

Semantic Web Based Architecture for Managing Hardware Heterogeneity in Wireless Sensor Network

Siniša Nikolić
Faculty of Technical
Sciences,
University of Novi Sad,
Novi Sad, Serbia

Valentin Penca
Faculty of Technical
Sciences,
University of Novi Sad,
Novi Sad, Serbia

Milan Segedinac
Faculty of Technical
Sciences,
University of Novi Sad,
Novi Sad, Serbia

Zora Konjović
Faculty of Technical
Sciences,
University of Novi Sad,
Novi Sad, Serbia

siniša_nikolic@uns.ac.rs

valentin_penca@uns.ac.rs

milansegedinac@uns.ac.rs

ftn_zora@uns.ac.rs

ABSTRACT

Wireless Sensor Networks (WSNs) are vivid and relatively new paradigm. Usually they are built for the specific purposes which, in the case of large systems of WSNs, can result in high hardware heterogeneity. In this paper we have proposed an ontology-based model aimed at resolving the hardware heterogeneity in large systems of WSNs, and the GLOSENT architecture that facilitates integration of heterogeneous WSNs. In addition, as a case study the concrete SN device SunSPOT is represented using proposed ontology.

Categories and Subject Descriptors

C.2.1 [Computer Communication Networks]: Network Architecture and Design - *wireless communication*.

I.2.4 [Computing Methodologies]: Knowledge Representation Formalisms and Methods - *Representation languages*.

General Terms

Design, Standardization, Languages

Keywords

Wireless Sensor Networks, Heterogeneity, Ontology, Middleware, SunSPOT.

1. INTRODUCTION

WSN is a network of spatially distributed autonomous sensors and it can include hundreds of thousands sensor nodes (SN). SNs are low cost, battery-powered and reduced size devices with limited processing, sensing and wireless communication capabilities aimed at completing specific application task.

Most of the current WSNs are built for specific purposes, with a tight coupling between the physical network components and user application. On the other hand, in real-world applications WSN must not be considered an isolated entity, but rather a part of a larger System of Wireless Sensor Networks (SWSN) in which

user applications interact with heterogeneous WSNs.

Heterogeneity of WSN appears both in hardware and software. Heterogeneity, which appears in the underlying hardware, changeable acquisition protocols, changeable user application requirements, scalability and prolonging life of WSNs are the main problems that SWSNs face. The existing solutions typically focus single aspect of the problem [24].

In this paper we propose a way to deal with heterogeneity in SWSN which is based on Semantic Web approach. We suggest a specific middleware that provides communication for other parts of the SWSN based on the exchange of ontologies. Here, ontologies are used for the semantic representation of WSN components and sense data.

The focus of the paper is hardware heterogeneity of WSN components.

The remaining part of the paper is organized as follows: Section 2 presents a brief overview of the related work. The proposed architecture is presented in Section 3. Section 4 describes WSN ontology. Section 5 is the case study describing the ontology for the particular SN. Section 6 concludes the paper and indicates some future work.

2. RELATED WORK

There are two main approaches aimed at linking network and applications in WSNs. In the first approach (*specific based*) [6] the whole system relies on the immutability and the specificity of application requirements and the environment in which WSN is implemented. In such type of systems even small modification in application requirements or in the structure of WSN, result in changes that typically need to be implemented all across the system. This approach sacrifices generality for simplicity of implementation. The second approach (*generic based*) includes the middle layer (*middleware*) [24]. Middleware is defined as software which is located in between software applications. Considering that devices forming a WSN have only little capabilities in terms of processing power and memory (hardware characteristic), their primitive operating systems only provide basic support for integration with other software components. Middleware compensates that deficiency, providing for abstraction and additional services. With middleware, applications implement their functionality in a more abstract manner. In [5], [12], [13] the classifications of middleware approaches for WSN are described. The one proposed in [12]

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

WIMS'11, May 25–27, 2011, Sogndal, Norway.

Copyright © 2011 ACM 978-1-4503-0148-0/11/05... \$10.00

identifies the following main middleware classes: *database approach*, *event-based approach*, *application driven approach*, *modular approach* and *virtual machine approach*. The rest of this section presents a brief overview of several characteristic implementations.

In *TinyDB* [11] the network established by the sensor nodes is understood as a distributed database which can be queried using a subset of SQL. Thus, to obtain a sensor data from the network, the user issues a query, which is then automatically routed to all nodes and processed by node's query processor. Here, heterogeneity of SN hardware may appear as a problem, because *TinyDB* resides as an application on the top of *TinyOS* [27] operating system at SN, and many nodes don't support *TinyOS*, so the heterogeneity is partial. Also, *TinyDB* does not allow to sense and react upon certain event.

Mate [8] and *Squawk* [19] WSN middleware use a Virtual Machine (VM) approach where *Mate* is implemented on the top of *TinyOS* and *Squawk* is JAVA based VM. Using VM approach to sensor nodes presumes that applications have to be written in specific language. In most cases VM are built for restricted set of SN hardware platforms. For application changes both solutions propose a spectrum of reprogram actions, from adjusting simple parameters to uploading complete program updates using a VM. Uploading operation causes large energy consumption, which makes heterogeneity problem in this middleware even more chargeable.

Impala [9], which uses event-based programming model, provides mechanisms for network updates that are efficient enough to support dynamic applications. This modular approach uses mobile code that is injected in the sensor network providing for collecting local data and migrating and copying itself to other nodes. The problem is the nature of its predefined instruction set that doesn't allow hardware heterogeneity.

MiLAN [14], which belongs to adaptive middleware, receives a description of application requirements, monitors network conditions and optimizes sensor and network configurations to maximize network lifetime. This application driven approach introduces a new dimension in middleware design by supplementing an architecture that tightly couples application execution with network protocol stack. Fine tuning of the network suggests close relation with application and may lead to specialized middleware. The communication is restricted only to the hardware supporting that specific network protocol stack which is a serious drawback regarding hardware heterogeneity.

Mires [21] is an event-based, message oriented distributed middleware that implements the publish/subscribe paradigm in WSN. It uses strong asynchronous communication model where consumers and producers are loosely coupled. In that system SNs advertise what data they can provide, while applications subscribe to data. *Mires* middleware is built upon *TinyOS*, thus sharing the same hardware heterogeneity problems.

In [18] the implementation of WSN based on Service Oriented Architecture (SOA) model and SWE standards [26] is described. Architecture of that system is split in 3 clearly distinguished layers (information production, information routing, information consumption) consisting of two SN types and a Sync node (a gateway). Simplified SN integrates services to retrieve data from

the network, while advanced SN integrates services for filtering, aggregation and routing of the data. Service architecture is modeled only for WSN, user application is considered only as a consumer of the top level layer services. The hardware heterogeneity problem is resolved by XML descriptions in accordance with SWE standard. There is no shared knowledge about services and the semantic web technologies are not utilized.

The *OX-Framework* [2], which relies on OGC standards [26] for presenting geospatial data, is an example of Web technologies utilization in WSNs. The OGC's Sensor Web Enablement (SWE) standards enable developers to make all types of sensors, transducers and sensor data repositories discoverable, accessible and useable via Web. Purpose of the framework is integration of real time or near real-time measurements with conventional geospatial data (e.g. maps) for visualization purposes. *OX-Framework* relies on service approach, where entities of WSN communicate by invoking different services of the system. The framework provides discoverable data repositories that include hardware descriptions, but does not include semantic descriptions and does not use semantic technologies.

One example of SWSN where the middleware layer uses Semantic Web technologies is presented in [10]. Here, middleware deals only with semantic data integration and does not consider at all any other type of heterogeneity. The sensor data are used to reconstruct (by reasoning process) the context of an event. The core of aforementioned solution consists of the hierarchical organization of semantic information and semantic services. Other WSN relevant issues (e.g. data streaming pattern, network elements description, description of network structure, etc.) are not described by ontologies. The system architecture is a three-tier one, consisting of sensors, field servers and gateway servers. Each field server, in the middle layer is directly responsible for a particular group of sensors. The collected measurements are converted into XML format at servers and the XML data is passed to other levels of the system.

In [1] the ontologies are used for providing adaptive WSN, i.e. optimal sensor nodes operational modes. Adaptation is achieved at the base stations/Rich nodes that are using ontologies to represent sensor nodes. The reasoning is based on observed and expected data that are also represented ontologically.

In [3] the use of contextual ontologies is investigated in order to achieve more flexible information processing by the middleware. Authors are claiming that flexibility can be achieved by the classification, characterization and definition of the relations among the pieces of contextual information. The hardware heterogeneity is not issues in this paper.

In [7] the use of sensors ontology for the representation of services in Ubiquitous computing systems is proposed. The paper only mentions possibility to use ontology for representing physical sensor information within the sensor ontology based on service oriented property of sensor data.

3. GLOSENT ARCHITECTURE

The presented approaches at best deal with some aspects of heterogeneity and, as a rule, do not cover semantic aspects of the hardware heterogeneity.

In this paper we propose the GLOSENT (GLOSENT - GLObal SENSor neTwork) architecture where we are trying to cope with heterogeneity issues in SWSNs at conceptual level, with emphasis on hardware heterogeneity resolution by using semantic technologies.

The illustration of proposed system architecture in which the client applications communicate with many diverse WSNs is shown in Figure 1.

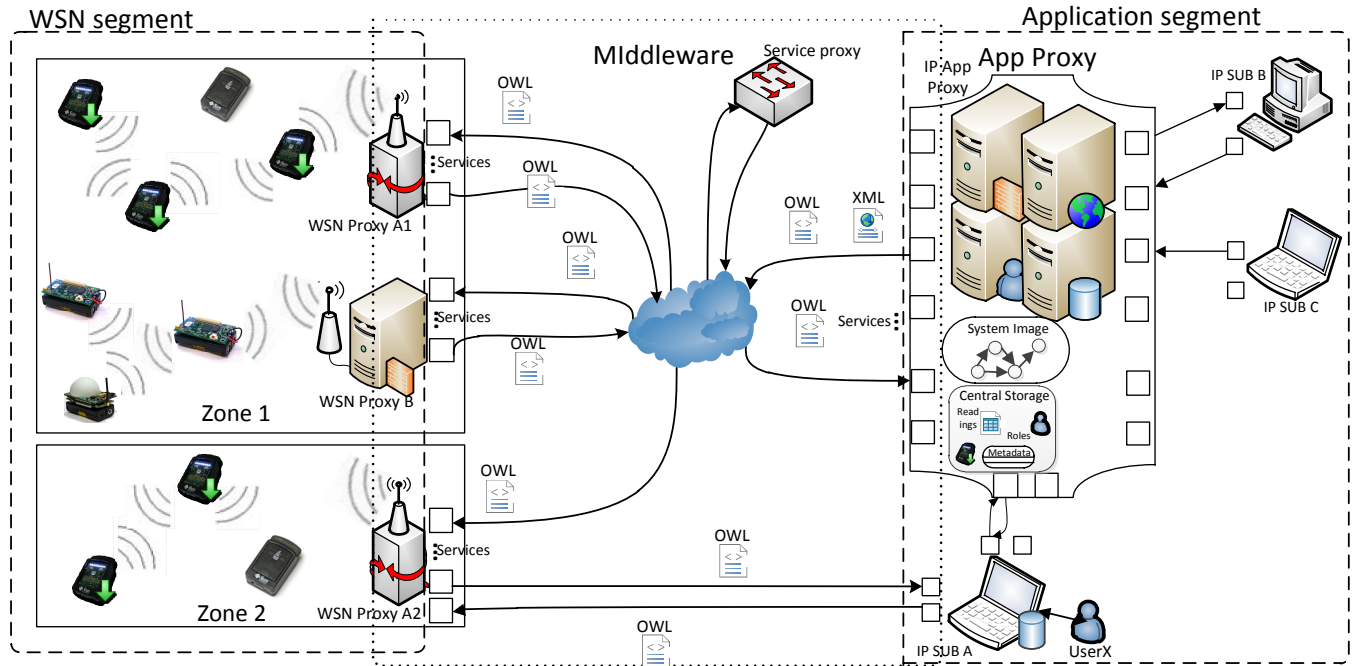


Figure 1. The GLOSENT architecture

The proposed architecture consists of three segments: *Application*, *Middleware* and *WSN*. The mediation among these segments is achieved by using proxies. The proxies that perform mediation among the segments are overlapping elements of the GLOSENT segments.

There are three kinds of proxies in the proposed architecture: *WSN proxy*, *Application proxy* and *Service proxy*.

The role of WSN proxy is to enable communication between WSNs and other segments. Due to the WSN hardware heterogeneity, the data delivered to WSN proxies are converted (by WSN proxies) to ontological representation.

Application proxy provides communication between user applications and the other segments of the architecture. Here the exchanged data are assumed to be represented by ontology.

Service proxy is aimed at finding and invoking appropriate services for applications.

We use an asynchronous communication model in which both application segment and WSN proxies are information consumers and providers.

3.1 Application segment

Application segment consists of an *Application proxy* and a number of *Subapplications*.

Subapplication is any end user application with an arbitrary implementation technology.

Application proxy mediates in the communication of the user Subapplications and other system entities (Figure 1). Subapplication In order to be integrated into the system, Subapplication must be able to use appropriate communication methods and data formats. Necessary information about communication methods and data formats are provided by Application proxy.

The main role of the Application proxy is to provide authorization for Subapplications. In addition, the Application proxy can provide specific locally residing applications that will be available to users (like those aimed at data acquisition and adaptation). Subapplications and WSN proxies can only access Application proxy by using its IP address.

GLOSENT is a distributed system which enables application segment to access multiple WSNs. For that purpose, a *Central storage* and the mechanism for keeping consistent WSNs states representation which is called *System image* are used.

System image contains the metadata describing all WSNs in SWSNs as well as the corresponding metadata values. It is stored at Application proxy. In order to obtain desired part of the system image from the Application proxy, Subapplication must be authorized by the Application proxy.

Managing WSNs is accomplished via Application proxy and Subapplications through WSN proxy services. When some change in WSN configuration takes place, the WSN proxy requests Application proxy to perform synchronization (update) procedure. All changes are recorded in the System image.

Although the authorization within the GLOSENT is performed by the Application proxy, for some specific Subapplications it can be done by a WSN proxy as well (for example, some Subapplication not intending to access database, but only some SN). For the sake of higher efficiency, the list of temporary permissions is maintained.

3.2 WSN segment

In the GLOSENT architecture, WSN segment is modeled by using the metadata that describe its structure (generalized SNs models, WSN topology, role that specific generalized SN plays in communication), the method of use and control (i.e. middleware or direct access to WSN, constraints imposed to sensor devices, etc.), and the data delivered by WSN (sensor data formats, SN hardware status, etc.).

In our model, SN is a generalized notion related to any WSN device (base stations, rich uncles, routers, etc.) thus enabling a uniform WSN representation. Such a representation is implemented at WSN proxies by a corresponding ontology.

The WSN segment consists of multiple WSNs that can be implemented in different ways, but with common task to mediate between the ontological representation of SN and the concrete SNs implementations. The concrete WSN implementation is described by using SN's metadata, identifiers and revisions. Since the WSN proxy maps each SN to its ontological representation, the communication procedure that SN uses to introduce itself, must be known to WSN proxy. Also, WSN proxies should implement appropriate communication channels which, in the case of GLOSENT are Internet.

The proposed model of the WSN proxies enables simulation of the operations that are not directly available in the WSN implementations (for example, *event based* access to SNs).

Due to its complex functionality (ontology creation, communication, and additional functionalities), the WSN proxy's hardware requirements in our architecture are significant.

3.3 Middleware segment

The Middleware segment mediates the communication of WSN segment and application segment in a SWSN.

The ontological representation of the middleware data facilitates the integration of SWSN segments. SWSN segments use appropriate proxies (WSN proxies, Application proxy, and Service proxy) to access middleware (see Figure 1).

The pivotal role of the middleware can be explained through an example of the data flow within the SWSN. SNs are the sources of data. The data are routed through a network of SNs to a WSN proxy. Since the WSN proxy is WSN sync but also a data source for the middleware, the appropriate data conversion takes place at this point. Here, raw data are converted to ontology and the WSN proxy delivers ontologically represented data to the middleware.

Additionally, Middleware provides up-to-date ontologically represented data to the Application proxy. At Application Proxy the received data are integrated into the System image and the Central storage is updated.

4. WSN model

WSN model consists of two parts: System image and Central storage.

4.1.1.1 System Image

System image is the representation of the current state of all WSNs. It consists of metadata models describing different WSN hardware platforms, metadata values describing particular devices and their relations, and sensor data (raw and/or aggregated sensor readings).

In our model, the System image was represented by using ontologies, where generalized sensor nodes were interpreted as sets of components similar to the model proposed by [20].

Components were modeled by class *Component* as attribute-value pairs, where the values are taken from the predefined sets of feasible values. In order to describe measured values that have the units of measurement, *MeasureUnit* class was introduced. A component is linked to its measure unit by *hasMeasureUnit* property. Class *Revision* and a property *hasRevision* are aimed at representing moment in which the values have been measured. According to the proposed model each concrete component (i.e. instance of the class *Component*) contains an attribute name, a value of the attribute and a timestamp for that value. It can also include the unit of measurement when needed.

Generalized sensor nodes were modeled by *SensorNode* class linked to its components by *hasComponent* property. Each *SensorNode* class that represents a particular WSN hardware platform has a feasible combination of *Component* classes. These constraints can be modeled by using class restrictions and implementing SWRL rules [23]. Each concrete, physical sensor node, including its current state is completely represented by an instance of the *SensorNode* class which is a collection of *Component* instances.

The relationships among generalized sensor nodes are represented by the class *Connection*. Due to communication roles of its nodes, WSN is considered an oriented graph and two properties (*hasSourceNode* and *hasDestinationNode*) which link *Connection* class to *SensorNode* class were introduced. Since it is expected that a connection between two generalized sensor nodes can have its own attributes, classes *Connection* and *Component* are linked by using *hasComponent* property. According to the proposed ontology, system image is seen as a collection of *Connection* instances connecting *SensorNode* instances.

In order to represent various WSN topologies like hierarchical organization of WSN nodes and their roles (cluster heads, aggregators, routers ...), it is assumed that a link between two SNs can be characterized by the set of components with appropriate constraints. For that purpose new restrictions and SWRL rules should be introduced. Figure 2 shows the proposed ontology of sensor nodes and their relationships in WSN.

For the implementation of ontology, we have used OWL DL [15] and Protégé [17].

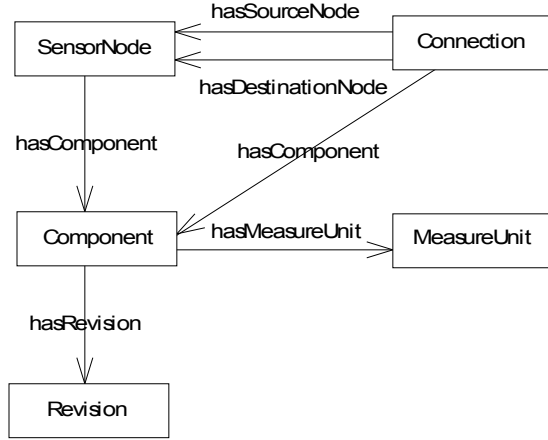


Figure 2. WSN Ontology

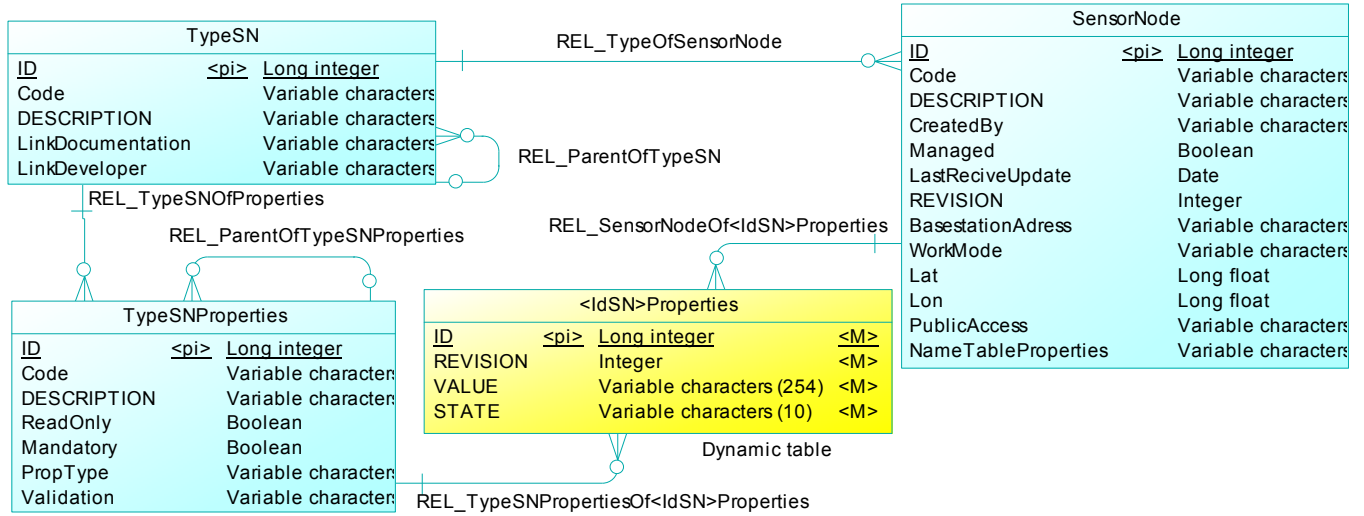


Figure 3. Part of database scheme for connecting metadata and data

The history (versioning) mechanism uses the *Revision* field in dynamic tables to represent WSN components changes. This mechanism serves the situations when the WSN proxy is unavailable in order to prevent provision of an invalid (outdated) *System image* to Subapplication.

Two typical situations that use revision mechanism are: (1) the situation when a client changes some parameters of generalized SNs and the Application proxy sends changes to the WSN proxy that is currently unavailable, and (2) the situation when a Subapplication directly accessing WSN proxy is trying to change the state of generalized SN without knowing the actual state of WSN.

In the first situation, as soon as the WSN proxy establishes communication with the Application proxy, the WSN proxy compares the state of WSN against its status in the database. If the revisions do not match, the WSN proxy delivers to client only the parameters that were changed in the meantime.

In the second situation, the discrepancy of the revision number appearing in subapplication request and the revision number stored at WSN proxy for the particular SN is detected by the WSN proxy. WSN Proxy rejects the request and sends the

4.1.1.2 Central Storage

Central Storage implements the data persistency. It stores the System image as well as the information about Subapplications (users) and their access rights. In our model the Central storage is implemented as relational data model using Postures DBMS [16].

Heterogeneous hardware platforms and particularly the need for representing changeable sensor data make the use of “conventional” relational data model inappropriate due to inefficient memory use (sparse data problem). This problem is solved by using dynamic tables (one table for each realization of the SN), that links specific metadata descriptions of hardware platform and instance of SN in the database (see Figure 3).

response containing version mismatch to Subapplication. Afterwards, Subapplication invokes the Application proxy service to obtain actual System image.

5. CASE STUDY: MODELING SUNSPOT IN GLOSENT

The case study presents the model of the WSN hardware platform SunSPOT.

SunSPOT is a WSN sensor node developed by Sun Microsystems. The device was developed based on IEEE 802.15.4 standard. Originally, SunSPOT is used to measure light, temperature and acceleration, but its functionality can be extended with additional sensors.

5.1 SunSPOT System image

In order to represent the concrete characteristics of SunSPOT, the proposed WSN ontology (Figure 2) was extended with classification of the SunSPOT’s components (Figure 4). The classification covers some basic functionalities of SunSPOT device [22]. The focus of the classification is on hardware characteristics and sensing functionalities.

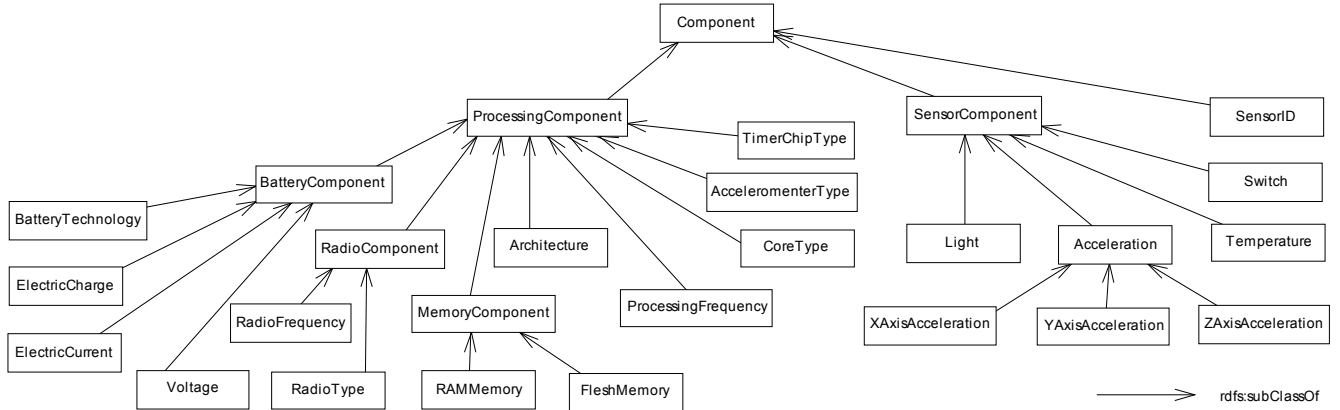


Figure 4. Classification of components

Classes of processor components (*ProcessingComponent*), sensor components (*SensorComponent*) and components for identifying a sensor node (*SensorNodeID*) were modeled.

Within the class of processing components, in addition to the classes that describe the processor itself (*Architecture*, *ProcessingFrequency*, *CoreType*, *AccelerometerType*, *TimerChipType*), the classes that describe battery (*BatteryComponent*), radio (*RadioComponent*) and memory (*MemoryComponent*) were modeled. The classes *BatteryTechnology*, *ElectricCharge*, *ElectricCurrent* and *Voltage* are subclasses of the class *BatteryComponent*. The classes *RAMMemory* and *FlashMemory* are subclasses of the class that models memory (*MemoryComponent*). The classes *RadioFrequency* and *RadioType* are subclasses of the class *RadioComponent*. The classes *Light*, *Switch*, *Temperature* and *Acceleration* are subclasses of the class *SensorComponent*.

The classification of components is extensible, meaning that the new SN hardware implementation could at most require an extension of the classification.

The proposed ontology is publicly available at [25].

5.2 SunSPOT Central storage

The Central storage for the SunSPOT is presented in this section. The Central storage is obtained by the transformation of the System image into relational database. The transformation is done by mapping ontologies to relational database.

Database table *TypeSN* describes general properties of each hardware platform (Table 1). The *CODE* field is the identification of the hardware platform ontology. *DESCRIPTION* field is a detailed description of SN type. *LinkDocumentation* and *LinkDeveloper* are user defined fields containing the URLs of official documentation and developer site respectively. The new SN type record can be derived from the existing one (Figure 3).

Table 1. Table *TypeSN*

TypeSN	
Field Name	Value
CODE	SunSPOT
DESCRIPTION	Sun SPOT (Sun Small Programmable Object Technology) is a wireless sensor network (WSN) mote developed by Sun Microsystems...
LinkDocumentation	http://www.sunspotworld.com/docs/index.html
LinkDeveloper	http://www.sunspotworld.com

TypeSNProperties table contains metadata that describe the specific property of SN.

The property type is COMPLEX (Table 2) if the property is the composite one.

Simple property (Table 3) is a property which holds a value of a basic data type (e.g. *Integer*).

The *TypeSNProperties* tables are obtained by transforming the *Component* classes from System image. *CODE* field defines the short code that is the name of the property (*Component*'s name). Fields *PropType*, *Validation*, *ReadOnly* and *Mandatory* are used for storing metadata of a specific property. *ReadOnly* field is used to specify that the property value can not be changed. *Mandatory* field is used to denote if this property is mandatory in the SN configuration. *Validation* field contains property constraints (e.g. ranges, domains, rules).

One example of the complex property is the *Radio* property (Table 2). It consists of *Channel*, *Pan ID* and *Transmit Power* subproperties.

Table 2. Complex property Radio

TypeSNProperties	
Field Name	Value
CODE	Radio
TypeSN_ID	SunSPOT
PARENT_ID	-
DESCRIPTION	Set of properties for maintaining radio connection
PropType	COMPLEX
Mandatory	TRUE
ReadOnly	FALSE
Validation	-

Table 3. Simple property Transmit Power

TypeSNProperties	
Field Name	Value
CODE	TransmitPower
TypeSN_ID	SunSPOT
PARENT_ID	Radio
DESCRIPTION	Property for maintaining transmit power
PropType	INTEGER
Mandatory	TRUE
ReadOnly	FALSE
Validation	MIN=-32,MAX=31

The table *TypeSensor* contains metadata that describe a sensor type. Table 4 presents SunSPOT sensor type.

Table 4. Temperature sensor type

TypeSensor	
Field Name	Value
CODE	ADT7411
DESCRIPTION	Temperature built-in sensor
SensorKind	Ambient
SensorUnits_ID	Celsius
TypeMeasurement_ID	Temperature
MINReadings	-45
MAXReadings	125
Granularity	0.5
WorkingConditions	MIN:-10,MAX: 60

Here the field *SensorKind* defines the type of environment in which the sensor operates. *SensorUnits_ID* field links sensor type to its measure unit. The sensed phenomena are represented by the

field *TypeMeasurement_ID*. The constraints of sensor readings are modeled by fields *MINReadings* and *MAXReadings* respectively, while the level of granularity is represented by the field *Granularity*, and the working conditions of sensors by the field *WorkingConditions*.

SensorNode table contains values that describe the states of concrete SNs (Table 5). The field values are combinations of data obtained from WSN proxies and/or those created by users. The field *LastReciveUpdate* contains the time when the latest communication of SN with a WSN proxy occurred. The fields *Lat* and *Lon* are geographical latitude/longitude of SN position. The field *TypeSN_ID* points to the corresponding SN type, while the field *CODE* is the SN ID (address). *WorkMode* field defines the role that the SN plays (e.g. emitter, router, basestation, etc.). The name of the corresponding dynamic table is stored in the *NameTableProperties* field.

Table 5. Sensor node

SensorNode	
Field Name	Value
CODE	0000.591F
TypeSN_ID	SunSpot
CreatedBy	BASESTATION
LastReciveUpdate	10/28/2010 5:53 PM
BasestationAdress	0000.6D29
REVISION	1
WorkMode	EMITER
Lat	45.251667
Lon	19.836944
NameTableProperties	0000.591FProperties

The table *Sensor* contains data representing the properties of the particular sensor (Table 6). It contains the revision of the last change (*REVISION*), sampling rate (*MeasuresEvery*), alarm configuration (*TresholdUP*, *TresholdDOWN*, *Alarm*), as well as *SensorNode_ID* and *TypeSensor_ID*. *SensorNode_ID* identifies SN, while *TypeSensor_ID* points to the corresponding *SensorType*.

Table 6. Sensor

Sensors	
Field Name	Value
SensorNode_ID	0000.591F
TypeSensor_ID	ADT7411
CODE	0000.591F_AD7411_1
REVISION	1
MeasuresEvery	3
TresholdUP	100
TresholdDOWN	-
Alarm	FALSE

Naming convention for dynamic tables is as follows. Entity id is used as a prefix to table name. The variable part of the table name is enclosed in $\langle \rangle$. $\langle IdSN \rangle Properties$ contains property values of SN. The metadata stored in the table $TypeSNProperties$ describe the usage of the data from dynamic tables. For each data row of the table $SensorNode$ a distinct table containing SN's attributes is created. The table names are created as combinations of SN identifier and predefined suffix "Properties", as in Table 7 (e.g. 0000.591Fproperties). The table fields are the revision number of the specified property ($REVISION$), its current value ($VALUE$) and its current state ($STATE$). $STATE$ field denotes whether the property is added, changed or removed (ADDED, CHANGED, REMOVED). The field $SensorNode_ID$ points to the SN table, while the field $TypeSNProperties_ID$ points to the property of SN type (Figure 3).

Table 7. Transmit Power property of SN entity

0000.591FProperties	
Field Name	Value
SensorNode_ID	0000.591F
TypeSNProperties_ID	TransmitPower
REVISION	1
VALUE	0
STATE	ADDED

Tables $\langle IdSNS \rangle ReadingsSensors$ contain readings of a particular sensor (an example for temperature readings is shown in Table 8). One table is created for each record of the table $Sensors$; it contains timestamp ($DateTime$), measured value ($VALUE$) and the identifier ($Sensor_ID$).

Table 8. Readings of temperature sensor

0000.591F_ADT7411_1ReadingsSensors	
Field Name	Value
Sensor_ID	0000.591F_ADT7411_1
DateTime	10/28/2010 5:53 PM
VALUE	45

6. CONCLUSION

This paper is concerned with the problem of hardware heterogeneity in large systems of WSNs. We have proposed an ontologically-based approach for modeling SWSNs. The topology of the WSN is represented by high-level ontology, while the semantics of WSN components is represented by appropriate classifications. The GLOSENT architecture that facilitates integration of heterogeneous SWSN segments is also proposed. The GLOSENT architecture relies on SOA and ontological representation of WSNs and data. The segments of the GLOSENT architecture aimed at resolving heterogeneity problems are described. In addition, as a case study the concrete SN device SunSPOT is represented using proposed ontology.

Since the aim of GLOSENT is to facilitate dealing with heterogeneity in SWSN in general, further research will be directed towards development of ontologies dealing with other aspects of heterogeneity in SWSN and deployment of these ontologies in GLOSENT. In order to provide advanced functionalities of the SWSN (adaptive SWSN) development of ontologies for context-sensitive usage should also be the subject of further research.

7. ACKNOWLEDGMENTS

The results presented in this paper are part of the research conducted within the Grant No. III-47003 funded by the Ministry of Science and Technological Development of the Republic of Serbia.

8. REFERENCES

- [1] Avancha, S., Patel, C., Joshi, A. 2004. Ontology-driven adaptive sensor networks. *Mobile and Ubiquitous Systems: Networking and Services*, 2004. *MOBIQUITOUS 2004. The First Annual International Conference on* (August 22-26, 2004). 194 - 202. DOI=<http://dx.doi.org/10.1109/MOBIQ.2004.1331726>
- [2] Bröring, A., Jürrens, E. H., Jirka, S., and Stasch, C. 2009. "Development of Sensor Web Applications with Open Source Software," in *OpenSource GIS 2009*, University of Nottingham (Suchith Anand, 2009)
- [3] Gomez, L., and Laube, A. 2009. Ontological Middleware for Dynamic Wireless Sensor Data Processing. In *Proceedings of the 2009 Third International Conference on Sensor Technologies and Applications* (SENSORCOMM '09). IEEE Computer Society, Washington, DC, USA, 145-151. DOI=<http://dx.doi.org/10.1109/SENSORCOMM.2009.121>
- [4] Gruber, T. R. 1993. A translation approach to portable ontology specifications. *Knowledge Acquisition*. 5, 2 (June 1993), 199-220. DOI=<http://dx.doi.org/10.1006/knac.1993.1008>
- [5] Hadim, S., and Mohamed, N. 2006. Middleware: Middleware Challenges and Approaches for Wireless Sensor Networks. *Distributed Systems Online, IEEE*. 7, 3 (March 2006), 1-23. DOI=<http://dx.doi.org/10.1109/MDSO.2006.19>
- [6] Jang, W., Healy, W.M., and Skibniewski, M.J. 2008. Wireless sensor networks as part of a web-based building environmental monitoring system. *Automation in Construction*. 17, 6 (August 2008), 729-736. DOI=<http://dx.doi.org/10.1016/j.autcon.2008.02.001>
- [7] Kim, J.H., Kwon, H., Kim, D.H., Kwak, H.Y., and Lee, S.J. 2008. Building a Service-Oriented Ontology for Wireless Sensor Networks. In *Proceedings of the Seventh IEEE/ACIS International Conference on Computer and Information Science (icis 2008)* (ICIS '08). IEEE Computer Society, Washington, DC, USA, 649-654. DOI=<http://dx.doi.org/10.1109/ICIS.2008.100>
- [8] Levis, P., and Culler, D. 2002. Maté: A tiny virtual machine for sensor networks. *SIGOPS Oper. Syst. Rev.* 36, 5 (October 2002), 85-95. DOI=<http://doi.acm.org/10.1145/635508.605407>

- [9] Liu, T., and Martonosi, M. 2003. Impala: a middleware system for managing autonomic, parallel sensor systems. *SIGPLAN Not.* 38, 10 (June 2003), 107-118. DOI=<http://dx.doi.org/10.1145/966049.781516>
- [10] Liu, J., Zhao, F. 2005. Towards semantic services for sensor-rich information systems. *Broadband Networks, 2005. BroadNets 2005. 2nd International Conference on* (Boston, MA, USA, October 7-7, 2005). 967 - 974. DOI=<http://dx.doi.org/10.1109/ICBN.2005.1589709>
- [11] Madden, S. R., Franklin, M. J., Hellerstein, J. M., and Hong, W. 2005. TinyDB: an acquisitional query processing system for sensor networks. *ACM Trans. Database Syst.* 30, 1 (March 2005), 122-173. DOI=<http://dx.doi.org/10.1145/1061318.1061322>
- [12] Masri, W., and Mammeri, Z. 2007. Middleware for Wireless Sensor Networks: A Comparative Analysis. *Network and Parallel Computing Workshops, 2007. NPC Workshops. IFIP International Conference on* (Liaoning, September 18-21, 2007). 349 - 356. DOI=<http://dx.doi.org/10.1109/NPC.2007.165>
- [13] Molla, M. M., and Ahamed, S. I. 2006. A Survey of Middleware for Sensor Network and Challenges. In *Proceedings of the 2006 International Conference Workshops on Parallel Processing* (Washington, DC, USA, 2006). IEEE Computer Society, 223-228. DOI=<http://dx.doi.org/10.1109/ICPPW.2006.18>
- [14] Murphy, A. L., and Heinzelman, W. B. 2002. *Milan: Middleware Linking Applications and Networks*. Technical Report. University of Rochester, Rochester, NY, USA.
- [15] OWL. WWW, available at: <http://www.w3.org/2004/OWL/#specs>
- [16] PostgreSQL. WWW, available at: <http://www.postgresql.org/>
- [17] Protégé. WWW, available at: <http://protege.stanford.edu/>
- [18] Ramos, M. L., Leguay, J., and Conan, V. 2007. Designing a Novel SOA Architecture for Security and Surveillance WSNs with COTS. *Mobile Adhoc and Sensor Systems, 2007. MASS 2007. IEEE International Conference on* (Pisa, October 8-11, 2007). 1 - 6. DOI=<http://dx.doi.org/10.1109/MOBHOC.2007.4428703>
- [19] Simon, D., Cifuentes, C., Cleal, D., Daniels, J., and White, D. 2006. JavaTM on the bare metal of wireless sensor devices: the Squawk Java virtual machine. In *Proceedings of the 2nd international conference on Virtual execution environments* (VEE '06). ACM, New York, NY, USA, 78-88. DOI=<http://doi.acm.org/10.1145/1134760.1134773>
- [20] Sharman, R., Kishore, R., and Ramesh, R. 2007. Using Ontologies in the Semantic Web: A Survey. *Ontologies: A Handbook of Principles, Concepts and Applications in Information Systems (Integrated Series in Information Systems)*. Springer-Verlag, New York, 2007, 79-113, DOI=http://dx.doi.org/10.1007/978-0-387-37022-4_4
- [21] Souto, E., Guimarães, G., Vasconcelos, G., Vieira, M., Rosa, N., Ferraz, C., and Kelner, J. 2005. Mires: a publish/subscribe middleware for sensor networks. *Personal Ubiquitous Comput.* 10, 1 (December 2005), 37-44. DOI=<http://dx.doi.org/10.1007/s00779-005-0038-3>
- [22] SunSpot. WWW, available at: <http://www.sunspotworld.com>
- [23] SWRL. WWW, available at: <http://www.w3.org/Submission/SWRL/>
- [24] Terfloth, K., Güneş, M., and Schiller, J. H. 2009. Middleware for Wireless Sensor Networks: The Comfortable Way of Application Development. *Guide to Wireless Sensor Networks*, Springer, London, 2009, 583-606. DOI=http://dx.doi.org/10.1007/978-1-84882-218-4_23
- [25] The Basic SunSPOT Ontology, available at: <http://informatika.ftn.uns.ac.rs/SinisaNikolic?action=AttachFile&do=get&target=SunspotBasicModel.owl>
- [26] The Open Geospatial Consortium . WWW, available at: <http://www.opengeospatial.org/>
- [27] TinyOS Project. WWW, available at: <http://www.tinyos.net/http://www.tinyos.net>