent was pinged by the HERA-SDN one time every 5 s during 1 h (7200 total pings tried). The experiment was run until both the platform and the agents were successfully started/instantiated 50 times. When the network could not be correctly created the run was discarded and not taken into account in the 50 runs. Furthermore, if the HERA platform could not be completely started and created (i.e., all 10 HERA Agents correctly instantiated), these runs were also discarded and not taken into account as forming part of the 50 runs. If any HERA agent crashed it was immediately restarted. HERACLES messages were registered to measure when a *ping-pong* failed and if a HERA agent had to be restarted. The results are shown in Table 1. As can be seen, it is necessary to try to create 56 times the SYLPH and HERA platforms to get 50 runs of the experiment with the 10 HERA Agents successfully instantiated in the HERA platform. This is because the SYLPH platform was not successfully created 3 times, as some of the registrations of the SYLPH nodes failed (the 0.17% of the total 2800). Likewise, the HERA platform was not successfully created (with the 10 HERA Agents correctly instantiated) 3 times, as some of the instantiations of the HERA Agents failed (the 0.56% of the total 530). This indicates that it is necessary to improve both the SYLPH network creation and the instantiation of HERA Agents. These fails are caused by the transmission fails in the ZigBee network. That is, 50 nodes in a ZigBee network trying to transmit their registrations and instantiations in the same time window provoke frame collisions and retransmissions. When the total number of retransmissions in the different layers of ZigBee and the SYLPH/HERA platforms are exhausted, nodes must finally give up. This also motivates the number of *ping-pongs* that failed (the 0.19% of the total 7200). A better Automatic Repeat Request (ARQ) mechanism could increase SSP-over-WSN transmissions. In addition, the robustness of the HERA Agents should be improved by introducing a mechanism to ping and keep running the HERA Agents and the HERA-SDNs.

The same experiments were repeated using the second infrastructure, which consisted of a heterogeneous sensor network made up of one 25-node ZigBee network and another 25-node Bluetooth *scatternet* [40], both of them interconnected through a SYLPH Gateway, in this case a computer. Bluetooth operates also in the ISM 2.4 GHz band. It allows creating star topology networks called *piconets* of up to eight devices in which one of them acts as master and

Table 1
Results of the HERA experiments comparing a homogeneous HERA WSN and a heterogeneous HERA WSN.

Only-ZigBee HERA experiments	
Total runs	56
SYLPH nodes not registered correctly (% of all tries in total runs)	5 (0.17%)
SYLPH platform not created correctly (% of total runs)	3 (5.36%)
HERA Agents not instantiated correctly (% of all tries in total runs)	3 (0.56%)
HERA platform not started correctly (% of SYLPH correctly created)	3 (5.66%)
All 10 HERA Agents correctly instantiated	50
Total pings tried	7200
Ping-pongs not completed (% of total tried)	14 (0.19%)
Total restarted HERA Agents in an hour	9
ZigBee + Bluetooth HERA experiments	
Total runs	58
SYLPH nodes not registered correctly (% of all tries in total runs)	7 (0.24%)
SYLPH platform not created correctly (% of total runs)	5 (8.62%)
HERA Agents not instantiated correctly (% of all tries in total runs)	6 (1.13%)
HERA platform not started correctly (% of SYLPH correctly created)	3 (5.66%)
All 10 HERA Agents correctly instantiated	50
Total pings tried	7200
Ping-pongs not completed (% of total tried)	27 (0.37%)
Total restarted HERA Agents in an hour	12

the rest as slaves. Several Bluetooth piconets can be interconnected by means of Bluetooth devices that belong simultaneously to two or more piconets, thus creating more extensive networks (known as *scatternets*) [40]. The Bluetooth nodes used in these experiments had a CSR BlueCore4 chip that included a RISC microcontroller with 48 KB of RAM. The 25-node ZigBee network had similar network characteristics as in the first experiment: 10 m as maximum between adjacent nodes (being all of them ZigBee routers except for the coordinator), five hops maximum depth network, eight neighbors maximum for any node and eight children maximum for any node. This way, the ZigBee network topology was different each time it was formed as in the first experiment. In the other hand, the Bluetooth network had a static topology formed by five Bluetooth *piconets*. Specifically, one of these piconets acted as the main piconet. The master of the main piconet (i.e., the main Bluetooth master) was also the node that interconnected the Bluetooth scatternet with the ZigBee network through the computer acting as SYLPH Gateway. Moreover, the four slave nodes in the main piconet were also slave nodes each of them in one of the other four Bluetooth piconets. These other piconets had each of them six nodes: a master node, a slave node being also slave in the main piconet and four more slave nodes. There were also two SDNs in this experiment, one in each WSN (i.e., a SDN in the ZigBee network and another one in the Bluetooth network). Similarly to the experiments made on the first infrastructure, the SDN in the ZigBee coordinator node tried to instance a HERA-SDN. The HERA-SDN instanced itself and started the HERA platform registering a special service called "HERA" on the SDN stored on the same ZigBee coordinator node. Then, five nodes in the ZigBee network and five nodes in the Bluetooth network tried to instance one HERA Agent in the HERA platform. Once the HERA-SDN and the 10 HERA Agents were successfully instantiated, the similar *ping-pong* process performed in the experiments with the first infrastructure was carried out. The results of these experiments are also shown in Table 1. As can be seen, it is necessary to try to create 58 times the SYLPH and HERA platforms to get 50 runs of the experiment with the 10 HERA Agents successfully instantiated in the HERA platform, a little more than in the homogeneous network. The SYLPH platform was not successfully created 5 times, as the 0.24% of the 2900 registrations of the SYLPH nodes failed. The HERA platform was not successfully created 3 times, as the 1.13% of the 530 instantiations of the HERA Agents failed. The number of *ping-pongs* that failed was the 0.37% of the total 7200. These results demonstrate that the inclusion of the SYLPH Gateway makes the formation of the whole SYLPH network a little harder. Moreover, once the network is successfully formed, there is no difference of in the service registration mechanism. However, the transmission of HERACLES frames and the robustness of HERA Agents seem to be a little more unstable in the heterogeneous HERA network. On the one hand, in a 25-node ZigBee network the number of frame retransmissions decreases with respect to a 50-node network as in the experiments made in the first infrastructure. On the other hand, the implementation of the HERA platform over the Bluetooth nodes needs to be debugged, as it is a newer development compared with the more stable HERA over ZigBee.

5.2. Implementation of HERA in a real scenario

In order to demonstrate the feasibility of the HERA platform for developing context-aware applications in a real scenario, two ZigBee networks based on the described n-Core Sirius devices [37] have been deployed in a laboratory belonging to the Bioinformatics, Intelligent Systems and Educational Technology (BISITE) Research Group (http://bisite.usal.es) of the University of Salamanca (Spain). The first ZigBee network is intended for sensing and automation purposes, while the second network is aimed at indoor locating. Both sensing and locating are key aspects when building a context-aware system in order to gather context information about the users and the environment. Both networks are formed by n-Core Sirius nodes. In addition to the n-Core Sirius A nodes, described in Section 5.1, two more kinds of nodes are used: n-Core Sirius B and n-Core Sirius D devices. These devices are smaller than n-Core Sirius A devices, even though they include almost the same internal communication ports (i.e., GPIO, I^2C, USB) to be connected to sensors or actuators. On the one hand, n-Core Sirius B devices are intended to be used with an internal battery and include two general-purpose buttons. On the other hand, n-Core Sirius D devices are aimed at being used as fixed ZigBee routers using the main power supply through a USB adaptor. These devices can be seen in Fig. 9, which also shows a plan of the laboratory where the ZigBee networks are deployed. Fig. 9a shows a Sirius

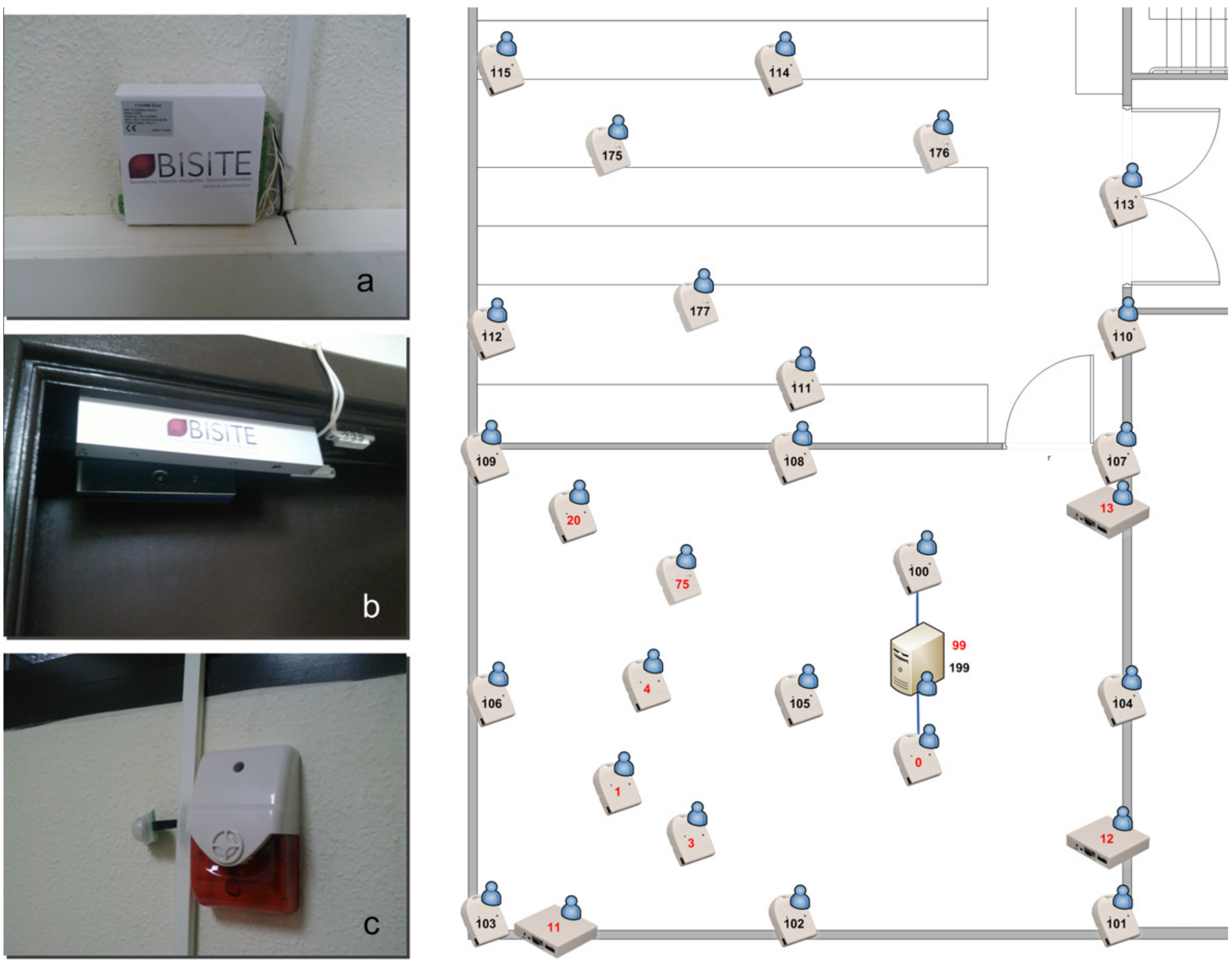

Fig. 9. Deployment of the ZigBee nodes in the implementation of HERA in a real scenario.

A deployed in the laboratory, while Fig. 9b shows an electromagnetic latch and Fig. 9c shows an opto-acoustic siren.

In a first stage of the development, the distinct layers of the SYLPH platform were implemented as C static link libraries. These SYLPH libraries can be further statically linked by code that is implemented as firmware that can be loaded into the Flash memory of the microcontrollers in the ZigBee devices. This firmware is usually loaded through a JTAG (Joint Test Action Group, common name for IEEE 1149.1 standard) programmer interface or a USB port using a bootloader application. The implementation of the SML layer and the SSP protocol are dependent of API provided by manufacturers to access microcontroller, communication ports and ZigBee features. However, the implementation of the SSDS and SAL layers, as well as the SSDL protocol, is independent of the ZigBee infrastructure and only depends on the microcontroller features. In a similar way, the implementation of the HERA layer, as well as the proper HERA Agents themselves and the HERACLES protocol, are also independent of the radio transmission protocol and are built over the SAL layer. Finally, both SYLPH and HERA layers are also implemented as dynamic link libraries over Windows operating system to be used by applications intended to access to SYLPH/HERA WSNs. These applications include monitoring applications, SDNs, HERA-SDNs, as well as HERA CBP Agents running on personal computers that are connected to one or more SYLPH/HERA WSNs through different communication ports (e.g., USB or RS-232).

Therefore, each of the nodes in the two ZigBee networks are loaded with firmware linked to SYLPH/HERA libraries. This way, all the functionalities of these devices are provided by HERA Agents. Each HERA Agent in the ZigBee nodes is intended to control one or more sensors or actuators. A monitoring application based on SYLPH/HERA runs on a PC to register and manage data gathered from distinct nodes in the networks. This PC is connected to the two ZigBee networks through two USB ports, each of them connected to the ZigBee coordinator of each network. Each of the ZigBee coordinators acts also as SDN and HERA-SDN. The HERA Agents running on the ZigBee nodes can communicate with HERA Agents running on the PC and with HERA Agents running on other ZigBee nodes to request data from sensors or send commands to actuators. Furthermore, a HERA Planning Agent runs on the PC for expanding the possibilities of the system.

Table 2 shows the description of the role for each node in the two ZigBee networks, as well as the sensors and actuators that are controlled locally or remotely through HERA Agents. The sensing and automation network allows gathering data from different sensors in the wireless nodes: temperature, humidity, magnetometer, accelerom-

Table 2
Description of the nodes of the context-aware real scenario with HERA running on it.

SYLPH node id	Description of role and sensors/actuators
Sensing and automation network	
0	ZigBee coordinator, SDN, HERA-SDN
1	– Accelerometer (I^2C), sends data to PC – Temperature (I^2C), sends data to PC – 2 buttons (I^2C), send data to PC – Joystick (I^2C), sends data to PC
3	– Magnetometer/compass (I^2C), sends data to PC
4	– Temperature (I^2C), sends data to PC – Humidity (I^2C), sends data to PC – Luminosity (I^2C), sends data to PC and controls dimmer in node #20
11	– Luminosity (ADC), sends data to PC – Presence (GPI), sends data to PC – Opto-acoustic siren (GPO/relay #2)
12	– Power socket (GPO/relay #1)
13	– Door sensor (GPI), sends data to PC and activates GPO/relay #1 in this node – Presence sensor (GPI), deactivates GPO/relay #1 in this node – Electromagnetic latch (GPO/relay #1) – Panic button (GPI), activates/deactivates GPO/relay #2 in node #11
20	– Lamp DAC dimmer (I^2C)
75	– Left button (IRQ #6), deactivates the GPO/relay #1 in node #13 – Right button (IRQ #7), deactivates the GPO/relay #2 in node #11
99	PC (node id in sensing and automation SYLPH network)
	Description of role
Indoor locating network	
100	ZigBee coordinator, SDN, HERA-SDN
101–115	Readers
175–177	Tags
199	PC (node id in indoor locating SYLPH network)

eter, joystick and buttons, luminosity, presence and door sensors. In addition, this network allows controlling different actuators (i.e., electromagnetic latch, lamp and siren) from the PC or from other ZigBee nodes. All sensors and actuators are controlled by HERA Agents running on the ZigBee nodes. For instance, a HERA Agent running on the PC can send a *request* HERACLES frame to a HERA Agent running on the node #1 (a n-Core Sirius D device) to ask for a temperature value, that will be delivered in a *inform* HERACLES frame to the first HERA Agent. Likewise, a HERA Agent running on node #4 can read itself the ambient luminosity for controlling a lamp in the node #20, which is managed by other HERA Agent. The first HERA Agent communicates with the later using *request* HERACLES frames for stating a certain value for the dimmer. Furthermore, HERA Agents running on the ZigBee nodes send periodically their sensor values to the HERA Planning Agent running on the PC using HERACLES frames. This way, the HERA CBP mechanism updates the dynamic rules from the initial static rules provided by users, as described in Section 4.5 and HERA Agents can consult the HERA Planning Agent to retrieve plans and consequently control the different actuators in the system.

Similarly, in the indoor locating network, n-Core Sirius B devices are used as tags, while n-Core Sirius D devices are used as readers. This way, n-Core Sirius B devices are carried by users and objects to be located, whereas n-Core Sirius D devices are placed at ceilings and walls to detect the tags. Each user or object to be located in the system carries an n-Core Sirius B acting as tag. Each of these tags runs a HERA Agent that broadcasts periodically an *inform* HERACLES frame including, amongst other information, its unique identifier in the SYLPH network. The rest of the time these devices are in a sleep mode, so that the power consumption is reduced. A set of n-Core Sirius D devices is used as readers throughout the environment, being placed on the ceiling and the walls. The broadcast HERACLES frames sent by each tag are received by the readers that are close to them. This way, HERA Agents running on readers store in their memory a table with an entry per each detected tag. Each entry contains the identifier of the tag, as well as the RSSI (Received Signal Strength Indication) and the LQI (Link Quality Indicator) gathered from the broadcast frame reception. Periodically, each HERA Agent running on each reader sends this table to the HERA Planning Agent running on the computer. Using these detection information tables, the HERA CBP mechanism estimates the position of each tag in the environment, which is shown by a HERA Agent running on the PC that acts as other actuator in the HERA platform, but acting over a Graphical User Interface.

6. Conclusions and future work

The HERA platform (*Hardware-Embedded Reactive Agents*) allows wireless devices from different technologies to work together in a distributed way. HERA Agents can communicate in a distributed way regardless of the technology or the programming language they use. Furthermore, HERA Agents are light enough to be run on WSN nodes with limited resources. The HERA Agents are reactive because they act on devices with critical response times. HERA is a model that successfully solves the problems it sets out to resolve. HERA is a platform specially designed to implement hardware agents. Because HERA is based on SYLPH, it allows devices from different radio and networks technologies to coexist in the same distributed network. However, HERA goes a step further than SYLPH and adds reactive agents and a Case-Based Planning mechanism to the platform, extending its context-aware features. In HERA, unlike other approaches already discussed, agents are directly embedded on the sensor nodes. HERA facilitates and speeds up the integration between agents and sensors for reusing resources in the context. This approach allows the development of multi-agent systems with increased scalability. It also expands the agents' capabilities to obtain information about the context and to automatically react over the environment. A totally distributed approach and the use of heterogeneous WSNs provides platform that is better capable of recovering from errors, and more flexible to adjust its behavior in execution time. Even though HERA is focused specially on sensor nodes with small resources, it can be implemented on almost any kind of device. HERA adds intelligence to sensors by means of light reactive agents, improving the experience of developers and users in context-aware technologies. The HERA CBP mechanism facilitates the inclusion of new sensors dynamically, without need of performing offline training for each of the devices. In addition, the CBP mechanism can determine automatically the influence of the sensors to establish the final state of each of the actuators, so it is not necessary to indicate the relation among sensors and actuators.

Future work includes the improvement of the overall performance of the HERA platform. This way, the underlying SYLPH platform will be also improved, especially in the network formation and the SYLPH Gateways. In this sense, it will be evaluated other characteristics related to HERA/SYLPH nodes such as agent execution time, sleep mode intervals and power consumption. In addition, it will be added a set of cross-layering services, so that HERA agents can modify the network parameters of each specific radio technology in a uniform way. Furthermore, we are working in the design of an efficient mechanism that allows HERA agents to move throughout different nodes, no matter the WSN technology they use. This way, we will get, for example, HERA agents to move from a ZigBee node to a Bluetooth node.

Acknowledgments

This work was supported in part by the following Spanish Ministry of Science and Innovation Projects: TIN 2009-13839-C03-03 and IPT-430000-2010-035.

References

[1] M. Marin-Perianu, N. Meratnia, P. Havinga, L. de Souza, J. Muller, P. Spiess, S. Haller, T. Riedel, C. Decker, G. Stromberg, Decentralized enterprise systems: a multiplatform wireless sensor network approach, Wireless Communications, IEEE 14 (2007) 57–66.
[2] J. Sarangapani, Wireless Ad Hoc and Sensor Networks: Protocols, Performance, and Control, first ed., CRC, 2007.
[3] S. Mukherjee, E. Aarts, R. Roovers, F. Widdershoven, M. Ouwerkerk, Amiware: Hardware Technology Drivers of Ambient Intelligence, illustrated ed., Springer, 2006.
[4] E. Cerami, Web Services Essentials: Distributed Applications with XML-RPC, SOAP, UDDI & WSDL, first ed., O'Reilly Media, Inc., 2002.
[5] M. Wooldridge, An Introduction to MultiAgent Systems, second ed., Wiley, 2009.
[6] L.M. Camarinha-Matos, H. Afsarmanesh, A comprehensive modeling framework for collaborative networked organizations, Journal of Intelligent Manufacturing 18 (2007) 529–542.
[7] A.K. Dey, G.D. Abowd, D. Salber, A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications, Human-Computer Interaction 16 (2001) 97–166.
[8] R.S. Alonso, O. García, C. Zato, O. Gil, F. De la Prieta, Intelligent agents and wireless sensor networks: a healthcare telemonitoring system, in: Y. Demazeau, F. Dignum, J.M. Corchado, J. Bajo, R. Corchuelo, E. Corchado, et al. (Eds.), Trends in Practical Applications of Agents and Multiagent Systems, Springer, Berlin/Heidelberg, 2010, pp. 429–436.
[9] J.M. Corchado, J. Bajo, D.I. Tapia, A. Abraham, Using Heterogeneous Wireless Sensor Networks in a Telemonitoring System for Healthcare, Information Technology in Biomedicine, IEEE Transactions on Information Technology in Biomedicine 14 (2010) 234–240.
[10] D.I. Tapia, R.S. Alonso, F. De la Prieta, C. Zato, S. Rodriguez, E. Corchado, et al., SYLPH: an ambient intelligence based platform for integrating heterogeneous wireless sensor Networks, in: IEEE International Conference on Fuzzy Systems (FUZZ), 2010, pp. 1–8.
[11] R.S. Alonso, J.F. de Paz, Ó. García, Ó. Gil, A. González, HERA: a new platform for embedding agents in heterogeneous wireless sensor networks, in: Hybrid Artificial Intelligence Systems, Springer, Berlin/Heidelberg, 2010, pp. 111–118.
[12] G. Anastasi, M. Conti, M. Di Francesco, A. Passarella, Energy conservation in wireless sensor networks: a survey, Ad Hoc Networks 7 (2009) 537–568.
[13] L. Ardissono, G. Petrone, M. Segnan, A conversational approach to the interaction with web services, Computational Intelligence 20 (2004) 693–709.
[14] R.S. Alonso, D.I. Tapia, J.M. Corchado, SYLPH: a platform for integrating heterogeneous wireless sensor networks in ambient intelligence systems, International Journal of Ambient Computing and Intelligence (IJACI) 2 (2011) 1–15.
[15] D.I. Tapia, A. Abraham, J.M. Corchado, R.S. Alonso, Agents and ambient intelligence: case studies, Journal of Ambient Intelligence and Humanized Computing 1 (2010) 85–93.
[16] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, S. Weerawarana, Unraveling the web services web: an introduction to SOAP, WSDL, and UDDI, IEEE Internet Computing 6 (2002) 86–93.
[17] A. Pereira, N. Costa, C. Serôdio, Peer-to-peer Jini for truly service-oriented WSNs, International Journal of Distributed Sensor Networks 20 (11) (2011) 13, http://dx.doi.org/10.1155/2011/616838 (Article ID 616838).
[18] S.J. Russell, P. Norvig, J.F. Canny, J. Malik, D.D. Edwards, Artificial Intelligence: A Modern Approach, Prentice Hall Englewood Cliffs, NJ, 1995.
[19] F. Bellifemine, A. Poggi, G. Rimassa, Developing multi-agent systems with a FIPA-compliant agent framework, Software: Practice and Experience. 31 (2001) 103–128.
[20] D.L. Martin, A.J. Cheyer, D.B. Moran, The open agent architecture: a framework for building distributed software systems, Applied Artificial Intelligence 13 (1999) 91–128.
[21] K. Sycara, M. Paolucci, M. Van Velsen, J. Giampapa, The RETSINA MAS infrastructure, Autonomous Agents and Multi-Agent Systems 7 (2003) 29–48.
[22] A.L. Bauer, C.A. Beauchemin, A.S. Perelson, Agent-based modeling of host-pathogen systems: the successes and challenges, Information Sciences 179 (2009) 1379–1389.
[23] C. Carrascosa, J. Bajo, V. Julian, J.M. Corchado, V. Botti, Hybrid multi-agent architecture as a real-time problem-solving model, Expert Systems with Applications 34 (2008) 2–17.
[24] F. Pecora, A. Cesta, DCOP for smart homes: a case study, Computational Intelligence 23 (2007) 395–419.
[25] D.I. Tapia, J.M. Corchado, An ambient intelligence based multi-agent system for Alzheimer health care, International Journal of Ambient Computing and Intelligence (IJACI) 1 (2009) 15–26.
[26] M.L. Borrajo, J.M. Corchado, E.S. Corchado, M.A. Pellicer, J. Bajo, Multi-agent neural business control system, Information Sciences 180 (2010) 911–927.
[27] D.I. Tapia, J. Bajo, J.M. Corchado, Distributing functionalities in a SOA-based multi-agent architecture, in: 7th International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS 2009), Springer Berlin/Heidelberg, 2009, pp. 20–29.
[28] B. Baruque, E. Corchado, A. Mata, J.M. Corchado, A forecasting solution to the oil spill problem based on a hybrid intelligent system, Information Sciences 180 (2010) 2029–2043.
[29] J.M. Corchado, J. Bajo, Y. de Paz, D.I. Tapia, Intelligent environment for monitoring Alzheimer patients, agent technology for health care, Decision Support Systems 44 (2008) 382–396.
[30] R. Tynan, G. O'Hare, A. Ruzzelli, Multi-agent system methodology for wireless sensor networks, Multiagent and Grid Systems 2 (2006) 491–503.
[31] Y. Kwon, S. Sundresh, K. Mechitov, G. Agha, ActorNet: an actor platform for wireless sensor networks, in: Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems, ACM, Hakodate, Japan, 2006, pp. 1297–1300.
[32] P. Baker, V. Catterson, S. McArthur, Integrating an agent-based wireless sensor network within an existing multi-agent condition monitoring system, in: ISAP '09. 15th International Conference on Intelligent System Applications to Power Systems, 2009, pp. 1–6.
[33] F. Zboril, J. Horacek, P. Spacil, Intelligent Agent Platform and Control Language for Wireless Sensor Networks, in: EMS '09. Third UKSim European Symposium on Computer Modeling and Simulation, 2009, pp. 482–487.
[34] E.Y. Song, K.B. Lee, STWS: a unified web service for IEEE 1451 smart transducers, IEEE Transactions on Instrumentation and Measurement 57 (2008) 1749–1756.
[35] G. Holmes, M. Hall, E. Frank, Generating rule sets from model trees, in: Proc. of the 12th Australian Joint Conf. on Artificial Intelligence, pp. 1–12.

[36] S.L. Salzberg, C4.5: programs for machine learning by J. Ross Quinlan. Morgan Kaufmann Publishers, Inc., 1993, Machine Learning 16 (1994) 235–240.
[37] n-Core®: A Faster and Easier Way to Create Wireless Sensor Networks, 2011. <http://www.n-core.info/>.
[38] Y.-K. Huang, A.-C. Pang, A comprehensive study of low-power operation in IEEE 802.15.4, in: Proceedings of the 10th ACM Symposium on Modeling, Analysis, and Simulation of Wireless and Mobile Systems, ACM, Chania, Crete Island, Greece, 2007, pp. 405–408.
[39] P. Medagliani, M. Martalò, G. Ferrari, Clustered Zigbee networks with data fusion: characterization and performance analysis, Ad Hoc Networks 9 (2011) 1083–1103.
[40] M. Ilyas, R.C. Dorf, eds., The Handbook of Ad Hoc Wireless Networks, CRC Press, Inc., 2003.

Ricardo S. Alonso is a doctoral candidate at the Faculty of Sciences of the University of Salamanca (Spain). His research interests include wireless sensor networks, embedded devices, distributed systems and AI techniques. He received a M.Sc. in Intelligent Systems from the University of Salamanca (Spain) in 2009, and a graduate engineering degree in Telecommunications from the University of Valladolid (Spain) in 2007.

Dante I. Tapia is a researcher at the BISITE Research Group of the University of Salamanca, Spain. His research interests include ubiquitous computing, wireless technologies, distributed architectures and middleware systems. He received a PhD in Computer Science from the University of Salamanca (Spain) in 2009.

Javier Bajo received a PhD in Computer Science from the University of Salamanca (Spain) in 2007. He is an Assistant Professor at the University of Salamanca (Spain). He obtained the Information Technology degree at the University of Valladolid (Spain) in 2001 and Engineering in Computer Sciences degree at the Pontifical University of Salamanca (Spain) in 2003. He has been a member of the organizing and scientific committee of several international symposiums. He has also been co-author of papers published in recognized journal, workshops and symposiums.

Óscar García is a PhD student researching in the area of e-learning and ambient intelligence at the Faculty of Sciences of the University of Salamanca (Spain). His research interests include ubiquitous communications, wireless technologies and distributed systems. He received a graduate degree in Telecommunications from the University of Valladolid (Spain) in 2006.

Juan F. De Paz received a PhD in Computer Science from the University of Salamanca (Spain) in 2010. He is Assistant Professor at the University of Salamanca and researcher at the BISITE research group. He obtained a Technical Engineering in Systems Computer Sciences degree in 2003, an Engineering in Computer Sciences degree in 2005 at the University of Salamanca and Statistic degree in 2007 in the same University. He has been co-author of published papers in several journals, workshops and symposiums.

Juan M. Corchado is Dean at the Faculty of Sciences and leader of the BISITE Research Group of the University of Salamanca, Spain. His research interests include hybrid AI and distributed systems. He received a PhD in Computer Science from the University of Salamanca (Spain) in 1998 and a PhD. in Artificial Intelligence (AI) from the University of Paisley, Glasgow (UK) in 2000.