2011

An Integrated environment for data acquisition with dynamic changes in wireless sensor networks

Albert Nurgaliev

An Integrated Environment for Data Acquisition with Dynamic Changes in Wireless Sensor Networks

Albert Nurgaliev
M. S. Computer Science Thesis
Department of Computer Science
Rochester Institute of Technology
Rochester, New York

Submitted in partial fulfillment of the requirements of a
Master of Science Degree in Computer Science

COMMITTEE

CHAIR: Professor Leon Reznik DATE

READER: Associate Professor Minseok Kwon DATE

OBSERVER: Professor Hans-Peter Bischof DATE

May 27, 2011

Abstract

The wireless sensor network (WSN) is an important technology with a wide variety of diverse applications in such domains as healthcare, military forces and environmental monitoring.

Our research aims at developing methods and tools capable of addressing WSN problems such as energy constraint, low memory, and computation capability of a sensor node by implementing a new WSN design concept, improving existing and developing new protocols. Our research goal is to develop novel generic methodologies supporting a higher level of design flexibility and possible architectural optimization against multiple criteria such as the quality of data (QoD), quality of service (QoS), and lifetime extension. Application requirements may vary in terms of abovementioned parameters and consequently there is no single platform that can be applied to all domains. Moreover, current methods do not provide opportunities for dynamic changes of either protocols or their parameters, which might improve WSN agility and survivability in a harsh environment. This problem can be solved by integrating various protocols at different layers within a single framework and supporting their dynamic selection in order to adapt the network to varying application requirements.

This thesis develops a mechanism which facilitates structural design and implementation of an Integrated Environment for Data Acquisition with Dynamic Changes (IEDADC). It features adaptation and integration of protocols, protocol switching and automatic or manual selection as well as the implementation of quality assurance and localization techniques.

The design methodology is tested by implementing a SN prototype consisting of a base station and sensor nodes. Sun Small Programmable Object Technology is used as a hardware basis for this work. The software has been developed in Java programming language including the host and sensor nodes' applications.

The conducted experiments have confirmed the higher level of design flexibility and optimization of the following criteria: energy consumption, QoD and QoS.

Table of Contents

List of figures

1. Introduction

A typical wireless sensor network (WSN) is composed from two kinds of nodes [3] (fig. 1.1). The first is the full-function device (FFD), which serves as a coordinator of the network. An example to FFD can be a base station, it can also be called a sink node. The second kind is the reduced-function device (RFD), a simple device with little resource requirements and computational capabilities. A sensor node is an RFD which is also called a mote.

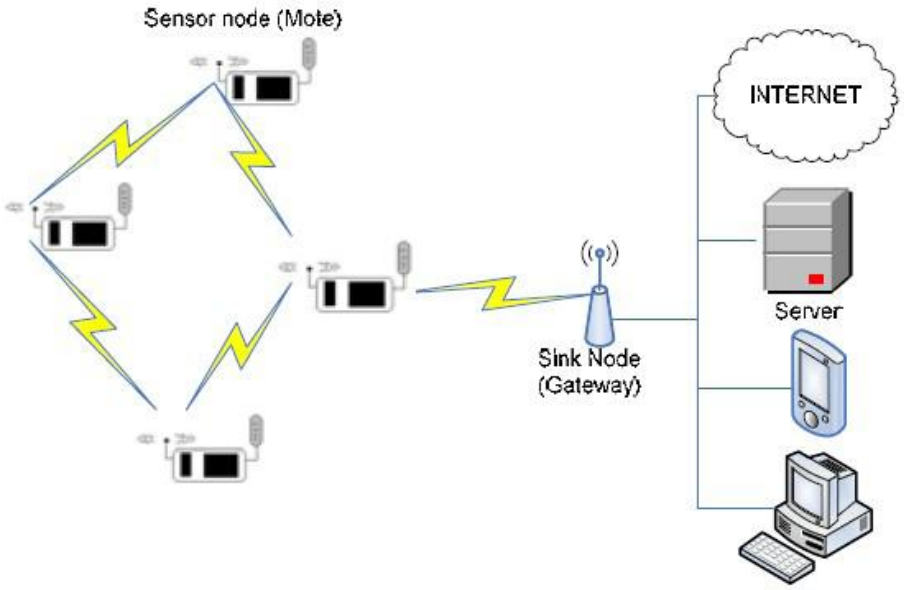


Fig. 1.1. *An example of a Wireless Sensor Network*[1].

There are four main components in a sensor node: a sensing unit, a processing unit, a communication unit, and power supply [20] [22] (fig. 1.2).

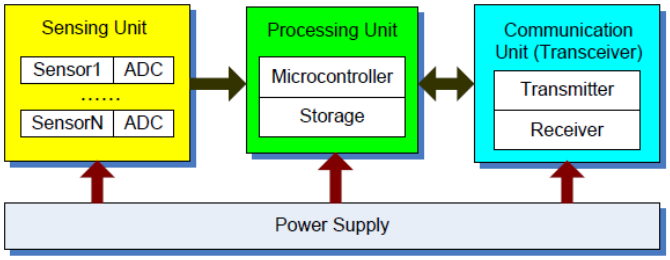


Fig. 1.2. *A wireless sensor node's architecture*[2].

[1] The figure is taken from [7].
[2] The figure is taken from [2].
[3] See the JUNG2 2.0 API at http://jung.sourceforge.net/doc/api/index.html for details.
[4] The picture is taken from http://blogs.sun.com/roger/entry/location_location_location_radio
[5] It may seem confusing, because previously it was mentioned that the Radiostream is also capable of sending

A sensing unit can be composed of several sensors, which monitor various environmental characteristics such as temperature, humidity, pressure or light. The processing unit includes a small storage in a form of flash memory. It is responsible for performing tasks, processing data and controlling certain functionalities of a sensor node. A communication unit is represented by a transceiver, which provides receiving and transmitting of data and wireless communication between nodes. The main power supply for a sensor node is a battery. The energy constraint is the main problem related to efficient application of a WSN. While energy is consumed for sensing, data processing and communication, the former task consumes the majority of energy recourses [20] [22].

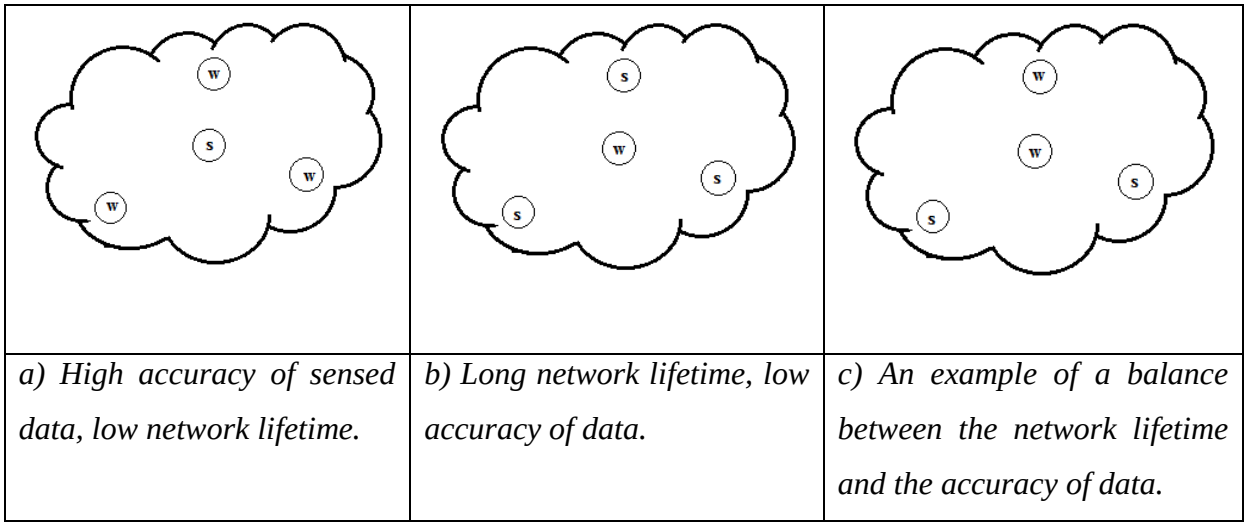
Our research aims at developing methods and tools capable of addressing typical WSN problems such as energy constraint, low memory and computation capability of a sensor node by implementing a new WSN design concept, as well as improving existing and developing new protocols. Our research goal is to develop novel generic methodologies supporting a higher level of design flexibility and possible architectural optimization against multiple criteria such as the quality of data (QoD), quality of service (QoS), and lifetime extension. Application requirements may vary in terms of abovementioned parameters and consequently there is no single platform that can be applied to all domains. Moreover, current methods do not provide opportunities for dynamic changes of either protocols or their parameters, which might improve WSN agility and survivability in a harsh environment. Therefore this problem can be solved by integrating various protocols at different layers within a single framework and supporting their dynamic selection in order to adapt the network to varying application requirements.

Although many efficient protocols have been proposed for WSNs [9-11] [21-22] there is still an open research problem related to cross-layer optimization. Optimizing and analyzing protocols can improve system performance and determine their benefits and limitations, but WSN can also benefit from cross-layered approach. For example, depending on the application, a higher degree of coverage may be required to increase the accuracy of the sensed data. However, the sensor network lifetime depends on the number of active nodes and connectivity of the network, so energy must be used efficiently in order to maximize the

[2] The figure is taken from [2].

network lifetime (fig. 1.3). In this case, providing an application layer service on top of the protocol stack can efficiently resolve the problem of network lifetime extension.

This research mainly concentrates on development and implementation of a design methodology which determines, regulates and possibly optimizes the following parameters: network lifetime, cross-layer integration, QoS, data aggregation and localization.



a) High accuracy of sensed data, low network lifetime.	*b) Long network lifetime, low accuracy of data.*	*c) An example of a balance between the network lifetime and the accuracy of data.*

Fig. 1.3. *The relationship between the network lifetime and the accuracy of data in a WSN (w – working node, s – sleeping node).*

Developing an efficiently operating platform for WSNs is an important task. Agility mechanisms must be incorporated in such a way that they are transparent to users and must introduce minimal functional and performance impacts. These techniques must be employed in combination with a system-level architecture that links these mechanisms together, to ensure optimal effectiveness and proper management.

This thesis develops a mechanism which facilitates structural design and implementation of an Integrated Environment for Data Acquisition with Dynamic Changes (IEDADC). It features adaptation and integration of protocols, protocol switching and selection either automatically or manually as well as some other features such as implementation of quality assurance and localization techniques.

The design methodology is tested by implementing a SN prototype consisting of a base station and sensor nodes. Sun SPOT (Small Programmable Object Technology) is used as a hardware basis for this work. The software has been developed in Java programming language including the host and sensor nodes' applications.

The organization of this paper is as following. Section 2 provides the analysis of the protocols used in this work. Section 3 discusses the related work. Section 4 states the hypothesis for the thesis. Section 5 introduces the design architecture and discusses the implementation of the optimization techniques used in this work. Section 6 presents the experiments performed using IEDADC and their results. Section 7 describes the results of this research and gives conclusion for this work.

2. Background

According to the TCP/IP model of network architecture there are five layers:

Application
Transport
Network
Data Link
Physical

A WSN also borrows this network architecture [22]. However, the sensor network constraints mentioned previously pushed the researchers to develop more energy efficient standards and protocols than the popular ad-hoc protocols. These standards take into account low computational capabilities of sensor nodes, the number of elements present in the network, which is relatively higher than in ad-hoc networks, and their limited power supply. As a result of these efforts the IEEE 802.15.4-2006 standard has been developed. This standard specifies the physical and MAC layers for low rate WPANs, with the main emphasis made on low-cost, low-speed communication between devices [3]. The standard does not define higher level layers, so additional specifications exist offering protocols on the higher layers. These protocols / standards provide additional integral solutions, such as quality of service, quality of data and topology control. Examples of such additional standards are ZigBee alliance [24], TinyOS [17] and others. Sun SPOT is not a member of ZigBee alliance, and IEEE 802.15.4 is the only standard implemented in this technology. This allows development of a network stack on top of the IEEE 802.15.4 standard. Development of numerous protocols [7] [9-10] [22] demonstrates the fact that there exist many application requirements for WSNs. Therefore integrating various protocols at different layers within a single framework and supporting their dynamic selection would allow for adapting the network to varying application requirements.

In order to address the issue of extending the network lifetime in a harsh working environment, the implementation of PEAS (Probing Environment and Adaptive Sleeping) [21] protocol is presented in this work. Customization of its parameters affects the network lifetime, network connectivity and the QoD. Two other protocols are provided as the part of the radio communication library of the Sun SPOT SDK. This work employs all three protocols to develop methods for dynamic switching of protocols or altering their network parameters to fit varying application requirements.

Each node in PEAS protocol can be in three different states: sleeping, probing and active. All nodes initially start with the sleeping state, from which they periodically enter a probing state. During this state the sensor will determine whether to activate and provide a coverage of sensing area or go back to sleep (Fig. 2.1).

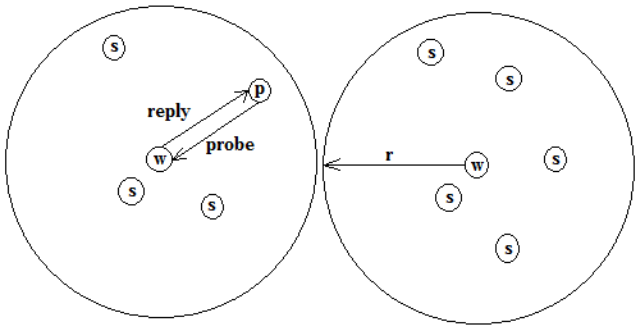


Fig. 2.1. *Schematic representation of a WSN working under the PEAS protocol; w – working sensor, s – sleeping sensors, p – probing sensor, r – sensing range of a working sensor.*

Probing time needs to be tuned carefully to preserve a balance between energy saving and robustness. If it is too long, a node failure would be unnoticed for a long time, on the other hand a probing period that is too short may result in wasted power. Also in this protocol, the wake-up intervals are spread over time for two particular reasons: preventing extended sensor outage, and avoiding collisions. For example, if a node fails, the sensing area might be uncovered for a long time, so a wake-up interval is needed to detect the outage. In case of many sensor nodes waking up at the same time, collisions can be mitigated by properly setting the wake-up intervals. The following figures show the wake-up intervals when using deterministic sleeping time (Fig. 2.2) and wake up intervals, which are spread over time (Fig. 2.3) [21].

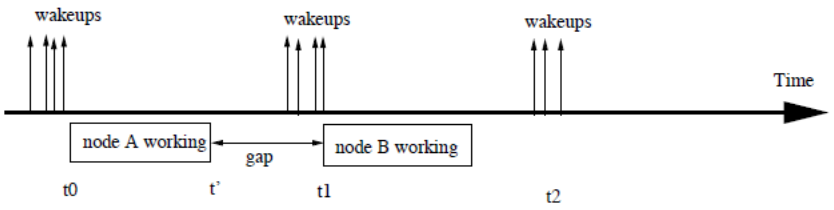


Fig. 2.2. *Wake-up intervals for deterministic sleeping time. Long gaps between two wake-ups, a lot of sensors wake-up at the same time.*

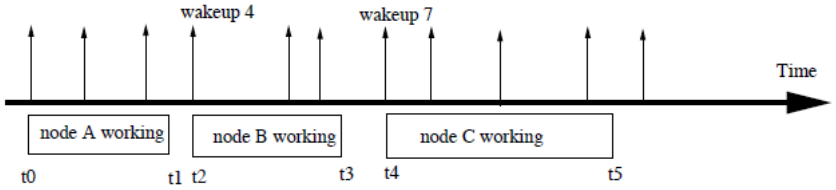


Fig. 2.3. *Spreading wake-up times throughout the time. Short gaps between two wake-ups and no collisions happen when one sensors wakes-up.*

Moreover, the PEAS protocol dynamically adjusts the sleeping time, so that node failures do not affect the overall spread of waking times. This protocol has been modified to support dynamic customization of the following parameters: the number of working nodes, the sleeping time duration and the probing range. Setting up the parameters of this protocol can influence the overall operation of the network.

The number of working sensors (fig. 2.4) parameter directly affects the network lifetime duration and the quality of data (QoD) for a certain region. Setting this parameter to one would maximize the network lifetime, minimizing the accuracy of the received data. Increasing the number of working sensors would lower the network lifetime, on the other hand increasing the QoD.

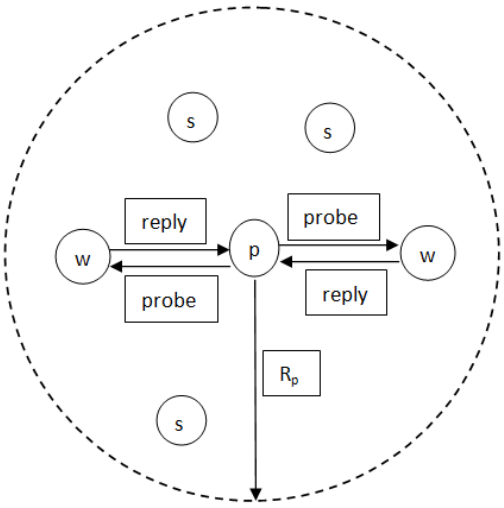


Fig. 2.4. *Schematic overview of a network with the degree of coverage set to two; where w – working sensor; s – sleeping sensor; p – probing sensor; R_p – probing range.*

The second parameter, the sleeping time duration, affects the frequency at which probing is performed by a sensor node. This influences both working and sleeping nodes' battery discharge levels and how fast the node failures are determined. The battery discharge level is affected when a probing node sends a probing message and waits for a reply from a working node(s) for a certain amount of time, while the working node spends its energy by receiving the probing message and sending back the reply. So the higher the frequency of entering the probing state, the higher is the energy overhead from using the PEAS protocol. On the other hand, setting the sleeping time duration higher would result in longer periods when a failure of a working sensor node would stay unnoticed. Balancing between the energy overhead and node failure notification is application specific and setting the sleeping time duration parameter should be considered individually for every case.

The last parameter, the probing range, affects the number of sensors that will enter the working state in the network. This is not to confuse with the first parameter, which determines the number of working sensors per one area. The probing range parameter determines the number of such areas and their size. In the implementation of the protocol we used a threshold filtering rule regarding the received signal strength [21] as a probing range parameter. This implies that a sensor will react to a message only if its signal strength of a message is greater than the threshold value (fig. 2.5.x).

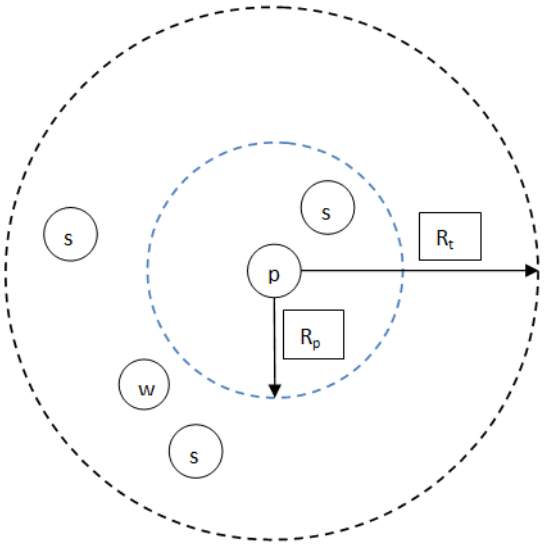


Fig. 2.5.1. *Schematic overview of a WSN, with a low probing range. It results in entering the probing node p to the working state. R_p – probing range; R_t – transmission range; w – working sensor node; s – sleeping sensor node; p – probing sensor node.*

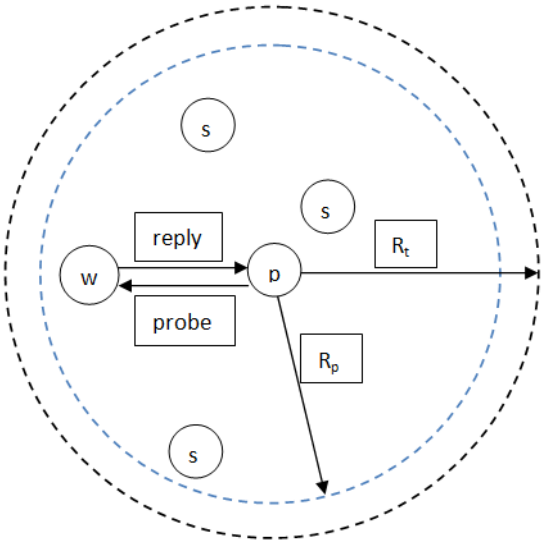


Fig. 2.5.2. *Schematic overview of a WSN, with a high probing range. It results in having more sleeping nodes around the working node. R_p – probing range; R_t – transmission range; w – working sensor node; s – sleeping sensor node; p – probing sensor node.*

The second protocol is called Radiogram. It is a part of a Sun SPOT SDK radio communication library. It provides datagram based communication between devices and there are no guarantees that the datagram will be delivered; also datagrams can be delivered out of order or delivered more than once [13]. As a result of this, the Radiogram protocol is unreliable. The protocol supports several modes: client-server, broadcast and point-to-point. When opening the client-server connection, the server side can receive any radiograms sent on a given port whether they are broadcast packets or packets sent specifically to this sensor. Broadcast mode delivers radiograms to all listeners with no delivery guarantees. Point-to-point mode creates connection between two peers only. Along with the payload every radiogram automatically carries, from one node to another, such information as the signal strength (measured in dB), the link quality (in %), the timestamp, etc.

The third is the Radiostream protocol, which provides a socket-like peer-to-peer radio protocol with reliable, buffered stream-based IO between two devices [13]. A connection using this protocol is reliable, since the delivery of packets is confirmed by sending ACKs back to the source. Every packet also carries some additional information about the connection as the previous protocol.

3. Related Work

This section discusses the existing methodologies proposed to address the issues of sensor nodes' localization, a framework design development, the QoS and the QoD optimizations.

3.1 Localization methodologies

Location awareness is an open research problem in a WSN. Precise location information may be unavailable due to the energy constraints peculiar to sensor nodes. The use of GPS for sensor localization is extremely energy-consuming, which makes it impractical for installation on sensor nodes. However the knowledge of sensor nodes' location can be very useful in such applications as tracking an object, route discovery and avoiding sensing coverage overlap [18]. A direction-based localization scheme (DLS) [18] determines a direction of a sensor node relative to the sink node. The main idea of DLS is the spatial locality property, which allows sensor nodes to determine their directions according to the packets received. The direction of a node is represented in a form of a gray code.

In localization techniques the nodes whose location is initially known are called anchors. Messages sent by anchors are used by other nodes in the network to calculate their relative direction. Different strategies are used in [18] to improve estimated correctness of the localization. Overall, the results of the simulations presented in [18] demonstrate the accuracy of DLS at around 85%.

In the current work, a localization algorithm is also introduced as a part of the QA mechanism of the framework. This algorithm calculates the approximate signal strength level between two sensor nodes. A helper SPOT is referred to as an anchor in this work. The selection of a helper SPOT is done during the localization process, while anchors in [18] are selected statically and their location is known in advance. The main idea of our localization algorithm is the capability to measure an approximate signal strength value between two nodes by means of triangulation.

Section 5.2 describes the details of our localization algorithm; meanwhile experiment results demonstrating the correctness of the algorithm and nodes' signal strength distribution over the distance are shown in section 6.

3.2 A framework designing methodologies

The application developed in [6] analyzes the number of neural network (NN) architectures and signal waveforms to detect novelties in a radio transmission coming from wireless sensors. In order to increase chances to detect novelties in signals, the application has different parameters to configure in its GUI, such as NN architecture, its learning rate, novelty distance, threshold, etc. The application includes twenty-six NN architectures, which a user can select from the GUI. Based on the choice of NN and its settings, the application detects novelties in the signal and notifies the user though email or alarm sound. In order to determine which NN architecture returns the least error with the signal, the user has to manually test the performance of each NN on the same signal and compare the results. However, the novel contribution done in this work is the ability to simulate signals, as well as to experiment with NN (either custom made or already implemented in MATLAB). Interaction with the program is done through a user-friendly GUI. This contribution opens new ways for developing and testing NN, serving as a testing field for new NN architectures. The approaches used in developing the application's front-end in [6] were used as the basis for the development of the IEDADC's GUI.

The work in [16] introduces the prototype framework for automated data quality assessment. In order to understand the problems related to QA consider a situation when three sensors are measuring the temperature in a room. Apparently, when one of the sensors starts sending different temperature information than the other two sensors, it seems suspicious, because the temperature in a room is most likely to stay constant over the area of the room. QA mechanisms do not try to determine the reasons which stand behind such misbehavior. Instead the data is assigned a green, a yellow or a red flag. The color of a flag has a certain meaning: the green flag is a good data, the red flag is a bad data and the yellow flag is the quality in-between. It is important to note that the red flag does not necessarily mean that the data is wrong, but instead it indicates that this data and the area from which it was sensed requires an immediate attention. These approaches are taken as a basis for the development of the QA aspects in IEDADC.

An Interactive and Extensible Framework for Execution and Monitoring of Wireless Sensor Networks [4] adds extra distributed services on top of Directed Diffusion protocol to gain some additional performance on the protocol execution. Although their work was built

on Directed Diffusion, the framework is extensible to be used by any other network protocol and distributed services related to the protocol.

The framework in [4] is written in C++ versus Java used in current work. Also, in [4] Sensoria WinNG sensor devices are used in the research versus Sun SPOTs in the current work. The current work also focuses on optimization of multiple criteria, while the framework in [4] puts the emphasis on the network protocol performance only, i.e. average network traffic per node. In addition, the current work investigates the ways to automate some processes in the framework, based on the data incoming from the sensor nodes. This would provide a certain level of intelligence to the system, aiming to reduce the workload from a user.

Another framework presented in [1] introduces the design and implementation of MAPS (Mobile Agent Platform for Sun SPOT). It is also a Java-based framework for wireless sensor networks based on Sun SPOT technology, like the framework presented in the current work. The authors of the paper utilize an agent-oriented programming paradigm on WSN applications. Novel contributions of this paper to the WSN programming research area are:

- Introducing a platform that allows an effective Java-based development of agent-based applications;
- Investigation of the Sun SPOT technology for supporting mobile agent-based applications and systems.

The introduced paradigm has already shown its effectiveness and efficiency in IP-based highly dynamic distributed environments [1], hence the authors of the paper integrate these paradigms into the MAPS framework to achieve same results in WSN applications.

The experiments in the healthcare domain, which involved real-time monitoring of human activities, demonstrated the effectiveness and the suitability of the agent-based approach to program WSN applications. However, the MAPS performance benchmarking demonstrated that the serialization and transportation of agents are very time and energy consuming operations [1].

IEDADC intends to harness effectiveness and efficiency of WSN application through dynamic selection and customization of the protocol parameters, compared to the implementation of the agent-based paradigm in the MAPS framework.

In [5] authors of the paper incorporate the concept of supply chain into a WSN. They propose a hybrid data dissemination framework based on the supply chain (SC) methodology. According to the proposed methodology, a supply chain is a series of links between suppliers and consumers, which involve such activities as the acquisition of raw materials, manufacturing, and delivery of produced goods to the end consumers [5].

This concept has been projected to a WSN model. As a result the network is partitioned into functional regions, each responsible for a certain task reflecting the SC methodology (fig. 3.1). The authors of the paper use a habitat monitoring example to illustrate their scheme.

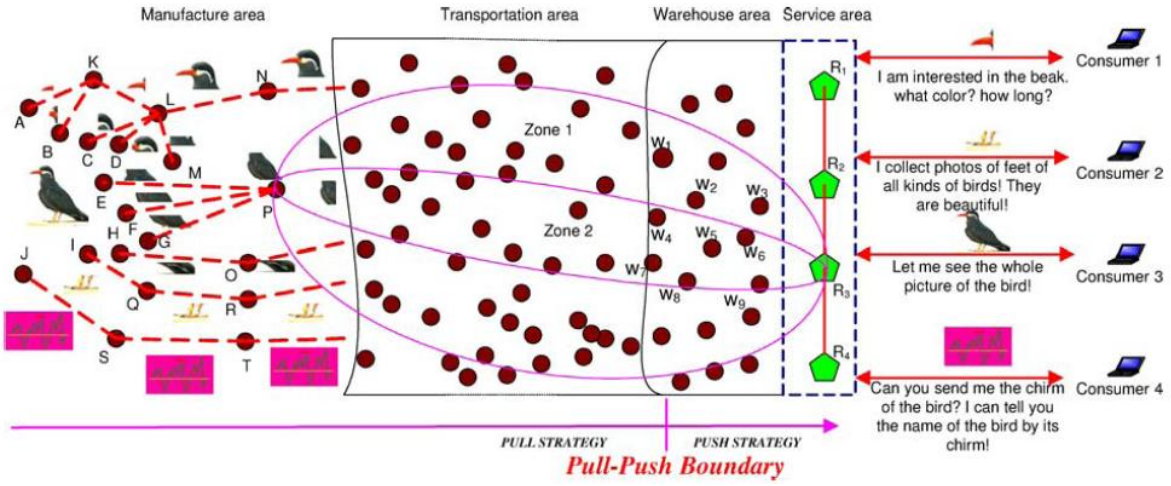


Fig. 3.1. *System architecture for habitat monitoring [5].*

Nodes in each area are responsible for different tasks. Sensor nodes in a manufacturing area, for example, are responsible for data acquisition and aggregation. Transportation area is responsible for delivery of data to the sink nodes. Warehouse area is responsible for providing a temporary storage of data until it has been requested by a sink node. Service area is where the sink nodes are located.

This kind of diversification of nodes' tasks provides such features as flexibility and scalability of WSN applications as well as improved reliability and energy efficiency of the network.

The framework developed in [5] uses various routing protocols to different regions in order to provide better performance in terms of reliability and efficient energy usage. This contrasts with the current work, which uses a single protocol for the entire network. However, same goals are achieved by IEDADC through support of dynamic protocol selection and

customization of its parameters. Both of the frameworks are designed to have the ability to extend their functionality by adding other WSN protocols. This feature provides the scalability of a framework and improves its flexibility for various applications.

Implementation of the SC methodology requires deployment of a comparatively large number of sensors to maintain functionality of the areas introduced in [5]. Therefore SC methodology is not applicable in a sparsely deployed network or a network with a small number of sensor nodes.

3.3 QoS optimization methodologies

The discussion below will define the QoS model used in this work as well as provide alternative approaches in defining a QoS model. Despite the fact that this parameter is always application specific, it is usually defined by the following categories: reliability, timeliness and throughput [19] [23]. Reliability implies certain level of a guarantee to deliver data from source to destination. The higher the guarantee to deliver data, the more reliable the protocol is. In terms of protocols, sending acknowledgements for each received packet is a way to make the protocol reliable. Radiostream is an example of a reliable protocol for WSN. The second parameter is a timeliness of a data delivery, which means the data should be delivered within a certain time limit. The throughput is specified as a bandwidth. It is the effective number of data, transported within a certain period of time [19]. If all of these parameters are satisfied then in this work we can state that the protocol provides a high QoS.

In order to achieve a high QoS it is always necessary to add some redundancy to the protocol. Therefore minimizing energy consumption and maximizing QoS are two typically conflicting requirements [19]. Indeed, acknowledgements (ACKs) used by Radiostream protocol make it reliable. However, sending ACKs consumes energy, which is vital for sensors. In general, addition of any kind of redundancy increases the sensors' workload resulting in increasing the QoS and trading-off the network lifetime. The problem of keeping both of these parameters at a high level becomes one of the major problems investigated in WSN. For this reason multiple alternative approaches addressing this issue exist [2] [7] [9] [12]. For example, an Adaptive Fault-tolerant QoS Control (AFTQC) algorithm [2] uses several delivery paths m_p instead of using ACKs to ensure data delivery (Fig. 3.2).

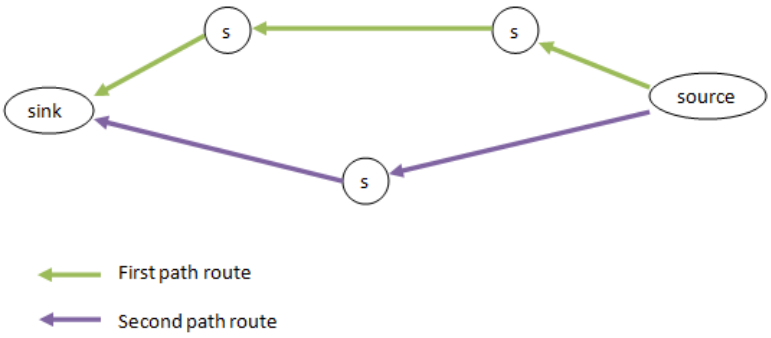


Fig. 3.2. *An example of a single-source multiple-path WSN.*

Determining an optimal redundancy level to achieve a high QoS and long network lifetime is one of the problems investigated by the authors of the paper. According to the results of the simulations, it appears that the value of m_p set to three is the optimal parameter. Also they demonstrate that in comparison with the traditional "acknowledgement and retransmission on timeout" mechanism, AFTQC algorithm performs better in terms of the network lifetime under various number of redundant paths and sources (m_s) [2] (Fig. 3.3).

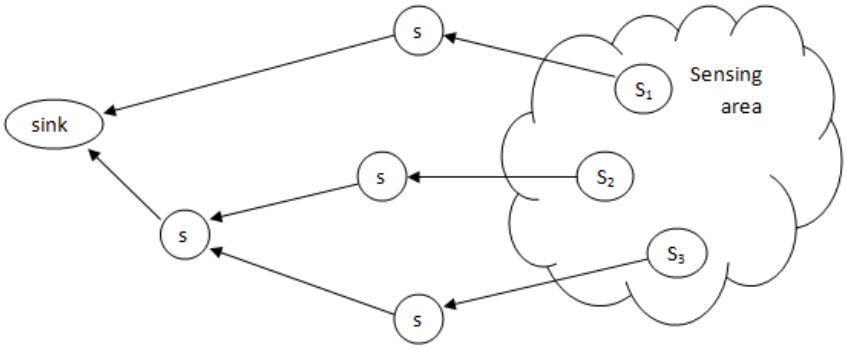


Fig. 3.3. *An example of a multiple-source single-path WSN. S_1, S_2, S_3 is the first, the second and the third sources respectively.*

The algorithm used in Sun SPOT network to find a path between source and sink is called Link Quality Routing Protocol (LQRP) [13], which is used by default and its functioning is transparent for a user. The user can only gain some level of control in routing by setting a routing policy for each sensor through system properties or programmatically [13]. IEDADC uses only one path to deliver data in comparison with multiple paths used by AFTQC. Also, our framework uses ACKs, while AFTQC uses redundant paths to guarantee data delivery. Even though the use of such traditional mechanism of providing good QoS as

sending ACKs is less energy efficient according to the simulation results from [2], it provides a basis for further developments of more complex algorithms for high QoS. Furthermore, in this work we will show that the use of additional service, such as PEAS protocol can increase the network lifetime, keeping the QoS on a desired level.

The protocol, called Event-to-Sink Reliable Transport (ESRT) [11], measures the QoS level according to the number of received packets at the sink node and notifications of congestion in the network. The network in this protocol can operate in one of five states: No congestion, low reliability (NC,LR), The optimal operating region (OOR), No congestion, high reliability (NC,HR), Congestion, high reliability (C,HR), Congestion, low reliability (C,LR). The idea here is that the sink node tries to control the sensors' reporting rate so that the network is in the OOR. It is achieved by broadcasting a revised reporting rate to the sensors (Fig. 3.4). The reliability is determined by the number of packets the sink node receives, while the congestion is detected by the routing nodes, which send notifications to the base station.

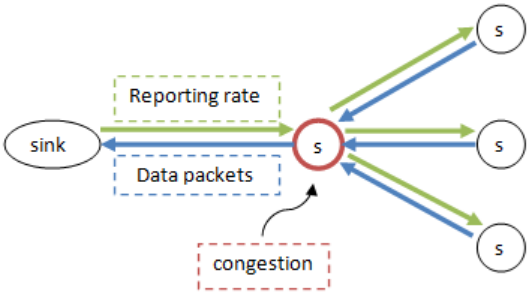


Fig. 3.4. *ESRT operation overview.*

3.4 QoD optimization methodologies

Alternative approaches exist for clustering and data aggregation in WSNs [2]. Data aggregation and clustering are mainly used to save energy by reducing the number of sensors sending information to the sink node. It also solves the problem of QA, because measurements from different sensors after aggregation and processing increase the accuracy of the sensed data.

The authors of the paper [2] consider a cluster-based WSN to increase the effectiveness of the energy use in the network. The AFTQC's clustering algorithm creates a cluster head (CH) which is responsible for managing the network within a cluster, i.e. send responses to the processing center and aggregate data (Fig. 3.5).

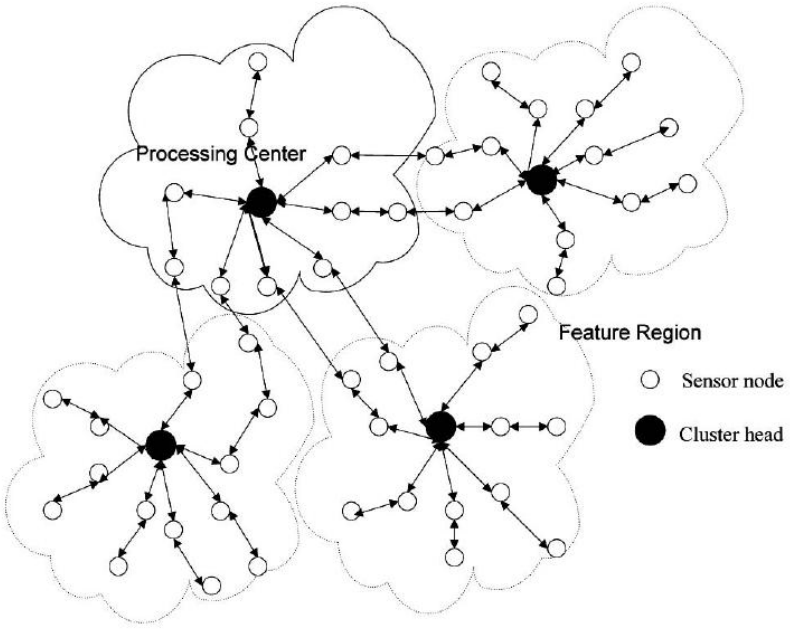


Fig. 3.5. *Cluster-based WSN architecture [2].*

This algorithm is executed periodically to make every sensor in the cluster a CH and therefore the workload is spread evenly among nodes.

The clustering algorithm implemented in IEDADC is provided as a part of the QA mechanism. It is implemented on the host application (the Model module of the framework). There are no cluster heads present in WSN and all data is sent to the base station for processing. This approach increases the accuracy of the data. In cases when this parameter is not prioritized, the degree of coverage can be dynamically reduced. The reason standing behind clustering on a host application rather than keeping CH is that a CH may fail for some reason in a harsh environment. It would result in leaving some part of the network without coverage until another clustering is executed. In the worst case the CH can fail right away, after the clustering. In that case the area will be uncovered for some period T_c, where T_c is the interval between two clustering algorithm executions. In our case even if a node fails in the network, the data is still sent by other sensor nodes in the region. In case PEAS protocol is enabled and the working sensor fails, the sleeping time duration of every sensor is generated in such a way that this failure shall be discovered within a short period of time and this failed node is replaced by a new working sensor node [21].

The source redundancy refers to the use of multiple sensor nodes to report sensing data. The authors of [2] add redundancy to achieve higher QoS – as more sensors respond to a

request, the higher the chance that data will be delivered before timeout. In this work the source redundancy is implemented as a mechanism of QA – as more sensors send data, the higher is the accuracy of the obtained results.

As we can see, there are multiple approaches for solving one issue. All of them may have advantages and disadvantages compared to each other. Also the same approach can serve as a solution for different purposes. Therefore integration of various algorithms within a single framework would make the interaction with a WSN efficient, effective and easy to use for many applications.

4. Hypothesis

This work introduces the design methodology and implementation of IEDADC that features adaptation and integration of protocols, protocol switching and automatic or manual selection as well as other features such as an implementation of quality assurance and localization techniques. Its GUI provides a user with such information about sensor nodes' state as battery level, signal strength, the temperature in the region, etc. The framework operates with Sun SPOT sensor nodes and is implemented entirely in a Java programming language.

Based on the proposed features of the developed environment we state the hypothesis for the thesis: The proposed design methodology of a WSN brings higher level of design flexibility and possible optimization for multiple criteria, minimizing energy consumption under the conditions of supporting the specified levels of QoD and QoS. QoD is assured by the QA mechanisms employed in this framework such as clustering, data processing and localization.

IEDADC is employed to conduct an empirical study to confirm the hypothesis.

The empirical study includes:

1) Measuring the average energy consumption of sensor nodes in the network under various network loads with and without using the PEAS protocol;
2) Measuring the accuracy of the localization algorithm;
3) The examination of the clustering algorithm and aggregation functions' operation.

In the following research experiments, two extreme cases are tested to evaluate the network lifetime duration. First case involves all sensors. This case defeats the purpose of

using the PEAS protocol, so this service is disabled and sensors use forwarding protocol to send their data (comparative analysis of two forwarding protocols based on their energy use is provided). The second case, on contrary, involves only one working sensor, providing the k-coverage parameter set to one. Also different forwarding protocols are used in combination with the PEAS protocol to evaluate its performance (in terms of network lifetime) under different network loads.

The results from these two experiments will demonstrate us the best and the worst cases for energy consumption of sensor nodes in a WSN. According to these results a user can customize the parameters of the protocol to achieve other goals, such as better QA. Further discussion of the effects from varying each of the PEAS protocol parameters will help to maintain the changing application demands in a WSN.

The accuracy of the localization algorithm is determined by comparing the calculated signal strength value to the expected value in that region. The experiment involves placing groups of sensor nodes at various distances from each other. The calculated signal strength value is considered to be accurate if its value does not deviate from the signal strength value of a sensor node located in the same group.

The aggregation functions' operation is evaluated by examining the calculated temperature values in a cluster against the real temperature values of the sensors. The operation of the clustering algorithm is demonstrated by placing one of the sensor nodes in a cluster into a colder environment than other nodes and observing the automatic changes made in clustering of nodes. Section 6 describes these moments in greater details.

5. Design architecture and implementation

Based on the problem specification, an Integrated Environment for Data Acquisition with Dynamic Changes (IEDADC) has been developed. This framework includes two protocols provided by Software Development Kit (SDK) and one protocol described in section 2, as well as the implementation of a QA and localization techniques. Sun SPOT (Small Programmable Object Technology) [15] is used as a hardware basis for this work. IEDADC is implemented using Java programming language.

5.1 The framework architecture

Generally, the architecture of the framework has to be simple and scalable. It must be simple to use from the user or a developer point of view. This means that the GUI must support simple user interactions with the WSN and human readable real-time networking information. It must be scalable, so that new features can be added (protocols on different network layers, plug-ins) to the framework with no or minor modification of the architecture [4].

The architecture of the framework is composed of four modules: view, model, proxy and plug-in. The schematic view of this architecture is shown below:

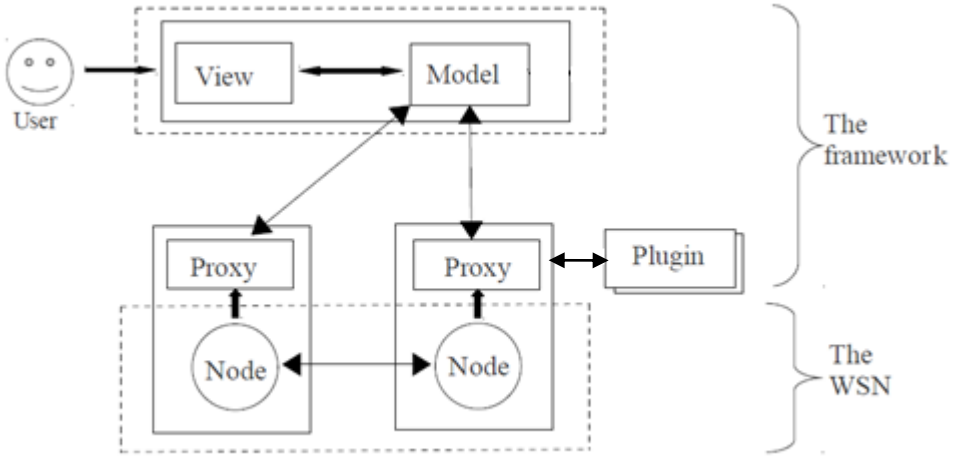


Fig. 5.1. *The framework architecture.*

5.1.1 View

The View module is responsible for user interaction, control and visualization of a WSN architecture and operation. It provides the following information: the network structure in a form of a table and a graph, where each node is a sensor and edges are the links between sensors; the state of each sensor, such as the sensor's signal strength, the quality of data. This module also supplies the user with more analytical, text- or table-based information about the WSN for its further analysis.

Below is the detailed description of the front-end.

The first tab is “Protocol information” (Fig. 5.2).

On top left corner is the “Address” label, which displays the address of the base station, below is the PAN ID, Radio power and Radio channel used by the nodes. In order to

have the base station start listening to the sensor nodes for connection, the user needs to press the “Connect” button. Before connecting, it is possible to choose the protocol that will be used from the dropdown box, next to the Connect button. Label below shows the protocol currently in use. “Switch protocol” button allows switching current protocol used to another one selected from the dropdown box. “Radio policy” button allows changing the policy used to control the radio antenna of sensors (see the developer’s guide for more details of how the Radio policy affects the sensors’ behavior). The Enable service button allows selecting a distributed service from the dropdown box in order to enhance some parameters of the current protocol. In the current implementation, this is the PEAS protocol.

The right column contains the table of currently connected sensors. First column of the table shows the address of the sensor, the second displays the status of the sensor. The status can be either connected or not. The third column shows the state of a sensor when running PEAS protocol. It is either “working” or “sleeping”. As it has been mentioned, there are three states in PEAS protocol: sleeping, probing and working, but since importance of the probing state is not so high, this state is not displayed in the View. In terms of implementation, this implies the sensor notifies the base station only if it is about to enter working state; if there was no such notification, by default the state is considered as “sleeping”. It is clear when the status of a node is “Connected”, however, some elaboration is needed regarding status “Not connected”. When the sensor node first connects to the base station its address is displayed at this table and will never be removed from it (only after restart of the framework). Its status can be changed to “Not connected” when the base station losses the connection to that sensor for a different reason: either a node runs out of the battery, or it fails.

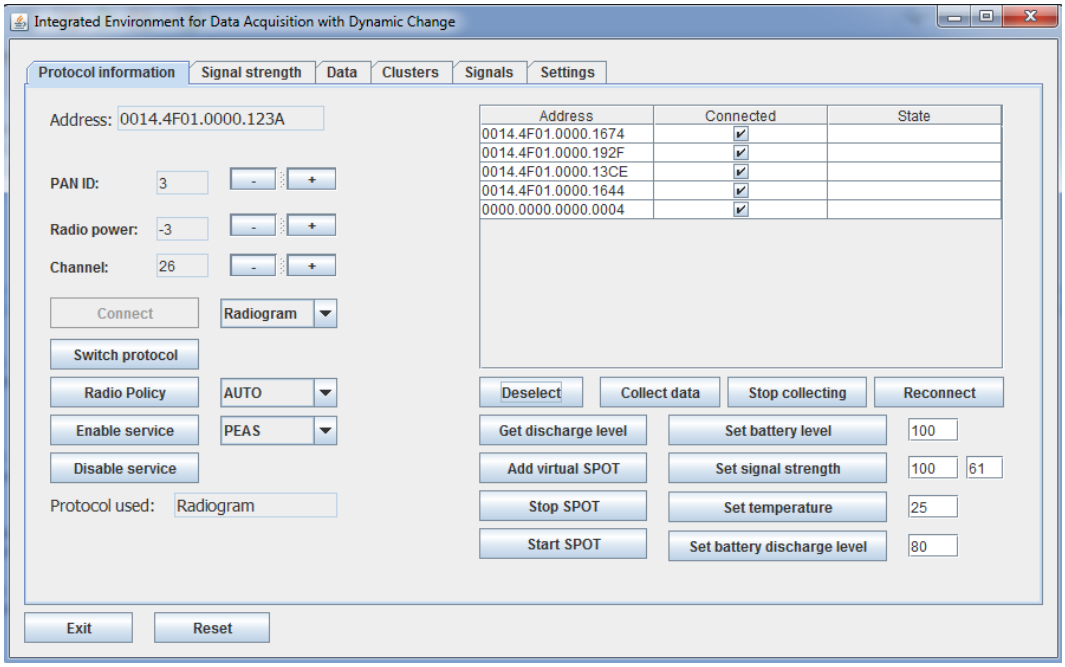


Fig. 5.2. *The "Protocol information" tab of the framework.*

Several operations can be performed with the sensor nodes using this table. The button "Collect data" can work in two modes: if the address of the sensor node is selected from the table the command to collect the sensing the data is sent to this particular sensor. However, if no sensor node has been selected, this command is sent to all of the nodes. "Deselect" button allows clearing the selection of the table. "Reconnect" button sends the command to the sensor node (its address is selected from table) to reestablish the connection to the base station. The same command, but sent to every node is done using the "Reset" button on the bottom left. Current version supports these commands only with the Radiogram protocol.

The next set of buttons provides IEDADC with the emulator capabilities. The "Add virtual SPOT" button creates a virtual sensor with default values of the signal strength to the base station and helper SPOT – these are 100 to the base station and 61 to the helper SPOT (on a 1 to 121 scale), temperature (t^0C) and battery level as percentage. Two text fields next to the "Set signal strength" button set the signal strength to the base station (first field) and to the helper SPOT (the second field). Corresponding buttons can change all these values. When creating a virtual sensor, the address is generated according to the size of the sensor nodes table. For example, if there are four entries in the table and a user decides to create a virtual SPOT the address of the spot will be: "0000.0000.0000.0005". This kind of address makes it

obvious that this SPOT is virtual, however apart from that the user doesn't see any difference in the behavior of a real and virtual SPOTs.

Last two buttons: Start/Stop SPOT – allows manual opening/closing connection from the base station to the SPOT.

The last button "Exit" closes the application, but prior to closing it informs the sensors about it, so that they can start sending advertisements about their availability for further connection.

The second tab is "Signal strength" (Fig. 5.3).

The host application supports variety of Over-The-Air (OTA) commands, which can be sent from a base station [13]. "Hello" command is one of them. It is used to discover the sensors in the radio range of the base station (more information on the OTA commands can be found in the Sun SPOT developer's guide). It is possible to get a number of parameters about current state of a sensor using this command. The upper table shows the link quality and strength from a sensor node to the base station, while the bottom one shows the same parameters, only from the base station to a sensor node. The battery level of a sensor is shown as a percentage. Link Quality is uniquely determined by the CORR value: CORR is a correlation measure provided directly by the radio part. Link Quality values range from 100% (the best, corresponding to a CORR value greater than 100) to 0% (the worst, which means the CORR value is less than 50) [13]. The signal strength value ranges from 1 (weakest) to 121 (strongest). In order to convert this value to dB a user needs to subtract 106 from it.

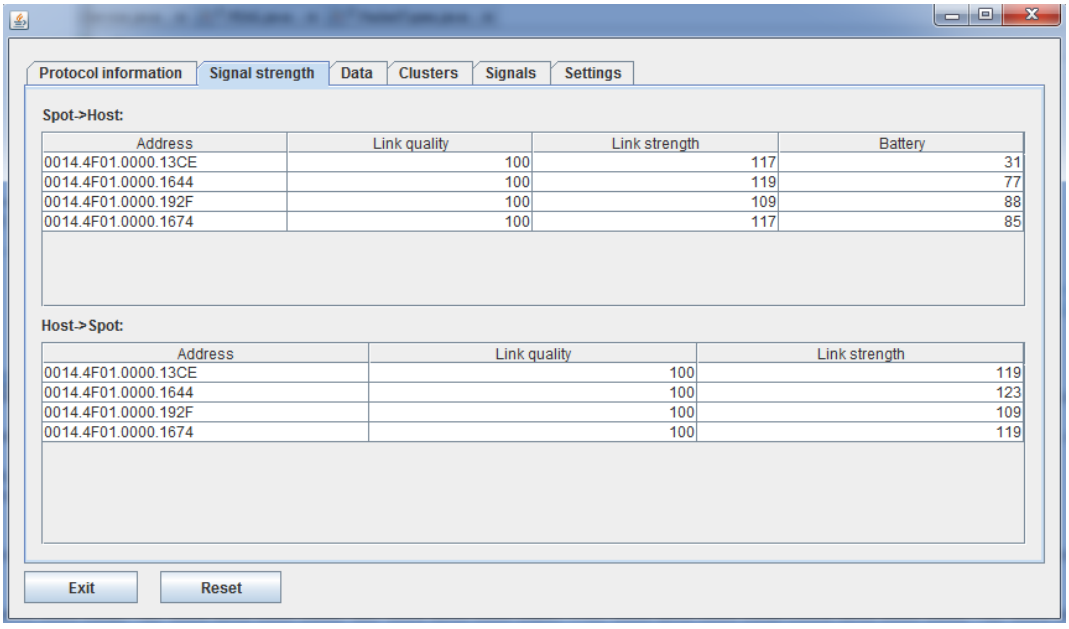


Fig. 5.3. *The "Signal strength" tab of the framework.*

The third tab is “Data” (Fig. 5.4).

The table present in this tab consists of sensing data information: the address of a sensor, date of last measurement and the sensing data itself – in this case it is temperature in Celsius.

The sample period represents the time interval (in milliseconds) between two successive temperature measurements. Also when using Radiogram protocol, the datagram fits six samples, thus in order to increase efficiency of energy use and to increase the quality of data (six measurements are better than one) the datagram will not be sent until it is full. Therefore, even though the sample period is one value, when using Radiogram protocol, the entry in the table will be updated six times slower. For example, if the sample period is set to one second, the measurements will arrive every six seconds.

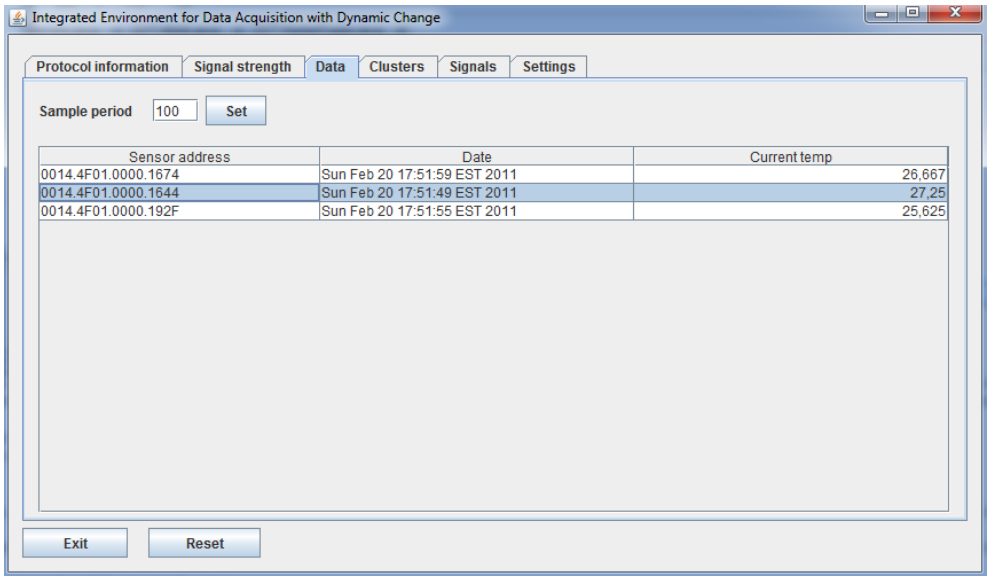


Fig. 5.4. *The “Data” tab of the framework.*

This feature is not supported when using Radiostream protocol, because this protocol opens a stream connection between two peers, instead of sending datagrams of certain fixed size. Virtual sensors possess the same behavior as real sensors, so this is true for both real and virtual sensors.

The next tab is “Clusters” (Fig. 5.5).

The table in this tab represents clusters to which sensors belong to. First column shows the cluster ID, the second column gives the elements of the cluster and the third one shows the

temperature within the cluster, calculated with a certain aggregation function. The aggregation functions implemented in the current version of IEDADC are the average and vote functions. The value of the calculated temperature in a cluster is displayed when the sensors start sending their data to the base station.

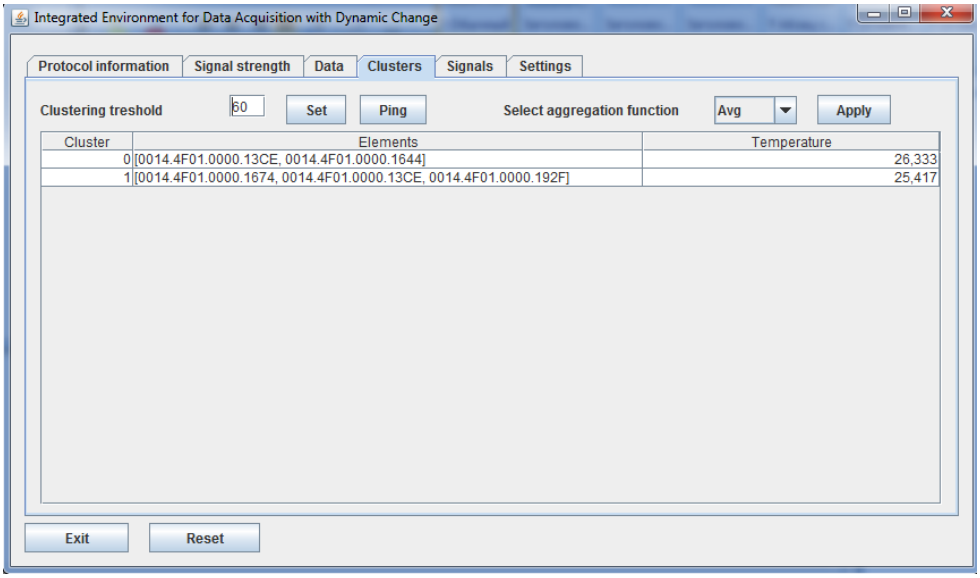


Fig. 5.5. *The "Clusters" tab of the framework.*

The elements are put in a cluster according to the clustering threshold (measured in dB), which can be customized at the top left of the panel. One element can be in several clusters (details on how the clustering algorithm works can be found in section 5.2). Also the aggregation function can be selected from the dropdown box on the right.

In IEDADC the "Ping" button has broader capabilities than sending a Hello command OTA. Additionally, it calculates signal strengths between sensor nodes, clusters them and draws a graph view of nodes according to the signal strengths between them (Fig. 5.6.1). It also plots a graph of average current being drawn from the batteries (mA) with respect to the time in minutes (Fig. 5.6.2). Section 5.2 provides details about this process.

The graph layout in this framework implements the Fruchterman-Reingold force-directed algorithm for node layout[3]. Among the parameters that affect the graph layout are the attraction and repulsion multipliers. First multiplier determines how much the edges try to keep their vertices together and the second multiplier determines how much the vertices try to

[3] See the JUNG2 2.0 API at http://jung.sourceforge.net/doc/api/index.html for details.

push each other apart. The graph represents the network where signal strength between nodes is the weight of an edge in the graph and vertices are the sensor nodes.

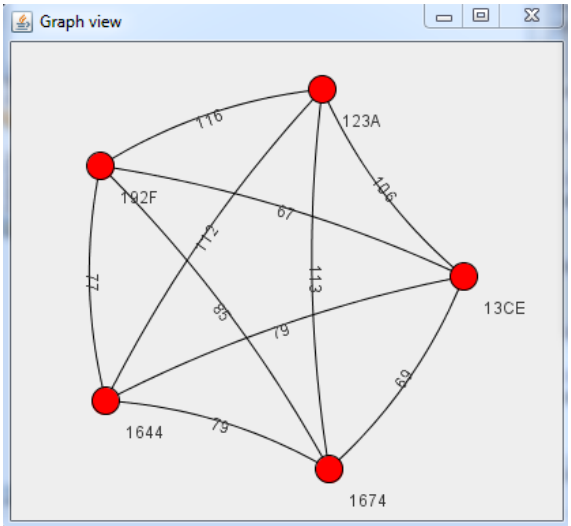


Fig. 5.6.1. *Graph view of the network. Vertices have labels of last four characters of a sensor node's address. Weights of edges are the signal strengths between two nodes.*

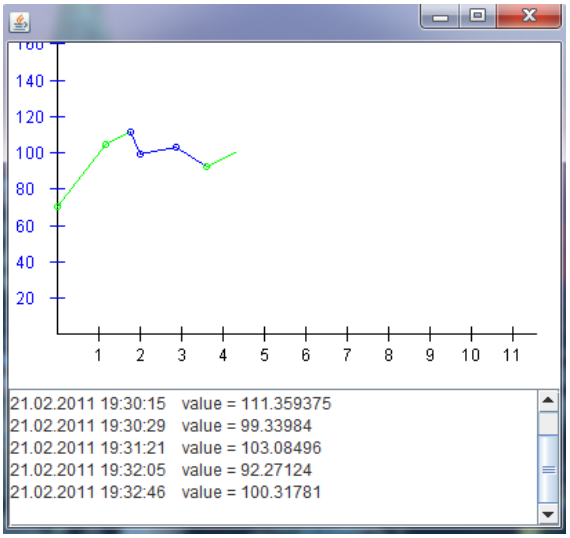


Fig. 5.6.2. *A graph of average energy use measured in mA (y-axis) over time measured in minutes (x-axis). Green line represents the use of the Radiogram protocol, blue line represents the usage of Radiostream protocol.*

The "Signals" tab (Fig. 5.7) shows the signal strengths between nodes. The table structure can be described in the following way: left column shows the node in point A and right column shows the node in point B with the signal strength between these nodes. The blank value in the left column means that the node in point A is the same. For example, in the figure below, the node with the last four characters of its address being 1674 has signal strength to node 13CE as 75, node 192F as 82 and node 1644 as 82.

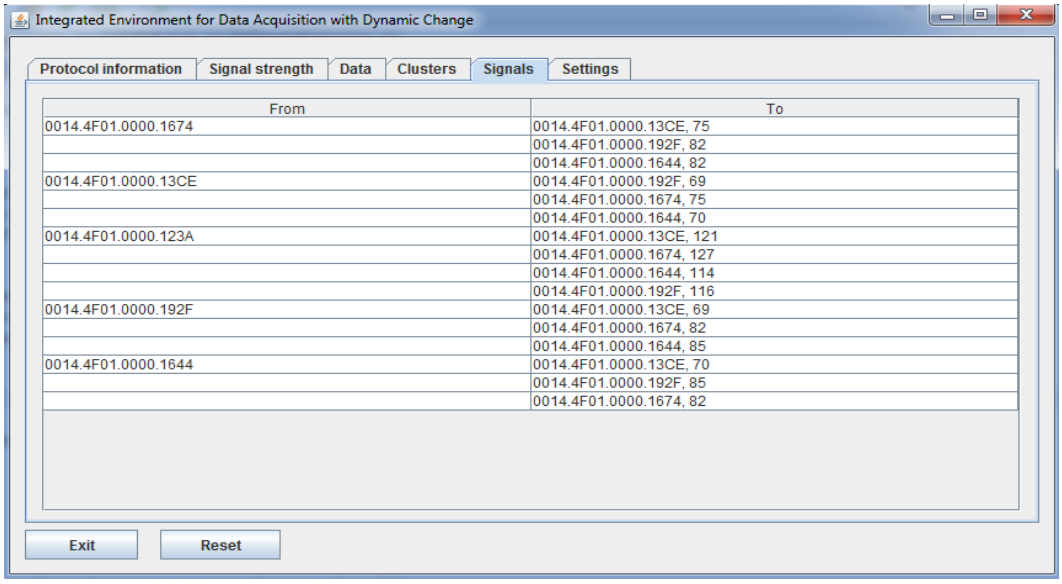


Fig. 5.7. *The "Signals" tab of the framework.*

The last tab, called "Settings" (Fig. 5.8) allows dynamic customization of the protocol settings and switching protocols. A user may also define settings for triggering actions automatically such as protocol switching, the period of network state information collection or the condition for enabling the PEAS protocol.

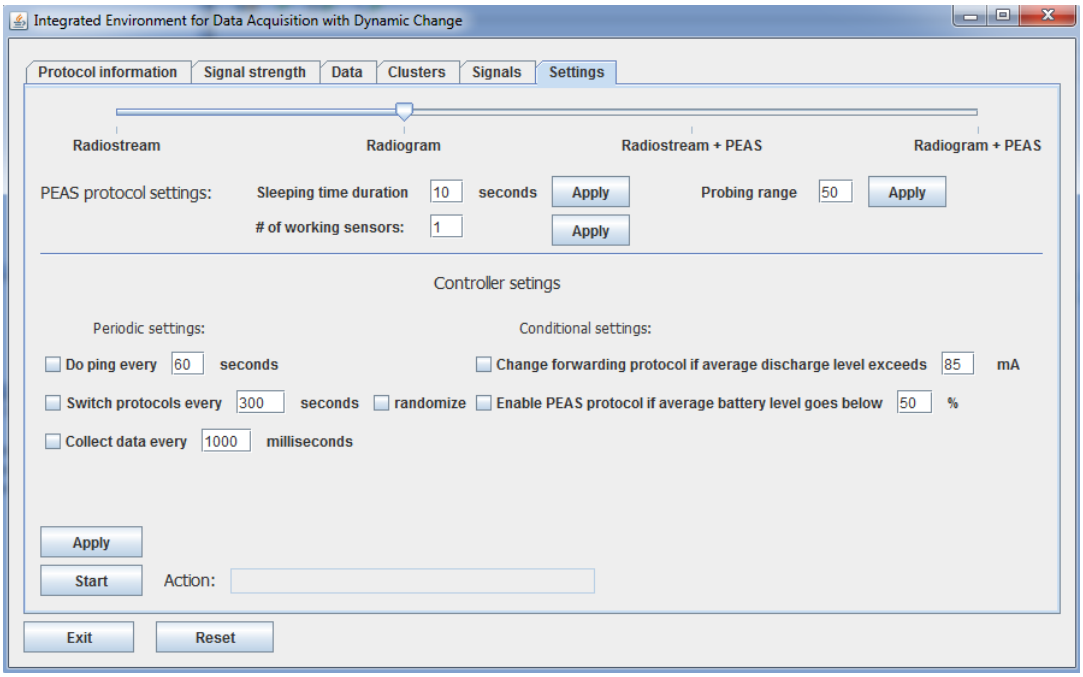


Fig. 5.8. *The "Settings" tab of the framework.*

For example, the degree of coverage can be customized in the PEAS protocol settings. By default this value is set to one. However it is possible to increase this number to attain higher accuracy through the "# of working sensors" filed. The second parameter we can customize is the sleeping time duration (in seconds), which affects the frequency of probing rate of sleeping sensors. The Probing range, the third parameter, determines the signal strength threshold level used by probing and working sensors to determine whether to react to the incoming messages.

The Controller settings bring some level of automatization to the process of the interaction with WSN. The user can dynamically customize them by pressing the "Apply" button. The settings are subdivided into two categories: periodic settings and conditional settings. The first category lets the controller repeat commands after predetermined periods of time, which are set in the corresponding text fields. The second category performs the command only if a certain condition is satisfied. For example, it is possible to switch protocols between Radiostream and Radiogram after every 300 seconds. The period of switching can be changed in the text field next to the corresponding check box. The check box "randomize" lets the period to be generated randomly within the range from the period text box. This option brings some level of security to the WSN communication. Although the security concepts are out of the scope of this work, this option enables random changing of communications protocols, which in turn will make eavesdropping or man-in-the-middle attacks more difficult. The use of this option can be later investigated in the future work. Among other periodic settings is the option of periodically pinging the sensor nodes and collecting data. Conditional settings include an option such as switching forwarding protocol if the average battery discharge level exceeds the threshold value and enabling the PEAS protocol if the average battery level of the sensors goes below the threshold value, which is also set in the corresponding text field.

Moving slider only takes effect when the controller is running. Moving it from one position to another triggers two commands: switching protocol from one to another and enabling/disabling PEAS protocol. For example, when the slider is at the Radiogram position, this means that the Radiogram protocol is currently in use. If it is moved to the Radiostream + PEAS position, the protocol would first switch to the Radiostream and then PEAS protocol would be enabled. If the slider is moved back to the Radiogram, the protocol would switch back to the Radiogram and PEAS protocol would be disabled.

The “Start” button at the bottom left corner of the panel starts the Controller. This entails that the framework will analyze the incoming data and trigger the corresponding actions automatically, specifically protocol switching, the period of network state information collection or the condition for enabling the PEAS protocol. The framework logic in Section 5.2 describes this part in detail.

5.1.2 Model

The Model module is run on a host application, which resides on a computer. It is the core part of the framework functionality. It is responsible for communicating with and management of the sensor nodes. The Model provides an overall optimization of the WSN operation against selected criteria. This way it assures the protocol adaptation to a particular application by supporting dynamic protocol selection. In this module, all the information from the sensor nodes is collected and analyzed. Based on this analysis, the Model may trigger corresponding actions through the Controller component. All actions taken are constantly reflected on the View, so the user can track down the behavior of the Model. The management of a WSN involves maintaining the state information about the network on a per node basis, such as the link status, state of a node (active or sleeping), battery level, and the sensed data. Communication with the sensor nodes is the means by which the interaction between the base station and the sensor network is established. The interaction includes sending the commands to one or multiple nodes as well as retrieving the sensor network state information or application specific data. Enabling plug-ins and dynamic protocol selection is the key to flexibility and adaptation of the IEDADC to various WSN applications. Once the network starts running, the protocol can be selected and the plug-in can be activated either by a user or automatically, according to the user predefined setting. The plug-in can be dynamically configured only by a directive from a user.

The model interacts with sensor nodes using the base station. Unlike the sensors, which run J2ME applications, the host application is a regular J2SE program. In this framework the base station operates in dedicated mode, meaning that it can be used only by one application, within the same JVM [13]. As a result, the overall representation of the network can be described as some number of sensor nodes sending information to a single sink node (our base station).

The model has references to the View module through the corresponding interface and two other objects: one plots the graph of the average current being drawn from batteries and the other plots the graph representation of the network (Fig. 5.9).

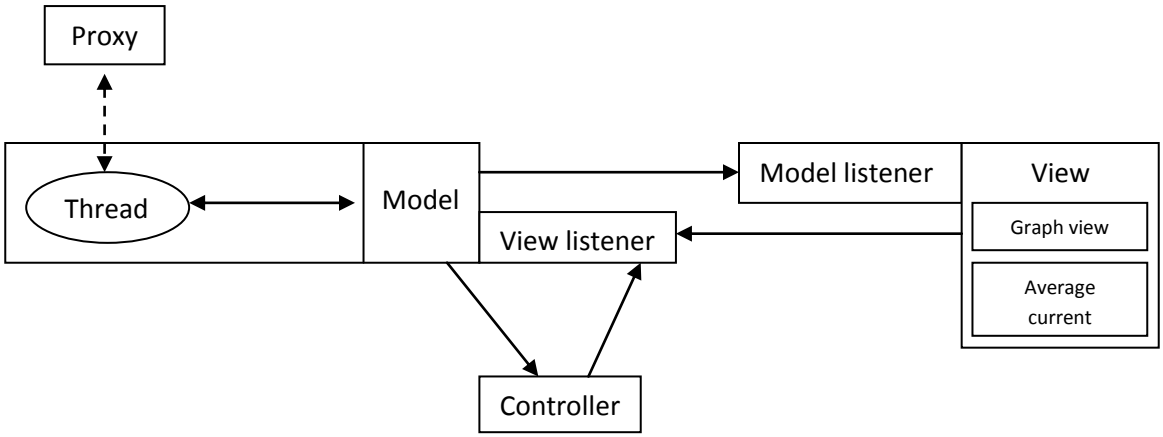


Fig. 5.9. *The references between modules of the IEDADC.*

When Model is initiated, it first checks if the base station is present. If true, then it performs all other necessary initializations, such as creating references to above mentioned objects. Then it broadcasts its availability and starts listening for the connection requests from the sensor nodes. The sensor nodes, in turn, try to establish connection to the base station by periodically broadcasting the request for connection. Once the base station receives the request, it replies to that node and creates a listening thread for this particular sensor. There is a separate thread for every sensor establishing connection to the model, including virtual sensors (Fig. 5.9). When sensor node sends some message to the base station, it is processed by the thread at the model, and then the message is sent to the corresponding method in the Model. It also works in the opposite direction: the message sent to a particular node is sent through the corresponding thread instead of broadcasting.

The Model module implements the localization and clustering algorithms. These algorithms are provided as part of the QA mechanism of the framework. The clustering algorithm is implemented in such a way that if the measurement of a sensor deviates by five degrees Celsius from the measurements of other sensors in the cluster, then this node is separated in a different cluster. This is done in order to avoid deviation of the calculated value of a temperature in a cluster and to attract the attention of a user. This approach was inspired by [16], where every data is given a flag of a certain color, where the green flag is given when

the data is “good”, i.e. there is no deviation from the overall measurements and the red flag otherwise. The paper mentioned that if the data is given a red flag, it does not necessarily mean that the data is wrong, it only lets the user know that the current data requires attention. This approach is helpful in the case when there are two sensor nodes located close to each other, but separated by a window – one measures the temperature indoors and the other one measures the temperature outdoors, where the temperature can be significantly different.

Also the mechanism of switching the protocols is implemented in this module. The model keeps track of which protocol is currently in use, so when switching protocol, the model broadcasts corresponding command to the sensor nodes. Sensor nodes in their turn reconnect to the base station. When the base station receives the request for connection from the node to which it already has a connection, it removes the previous connection and creates a new one with the new protocol in use. The reply message from the base station contains the information about the protocol which should be used by the nodes.

The framework interaction with the sensor network can be represented as the following (fig. 5.10). Upon a user’s command, the Model sends commands to the proxy modules. The proxy modules control a sensor node’s operational mode and, possibly, send back the reply. The plug-ins can influence the operation of a sensor node through interaction with the proxy module. Sensor nodes can interact with each other when using miltihop routing or probing the environment for locating working sensor nodes as part of the PEAS protocol operation.

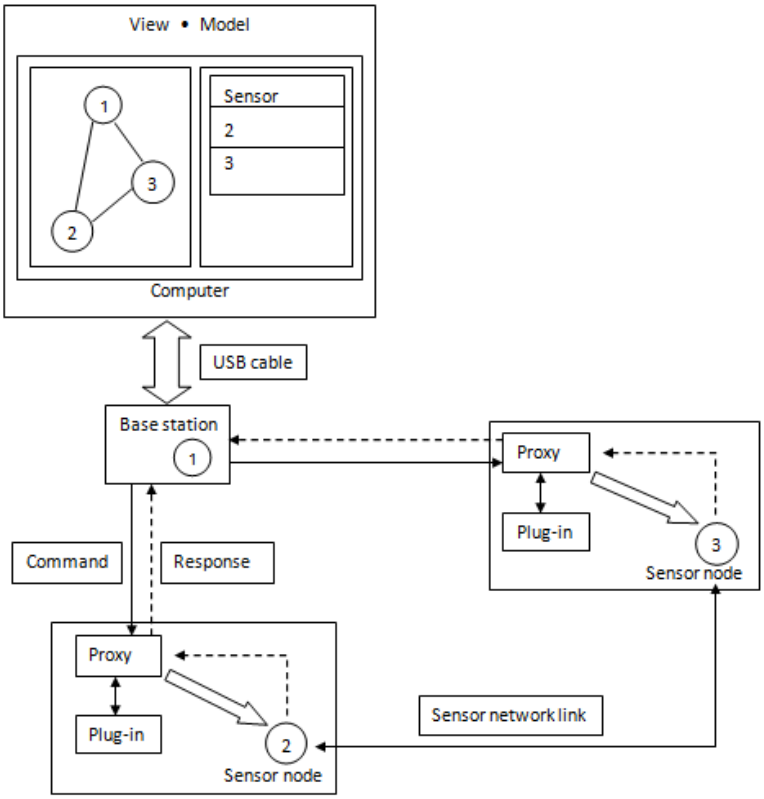


Fig. 5.10. *IEDADC sensor network interaction overview.*

Another feature of the Model is the support of virtual sensor nodes, which emulate the behavior of real sensors. In this case no connection is created between a virtual sensor and the base station. As before, a separate thread is created for each virtual sensor. A user can manually set the parameters of the virtual sensor through the View module, such as the signal strength to the base station, the temperature measurement, and the battery level of the sensor. This information will be taken from the thread by the Model for analysis. Virtual sensor nodes allow testing the behavior of algorithms implemented in the framework, such as the localization and clustering algorithms.

5.1.3 Plug-in

The goal of the Plug-in module is to optimize a WSN operation for a particular application by selecting certain parameter values. The plug-in parameters can be dynamically customized by the directives from a user. For example, managing the degree of coverage of the monitoring area by putting the redundant sensor nodes to sleep, keeping only necessary number of working nodes, is done by the PEAS protocol.

This protocol is implemented as a part sensor application of the framework. However, the parameters of the protocol, which affect its work, are configured from the GUI by a user. The new parameter values are broadcasted from the base station to every sensor node.

The following parameters can be configured in this protocol.

As mentioned in section 2, sensor sleeps for exponentially distributed duration generated according to a probability density function $f(t_s) = \lambda e^{-\lambda t_s}$ [21], where t_s is the sleeping time duration (measured in seconds) and λ is the probing rate – the frequency of a sensor node probing the environment for a working node (measured in wake-ups per seconds). There is also another parameter: λ_d – desired probing rate. Eventually, after some time the λ will float around the desired rate, according to the proof provided in [21], with the help of adaptive sleeping mechanism. These two lambdas are initialized when the protocol starts. The reason for having two separate lambdas is that initially it is required to establish working nodes as fast as possible, that is why the value of λ is higher than λ_d, so that the network can start operating. The value of sleeping time duration t_s directly affects the value of the λ according to the formula: $\lambda = \frac{1}{t_s}$. Since the frequency of wake-ups is not needed to be high after the working nodes are found, the value of λ_d, towards which the λ is moving, is calculated as: $\lambda_d = \frac{1}{5t_s}$. The sleeping time duration parameter can be configured through the View module.

Another parameter is the number of the working nodes. In order to increase the accuracy of data, the number of working sensors within one region can be increased from one to some number k. This parameter customizes the degree of coverage of the sensing area.

It is important to note that pseudorandom generators provided by Java programming language only return us a uniformly distributed value within certain range. In order to generate exponentially distributed values, the Inverse Transform Sampling formula is used:

Let $U \sim U(0,1]$, then $X = -\frac{1}{\lambda}\ln U \sim Exp(\lambda)$,

where λ is the probing rate, U is the uniformly distributed value between 0 (exclusive) and 1 (inclusive) and value of X is the generated sleeping time duration of a sensor node.

5.1.4 Proxy

The proxy is the IEDADC module that is run on a sensor node. It has three main responsibilities:

1) controlling the sensor node's operation;

2) sending the application specific data (i.e. temperature) to the sink node;
3) retrieving the sensor state information (battery level, signal strength).

The first responsibility of node's operation is controlled by maintaining the processes running in the node according to the commands sent by the Model. The second responsibility involves monitoring the environmental conditions and sending this data to the sink node. The last responsibility requires informing the base station about current state of the sensor node if this information was requested by the system.

This module is composed of three packages: net.java.dev.netbeansspot, net.java.dev.service and org.sunspotworld.demo.util.

The first package contains the main program Application.java of the MIDlet, which handles most of the requests sent by the base station and creates connection with the base station with the help of the LocateSevice utility. Monitor.java is responsible for periodically sensing the environment. Current implementation measures the temperature in Celsius.

The second package contains the implementation of the PEAS protocol.

The third package contains helper programs that do most of the routine work. LocateService object is responsible for locating the base station and returning its address to the main program. PacketReceiver object is responsible for listening for commands from the base station and sending the packet for processing to the corresponding handler. PacketTransmitter sends the data from a sensor node to the base station. PeriodicTask is an object, which is responsible for performing periodic tasks, such as measuring temperature. Thus the Monitor object extends PeriodicTask object. This period can be set to any value through the View module. Also in order to save energy while monitoring the environment, the datagram is not sent unless it is full. Therefore several samples of temperature measurements are transported in one datagram. This feature affects the periods after which the base station receives the samples. Particularly, a datagram can fit six temperature samples and if the sampling period is ten seconds, then the packet is sent every minute, once it fills up with the data. One needs to consider this fact when changing the sampling period. Virtual sensors demonstrate this behavior as well.

5.2 The framework implementation

This section describes QoD assurance and sensor nodes' localization methods developed and implemented in IEDADC as well as the mechanisms of automatic control of the network.

The goal of current version of localization is to calculate an approximate signal strength value between sensor nodes. This information is later used to merge sensors into clusters. The advantage of this algorithm over other localization techniques [18] [22] is that it does not require the knowledge of the location of the anchor node, which is used to determine the relative position of other nodes in the network.

This algorithm is performed in three steps. First step involves the sink node, which beacons the hello message to all of the nodes in the network and waits for the replies. The reply message contains various information about the state of the sensor including battery level (in percent), the current being drawn from the battery (in mA) and the signal strength of the message (in dB). The first node that replied to this message is determined to be an anchor. The second step involves the anchor node, which beacons another hello message. The nodes respond to that message by sending a reply to the anchor. The signal strengths of the reply messages are used in the third step to perform the actual calculation of signal strengths between sensor nodes. The following figure will demonstrate the scheme of sensors' locations with signal strengths as their distance to each other:

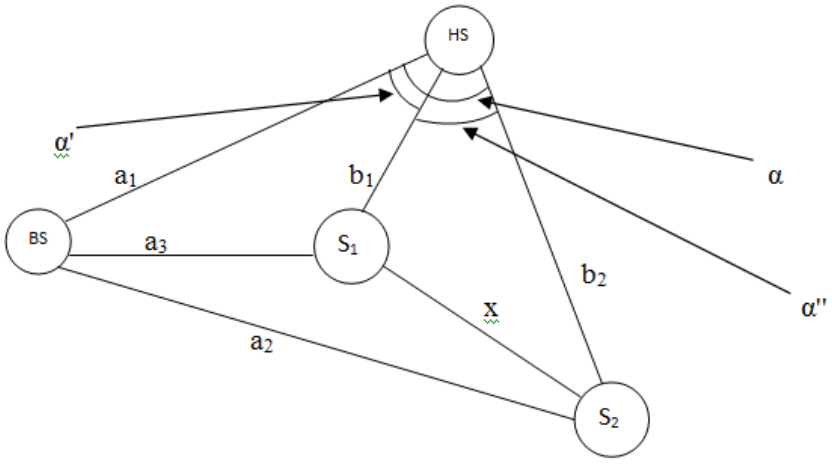


Fig. 5.11. *The schematic representation of the sensor nodes' location. BS – base station; HS – helper SPOT (anchor); S_1 – sensor node #1; S_2 – sensor node #2; a_i – signal strength between base station and a sensor node; b_i – signal strength between the anchor and a regular sensor node; x is the calculated value.*

The replies to the hello message from the sensor nodes to the base station are denoted as a_1, a_2 and a_3. The replies to the hello message from the sensor nodes to the anchor are denoted as b_1 and b_2. The signal strength value is calculated between two sensor nodes that

have replied to the anchor. Thus in order to make the calculation there must be at least three sensors and the base station in the network.

The calculation is done as follows. Firstly, the angle α is calculated according to the law of cosines:

$$\alpha = \cos^{-1}\left(\frac{a_1^2 + b_2^2 - a_2^2}{2a_1b_2}\right) \qquad (1)$$

Then according to the same law α' is calculated:

$$\alpha' = \cos^{-1}\left(\frac{a_1^2 + b_1^2 - a_3^2}{2a_1b_1}\right) \qquad (2)$$

Then α" can be calculated as:

$$\alpha'' = |\alpha - \alpha'|$$

Once value of α" is found, the variable x can also be calculated according to the law of cosines:

$$x = \sqrt{b_1^2 + b_2^2 - 2\,b_1b_2\cos\alpha''}$$

There are some transformations required in order to make this algorithm work. The method getRSSI() (received signal strength indicator) returns the signal strength of the signal for the packet. It ranges from +60 (strong) to -60 (weak). To convert it to dB we need to subtract 45 from this value, e.g. for an RSSI of -20 the RF input power is approximately -65 dB [14]. However, the geometrical length cannot be less than or equal to zero. In order to avoid this, 61 is added to the returned value of the method. This means that the weakest signal would have the value of 1 and the strongest – the value of 121. To convert this value back to dB, we would have to subtract 106 from it.

There are other transformations performed in this algorithm to increase the accuracy of the results. We assume there is strong signal strength between two sensor nodes which are close to each other. Therefore, in reality, when the value of the signal strength is high, the distance to that sensor node is short. The values of the signal strengths affect the values of the angles calculated above. In order to avoid wrong calculation results, the signal strengths need to be converted to get real values of alphas. To achieve this, the value of the signal strength is subtracted from 121 (the highest value of signal strength attainable). In another

transformation, the value of the signal strength is inverted modulus 121. When the value of x has been calculated, it is converted back as: $x = 121 - x$. The experimentations on the signal strengths between nodes demonstrated that the signal strength between the base station and a sensor node is always higher compared to the signal strength between two sensor nodes positioned equally distant from each other. Therefore, high signal strength from the base station affects the overall result of the calculation. The result becomes higher than it should be. To solve this problem, after the value x is converted back, it is divided by 1.6. This coefficient has been received experimentally. It appears that the ratio of the signal strength from a sensor node to a base station and the signal strength between two sensor nodes is 1.6. So the value of x is adjusted accordingly.

Also experiments on signal strengths demonstrated that they do not always follow the property of triangles, which states that the length of one side of the triangle cannot be greater than the sum of other two sides. These experiments demonstrated that in the worst case the value of the highest signal strength among three sensor nodes may be few units higher than the sum of other two signal strengths. This fact results in having a value slightly greater than 1 in the argument of the arccosine function of the formula (1) and (2). To avoid this, the algorithm is implemented in such a way that every time the argument of the arccosine function exceeds one, it is assigned to one.

This method is only supported when using the Radiogram protocol. The reason is related to the property of the Radiostream protocol. As it was previously mentioned, Radiostream protocol opens point to point connection between two nodes and this protocol doesn't support broadcasting. In order to achieve step two, when anchor sends the hello message to other sensor nodes, the anchor would need to create a connection with every node. In order to do so, this node needs to know the addresses of the rest of the nodes, so the base station has to send this information to the anchor. Then the anchor needs to send the hello message individually to every node. It is known that sending a signal is the most expensive operation in terms of energy consumption. So if there are N nodes in the network, not including the base station and the anchor, the number of hello messages sent using this protocol will be also N. While broadcasting the same message with Radiogram is done only once. As the number of N increases, the number of messages that need to be sent also increases proportionally. This is why it is not worth implementing the algorithm with the Radiostream protocol. In this case no further clustering is done and every sensor is treated as a

separate, individual source of information. Therefore there is no QA performed when using the Radiostream protocol. On the other hand this protocol provides a reliable connection between the base station and the sensor nodes. Overall network scheme can be viewed as the following figure:

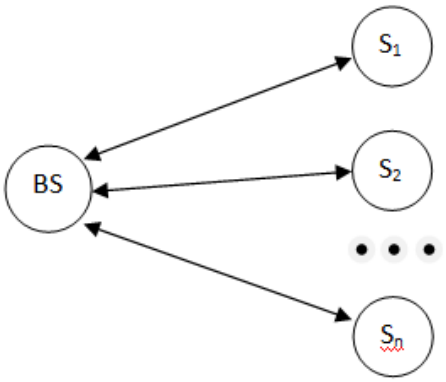


Fig. 5.12. *The schematic overview of a network running on Radiostream protocol.*

Section 6 describes the results obtained from experiments localization algorithm.

Clustering of sensors is performed according to the relative position of sensor nodes with respect to each other, which is indicated by the signal strength value. High signal strength value represents a situation when sensor nodes are located close to each other. The clustering mechanism is performed in the following way: given the clustering threshold, sensors are merged into clusters if and only if the signal strength exceeds the threshold value.

One sensor node may belong to several clusters. This feature is provided to increase the accuracy of the data by avoiding the situation when a sensor node is left alone in a separate cluster. For example, there are three sensor nodes are in one cluster and two others are in another. One node in the first cluster has enough signal strength to create a cluster with a node, which is in the second cluster. However these two clusters cannot merge, because not all of these sensors are close to each other. Next, these two sensors, apart from being in their respective cluster, create another, the third cluster, which partially intersects with the first and second clusters (Fig. 5.13).

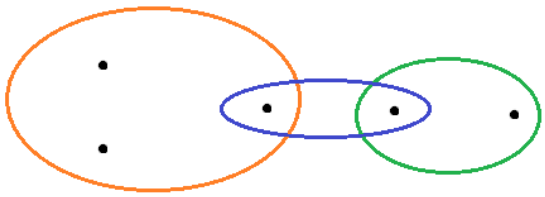


Fig. 5.13. *Intersection of clusters. Black spots represent sensor nodes, which are clustered together as shown with colored circles.*

The value of the clustering threshold is always application specific. It depends on the application requirements and the trust a user gives to every sensor node's measurements. If the confidence in the received data from every sensor node is high and an application requires knowledge of an exact temperature level in every particular location, then the clustering threshold should be set high, so that there will be few sensor nodes per cluster. On the other hand, if the individual measurement is not important or a user doesn't rely on the individual measurements then the threshold value can be set lower. In this case more sensors will merge together in a cluster and their measurements will be analyzed according to the aggregation function described below.

The measurements (i.e. temperature) within one cluster are processed according to one of the aggregation functions selected by a user. There are numerous approaches of processing the incoming data. To demonstrate a possible function selection mechanism the current version of IEDADC implements average and vote functions.

The averaging function calculates the mean value of incoming data within a cluster. When the voting function is used the value with the highest number of votes is determined as a value within a cluster. The third column in the Clusters tab of the GUI displays this value. In a vote function two temperature values are considered to be the same when their absolute difference does not exceed two degrees Celsius. Also values are only compared if they are up-to-date with respect to each other, i.e. difference of their timestamps is no longer than five minutes.

There is also a sub-clustering mechanism implemented.

If one of the sensors within a cluster shows a completely different result – in the implementation of the framework the difference in 5℃ is considered as such difference - then

sub-clustering is performed. As before the value is first compared with other values for being up-to-date. Nevertheless, if all the measurements are up-to-date and the difference in temperature exceeds the predetermined value, then this sensor is separated to a different cluster to attract the attention of a user to this sensor node. In real life it is not always true that the sensor starts functioning improperly, most likely it was moved to other environmental conditions or the conditions around that sensor had changed, without changing position of the sensor. That is why it is important to keep these sensor readings.

After certain periods of time the framework running in automatic mode will recalculate the signal strengths and cluster the sensors. Therefore if the sensor node wasn't moved, then we expect the signal strength to stay the same and therefore this sensor will be put back to the cluster it belonged before. If more and more sensors start obtaining the same different result within the same cluster, then it is most likely to be true and eventually the sensors will not be separated to another cluster.

Controller has an access to the Model through the ViewListener interface and so does the View module, therefore it is capable of doing the same operations. The main duty of the Controller is to automate some processes in the framework. For example, it can send a hello message every minute to measure the battery level. Every time this command is performed, the controller takes the results of average battery level and the average current being drawn from the batteries, compares them to the threshold values and triggers an appropriate action.

Settings of the Controller can be changed dynamically. A user only need to press the Apply button (see section 5.1.1 for details). However when changing the periodic settings, one needs to consider the fact that in between the periods the thread which is responsible for a particular periodic command is in sleep. So new values will only take effect after the thread wakes up from the old period.

6. Experiments and results

This section presents the experimentations performed using IEDADC and their results.

6.1 Investigation of the localization algorithm accuracy

The goal of the first experiment performed in this work was to verify if the calculated signal strength between sensor nodes reflects the real values and to quantify the accuracy of such calculation. The results of this algorithm will be later used by the clustering algorithm, as

a part of the QA mechanism supported in this framework, so the correctness of the results of this algorithm will directly affect the accuracy of the data. The experiment involved placing various numbers of sensor nodes in different locations such that the signal strength can be easily predicted and compared with the values shown by the framework. For example, if two nodes are next to each other, we expect them to have strong signal strength and experiments show that the signal strength between such sensors is around 70 on a 1 to 121 scale. As mentioned earlier, to perform the calculation of the signal strength we need at least four nodes, otherwise there is no need in doing the calculation, since the signal strengths to all the nodes are already known. To clearly see it, return to the figure 5.11 and remove S_2 from it, then a_1, a_3 and b_1 will be the only signal strengths in the network. All these values are known and no additional calculation is needed. This property was taken into consideration when performing this experiment. The experiment began by placing fewer than four nodes in the network in different locations. This helped to see what signal strengths should be expected at certain locations. Then the fourth sensor node has been added to the network. This node was placed to the same locations as sensors before. Different combinations of placing have been tried: the sensors were placed in pairs and moved apart every time by some distance. This was done in order to see how different placements affect the calculation of the signal strength.

At first, there were three nodes in the network: two sensor nodes and one base station. The base station and one sensor node remain close to each other and the second node (its address ends with 1674) is moved away every time by one/ten meter(s). The same experiment was done both for short and long distances. The following two charts show the signal strength distribution over the distance:

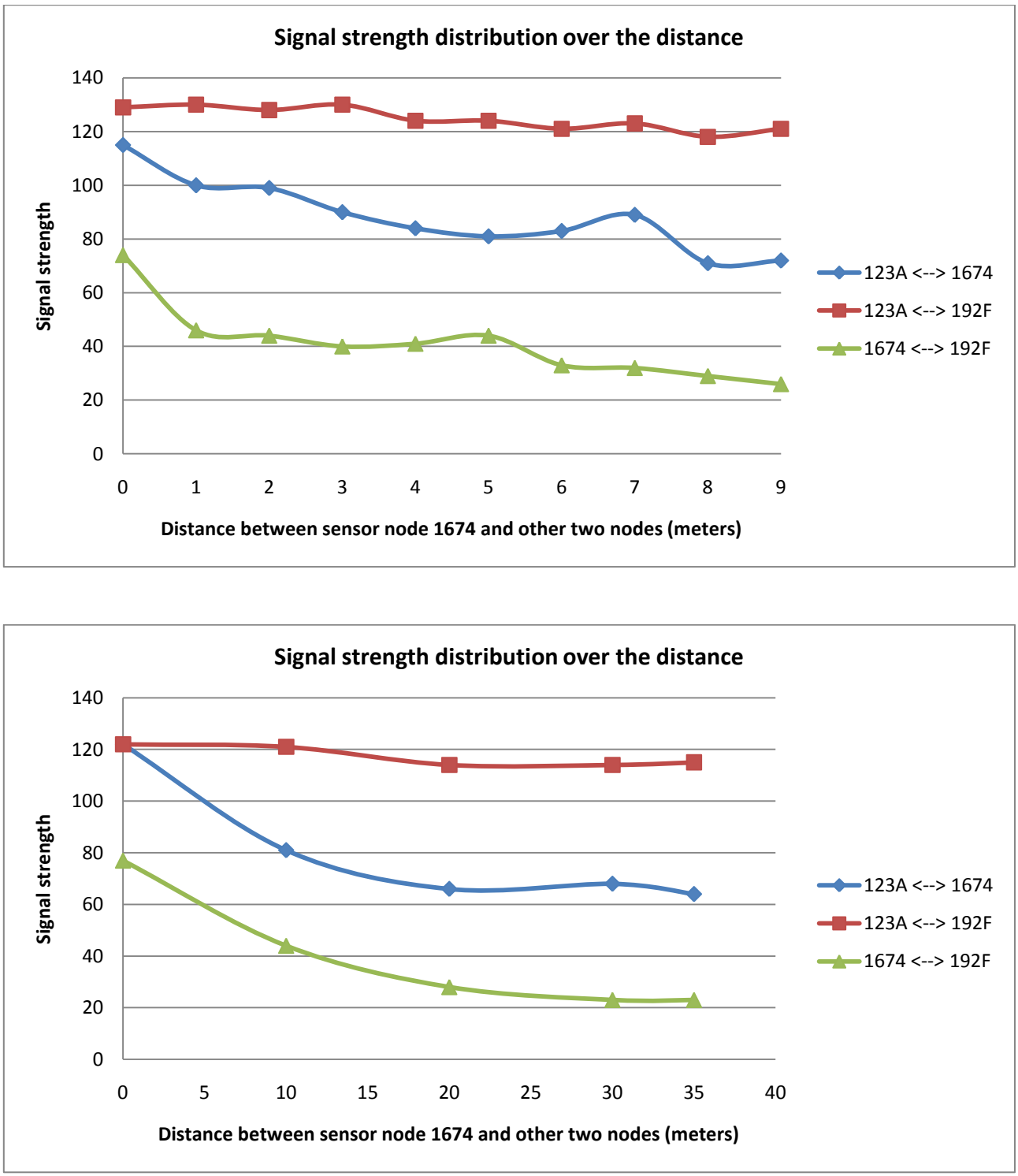


Fig. 6.1. *The signal strength distribution over the distance. Top chart shows the results of the experiment for short distances, and bottom chart shows the results of the same experiment for long distances.*

As we can see from the plot, as the node 1674 is moved away from the base station (its address ends with 123A) and from sensor node 192F, the signal strength decreases. The

higher signal strength received by the base station is explained by the fact that it is always connected to the host computer and therefore has "unlimited" power supply. Consequently, it can spend all the power it has to send and receive the signals, unlike regular sensors. Nevertheless, as the node 1674 is moved away, its signal strength decreases proportionally with regards to both the base station and the node 192F. The signal strength between 123A and 192F stay approximately the same, since they remained close to each other all the time. The maximum distance between two nodes until the signal is lost is 35 meters.

The reason why sometimes the signal strength peaked instead of decreasing is explained by the fact that the sensors use 2.4GHz frequency range radio waves for communication and therefore many other interferers tend to affect the signal strength, such as metal or water containing objects. As explained in [8], the antennas on the Sun SPOT sensors send their signals in all directions, since they do not know the location of their receiving side. So there is a possibility when one signal propagates directly to the receiving side and another takes a longer path through reflections causing multiple versions of the signal with different phases at the receiver (Fig. 6.2).

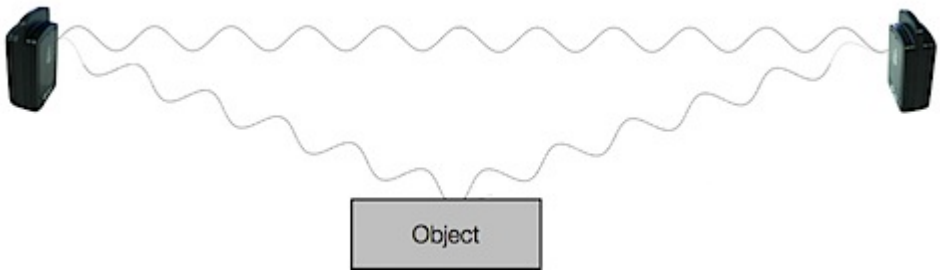


Fig. 6.2. *Distribution of two radio signals: one going directly to the destination and another bounces from an object and then reaches the sensor node*[4].

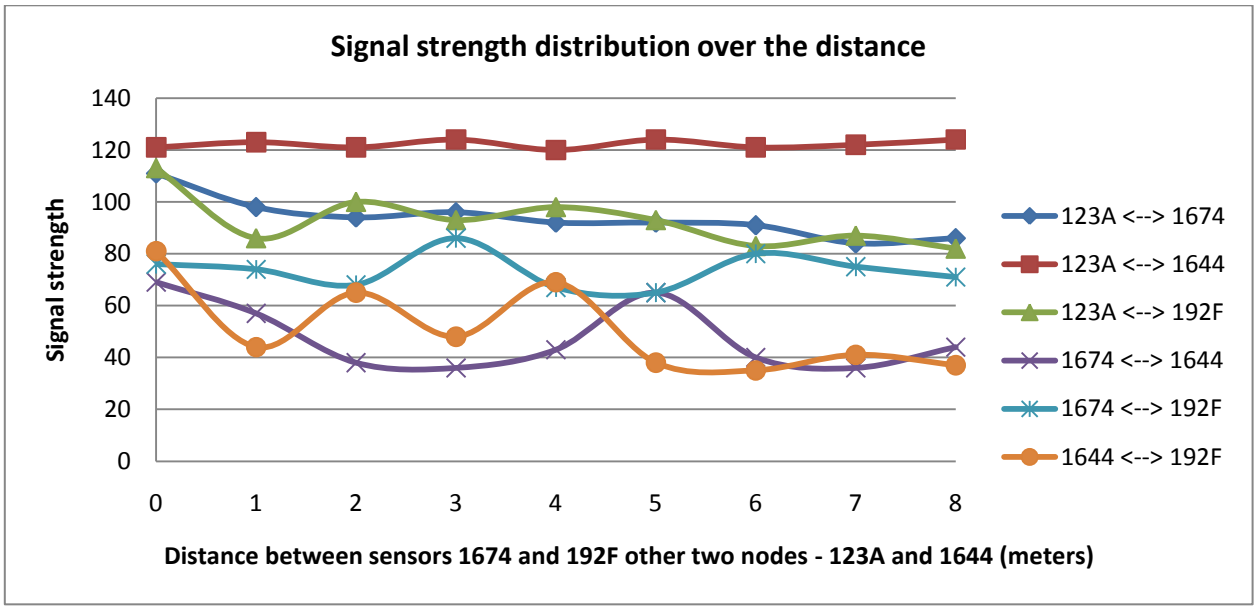
The receiving node will receive the mixture of signals shifted in time, which are added up to form a single signal. There could be two scenarios at the receiver. The best scenario is the constructive addition of signals amplifying the overall signal, because there is an insignificantly small shift in the phase between signals. The worst scenario is when the phase is shifted enough to cause overlap and cancellation of the received signals. The RSSI value is calculated from the amplitude of the received signal and therefore the nature of the channel together with the multipath conditions play an important role during reception.

[4] The picture is taken from http://blogs.sun.com/roger/entry/location_location_location_radio

Even the orientation of the SPOT can indirectly affect the signal strength [8]. Since the signal strength depends on so many factors that cannot be predicted or foreseen, the signal strength in reality may be quite different from what we may expect in theory. The experiment involved moving sensors apart in the environment where nodes had little obstacles between each other. Despite this fact, the results from the experiment are very important, since they show that increasing distance between two nodes eventually affect the signal strength.

Also if two sensor nodes are in the same room then their signal strengths on average should be stronger that two nodes separated by a wall. The clustering threshold can be customized to make a decision about which sensor nodes should be considered close to each other and which ones are not. In case when RSSI between the nodes is high, but they are actually located in different environmental conditions, then proper clustering is handled by the sub-clustering mechanism.

The next experiment was intended to determine the accuracy of the signal strength calculation algorithm. The third sensor node was added to the network. Two nodes – the base station (123A) and one node (1644) remained together and other two (192F and 1674) were moved away by one/ten meter(s), remaining in pairs. So we should expect the RSSI between 192F and 1674 to have a high value, and the same is applicable to 123A and 1644. On the contrary, the two nodes (192F and 1674), which are moving away, are expected to have a weaker signal strength compared to 123A and 1644. The following two plots show the results of this experiment:



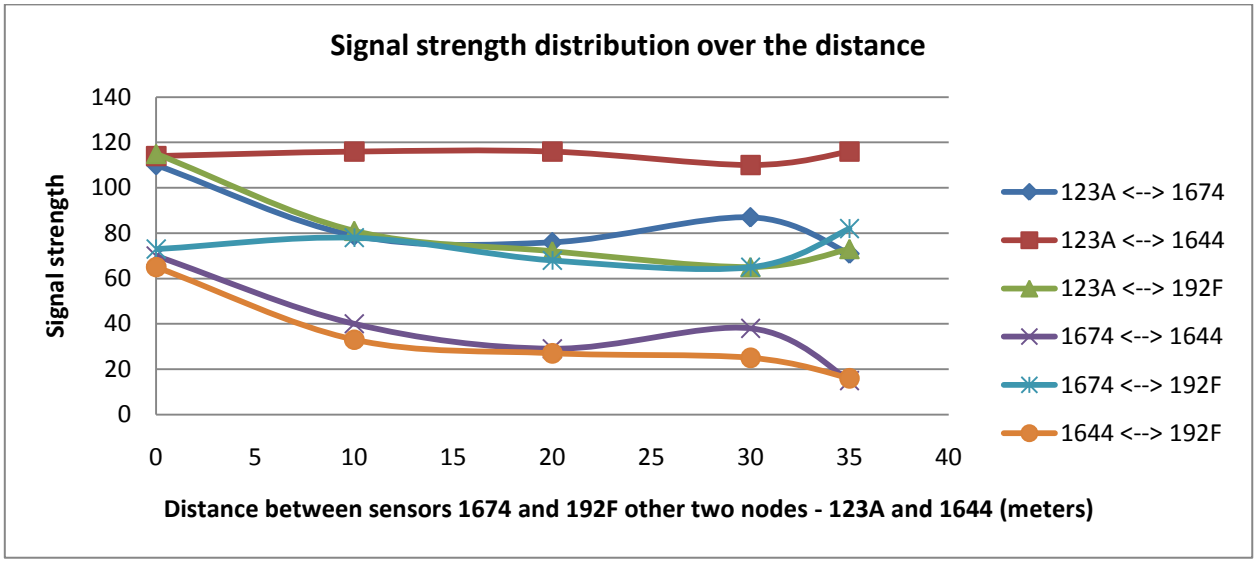


Fig. 6.3. *The signal strength distribution over the distance, using method to calculate the signal strength between sensor nodes. Top chart shows the results of the experiment for short distances, and bottom chart shows the results of the same experiment for long distances.*

As we can see from the plots, the signal strength between 1644 and 123A (red line) remains consistently high. This is because they were placed together. Also as two nodes are moved away, the signal strength from the base station to these sensor nodes decreases as in previous experiment with two nodes (dark blue and green lines). The signal strength between 192F and 1674 (light blue) stays in average at about 75. This is what we expect from two sensor nodes being close to each other; we observed the same results from the previous experiments as well. However, if we look at the results of the signal strength between 1644 and 192F (orange line), in the experiment when sensor nodes are moved in short distances, we can see that our expectations do not always match the results. In theory we expect orange and violet lines to be almost identical to each other, but when these two sensors are 2 and 4 meters away from each other, the signal strength is about the same level as if they were close to each other. However, as the distance between them is increased, the signal strength returns to our expectations. The same observation in RSSI levels can be made between 1674 and 1644 (violet line), when the distance between them is 5 meters. What is more important is that out of 9 measurements taken, there were 3 unexpected results: on two, four and five meters away from the 123A and 1644. The second plot demonstrates more idealistic results, i.e. they are closer to our expectations.

Next, we look at how the calculation algorithm works on four sensor nodes and one base station. The same approach: three nodes remain close to each other – these are the base station (123A) and two nodes 13CE and 1674. Two other nodes 192F and 1644 are moved away every time by one/ten meter(s). In order to make the results more readable, the plot was divided into two (Fig. 6.4.x). Since three nodes 123A, 13CE and 1674 are close to each other, we expect them to have high signal strength; the same is expected for 192F and 1644. Figure 6.4.1 demonstrates that this is true in all cases except one: when 192F and 1644 are five meters away, the calculation gives higher RSSI for distant nodes and lower one for close nodes. However, in the next measurement, the values are again within the expected range. We also expect that since 192F and 1644 are moved away from 123A, 13CE and 1674, the signal strength between these two groups of sensors should decrease. This is true for all the cases except the one already mentioned; even a quick look at the plot can show that the two lines identical: 123A <-> 1672 and 123A <->13CE; 123A <->1644 and 123A <->192F; 1674<-> 1644 and 1674 <->192F; 13CE <->1644 and 13CE <-> 192F.

As for the experiment on long distances (fig. 6.4.2) we can observe that the results are generally are within the predicted range, except for the case when the signal strength from 1674 to 192F and 1644 increases as the distance is increased between them. Nevertheless, two lines on the plot: 1674 <-> 192F and 1674 <-> 1644 are close to each other – this is what we would expect from two sensor nodes (192F, 1674) being equally distant from the third node (1644). The signal strengths from these two sensor nodes to another node which is close to the base station (13CE) are decreasing as they are moved away; also two lines corresponding to these signal strengths are close to each other. As we can see from the figure 6.4.2, the signal strengths between sensors that remain close to each other are always at about the same level ~70 – 75. The signal strength from the base station to the closer sensor nodes (13CE, 1674) remains at the same level and for the sensors which are moved away, the signal strength decreases proportionally.

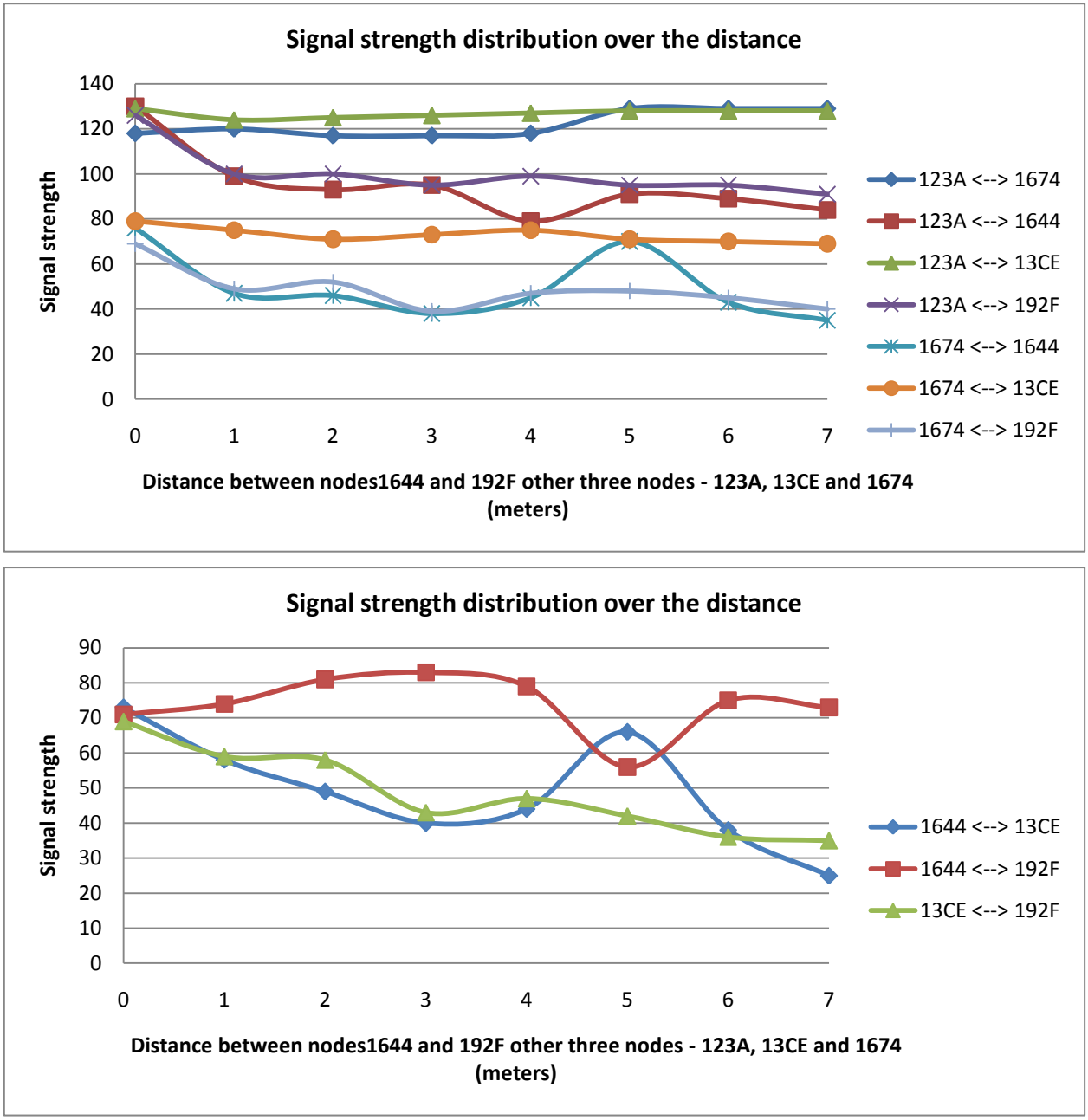
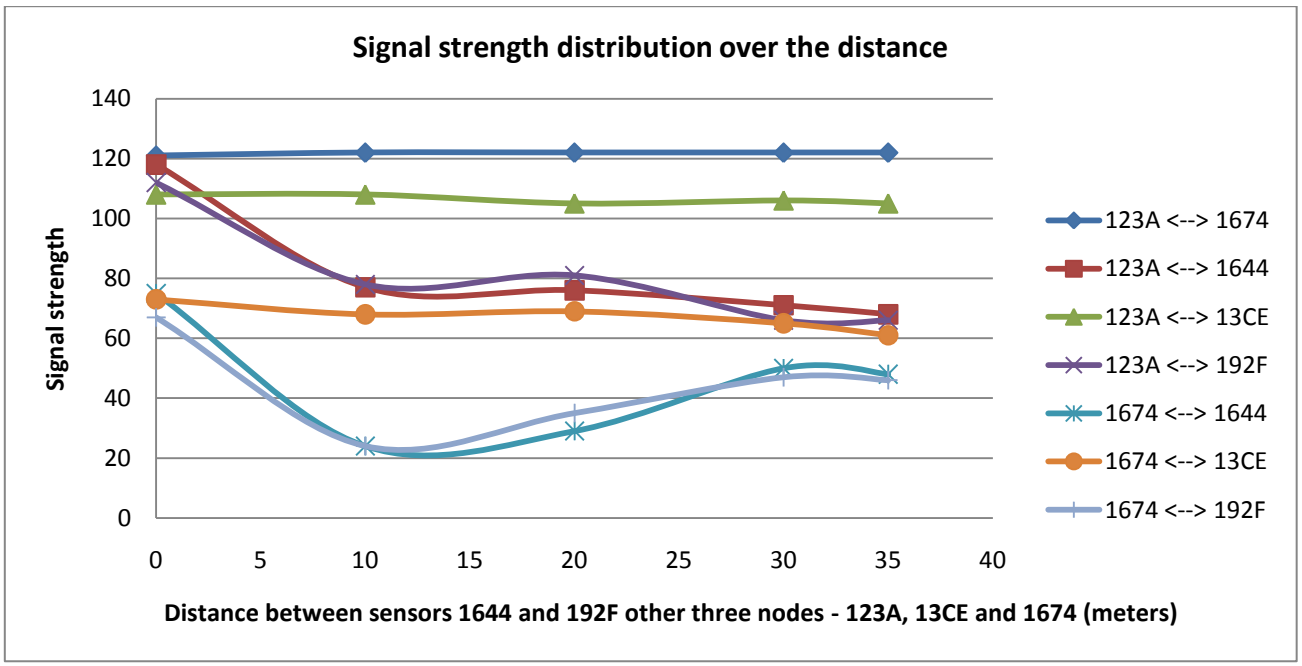


Fig. 6.4.1. *The signal strength distribution over the distance, using method to calculate the signal strength between sensor nodes with four nodes and one base station in the network. Experiment on short distances.*



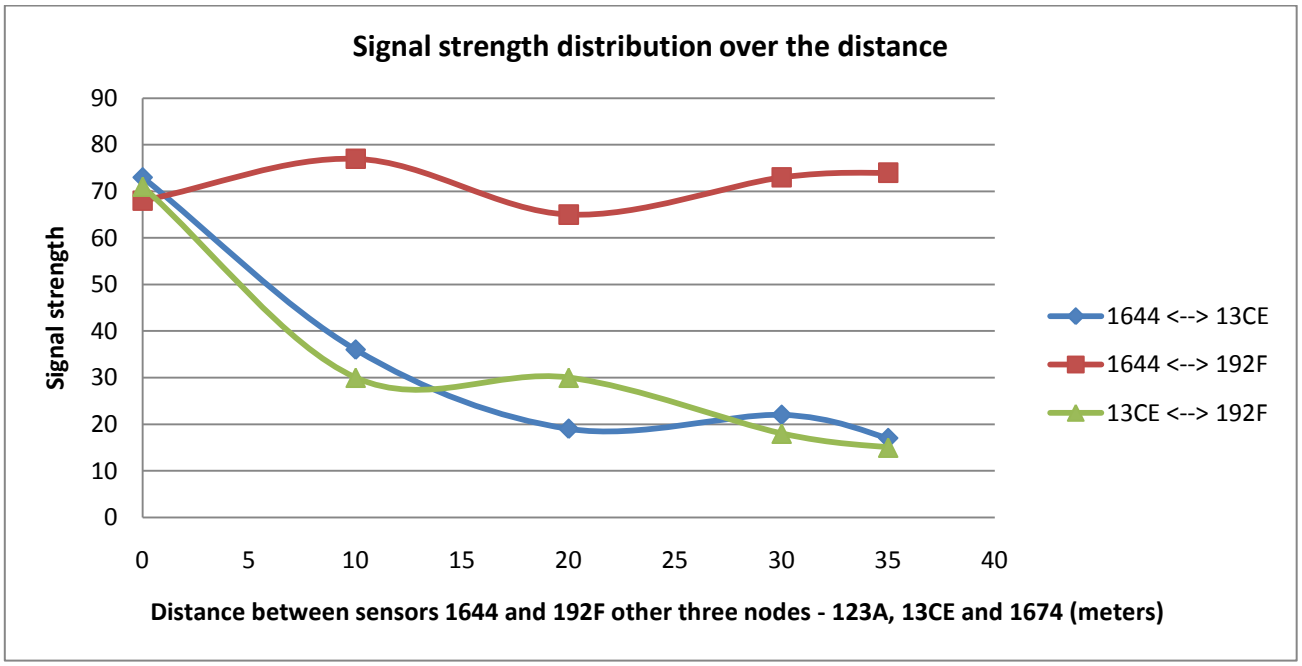


Fig. 6.4.2. *The signal strength distribution over the distance, using method to calculate the signal strength between sensor nodes with four nodes and one base station in the network. Experiment on long distances.*

In conclusion, we can certainly say that the signal strength depends on the distance between two nodes, but there are so many factors affecting the radio wave propagation, making the process of calculating a relative position of sensor nodes based on the signal strengths not highly accurate. Nevertheless this algorithm gives more accurate results when

applied to sensors that are at a longer distance of about 10 meters or more from each other. So, at greater distances, the algorithm does reflect theoretical expectations about signal strength distribution between sensor nodes. For shorter distances, this algorithm can be applied if the accuracy of determining relative positions of nodes is not required to be high.

6.2 Investigation of the QA mechanisms' operation

The next experiment demonstrates the work of the clustering algorithm implemented in this framework. The experiment relies on the results of calculation of signal strengths. Therefore clustering is only supported with the Radiogram protocol. The experiment aims to demonstrate the operation of the clustering algorithm. As it was proposed in the section 5.2, the sensor node, whose readings deviate from the readings of the other sensor nodes in the cluster, is moved to a separate cluster. This is done to avoid deviation from the average, in case the averaging function is used to calculate the temperature in the cluster. In case, however, the vote function is used, the vote of such sensor reading will not be heard as the majority of sensor readings will vote for a different measurement value.

This experiment involved four real sensor nodes. The sensors were placed together to form a single cluster and were given a command to measure temperature. The result of measurements is in the following table:

Sensor address	**Date of measurement**	**Temperature**
0014.4F01.0000.1644	Sun Mar 13 17:59:35 EDT 2011	25.708333333333332
0014.4F01.0000.192F	Sun Mar 13 17:59:35 EDT 2011	24.375
0014.4F01.0000.13CE	Sun Mar 13 17:59:35 EDT 2011	23.916666666666668
0014.4F01.0000.1674	Sun Mar 13 17:59:36 EDT 2011	25.375

The cluster table with the average temperature within the cluster is:

Cluster #	**Sensors**	**Temperature**
0	[0014.4F01.0000.13CE, 0014.4F01.0000.1674, 0014.4F01.0000.1644, 0014.4F01.0000.192F]	24.84375

As we can easily verify, the average of the four sensors' measurements is calculated correctly.

As for the vote function given the table:

Sensor address	Date	Temperature
0014.4F01.0000.1644	Sun Mar 13 18:00:35 EDT 2011	26.041666666666668
0014.4F01.0000.192F	Sun Mar 13 18:00:36 EDT 2011	24.791666666666668
0014.4F01.0000.13CE	Sun Mar 13 18:00:35 EDT 2011	24.375
0014.4F01.0000.1674	Sun Mar 13 18:00:36 EDT 2011	25.833333333333332

The vote function gave the following result:

Cluster #	Sensors	Temperature
0	[0014.4F01.0000.13CE, 0014.4F01.0000.1674, 0014.4F01.0000.1644, 0014.4F01.0000.192F]	24.375

As we can see from these two tables, the algorithm performed collecting of votes for the measurements from the sensor 13CE. Since all other values of temperatures don't differ from 24.37 by two degrees Celcius, the vote for this temperature was four. The value which received maximum votes is therefore 24.375 and this value was chosen to be the actual temperature for the cluster. In situations when there are equal number of votes for several candidates, one of those values is chosen arbitrarily.

Next, the sensor 192F was placed in a cold environment, still being close enough to other sensor nodes, such that the nodes can be in the same cluster. After some time, when the sensor 192F cooled down, the sensors showed the following result:

Sensor address	Date	Temperature
0014.4F01.0000.1644	Sun Mar 13 18:16:11 EDT 2011	25.166666666666668
0014.4F01.0000.192F	Sun Mar 13 18:16:12 EDT 2011	16.958333333333332
0014.4F01.0000.13CE	Sun Mar 13 18:16:12 EDT 2011	21.916666666666668
0014.4F01.0000.1674	Sun Mar 13 18:16:12 EDT 2011	24.041666666666668

The algorithm automatically placed the sensor 192F to a separate cluster:

Cluster #	Sensors	Temperature
0	[0014.4F01.0000.1644, 0014.4F01.0000.1674, 0014.4F01.0000.13CE]	23.708333333333332
1	[0014.4F01.0000.192F]	16.958333333333332

The temperature in the cluster in this table was calculated with the averaging function.

6.3 Investigation of the cross-layer integration of protocols

The next set of experiments is intended to measure the average energy consumption of sensor nodes in the network under various network loads with and without using the PEAS protocol. Then, by optimizing the parameters of the PEAS protocol it will be possible to stay in balance between the power consumption and number of sensors involved in work per area for QA.

Two transport layer protocols employed in IEDADC have different levels of reliability. Radiostream sends an end-to-end ACK in multihop transmissions for each received packet, while Radiogram does not. The radio used by the SPOT consumes roughly the same amount of power to receive as it does to transmit, and as we know, this is the most expensive process for sensors in terms of power usage. Acknowledgements are also needed to be sent back and forth to confirm packet delivery, leading to higher power usage. On the other hand, the datagram carries such information as the signal strength, data correlation, which makes the datagram size larger than a packet from the data output stream, which doesn't carry this information[5]. Thus, sending a datagram may consume more power than a stream packet. In the following experiments, we want to determine the average power usage of sensors under different network loads while running on different protocols. To see the difference in the

[5] It may seem confusing, because previously it was mentioned that the Radiostream is also capable of sending additional information about the packet such as the signal strength, link quality and so on, but this can be done only if radio input stream is opened. In the implementation of the framework, data input stream is opened instead, because additional information is not needed when using Radiostream connection for reasons explained earlier in this paper.

protocols, the sensors will be placed so that only one sensor is in the range of the base station while others are placed out of range, so the sensors would have to use multihop transmission to send their data and thus use ACKs for packet delivery in case using Radiostream protocol.

The energy consumption in this experiment is evaluated according to two parameters: the battery level of a sensor node and the current being drawn from the battery of a node. The first parameter is measured as the difference in the battery levels at the beginning of the experiment and at the end. The battery level unit in the framework is represented as the percentage of the battery volume. The second parameter is obtained from periodic measurement of the current being drawn from the battery of each sensor. Both of those parameters are averaged to see the protocol behavior in general for all of the sensor nodes. Although there are two parameters measured in this experiment, the main emphasis is put on the battery level difference, because this parameter is more objective than the second one. The current being drawn from the battery is measured only when the sensor is working, however when all threads in the sensor are idle (including the thread that measures the current), it may enter power-saving mode called "deep sleep". During deep-sleep, the sensor node switches off its primary power supply and only maintains power to the low-level firmware and RAM [21] so the energy usage is greatly reduced. This fact is not put into consideration when measuring the second parameter. That is why the results of these measurements do not reflect power saving mode and, in case running the PEAS protocol, there would be roughly the same current in the working sensor as in the sleeping one. Nevertheless, this parameter helps us to see possible deviations in the current flow under different protocols and network loads.

In the first experiment we measure the average power usage of nodes in a WSN. In order to achieve this, sensor nodes will send measurement data to the base station after certain periods of time: short period (every 1.2 seconds, sampling rate 200 milliseconds), average period (every twenty four seconds, sampling rate 4000 milliseconds) and long period (every one minute, sampling rate 10000 milliseconds). As previously mentioned, in a Radiogram protocol, a datagram can fit six measurements and only then it is sent to the base station, however, when using Radiostream the measurement is sent straight away. In order to put these two protocols in equal conditions, the process of sending data in Radiostream protocol is modified in such a way that the data will be sent after six samples have been collected. Then the host application will take the average of those values as it does when using Radiogram protocol.

With the sampling rate set to 200 milliseconds the data arrives every 1.2 seconds to the base station. The following plot shows the average value of current being drawn from the batteries, when using Radiogram and Radiostream protocols respectively (Fig. 6.5.x). The average battery discharge level is shown in the following table:

Table 1. Battery discharge level of sensor nodes using transport layer protocols only.

Sampling rate (msec.)\Protocol	Radiostream	Radiogram
200	8.5%	8.75%
4000	5.25%	4.5%
10000	7.75%	8.25%

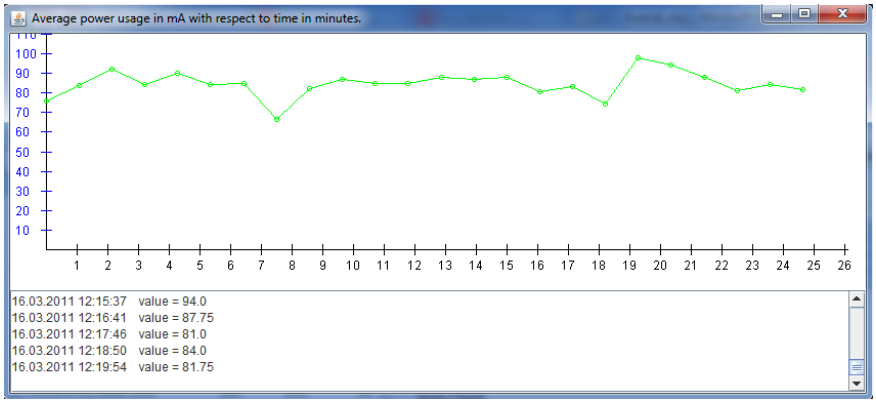


Fig. 6.5.1. *The average current drawn from the battery, when using Radiogram protocol, with the sampling rate set to 200 msec. Average current drawn from the battery is 84,49 mA, average battery discharge is 8,75%.*

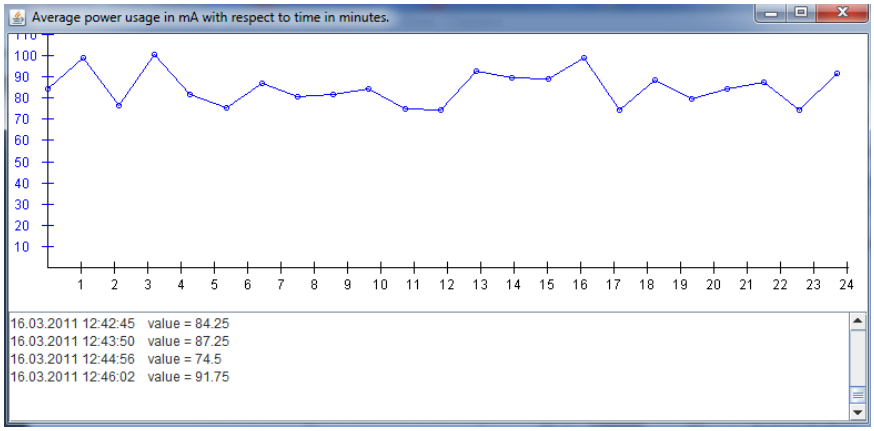


Fig. 6.5.2. *The average current drawn from the battery, when using Radiostream protocol, with the sampling rate set to 200 msec. Average current drawn from the battery is 84,75 mA, average battery discharge is 8,5%.*

Given the sampling rate of the sensor set to four seconds, implying the data would arrive every twenty four seconds for both of the protocols, we get two plots of average value of the current being drawn from sensors' batteries shown in Fig. 6.6.x.

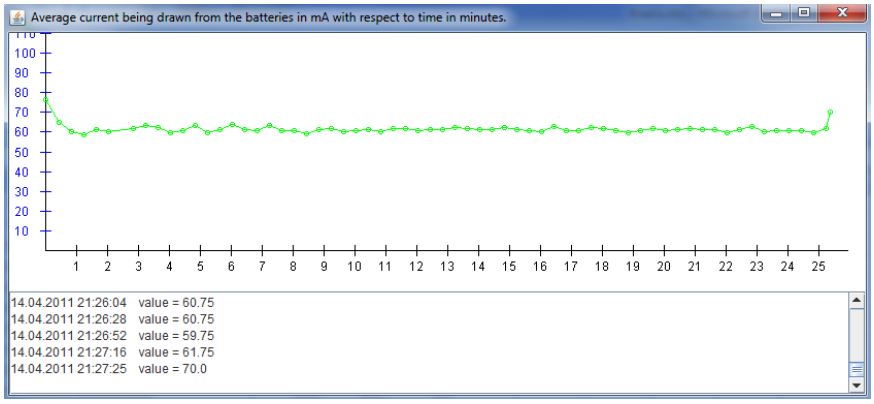


Fig. 6.6.1. *The average current drawn from the battery, when using Radiogram protocol, with the sampling rate set to 4000 msec. Average current drawn from the battery is 61.67 mA, average battery discharge is 4.5%.*

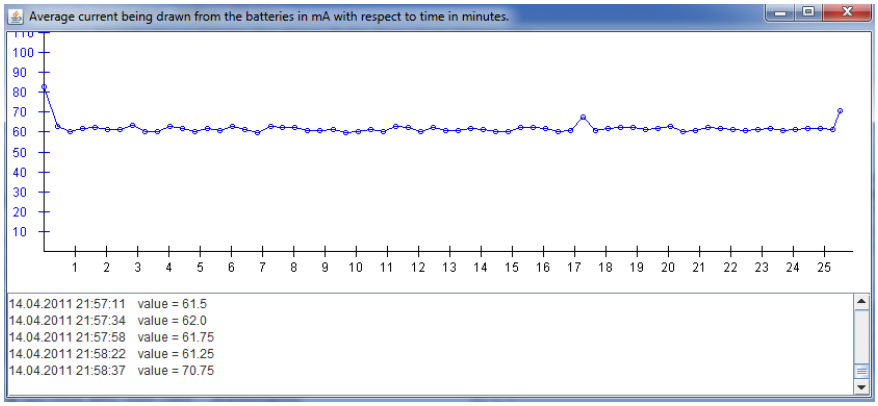


Fig. 6.6.2. *The average current drawn from the battery, when using Radiostream protocol, with the sampling rate set to 4000 msec. Average current drawn from the battery is 62.03 mA, average battery discharge is 5.25%.*

The results of power usage with the sampling rate set to 10000 msec., implying that the data is sent every one minute, is shown in the following two charts:

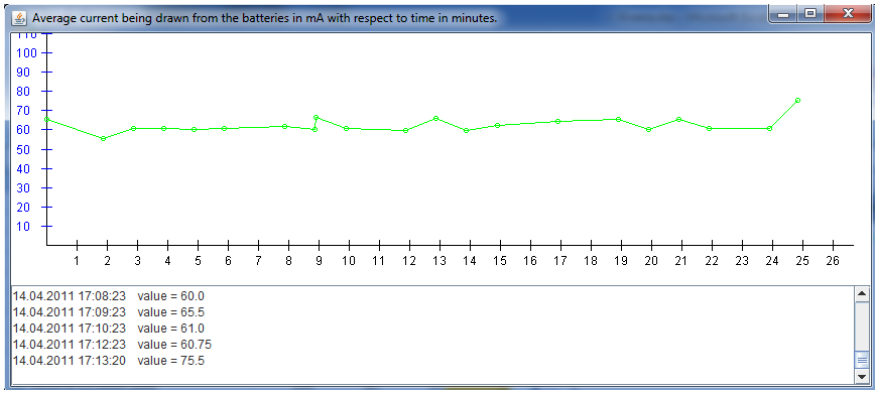


Fig. 6.7.1. *The average current drawn from the battery, when using Radiogram protocol, with the sampling rate set to 10000 msec. Average current drawn from the battery is 61.81 mA, average battery discharge is 8.25%.*

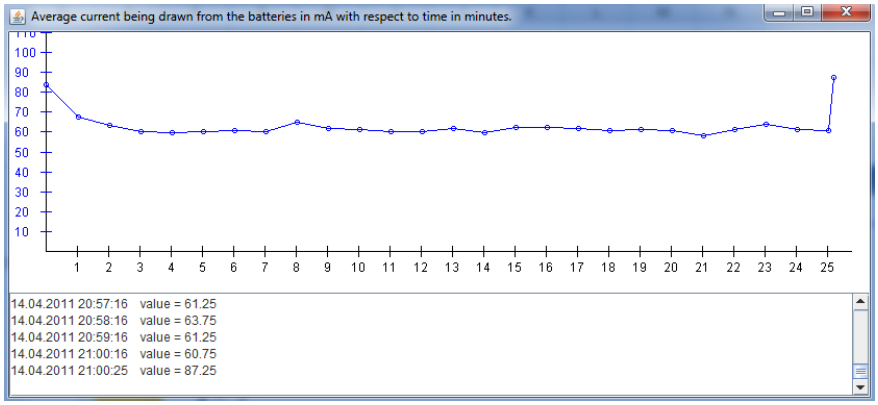


Fig. 6.7.2. *The average current drawn from the battery, when using Radiostream protocol, with the sampling rate set to 10000 msec. Average current drawn from the battery is 61,75 mA, average battery discharge is 7.75%.*

With the sampling rate set to 200 and 10000 msec. the sensors did not enter the deep sleep mode in either of the two protocols. The reason for that behavior is that certain conditions need to be satisfied for a sensor to deep sleep. First condition is that all threads must be on a timed wait or blocked on synchronization primitives, with at least one thread on a timed wait, with the waiting interval at least as long as the minimum deep sleep time, which is equal to 3515 msec. As we can see, setting the sampling rate to 200 msec. means that this period is shorter than the above requirement, therefore the sensors will not deep sleep. Setting the sampling interval to 10000 msec. satisfies this condition, but there is another condition that needs to be addressed. It states that a sensor node's device driver may veto deep sleep if switching off its associated hardware would cause problems [13]. Satisfying this condition is

transparent to a user, i.e. there is no control from a user perspective to make it satisfy programmatically.

By looking at the results of the experiment, we can observe a small difference or none at all in power consumption when the radio is always on, i.e. the nodes do not deep sleep. The average current being drawn from the batteries is around 85 and 95 mA for sampling rates 200 and 10000 msec. respectively. The difference in discharges of the batteries in these two sampling rates is also small. We can see significant difference in energy consumption with the sampling rate set to 4000. In this mode, nodes did enter deep sleep mode which allowed them to save some energy; the battery discharge level decreased by from 3 to 4% depending on the protocol used. Also, we can observe that in this mode Radiogram outperforms the Radiostream protocol in energy consumption. This is due to the fact that Radiostream protocol multihop transmissions have longer interval since they need to wait for the end-to-end ACKs, resulting in higher energy consumption in comparison with the Radiogram protocol[6]. This leads us to a conclusion that we can only see the difference in energy consumption between these two protocols only when a packet is transported over multiple hops and sensor nodes deep sleep in between these transmissions.

Next, the sensors will be using PEAS protocol on top of their forwarding protocols in order to extend network lifetime by reducing battery usage. The experiment is similar to the previous one, but now only working sensor sends the data, while those which are in the sleep mode are idle (i.e. they don't sense the area, thus they don't send anything, but their antenna is still operating to receive any commands from the base station). All sensors are in the direct range of the base station. Multihoping is not possible in this experiment since there is only one working sensor. All sensor nodes are deployed close to each other so that the RSSI is high for that group. This enables the sensors running PEAS protocol to select one working sensor and put the rest of the sensors to sleep, i.e. the k-coverage parameter of this protocol is set to one in this experiment.

[6] The question of difference in energy consumption between the two protocols, Radiogram and Radiostream, have been discussed by me on forum https://www.sunspotworld.com/forums/viewtopic.php?f=38&t=3580, where one of the SPOT software researchers explained the fact that this difference can be seen when using multihop transmissions in WSN, with sensors deep sleeping between transmissions.

The following table shows us the results in battery discharge level (%).

Table 2. Battery discharge level of nodes under PEAS protocol.

Sampling rate\Protocol	Radiostream		Radiogram	
	Working	Sleeping	Working	Sleeping
200	11	3.67	10	2.33
4000	10	1.33	11	2.67
10000	12	1.67	12	2.67

One observation needs to be made regarding the battery discharge levels of a sleeping node. Even though the battery discharge levels are different for various sampling rates, normally they should not depend on this parameter, because sleeping sensors are not sensing the area. The reason behind different discharge values lies in exponentially distributed sleeping time generated by a sensor. Higher battery discharge level in the nodes implies that the generated sleeping time duration was short, so the sensors entered the probing state frequently, resulting in higher battery discharge. To lower the battery discharge of a sleeping sensor, the sleeping time duration parameter of the PEAS protocol could be set to a higher value. This will cause the sensors to generate longer sleeping time durations. One should be careful setting this parameter too high, because unexpected node failures can be left unnoticed for a longer time. The results, nevertheless, of this experiment will be used for further calculations of the network lifetime.

The plots of current being drawn from batteries under different sampling rates are provided below.

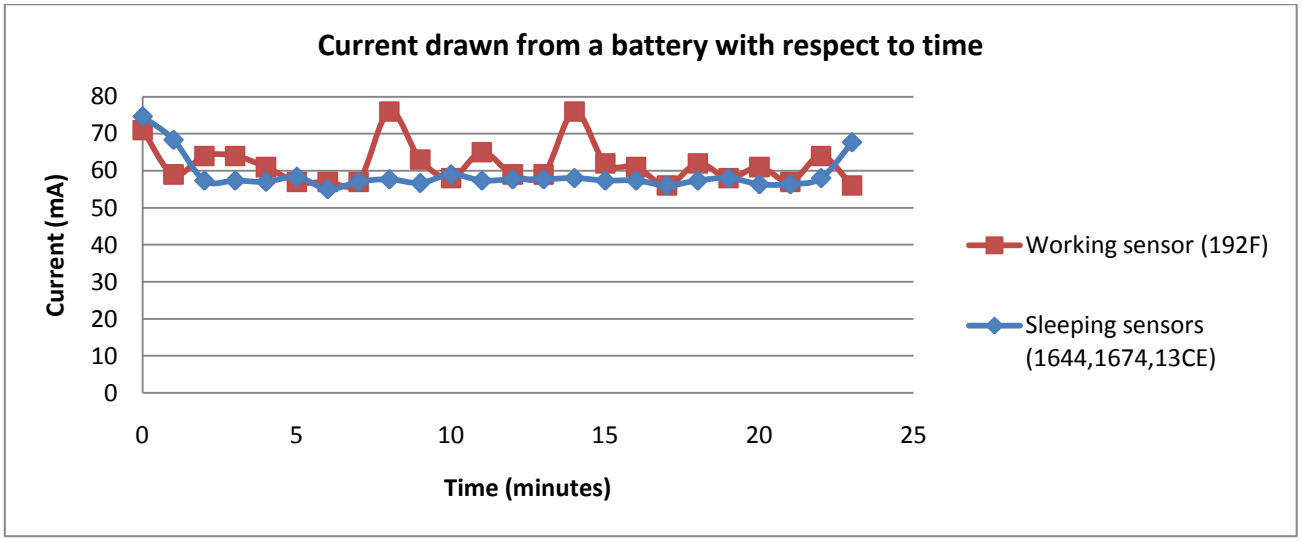


Fig. 6.8.1. *The power drain from using Radiogram and PEAS protocols together, with the sampling rate set to 200 msec. Average current drawn from a battery of a sleeping sensor is* ***58.8*** *mA; of a working sensor* ***61.8*** *mA.*

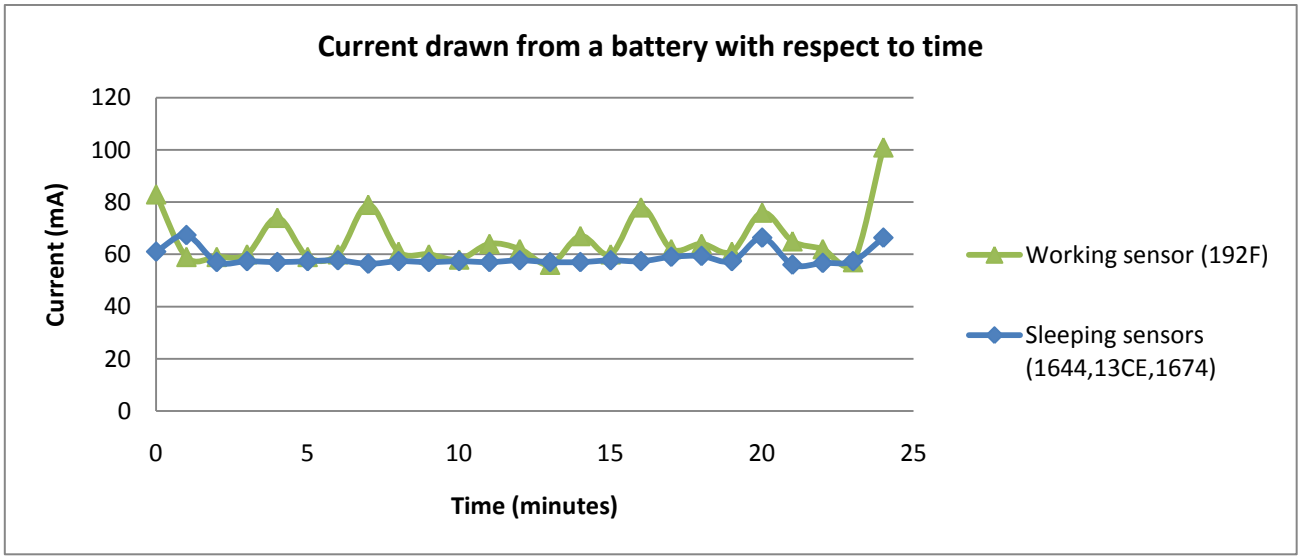


Fig. 6.8.2. *The power drain from using Radiostream and PEAS protocols together, with the sampling rate set to 200 msec. Average current drawn from a battery of a sleeping sensor is* ***58.6*** *mA; of a working sensor* ***65.9*** *mA.*

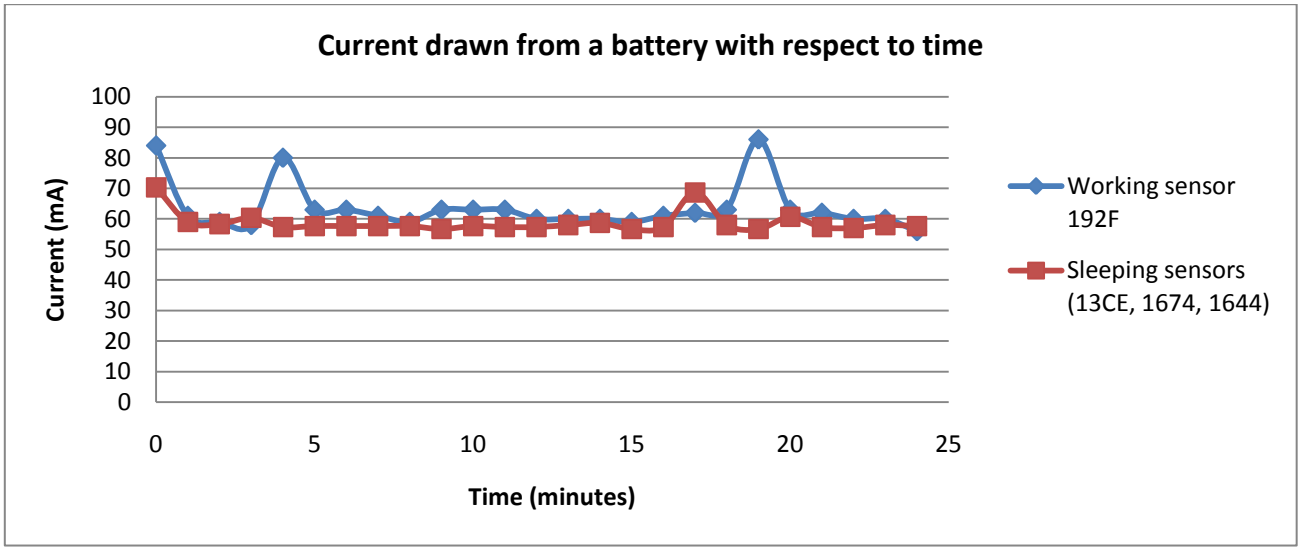


Fig. 6.8.3. *The power drain from using Radiogram and PEAS protocols together, with the sampling rate set to 4000 msec. Average current drawn from a battery of a sleeping sensor is **59** mA; of a working sensor **64** mA.*

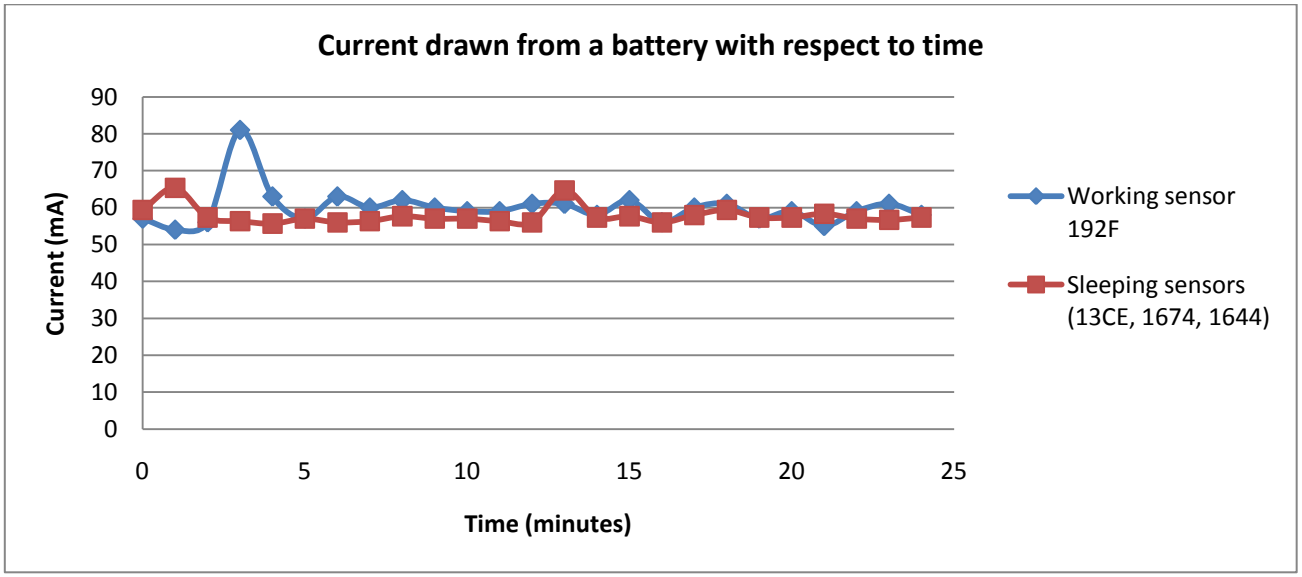


Fig. 6.8.4. *The power drain from using Radiostream and PEAS protocols together, with the sampling rate set to 4000 msec. Average current drawn from a battery of a sleeping sensor is **58** mA; of a working sensor **60** mA.*

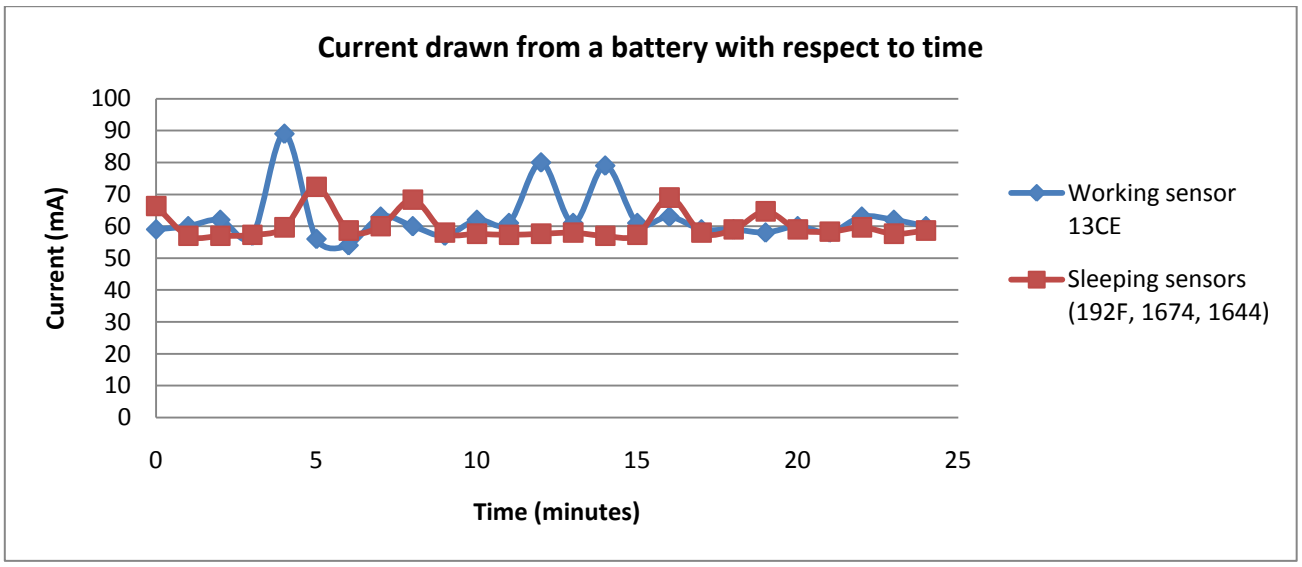


Fig. 6.8.5. *The power drain from using Radiogram and PEAS protocols together, with the sampling rate set to 10000 msec. Average current drawn from a battery of a sleeping sensor is **60** mA; of a working sensor **63** mA.*

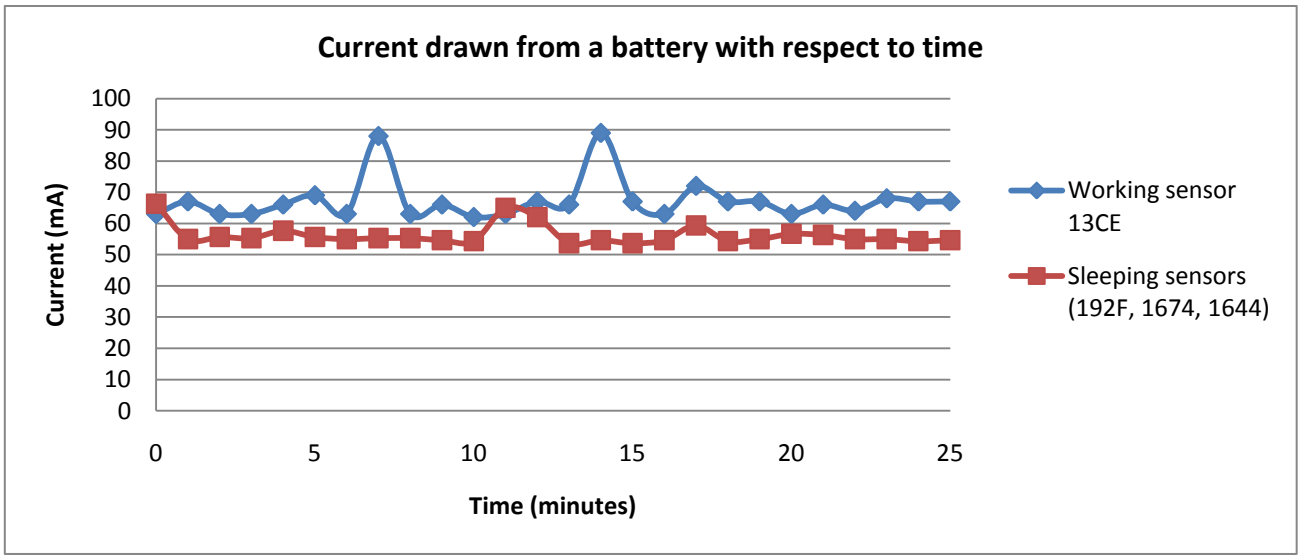


Fig. 6.8.6. *The power drain from using Radiostream and PEAS protocols together, with the sampling rate set to 10000 msec. Average current drawn from a battery of a sleeping sensor is **59** mA; of a working sensor **67** mA.*

As we can see from the above plots, the current drawn from a working sensor's battery is slightly higher than the average current drawn from the sleeping nodes' batteries. As it was previously mentioned, this parameter does not cover the situation when sensor nodes enter the power saving mode. Because of this, the difference in battery discharge levels between working and sleeping sensors is very noticeable.

By comparing the battery discharge levels, working with and without PEAS protocol, we can now determine if adding an overhead to a working sensor could prolong the network lifetime.

If we look at the first result of the table 1 (sampling rate: 200 msec.; protocol: Radiostream) we can see that if on average, the sensors discharge 8.5% in 25 minutes, this indicates that the sensors will work for about 280 minutes until their batteries are depleted.

Now, using the same parameters while running the PEAS protocol, we observe from the table 2 that the working sensor discharges by 11% in 25 minutes, which leads to $\frac{100}{11} \times 25 \cong 227$ minutes of working time. By that time, sleeping sensors will discharge by $\frac{100}{11} \times 3.67 \cong 33\%$, so the next ($2^{nd}$) working sensor would have about $100 - 33 = 67\%$ of battery level, which is enough for $\frac{67}{11} \times 25 = 152$ minutes. Again the rest of the sleeping sensors will discharge by $\frac{67}{11} \times 3.67 \cong 22\%$, so the 3^{rd} sensor would start working with $67 - 22 = 45\%$ of its battery level, which is enough for $\frac{45}{11} \times 25 = 100$ minutes. The 4^{th} sensor would start working with 30% of its battery capacity, lasting for about 63 minutes. And so on, until all sensors in this area run out of their batteries. If there were more sensor nodes available, then they would continue to operate by replacing each other. According to the above calculations, the last sensor which would enter working state would be the 9^{th} sensor. It would enter the working state with only 2% of its battery level, which would allow it to work for about 4 minutes. Obviously, the rest of the sleeping nodes which would also have 2% of their battery level, would work after the 9^{th} sensor for a negligible amount of time. So we can state that after the 9^{th} sensor runs out of battery life, there will be no sensing of this area. Summing up all the working periods of the working sensors in total the area can be sensed for 662 minutes.

Using the same calculations as above, the following comparative table will demonstrate the approximate network lifetime (in minutes) under different protocols and sampling rates. The "Number of the working sensors" column in this table shows the number of the sensors that will manage to enter the working state for a sufficient amount of time (> 4 minutes) before all the sensors in this area run out of their batteries.

Table 3. Network lifetime under various protocols.

Sampling rate\Protocol	Radiostream	Radiogram	Radiostream + PEAS		Radiogram + PEAS	
	Lifetime	Lifetime	Lifetime	# of working sensors	Lifetime	# of working sensors
200	280	285	662	9	1053	15
4000	476	555	1854	30	922	15
10000	323	303	1471	27	919	16

As we can observe from the table 3, even with the sensors deep sleeping between transmissions as in the experiment with the sampling rate set to 4000, the network lifetime is shorter than the worst result from using the PEAS protocol on top of the forwarding protocol. In general, the network lifetime is extended two to four times.

There could be a situation in a highly dense network when not all the sensor nodes could manage to enter the working state and would run out of their battery, staying in a sleeping state. This can happen when the number of nodes in the area exceeds the number of nodes that actually entered the working state (the value from the table above). To avoid this, it is possible to set the parameter of k-coverage to a value higher than one. For example, for the case when Radiostream and PEAS protocols are running together, with the sampling rate set to 200 msec., we know that only nine sensor nodes will enter the working state. However, when there are eighteen nodes in the area, resulting in nine of the sensors never entering the working state. So by setting the k-coverage parameter of the PEAS protocol to two, two sensors will work together without trading off the network lifetime, thus improving the QoD.

On the other hand, when the network is rarefied, and the number of sensors per area is only one, then using the PEAS protocol is worthless, because an extra overhead added to a sensor will discharge its battery faster.

Putting these observations together the table 4 will summarize all the features supported by the protocols used in this work.

Table 4. Supported features of the protocols in IEDADC.

Feature \ Protocol	Radiogram	Radiostream	PEAS
Protocol layer	Transport	Transport	Application
Broadcast support	Yes	No	-
Reliable (use of ACKs)	No	Yes	-
Customizable parameters	No	No	Yes
QA support	Yes	No	-

The following instructions on the selection of protocols and parameters will help a user adapt the framework for application requirements. If an application requires the highest accuracy of data, trading-off the network lifetime, the Radiogram protocol without enabling the PEAS service should be selected. If a reliable connection between the base station and sensor nodes, trading-off the QA, is required, the Radiostream protocol should be selected. If the network lifetime is a priority in addition to the previously mentioned requirements then PEAS protocol should be enabled as well. Note that using Radiogram with PEAS protocol together, in a hope to have the QA support, also requires setting the number of working sensors per area to a value greater than one. If a user is not concerned with the fact that node failures can be left unnoticed for a long period of time and network lifetime is the highest priority, then the sleeping time duration parameter of the PEAS protocol can be set to a higher value than ten seconds. Consequently, the sensors will generate longer sleeping time durations, which reduces the frequency of entering the probing state and thus saving some energy.

The experiments determining the signal strength distribution over the distance demonstrated that the maximum transmission range of a sensor is about 35 meters and the signal strength at that point is around 23 (on a 1 to 121 scale). In order to keep the network connected, a user needs to set the probing range parameter lower than the maximum transmission range. The default value of 50 in the implementation of the protocol corresponds to the distance up to approximately seven meters. Setting this parameter too low would cause the network to be densely connected, trading-off the network lifetime. On the other hand,

increasing the probing range value would result in having less working sensors in the network, with greater number of sleeping sensors around them. The distance between two working sensors would also increase, creating a sparse network.

7. Conclusion. Main research results

This thesis has developed a novel generic design methodology, which allows for both multi-criteria optimization and flexibility.

This methodology includes:

1) The dynamic switching of the transport layer protocols provides improving the QoS through the protocol adaptation to a particular situation or an application.
2) Reducing the number of working sensor nodes in the network increases the network lifetime.
3) Dynamic selection of the degree of coverage and QA mechanisms improve the accuracy of data.
4) QA mechanisms include localization and clustering algorithms as well as functions for data processing such as averaging and voting functions.

The design of the framework architecture has a potential for further improvement, such as addition of new algorithms, which will improve a wider spectrum of criteria.

To verify the hypothesis and prove the design concept, the IEDADC has been implemented. This framework supports dynamic switching of two transport layer protocols: Radiogram and Radiostream. The application layer protocol PEAS reduces the number of working sensors by putting them to sleep mode, where sensors stay idle and do not participate in data collection. The QA mechanism includes the following operations:

1) Localization, i.e. calculating the relative position of a sensor node with respect to other nodes;
2) Clustering, i.e. merging the sensor nodes together into monitoring areas;
3) Data processing, i.e. calculating the temperature value within a cluster according to the implemented aggregation functions – average and vote.

This framework has been employed to conduct an empirical study, which has confirmed the higher level of design flexibility and optimization of the following criteria: energy consumption, QoD and QoS.

The empirical study included:

1) Measuring the average energy consumption of sensor nodes in the network under various network loads with and without using the PEAS protocol;
2) Measuring the accuracy of the localization algorithm;
3) The examination of the clustering algorithm and aggregation functions' operation.

The experiments demonstrated the following results:

1) The network lifetime significantly increased under the operation of the PEAS protocol on top of the transport layer protocol.
2) The accuracy of the localization algorithm is high enough to reflect the real positioning of the sensor nodes. The result of this experiment allowed using the signal strength values between the nodes in the clustering algorithm.
3) The clustering algorithm separates the sensor node into a different cluster in case when the sensor's measurement value is significantly different from other sensors' measurements within a cluster.
4) Averaging and vote functions provide a correct representation of the temperature values within a cluster from to the incoming data.

This work focuses on optimization of multiple criteria, while the framework in [4] puts the emphasis on the network protocol performance only, i.e. average network traffic per node. Also current work investigates the ways to automate some processes in the framework, based on the data incoming from sensor nodes, aiming to reduce the workload from a user. The MAPS performance benchmarking demonstrated that the serialization and transportation of agents is a very time and energy consuming operations [1]. Implementation of the SC methodology requires deployment of a comparatively large number of sensors to maintain functionality of the areas proposed in [5]. Therefore SC methodology is not applicable in a sparsely deployed network or a network with a small number of sensor nodes.

The developed methods and their implementations have proved the advantage of the proposed design against existing tools.

References

[1] Aiello, F., Fortino, G., Gravina, R., & Guerrieri, A. (2011). A java-based agent platform for programming wireless sensor networks. *Computer Journal, 54*(3), 439-454.

[2] Chen, I., Speer, A., & Eltoweissy, M. (2011). Adaptive fault-tolerant QoS control algorithms for maximizing system lifetime of query-based wireless sensor networks. *Dependable and Secure Computing, IEEE Transactions, 8*(2), 161-176.

[3] IEEE Computer Society. (2006, September). IEEE Standard for Information Technology, Telecommunications and Information Exchange between Systems, Local and Metropolitan Area Networks, Specific requirements. *Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs).* Revision of IEEE std. 802.15.4-2003. Retrieved from http://standards.ieee.org/getieee802/download/802.15.4-2006.pdf

[4] Ivester, M., & Lim, A. (2006). Interactive and extensible framework for execution and monitoring of wireless sensor networks. *First International Conference on Communication System Software and Middleware, Comsware 2006,* 1-10.

[5] Liu, W., Fang, Y., Zhang, Y., & Lou, W. (2006). A robust and energy-efficient data dissemination framework for wireless sensor networks. *Wireless Networks, 12*(4), 465-479.

[6] Mahmood, M. (2010). *Developing an Intelligent Signal Simulation and Novelty Detection Application for Sensor Signals using Neural Networks* (Master's Project).

[7] Martínez J., Garcí A., Corredor I., López L., Hernández V., & Dasilva A. (2007). QoS in wireless sensor networks: survey and approach. *Proceedings of the 2007 Euro American conference on Telematics and information systems* (EATIS '07). ACM, New York, NY, USA, article 20, 8 pages. doi: 10.1145/1352694.1352715

[8] Meike, R. (2001, March). Location, Location, Location (Radio) [Web blog]. Retrieved from http://blogs.sun.com/roger/entry/location_location_location_radio.

[9] Perillo, M., & Heinzelman, W. (2004). Sensor management. In *Wireless sensor networks*, C. S. Raghavendra, Krishna M. Sivalingam, and Taieb Znati (Eds.). Kluwer Academic Publishers, Norwell, MA, USA, 351-372.

[10] Perillo, M., & Heinzelman, W. (2005). Wireless Sensor Network Protocols. To appear in *Fundamental Algorithms and Protocols for Wireless and Mobile Networks,* CRC Hall. Retrieved from http://www.ece.rochester.edu/courses/ECE586/readings/perillo.pdf

[11] Sankarasubramaniam, Y., Akan, O., & Akyildiz, I. (2003). ESRT: Event-to-sink reliable transport in wireless sensor networks. *Proceedings of the International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 177-188.

[12] Shi, W., & Tang, X. (2009). Data Quality Management in Wireless Sensor Networks: Guest Editorial. Inderscience Enterprises. Retrieved from http://www.cs.wayne.edu/~weisong/papers/shi09-data-quality.pdf

[13] Sun Labs. (2010, November). *Sun™ SPOT Programmer's Manual, Release v6.0 (Yellow)*. Retrieved from http://www.sunspotworld.com/docs/Yellow/SunSPOT-Programmers-Manual.pdf

[14] Sun Labs. (2011, March). *Sun SPOT Library APIs, Release v6.0.* Retrieved from https://www.sunspotworld.com/docs/Yellow/javadoc/index.html.

[15] Sun SPOT World. (2011, February). Retrieved from http://www.sunspotworld.com.

[16] Timms G. P., De Souza P. A., Jr., L. Reznik. Automated Data Quality Control in Marine Sensor Networks (current research).

[17] Tiny OS. (2010, December). Retrieved from http://www.tinyos.net.

[18] Wang, S., Shih, K., & Chang, C. (2007). Distributed direction-based localization in wireless sensor networks. *Computer Communications, 30*(6), 1424-1439.

[19] Xia, F. (2008). Qos challenges and opportunities in wireless sensor/actuator networks. *SENSORS, 8*(2), 1099-1110.

[20] Xia, F. (2009). Wireless sensor technologies and applications. *SENSORS, 9*(11), 8824-8830.

[21] Ye, F., Lu, S., Zhong, G., Zhang, L., & Cheng, J. (2003). PEAS: A robust energy conserving protocol for long-lived sensor networks. *Proceedings - International Conference on Distributed Computing Systems,* IEEE Computer Society, Washington, DC, USA, 28-37.

[22] Yick, J., Mukherjee, B., & Ghosal, D. (2008). Wireless sensor network survey. *Computer Networks, 52*(12), 2292-2330.

[23] Yuanli, W., Xianghui, L., & Jianping, Y. (2006). Requirements of quality of service in wireless sensor network. *Proceedings of the International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies (ICN/ICONS/MCL'06).* IEEE Computer Society, Washington, DC, USA, 116-. doi: 10.1109/ICNICONSMCL.2006.185

[24] ZigBee Alliance. (2010, December). Retrieved from http://www.zigbee.org.