Embedding Reactive Hardware Agents into Heterogeneous Sensor Networks

Dante I. Tapia, Ricardo S. Alonso, Sara Rodríguez, Juan F. de Paz, Angélica González and Juan M. Corchado
Department of Informatics
University of Salamanca
Salamanca, Spain
{dantetapia, ralorin, srg, fcofds, angelica, corchado}@usal.es

Abstract - Information fusion systems can be improved by developing solutions that integrate distributed intelligent systems with context-aware technologies. In this sense, Wireless Sensor Networks is a key technology for gathering remote and heterogeneous context information. However, it is not easy to integrate devices from different technologies into a single network. Distributed architectures, such as Multi-Agent Systems, can facilitate integrating such heterogeneous devices. In addition, Multi-Agent Systems expand the sensors' context-aware capabilities changing their behavior dynamically and personalizing their reactions. This paper presents the new Hardware-Embedded Reactive Agents (HERA) platform, that allows developing information fusion applications where agents are directly embedded in heterogeneous wireless sensor nodes with small resources.

Keywords: Information fusion, intelligent systems, context awareness, wireless sensor networks, service-oriented architectures, multi-agent systems.

1 Introduction

Intelligent fusion systems capture and manage relevant information that surrounds them and constitutes the context [1]. Nowadays, there are many small, portable and non-intrusive devices [2] that allow agents to gather context-information in a dynamic and distributed way [3]. However, the integration of such devices is not an easy task. Therefore, it is necessary to develop innovative solutions that integrate different approaches in order to create flexible and adaptable systems.

Context-aware systems require the use of sensors strategically distributed over the environment. Sensor networks are used for gathering the information needed by intelligent environments, ranging from home automation to industrial applications [4]. Sensor networks need to be fast and easy to install and maintain [4]. It is possible to distinguish between two types of sensor networks: wired and wireless. Wireless Sensor Networks (WSN) are more flexible and require less infrastructural support than wired sensor networks. 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 [2]. The lack of a common architecture may lead to additional costs due to the necessity of deploying non-transparent interconnection elements amongst different networks [4]. Moreover, the developed elements are dependent on the application to which they belong, thus complicating their reutilization.

The implementation of distributed architectures has been presented as a solution to such problems [2]. In addition, the development of information fusion 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. Modern functional architectures like SOA (*Service-Oriented Architecture*) consider integration and performance aspects that must be taken into account when functionalities are created outside the system [5].

Agents and multi-agent systems is one of the most prevalent alternatives in distributed architectures which can help to distribute resources and reduce the central unit tasks [6]. A distributed agent-based architecture provides more flexible ways to move functions to where actions are needed, thus obtaining better responses at execution time, autonomy, services continuity, and superior levels of flexibility and scalability than centralized architectures [3]. In this sense, agents expand the sensors' context-aware capabilities changing their behavior dynamically and personalizing their reactions [1].

This paper describes the new HERA (*Hardware-Embedded Reactive Agents*) platform. HERA is an evolution of SYLPH (*Services laYers over Light PHysical devices*) [7], which has the ability of using dynamic and self-adaptable heterogeneous WSNs. In HERA, unlike other approaches, agents are directly embedded on the WSN nodes and their services can be invoked from other nodes in the same WSN or other WSN connected to the former one. As SYLPH, HERA focuses specially on devices with small resources to save CPU time, memory size and power consumption.

The next section presents the problem description that essentially motivated the development of HERA. Section 3 describes the main characteristics and components of HERA and its underlying platform, SYLPH, as well as some experiments done to test the performance of HERA. Finally section 4 presents the conclusions and future work.

2 Problem Description

This section discusses some of the most important problems of existing approaches that integrate agents into WSNs, including their suitability for constructing intelligent environments. This section also presents the strengths and weaknesses of existing platforms and analyzes the feasibility of a new alternative: HERA.

Any intelligent fusion system has to take into account the information about the context, which can be gathered by 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. Each sensor node is habitually formed by a microcontroller, a transceiver for radio or cable transmission and a sensor or actuator mechanism [2]. Some nodes act as routers, so that they can forward data that must be delivered to other nodes in the network. There are wireless technologies such as IEEE 802.15.4/ZigBee and Bluetooth that enable easier deployments than wired sensor networks [4], avoiding the need of wiring buildings and reducing the costs and disadvantages of the setup stage. Whilst traditional networks aim at providing high QoS (*Quality of Service*) transmissions, WSNs protocols concentrate their main efforts on energy saving. Thus, WSN nodes must include some power manager and certain smartness that increase battery lifetime by means of having worse throughput or transmission delay through the network [4].

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. For this reason, it is difficult for the system to dynamically adapt its behavior to changes in the infrastructure. In addition, excessive centralization negatively affects system functionalities, overcharging or limiting their capabilities. 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 their technology or the sub-network to which they belong.

One of the most prevalent alternatives in distributed architectures is agent and multi-agent systems. 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 [6]. 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 [8]. A Multi-Agent System (MAS) is defined as any system composed of multiple autonomous agents, where each agent is incapable of solving a global problem on its own, there is no global control system, the data is decentralized and the computing is asynchronous [6]. There are several agent frameworks and platforms [9] 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. Furthermore, agents can use reasoning mechanisms and methods in order to learn from past experiences and to adapt their behavior according the context [8].

The fusion of the multi-agent technology and WSNs is not easy due to the difficulty in developing, debugging and testing distributed applications for devices with limited resources. The interfaces developed for these distributed applications are either too simple or, in some case, do not even exist, which even further complicates their maintenance. Therefore, there are researches [10] that develop methodologies for the systematic development of muti-agent systems for WSNs. Furthermore, most of the works that relate multi-agent systems and WSNs talk about *Mobile Agents based on WSN* (MAWSN). For instance, the study presented by Hairong Qi, *et al.* [11] deals with planning routes through MAWSNs that consume the minimum amount of resources. The analysis of such a study is centered on the optimal design of the itinerary of the mobile agent. Fok, *et al.* [12] also describe their system, Agilla, as a mobile agent that facilitates the fast development of applications on WSNs. These two studies [11] [12] relating mobile agents and WSNs are very specific, while the HERA platform is not. The HERA platform produces a multi-agent platform that embeds agents into wireless sensor nodes, allowing them to be applied to different scopes and environments. Other studies, as this one described by Chen, *et al.* [13], try to reduce the redundancy of the data gathered by sensors from different types of networks by using mobile agents that *pack* the data. This achieves faster data delivering and energy savings in the reception and delivery of the data. Other research presented by Chen, *et al.* [14] adds to the previously described a MAWSN system that improves the communication of the nodes and mobile agents with regards to the client-server model. It also packs the data gathered by the sensor network, reduces the reception and delivery times and saves energy in devices with a low autonomy capacity. Along the lines of fusing or packing data gathered by sensors, Rajagopalan, *et al.* [15] use a

mobile routing agents' model to try to state the quality of the fused data and the cost of communication between the transmitter and the receiver. With their model, the authors explain the problem of multiple optimization goals, maximizing the detected signal energy and reducing the maximum energy consumption lost in the process. These three studies [13] [14] [15] focus on the fusing and optimizing the data communication process in a WSN through mobile agents.

The HERA 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 its networks. As a result of its underlying platform, SYLPH [16], HERA contemplates the possibility of connecting wireless sensor networks based on different radio and link technologies, whereas other approaches do not. 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 amongst the nodes that form part of the network (i.e. data link and physical layers). HERA can be executed over multiple wireless devices independently of their microcontroller or the programming language they use. HERA 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. This facilitates the inclusion of context-aware capabilities into intelligent fusion systems because developers can dynamically integrate and remove nodes on demand.

3 The HERA Platform

This section describes the new HERA (*Hardware-Embedded Reactive* Agents) platform and its underlying WSN-SOA platform, SYLPH (*Services laYers over Light PHysical devices*). Furthermore, some experiments conducted to test the performance of the HERA platform are presented and the results obtained from them are analyzed.

3.1 The Platform Description

HERA facilitates agents, applications and services communication through of using dynamic and self-adaptable heterogeneous WSNs. In HERA, agents are directly embedded on the WSN nodes and their services can be invoked from other nodes in the same network or other network connected to the former one. HERA is an evolution of the SYLPH platform [7]. SYLPH follows a SOA model [5] for integrating heterogeneous WSNs in intelligent systems. In SYLPH, the main aim is to distribute resources over multiple WSNs by modeling the functionalities as independent services. The information gathered by SYLPH nodes can be managed by intelligent agents by means of the integration of SYLPH into FUSION@ (*Flexible and User Services Oriented Multi-agent Architecture*) [8] [17]. Thus, the agents running on FUSION@ can use reasoning mechanisms to adapt their behavior to the context information obtained through SYLPH nodes. However, HERA takes a step ahead over SYLPH, embedding agents directly into the wireless nodes and allowing them to be invoked from other nodes either in the same network or another network connected to the original.

SYLPH covers aspects related 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 service directories for distributing registration and discovering services. Node registration is done in the corresponding WSN 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 SOA model was chosen in SYLPH 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) [5]. 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 a single 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 a set of intermediate gateways simultaneously connected to several wireless interfaces. 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 intelligent fusion systems because developers can dynamically integrate and remove nodes on demand.

SYLPH implements an organization based on a stack of layers [4]. 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 HERA agent platform adds its own agents layer over the SYLPH stack of layers. Thus, HERA takes advantage

of one of the main features of SYLPH: it can be run over any wireless sensor node regardless its radio technology or the programming language used for development. In addition, HERA agents running on WSNs with different radio technology can communicate amongst themselves through one or more SYLPH Gateways, as explained below. The different layers and protocols that form part of SYLPH and HERA are now described. These components can be shown on Figure 1, which depicts both platforms running over two ZigBee nodes.

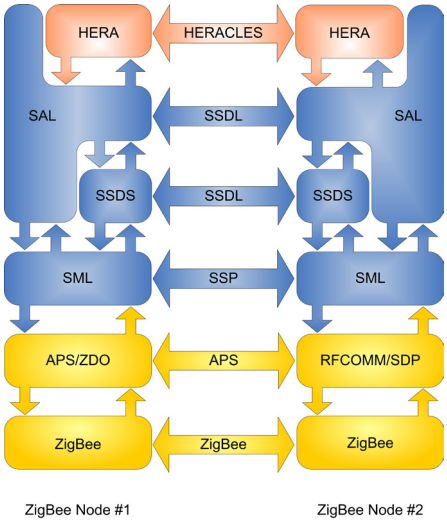

Fig. 1. The HERA platform over the SYLPH platform running over two ZigBee nodes.

The *SYLPH Message Layer* (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. This SML not only transports the services invocations over the network, but also the services discovering functions.

The *SYLPH Application Layer* (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. 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, SAL can use the interlayer services of the SML either directly or through the SSDS.

The *SYLPH Services Protocol* (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).

The *SYLPH Services Definition Language* (SSDL) is the IDL (*Interface Definition Language*) used by SYLPH. Unlike other IDLs such as WSDL (*Web Services Definition Language*) [5], 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. 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 lower cost. In most cases a few float point data for informing the status of a sensor is sufficient. Thus, most service definitions require only a few bytes. SSDL considers the basic types of data (e.g. integer, float or Boolean), allowing the use of more complex data structures, such as variable length arrays or character strings. This makes SSDL flexible enough to specify more complex services if required.

The *SYLPH Services Directory Sub-layer* (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 service 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. SSDS is also used by the SAL when registering a new service.

The *HERA Agents Layer* (or just *HERA*) allows running HERA agents. 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 below, 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*).

The *HERA Communication Language Emphasized to Simplicity* (HERACLES) 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 amongst nodes, they are encapsulated into simple SSDL frames using "HERA" as their service id field.

The behavior of SYLPH is essentially similar to that of any other service oriented architecture. However, SYLPH has several characteristics and functionalities that make it different from other models. 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. To do this, the service uses SSDL, which was created to work with limited resources nodes. Once the service has been registered in the SDN, it can be invoked by any application using 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 PC 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 particular service and its specification using SSDL. Because the aim of the architecture is to be as distributed as possible, it is possible to have more than one SDN in the same network, which makes it possible for redundancies to exist or services to be organized in different directories. The SDN can be stored in one of the network nodes, with a memory external to the microcontroller if necessary, or it can 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. Any node in the network can not only offer or invoke SYLPH services, but also includes SDN functionalities to provide services descriptions to other network nodes. SDNs include additional information about services, for example, a *Quality of Service* rate and a timestamp that represents the last time the SDN checked if the service was available. A SDN can be configured to check the services periodically or to allow service broadcasts.

Every agent platform needs some kind of *facilitator agent* that needs to be created before other agents are instantiated in the platform [9]. 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 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. To do this, HERA's facilitator agents, called HERA-SDNs (*HERA Spanned Directory Nodes*), 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. The first 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".

In order to start the HERA platform, an initial HERA-SDN must be created. 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 transmission), the HERA platform will not fail and will not need to be started again. 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.

Next we demonstrate a simple example of the use of SSDL to define a SYLPH service. This representation of SSDL is the human-readable version, used by developers when defining services, not the version actually transmitted. It is clear that its syntax is slightly similar to that of C language. Assembler and C are the most used programming languages for coding microcontroller firmware. However, C language is more human-readable than assembler. The example defines a simple service called *registerServiceOnFireAlarm*. This service is stored in a sensor device that belongs to a wireless network with the SYLPH platform running over it.

```
service registerServiceOnFireAlarm {
  input {
    uint16 threshold;
    servicepoint callback {
      output {
        boolean status;};};};
  output {
    boolean status;};};
```

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 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. When a node asks a SDN for the service definition, the SDN answers with a *definition* frame that describes the service identification, the address of the node that stores the service, the definition of the input and output parameters, and the QoS offered by the service. It has a SSDL header that specifies the SSDL data length and the type of frame it is (*registration*, *definition*, *invocation* or *response*). There is an *outputs mark* that denotes the input parameters and the output parameters that follow it. Once the invoker node knows the service definition, it can call the service by sending a SSP frame to the node that stores the service. This frame 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 parameters used. The SSP header includes the destination node SSP address. The response frame contains only the output parameters.

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 [7]: 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 the following:

```
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;};
```

However, similar to SYLPH, HERA agents transmit the more compact representation of HERACLES as frames. This kind of compact frames is what HERA agents transmit in a heterogeneous WSN based on HERA-SYLPH over the SSDL/SSP protocols.

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, each of which is connected to a distinct WSN. As with 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 either 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. Figure 2 shows a SYLPH Gateway for interconnecting a ZigBee and a Bluetooth network.

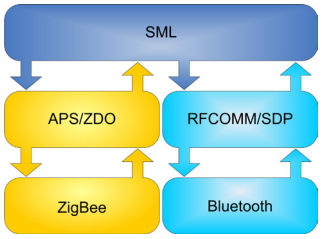

Fig. 2. A ZigBee-Bluetooth SYLPH Gateway.

Because 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 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. 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.

3.2 Experiments and Results

In order to test the HERA platform we deployed a distributed WSN infrastructure with HERA running over it. An experiment were carried out to evaluate the performance HERA, mainly to test how it handled the instances of HERA-SDNs and HERA Agents and the exchange of frames between agents through HERACLES. This experiment consisted of trying to start a platform with HERA over a ZigBee SYLPH network infrastructure.

The infrastructure consisted of a ZigBee network with 31 nodes, one acting as ZigBee coordinator and the rest as ZigBee routers. Each ZigBee node included an 8-bit C51-based microcontroller with 8448 bytes of RAM and 128 kilobytes of Flash memory and an IEEE 802.15.4/ZigBee transceiver. These devices include temperature and light sensors, some buttons and some LEDs, which allow them to be used as sensors and actuators. In fact, these devices were selected because we had already used them in previous developments [16] [17] and we intend to integrate them in future projects as well. The ZigBee nodes were distributed in a short-range simple mesh, with less than 10 meters between any router and the coordinator. Each time the ZigBee network was formed, 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. Each time the ZigBee network was formed, the SYLPH platform was started over it, with the ZigBee coordinator acting as a SDN and the 30 ZigBee routers acting as SYLPH nodes.

After the entire SYLPH network was correctly created the coordinator and SDN tried to instance a HERA-SDN. HERA-SDN instanced itself and started the HERA platform registering a special SYLPH service called "HERA" on the SDN stored on the same ZigBee coordinator node. Then, 10 of the 30 SYLPH nodes tried to instance one HERA agent, each of them in the HERA platform. Once the HERA-SDN and the 10 HERA agents were successfully instantiated, HERA-SDN started to "ping" every of the ten HERA Agents with a *request* HERACLES frame including an *inform-if* command and waiting for a *inform* frame as a "pong" response. Each HERA Agent was pinged by the HERA-SDN one time every 5 seconds during one hour. This experiment was run 50 times to measure the success ratio of the platform start and the agent instantiation. However, if the SYLPH network could not be correctly created the run was discarded and was 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 on Table 1.

Table 1. Results of the HERA performance experiments.

Total runs	55
SYLPH created correctly	53
HERA started correctly	50
All 10 HERA agents correctly instantiated	50
Total pings tried	7200
Ping-pongs not completed	15
Total restarted HERA agents in an hour	8

These results indicate that it is necessary to improve SYLPH creation and the instantiation of HERA Agents. In the first case, a better ARQ (*Automatic Repeat Request*) mechanism could increase SSP-over-WSN transmissions. In the second case, it is necessary to debug the implementation of the agents and fix errors. 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.

4 Conclusions and Future Work

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. 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 any kind of devices. HERA adds intelligence to sensors by means of light reactive agents, improving the experience of developers and users in context-aware technologies.

Current work includes the integration of the HERA platform into FUSION@ [8]. In this new approach, each wireless node implements some HERA agents that work inside FUSION@ as other software agents running on

platforms as JADE or RETSINA. This way, there is no difference between a software agent and a hardware agent. Future work consists of implementing this new approach in real scenarios, especially in systems already developed by the BISITE Research Group, as healthcare telemonitoring scenarios where SYLPH and FUSION@ were already tested [17] [16] [7]. In order to achieve this, HERA will be implemented on Bluetooth nodes to control biomedical sensors through HERA agents.

Acknowledgements

This project has been supported by the Spanish Ministry of Science and Technology project TIN 2009-13839-C03-03: Organizaciones Virtuales Adaptativas: Mecanismos, Arquitecturas y Herramientas (OVAMAH).

References

[1] 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.

[2] 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.

[3] M.L. Borrajo, J.M. Corchado, E.S. Corchado, M.A. Pellicer, and J. Bajo, *"Multi-Agent Neural Business Control System,"* Information Sciences (Informatics and Computer Science Intelligent Systems Applications An International Journal). Elsevier Science. Vol. 180, Issue 6, 2010, pp. 911-927.

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

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

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

[7] 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. Special Issue: Affective and Pervasive Computing for Healthcare. Vol. 5518, 2009, pp. 663-670.

[8] D.I. Tapia, S. Rodríguez, J. Bajo, and J.M. Corchado, *"FUSION@, A SOA-Based Multi-agent Architecture,"* International Symposium on Distributed Computing and Artificial Intelligence 2008 (DCAI 2008), 2009, pp. 99-107.

[9] K. Sycara, M. Paolucci, M. Van Velsen, and J. Giampapa, *"The RETSINA MAS Infrastructure,"* Autonomous Agents and Multi-Agent Systems, vol. 7, Jul. 2003, pp. 29-48.

[10] R. Tynan, G. O'Hare, and A. Ruzzelli, *"Multi-Agent System Methodology for Wireless Sensor Networks,"* Multiagent and Grid Systems, vol. 2, Jan. 2006, pp. 491-503.

[11] Y. Wu, and L. Cheng, *"A Study of Mobile Agent Tree Routes for Data Fusion in WSN,"* 2009 WRI International Conference on Communications and Mobile Computing, Kunming, Yunnan, China: 2009, pp. 57-60.

[12] C. Fok, G. Roman, and C. Lu, *"Mobile agent middleware for sensor networks: An application case study,"* In Proc. of the 4th Int. Conf. on Information Processing in Sensor Networks (IPSN'05), Los Angeles, California, USA, 2005, pp. 382-387.

[13] M. Chen, T. Kwon, Y. Yuan, Y. Choi, and V.C.M. Leung, *"Mobile agent-based directed diffusion in wireless sensor networks,"* EURASIP J. Appl. Signal Process., vol. 2007, 2007, pp. 219-219.

[14] M. Chen, T. Kwon, Y. Yuan, and V.C. Leung, *"Mobile Agent Based Wireless Sensor Networks,"* Journal of Computers, vol. 1, 2006.

[15] R. Rajagopalan, C.K. Mohan, P. Varshney, and K. Mehrotra, *"Multi-objective mobile agent routing in wireless sensor networks,"* In Proc. of IEEE Congress on Evolutionary Comp., 2005.

[16] D.I. Tapia, R.S. Alonso, J.F. De Paz, and J.M. Corchado, *"Introducing a Distributed Architecture for Heterogeneous Wireless Sensor Networks,"* Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living, 2009, pp. 116-123.

[17] R.S. Alonso, Ó. García, A. Saavedra, D.I. Tapia, J.F. de Paz, and J.M. Corchado, *"Heterogeneous Wireless Sensor Networks in a Tele-monitoring System for Homecare,"* Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living, 2009, pp. 663-670.