Journal of Network and Computer Applications 35 (2012) 562–576

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

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

Knowledge-Aware and Service-Oriented Middleware for deploying pervasive services

Iván Corredor*, José F. Martínez, Miguel S. Familiar, Lourdes López

Departamento de Ingeniería y Arquitecturas Telemáticas, Escuela Universitaria de Ingeniería Técnica de Telecomunicación, Universidad Politécnica de Madrid, Ctra. Valencia, Km. 7, 28031 Madrid, Spain

ARTICLE INFO

Article history:
Received 2 July 2010
Received in revised form
20 January 2011
Accepted 27 May 2011
Available online 6 June 2011

Keywords:
Internetworking architecture
Service-Oriented Middleware
Knowledge Management
Internet of Things
Wireless Sensor and Actuator Network

ABSTRACT

Many applications and services have emerged in the frame of new Internet of Things paradigm. This novel view has opened the Web services to a variety of devices especially to tiny and resource-constrained devices. Wireless Sensor and Actuator Networks belong to that kind of devices. Those networks have become one of the more promising technologies to take part in the Future Internet. However, the integration of Sensor and Actuator Networks into the Service Cloud is a hard challenge requiring specific new architectures and protocols. This paper presents a middleware approach addressing this important issue. A Knowledge-Aware and Service-Oriented Middleware (KASOM) for pervasive embedded networks is proposed. The major aim of KASOM is to offer advanced and enriched pervasive services to everyone connected to Internet. In this sense, KASOM implements mechanisms and protocols which allow managing the knowledge generated in pervasive embedded networks in order to expose it to Internet users in a readable way. General functional requirements of embedded sensor and actuator platforms have been taken into account when designing KASOM, with special attention in energy consumption, memory and bandwidth. The KASOM evaluation and validation will be demonstrated through a real Wireless Sensor and Actuator Network deployment based on integral healthcare services in a sanatorium.

1. Introduction

The concept of *Ubiquitous Computing* was introduced by Mark Weiser. Weiser (1993) stated that for 30 years the most interfaces had been designed following the *Spectacular Machine* model. A less followed research line till 90's was the *Invisible Computing*, or Ubiquitous Computing, so called by Weiser. Weiser's idea was to integrate the computers with the physical environment so that users could use them without actual awareness about their existence. This computing paradigm has been called the *Third Computing Paradigm*. Pervasive embedded networks fit perfectly with the Third Computing Paradigm: they are networks composed of tiny nodes being able to hide in the environment; they are *invisible* networks according to Weiser's conception.

Perhaps, Wireless Sensor and Actuator Networks (WSANs) are the most paradigmatic kind of pervasive embedded network. WSANs are networks composed of hundred, even thousand, of autonomous small nodes spatially distributed. Those devices use sensors and actuators in order to perform an amount of activities such as cooperative monitoring of physical and environmental conditions, as temperature, sound, vibrations, pressure, movement, or gases in different locations (Mattern, 2004).

Early Wireless Sensor Networks were designed for military uses such as surveillance in battle field. Currently, the complexity of deployments in WSANs are increasing and evolving quickly, with addition of new hardware components and software architectures. Such innovative WSANs are reflected on a broad of environments where a variety of kinds of sensors and actuator have to be deployed for multiple context-aware applications within a single physical space. In a recent future, we will be able to find WSAN deployments in malls, office, homes, or hospitals, mostly related to energy saving, logistic, security, or healthcare. The latter will be an important topic in the next decade. In 2020s the healthcare demand will hugely grow majorly from elderly and disable people. The World Health Organization has foreseen that in the year 2025 at least 1 billion people in the world will be 60 years old and this value will be duplicated by 2050. The 80% of this people will be concentrated in developed countries (World Health Organization, 2007). Therefore, new healthcare systems have to be designed to provide more effective ways for the distribution of resources both economical and human. WSANs are particularly interesting platforms to reach this objective.

The new WSAN approaches will have to solve some critical challenges. The huge diversity of information transactions of

* Corresponding author. Tel.: +34 91 336 55 20; fax: +34 91 336 78 17.
E-mail address: icorredor@diatel.upm.es (I. Corredor).

1084-8045/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jnca.2011.05.009

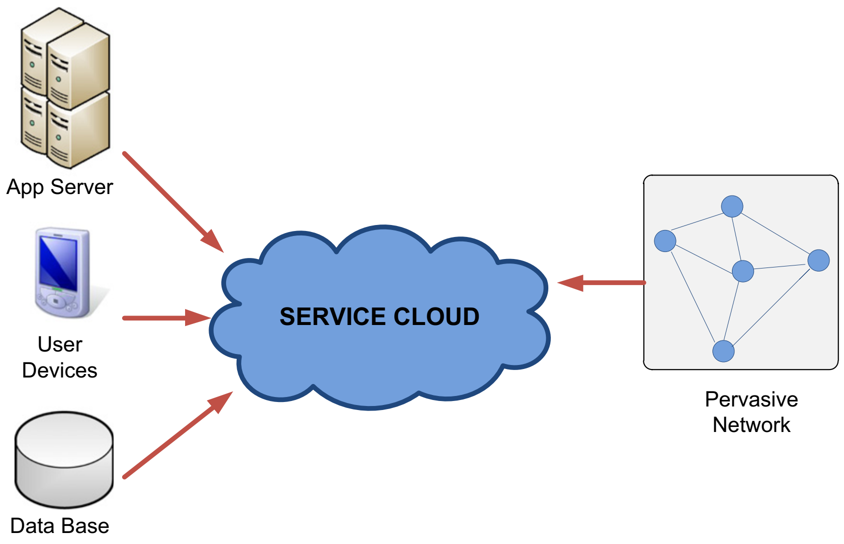


Fig. 1. Integration of pervasive embedded networks into the Service Cloud.

WSN中大规模的情景感知应用的信息传递

context-aware applications makes it essential to design several mechanisms which allow managing such diverse of information sources. Typically, those mechanisms are solved by addressing two approaches: (a) in-network data processing, or (b) data processing using a middleware layer interceding in communication between WSANs and business applications.

In-network data processing is the general procedure of gathering data and routing through the network, processing data in sensor or intermediate nodes in order to optimize the resources of the network, particularly from point of view of energy consumption for optimizing the network lifetime (Gomez et al., 2007). In this respect, many works has been performed in recent years such as those regarding aggregation, metadata negotiation or data fusion (Intanagonwiwat, 2000; Heinzelman et al., 1999; Kulik et al., 2002; Chih-Min and Hsiao, 2009; Feng and Xu, 2009). In spite of the work done around this issue, it is a big research topic yet. However, most of the times, in-network processing is specifically designed for a single type of tasks and very often it is not generic enough to support multiple services demanded by current context-aware applications. Adding new sensor capabilities, custom protocols, data types, and new functionalities could be difficult due to lack of access to the node resources.

Middleware architectures for WSANs should be able to provide mechanisms to support multiple applications with a variety of objectives in order to deploy more generic systems rather than *ad hoc* systems managed by the above mentioned in-network mechanisms. These middleware architectures will manage any type of data generated in the WSAN to provide good Quality of Service (QoS) level to applications. The major objective of a middleware layer for WSANs is to hide the heterogeneity in terms of hardware and software running on it with the aim of offering all those resources to the applications in a standardized way. There are two fundamental features needed to reach such objectives. Firstly, the data generated in the nodes of the network have to comply with a concrete semantic structure. The second feature consists of creating a programmatic way of describing the node functionalities. These characteristics will allow discovery and exposing all network resources provided by the network as well as developing extensible and reusable applications using those resources.

1. 固定的语义结构
2.编码的方式描述节点功能

这些特性将允许发现和暴露出网络提供的所有网络资源以及使用这些资源开发可扩展和可重用的应用程序。

From the appearance of the *Internet of Things* paradigm, the discussed characteristics have taken more expansion because of the need of integrating pervasive networks with large scale networks like Internet. The capabilities commented before are perfectly assumed by conventional devices currently connected to Internet through Web services: application can get structured data as response of web method calls and the Web service description allow defining service functionalities programmatically. However, the resource cost of such Web service mechanisms is not feasible for tiny resource-constrained nodes composing pervasive embedded networks. In that sense, the major challenge is to address similar capabilities in a cost effective way in order to be acceptable by pervasive embedded networks (Fig. 1).

The aim of this research is to design a middleware approach which enables Service-Oriented Computing in pervasive network based on lightweight REST (Fielding, 2000) Web services. We propose a *Knowledge-Aware and Service-Oriented Middleware* (KASOM) for pervasive embedded networks, especially for WSANs, based on two fundamental keys. The first one consists of provisioning several *Knowledge Management* services which enable an effective way of managing the amount of information generated in a so heterogeneous network as the WSAN. The Knowledge Management services are founded on complex reasoning mechanisms and protocols based on the WSAN's *Contextual Model*. That Contextual Model represents a semantic description of low and high level resources of the WSAN. The second feature provides means for enabling Service-Oriented Computing capabilities on pervasive embedded nodes, which allow creating an appropriate integration point between pervasive embedded networks and conventional networks by means of an in-network REST approach.

知识管理服务

第二个特征提供了面向服务的计算能力的嵌入式节点，其允许通过网络内的适当方法来创建嵌入式网络和传统网络之间的适当集成点。

The rest of the paper is organized as follows. Section 2 discusses related works emphasizing their drawbacks. Section 3 defines the foundation of our internetworking architecture. Section 4 describes the conceptual model of the KASOM architecture. In Section 5 the programming model of KASOM is explained through examples. Section 6 describes the validation results of the overall architecture over a real healthcare scenario. We discuss the results of this research and some possible future works in Section 7.

2. Related works

Several works can be found in literature addressing middleware layers interceding between low and high layers in wireless embedded sensor platforms. In this section, both general purpose middleware and specific context-aware middleware are described.

Their main features and drawbacks are emphasized with the aim of indentifying current problems to be solved for the next generation of sensor and actuator networks based on Service-Oriented Architectures (SOA).

2.1. General purpose middleware for WSN

Traditionally middleware layers have been used to bridge the gap between applications and low-level layers to solve many issues from the development of distributed systems. The major purpose of a middleware layer in WSN is to hide the great heterogeneity of resources by offering a single and usable interface. A middleware model allows performing an easy development of sensing-based applications as well as its deployment and maintenance (Römer et al., 2002). The Reconfigurable, Ubiquitous, Networked Embedded System project (RUNES) (RUNES Consortium) tries to solve the common challenges (mostly the maintenance) using a component-based programming model. This model is based on encapsulated units interacting between them through "interfaces" and "receptacles". This model allows developing scalable, self-adaptive, self-configuring, and robust applications. The Middleware Linking Applications and Networks middleware (MiLAN) (Heinzelman et al., 2004) allows the applications to specify their QoS needs and to adjust the network characteristics to increase WSN lifetime while still meeting those needs. In spite of MiLAN tackling the challenge of QoS requirements, its programming model is not suitable for as heterogeneous platforms as WSNs. Cougar project is a middleware, which adopts a database approach where WSN is managed like a "virtual" relational database and the operations are implemented in forms of queries that are coded like SQL language. This approach is very advisable for large sensor networks. However, the dynamic nature of large-scale sensor networks could be a problem for a centralized proposal that Cougar use to keep a global knowledge of the network. System Information Networking Architecture (Srisathapornphat et al., 2000) (SINA) is a more complex database approach than Cougar. SINA makes requests in a form of SQL statements. Besides, SINA implements low-level mechanisms consisting of hierarchical clustering of nodes for efficient data aggregation. On the other hand, it exposes problems, as Cougar, when trying support distributed systems heterogeneity.

Some pitfalls can be found in the middleware proposal described in this section. Firstly, aggregation is the only in-network processing performed by those middleware proposals. A more complex reasoning (e.g. fuzzy logic) in order to process information is not addressed by the sensor nodes. Dynamic behavior in nodes and sensors as well as nodes' mobility are not clearly supported. This lack prevents from developing more robust systems and composing information about the WSAN's features and sensor data types managed by it in order to build high level contextual information of the whole system in a dynamical way.

2.2. Context-aware middleware

Context-aware applications are the trend under pervasive computing environments as WSAN-based ones. Contextual information in smart environments is usually gathered through the use of heterogeneous sensors distributed throughout the space. Therefore, it is necessary to found mechanisms in order to get the contextual information from the environment in a robust and structured way. These mechanisms are usually achieved by an approach based on three phases in charge of managing the contextual information: *discovery*, *acquisition*, and *reasoning*. Since each application interprets the underlying sensor network differently according to their objectives, middleware layer has to manage different contextual requirements.

In Taherkordi et al. (2009) a Framework-based middleware is proposed. That Framework is able to manage contextual information using an architecture divided in three sub-layers: *Context Adapter*, *Context Process* and *Context Provider*. In this proposal there are *context nodes* which provide context information using five primitive components: *Context Process*, *Context Reasoning*, *Context Configuration*, *Activity Manager*, and *Message Manager*.

In Ranganathan et al. (2003) a middleware for *contextual agents* was proposed. This middleware layer was thought in order to compose an execution Framework suitable for agents in Ubiquitous Computing environments. The Contextual Model implemented by this proposal allows using different reasoning mechanisms like first order logic and temporal logic. The types of agents and services coexisting in this Middleware are the following ones: *Context Providers*, *Context Synthesizer*, *Context Consumer*, *Context Provider*, *Searching Service*, *Historic Context Service*, and *Ontology Service*. In Mudasser et al. (2009) a service-oriented model for semantics-based data management in WSANs is proposed. This proposal tries to hide platform heterogeneity providing interfaces to its various components via semantics services. A semantic layer is added to the middleware architecture to achieve that objective. The semantic layer builds and manages two ontologies: data ontology and process ontology. By means of these ontologies, the proposed model encapsulates both sensor data and services as well as provides interfaces for service advertisement and discovery.

We can found some features lacking in previous middleware approaches most of them related to the network service management. In first place, dynamics is again a strong deficient in service self-discovery and announcement. The discovery mechanisms and protocols implemented by these approaches are generally poor when treating with new services added during runtime (e.g. when a node is newly added or additional sensors or actuator are added to a node). On the other hand, the announcement protocol of services managed by the network from service providers to the Gateway (point interconnecting the business applications with the sensor network) are not well defined in most of cases. This characteristic makes difficult the interaction between the WSANs and the outside world, i.e. Internet or other conventional local networks.

3. Foundation of the internetworking architecture

The approaches for pervasive networks that have been proposed up to now (Taherkordi et al., 2009; Mudasser et al., 2009; Lionel et al., 2009) did not achieve to solve the main challenges derived from discovering and exposing network's resources. Our middleware approach addresses the management of resources in pervasive embedded networks, especially WSANs, by means of Knowledge Management (KM) services. The mechanisms and protocols supporting KM services are characterized by collecting and exposing well-structured information about WSAN's resources. KM services allow making and exposing the WSAN's Contextual Model, which is the starting point for achieving the major goal of our research: to bridge the gap between WSANs and the Service Cloud.

In this section, the key characteristics of our internetworking architecture are explained. These main points are related to the *reference infrastructure*, *Contextual Model*, *in-network agents*, and *user applications*.

3.1. Reference infrastructure

The reference infrastructure consists of diverse hardware devices strategically deployed. These devices range from tiny

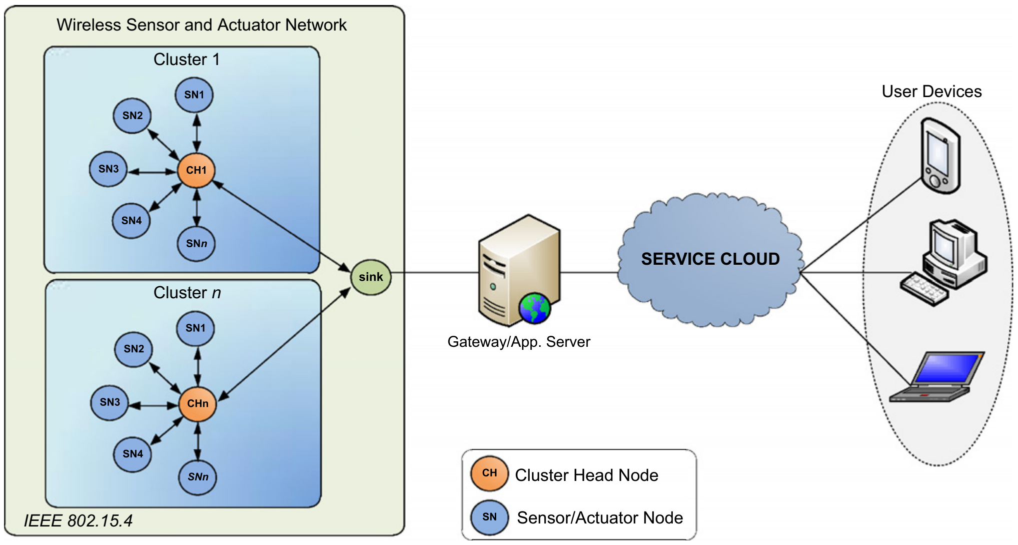


Fig. 2. The reference infrastructure.

nodes, which compose the WSAN infrastructure, to Gateways which gather, classify, and store the information received from the pervasive environment. Figure 2 shows the reference infrastructure. Note that there are three kinds of roles in the WSAN: (i) Sensor and Actuator (SA), (ii) Cluster Head (CH), and (iii) Sink. In summary, SA nodes provide simple services which are discovered by CH nodes in order to compose and orchestrate more complex services called *contextual services*. Sink nodes are in charge of discovering every service exposed by the WSAN being offered to the Gateway. Moreover, Gateways work as application servers in order to receive and dispatch service requests from user applications to the WSAN and to return results from those requested services.

CH负责将SA提供的简单服务组成更复杂的语义服务，汇点负责发现WSN的服务，提供到网关。

Different kind of sensorial hardware and actuators will be connected to the nodes of the WSAN in order to support a number of different services. This means that the same infrastructure will be available for a wide range of applications what made the WSAN cost-effective and evolutionary. KM services provided by KASOM will enable this important characteristic through its mechanisms and protocols, which allow managing appropriately the WSAN's resources. During start up phase, WSAN's nodes send presence announcements carrying descriptive information about the services and resources offered by them. Such information is mapped over semantic pieces, which are processed in intermediate nodes (usually CH nodes) or gathered directly into the sink node. The gathering and processing of semantic description about in-node resources will contribute to generate the Contextual Model of the WSAN.

网关=应用服务器，接收和分发用户的服务请求到WSN，返回相应的结果给用户

3.2. *WSAN's Contextual Model*

Every node of the WSAN has a repository in which semantic information about low level resources (hardware resources: sensors, actuators, memory, energy, etc.) and high level resources (logical resources: *simple services*, *contextual services* and *service composition rules*) are stored. In our approach this semantic information makes part of the Knowledge of the system. Repositories storing the Knowledge of the system are called *Knowledge Bases* and the set of such information makes the WSAN's Contextual Model.

这些都是系统知识的一部分

The Knowledge Bases (KB) will be hierarchically organized according to the mentioned cluster topology storing different information about the Contextual Model of the overall WSAN. In this base scheme there are three kinds of KBs: *in-Node Knowledge Base*, which stores information about simple services as well as available physical resources; *in-Cluster Knowledge Base*, which stores information about simple services provided by SA nodes, service composition rules, workflow plans for managing the orchestration of contextual services as well as a list of available physical resources belonging to the CH node; *WSAN Knowledge Base*, which collects and stores semantic pieces in order to compose the Contextual Model of the whole WSAN into an ontology. This process results in a standardized document which exposes the WSAN services as Web services. The organization schema of the KBs is shown in Fig. 3.

The repository on the top of the hierarchy is located in the Gateway which interconnects the WSAN with conventional networks. This KB is intended to be structured as ontology over a RDF/OWL 2 (R.C.W. Group; Motik et al., 2009) document. Such ontology will be parsed in order to create a WSDL 2.0 document (Chinnici et al., 2007). That WSDL document will allow exposing the WSAN's services in a REST fashion so that they can be accessed by HTTP methods (GET, PUT, POST, and DELETE). The mechanisms previously explained is a *Virtualization Process* of pervasive environments that allows treating a WSAN as conventional Web services which can be easily integrated with enterprise systems through the Internet.

WSDL网络服务描述语言

3.3. *In-network agents*

KASOM enables an execution platform over which a number of different in-network agents can be run. Those in-network agents are called Perceptual Reasoning Agents (PRAs). PRAs will be deployed both in SA nodes and CH nodes according to the cluster topology (to distinguish they are called N-PRAs and C-PRAs, respectively). KASOM provides to developers a common API to register PRAs in the system hiding the details of underlying software as well as the allocation and management of resources.

PRAs will take advantage from the cluster topology used in the WSAN. Using the API provided by KASOM, PRAs that are running on CH nodes are able to set up service composition rules to generate workflow plans managing contextual services. When invoking contextual services, high level contexts can be identified and classified (e.g. person/object detection, pollution level, average temperature). Every CH node will announce the high level services which are available according the service composition rules provided by C-PRAs. Such announcements of services will be collected by the sink node, and from there, to the Gateway which will be in-charge of exposing the WSAN's services to user application as it was explained in previous section. In a usual working, the WSAN only exposes contextual services since normally user applications

Sink node收集服务announcement到gateway，gateway负责把服务暴露给用户

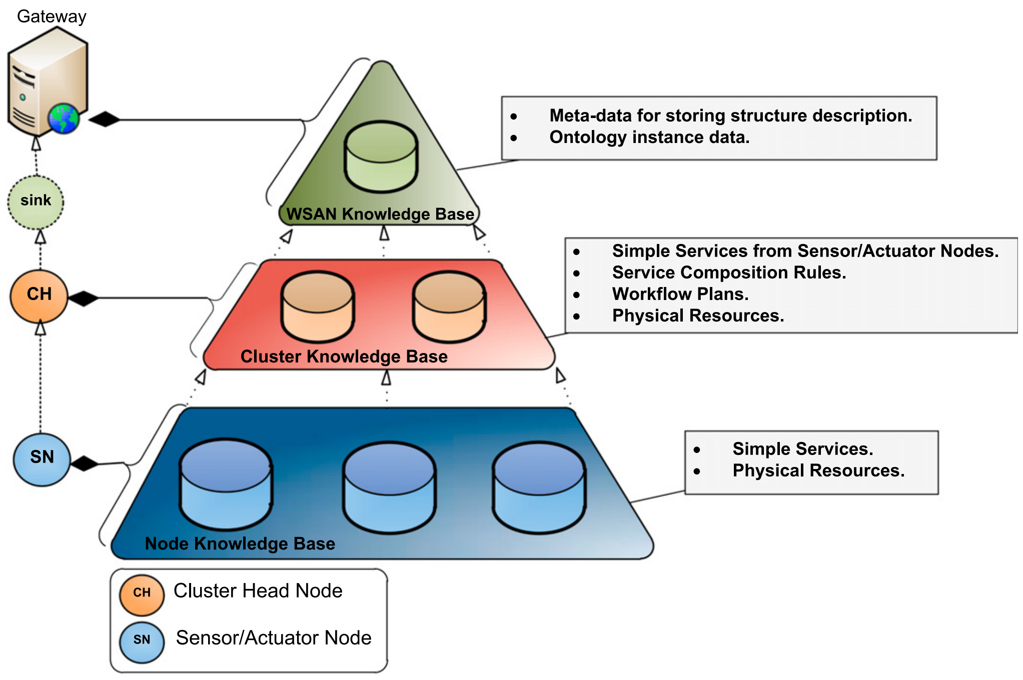


Fig. 3. Hierarchical scheme of Knowledge Bases mapped over the infrastructure.

are interested in high level information. However, simple services are occasionally required by user applications to perform more fine processing of contexts.

Mechanisms to carry out the service exposure and discovery are provided by the *Simple Inter-Knowledge Protocol* (SIKP), which enables an in-network communication interface for the middleware layer. SIKP's main functionalities will be explained in Section 5.

3.4. User applications

User applications can access to WSAN's services through the Gateway taking into account the description of those services exposed in WSDL documents. A WSDL document specifies *what the service does*, *how the service works*, and *how to invoke that service*. In short, a WSDL document provides information that a user application requires to invoke specific Web service by means of API contracts. There are several Web services tools, as Axis2 (The Apache Software Foundation, 2010), which consume WSDLs that describe RESTful Web services in order to generate clients and service stubs in diverse languages, publish a Web service or test a Web service dynamically.

User applications can perform actions in all levels of the KASOM architecture by invoking Web services. In Section 5.5, we show the Web service method encapsulation used by client applications. The method invocations range from a simple sensor monitoring values to complex configuration of a software module. Applications will have access: (i) at infrastructure level by managing, tuning or configuring sensor, actuators, nodes, etc. (e.g. modifying an actuator state to adapt the environment); (ii) at middleware or PRA level, to configure the in-node agents in order to provide an optimal performance or to control them (e.g. start/stop or load/unload them); (iii) at the contextual modeling level to dynamically adapt (e.g. augmenting or restricting) the Contextual Model; (iv) at the application level, to take advantage of other computing service that might be available within the same IP network or in other place of Internet.

In this section, the major foundation of our internetworking architecture for pervasive embedded networks was explained. In following sections, we describe the core of the KASOM which

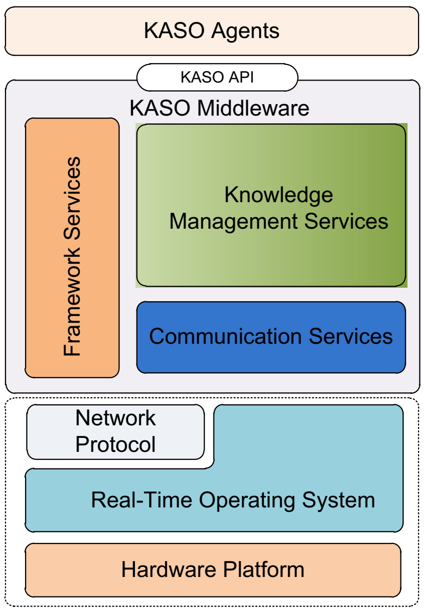


Fig. 4. In-node architecture proposed to contextualize KASOM.

provides in-network mechanisms to perform basic execution actions. In subsequent sections, specific services to provide Knowledge Management in KASOM are defined.

4. The KASO middleware architecture

4.1. Conceptual architecture

The in-network architecture approach of KASOM takes influence from the Service-Oriented Computing (SOC) models widely extend in the current Internet (M. Corporation; OMG; Fielding and Taylor, 2002) as well as the semantic models for ontology (Motik et al., 2009; R.C.W. Group). Figure 4 shows a conceptual diagram in which each one of the subsystems making the KASOM architecture are represented.

The proposed architecture is stratified in layers encapsulating different functionalities. Firstly, the common platform to the most of middleware approaches for sensor nodes is explained. The KASOM specific layer which provides the SOC features to the system is detailed later.

4.2. Generic platform

The lowest of these layers corresponds to the Hardware Platform. In this layer, all the hardware modules (e.g. memory, CPU, radio, analog and digital interfaces, etc.) are located which characterize the physical platform of the WSAN. Currently, a variety of products for sensor and actuator networks can be found, both commercial (e.g. Crossbow; SunSpot) and open-source (e.g. Arduino; Libelium).

Over the *Hardware Platform*, a Real-Time Operating System is implemented whose major aim is to hide the hardware heterogeneity to the higher layers. Software components in upper layers can use the underlying platform through OS primitives which abstract the hardware complexity. For instances, OS primitives could offer access to flash memory, management of peripheral interfaces (e.g. USART (Universal Synchronous Asynchronous Receiver Transmitter), ADC (Analog-to-Digital Converter), DAC (Digital-to-Analog Converter), etc.), CPU energy modes, radio signal strength, etc. There are several open-source OS for sensor and actuator networks such as TinyOS, Contiki OS, or LiteOS.

The *Network Protocol* layer implements routing protocols and algorithms in order to transmit optimally packets between different nodes of the network. Such routing mechanisms are implemented to be used in a multi-hop communication taking into account some essential factors: network topology, energy consumption, light footprint, and QoS requirements as timeliness or reliability. In the proposed model, the Network Protocol has to support a cluster topology and implements some mechanisms to save energy in every node of the network.

Together, the Network Protocol layer and Real-Time Operating System, provide a platform which facilities to the WSAN's node basic functionalities regarding communications and hardware management. From this generic platform, specific protocols and mechanisms are needed to create a system in which every node of the WSAN can establish effective collaborations between them. To achieve that aim, we propose to add a Knowledge-Aware and Service-Oriented Middleware (KASOM) to the architecture in order to generate environments in which distributed resources can collaboratively be served to multiple context-aware applications.

4.3. The middleware architecture

KASOM architecture consists of three subsystems (see Fig. 5).

The first subsystem provides *Framework Services* which allow creating an execution Framework which provides robust life-cycle to components and PRAs running on it. *Query Service* allows to PRAs and external applications to query information about specific network parameters. Such queries are thrown to the network in a dialect of SQL, which is interpreted in the destination to retrieve information. *Runtime Manager Service* is in charge of loading and unloading middleware components and PRAs. This task is usually carried out at node bootstrapping or shutdown,

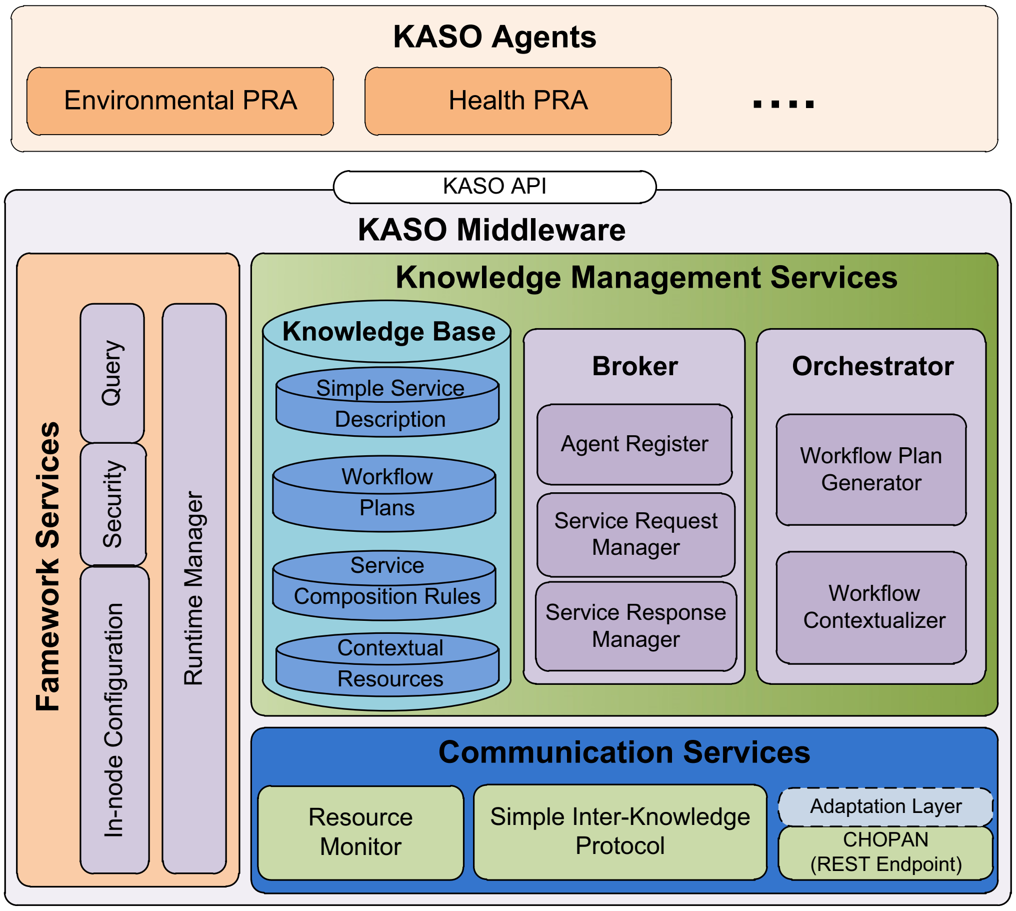


Fig. 5. Block diagram of the Knowledge-Aware and Service-Oriented Middleware (KASOM).

respectively. However, such processes can take place at any moment during runtime cycle. Moreover, Runtime Manager Service implements some concurrency mechanisms according to a *wait-by-necessity* principle, which only blocks a calling process if a return value is needed. *Security Service* provides procedures to manage various major security issues (e.g. permissions for accessing to PRA services). *In-node Configuration Service* allows setting up parameters such as routing priorities in network protocol, radio signal strength, allocation of resources, instantiations of components, security for communications, etc.

The second subsystem is composed by a set of *Communication Services*. Such services are implemented by means of several mechanisms and protocols. The *Resource Monitor* implements mechanisms to communicate with physical resources of the node, majorly sensors and actuators, but also batteries or radio module, by means of OS primitive callings. Optionally, more complex protocols can be added to communicate with sensors and actuator through Inter-Integrated Circuit (I^2C) or USART interfaces. As it is shown in Fig. 5, two applications protocols coexist in this subsystem: *Simple Inter-Knowledge Protocol* (SIKP) and *Compressed HTTP over PANs* (CHOPAN). SIKP is a lightweight application protocol which enables service discovery mechanisms between nodes belonging to the same WSAN. SIKP was designed from a reduced specification of WS-Discovery standard (OASIS, 2010). On the other hand, a REST Endpoint is implemented with the aim of offering the WSAN's services in a RESTful way to the Service Cloud. That REST Endpoint is offered by means of the CHOPAN protocol. CHOPAN is based on a binary specification of HTTP (Internet Engineering Task Force, 2010), which saves radio and energy resources in PANs. An adaptation layer has been designed over CHOPAN to translate REST method structures into the KASOM compatible format.

The major feature that has to characterize a pervasive embedded network is its capability to carry out common tasks by sharing resources that are placed in different nodes. In this respect, we have found a lack in related work (see Section 2) when managing resources and services between nodes in a pervasive embedded network. The most of them are device-centric approaches which were designed to manage resources of a single computing platform. In order to solve this key lack, we have added a novel subsystem to the middleware architecture consisting of *Knowledge Management Services* (KMS) which notably improves the cooperation between nodes in pervasive systems especially in those managing sensors and actuators as WSANs. KMS are provided by a *Broker* and an *Orchestrator*.

The Broker provides an API to developers of PRAs in order to register them into the system. Table 1 shows the Broker's API.

Using this API, developers of PRAs do not have to be concerned about underlying processes as resource allocation to assure the provision of simple services and contextual services. Parameters having to be provided to the Broker in order to generate a service are: *Service Description*, *Service Composition Rules*, and two callback functions to handle results from service requests and to manage partial steps of a service workflow.

The Orchestrator implements lightweight orchestration mechanisms to support a number of services based on *Service Composition Rules*. Such document establishes which resources are needed to be allocated (both physical and logical) as well how they have to be orchestrated in order to assure a correct provision of services. Orchestrator uses the Resources Monitor and the SIKP to explore for available resources both in-node and in-network domains, respectively.

In next section, major lines about developing PRAs to deploy services within KASOM are explained.

5. Deploying services in a KASOM-based pervasive environment

In this section the programming model for KASOM approach is analyzed. Outlines to develop Perceptual Reasoning Agents taking advance from services offered by KASOM are detailed as well as the tasks needed to generate and deploy simple and contextual services. Examples are explained showing the semantic pieces and business logic to be implemented by the developer.

5.1. Specifications of Perceptual Reasoning Agents

When developing Perceptual Reasoning Agents (PRAs) the developer has to keep in mind the two major features which characterize PRA: *independence* and *reusability*. PRA was designed following a lightweight agent philosophy, which facilitates a rapid design, development and deployment while optimizing the physical resources of nodes composing the pervasive embedded network. To develop agents using PRA methodology is as easy as developing plain Java objects (POJOs) what guarantees the Framework independence and future reusability. The only requirement when designing PRAs is to realize an interface defined in a UML's metamodel. Such interface allows to PRAs to be managed and scheduled seamlessly by the Framework Services and offer service handlers. The PRA's interface is illustrated below.

Framework Services manage PRAs by means of invocations to `load()`, `run()`, `stop()`, and `unload()` methods of every PRA running on KASOM. When Framework invokes `load()` method, PRAs instantiate semantic structures describing services which have to be provided to the rest of node within the same pervasive domain. Later, when invocating `run()` method, the PRA registers themselves in the system by a `registerPRA()` method calling of the Broker API which enables a service point between PRA and KASOM. That service point offers the PRA's business logic by means of two service handlers. During a controlled shutdown `stop()` and `unload()` methods are invocated by the Framework taking place the inverse process to the explained previously.

5.2. Registration of services and contextual resources

As it was explained in Section 4.3, the Broker API allows PRAs to register its services by callings to method `PRA_ID=registerPRA(Service_Description, Service_Composition_Rules, Service_Request_Handler, Workflow_Handler)`. In order to facilitated the PRA's service discovery to other entities of the system (e.g. other N-PRA or C-PRAs or user applications.) the developer has to specify a *service description document* (`Service_Description` parameter) according to

Table 1
Broker's API.

`PRA_ID=registerPRA(Service_Description, Service_Composition_Rules, Service_Request_Handler, Workflow_Handler)`	It registers a PRA. Its arguments are the description of services provided by such PRA, service composition rules, a callback function in which results of service requests will be got, and a callback function to handle partial workflow results.
`deregisterPRA(PRA_ID)`	It deregisters a PRA.
`updatePRA(PRA_ID, Service_Description, Composition_Rules)`	It updates information about a PRA register.

the Service Mapping Description (SMD) proposal (Proposal by anonymous author, 2010). SMD is a JSON representation based on various semantic rules to define aspects of Web service: *the SMD format was designed to be flexible*, *compact*, *simple*, *readable*, *and easily implemented* (Proposal by anonymous author, 2010). Such characteristics accomplishes completely with the light-weight implementation methodology of PRAs. Every SMD document collected by the Broker is stored in *Service Descriptions Table* of the Knowledge Base. The following list shows an example of simple service description (Listing 1).

We have extended the JSON scheme of SMD to support service composition requirements in resource-constraint devices by adding two new attributes: ``discoverable'' and ``composition''. ``Discoverable'' attribute controls the exposition of such service. If that parameter gets ``true'' value the specific service can be exposed and discovered to/by external entities using SIKP. ``Composition'' parameter is related to the permission for composing a simple service into a workflow.

Moreover, we have designed the *Sensor and Actuator Mapping Description* (SAMD) language to model semantics describing sensors and actuators deployed in pervasive embedded network. Using SAMD proposal it can be modeled parameters such sensor accuracy (``accuracy''), unit representation (``unit''), asynchronous behavior (``eventing'') or actuator actions (``actions''). An example of SAMD document is shown below (Listing 2).

During node's bootstrapping process contextual resources (majorly sensors and actuators) are discovered through

```
{
   "transport": "REST",
   "envelope": "PATH",
   "target": "environmentalPRA",
   "additionalParameters": true,
   "parameters": [
       {
           "name":"outputType",
           "default": "json"
       },
       {
           "name":"ignoreErrors",
           "optional": true
       }
   ],
   "services": {
       "getPolutionLevel": {
           "transport": "GET",
           "target": "Polution",
            "discoverable":"true",
           "composition":"false",
           "parameters": [
               {"name":"paramOne", "type": "string"},
               {"name":"paramTwo", "type": "integer", "default": 5},
               {"name":"paramThree", "type": "integer", "optional": true}
           ]
       },
       "getBatteryLevel": {
           "transport": "POST",
           "envelope": "JSON",
           "discoverable":"false"
           "composition":"false"
           "additionalParameters": {"type": "integer", "default": 0},
           "parameters": [
               {"name":"paramOne", "type": "integer", "default": 0},
               {"name":"paramTwo", "type": "integer", "default": 0}
           ]
       }
   }
 }
```

Listing 1. Simple service description based on SMD proposal.

```
        "sensors":{
               "temperature":{
                           "unit": {"Celsius","Farenheit"},
                           "accuracy": 0.5,
                           "eventing": {"byPeriod":true, "byThreshold":true}
               },
               "humidity":{
                           "unit": {"RelativeHumidity"},
                           "accuracy": 2,
                           "eventing": {"byPeriod":true, "byThreshold":true}
               }
        },
        "actuators":{
               "climatizer": {
                           "actions": {"cooling","heating","off"}
               }
        }
```

Listing 2. Contextual resource description document based on the SAMD proposal.

mechanisms performed by the Resource Monitor. During such process a SAMD document is instantiated with semantic information describing contextual resources. Finally this document is stored in the *Contextual Resources Table* of the Knowledge Base.

In following section, mechanisms to compose and orchestrate services in KASOM are explained. Such capacities are essential in order to get contextual services by dynamical and optimal allocation of resources (Fig. 6).

5.3. *Service composition and Orchestration*

The composition of services in KASOM is based on the Contextual Model described in Section 3.2. The basic information units of that model are provided by contextual resources (physical resources) from which simple services (logical resources) can be provided. That is the simplest service composition in KASOM. The composition tasks are performed by the Orchestrator through the Workflow Generator module (see Fig. 5). For instance, if a PRA

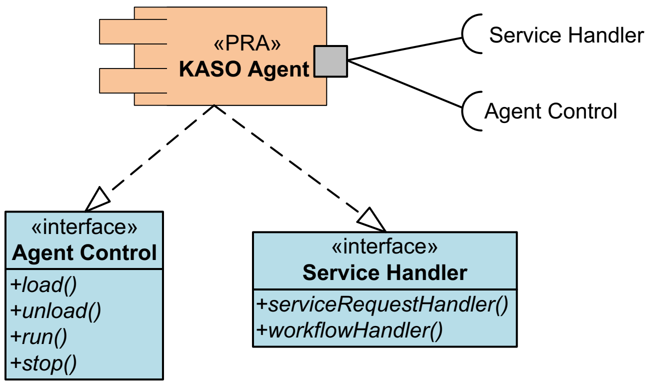


Fig. 6. Perceptual Reasoning Agent realizes both Agent Control and Service Handler interfaces.

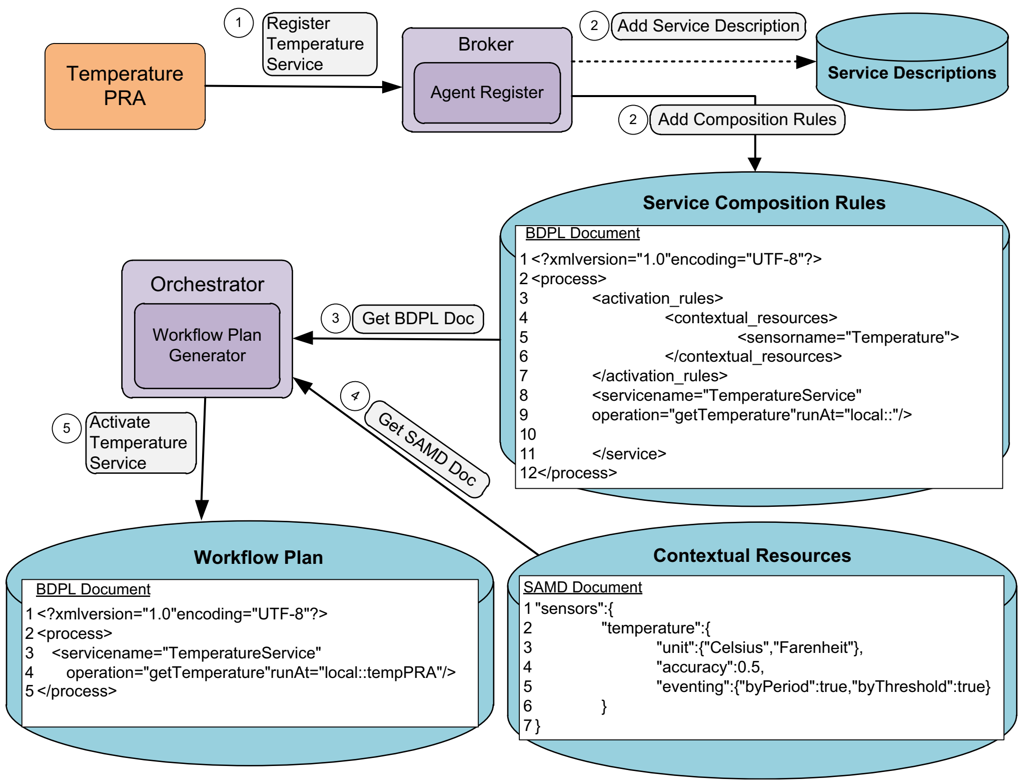
registers a simple service for providing temperature measurements and KASOM detects the presence of a temperature sensor, i.e. a contextual resource, being managed by the node, then `getTemperature` service can be enable by KASOM. Finally, this service is offered to SIKP which expose it to other entities.

When deploying services, PRA's developers have to be aware about the availability of physical and logical resources to assure the composition and orchestration of such services, i.e. concrete sensors or specific simple services, respectively. If necessary resources for the service provision are available, KASOM enables the service and allocates those resources to assure its provision. It is essential that PRA's developers design services keeping in mind the use of resources, majorly radio transmission and memory, in order to minimize the impact of the service management over the WSAN as well to increase its lifetime in energy terms.

A *service composition rules* document has to be defined and passed as argument to `registerPRA()` method (`Service_Composition_Rules` parameter) in order to prepare KASOM to carry out the resource allocation. Those service composition rules documents are stored in *Service Composition Rules Table* of the Knowledge Base.

Traditional languages for business process definition, as WS-BPEL (Crossbow Corporation Inc., 2010), need expensive parsing processes so that they are not optimized to be used in resource-constraint embedded devices. In this sense, we have designed a new XML-based language, so called *Pervasive Business Definition Language* (PBDL), with the aim of defining composition rules as well as service workflows both for simple and contextual services. PBDL allows to embedded networks, as WSANS, to perform complex orchestrations of services while saves memory and computational resources. In figure below, an example of simple service composition is shown.

As it can be seen in Fig. 7, the PBDL in Service Composition Rules Table contains two main tags: `<activation_rules>` and `<service>`. The first one describes contextual resources needed to activate the simple service. The second one describes



```
<?xmlversion="1.0"encoding="UTF-8"?>
<process>
        <activation_rules>
                <contextual_resources>
                        <sensorname="Temperature">
                </contextual_resources>
        </activation_rules>
        <servicename="TemperatureService"
        operation="getTemperature"runAt="local::"/>

        </service>
</process>
```

```
<?xmlversion="1.0"encoding="UTF-8"?>
<process>
  <servicename="TemperatureService"
    operation="getTemperature"runAt="local::tempPRA"/>
</process>
```

```
"sensors":{
        "temperature":{
                "unit":{"Celsius","Farenheit"},
                "accuracy":0.5,
                "eventing":{"byPeriod":true,"byThreshold":true}
        }
}
```

Fig. 7. Composition process for a simple service (`getTemperature`).

needed invocations to carry out a service request. Once the Workflow Plan Generator checks activation rules are satisfied according to the contextual resources available in the node, a new PBDL describing the workflow is instantiated and stored in *Workflow Plan Table*. The main divergences from the PBDL stored in *Service Composition Rules Table* is that `runAt` parameter in tag `<service>` is instantiated to a specific PRA (local or external). In addition, section `<activation_rules>` is removed.

Composition and orchestration of contextual services take place in CH nodes. This process starts when a C-PRA registers a contextual service into KASOM. Mechanisms for composing contextual services are similar to those used for composing simple service but, in this case, the basic units of composition are simple services instead of contextual resources. This specific characteristic is illustrated in the following section `<activation_rules>` of a PBDL (Listing 3).

The tag `<activation_rules>` specifies requirements for the activation of a contextual service, i.e. simple services (`<simple_services>`) which will have to be discovered by SIKP. Moreover, the workflow plan that is generated for a contextual service indicates the business process which will orchestrate invocations to those simple services. The following list shows a workflow plan for a hypothetical contextual service (`Set Comfortable Surrounding`), which would be activated by the previous activation rules (see Listings 3 and 4).

The previous lists illustrate a simple workflow defined in PBDL. Two of the most usual tags for business process definition are `<fork>` (parallel invocations) and `<if>` (conditional invocation). Other tag that can be used in PBDL is `<choose>` for alternative service invocations.

5.4. Service discovery

In the reference infrastructure shown in Fig. 2, WSAN's nodes and external devices implement different service discovery protocols and mechanisms offered by the WSAN. The first ones use the SIKP in order to explore service descriptions stored in Knowledge Base of sensor and actuator nodes around the WSAN. SIKP is based on an optimized version of WS-Discovery standard (OASIS, 2010) but is not compatible with traditional discovery protocols as DPWS (Driscoll and Mensch, 2009) or UPnP (Presser et al., 2008). Compatibility between SIKP and standardized discovery protocols can be carried out by means of a suitable mapping through the Gateway connecting the WSAN and a conventional LAN. However, this interaction is not envisioned in current design of the internetworking system.

From the previous scenario, sink nodes collect every service description from each cluster of nodes. In the Gateway, a Semantic Engine translates each one of the SMD document into

```
<activation_rules>

<simple_services>
   <service operation="getHumidity" runAt="externalNode::humidity_RPA "/>
   <service operation="getTemperature" runAt="externalNode::temperature_RPA"
   minNumber="2" />
   <service operation="getComfortLevel" runAt="localNode::confort_RPA"/>
   <service operation="setClimatizer" runAt="externalNode::climatizer_RPA">
            <parameter value="context.humidity"/>
            <parameter value="context.avgTemperature"/>
  </service>
  </simple_services>

</activation_rules>
```

Listing 3. The `<activation_rules>` section included in a PBDL for a contextual service.

```
<process name="Set_Comfortable_Surrounding">

   <service operation="getHumidity" runAt="nodeA::humidity_RPA"
   context="context" returnVar="context.humidity"/>

  <fork>
        <group>
            <service operation="getTemperature" runAt="nodeB::temperature
              RPA" context="context" returnVar="context.temperature1">
               <parameter value="context.pollingMode"/>
            </service>
            <service operation="getTemperature" runAt="nodeC::temperature
            RPA" context="context" returnVar="context.temperature2"/>
               <parameter value="context.pollingMode"/>
            </service>
        </group>
  </fork>

   <service operation="getConfortLevel"  context="context"
   returnVar="confortLevel" runAt="localNode::confortRPA" >
         <parameter value="context.avgTemperature"/>
         <parameter value="context.humidity"/>
    </service>

     <if test="{context.currentComfortLevel < context.mediumLevel}">
        <service operation="setClimatizer" runAt="nodeD::climatizer RPA"
        context="context" returnVar="context.climatizerStatus">
            <parameter value="context.humidityLevel"/>
            <parameter value="context.temperatureLevel"/>
        </service>

     </if>

</process>
```

Listing 4. PBDL-based workflow plan for a contextual service orchestration.

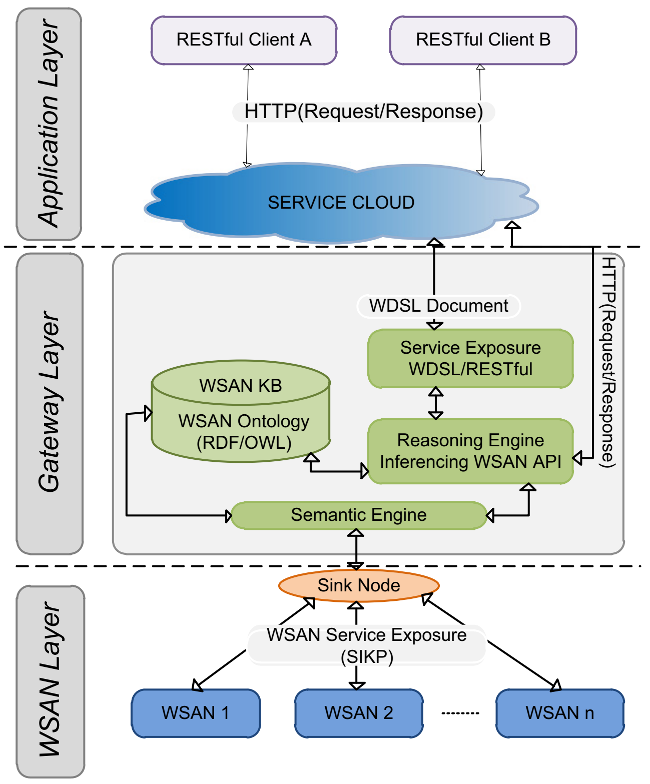


Fig. 8. The internetworking approach based on a synergy between KASOM and RESTful technologies.

a RDF model which will be aligned with the WSAN's ontology structured in OWL. The WSAN's ontology will evolve according to the reception of SMDs. This ontology will be stored in the Knowledge Base sited in the Gateway. The following figure shows the mapping between the physical and logical domains of the scenario explained previously (Fig. 8).

A RESTful (Fielding, 2000) architecture is deployed externally to the WSAN in order to expose and manage the WSAN's services as Web services. WSAN's services are exposed in a WSDL 2.0 document which describes the WSAN's services as REST resources. We have chosen the WSDL 2.0 standard to describe WSAN's services because it offers better support for RESTful Web services and it is simpler to implement than 1.1 version. Such WSDL 2.0 document is automatically generated by means of a parsing process from an instance of the WSAN's ontology. Finally, this WSDL is stored in a public UDDI registry from which external clients can consult service interfaces of the WSAN.

5.5. *Accessing to the WSAN's contextual services*

Once the discovery mechanisms are completed, in-network agents and external user applications could invoke services and receive results or error back. Following a RESTful architecture, WSAN's contextual services can be accessed through HTTP methods (GET, PUT, POST, and DELETE).

External applications are REST clients, which obtain interface description of the WSAN's services by consulting WSDL 2.0 documents. User's requests for WSAN's services are received by the Gateway which translates those requests (encoded in XML format) into a JSON format in order to be interpretable for Service Request Manager of KASOM. Such structure will be transported over a CHOPAN PDU to the specific node which has to provide results of the requested service. An example of a service transaction is shown in Fig. 9.

The Orchestrator is in charge of solving concurrency situations when clients compete for shared resources. The Orchestrator is able to analyze which contextual resources or simple services are involved in each business processes in order to optimize its execution regarding response time and throughput. Those mechanisms provide transparent concurrency which hides the complexity of resource sharing to the developer while guarantees the system robustness. For event-based services, concurrency is solved in a simpler way. A subscription table is implemented in the Service Response Manager, which stores subscriptions to event made by event consumers. If event trigger conditions are complied then one or more events are thrown to event consumers indicated in the subscription table. Event subscription criteria can be established by a period of time or a threshold.

6. Scenario for KASO middleware validation

This section describes the major features of an application scenario of healthcare telemonitoring, which was used to validate the KASOM architecture. The motivating scenario was deployed in the Sanatorium Versmè in Birštonas, Lithuania. This scenario was chosen as an ideal environment in order to deploy a KASOM-based system over a real WSAN and to verify the capacities to support a variety of simple and contextual services.

6.1. *System infrastructure*

The integral healthcare telemonitoring system was designed according to three major objectives: (i) surveillance of the Sanatorium perimeter, (ii) tracking of patient and staff, and (iii) critical vital signs monitoring. The scenario deployed is shown in Fig. 10.

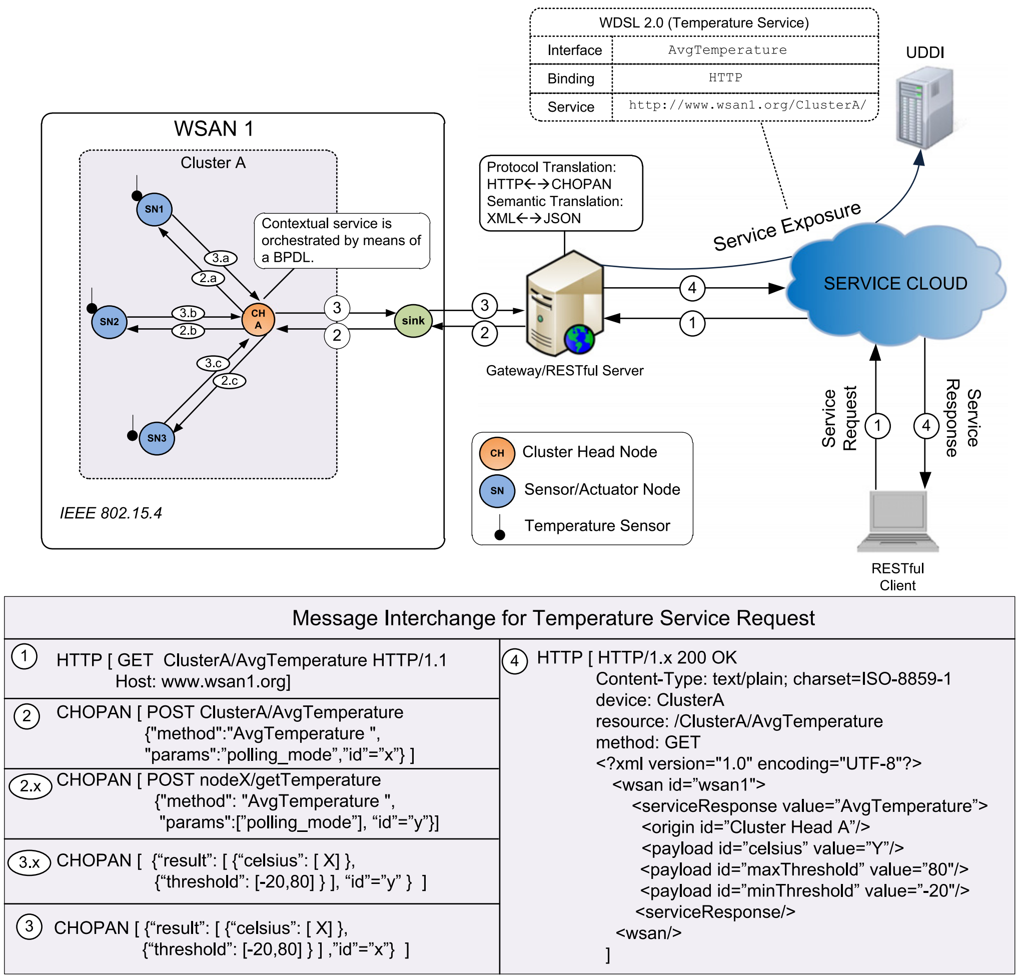
The Sanatorium area was covered by a WSAN whose nodes had different roles within the network. The node platform composing the WSAN was the Crossbow TelosB (Crossbow Corporation Inc., 2010). The major specifications of TelosB are wireless interface based on IEEE 802.15.4 (250 kbps), 48 kb of Flash memory, 16 kb of configuration EEPROM, and 10 kb of RAM. It supplies by means of 2xAA batteries. Also, a Smart Sensor Board (SSB) was used in order to connect a variety of different sensor and actuator over the nodes. This is a very resource-constrained platform compared with latest platforms as Imote2 (Crossbow Corporation Inc., 2010) or SunSpot. Taking into account such restrictions, performance and reliability testing performed with this type of nodes are decisive to check the feasibility of KASOM in reduced function devices (Fig. 11).

The deployed WSAN was composed of five kinds of nodes. Each one plays a specific role within the network. Characteristics of each role in the WSAN, are explained below.

Virtual perimeter node: the Sanatorium is surrounded by virtual perimeter nodes. Those nodes are provisioned with GSN PATROL-701 Passive Infrared Sensors (PIR) (GSN Corporation Inc., 2010). Surveillance segments are created between two virtual perimeter nodes. A Surveillance PRA is running on those nodes providing presence event service (Fig. 12).

Moreover, Surveillance PRA is able to identify patient and Sanatorium staff that carries a bracelet node. Part of Surveillance PRA is contained in bracelet node, which allows sending a personal and unique Id to the WSAN when a presence event is announced to the bracelet node.

Bracelet node: patients of the Sanatorium Versmè carried a bracelet consisting of a node equipped with some tiny biomedical sensors: heart-monitor, blood pressure, or body temperature, depending on their illness. The bracelet node was intended to be carried by the patient who wishes walk around the Sanatorium while preserving his/her privacy. Carrying this bracelet, vital signs of the patients can be continuously monitoring by medical staff.



Message Interchange for Temperature Service Request

```
(1) HTTP [ GET  ClusterA/AvgTemperature HTTP/1.1
          Host: www.wsan1.org]

(2) CHOPAN [ POST ClusterA/AvgTemperature
                {"method":"AvgTemperature ",
                "params":"polling_mode","id"="x"} ]

(2.x) CHOPAN [ POST nodeX/getTemperature
                {"method": "AvgTemperature ",
                 "params":["polling_mode"], "id"="y"}]

(3.x) CHOPAN [  {"result": [ {"celsius": [ X] },
                {"threshold": [-20,80] } ], "id"="y" }  ]

(3) CHOPAN [ {"result": [ {"celsius": [ X] },
             {"threshold": [-20,80] } ] ,"id"="x"}  ]

(4) HTTP [ HTTP/1.x 200 OK
          Content-Type: text/plain; charset=ISO-8859-1
          device: ClusterA
          resource: /ClusterA/AvgTemperature
          method: GET
          <?xml version="1.0" encoding="UTF-8"?>
            <wsan id="wsan1">
              <serviceResponse value="AvgTemperature">
                <origin id="Cluster Head A"/>
                <payload id="celsius" value="Y"/>
                <payload id="maxThreshold" value="80"/>
                <payload id="minThreshold" value="-20"/>
              <serviceResponse/>
            <wsan/>
          ]
```

Fig. 9. An example of temperature service request showing the interaction between the whole infrastructure as well as the involved message interchange.

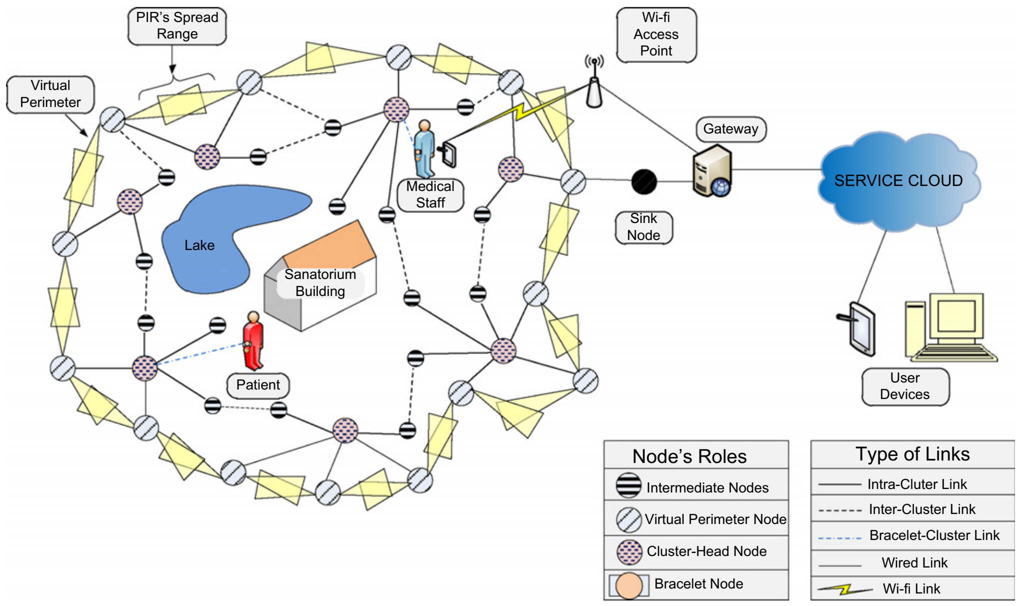


Fig. 10. Diagram of the scenario deployed for validating KASOM architecture, in Sanatorium Versmè in Birštonas, Lithuania.

This node is the only mobile one in the WSAN. According to the area in which bracelet node is moving, it synchronizes with clusters by exposing its services.

A *critical monitoring* PRA is in charge of collecting information from biomedical sensors in order to provide vital sign telemonitoring service. This agent is able to send patient's health reports

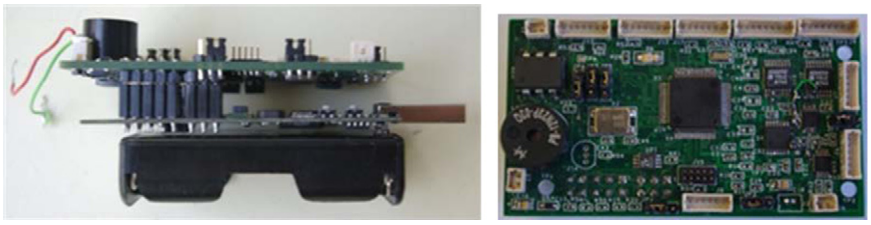


Fig. 11. TelosB mote (left side) and Smart Sensor Board (right side) used in the deployment.

Fig. 12. Virtual perimeter node deployed in the real scenario.

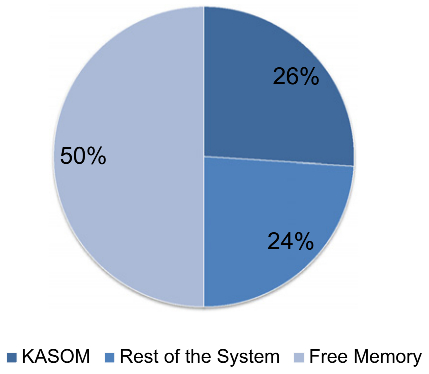


Fig. 13. KASOM footprint in ROM.

periodically. Health reports can be sent by two ways: on-demand or event-based.

Environmental monitoring node: every virtual perimeter and intermediate node around the WSAN integrates several automation sensors (e.g. temperature, humidity, light sensors, etc.). These sensors allow monitoring the most important environmental parameters in order to decide the convenience for patient to get out of the Sanatorium building to take a walk.

The environmental monitoring tasks are performed by an *ambient monitoring* PRA running on both virtual perimeter and other intermediate nodes. This PRA offers several simple services which are taken by the CH node to compose contextual services consisting of computing environmental parameters in order to get some important values of a certain sector of the Sanatorium (e.g. average temperature, max/min humidity value, etc.).

Cluster head node: the role played by CH nodes is just to provide contextual services from simple services offered by the nodes in their neighborhood in order to obtain aggregated values of environmental parameters.

Sink Node and Gateway: Sink nodes are in charge of collecting information generated in the WSAN. All this collected information is transmitted to the Gateway by USB connection. The Gateway collected SMD documents disseminated around the WSAN in order to instantiate the WSAN's ontology. From that ontology a WSDL 2.0 which described the WSAN's services like REST Web services is generated.

6.2. Validation results

The healthcare scenario proposed in previous section improved notably the security and safety of the users of the Sanatorium Versmè: patients, staff, and visitors. A test suits were designed taking as reference that scenario. The test cases performed over the deployed system founded on KASOM were carried out during three weeks between August and September, 2009. The WSAN deployed in Sanatorium Versmè was made up of 106 SA nodes and 21 CH nodes. These test cases involved to 20 patients and 5 medical staff who helped performing a number of valuable tests about memory footprint, performance, reliability, and efficiency of a KASOM-based system.

6.2.1. Memory footprint

Two measurements were performed in order to calculate the KASOM's footprint. The first one was focused on checking how much ROM was consumed by a full implementation of KASOM in a TelosB. This measurement is shown in Fig. 13.

The previous graphic shows as KASOM's footprint in ROM is 50% of the total ROM (48 kb). The rest of the system, basically the network protocol (6LowPAN (IETF, 2010)) and Operating System (TinyOS), required 24% of the ROM. Therefore, the developer has 26% of the ROM (more than 12 kb) to deploy PRAs. From our experience, we estimate that memory sufficient for deploy 4 C-PRAs or 6 N-PRAs providing contextual and simple services, respectively.

Also, we analyzed the footprint of a Knowledge Base during run-time. We take as reference a demanding environment, i.e. a node running three C-PRAs. The results of footprint impact over RAM are shown in Fig. 14.

The used RAM (38%) is negligible if we consider the total available in TelosB. This fact assures a good performance of the node since internal buffers for network protocol has much available memory.

6.2.2. Performance and reliability

These tests consisted of testing the performance and reliability of the system. Two kinds of services were deployed according to service dispatching mechanisms supported by KASOM: event-based and on-demand services (Tables 2 and 3).

Reliability and performance tests were performed when dealing with event-based and on-demand services. To this aim four test cases were planned consisting of different number of event-based and on-demand service requests from user devices: 10, 15, 20, and 25 for each kind of service. The services were requested to clusters homogenously disseminated so that service data flows

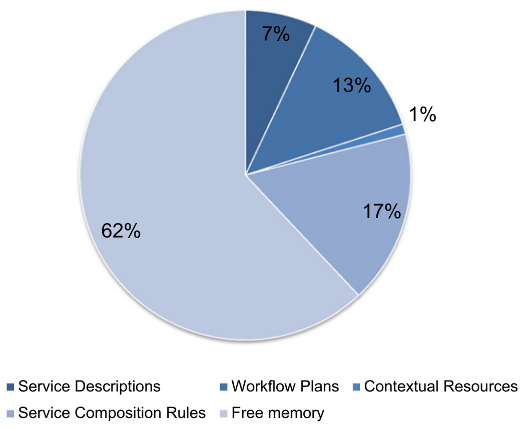


Fig. 14. Knowledge Base footprint in RAM.

Table 2
Event-based services designed for the validation scenario.

Service name	Service type	Returned event
Environmental control	Contextual service	Temperature exceeded in a region of the Sanatorium
Perimeter intrusion	Simple service	No authorized person crossed the virtual perimeter
Health condition	Simple service	Abnormal health condition of a patient
Technical control	Simple service	Low batteries

Table 3
On-demand services designed for the validation scenario.

Service name	Service type	Returned result
Environmental monitoring	Contextual service	Average humidity level in a region of the Sanatorium
Health monitoring	Contextual service	Health measurements: blood glucose, body temperature, and hearth rate
Technical supervision	Simple service	Battery level

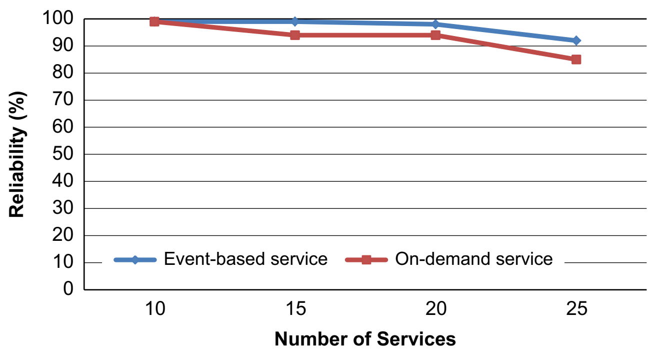


Fig. 15. Reliability test case: reliability when dispatching services.

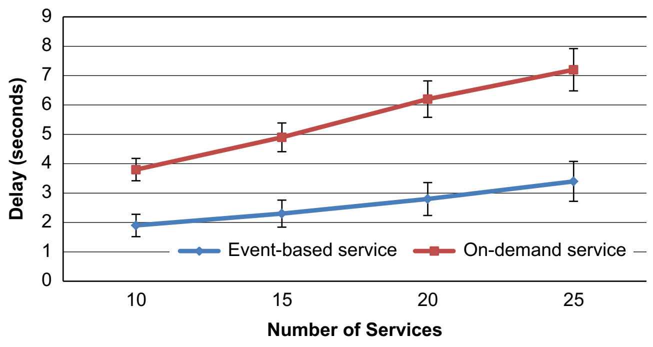


Fig. 16. Performance test case: average delay from service source to sink node.

were distributed fairly between nodes of the WSAN. A multi-hop communication was used between service providers and clients; it was needed between 1 and 8 hops to tranmit packets from the origin to the destination. Results of test cases are shown in Figs. 15 and 16.

The first graphic shows the reliability level reached by the system. In case of on-demand services, system reliability is calculated according to the correctly send and received service information. If event-based services, calculations are performed over the received events (let consider a scenario in which event-based service request have been correctly dispatched). As it can be seen, when event-based services are handled, reliability level decreases slightly (98%) when the number of involved services increases from 25. On the other hand, when dealing with on-demand service the reliability is 96% from 15 dispatched services. There are several factors which affect to the global reliability of the system in a WSAN. Firstly, the more are the hops that packets have to perform in order to reach a node (an intermediate or sink node), the less the reliability of the message reception. This is due to the wireless links connecting nodes becomes unstable and unreliable in certain situations (e.g. reflective objects in the scenario which weaken the radio signal or excess traffic which produces an overflow in Tx/Rx buffers). This factor is especially critical in links between sink nodes and the rest of the WSAN, because it can become a bottleneck. In this sense, we concluded that to use *one-way* data flow paradigm is more advisable than use request-reply paradigm; in our approach those two paradigms correspond with event-based and on-demand services, respectively. Therefore, if event-based services are used, traffic is reduced notably while the global reliability of the WSAN is augmented.

In the second graphic, delay when dispatching both event-based and on-demand services is shown. In that graphic, we report the average and the standard deviation of the results. If event-based services are managed, the service data traffic transmitted through the WSAN can be almost halved. This allows decreasing delay, jitter and other no desiderable factors affecting to the QoS of the service transactions, specially when using so bandwidth-constrained devices as TelosB mote (250 kbps). Results show as deviation is higher when service invokations are increased. That is because MAC protocol of IEEE 802.15.4 has to activate contention mecanisms based on CSMA/CA to be able to transmit all frames which travel through the WSAN.

From the previous results, we concluded that on-demand services must be only used in special situacions (e.g. an emergency) and their use must not be excesive.

6.2.3. System efficiency

The efficiency of the healthcare telemonitoring system was also analyzed. We checked that the KASOM-based system improved the response time when occurring an alert because of a patient health emergency or a perimeter crossing. The response time of the sanatorium staff, when managing an emergency situation, was reduced by 36% on average. Threfore, after deploying this system, only 6 min of elapsed time was spent to attend an emergency.

Table 4
A comparative of efficiency without and with telemonitoring healthcare service.

	False alarms per day	Emergency response time (average) (min)
Without telemonitoring healthcare system	20	9.4
With telemonitoring healthcare system	16	6

We also observed that the false positives was reduced 20% decreasing the number of assistances from 20 to 16 (Table 4).

7. Conclusions and future works

The Knowledge-Aware and Service-Oriented Middleware (KASOM) was described as a novel approach to integrate pervasive embedded networks into the Service Cloud which comprise the foundation of the new Internet of Things paradigm. KASOM consist of three major subsystems: *Framework Services*, *Communication Services*, and *Knowledge Management Services*. Such services allow addressing a Service-Oriented Architecture for pervasive environments by means of four axis: *registration*, *discovery*, *composition*, and *orchestration* of services. These caracteristics provides to developer several tools which facilitates the design and deployed of services in pervasive network.

The results from validation tests which was carried out over a healthcare scenario were very promising in terms of QoS and human resources saving. A good reliability-delay balance was also reached allowing increase the efficiency of the medical staff to face up an emergency in terms of assistant delay and false alarms.

Our future researches will focus on improving the dynamicity of KASOM in two ways. Firstly, we will redesign the Workflow Plan Generator of the Orchestrator in order to augment its dynamic capacities. The main idea behind this aim is to design a reasoning module based on genetic algorithms which takes advance from service redundancy to generate alternative workflows which allow overcoming inconsistent situations or save resources.

Secondly, we propose to improve the reusability of network deployments, independently of its initial purpose. In this sense, we propose to design an Over-the-Air (OTA) reprogramming which allows transferring executable code through the WSAN and to load PRAs on specific nodes. This challenge will require using reliable transport protocols to send bytes in multi-hop communication as well to introduce new in-node mechanisms to reach a robust node reboot with new PRAs.

Acknowledgements

This work has been supported in part by the European Commission under the IST-034642, μSWN: Solving Major Problems in Micro Sensorial Networks research project within the Sixth Framework Program (FP6) (http://www.uswn.eu); and under the ITEA 2-08005 DiYSE: Do-it-Yourself Smart Experiences (http://www.diyse.org).

References

Arduino. 〈http://www.arduino.cc/〉 [last accessed on June 2010].
Chih-Min C, Hsiao T-Y. Design of structure-free and energy-balanced data aggregation in wireless sensor networks. In: Proceedings of 11th IEEE international conference on high performance computing and communications. HPCC '09; 2009.
Chinnici R, Moreau J, Ryman A, Weerawarana. S. W3C recommendation, 〈http://www.w3.org/TR/wsdl20/〉; June 2007 [online].
Contiki OS. 〈http://www.sics.se/contiki/〉 [last accessed on June 2010].
Cougar Project. 〈www.cs.cornell.edu/database/cougar〉 [last accessed on June 2010].
Crossbow. 〈http://www.xbow.com〉 [last accessed on June 2010].
Crossbow Corporation Inc.. TelosB specification, 〈http://www.xbow.com/Products/productdetails.aspx?sid=252〉; 2010 [online].
Crossbow Corporation Inc.Imote2 specification, 〈http://www.xbow.com/Products/productdetails.aspx?sid=253〉; 2010 [online].
Driscoll D, Mensch A. Devices profile for web services (DPWS) version 1.1. OASIS. 2009.
Feng X., Xu Z. A neural data fusion algorithm in wireless sensor network. In: Proceedings of Pacific-Asia conference on circuits, communications and systems. PACCS '09; 2009. p. 54–7.
Fielding R, Taylor R. Principled design of the modern. ACM Transactions on Internet Technology 2002:115–50.
Fielding. RT. Chapter 5 of fielding's dissertation, 〈http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm〉; 2000 [online].
Gomez L, Laube L, Sorniotty A, Wrona K. Secure and trusted in-network data processing in wireless sensor networks. Journal of Infomration Assurance and Security 2007.
GSN Corporation Inc.PATROL-701 PIR device, 〈http://www.gsncompany.com〉; 2010 [online].
Heinzelman W, Kulik J, Balakrishnan B. Adaptive protocols for information dissemination in wireless sensor networks. In: Proceedings of fifth ACM/IEEE mobicom conference (MobiCom '99). Seattle (WA); 1999. p. 174–85.
Heinzelman W, Murphy A, Carvalho H, Perillo M. Middleware to support sensor network applications. IEEE Network Magazine 2004. [special issue, January].
IETF IPv6 over low power WPAN (6lowpan), 〈http://datatracker.ietf.org/wg/6lowpan/charter/〉; June 2010 [online].
Intanagonwiwat C. Directed diffusion: a scalable and robust communication paradigm for sensor network. In: Proceedings of sixth ACM/IEEE mobicom conference (MobiCom '00). Boston (MA); 2000.
Internet Engineering Task Force. Chopan—compressed HTTP over PANs. 〈http://tools.ietf.org/html/draft-frank-6lowpan-chopan-00〉; June 2010 [online].
Kulik J, Heinzelman WR, Blakrishnan H. Negotiation-based protocols for disseminating information in wireless sensor networks. Wireless Networks 2002:169–85.
Lionel M, et al. Semantic sensor net: an extensible framework. International Journal of Ad Hoc and Ubiquitous Computing 2009:157–67.
Libelium. 〈http://www.libelium.com/〉 [last accessed on June 2010].
LiteOS. 〈http://www.liteos.net/〉 [last accessed on June 2010].
Mattern F. The design space of wireless sensor networks. IEEE Wireless Communications 2004;3(11):54–61. [December].
M. Corporation. DCOM technical overview, 〈http://technet.microsoft.com/en-us/library/cc722925.aspx〉 [last accessed on June 2010].
Motik B et al. OWL 2 web ontology language: structural specification and functional-style syntax. W3C recommendation. 〈http://www.w3.org/TR/2009/REC-owl2-syntax-20091027/〉; 27.10.2009.
Mudasser I, Hock BL, Wenqiang W., Yuxia Y.A service-oriented model for semantics-based data management in wireless sensor networks. In: Proceedings of international conference on advanced information networking and applications workshops. Bradford, United Kingdom; 2009. p. 395–400.
OASIS. Web services dynamic discovery (WS-discovery) version 1.1. 〈http://docs.oasis-open.org/ws-dd/discovery/1.1/os/wsdd-discovery-1.1-spec-os.pdf〉; June 2010 [online].
OMG. Catalog of OMG CORBA®/IIOP® specifications, 〈http://www.omg.org/technology/documents/spec_catalog.htm〉 [last accessed on June 2010].
Presser A, et al. UPnP^{TM} device architecture 1.1. UPnP Forum Specification 2008.
Proposal by anonymous author. Service mapping description proposal, 〈http://groups.google.com/group/json-schema/web/service-mapping-description-proposal〉; June 2010 [online].
Ranganathan A, et al. Use of ontologies in a pervasive computing environment, vol. 18. Cambridge University Press; 2003. p. 209–20.
Römer K, Kasten C, Mattern F. Middleware challenges for wireless sensor networks. ACM Mobile Computing and Communication Review (MC2R) 2002;6(4):59–61. [October].
R.C.W. Group. Resource description framework (RDF), 〈http://www.w3.org/RDF/#specs〉 [last accessed on June 2010].
RUNES Consortium. IST-004536-RUNES, 〈www.ist-runes.org〉 [last accessed on June 2010].
Srisathapornphat C, Jaikaeo C, Shen C. Sensor information networking architecture. In: Proceedings of international workshop on parallel processing; September; 2000. p. 23–30.
SunSpot. 〈http://www.sunspotworld.com/〉 [last accessed on June 2010].
Taherkordi A, Le-Trung Q, Rouvoy R, Eliassen F. WiSeKit: a distributed middleware to support application-level adaptation in sensor networks. In: Proceedings of ninth IFIP international conference on distributed applications and interoperable systems; 2009. p. 44–58.
The Apache Software Foundation. Apache axis2. Version 1.5.1, 〈http://ws.apache.org/axis2/〉; June 2010 [online].
TinyOS. 〈http://www.tinyos.net/〉 [last accessed on June 2010].
World Health Organization. Global age-friendly cities; 2007.
Weiser M. Hot topics: ubiquitous computing. IEEE Computer 1993.