Journal of Network and Computer Applications 39 (2014) 191–201

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

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

# Simulation framework for real-time database on WSNs

Ousmane Diallo [a,b], Joel J.P.C. Rodrigues [a,*], Mbaye Sene [b], Jaime Lloret [c]

[a] *Instituto de Telecomunicações, University of Beira Interior, Portugal*
[b] *Department of Mathematics and Informatics, UCAD, Senegal*
[c] *Integrated Management Coastal Research Institute, Universidad Politécnica de Valencia, Spain*

ARTICLE INFO

*Article history:*
Received 21 December 2012
Received in revised form
7 July 2013
Accepted 7 July 2013
Available online 18 July 2013

*Keywords:*
Wireless sensor networks
Real-time database management
Model
Simulator
Performance evaluation

ABSTRACT

Wireless sensor networks (WSNs) have been the focus of many research works. Nowadays, because of the time-critical tasks of several WSN applications, one of the new challenges faced by WSNs is handling real-time storage and querying the data they process. This is the real-time database management on WSN and it deals with time-constrained data, time-constrained transactions, and limited resources of wireless sensors. Developing, testing, and debugging this kind of complex system are time-consuming and hard work. The deployment is also generally very costly in both time and money. Therefore in this context, the use of a simulator for a validation phase before implementation and deploying is proved to be very useful. The aim of this paper is to describe the different specificities of real-time databases on WSN and to present a model for a simulation framework of the real-time databases management on WSN that uses a distributed approach. Then, the model of the simulator is described and developed in Java and a case study with some results demonstrates the validity of the structural model.

## 1. Introduction

Wireless sensor networks (WSNs) may be defined as a set of smart devices, called sensors, which are able to sense, process and transmit information about the environment on which they are deployed. These devices, distributed in a geographical area, collect information for users interested in monitoring and controlling a given phenomenon and transfer them to a sink node. The latter makes the information available to a gateway where remote users can access. So as to obtain those information, users use applications that communicate with the network through queries (Callaway, 2004; Sacks et al., 2003; Baronti et al., 2007).

Systems based on sensor networks are more and more used in many areas providing various types of WSNs (Mendes and Rodrigues, 2010). These different WSNs involved the development of many applications, which are generally connected to databases treating the amount of data collected from sensors. However, the processing time becomes increasingly critical for certain applications. These applications must query and analyze the data in a bounded time in order to make decisions and to react as soon as possible (Diallo et al., 2011). Some examples of the most popular applications are the following: the control of network traffic (Cranor et al., 2002), transactional analysis (web, banking or telecommunication transactions) (Cortes et al., 2000), human motion tracking application (Chen and Ferreira, 2009), the tracking of actions on dynamic Web pages (Zhu and Shasha, 2002; Chen et al., 2000), monitoring of urban or environmental phenomena (Mainwaring et al., 2002; Ulmer et al., 2003), and the sensors data management (Arasu et al., 2003).

Once the sensors perform their measurement, the problem of data storing and querying arises. Indeed, the sensors have restricted storage capacity (Silva et al., 2004) and the ongoing interaction between network devices and environment results in huge amounts of data.

There are two main approaches to data storage and querying in WSN: distributed and warehousing (Bonnet et al., 2000b). The first approach aims at exploiting the capacities of calculation of sensors. Some queries are distributed and evaluated among the nodes into the network. The objective is to locally calculate in order to limit sending of messages, reducing thus the energy consumption. In the second approach, warehousing, one has a centralized system. Collected data from sensors (in stream) are sent to a central database server, in which user queries are processed. Note that this last technique generates large data flows.

The data collected by the WSN must closely reflect the current state of the targeted environment. However, the environment changes constantly and the data are collected in discreet times. So, the collected data have temporal validity, as time advances they become less and less accurate, until the time where they do not reflect the state of the environment (Idoudi et al., 2009a, 2009b). It is fundamental that responses to application queries

* Corresponding author. Tel.: +351 27 524 2081; fax: +351 27 531 9899.
*E-mail addresses:* ousmane.diallo@it.ubi.pt (O. Diallo), joeljr@ieee.org (J.J.P.C. Rodrigues), mbaye.sene@ucad.sn (M. Sene), jlloret@dcom.upv.es (J. Lloret).

1084-8045/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.jnca.2013.07.001

ensure that returned data comply with logic and temporal constraints. In this context, real-time data management on WSNs is necessary to deal with those constraints. The main goal of real-time data management is to ensure temporal consistence data and process transactions within real-time constraints.

Developing, testing, and debugging this kind of complex system are time-consuming and hard work. The deployment is also generally very costly in both time and money. Therefore in this context, the use of a simulator for a validation phase before implementation and deploying is proved to be very useful (Sundani et al., 2010; Nasreddine, 2012; Egea-López et al., 2005).

The main contributions of this paper are the following:

- Description of the different characteristics of real-time databases management on WSN.
- Proposition of a model for a simulation framework of the real-time databases management on WSN that uses a distributed approach.
- Implementation of the model and performance evaluation.

The remainder of this paper is organized as follows: Section 2 presents the Wireless sensor networks. Section 3 discusses the related work, while the characteristics of real-time database management systems are exposed in Section 4. Section 5 briefly presents the data management on WSNs. In Section 6, the description of the real-time database simulation model is provided. In Section 7, some screenshots of the case study is presented to validate the model and Section 8 concludes the paper.

## 2. Wireless sensor networks

Generally a WSN has a large number of nodes distributed on an interest area and communicating between them so as to measure a physical quantity (e.g., pollution level in a given area) or to do an event monitoring (e.g., vehicles tracking). WSNs are used with different applications in many areas and are very important for applications that should be deployed in places hostile to human interventions (e.g., volcano monitoring). Each network node is considered smart and embeds these units: a sensor unit which provides a measure of environmental data (such as temperature, humidity, pressure, acceleration, sound, etc.), a processing unit, a storage unit, a communication unit, and an energy unit. The communication unit usually performs data transmission by means of radio (Akyildiz et al., 2002, 2007; Oliveira et al., 2011a; Oliveira and Rodrigues Joel, 2011b).

However, these resources are generally very limited, particularly those of storage and energy. The sensor network lifetime depends on the available energy in the nodes composing the network (Lin et al., 2011, 2012). This available energy is consumed by three activities (Akyildiz et al., 2002): sensing activity (data acquisition from the environment), communication (sending and receiving packets), which is essential to form a WSN, and data processing, which consists in some operations applied over data by smart sensors (Cho et al., 2001; Madden et al., 2005). However, the sensing and processing activities are much less expensive in energy consumption than the wireless communication activities (Madden et al., 2002). Therefore, saving energy by optimizing the communication activities is the main point attention in many algorithms designed for sensor networks.

Generally, proposed architectures for WSNs are based on the distribution of sensor nodes in a geographical area in such a way that sensors send collected data to a base station by using the routing techniques like in (Li et al., 2011); Al-karaki and Kamal, 2004).

## 3. Related work

A lot of simulation tools for WSNs have been already proposed. These simulators are generally designed to meet the development constraints (e.g. communication constraints, constraints on the nodes) related to WSNs and can be divided into classes according to the nature of the specified constraints.

The first class of simulators is the oriented network. They emphasize on the behavior of the wireless network and the protocol stack of the operation. These simulators are based on the computer network simulators designed for computers, such as NS-2 (Zhang et al., 2009), OMNeT (Xian et al., 2008; Varga, 2001), J-Sim (Sobeih et al., 2006), etc. Among these simulators it may be noted: SensorSim (Park et al., 2000), Castalia (Boulis, 2007, 2009), J-Sim sensor simulator (Sobeih et al., 2006), and Prowler (Simon et al., 2003). SensorSim is built on top of a NS-2 802.11 network model and it models a software model of the sensor node and a hardware model. The power models of the hardware components have been implemented and the state of the hardware model changes according to the function performed by the software model. Therefore, the power consumption of the network could be simulated. However, the CPU has not been implemented (Park et al., 2000). Castalia is an application-level simulator for wireless sensor network based on OMNeT++. It is parametric and can be used to evaluate various characteristics for specific applications. In Castalia, multiple simple modules can be linked together and form a compound module, e.g. a sensor node. The simple modules implement the atomic behavior of a model, for instance, the network stack layers of a sensor. Node modules are connected to wireless channel and physical process modules (Boulis, 2007, 2009). J-Sim has been developed at the University of Washington by the National Simulation Resource. It is a general-purpose network simulator modeled after NS-2, but unlike NS-2, J-Sim uses the concept of components, replacing the representation of each node as an object. J-Sim provides support for sensors and physical phenomena. Moreover, the energy modeling, except the radio energy consumption, is provided for sensor networks (Sobeih et al., 2006). Prowler is a probabilistic wireless sensor network simulator running on Matlab environment. It is an event-driven simulator that can be set to operate, either in deterministic mode, or in probabilistic mode in order to simulate the nondeterministic nature of certain component of the network such as the communication channel. Prowler is adapted to the development of algorithms and optimized protocols. However, it does not model the energy consumption of the sensor node (Simon et al., 2003).These simulators are oriented network and they do not deal with the databases management techniques, and particularly the real-time databases management techniques.

The second class of simulators is oriented node. They emphasize on the function of a single node with simple communication models. They are specific to targeted nodes and their operating systems. In addition, these simulators are used to verify the compatibility of a node with a given application. Among these simulators include: TOSSIM (Levis et al., 2003), ATEMU (Karir et al., 2004), and SEN (Sundresh et al., 2004). TOSSIM is a discrete event simulator for TinyOS (TinyOS, 2012) wireless sensor networks. TOSSIM can capture the behavior of the network of several TinyOS nodes at bit granularity. It is designed specifically for TinyOS/NesC applications to be run on MICA Mote platforms. With TOSSIM, designers can easily translate between running an application in the simulation environment and on motes (Levis et al., 2003). ATEMU is an instruction-level cycle-accurate emulator for WSN written in C language. It simulates programs of nodes with an accuracy clock cycle. ATEMU provides support for the AVR processor, and other function units on the MICA2 sensor node platform, such as the transceiver. Moreover, it provides users a complete system for debugging and monitoring the execution of their code via a GUI, called Xatdb (Karir et al., 2004). Sensor Environment and Network Simulator (SENS) is a wireless sensor network simulator with a layered architecture, modular. The components of this architecture are customizable and they model the application layer, network communication, and physical environment. SENS allows realistic simulations by using values from real sensors to represent the behavior of component implementations (Sundresh et al., 2004).

All these simulators do not fit within the real-time databases on WSNs. To the best of our knowledge, there is not a tool that takes into account the specific characteristics of the models of real-time databases for WSNs. However, true that our simulator is not a full end-to-end simulator like for instance OMNeT or any other full end-to-end simulator tool with all protocols of WSNs, it is based on a model that focuses on the real-time databases management on WSNs, which are by their characteristics and constraints very difficult to study and model.

## 4. Characteristics of data and transactions in real-time database systems

Like a traditional database management system (DBMS), a real-time DBMS (RT-DBMS) must process transactions and ensures that the logical consistency of the data is not violated. However, unlike a traditional DBMS, a RT-DBMS emphasizes on the temporal validity of the data and the time constraints or deadlines for transactions (Idoudi et al., 2009a, 2009b; DiPippo and Wolfe, 1997).

The main purpose of a RT-DBMS is to process transactions on time, while maintaining logical and temporal consistency of data. The temporal consistency expresses the need to maintain consistency between the current state of the targeted environment and the state as reflected by the database contents. The temporal consistency can be measured in two ways (Ramamritham, 1993):

- Absolute consistency, which deals with the need to maintain the view representing the state of the targeted environment consistent with the real state of the environment.
- Relative consistency, which concerns data derived from other ones.

To satisfy these temporal constraints, the structure of the data must include these attributes: (i) timestamp, which indicates the instant when the observation relating the data was made; and (ii) absolute validity interval (avi) that denotes the time interval following the timestamp during which the data are considered valid. Another attribute can be considered; the imprecision or data error (*DE*), which refers to how the current state of the targeted environment may differ from the measured data (Ramamritham, 1993). The data error on a data version *d* is defined by:

$$DE\,(d) = 100\left|\frac{current\ value\,(d) - update\ value\,(d)}{current\ value\,(d)}\right|(\%) \tag{1}$$

There are two kinds of transactions in real-time databases: update transactions and user transactions. Update transactions are executed periodically to update real-time sensor data, or sporadically to update the derived data in order to reflect the state of the real world. Derived data are the data computed using sensor data. User transactions, representing user queries, arrive aperiodically. They may read or write non real-time data, but only read real-time data (Idoudi et al., 2009a, 2009b). These update transactions also to comply with temporal constraints must have these following attributes: (i) liberation time that represents the moment on which all the resources for the transaction processing is available; (ii) computing time that indicates the execution time needed for the transaction; and (iii) maximum time, which indicates the maximum time limit for the transaction execution and the periodicity that refers to the frequency with which the transaction happens (Chagas et al., 2010).

To satisfy the logical consistency of the data, transactions must be processed with Atomicity, Consistency, Isolation and Durability (ACID) properties. But unlike the conventional databases, in real-time databases these properties are relaxed. Firstly, the atomicity may be relaxed. It is only applied to the sub-transaction that wants to deal with completely data consistency. Secondly, since timeliness is more important than correctness, in many situations, correctness can be traded for timeliness. Thirdly, the isolation allows transactions to communicate with others to better perform control functions. Similarly, in real-time databases, not all data must be permanent and some of them are temporal (DiPippo and Wolfe, 1997; Chagas et al., 2010; Ramamritham, 1993).

## 5. Data management on wireless sensor networks

Among the activities of wireless sensor nodes, the communication activities are more expensive in terms of energy consumption (Madden et al., 2002, 2005). Therefore, saving energy by optimizing the communication activities is the main point attention in many algorithms and architectures designed for sensor networks (Madden et al., 2005). There are two main approaches to data storage and query in WSNs (Bonnet and Seshadri, 2000; Bonnet et al., 2000): the warehousing approach and the distributed approach.

The warehousing approach is a centralized system (Bonnet et al., 2001). The data gathered by the sensors are sent to a central database server where user queries are processed. In this case, the sensors act as simply collectors. The warehousing approach is the most used one in data storage and query processing. However, it has some drawbacks, such as, the huge number of generated data can easily create a bottleneck on the central server; and the huge amount of information transferred wastes the resources. The wireless data transmission decreases consequently the WSN lifetime. Indeed, it is shown in (Madden et al., 2002) that the wireless data transmission is more expensive in terms of energy consumption than the data processing and the sensing activities. Moreover, it is clear that this approach is not adequate for non-historical queries because it involves time delay for the results.

In the other hand, the distributed approach exploits the capabilities of sensors calculation and the sensors act as local databases (Bonnet and Seshadri, 2000). It aims to locally calculate in order to limit sending of messages, reducing thus the energy consumption in the network. In this approach, the data are stored in a central database server and in the devices themselves, enabling one to query both. Here, the devices act as part of the database. This approach can offer several advantages: on sensors, the processing of queries is done on quasi real-time. In fact, query processing inside the devices themselves means that the most current data will be acquired. The data will certainly arrive to the users with their temporal validity. It supports long-running queries and instant queries. Furthermore, the distributed approach increases the lifetime of the network by reducing the communications activities among the devices.

There are three sorts of queries in sensor networks: (i) historical data queries, which are run against the server; (ii) instant queries, which are run against a device in an instant of time; and (iii) long-running queries, which refer to queries run against a device during a time interval (Bonnet and Seshadri, 2000; Neto et al., 2004).

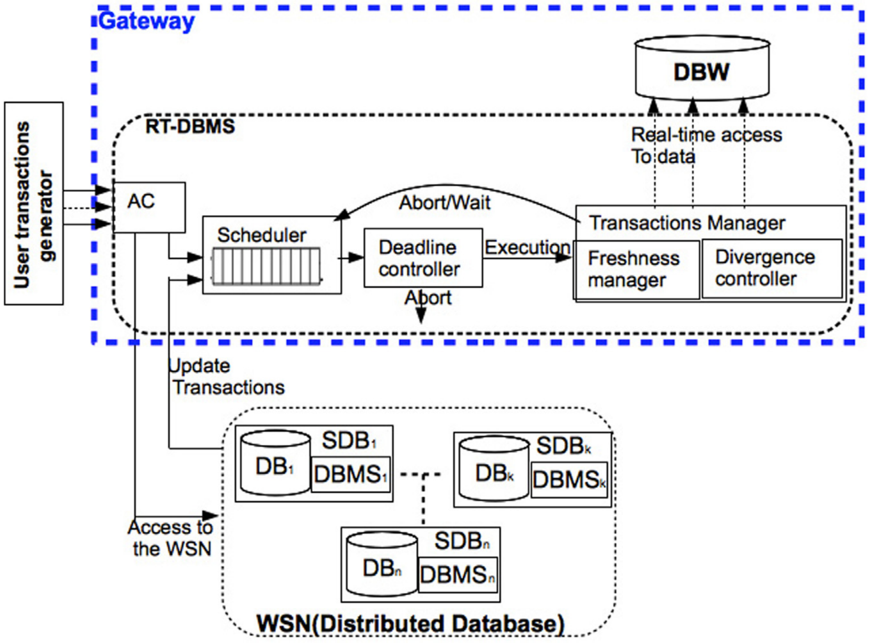

**Fig. 1.** Components of the simulator model for real-time database management techniques on WSNs.

## 6. Real-time database simulation model for WSNs

The design of the simulator involves the development of a library of models from which it will be possible to build our architecture of real-time databases management for WSN. These models are descriptions of behaviors and functions supported by each component of the system and can replace, in the simulation, the actual components of the architecture to be modeled.

Many researchers argue that the object-oriented model is more suitable than the relational model (Kim, 1995) to model real-time data, because of the nature of several real-time applications, which handle complex real-world objects with time constraints. Thus, many projects on real-time databases have chosen the object-oriented model for their systems (Terrier et al., 1997; Wolfe et al., 1997). However, the relational model (Madden et al., 2005) is the most used on real-time or distributed databases modeling.

The global architecture of our simulator model of real-time database management techniques on WSNs (see Fig. 1) uses the distributed data storage and the query approach and is inspired by the works made in (Neto et al., 2004). Thus, our model considers the gateway more general by integrating into it a database warehousing (DBW) that stores real-time data and a real-time database management system (RT-DBMS) that handles real-time transactions. The model of the wireless sensor network (WSN) nodes is built by using an event-based model, particularly an event-advance design (Granat, 2006). Thus, the nodes are programmed as follow: each time an event (user transaction) occurs in the network; all the nodes are triggered to compute the previous acquisitions, store the sensor data, and send all the underlying periodic update transactions to the database server. Moreover, each node targeted by the user transaction at the same time, after completion of its previous works, sends the results on required time. All these actions are scheduled on time based on the instant time that the network was initialized, the periodicity of acquisition and database update transactions, and the instant time that the user transaction arrives in the network. Therefore, the devices act as local databases, including a software component that manages the data storage and user queries with time constraints. The database server incorporates also an application that deals with time-constrained data and time and logic constrained transactions. We distinguish two kinds of transactions: update transactions and user transactions. The update transactions can be local updates for periodically updating the local sensor databases or server updates that periodically update the real-time data in the database server. The user transactions are real-time user queries that only read real-time data or historical queries that may read or write historical data.

Below, the description of the behavioral models of various elements that compose the simulator model of real-time database management techniques for WSNs is made, by explaining clearly the principle of object-oriented modeling used in the simulator developed in Java.

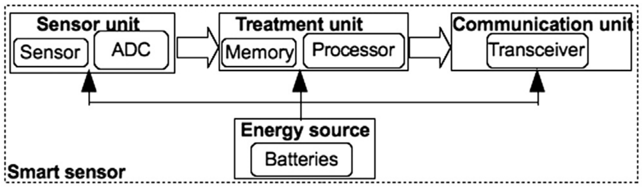

**Fig. 2.** Illustration of a sensor generic model.

### 6.1. Wireless sensor network

The WSN is composed by smart sensors (Ruiz et al., 2004), which are able to sense, process and transmit information about the environment on which they are deployed. These sensors act as local databases, named sensor database (SDB). Thus, the WSN forms a distributed database. An illustration of a generic sensor model with its basic components is given in Fig. 2.

The sensor unit is formed by the sensor and the Analog-to-Digital Converter (ADC). The sensor senses a change in the physical condition and transforms it to electrical signal. This signal is converted by the ADC to digital data, which can be stored in the memory and processed by the processor. The processor and the memory compose the treatment unit. The data communication into the network is performed by the transceiver located in the communication unit. This communication can be performed with or without wire. In the case of wireless communication, the most used technique is the radio frequency transmission (Adelstein et al., 2005). The batteries are used as energy sources.

For modeling our generic sensor as a local database, we use an object-oriented modeling as follow: a *SensorNetworkNode* class encapsulates all the properties and functionalities of the device. Thus, the measured physical phenomenon, because our case study uses temperature measures, is modeled by the *nextGaussian( )* function of the class Random that will generate pseudo-random available temperature measures (Deshpande et al., 2004) with best precision. This function is used in the acquisition phase (*acquisition()* function in our model). However, depending in the case study, others methods can be used, for example, a memory that contains the measures of the physical phenomenon (Nasreddine, 2012). Depending of the sampling frequency of the local updates, the *acquisition()* function is called in each period and the temperature value returned by the *nextGaussian( )*function is stored in coherent way in the memory (list of sensor data) of the sensor that stores sensor data. Moreover, functions that periodically send the acquired data, according to the periodicity of the database updates, to the database server are implemented. Concerning the other components of the sensor model, functions that receive queries, process them in real-time according to their time constraints, and send the data to the users are added. Thus, these functions model the *t*reatment unit and the communication unit. For the energy consumption, functions are also used to retrieve an amount of energy from the global energy remained on the sensor according to the data sensing, processing, transmission, and the sensor's duty cycle operation. Formula (2) below gives the energy consumption of the sensor node. For more details about the calculation of each term of the formula, readers can refer to (Briand et al., 2010).

$$E_{node} = E_{sensor} + E_{ADC} + E_{\mu c} + E_{transceiver} \tag{2}$$

To capture the actual energy consumption of each sensor's activity in the simulation, each activity needs to be calibrated by real measures. For that, the current consumptions of the main sensor's activities are configured according to the TelosB datasheet (CrossBow Inc, 2013; Nguyen et al., 2011), as presented in Table 1. In addition, the simulation used AA-size batteries with each nominal ampere-hour capacity of 2.4 Åh. However, the values of the parameters (the initial battery capacity according to its type, the cost of the operations with respect to the sensor type, etc.) can be configured in the main configuration file.

In real-time databases, the data should comply with logic and temporal constraints. Thus to comply with temporal constraints, the structure of the data should have these properties: (i) timestamp, (ii) absolute validity interval (avi), and the imprecision (Ramamritham, 1993). All these attributes including the value of the data and the location-identification of the sensor that acquires the data are encapsulated in our model of sensor data with a class named *SensorData*, which implements the data acquired by a sensor.

Because of the temporal constraints of sensor data, they become quickly invalid and should be periodically updated. Thus to deal with theses constraints, transactions have logical and temporal constraints. In the sensors we distinguish two kinds of transactions: the local update transactions and the read-only transactions. The local update transactions, represented in our implementation by the *LocalUpdate* class, represent a new acquisition of a data that will be stored in the sensor database. These transactions are periodic and firm (Pailler, 2006; Neto et al., 2004) and they are executed by call of the *acquisition()* function. The read-only transactions are the transactions sent by the user applications to read the local databases. In our case study, the transactions are the instant query transactions and the long-running transactions. They can be executed by calling the *instantQueryExecution()* and *longRunningQueryExecution()* functions respectively. These kinds of transactions are aperiodic.

**Table 1**
Nominal power consumption of TelosB nodes (Nguyen et al., 2011).

| Components | Mode | Current draw |
|---|---|---|
| Module | Active | 1.8 mA |
| | Sleep | 5.1 μA |
| | Receive | 19.7 mA |
| RF-Transceiver | Transmit(at 0 dBm) | 17.4 mA |
| | Sleep | 0.01 mA |

As already mentioned above, the processing model in the sensor nodes is built by using an event-based model, particularly an event-advance design (Granat, 2006). Thus, the nodes are programmed as follow: each time an event (a read-only transaction) arrives in the network, all the nodes are triggered to compute the previous acquisitions, store the sensor data, and send all the underlying periodic update transactions to the database server. Moreover, each node targeted by the user transaction at the same time, after completion of its previous works, sends the results on required time. All these actions are scheduled on time based on the instant time that the network was initialized, the periodicity of acquisition and database update transactions, and the instant time that the user transaction arrives in the network.

The implementation of this event-advanced design technique is done as follow: we know the instant time that the network was initialized, let call it initialize-time. We know the arrival time of the user query, let call it now, and finally we know the periodicity (let call it $T$) of the local update transactions. Thus, the difference between now and initialize-time, divided by the periodicity ($T$) gives us the number of local updates (let call it $Q$) that the sensor should perform before processing the query. Finally, having the release time (the date on which the first instance of the query should be activated) included in the local update transaction, let call it $r$, we can compute the activation dates for each instance of the local update transaction with this formula (Pailler, 2006):

Let $Q \in N^+$, by using $Q$ the size of a loop with $k$ as index started at one, then the waking date of the $k$th instance of the local update transaction is

$$r^k = r + (k-1)T \tag{3}$$

The same technique is used to calculate the activation date for all the instances of the server update transactions (explained hereafter) that should be done before the arrival of user transactions. Thus, by scheduling all the instances of the local and server update transactions, and the user transaction with respect to their activation dates, we can manage the data storage and transaction processing with respect to the logic and time constraints of data and transactions.

## 6.2. User transactions generator

The transactions generator is a component, which is responsible for generating the user transactions aperiodically. Thus, it is composed of an interface where the user transactions will be parametered to be sent into the system and a pseudo-random function that provides the time intervals representing the sending moments of transactions. As mentioned above, the properties of a specific transaction are also encapsulated into a specific transaction class. For example using the transaction class, a long-running query is parametered as: *Transaction <idTrans=12 startTrans="Dec 03, 2012 11:09:34" perTrans=1 time unit (t.u.), idLocSensor=0…5, endTrans="Dec 03, 2012 11:09:40">*. In this example, the query numbered 12 is interested of the values of sensor readings for sensors in the locations 0 through 5. The query should be executed in each targeted sensor from from *Dec 03, 2012 11:09:34* to *Dec 03, 2012 11:09:40* and the values is sampled each 1 t.u.

### 6.3. Gateway

The gateway usually is used to make the connection between the wireless sensor network and the external networks (e.g. IP). It has the same power as a PC. In our model the gateway is more general by including a database warehousing (DBW) that stores real-time data and a real-time database management system (RT-DBMS) that handles in real-time the transactions.

#### 6.3.1. RT-DBMS

The admission controller (AC) has the task of controlling user transactions that are accepted in the system. Thus, the user transactions that represent instant or long-running queries will be sent to the WSN, whereas the real-time user transactions and the historical queries will be directed towards the server database.

In that phase, we consider now the real-time user transactions that arrive aperiodically to read the real-time data storing in the database server, modeled by the *RTreadTransaction* class and the real-time update transactions that arrive periodically from the WSN for updating the real-time data in the server database, modeled by the *ServerUpdate* class (Idoudi et al., 2009b). Thus, the real-time user transactions include among other properties their computing time (Pailler, 2006) and the real-time update transactions have among other attributes (i) the liberation time, (ii) the computing time, (iii) the deadline, and (iv) the periodicity (Chagas et al., 2010).

The real-time user transactions admitted in the system and the server update transactions from the sensors are placed in a waiting queue before being sent to the transaction manager. All these transactions in wait are scheduled by the scheduler according to their priorities, particularly their deadlines. In fact, we use the Earliest Deadline First (EDF) protocol (Liu and Leyland, 1973), which orders the transactions according to the principle that the transaction that has the nearest deadline must be executed in priority.

The transaction manager consults regularly the waiting queue, updates it, and checks the validity of transactions (i.e. transactions that have not yet missed their deadline in relation to the current time). The transactions that have lost their deadline are aborted, the others are still handled by the transaction manager for a possible execution: the freshness manager checks the freshness of data that will be accessed by a transaction. If the data is outdated then the transaction is waiting in a queue. Since several transactions can access the data stored on the database server, a concurrence control protocol is needed to deal with the cases in which concurrent transactions may lead to inconsistent data in the database. In our model we use the Epsilon Serializability (ESR) criterion (Pu, 1991) to deal with concurrent transactions. Indeed, the Epsilon Serializability (ESR) criterion is well adapted in the processing of firm (the deadline of the transaction must be met, otherwise the transaction is rejected) real-time transactions (Bouazizi et al., 2004).

The ESR relaxes the severity of the classic serializability (SR) in the transaction processing by allowing a limited inconsistency in the database. This limited inconsistency is automatically maintained by the divergence control algorithms (DC), in the same way that the SR is managed by the concurrency control techniques (CC). In our model we use the DC algorithm with two phases (Bouazizi et al., 2004) named *2PLDC*, which is an extension of the algorithm of *2PL* concurrency control of the classic serializability. In this algorithm, the update transactions called write Epsilon-transaction (*ETw*) export an inconsistency in the database, while user transactions called read Epsilon transaction (*ETr*) import an incoherence from the base. These inconsistencies are gathered in two accumulators and the transactions can continue their executions as long as these accumulators do not exceed a certain given limit value.

Thus, in addition to the attributes mentioned early in this paragraph, the real-time update transaction becomes an *ETw* and incorporates two other attributes in our model: an attribute that records the total amount of inconsistency exported in the database and an attribute that records the maximum imprecision that the real-time update transactions (*ETw*) can export in the database. These attributes are modeled respectively by *accumuExport* and *limExport* in the model. Similarly, the real-time user transaction becomes an *ETr* and has two more attributes: an attribute that records the total amount of inconsistency imported by the *ETr* modeled by *accumuImport* and an attribute that records the maximum imprecision that the *ETr* can import modeled by *limImport* in the model. All such characteristics will permit to deal with both the time and logic restrictions.

This divergence control algorithm is implemented to allow conflicting transactions (Read/Write in the serialisability classic) to execute concurrently in way that their scheduling does not lead an imprecision that is higher than the one accepted in the data. Thus, when an *ETr* wants to access a data being running by an *ETw*, we increment the value of the variables *accumuImport* and *accumuExport*, respectively of the *ETr* and the *ETw* and verify if

$$accumu\ Import \leq lim\ Import \tag{4}$$

$$accumu\ Export \leq lim\ Export \tag{5}$$

In the same fashion, when an *ETw* demands to access a data being executed by several *ETr*, all the *ETr* increments by one unit their *accumuImport* and verify if formula (4) is true. However, the *ETw* should verify its *accumuExport* regarding the incrementation by the number of *ETr* in conflict with it. If condition (5) is false or if any *accumuImport* of one *ETr* exceeds its *limImport*, then we make wait or abort the transaction which demands to access the data in processing. Thus, this mechanism allows several real-time transactions to meet their deadlines.

#### 6.3.2. Database warehousing

The sensor data are modeled by the *SensorData* class, which encapsulates these attributes: the location identification of the sensor that acquires the data, the timestamp, absolute validity interval (avi), the imprecision, and the value of the data. Thus, the database warehousing (DBW) is a list of a sensor data and is modeled by the *Memory class*. The size of this memory can be fixed on the beginning of the simulation experience. The coherency of the DBW is maintained by the real-time update transactions and the Epsilon Serializability (ESR) criterion. This allows to simulate a real-time database as in the real databases.

Fig. 3 illustrates the class diagram of the implementation of the whole simulation framework.

## 7. Performance evaluation of a simulator model for real-time databases management techniques in WSN based on a distributed approach

The model considers a WSN where the sensor nodes, by using an event-based design, receives user queries, computes the previous acquisitions of temperature, send all the underlying periodic update transactions to the database server, and process user queries with respect to time and logic constraints as explained

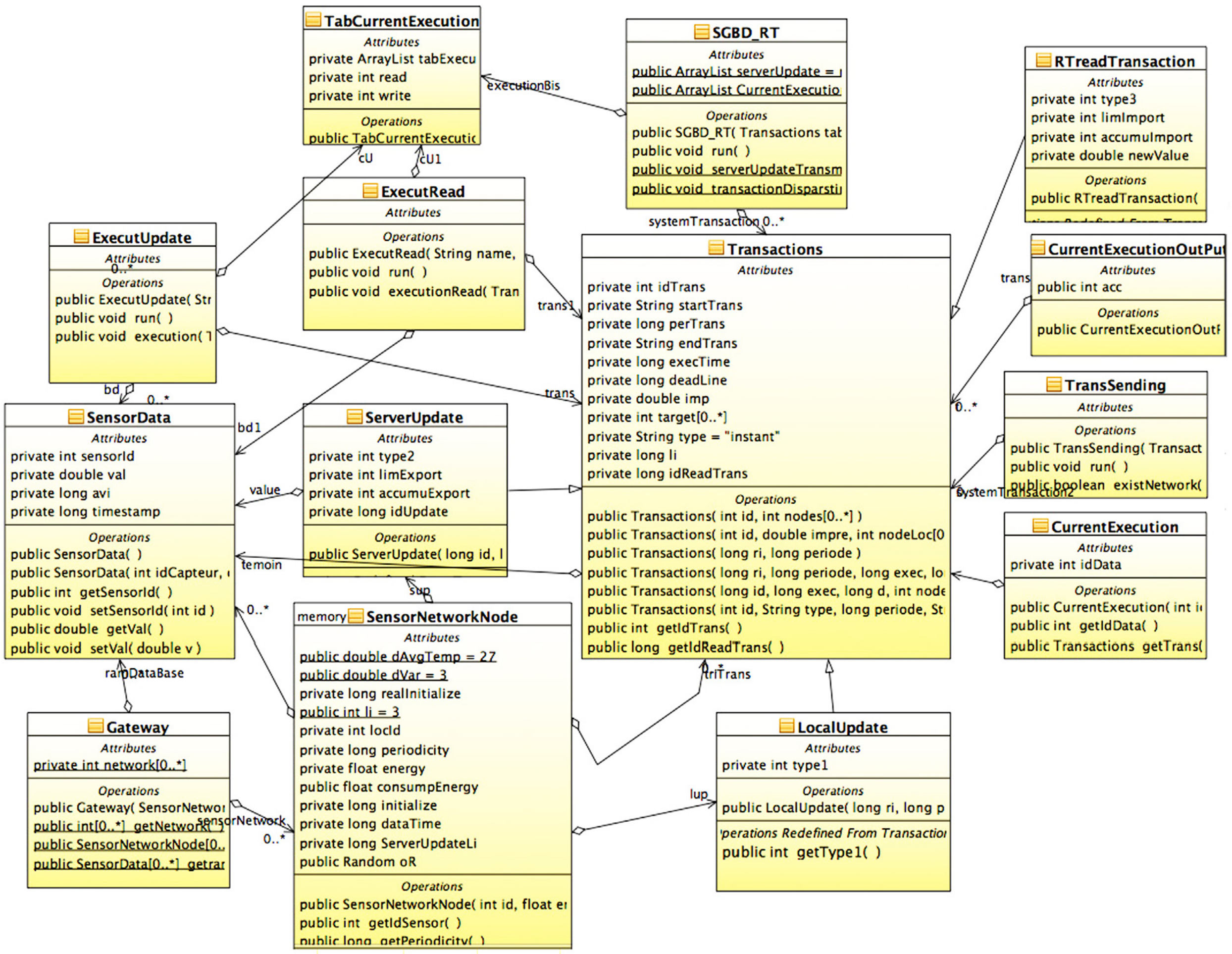

**Fig. 3.** Class diagram of the model.

early. The RT-DBMS receives the real-time update transactions and processes them according to logic and time constraints of data and transactions. Thus, the RT-DBMS schedules the transactions with respect to the Earliest Dead-line First (EDF) protocol (Liu and Leyland, 1973) and uses the Epsilon Serializability criterion to deal with concurrent transactions.

The execution of the simulator is made by setting up some parameters according to a particular experiment. Indeed, the simulator is implemented in Java programming language and in the file containing the *MainSimulation class* one can fix almost all the parameters of the simulation, e.g., the number of sensor nodes composing the network. Each sensor node has to be configurated also in this file by setting up the attributes (identification of the sensor location, the maximum power of the batteries, the parameters of the local and sever updates) of the *SensorNetworkNode* class. In addition, the practitioner has to configure here the maximum number of user transactions (instant query transactions and long-running query transactions) and their characteristics to be sent to the network nodes. The real-time user transactions (*RTreadTransaction* class) that want to access the real-time data storing in the database server has to be configured also in that file. Thus for analysis the validity of the model, five (5) sensors forming the sensor network were considered in the configuration file. Table 2 summarizes the configurations of the local updates (periodic acquisitions) and the server updates (periodic updates of the database server) of each sensor node.

The parameters in the above table are specific to those transactions. The transactions are like the Query Language for

**Table 2**
Summary of the configuration parameters of the sensors.

| | | | | | Local updates | |
|---|---|---|---|---|---|---|
| **Server updates idLocSensor** | **Li In time unit (t.u.)** | **Ct (t.u.)** | **Dl (t.u.)** | **Pr (t.u.)** | **Li (t.u.)** | **Pr (t.u.)** |
| 0 | 3 | 2 | 12 | 12 | 1 | 6 |
| 1 | 9 | 4 | 18 | 18 | 9 | 9 |
| 2 | 4 | 2 | 14 | 14 | 1 | 7 |
| 3 | 5 | 3 | 15 | 15 | 3 | 7 |
| 4 | 2 | 2 | 12 | 12 | 2 | 6 |

Real-Time Database (QL-RTDB) (Chagas et al., 2010) with properties encapsulated in a class as already explained early in the paper. This class, depending of the transaction type, is composed of attributes that indicate the identification of the transaction (*idTrans*), the liberation time (*Li*), the computational time (*Ct*), the periodicity (*Pr*), the deadline (*Dl*), the total amount of inconsistency exported in the database or imported from the database (*accumu-Export/Import*), the imprecision limit (*Lim-Imp/Exp*), and the location identifications (*idLocSensor*) of the temperature sensors targeted by the transaction. The syntax is like: *Transaction* $<$ *idTrans*, *Li*, *Ct*, *Pr*, *Dl*, *accumu-Export/Import*, *Lim-Imp/Exp*, *idLocSensor0..idLocSensorN* $>$.

This experiment considers also two real-time user transactions (rt-uT): a real-time user transaction with an identification 500, a computational time 6 t.u., a deadline 12 t.u., and a real-time user transaction with an identification 600, a computational time 5 t.u., a deadline 12 t.u.. The user transaction with an identification 500 demands to read a data sent by the sensor in location 0 from the real-time database server and the user transaction with an identification 600 wants to read the data item sent by the sensor 1. These transactions are aperiodically sent towards the database server.

According to the Epsilon Serializability criterion, the server update transactions and the real-time user transactions include respectively an attribute that stores a value representing the amount of inconsistency exported in the database and an attribute that stores a value representing the amount of inconsistency imported from the database. The structure of these transactions has also a value that represents the limit of imprecision exported in the database and a value that represents the limit of imprecision imported from the database. For more efficient analysis, these values are generated by a random function in the model.

Depending on the parameters and transaction used, the results are given by numerical values.

### 7.1. *Case where an instant query is sent to the network to query sensor data*

We first analyze the case where an instant query is sent in the network. The properties of these queries are encapsulated in class that is composed of attributes that indicate the identification of the query and the locations of the temperature sensors that the user wishes to query. The syntax is like: *Transaction* $<$*idTrans*, *idLocSensor0..idLocSensorN*$>$. As the case study, the instant query $<$*idTrans*$=$*0*, *idLocSensor0..idLocSensor1*$>$ gives the results in Table 3.

An instant query is a query that is run against a device in an instant of time. Thus, according to Table 3, the query arrives in the network at *Oct 31, 2012 23:17:22* and is immediately answered by the targeted sensors. By using an event-advance desing, the sensor nodes are based on formula (3) and compute all the previous local updates and send all the underlying server updates towards the database server before returning the required data with their temporal parameters. Thus, the simulation model shows also for each sensor the number of local updates and server updates to compute before processing the query. Here, only the sensors in locations 0 and 1 reply because they were targeted by the query. However, the remaining sensors perform their local and server updates.

Once the server updates (sup) arrive in the Gateway, they are handled with the real-time user transactions (rt-uT) by the RT-DBMS with respect to time and logic constraints (see Table 4).

Table 4 presents the sequence of both real-time user transactions and real-time update transactions in the server database. It is also possible to observe the time properties for each transaction, the sensor performing the updates or the sensor data targeted by the user transaction, the parameters to deal with the Epsilon Serialisability, and the data value added or read including its eventual error. The liberation times (*Li*) indicate the executions of the transactions according to their periods. Considering the quality of service management, the model uses the Earliest Deadline First (EDF) protocol (Liu and Leyland, 1973) to schedule the transactions and performs the Epsilon Serializability techniques to deal with concurrent transactions. Formulas (4) and (5) are used here to allow or forbid the simultaneous execution of transactions. Thus as illustrated in the table, the user transaction 500 allows to read the data with an imprecision 0.241%, while the user transaction 600 does not allow any imprecision in the read value.

The model allows also to compute the energy consumed in the network, see Table 5.

Table 5 above presents the percentage of energy consumed and remaining energy for each sensor node according to its activities performed. These activities are mainly data sensing, processing, communication, and the sensor's duty cycle operation. In the table, one can observe that the sensors in locations 0 and 4 have spent most of the energy. This is due to the fact that they perform more server updates (see Table 3), leading then to more data transmission. In fact generally in WSNs and with respect to the TelosB datasheet (see Table 1), the wireless communication consumes more energy than the other activities performed by the sensors. The energy consumed by the sensor 0 is the biggest because, regarding the sensor 4 of which it has the same number of local updates and server updates, it executes the instant query and sends the result to the base station. This leads to more energy consumption. The energy consumption is computed based on formula (2) in (Briand et al., 2010).

### 7.2. *Case where a long-running query is sent to the network to query sensor data*

The configurations of the network and the two real-time user transactions addressed to the database server are not changed. However, we sent a long-running query to trigger the process

**Table 3**
Summary of the results of the instant query that queries the temperature sensors in locations 0 and 1.

| **Instant query with an idTrans**$=$0, **idLocSenso**$=$0.0.1 | | | | | | |
|---|---|---|---|---|---|---|
| **IdLocSensor** | **Access date** | **Returned data (°C)** | **AVI (t. u.)** | **Timestamp** | **Number of local updates (lup)** | **Number of server updates (sup)** |
| 0 | Oct 31, 2012 23:17:22 | 28.032 | 14 | Oct 31, 2012 23:17:18 | 3 | 2 |
| 1 | Oct 31, 2012 23:17:22 | 28.033 | 14 | Oct 31, 2012 23:17:14 | 1 | 1 |
| 2 | X | X | X | X | 3 | 1 |
| 3 | X | X | X | X | 2 | 1 |
| 4 | X | X | X | X | 3 | 2 |

**Table 4**
Summary of the sequence of executions of real-time transactions in the database server.

| **idTrans** | **idLocSen-sor** | **Li** | **Ct (t.u.)** | **Pr (t.u.)** | **Dl (t.u.)** | **LimExp** | **LimImp** | **Value (°C)** |
|---|---|---|---|---|---|---|---|---|
| 2000 (sup) | 4 | Oct 31, 2012 23:17:07 | 2 | 12 | 12 | 2 | X | 21.152 |
| 3000 (sup) | 0 | Oct 31, 2012 23:17:08 | 2 | 12 | 12 | 1 | X | 27.964 |
| 4000 (sup) | 2 | Oct 31, 2012 23:17:09 | 2 | 14 | 14 | 2 | X | 21.914 |
| 5000 (sup) | 3 | Oct 31, 2012 23:17:10 | 3 | 15 | 15 | 4 | X | 26.452 |
| 9000 (sup) | 1 | Oct 31, 2012 23:17:14 | 4 | 18 | 18 | 8 | X | 28.033 |
| 2000 (sup) | 4 | Oct 31, 2012 23:17:19 | 2 | 12 | 12 | 2 | X | 21.151 |
| 3000 (sup) | 0 | Oct 31, 2012 23:17:20 | 2 | 12 | 12 | 1 | X | 28.032 |
| 500 (rt-uT) | 0 | Oct 31, 2012 23:18:45 | 6 | X | 12 | X | 6 | 28.032 $\pm$ 0.241% |
| 600 (rt-uT) | 1 | Oct 31, 2012 23:18:45 | 5 | X | 12 | X | 0 | 28.033 $\pm$ 0.0% |

**Table 5**
Summary of the energy consumed in the network.

| idLocSensor | Consumed energy (%) | Remaining energy (%) |
|---|---|---|
| 0 | 1.64817083 | 98.3518292 |
| 1 | 1.2104375 | 98.7895625 |
| 2 | 0.88565 | 99.11435 |
| 3 | 0.84804375 | 99.1519563 |
| 4 | 1.24817083 | 98.7518292 |

**Table 6**
Summary of the results of the long-running query that queries the temperature sensors in locations 0 and 1.

**Long-running query with: idTrans**=*2*, **initialize date**=“*Sep 03*, *2012 11*:*09*:*34*”, **period**=1 t.u., **idLocSenso**r=0.0.1, **termination date**=**“***Sep 03*, *2012 11*:*09*:*37*”

| idLocSensor | Returned data (°C) | AVI (t. u.) | Timestamp | Number of local updates (lup) | Number of server updates (sup) |
|---|---|---|---|---|---|
| 0 | 25.647 | 14 | Sep 03, 2012 11:09:34 | 4 | 1 |
| | 25.825 | 14 | Sep 03, 2012 11:09:35 | | |
| | 25.596 | 14 | Sep 03, 2012 11:09:36 | | |
| | 25.630 | 14 | Sep 03, 2012 11:09:37 | | |
| 1 | 26.088 | 14 | Sep 03, 2012 11:09:34 | 4 | 1 |
| | 26.024 | 14 | Sep 03, 2012 11:09:35 | | |
| | 26.109 | 14 | Sep 03, 2012 11:09:36 | | |
| | 26.094 | 14 | Sep 03, 2012 11:09:37 | | |
| 2 | X | X | X | 4 | 1 |
| 3 | X | X | X | 4 | 1 |
| 4 | X | X | X | 4 | 1 |

of the network. A long-running query refers to a query runs against a device during a time interval (Bonnet and Seshadri, 2000; Neto et al., 2004). The characteristics of the query are encapsulated in class that is composed of attributes that indicate the identification of the query (*idTrans*), the initiate date of the query execution (*startTrans*), the period of the data sampling (*perTrans*), the terminate date (*endTrans*) of the query execution, and the locations of the temperature sensors (*idLocSensori*) that the query targets. The syntax is: *Transaction* < *idTrans, startTrans, perTrans, idLocSensor0..idLocSensorN, endTrans* >. For the case study, the long-running query parameters are: < *idTrans*=*2*, *startTrans*= “*Sep 03*, *2012 11*:*09*:*34*”, *perTrans*=*1 t.u.*, *idLocSensor*=*0.0.1*, *endTrans*=“*Sep 03*, *2012 11*:*09*:*37*” >. The results are summarized in Table 6.

Table 6 illustrates the execution of the long-running query, which returns the temperature values stored in the sensors 0 and 1. This query samples the temperature values each second from “*Sep 03*, *2012 11*:*09*:*34*” to “*Sep 03*, *2012 11*:*09*:*37*”. Moreover, all the nodes perfoms their local updates and sends the update transactions towards the database server.

**Table 7**
Summary of the energy consumed in the network.

| idLocSensor | Consumed energy (%) | Remaining energy (%) |
|---|---|---|
| 0 | 1.32325625 | 98.6767438 |
| 1 | 1.32325625 | 98.6767438 |
| 2 | 0.92325625 | 99.0767438 |
| 3 | 0.92325625 | 99.0767438 |
| 4 | 0.92325625 | 99.0767438 |

Table 7 shows the summary of the energy consumed by each device according to their activities completed.

## 8. Conclusion and future work

This paper presented a simulation framework for real-time database management on wireless sensor networks. Thus, the description of the different characteristics to design real-time databases on WSN has been done and we proposed a model for a simulation framework of real-time databases on WSN that uses a distributed approach.

The distributed approach was chosen considering its advantages, unlike the warehousing approach. Thus, with the distributed approach the queries could be processed inside the network that diminishes the data flow in the network and increases the lifetime of the network. This is due to the fact that the communication activities use more power than the local process activities. Moreover, by using the distributed approach it is possible to process instant-queries and long-running queries, which are quasi-real-time queries.

In real-time database management on wireless sensor network, the data and transactions have temporal restrictions. For that, the proposed model uses the Earliest Dead-line First (EDF) protocol to schedule transactions and the Epsilon Serialisability techniques to allow conflicting transactions to execute simultaneously in way that their scheduling does not lead an imprecision that is higher than the one accepted in the data. Unlike to usual sensor network simulators oriented network that study the network in the point of view communication or simulators oriented node, which study the functions of a single node, this simulator model emphasizes on the real-time database management techniques for WSNs that use a distributed approach. Thus, this model integrates all the components of a sensor network and the characteristics of real-time database techniques. The full model has been implemented in Java. The object-oriented model was chosen because it is more suitable to model the complex real-world objects with time constraints than the relational model (Kim, 1995). Moreover, the simulator is configurable and the protocols may be changed as required.

A case study shows the execution of real-time transactions and the energy consumed in the network that demonstrates, thus, the validity of the model.

We are currently working on a probabilistic study of real-time transactions with an architecture of real-time database for WSN that uses a distributed approach. This study should allow us to optimize the performance of real-time transaction executions and network resources.

## Acknowledgments

This work has been partially supported by the Instituto de Telecomunicações, Next Generation Networks and Applications Group (NetGNA), Portugal, by National Funding from the FCT

Fundação para a Ciência e a Tecnologia through the PEst-OE/EEI/LA0008/2011 Project, and by the Ministerio de Ciencia e Innovación, Spain, through the Plan Nacional de I+D+i 2008–2011 in the Subprograma de Proyectos de Investigación Fundamental, Project TEC2011-27516.

## References

Adelstein F, Gupta SKS, Richard III GG, Schwiebert L. Fundamentals of mobile and pervasive computing. McGraw-Hill Companies, Inc.: NY, New York; 2005 (ISBN 0-07-141237-9).

Akyildiz FI, Melodia T, Chowdhury RK. A survey on wireless multimedia sensor networks. Journal of Computer Networks (Elsevier) 2007;51(4):921–60.

Akyildiz FI, Sankarasubramaniam Y, Cayirci E. Wireless sensor networks: a survey. Computer Networks (Elsevier) 2002;38:393–422.

Al-karaki JJ, Kamal AE. Routing techniques in wireless sensor networks: a survey. IEEE Wireless Communications 2004;11(6):6–28.

Arasu A, Babcock B, Babu S, Datar M, Ito K, Motwani R. STREAM: The Stanford stream data manager. IEEE Data Engineering Bulletin (Citeseer) 2003;26(1):19–26.

Baronti P, Pillai P, Chook CVW, Chessa S, Gotta A, Hu FY. Wireless sensor networks: a survey on the state of the art and the 802.15. 4 and ZigBee standards. Journal of Computer Communications (Elsevier) 2007;30(7):1655–95.

Bonnet Ph, Seshadri P. Device database systems. In: Proceedings of the international conference on data engineering ICDE'99. San Diego, CA: Published by the IEEE Computer Society; 2000.

Bonnet Ph, Gehrke J, Seshadri P. Querying the physical world. IEEE personal communications. Special issue ìnetworking the physical world; October 2000.

Bonnet Philippe, Gehrke J, Seshadri P. Towards sensor database systems. In: Proceedings of the second international conference on mobile data management MDM '01. London, UK: Springer-Verlag; 2001. p. 3–14.

Bouazizi E, Sadez B, Duvallet C. Applicabilité du critère d'epsilon-sérialisabilité dans les SGBD temps réel. Special issue ISDM MCSEA; 2004.

Boulis Athanassios. Demo abstract: Castalia: revealing pitfalls in designing distributed algorithms in WSN. In: Proceedings of SenSys '07; November 2007.

Boulis Athanassios. Castalia, a simulator for wireless sensor networks and body area networks. Version 2.0, user's manual; May 2009. [retrieved 27.06.09].

Briand R, Arrijuria O, Dupé V, Richon C, Terrasson G. Evaluation de la consommation énergétique d'un nœud: du hard au soft. RESSACS 2010; 2010.

Callaway EH. Wireless sensor networks: architectures and protocols Auerbach Publications. NY, New York, USA: CRC Press; 2004. (ISBN: 0849318238).

Chagas Lunas Dantas, Lima Eduardo Pereira, Neto Pedro Fernandes Ribeiro. Real-time databases techniques in wireless sensor networks. In: Proceedings of the sixth international conference on networking and services, IEEE; 2010. p. 182–7.

Chen J, DeWit JD, Tian F, Wang Y. NiagaraCQ: A scalable continuous query system for internet databases. In: Proceedings of the 2000 ACM SIGMOD international conference on management of data. New York: ACM Press; 2000. p. 379–90.

Chen J, Ferreira L. Human motion tracking using wireless sensor networks. Technical report HURRAY-TR-090407. Available from: www.cister.isep.ipp.pt; 2009.

Cho S, Shih E, Ickes I, Min R. Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks. ACM/IEEE MOBICOM.ACM; 2001. p. 272–287.

Cortes C, Fisher K, Pregibon D, Rogers A. A language for extracting signatures from data streams. In: Proceedings of the sixth ACM SIGKDD international conference on knowledge discovery and data mining. New York: ACM Press; 2000. p 9–17.

Cranor C, Gao Y, Johnson T, Shkapenyuk V, Spatscheck O. Gigascope: high performance network monitoring with an SQL interface.Proceedings of the 2002 ACM SIGMOD international conference on Management of data. New York: ACM Press; 2002. p. 623.

CrossBow Inc., TelosB (TPR2400) product datasheet. Available from: ⟨http://www.willow.co.uk/TelosB_Datasheet.pdf⟩. [retrieved 03.07.13].

Deshpande Amol, Guestrin Carlos, Madden R Samuel, Hellerstein M Joseph, Hong Wei. Model-driven data acquisition in sensor networks. In: Proceedings of the thirtieth international conference on very large data bases. Toronto, Canada; August 31–September 03 2004. p. 588–99.

Diallo O, Rodrigues JJPC, Sene M. Real-time data management on wireless sensor networks: a survey. Journal of Network and Computer Applications (Elsevier) 2011;35:1013–21.

DiPippo LC, Wolfe VF. Real-time databases. Journal of Database Systems Handbook, Multiscience Press (Citeseer); 1997.

Egea-López E, Vale-Alonso J, Martínez-Sala AS, Pavón- Mariño P, García-Haro J. simulation tools for wireless sensor networks. In: Proceedings of summer simulation multiconference—SPECTS 2005; 2005.

Granat J. A framework for event based modeling and analysis. Journal of Telecommunications and Information Technology 2006;4:88–90.

Idoudi N, Duvallet C, Sadeg B, Bouaziz R, Gargouri F. How to model a real-time database? In: Proceedings of the 12th IEEE international symposium on object-oriented real-time distributed computing (IEEE ISORC'2009). Tokyo, Japan: IEEE; 2009a. p. 321–5.

Idoudi N, Louati N, Duvallet C, Bouaziz R, Sadeg B, Gargouri F. A framework to model real-time databases. International Journal of Computing and Information Sciences 2009b;7:1–11.

Karir M, Polley J, Blazakis D, McGee J, Rusk D, Baras JS. ATEMU: a fine-grained sensor network simulator. In Proceedings of the 1st IEEE international conference on sensor ad hoc communication networks; 2004.

Kim W. Object-oriented database systems: promises, reality, and future. Moderne database systems. Addison Wesly; 1995. p. 255–280.

Levis P, Lee N, Welsh M, Culler D. TOSSIM: accurate and scalable simulation of entire TinyOS applications. In: Proceedings of the 1st ACM Conference on embedded networked sensor systems; 2003. p. 126–37.

Li C, Zhang H, Hao B, Li J. A survey on routing protocols for large-scale wireless sensor networks. Sensors 2011;11(4):3498–526.

Lin K, Chen M, Zeadally S, Rodrigues Joel JPC. Balancing energy consumption with mobile agents in wireless sensor networks. Future Generation Computer Systems, (Elsevier, ISSN: 0167-739X) 2012;28(2):446–56, http://dx.doi.org/10.1016/j.future.2011a.03.001.

Lin K, Rodrigues Joel JPC, Ge H, Xiong N, Liang X. Energy efficiency QoS assurance routing in wireless multimedia sensor networks. IEEE Systems Journal, IEEE, (ISSN: 1932-8184) 2011;5(4):495–505, http://dx.doi.org/10.1109/JSYST.2011.2165599.

Liu C, Leyland J. Scheduling algorithms for multiprogramming in hard real-time environment. Journal of the ACM 1973;20(1):46–61.

Madden S, Franklin MJ, Hellerstein JM, Hong W. TAG: a tiny aggregation service for ad-hoc sensor networks. In: Proceedings of OSDI. Citeseer; 2002.

Madden SR, Franklin MJ, Hellerstein JM, Hong W. TinyDB: an aquisitional query processing system for sensor networks. ACM Transactions on Database Systems (TODS) (ACM) 2005;30(1):122–73.

Mainwaring A, Culler D, Polastre J, Szewczyk R, Anderson J. Wireless sensor networks for habitat monitoring. In: Proceedings of the 1st ACM international workshop on wireless sensor networks and applications. New York: ACM Press; 2002. p. 88–97.

Mendes LDP, Rodrigues JJPC. A survey on cross-layer solutions for wireless sensor networks. Journal of Network and Computer Applications (Elsevier) 2010;34(2):523–34.

Nasreddine N. Conception et modélisation d'un émulateur d'un réseau de capteurs communicant sans fil; 2012.

Neto Pedro Fernandes R, Perkusich Maria LB, Perkusich Angelo. Real-time database for sensor networks. In: Proceedings of the 6th international conference on enterprise information systems; 2004. p. 599–603.

Nguyen AH, Foerster A, Puccinelli D, Giordano S. Sensor node lifetime: an experimental study. In: Proceedings of the 7th IEEE international workshop on sensor networks and systems for pervasive computing; 2011. p. 146–51.

Oliveira LM Luís, de Sousa Amaro F, Rodrigues Joel JPC. Routing and Mobility approaches in IPv6 over LoWPAN mesh networks. International Journal of Communication Systems, Wiley (ISSN: 1074-5351), 2011a;24(11):1445–66, http://dx.doi.org/10.1002/dac.1228.

Oliveira LMLuís, Rodrigues Joel JPC. Wireless sensor networks: a survey on environmental monitoring. Journal of Communications (JCM), Academy Publisher, (ISSN: 1796–2021), 2011b;6(2):143–51, http://dx.doi.org/10.4304/jcm.6.2.143-151.

Pailler S. Analyse hors ligne d'ordonnançabilité d' applications temps réel comportant des tâches conditionnelles et sporadiques 2006:21–30.

Park S, Savvides A, Srivastava MB. Sensorsim: a simulation framework for sensor networks. In: Proceedings of the 3rd ACM international workshop on modeling, analysis and simulation of wireless and mobile systems, ser. MSWIM '00. New York, NY, USA: ACM; 2000. p. 104–11.

Pu C. Generalized transactions processing with epsilon serializability. In: Proceedings of 4th international workshop on high performance transactions systems. Asimolar, California; 1991.

Ramamritham K. Real-time databases. Journal of Distributed and Parallel Databases (Springer) 1993;1(2):199–226.

Ruiz LB, et al. Architectures for wireless sensor networks. In: Proceedings of the 22nd brazilian symposium on computer networks SBRC'04. Gramado, RS, Brazil: [s.n.]; 2004. p. 167–218. Tutorial. [ISBN: 85-88442-82-5] [in Portuguese].

Sacks L., et al. The development of a robust, autonomous sensor network platform for environmental monitoring. Sensors and their applications XXII (Citeseer); 2003.

Silva FA, Braga TRM, Ruiz LB. Nogueira JMS. Tecnologia de Nós Sensores Sem Fio. Contrôle & Instrumentação 92; Agosto 2004:76–87.

Simon G, Igyesi P Vo, Maroti M, Ledeczi A. Simulation based optimization of communication protocols for large scale wireless sensor networks. Proceedings of IEEE 2003 aerospace conference, vol. 3; 2003. p. 1339–46.

Sobeih A, Hou JC, Kung Lu-Chuan, Li Ning, Zhang Honghai, Chen Wei-Peng, et al. J-Sim: a simulation and emulation environment for wireless sensor networks. IEEE Wireless Communications 2006;13(4):104–19.

Sundani H, Li H, Devabhaktuniv K, Alam M, Bhattacharya P. Wireless sensor network simulators: a survey and comparisons. International Journal Of Computer Networks (IJCN) 2010;2(5):249–65.

Sundresh S, Wooyoung K, Gul A. SENS: a sensor, environment and network simulator. In: Proceedings of the 37th annual simulation symposium; 2004. p. 221–8.

Terrier F, Lanusse A, Bras D, Roux P, Vanuxeem P. Concurrent object for multitasking. L'objet, v3. Paris; 1997. p. 179–96.

TinyOS, an open-source operating system for wireless embedded sensor networks [online]. Available from: ⟨http://www.tinyos.net/⟩. [retrieved: 01.11.12].

Ulmer C, Yalamanchili S, Alkalai L. Wireless distributed sensor networks for in-situ exploration of mars. Work in progress for NASA technical report (Citeseer); 2003.

Varga A. The omnet++ discrete event simulation system. In: Proceedings of the european simulation multiconference (ESM); June 2001.

Wolfe VF, Prichard JJ, Dipippo LC, Black J. Real-time database system: issues and applications. The RTSORAC real-time-object-oriented data base prototype. Kluwer Academic Publishers; 1997. p. 279–301.

Xian Xiaodong., Shi W, Huang H. Comparison of OMNET++ and other simulator for WSN simulation. In: Proceedings of the 3rd IEEE conference on industrial electronics and applications, ICIEA 2008; 2008. p. 1439–43.

Zhang Junguo, Li W, Cui D, Zhao X, Yin Z. The NS2-based simulation and research on wireless sensor network route protocol. In: Proceedings of the 5th international conference on wireless communication, networking and mobile computing, WiCom'09; 2009. p. 1–4.

Zhu Y, Shasha D. Statstream: statistical monitoring of thousands of data streams in real time. In: Proceedings of the 28th international conference on very large data bases, VLDB Endowment; 2002. p. 358–69.