Introducing a Distributed Architecture for Heterogeneous Wireless Sensor Networks

Dante I. Tapia, Ricardo S. Alonso, Juan F. De Paz, and Juan M. Corchado

Departamento de Informática y Automática, Universidad de Salamanca, Plaza de la Merced, S/N, 37002, Salamanca, Spain
{dantetapia,ralorin,fcofds,corchado}@usal.es

Abstract. This paper presents SYLPH, a novel distributed architecture which integrates a service-oriented approach into Wireless Sensor Networks. One of the characteristics of SYLPH is that it 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 most of the application code does not have to reside in a central node. Furthermore, SYLPH allows the interconnection of several networks from different wireless technologies, such as ZigBee or Bluetooth. This paper focuses on describing the main components of SYLPH and the issues that lead to design and develop this new approach. Results and conclusions are presented after evaluating a preliminary version of this architecture.

Keywords: Distributed Architectures, Wireless Sensor Networks, Service Oriented Architectures, Embedded Devices.

1 Introduction

Whether in home automation, industrial applications or smart hospitals, sensor networks are used for collecting useful information for intelligent environments [1]. Sensor networks are made up of a set of devices called sensor nodes, each of which 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 [2], 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, Wireless Sensors Networks (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 [1].

This paper describes the *Services laYers over Light PHysical devices* (SYLPH) architecture and explains its main features and components. SYLPH is a functional architecture which integrates a SOA approach over WSNs for building systems that allow communicating devices from different technologies. The architecture focuses on distributing the systems' functionalities into independent services. This model provides a flexible distribution of resources and facilitates the inclusion of new functionalities in highly dynamic environments.

Next, the problem description is introduced and it is explained why there is a need for defining a new architecture. Then, the proposed architecture is described. Finally, the results and conclusions are presented, including future lines of work.

2 Problem Description

This section discusses some of the most important problems of existent functional architectures for WSNs, including their suitability for constructing intelligent and dynamic environments. This section also presents the strengths and weaknesses of existent developments and analyzes the feasibility of a new alternative: SYLPH.

There are different technologies for implementing WSNs, such as ZigBee, Bluetooth or Wi-Fi. However, their main problem is the difficulty when integrating devices from different technologies in a single network [2]. In addition, the lack of a common architecture may lead to additional costs due to the necessity of deploying non-transparent interconnection elements amongst different networks and technologies. Moreover, the developed elements (i.e. devices) are dependent of the application to which they belong, thus complicating their reutilization.

Excessive centralization of services negatively affects the systems' functionalities, overcharging or limiting their capabilities [3]. In classic functional architectures their modularity and structure are oriented to the systems themselves [4]. Otherwise, modern functional architectures, such as SOA, allow functionalities to be created outside the system. That is, as external services linked to the system. Thus, distributed architectures look for the interoperability amongst different systems, the distribution of resources and the independency on programming languages [5]. Services are integrated by means of communication protocols which have to be used by applications in order to share resources in the network [6]. The compatibility and management of messages that the services generate to provide their functionalities are important and complex elements in any of these approaches. Some developments try to reach integration between devices by implementing some kind of middleware, which can be implemented, for instance, as message-oriented middleware [7] or a multi-agents approach [8] [9]. However, these solutions require devices whose microcontrollers have large memory and high computational power, increasing costs and physical size. These drawbacks are very important regarding wireless sensor networks, as it is essential to deploy applications with reduced resources and low infrastructural impact.

SYLPH faces some of these issues by enabling an extensive integration of WSNs and providing a greater simplicity of deployment, optimizing the reutilization of the available resources in such networks. SYLPH integrates a SOA approach for facilitating the distribution and management of resources (i.e. services). A distributed 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]. Unfortunately, the difficulty in developing a distributed architecture is higher [8]. It is also necessary to have more complex system analysis and design, which imply more time to reach the implementation stage.

There are several attempts to integrate WSNs and a SOA approach [10] [11] [12]. In SYLPH, unlike those approaches, services are directly embedded on the WSN nodes and can be invoked from other nodes in the same network or other network connected to the former one. Moreover, in those developments it is not enough considered the necessity of minimize the overload of the services architecture on the devices. In this sense, SYLPH focuses specially on devices with small resources in order to save CPU time, memory size and energy consumption. For instance, SYLPH is able to run over ZigBee nodes having a C8051F121 microcontroller with only 8448 bytes of RAM and 128 kilobytes of Flash memory for program code. Furthermore, as said above, SYLPH contemplates the possibility of connecting WSNs based on different technologies (e.g. ZigBee and Bluetooth), whilst other approaches do not.

3 SYLPH: A New SOA-Based Architecture for WSNs

SYLPH (*Service laYers over Light PHysical devices*) is a distributed architecture which integrates a SOA approach over WSNs. The main objective of this proposal is to distribute resources over multiple WSNs by modelling the functionalities as independent services. As described by [13], "A SOA-based system is a network of independent services, machines, the people who operate, affect, use, and govern those services as well as the suppliers of equipment and personnel to these people and services". The term service can be defined as a mechanism that facilitates the access to one or more functionalities (e.g. functions, network capabilities, etc.). Services are linked by means of standard communication protocols that must be used by applications in order to share resources in the services network [6]. A SOA approach has been chosen because such architectures are asynchronous and non-dependent on context (i.e. previous states of the system) [14]. Thus, devices working on them do not take up continuously processing time and are freer to do other tasks or consume less energy. SYLPH is based on a SOA approach, but modifying this model to fit our requirements and designing goals.

Using SYLPH, a node designed over a specific technology (e.g. ZigBee or Bluetooth) can communicate to a node from a different technology. In this case, both WSNs are interconnected by means of a set of intermediate gateways connected simultaneously to several wireless interfaces. Such gateways are called SYLPH Gateways and can be, for instance, a personal computer with both Bluetooth and ZigBee network adapters (i.e. network cards). SYLPH allows applications to work in a distributed way and independently of the technology (i.e. network standard) used by each node. Thus, neither developers nor users have to worry about what kind of technology each node in the system uses.

SYLPH implements an organization based on a stack of layers. A layer is a set of conceptually similar functions that offers some services to the higher layer over it, but hiding to it the details of the implementation of such services [1]. Moreover, each layer in one node communicates with its peer in another node through an established protocol [2]. The stack layers organization allows to SYLPH layers to be reutilized over multiple WSNs' technologies or standards. Thus, the SYLPH layers are added over the application layer of each WSN stack. Figure 1 shows the basic schema of the communication between two ZigBee devices using the SYLPH architecture layers. In such Figure, it can be seen the next layers: the SYLPH Message Layer (SML), the Application Layer (Apps) and the SYLPH Services Directory (SSD) Layer. The SML layer offers to the upper layers the possibility of sending asynchronous messages between two wireless devices through the SYLPH Services Protocol (SSP). Such messages specify the origin and target nodes and the service invocation in a SYLPH Services Definition Language (SSDL) format. SSDL describes the service itself and its parameters to be invoked. The Application Layer can communicate directly amongst devices using the SML layer or by means of the SYLPH Services Directory (SSD) layer, that uses, in turn, the mentioned SML layer. The SSD layer is used by nodes for locating services on other nodes in the network. SSD nodes act as directories of the services offered by the network nodes. Thus, any node in the network can ask a SSD for the location of a certain service. As the Apps layer, SSD layer uses the SSDL language as protocol for register and locate services over SSP messages.

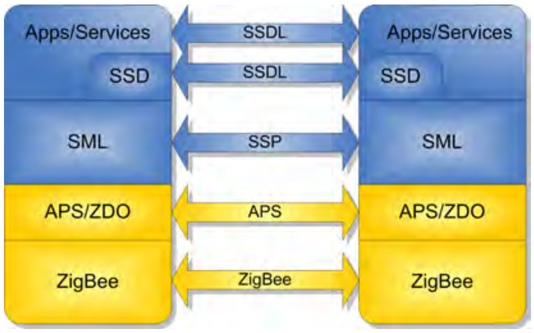

Fig. 1. Communication between two ZigBee devices using SYLPH architecture layers

SSDL is the IDL (Interface Definition Language) [5] used by SYLPH. Distributed architectures use an IDL in order to enable communication between software components regardless their programming language or hardware implementation. Unlike other IDLs as WSDL (Web Services Definition Language), based on XML and used on Web Services [14], SSDL does not use so many intermediate separating tags and the order of its elements is fixed. Using a simple IDL allows utilizing nodes with fewer resources, less power consumption and at a lower cost; all of them key challenges when using embedded devices. In most cases it is enough with a few float point data for informing the status of a sensor. Thus, most service definitions require only a few bytes. SSDL considers the basic types of data (e.g. Integer, Float or Boolean), allowing more complex data structures as variable length arrays or character strings.

The behavior of SYLPH is in essence similar to other Service Oriented Architectures [14]. However, SYLPH has several characteristics and functionalities that make it different to other models [5]. The first step in SYLPH begins when a service registers itself on the SSD and informs of its location in the network, the parameters it requires and the types of output values returned in the response message after its execution. In order to do that, it is used SSDL which has been created to work with limited resources nodes.

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

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

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 languages) and microcontroller where service will run. When the node registers its service in a SSD, SYLPH layers do not transmit the human-readable SSDL message, but a more compact array of bytes which describes the service and how to invoke it from other nodes. Once the service has been registered in the SSD, it can be invoked by any application by means of SYLPH. Both the SSD and the services can be stored in any node of any WSN that forms part of the SYLPH network. Thus, the developers decide which nodes will implement each part of the distributed application. Any node in the network can ask the SSD for the location of a certain service and its specification using SSDL. The sequence diagram of the fire alarm using the `registerServiceOnFireAlarm` service defined above is shown in Figure 2. In such Figure, it can be seen how node 2 registers the mentioned service on node 0 (SSD). Node 1 asks node 2 for the service location and definition. Then, node 1 invokes the service

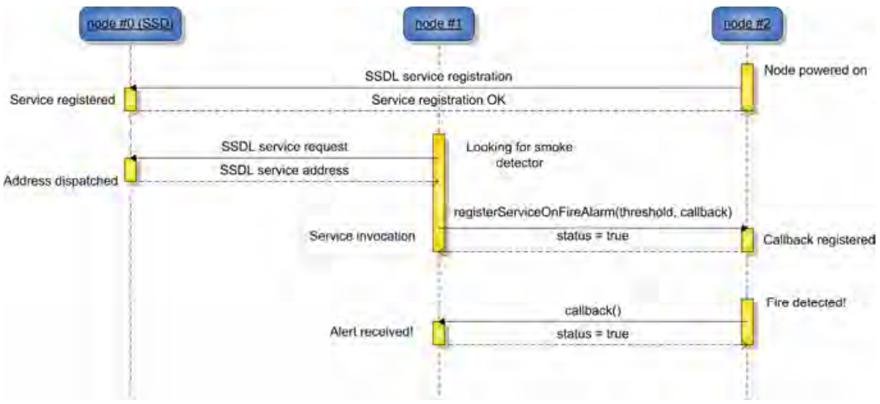

Fig. 2. Example of service registration and invocation

through a message to node 2. On such message node 1 registers some service to act as callback, as defined before. When the smoke sensor on node 2 detects a fire, it invokes callback on node 1.

With the aim of the architecture to be as distributed as possible, it is allowed to be more than one SSD in the same network, so that can exist redundancy or services organized in different directories. Any node in the network can not only offer or invoke SYLPH services but also include SSD functionalities to provide services descriptions to other network nodes. SSDs stores an entry for each service including its invocation and response descriptions, what node offers it, a Quality of Service (QoS) rate and a timestamp that represents the last time the SSD checked the service was available.

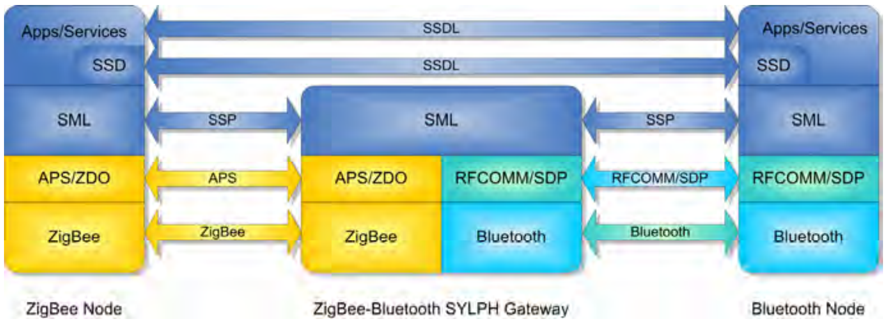

Fig. 3. SYLPH over ZigBee and Bluetooth networks

As mentioned above, several heterogeneous WSNs can be connected using a SYLPH Gateway. Figure 3 shows a ZigBee network and a Bluetooth network working together using SYLPH over them. The SYLPH Gateway is connected to several sensor networks through different hardware interfaces. Thus, it can forward messages amongst the different networks to which it belongs. From the Application Layer's point of view, there is no difference between invoking a service stored in a node in the same sensor network or invoke another one stored in a remote node in a different network. In the example of the Figure 3, if a ZigBee node invokes a service in a Bluetooth node, the ZigBee node will look for the service in a SSD belonging to the ZigBee network. The entry stored in the services table of the SSD points, in fact, to the SSP address of the SYLPH Gateway. When the ZigBee node invokes the service in the Bluetooth node, the SYLPH Gateway forwards the call message to the Bluetooth node through its Bluetooth hardware interface. The inverse process is done by the SYLPH Gateway in order to forward the response message from the Bluetooth node to the ZigBee one.

4 Results and Conclusions

SYLPH allows integrating heterogeneous WSNs in a distributed way. It has been taken into account a SOA approach for designing this architecture. Thus, functionalities are modeled as independent services offered by nodes (i.e. wireless devices) in

Table 1. Performance of SYLPH networks formation

Factor	Only ZigBee WSN	Dual WSN
Runs network successfully formed	48 (96%)	42 (84%)
Average number of successfully registered services	39.12 (97.8%)	39.40 (98.5%)

the network. These services can be invoked by any node in the SYLPH infrastructure, regardless the physical WSN which they belong (e.g. ZigBee, Bluetooth, etc.). In addition, SYLPH nodes do not need large memory chips or fast microprocessors. The easy deployment of SYLPH-based systems reduces the implementation costs in terms of development and infrastructure support.

Several experiments were carried out to evaluate the performance of SYLPH, mainly to test the network formation and the services registration. Table 1 shows the main results of two different experiments. The first experiment consisted on trying to form a ZigBee WSN using SYLPH. Such network was intended to be made up of 20 ZigBee nodes, two of them acting as SSDs. Each node was instructed to try to register one service on each SSD after joining the network. This test was run 50 times in order to measure the network successfully formed ratio and the services successfully registered ratio. The second experiment consisted on a dual SYLPH network made up of one 10-node ZigBee network and another 10-node Bluetooth network, both of them interconnected by means of a SYLPH Gateway. There were also two nodes in this experiment acting as SSDs, one in each WSN. The introduction of the SYLPH Gateway makes harder the formation of the whole SYLPH network. However, once the network is successfully made up, there is almost no difference in the service successfully registered ratio. In addition, the SSDs worked correctly over the hybrid SYLPH network.

Future work on SYLPH includes an improved SYLPH Gateways performance and support for other WSNs different from ZigBee or Bluetooth (e.g. Wi-Fi). In order to reduce design and implementation times, it is in progress the development of a tool for generating code skeletons from the human-readable SSDL language, so that the building and delivering of SSDL frames will be coded directly from the services definitions. We are currently exploring alternative case studies for applying this architecture and demonstrate that the approach presented is flexible enough to be implemented in other scenarios. However, one main issue to be taken into account is that the architecture is still under development so it is necessary to define it by means of formal analysis and design methodologies and tools.

Acknowledgments. This work has been supported by the Spanish Ministry of Science and Technology project TIN2006-14630-C03-03.

References

1. Sarangapani, J.: Wireless Ad hoc and Sensor Networks: Protocols, Performance, and Control. Control Engineering Series (2007)
2. Ilyas, M., Mahgoub, I.: Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems, 1st edn. CRC, Boca Raton (2004)

3. Fuentes, R., Gómez-Sanz, J.J., Pavón, J.: Managing Contradictions in Multi-Agent Systems. IEICE Trans. Inf. Syst. E90-D(8), 1243–1250 (2007)
4. Molina, J.M., García, J., Jiménez, F.J., Casar, J.R.: Cooperative Management of a Net of Intelligent Surveillance Agent Sensors. International Journal of Intelligent Systems 18(3), 279–307 (2003)
5. Cerami, E.: Web Services Essentials Distributed Applications with XML-RPC, SOAP, UDDI & WSDL, 1st edn. O'Reilly & Associates, Inc., Sebastopol (2002)
6. Ardissono, L., Petrone, G., Segnan, M.: A Conversational Approach to the Interaction with Web Services. Computational Intelligence 20, 693–709 (2004)
7. Souto, E., Guimarães, G., Vasconcelos, G., Vieira, M., Rosa, N., Ferraz, C., et al.: Mires: a Publish/Subscribe Middleware for Sensor Networks. Personal Ubiquitous Computing 10(1), 37–44 (2005)
8. Molina, J.M., Herrero, J., Jiménez, F.J., et al.: Fuzzy Reasoning in a Multiagent System of Surveillance Sensors to Manage Cooperatively the Sensor-to-Task Assignment Problem. Applied Artificial Intelligence 18, 673–711 (2004)
9. Pavón, J., Gómez, J., Fernández, A., Valencia, J.J.: Development of Intelligent Multisensor Surveillance Systems with Agents. Robot. Auton. Syst. 55(12), 892–903 (2007)
10. Meshkova, E., Riihijärvi, J., Oldewurtel, F., Jardak, C., Mähönen, P.: Service-oriented Design Methodology for Wireless Sensor Networks: A View Through Case Studies. In: Proceedings of the IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing, pp. 146–153 (2008)
11. Yang, C.C., Lin, C.M., Yu, C.Y.: Integration of Web Services with Mobile Home Automation. Journal of Internet Technology 7(3), 269–273 (2006)
12. Song, E.Y., Lee, K.B.: STWS: A Unified Web Service for IEEE 1451 Smart Transducers. IEEE Transactions on Instrumentation and Measurement 57(8), 1749–1756 (2008)
13. OASIS: Reference Architecture for Service Oriented Architecture Version 1.0 (2008)
14. Curbera, F., Duftler, M., Khalaf, R., Nagy, W., Mukhi, N., Weerawarana, S.: Unraveling the Web services Web: An introduction to SOAP, WSDL, and UDDI. IEEE Internet Computing 6(2), 86–93 (2002)